emnlp 2017

Copenhagen, Denmark
September 7-11, 2017

Conference on Empirical Methods in Natural Language Processing

Conference Proceedings

www.emnlp2017.net

The EMNLP organizers gratefully acknowledge support from the following sponsors:

Platinum

Google

facebook

Bloomberg

Baidu 百度
百度一下 你就知道

Siteimprove

amazon

Gold

IBM Research

Microsoft

ebay

zalando

KPMG

SAP

textkernel
Machine Intelligence for People and Jobs

Maluuba
A Microsoft company

RECRUIT
Institute of Technology

Deloitte.

Silver

NUANCE

ORACLE

搜狗搜索

HUAWEI

duolingo

CVTE Dream-Future 视源股份

UNSILO

TRUSTPILOT

Wizkids

NEXT AI
a NEXT Canada program

Bronze

grammarly

Yandex

Snap Inc.

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-945626-83-8

Preface by the General Chair

Thank you so much for joining us in Copenhagen! Welcome to a cosmopolitan city of fantastic restaurants, lovely seascapes, rich history, and lots and lots of cyclists!

We have an exciting program lined up for you, with three Invited talks, fifteen workshops, seven tutorials, nine TACL presentations, 322 reviewed papers presented as both oral talks and posters, and twenty-one demos. I am especially grateful to our Program Chairs, Rebecca Hwa and Sebastian Riedel, who did a fantastic job managing a backbreaking 1,500 paper submissions (1466 reviewed papers). This involved 51 Area chairs and 980 reviewers. We tried some new things this year (never conducive to a smooth process) including a more careful handling of the COIs that result from Area Chair submissions, and the addition of a meta-review step to encourage more thoughtful reviewing. We are soliciting feedback on the meta-review process, from both reviewers and authors. Despite the additional time involvement, many of the Area Chairs embraced this new approach, and would like to repeat it. However, there are clearly a few dissenters, since Rebecca and Sebastian ended up writing around 200 meta-reviews themselves at the last minute! We are also trying to raise the visibility and status of the poster sessions by integrating them as parallel sessions alongside oral talks, with poster session chairs. This is in response to the survey results from EMNLP 2015 that indicated a decided preference for smaller, more frequent poster sessions during the day rather than evening mega-sessions. Finally, Rebecca and Sebastian are bringing you three outstanding invited speakers, Dan Jurafsky, Sharon Goldwater, and Nando de Freitas. No program chairs ever worked harder to bring you a superb set of presentations in an attendee friendly setting.

I am also very grateful to Victoria Fossum and Karl Moritz Hermann, our Workshop Chairs, who put together a terrific slate of fifteen workshops, and paid meticulous attention to ensuring that each workshop could hold exactly the poster sessions, invited talks and special events that it required. Our tutorial chairs, Alexandra Birch and Nathan Schneider, also outdid themselves, providing especially tempting tutorial offerings. Matt Post deserves to be singled out, for being an Advisor to our conscientious and successful Handbook Chair, Joachim Bingel, as well as becoming a welcome last minute addition to our excellent team of Demo Chairs, Lucia Specia and Michael Paul. Thanks are due to our Website Chair, Anders Johannsen, who responded promptly and deftly to all of our requests, and to our Student Volunteer and Student Sponsorship Chairs, Zeljko Agic and Yonatan Bisk, who brought you the helpful and energetic volunteers who keep things running smoothly.

Last but not least, many thanks to your hosts, our Local Arrangements Chairs, Dirk Hovy and Anders Søgaard and their team. Their concern has been increasing the enjoyment of your experience, and to that end they proposed a stunning venue, put together an amazing reception and Social Event, chose your conference bags, issued all the invitation letters for visas, helped create all the signs, etc., etc., etc. Dan Hardt, our Sponsorship chair, working with Anders and Dirk, raised an unusual amount of local sponsorships, all to defray the cost of the Social Event.

As always, we are extremely indebted to our generous sponsors. Our platinum sponsors are Google, Amazon, Baidu, Apple, Facebook, Bloomberg and Siteimprove. Gold sponsors include IBM Research, Microsoft, eBay, SAP, Textkernel, Maluuba, Zalando, Recruit Institute of Technology and Deloitte. Silver sponsors are Nuance, Oracle, Sogou, Huawei, Duolingo, CVTE, Unsilo and Wizkids. Snap Inc., Grammarly and Yandex are our Bronze sponsors.

Finally, many, many thanks to our Area Chairs, our reviewers, and our authors, whose outstanding research is being showcased here for your delectation. *Nyd det mens det varer!*

Best Regards,
Martha Palmer
EMNLP 2017 General Chair

Preface by the Program Committee Co-Chairs

Welcome to the 2017 Conference on Empirical Methods in Natural Language Processing! This is an exciting year; we have received a new record-high in the number of submissions: 1,509 papers. After discounting early withdraws, duplicates, and other invalid submissions, we sent out 1,418 submissions (836 long papers, 582 short papers) to be reviewed by the program committee. Ultimately, 216 long papers (25.8% acceptance rate) and 107 short papers (18.4% acceptance rate) have been accepted for presentation, making a total of 323 papers and an overall acceptance rate of 22.8%.

This year's technical program consists of three invited talks and 113 oral presentations and 219 poster presentations for the 323 long and short accepted papers as well as nine papers accepted to the Transactions of the Association for Computational Linguistics. To accommodate all the presentations in a compressed timeframe, we opted to have plenary sessions for the invited talks and the winners of the Best Paper Awards, while allotting three parallel oral sessions and thematically related poster sessions for all other presentations. We chose to have concurrent poster and oral sessions for several reasons. First, this is the preferred model of the majority (51.6%) of participants who filled out the EMNLP 2015 post-conference survey. Second, this allows us to spread out the poster presentations across three days in smaller thematically related clusters. Finally, this maximises the number of acceptances for the high quality submissions we received; by having more poster sessions, we are able to maintain the acceptance rates at the previous year's level despite an increase in submissions by 40%.

It would not have been possible to properly handle such a large number of submissions without the generous voluntary help from all the members of the program committee, which consists of 980 reviewers overseen by 51 area chairs. We continued last year's experiment of defining twelve relatively broad topic areas and assigning multiple area chairs to facilitate consistent ranking of larger sets of papers. Most technical program decisions, from the selection of papers to the modes of presentation to the choice of outstanding papers, are primarily made in a bottom-up fashion: reviewers assessed and scored papers, made recommendations for oral vs poster decisions, and marked papers suitable for best paper awards; area chairs ensured the quality of assessments, encouraged discussions and assembled opinions into their own recommendations; finally, we construct the technical program, considering the recommendations from the area chairs while taking into account venue constraints and balance across areas. A new experimental feature of this year's EMNLP reviewing process is the “meta review,” in which the area chairs briefly summarize the major discussions between the reviewers to give authors a more transparent view of the process.

Per EMNLP tradition, awards are given to outstanding papers in three categories: Best Long Paper, Best Short Paper, and Best Resource Paper. The selection process is bottom-up: based on the reviewers and area chairs' recommendations, we nominated four papers for each category; we invited expert members to form a Best Papers committee for each category; each committee reviews the candidates and select the winners. The awarded papers will be presented at a special plenary session on the last day of the conference.

We are extremely grateful that three amazing speakers have agreed to give invited talks at EMNLP. Nando de Freitas (Google Deepmind) will discuss simulated physical environments, and whether language would benefit from the development of such environments, and could contribute toward improving such environments and agents within them. Sharon Goldwater (University of Edinburgh) will describe work on developing unsupervised speech technology for those of the world's 7,000 or so languages not spoken in large rich countries. Dan Jurafsky (Stanford University) will talk about processing the language of policing to automatically measure linguistic aspects of the interaction from discourse factors like conversational structure to social factors like respect.

The conference would not have been possible without the support of various people inside and outside of the committee. In particular, we would like to thank:

- Martha Palmer, whose encouragement and advice as the general chair has been invaluable every step of the way;

- Chris Callison-Burch, who has given us excellent advice and support in his capacity as the SIGDAT Secretary;
- Priscilla Rasmussen, who always has the right answers;
- Xavier Carreras and Kevin Duh, who generously shared their experiences as the chairs of EMNLP 2016;
- Anders Johannsen, who is lightning fast with website updates;
- Our 51 area chairs: David Bamman, Mohit Bansal, Roberto Basili, Chris Biemann, Jordan Boyd-Graber, Marine Carpuat, Joyce Chai, David Chiang, Jinho Choi, Jennifer Chu-Carroll, Trevor Cohn, Cristian Danescu-Niculescu-Mizil, Dipanjan Das, Hal Daume, Mona Diab, Mark Dredze, Jacob Eisenstein, Sanja Fidler, Alona Fyshe, Dan Gildea, Ed Grefenstette, Hannaneh Hajishirzi, Julia Hockenmaier, Kentaro Inui, Jing Jiang, Philipp Koehn, Mamoru Komachi, Anna Korhonen, Tom Kwiatkowski, Gina Levow, Bing Liu, Nitin Madnani, Mausam, Rada Mihalcea, Marie-Francine Moens, Saif M. Mohammad, Mari Ostendorf, Sameer Pradhan, Alexander Rush, Anoop Sarkar, William Schuler, Hinrich Schütze, Sameer Singh, Thamar Solorio, Vivek Srikumar, Amanda Stent, Tomek Strzalkowski, Mihai Surdeanu, Andreas Vlachos, Scott Wen-tau Yih, Zhang Yue;
- The best papers award committee members: Chris Brew, Mike Collins, Kevin Duh, Adam Lopez, Ani Nenkova, Bonnie Webber, Luke Zettlemoyer;
- Preethi Raghavan and Siddharth Patwardhan, the publications co-chairs and Joachim Bingel, the conference handbook chair;
- Dirk Hovy and Anders Søgaard, the local arrangements co-chairs;
- Rich Gerber and Paolo Gai at SoftConf.

Finally, we'd like to thank SIGDAT for the opportunity to serve as Program Co-Chairs of EMNLP 2017. It is an honor and a rewarding learning experience. We hope you will be as inspired by the technical program as we are.

EMNLP 2017 Program Co-Chairs
Rebecca Hwa, University of Pittsburg
Sebastian Riedel, University College London

Organizing Committee

General Chair
Martha Palmer, University of Colorado

Local Arrangements Chair
Priscilla Rasmussen, ACL Business Manager

Program Committee Co-chairs
Rebecca Hwa, University of Pittsburgh
Sebastian Riedel, University College London

Local Arrangements Co-chairs
Dirk Hovy, University of Copenhagen
Anders Søgaard, University of Copenhagen

Local Sponsorship Chair
Daniel Hardt, Copenhagen Business School

Workshop Co-chairs
Victoria Fossum, Google
Karl Moritz Hermann, DeepMind

Tutorial Co-chairs
Alexandra Birch, University of Edinburgh
Nathan Schneider, Georgetown University

Demos Co-chairs
Lucia Specia, University of Sheffield
Matt Post, Johns Hopkins University
Michael Paul, University of Colorado

Publications Sr Chair
Siddharth Patwardhan, Apple

Publications Jr Chair
Preethi Raghavan, TJ Watson Lab, IBM

Publicity Chair
Isabelle Augenstein, University of Copenhagen

Web Chair
Anders Johannsen, Apple

Conference Handbook Chair
Joachim Bingel, University of Copenhagen

Conference Handbook Advisor
Matt Post, Johns Hopkins University

Handbook Proofreader
Pontus Stenetorp, University College London

Conference App Chair
Chloé Braud, University of Copenhagen

Student Scholarship Co-chair and Student Volunteer Coordinator
Željko Agić, IT University of Copenhagen
Yonatan Bisk, University of Southern California, ISI

SIGDAT Liason
Chris Callison-Birch, University of Pennsylvania

Program Committee Co-chairs
Rebecca Hwa, University of Pittsburgh
Sebastian Riedel, University College London

Area Chairs
Information Extraction, Information Retrieval, and Question Answering
Mihai Surdeanu, University of Arizona
Jing Jiang, Singapore Management University
Hinrich Schütze, LMU Munich
Sameer Singh, UC Irvine
Scott Wen-tau Yih, MSR
Tomek Strzalkowski, SUNY Albany

Language and Vision
Sanja Fidler, University of Toronto
Hannaneh Hajishirzi, University of Washington

Linguistic Theories and Psycholinguistics
William Schuler, The Ohio State University

Machine Learning
Mohit Bansal, UNC Chapel Hill
Jordan Boyd-Graber, University of Colorado
Trevor Cohn, University of Melbourne
Hal Daumé, University of Maryland
Alona Fyshe, University of Victoria
Anoop Sarkar, Simon Fraser University

Machine Translation and Multilinguality
Marine Carpuat, University of Maryland
David Chiang, University of Notre Dame
Mona Diab, George Washington University
Dan Gildea, University of Rochester
Philipp Koehn, Johns Hopkins University

Segmentation, Tagging, and Parsing
Jinho Choi, Emory University
Julia Hockenmaier, University of Illinois at Urbana-Champaign
Alexander Rush, Harvard University
Zhang Yue, Singapore University of Technology and Design

Semantics
Roberto Basili, University of Roma, Tor Vergata
Chris Biemann, University of Hamburg
Ed Grefenstette, DeepMind
Tom Kwiatkowski, Google
Sameer Pradhan, cemantix.org and Boulder Learning, Inc
Vivek Srikumar, University of Utah

Sentiment Analysis and Opinion Mining
Bing Liu, University of Illinois at Chicago
Rada Mihalcea, University of Michigan
Saif M. Mohammad, National Research Council Canada

Social Media and Computational Social Science
David Bamman, University of California, Berkeley
Cristian Danescu-Niculescu-Mizil, Cornell University
Mark Dredze, Johns Hopkins University
Jacob Eisenstein, Georgia Tech

Spoken Language Processing
Mari Ostendorf, University of Washington

Summarization, Generation, Discourse, Dialogue
Joyce Chai, Michigan State University
Jennifer Chu-Carroll, Elemental Cognition
Kentaro Inui, Tohoku University
Gina Levow, University of Washington
Amanda Stent, Bloomberg LP

Text Mining and NLP Applications
Dipanjan Das, Google
Mamoru Komachi, Tokyo Metropolitan University
Anna Korhonen, University of Cambridge
Nitin Madnani, Educational Testing Service (ETS)
Marie-Francine Moens, KU Leuven
Thamar Solorio, University of Houston
Andreas Vlachos, University of Sheffield

Primary Reviewers

Muhammad Abdul-Mageed; Amjad Abu-Jbara; Heike Adel; Željko Agić; Eneko Agirre; Salah Ait-Mokhtar; Ahmet Aker; Cem Akkaya; Afra Alishahi; Alexandre Allauzen; Tim Althoff; Carlos Alzate; Bharat Ram Ambati; Antonios Anastasopoulos; Daniel Andor; Jacob Andreas; Nicholas Andrews;

Anietie Andy; Gabor Angeli; Marianna Apidianaki; Yuki Arase; Arturo Argueta; Ron Artstein; Yoav Artzi; Ehsaneddin Asgari; Nicholas Asher; Ramón Astudillo; Isabelle Augenstein; Michael Auli; Eleftherios Avramidis; amittai axelrod; Wilker Aziz; Yoram Bachrach; Hessam Bagherinezhad; Collin Baker; Niranjan Balasubramanian; Jason Baldridge; Timothy Baldwin; Tyler Baldwin; Kalika Bali; Miguel Ballesteros; Siddhartha Banerjee; Chitta Baral; Marco Baroni; Alberto Barrón-Cedeño; Pierpaolo Basile; Fernando Batista; Riza Theresa Batista-Navarro; Timo Baumann; Daniel Beck; Beata Beigman Klebanov; Núria Bel; Yonatan Belinkov; Dane Bell; Eric Bell; Kedar Bellare; Islam Beltagy; Anja Belz; Emily M. Bender; Darina Benikova; Luciana Benotti; Jonathan Berant; Taylor Berg-Kirkpatrick; Nicola Bertoldi; Laurent Besacier; Steven Bethard; Chandra Bhagavatula; Suma Bhat; Archna Bhatia; Joachim Bingel; Alexandra Birch; Arianna Bisazza; Yonatan Bisk; Johannes Bjerva; Anders Björkelund; Alan W Black; Eduardo Blanco; John Blitzer; Ivan Bogatyy; Bernd Bohnet; Gemma Boleda; Kalina Bontcheva; Stefan Bordag; Johan Bos; Houda Bouamor; Guillaume Bouchard; Samuel Bowman; Kristy Boyer; S.R.K. Branavan; Chloé Braud; Fabienne Braune; Felipe Bravo-Marquez; Chris Brew; Chris Brockett; Julian Brooke; Caroline Brun; Dominique Brunato; William Bryce; Christian Buck; Paul Buitelaar; Florin Bulgarov; Wray Buntine; Jill Burstein; Bill Byrne; Donna Byron; Elena Cabrio; Aoife Cahill; Nicoletta Calzolari; Jose Camacho-Collados; Erik Cambria; Nicola Cancedda; Marie Candito; Hailong Cao; Cornelia Caragea; Dallas Card; Xavier Carreras; Francisco Casacuberta; Tommaso Caselli; Taylor Cassidy; Vittorio Castelli; Asli Celikyilmaz; Daniel Cer; Özlem Çetinoğlu; Mauro Cettolo; Nathanael Chambers; Yee Seng Chan; Angel Chang; Baobao Chang; Kai-Wei Chang; Snigdha Chaturvedi; Wanxiang Che; Ciprian Chelba; Boxing Chen; Chen Chen; Danqi Chen; Hsin-Hsi Chen; John Chen; Lei Chen; Tao Chen; Wenliang Chen; Yidong Chen; Yun-Nung Chen; Zhiyuan Chen; Jianpeng Cheng; Colin Cherry; Jackie Chi Kit Cheung; Hai Leong Chieu; laura chiticariu; Eunsol Choi; Yejin Choi; Christos Christodoulopoulos; Grzegorz Chrupała; Tagyoung Chung; Philipp Cimiano; Kevin Clark; Stephen Clark; Ann Clifton; Maximin Coavoux; Anne Cocos; Nigel Collier; Michael Collins; Miriam Connor; John Conroy; Matthieu Constant; Danish Contractor; Ryan Cotterell; Benoit Crabbé; Danilo Croce; Montse Cuadros; Iria da Cunha; Andrew Dai; Bhavana Dalvi; Lena Dankin; Amitava Das; Pradeep Dasigi; Munmun De Choudhury; Adrià de Gispert; Daniël de Kok; Eric De La Clergerie; Gerard de Melo; Vera Demberg; Thomas Demeester; Dina Demner-Fushman; Steve DeNeefe; John DeNero; Lingjia Deng; Pascal Denis; Michael Denkowski; Tejaswini Deoskar; Valeria dePaiva; Leon Derczynski; Nina Dethlefs; Ann devitt; Jacob Devlin; Giuseppe Di Fabbrizio; Georgiana Dinu; Ellen Dodge; Jesse Dodge; A. Seza Doğruöz; Li Dong; Qing Dou; Doug Downey; Gabriel Doyle; Eduard Dragut; Mark Dras; Markus Dreyer; Lan Du; Loic Dugast; Kevin Duh; Greg Durrett; Chris Dyer; Marc Dymetman; Richard Eckart de Castilho; Judith Eckle-Kohler; Maud Ehrmann; Andreas Eisele; Jason Eisner; Asif Ekbal; Michael Elhadad; Messina Enza; Gülşen Eryiğit; Hugo Jair Escalante; Keelan Evanini; James Fan; Federico Fancellu; Hao Fang; Stefano Faralli; Richárd Farkas; Noura Farra; Manaal Faruqui; Benoit Favre; Afsaneh Fazly; Marcello Federico; Christian Federmann; Geli Fei; Anna Feldman; Yang Feng; Yansong Feng; Raquel Fernandez; Daniel Fernández-González; Olivier Ferret; Katja Filippova; Andrew Finch; Nicholas FitzGerald; Jeffrey Flanigan; Lucie Flekova; Michael Flor; Radu Florian; Antske Fokkens; José A. R. Fonollosa; Mikel L. Forcada; Eric Fosler-Lussier; George Foster; Jennifer Foster; Stefan L. Frank; Stella Frank; Alexander Fraser; Lea Fremann; Daniel Fried; Mario Fritz; Xiao Fu; Atsushi Fujii; Matthias Gallé; Michel Galley; Michael Gamon; Kuzman Ganchev; Juri Ganitkevitch; Wei Gao; Claire Gardent; Matt Gardner; Ekaterina Garmash; Dan Garrette; Milica Gasic; Tao Ge; Michaela Geierhos; Spandana Gella; Georgi Georgiev; Kallirroi Georgila; Ulrich Germann; George Giannakopoulos; Kevin Gimpel; Filip Ginter; Filip Ginter; Goran Glavaš; Alfio Gliozzo; Koldo Gojenola; Yoav Goldberg; Dan Goldwasser; Juan Carlos Gomez; Carlos Gómez-Rodríguez; Graciela

Gonzalez; Matthew R. Gormley; Cyril Goutte; Amit Goyal; Pawan Goyal; Joao Graca; David Grangier; Spence Green; Eleni Gregoromichelaki; Cyril Grouin; Adam Grycner; Curry Guinn; Hongyu GUO; Jiafeng Guo; Jiang Guo; Weiwei Guo; Nitish Gupta; Pankaj Gupta; Sonal Gupta; Francisco Guzmán; Nizar Habash; Barry Haddow; Gholamreza Haffari; Masato Hagiwara; Udo Hahn; Gus Hahn-Powell; Dilek Hakkani-Tur; David Hall; Keith Hall; Na-Rae Han; Oul Han; Shuguang Han; Xianpei Han; Sanda Harabagiu; Daniel Hardt; Mark Hasegawa-Johnson; Kazuma Hashimoto; Eva Hasler; Helen Hastie; Katsuhiko Hayashi; He He; Hua He; Luheng He; Wenqi He; Xiaodong He; Yifan He; Yulan He; Kenneth Heafield; Michael Heilman; James Henderson; John Henderson; Aron Henriksson; Aurélie Herbelot; Teresa Herrmann; Daniel Hershcovich; Jack Hessel; Ryuichiro Higashinaka; Derrick Higgins; Felix Hill; Erhard Hinrichs; Gerold Hintz; Tsutomu Hirao; Julia Hirschberg; Graeme Hirst; Hieu Hoang; Nathan Hodas; Kristy Hollingshead; Ales Horak; Chiori Hori; Julian Hough; Yifan Hu; Fei Huang; Heyan Huang; Liang Huang; Lifu Huang; Minlie Huang; Ruihong Huang; Shujian Huang; Xuanjing Huang; Zhongqiang Huang; Luwen Huangfu; Samar Husain; Young-Sook Hwang; Gonzalo Iglesias; Ryu Iida; Diana Inkpen; Naoya Inoue; Radu Tudor Ionescu; Ozan Irsoy; Alexei V. Ivanov; Mohit Iyyer; Guillaume Jacquet; Peter Jansen; Yangfeng Ji; Ping Jian; Wenbin Jiang; Anders Johannsen; Michael Johnston; Kristiina Jokinen; Doug Jones; Shafiq Joty; Marcin Junczys-Dowmunt; Dan Jurafsky; David Jurgens; Nobuhiro Kaji; Pallika Kanani; Hiroshi Kanayama; Dimitri Kartsaklis; Arzoo Katiyar; Daisuke Kawahara; Aniruddha Kembhavi; Ruth Kempson; Casey Kennington; Mitesh M. Khapra; Tushar Khot; Bernd Kiefer; Douwe Kiela; Yuta Kikuchi; Jin-Dong Kim; Seokhwan Kim; Tracy Holloway King; Brian Kingsbury; Svetlana Kiritchenko; Chunyu Kit; Roman Klinger; Julien Kloetzer; Kevin Knight; Simon Kocbek; Ekaterina Kochmar; Thomas Kollar; Kazunori Komatani; Rik Koncel-Kedziorski; Lingpeng Kong; Ioannis Konstas; Parisa Kordjamshidi; Alexander Kotov; Zornitsa Kozareva; Mikhail Kozhevnikov; Martin Krallinger; Jayant Krishnamurthy; Canasai Kruengkrai; Lun-Wei Ku; Sandra Kübler; Marco Kuhlmann; Roland Kuhn; Shankar Kumar; Jonathan K. Kummerfeld; Tsung-Ting Kuo; Sadao Kurohashi; Nate Kushman; Polina Kuznetsova; Igor Labutov; Mathias Lambert; Patrik Lambert; Vasileios Lampos; Ian Lane; Ni Lao; Mirella Lapata; Jey Han Lau; Alon Lavie; Joseph Le Roux; John Lee; Kenton Lee; Sungjin Lee; Els Lefever; Tao Lei; Alessandro Lenci; Omer Levy; Roger Levy; Mike Lewis; Fangtao Li; Haibo Li; Jing Li; Junyi Jessy Li; Qi Li; Sujian Li; Wenjie Li; Yanen Li; Zhenghua Li; Maria Liakata; Constantine Lignos; Chuan-Jie Lin; Victoria Lin; Wang Ling; Xiao Ling; Tal Linzen; Pierre Lison; Diane Litman; Changsong Liu; Fei Liu; Fei Liu; Jiangming Liu; Jing Liu; Kang Liu; Qian Liu; Qun Liu; Ting Liu; Yang Liu; Yang Liu; Yiqun Liu; Zhanyi Liu; Zhiyuan Liu; Adam Lopez; Oier Lopez de Lacalle; Adrian Pastor López Monroy; Annie Louis; Ryan Lowe; Bin Lu; Wei Lu; Yi Luan; Marco Lui; Minh-Thang Luong; Franco M. Luque; Anh Tuan Luu; Teresa Lynn; Ji Ma; Xuezhe Ma; Klaus Macherey; Wolfgang Macherey; Pierre Magistry; Suraj Maharjan; Wolfgang Maier; Igor Malioutov; Shervin Malmasi; Suresh Manandhar; Gideon Mann; Christopher D. Manning; Saab Mansour; Amin Mantrach; Diego Marcheggiani; Daniel Marcu; Anna Margolis; Alex Marin; Héctor Martínez Alonso; André F. T. Martins; Bruno Martins; Yuichiroh Matsubayashi; Yuji Matsumoto; Takuya Matsuzaki; Austin Matthews; Arne Mauser; Jonathan May; Diana Maynard; Andrew McCallum; Diana McCarthy; David McClosky; Yashar Mehdad; Yelena Mejova; Pablo Mendes; Helen Meng; Haitao Mi; Yishu Miao; Claudiu Mihăilă; Timothy Miller; Tristan Miller; Bonan Min; Paramita Mirza; Dipendra Misra; Dipendra Misra; Makoto Miwa; Daichi Mochihashi; Ashutosh Modi; Karo Moilanen; Manuel Montes; Christof Monz; Taesun Moon; Raymond Mooney; Roser Morante; Véronique MORICEAU; Alessandro Moschitti; Nasrin Mostafazadeh; Roozbeh Mottaghi; Animesh Mukherjee; Dragos Munteanu; Yugo Murawaki; Smaranda Muresan; Kenton Murray; Maria Nadejde; Ajay Nagesh; Mikio Nakano; Ndapandula Nakashole; Preslav Nakov; Courtney Napoles; Jason Naradowsky; Karthik Narasimhan; Shashi Narayan; Alexis Nasr; Vivi Nastase; Borja

Navarro; Roberto Navigli; Adeline Nazarenko; Mark-Jan Nederhof; Arvind Neelakantan; Sapna Negi; Aida Nematzadeh; Graham Neubig; Hwee Tou Ng; Jun-Ping Ng; Vincent Ng; Axel-Cyrille Ngonga Ngomo; Dong Nguyen; Thien Huu Nguyen; Toan Nguyen; Jian-Yun NIE; Nobal Bikram Niraula; Joakim Nivre; Hiroshi Noji; Joel Nothman; Pierre Nugues; Diarmuid Ó Séaghdha; Brendan O'Connor; Stephan Oepen; Kemal Oflazer; Alice Oh; Jong-Hoon Oh; Naoaki Okazaki; Manabu Okumura; Constantin Orasan; Vicente Ordonez; Myle Ott; Muntsa Padró; Alexis Palmer; Martha Palmer; Shimei Pan; Sinno Jialin Pan; Alexander Panchenko; Bo Pang; Denis Paperno; Aasish Pappu; Natalie Parde; Ankur Parikh; Niki Parmar; Patrick Paroubek; Rebecca J. Passonneau; Panupong Pasupat; John K Pate; Siddharth Patwardhan; Michael J. Paul; Adam Pauls; Umashanthi Pavalanathan; Ellie Pavlick; Adam Pease; Viktor Pekar; Anselmo Peñas; Xiaochang Peng; Sergio Penkale; Gerald Penn; Julien PEREZ; Verónica Pérez-Rosas; Johann Petrak; Slav Petrov; Nghia The Pham; Peter Phandi; Olivier Pietquin; Daniele Pighin; Mohammad Taher Pilehvar; Joelle Pineau; Yuval Pinter; Emily Pitler; Barbara Plank; Julien Plu; Massimo Poesio; Tamara Polajnar; Heather Pon-Barry; Simone Paolo Ponzetto; Ana-Maria Popescu; Andrei Popescu-Belis; Maja Popović; Fred Popowich; Soujanya Poria; Matt Post; Christopher Potts; Vinodkumar Prabhakaran; Daniel Preoţiuc-Pietro; Prokopis Prokopidis; Emily Prud'hommeaux; Jay Pujara; Matthew Purver; Ashequl Qadir; Longhua Qian; Xian Qian; Long Qiu; Minghui Qiu; Ella Rabinovich; Will Radford; Alessandro Raganato; Preethi Raghavan; Altaf Rahman; Nazneen Fatema Rajani; Rafal Rak; Bhuvana Ramabhadran; Carlos Ramisch; Delip Rao; Ari Rappoport; Mohammad Sadegh Rasooli; Sravana Reddy; Ines Rehbein; Roi Reichart; Nils Reimers; David Reitter; Steffen Remus; Xiang Ren; Adithya Renduchintala; Corentin Ribeyre; Giuseppe Riccardi; Matthew Richardson; Martin Riedl; Verena Rieser; Stefan Riezler; German Rigau; Ellen Riloff; Laura Rimell; Fabio Rinaldi; Eric Ringger; Brian Riordan; Alan Ritter; Brian Roark; Kirk Roberts; Molly Roberts; Tim Rocktäschel; Anna Rohrbach; Marcus Rohrbach; Lina M. Rojas Barahona; Stephen Roller; Salvatore Romeo; Paolo Rosso; Mihai Rotaru; Benjamin Roth; Michael Roth; Alla Rozovskaya; Vasile Rus; Attapol Rutherford; Derek Ruths; Fatiha Sadat; Mehrnoosh Sadrzadeh; Markus Saers; Kenji Sagae; Horacio Saggion; Rishiraj Saha Roy; Magnus Sahlgren; Keisuke Sakaguchi; Mohammad Salameh; Shimi Salant; Yunita Sari; Ruhi Sarikaya; Ryohei Sasano; Hassan Sawaf; Asad Sayeed; David Schlangen; Jonathan Schler; Natalie Schluter; Helmut Schmid; Sebastian Schuster; H. Andrew Schwartz; Lane Schwartz; Roy Schwartz; Stephanie Schwartz; Holger Schwenk; Djamé Seddah; Satoshi Sekine; Ethan Selfridge; Jean Senellart; Rico Sennrich; Minjoon Seo; BURR SETTLES; Izhak Shafran; Kashif Shah; Samira Shaikh; Amr Sharaf; Rebecca Sharp; Wade Shen; xiaodong shi; Chaitanya Shivade; Prasha Shrestha; Avirup Sil; Fabrizio Silvestri; Yanchuan Sim; Dan Simonson; Kiril Simov; Steve Skiena; Kevin Small; Sharon Small; Noah A. Smith; Jan Šnajder; Parinaz Sobhani; Richard Socher; Anders Søgaard; Swapna Somasundaran; Hyun-Je Song; Linfeng Song; Sandeep Soni; Alessandro Sordoni; Aitor Soroa; Victor Soto; Rohini Srihari; Somayajulu Sripada; Christian Stab; Sanja Štajner; Miloš Stanojević; Gabriel Stanovsky; Manfred Stede; Mark Steedman; Pontus Stenetorp; Lucas Sterckx; Mark Stevenson; Brandon Stewart; Ian Stewart; Svetlana Stoyanchev; Veselin Stoyanov; Carlo Strapparava; Karl Stratos; Keh-Yih Su; Qi Su; Yu Su; Fabian Suchanek; Alane Suhr; Fei Sun; Huan Sun; Le Sun; Ming Sun; Jun Suzuki; Swabha Swayamdipta; Idan Szpektor; Oscar Täckström; Shabnam Tafreshi; Hiroya Takamura; Partha Talukdar; Chenhao Tan; Hristo Tanev; Duyu Tang; Jian Tang; Xavier Tannier; Makarand Tapaswi; Christoph Teichmann; TestSeb TestRiedel; Joel Tetreault; Simone Teufel; Kapil Thadani; Sam Thomson; Jörg Tiedemann; Christoph Tillmann; Ivan Titov; Takenobu Tokunaga; Gaurav Singh Tomar; Nadi Tomeh; Marc Tomlinson; Sara Tonelli; Kentaro Torisawa; Lamia Tounsi; Isabel Trancoso; Adam Trischler; Yuta Tsuboi; Oren Tsur; Yoshimasa Tsuruoka; Yulia Tsvetkov; Marco Turchi; Ferhan Ture; Kateryna Tymoshenko; Raghavendra Udupa; Stefan Ultes; Lyle Ungar; Shyam Upadhyay; L. Alfonso Urena Lopez; Dmitry Ustalov; Jakob Uszkoreit; Masao Utiyama; Naushad UzZaman; Sowmya Vajjala;

Marco A. Valenzuela-Escárcega; Tim Van de Cruys; Kees van Deemter; Ielka van der Sluis; Benjamin Van Durme; Gertjan van Noord; Marten van Schijndel; David Vandyke; Tony Veale; Eva Maria Vecchi; sriram venkatapathy; Giulia Venturi; Ashish Venugopal; Patrick Verga; Marc Verhagen; Yannick Versley; Guido Vetere; Paul Vicol; David Vilar; Aline Villavicencio; Rob Voigt; Svitlana Volkova; Ivan Vulić; Yogarshi Vyas; V.G.Vinod Vydiswaran; Henning Wachsmuth; Joachim Wagner; Byron Wallace; Matthew Walter; Stephen Wan; Xiaojun Wan; Chuan Wang; Dingquan Wang; Hai Wang; Houfeng WANG; Lu Wang; Qin Iris Wang; Shuai Wang; Shuohang Wang; William Yang Wang; Zhiguo Wang; Zhongqing Wang; Leo Wanner; Taro Watanabe; Nick Webb; Bonnie Webber; Ingmar Weber; Kellie Webster; Julie Weeds; Furu Wei; Gerhard Weikum; David Weir; Ralph Weischedel; David Weiss; Dirk Weissenborn; Robert West; Michael White; Michael Wick; Michael Wiegand; John Wieting; Jason D Williams; Steven Wilson; Guillaume Wisniewski; Michael Wojatzki; Kam-Fai Wong; Hua Wu; Stephen Wu; Joern Wuebker; Rui Xia; Deyi Xiong; Wei Xu; Wenduan Xu; Yadollah Yaghoobzadeh; Adam Yala; Rui Yan; Bishan Yang; Diyi Yang; Yi Yang; Roman Yangarber; Berrin Yanikoglu; Helen Yannakoudakis; tae yano; Mark Yatskar; Seid Muhie Yimam; Wenpeng Yin; Dani Yogatama; Naoki Yoshinaga; Jianfei Yu; Mo Yu; Ning Yu; Zhuoran Yu; François Yvon; Taras Zagibalov; Marcos Zampieri; Fabio Massimo Zanzotto; Sina Zarrieß; Victoria Zayats; Richard Zens; Torsten Zesch; Luke Zettlemoyer; Congle Zhang; Hao Zhang; Justine Zhang; Lei Zhang; Meishan Zhang; Min Zhang; Qi Zhang; Yuan Zhang; Hai Zhao; Jun Zhao; Shiqi Zhao; Wayne Xin Zhao; Bowen Zhou; Muhua Zhu; Xiaodan Zhu; Yukun Zhu; Michael Zock; Chengqing Zong; Ingrid Zukerman; Pierre Zweigenbaum

Invited Speaker: Dan Jurafsky, Stanford University
"Does This Vehicle Belong to You”? Processing the Language of Policing for Improving Police-Community Relations"

Abstract: Police body-cameras have the potential to play an important role in understanding and improving police-community relations. In this talk I describe a series of studies conducted by our large interdisciplinary team at Stanford that use speech and natural language processing on body-camera recordings to model the interactions between police officers and community members in traffic stops. We use text and speech features to automatically measure linguistic aspects of the interaction, from discourse factors like conversational structure to social factors like respect. I describe the differences we find in the language directed toward black versus white community members, and offer suggestions for how these findings can be used to help improve the fraught relations between police officers and the communities they serve.

Bio: Dan Jurafsky is Professor and Chair of Linguistics and Professor of Computer Science, at Stanford University. His research has focused on the extraction of meaning, intention, and affect from text and speech, on the processing of Chinese, and on applying natural language processing to the cognitive and social sciences. Dan’s deep interest in NLP education led him to co-write with Jim Martin the widely-used textbook "Speech and Language Processing” (whose 3rd edition is in (slow) progress) and co-teach with Chris Manning the first massive open online class on natural language processing. Dan was the recipient of the 2002 MacArthur Fellowship and is a 2015 James Beard Award Nominee for his book, "The Language of Food: A Linguist Reads the Menu".

Invited Speaker: Sharon Goldwater, University of Edinburgh
Towards more universal language technology: unsupervised learning from speech

Abstract: Speech and language processing has advanced enormously in the last decade, with successful applications in machine translation, voice-activated search, and even language-enabled personal assistants. Yet these systems typically still rely on learning from very large quantities of human-annotated data. These resource-intensive methods mean that effective technology is available for only a tiny fraction of the world’s 7000 or so languages, mainly those spoken in large rich countries.

This talk describes our recent work on developing unsupervised speech technology, where transcripts and pronunciation dictionaries are not used. The work is inspired by considering both how young infants may begin to acquire the sounds and words of their language, and how we might develop systems to help linguists analyze and document endangered languages. I will first present work on learning from speech audio alone, where the system must learn to segment the speech stream into word tokens and cluster repeated instances of the same word together to learn a lexicon of vocabulary items. The approach combines Bayesian and neural network methods to address learning at the word and sub-word levels.

Bio: Sharon Goldwater is a Reader at the University of Edinburgh’s School of Informatics, where she is a member of the Institute for Language, Cognition and Computation. She received her PhD in 2007 from Brown University and spent two years as a postdoctoral researcher at Stanford University before moving to Edinburgh. Her research interests include unsupervised learning for speech and language processing, computer modelling of language acquisition in children, and computational studies of language use. Dr. Goldwater co-chaired the 2014 Conference of the European Chapter of the Association for Computational Linguistics and is Chair-Elect of EACL. She has served on the editorial boards of the Transactions of the Association for Computational Linguistics, the Computational Linguistics journal, and OPEN MIND: Advances in Cognitive Science (a new open-access journal). In 2016, she received the Roger Needham Award from the British Computer Society, awarded for "distinguished research contribution in computer science by a UK-based researcher who has completed up to 10 years of post-doctoral research."

Invited Speaker: Nando de Freitas, Google Deepmind
Physical simulation, learning and language

Abstract: Simulated physical environments, with common physical laws, objects and agents with bodies, provide us with consistency to facilitate transfer and continual learning. In such environments, research topics such as learning to experiment, learning to learn and emergent communication can be easily explored. Given the relevance of these topics to language, it is natural to ask ourselves whether research in language would benefit from the development of such environments, and whether language can contribute toward improving such environments and agents within them. This talk will provide an overview of some of these environments, discuss learning to learn and its potential relevance to language, and present some deep reinforcement learning agents that capitalize on formal language instructions to develop disentangled interpretable representations that allow them to generalize to a wide variety of zero-shot semantic tasks. The talk will pose more questions than answers in the hope of stimulating discussion.

Bio: I was born in Zimbabwe, with malaria. I was a refugee from the war in Mocambique and thanks to my parents getting in debt to buy me a passport from a corrupt official, I grew up in Portugal without water and electricity, before the EU got there, and without my parents who were busy making money to pay their debt. At 8, I joined my parents in Venezuela and began school in the hood; see City of God. I moved to South Africa after high-school and sold beer illegally in black-townships for a living until 1991. Apartheid was the worst thing I ever experienced. I did my BSc in electrical engineering and MSc in control at the University of the Witwatersrand, where I strived to be the best student to prove to racists that anyone can do it. I did my PhD on Bayesian methods for neural networks at Trinity College, Cambridge University. I did a postdoc in Artificial Intelligence at UC Berkeley. I became a Full Professor at the University of British Columbia, before joining the University of Oxford in 2013. I quit Oxford in 2017 to join DeepMind full-time, where I lead the Machine Learning team. I aim to solve intelligence so that future generations have a better life. I have been a Senior Fellow of the Canadian Institute for Advanced Research for a long time. Some of my recent awards, mostly thanks to my collaborators, include: Best Paper Award at the International Conference on Machine Learning (2016), Best Paper Award at the International Conference on Learning Representations (2016), Winner of round 5 of the Yelp Dataset Challenge (2015), Distinguished Paper Award at the International Joint Conference on Artificial Intelligence (2013), Charles A. McDowell Award for Excellence in Research (2012), and Mathematics of Information Technology and Complex Systems Young Researcher Award (2010).

Table of Contents

Conference Program

Friday, September 8, 2017

19:00–22:00 Welcome Reception

Saturday, September 9, 2017

07:30–17:30 Registration Day 1

08:00–08:30 ***Morning Coffee***

08:30–09:00 Plenary Session. Opening Remarks

08:30–09:00 *Opening Remarks*
General Chair, PC Co-Chairs

09:00–10:00 Plenary Session. Invited Talk by Nando de Freitas

09:00–10:00 *Physical simulation, learning and language*
Nando de Freitas

10:00–10:30 ***Coffee Break***

Saturday, September 9, 2017 (continued)

10:30–12:10 Session 1A: Syntax 1

10:30–10:55 *Monolingual Phrase Alignment on Parse Forests*
Yuki Arase and Jun'ichi Tsujii

10:55–11:20 *Fast(er) Exact Decoding and Global Training for Transition-Based Dependency Parsing via a Minimal Feature Set*
Tianze Shi, Liang Huang and Lillian Lee

11:20–11:45 *Parsing with Traces: An O(nˆ4) Algorithm and a Structural Representation*
Jonathan K. Kummerfeld and Dan Klein

11:45–12:10 *Quasi-Second-Order Parsing for 1-Endpoint-Crossing, Pagenumber-2 Graphs*
Junjie Cao, Sheng Huang, Weiwei Sun and Xiaojun Wan

10:30–12:10 Session 1B: Information Extraction 1

10:30–10:55 *Position-aware Attention and Supervised Data Improve Slot Filling*
Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli and Christopher D. Manning

10:55–11:20 *Heterogeneous Supervision for Relation Extraction: A Representation Learning Approach*
Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji and Jiawei Han

11:20–11:45 *Integrating Order Information and Event Relation for Script Event Prediction*
Zhongqing Wang, Yue Zhang and Ching-Yun Chang

11:45–12:10 *Entity Linking for Queries by Searching Wikipedia Sentences*
Chuanqi Tan, Furu Wei, Pengjie Ren, Weifeng Lv and Ming Zhou

Saturday, September 9, 2017 (continued)

10:30–12:10 Session 1C: Multilingual NLP

10:30–10:55 *Train-O-Matic: Large-Scale Supervised Word Sense Disambiguation in Multiple Languages without Manual Training Data*
Tommaso Pasini and Roberto Navigli

10:55–11:20 *Universal Semantic Parsing*
Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman and Mirella Lapata

11:20–11:45 *Mimicking Word Embeddings using Subword RNNs*
Yuval Pinter, Robert Guthrie and Jacob Eisenstein

11:45–12:10 *Past, Present, Future: A Computational Investigation of the Typology of Tense in 1000 Languages*
Ehsaneddin Asgari and Hinrich Schütze

10:30–12:10 Session 1D: Poster Session: Demo

12:10–13:40 *Lunch*

13:40–15:20 Session 2A: Machine Translation 1

13:40–14:05 *Neural Machine Translation with Source-Side Latent Graph Parsing*
Kazuma Hashimoto and Yoshimasa Tsuruoka

14:05–14:30 *Neural Machine Translation with Word Predictions*
Rongxiang Weng, Shujian Huang, Zaixiang Zheng, XIN-YU DAI and Jiajun CHEN

14:30–14:55 *Towards Decoding as Continuous Optimisation in Neural Machine Translation*
Cong Duy Vu Hoang, Gholamreza Haffari and Trevor Cohn

14:55–15:20 *Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation*
Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes and Jeffrey Dean

Saturday, September 9, 2017 (continued)

13:40–15:20 Session 2B: Language Grounding

13:40–14:05 *Where is Misty? Interpreting Spatial Descriptors by Modeling Regions in Space*
Nikita Kitaev and Dan Klein

14:05–14:30 *Continuous Representation of Location for Geolocation and Lexical Dialectology using Mixture Density Networks*
Afshin Rahimi, Timothy Baldwin and Trevor Cohn

14:30–14:55 *Colors in Context: A Pragmatic Neural Model for Grounded Language Understanding*
Will Monroe, Robert X. D. Hawkins, Noah D. Goodman and Christopher Potts

14:55–15:20 *Obj2Text: Generating Visually Descriptive Language from Object Layouts*
Xuwang Yin and Vicente Ordonez

13:40–15:20 Session 2C: Discourse and Summarization

13:40–14:05 *End-to-end Neural Coreference Resolution*
Kenton Lee, Luheng He, Mike Lewis and Luke Zettlemoyer

14:05–14:30 *Neural Net Models of Open-domain Discourse Coherence*
Jiwei Li and Dan Jurafsky

14:30–14:55 *Affinity-Preserving Random Walk for Multi-Document Summarization*
Kexiang Wang, Tianyu Liu, Zhifang Sui and Baobao Chang

14:55–15:20 *A Mention-Ranking Model for Abstract Anaphora Resolution*
Ana Marasovic, Leo Born, Juri Opitz and Anette Frank

Saturday, September 9, 2017 (continued)

13:40–15:20 Session 2D: Poster Session. Embeddings

Hierarchical Embeddings for Hypernymy Detection and Directionality
Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde and Ngoc Thang Vu

Ngram2vec: Learning Improved Word Representations from Ngram Co-occurrence Statistics
Zhe Zhao, Tao Liu, Shen Li, Bofang Li and Xiaoyong Du

Dict2vec : Learning Word Embeddings using Lexical Dictionaries
Julien Tissier, Christopher Gravier and Amaury Habrard

Learning Chinese Word Representations From Glyphs Of Characters
Tzu-ray Su and Hung-yi Lee

Learning Paraphrastic Sentence Embeddings from Back-Translated Bitext
John Wieting, Jonathan Mallinson and Kevin Gimpel

Joint Embeddings of Chinese Words, Characters, and Fine-grained Subcharacter Components
Jinxing Yu, Xun Jian, Hao Xin and Yangqiu Song

Exploiting Morphological Regularities in Distributional Word Representations
Arihant Gupta, Syed Sarfaraz Akhtar, Avijit Vajpayee, Arjit Srivastava, Madan Gopal Jhanwar and Manish Shrivastava

Exploiting Word Internal Structures for Generic Chinese Sentence Representation
Shaonan Wang, Jiajun Zhang and Chengqing Zong

High-risk learning: acquiring new word vectors from tiny data
Aurélie Herbelot and Marco Baroni

Word Embeddings based on Fixed-Size Ordinally Forgetting Encoding
Joseph Sanu, Mingbin Xu, Hui Jiang and Quan Liu

VecShare: A Framework for Sharing Word Representation Vectors
Jared Fernandez, Zhaocheng Yu and Doug Downey

Word Re-Embedding via Manifold Dimensionality Retention
Souleiman Hasan and Edward Curry

MUSE: Modularizing Unsupervised Sense Embeddings
Guang-He Lee and Yun-Nung Chen

13:40–15:20 Session 2E: Poster Session. Machine Learning 1

Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging
Nils Reimers and Iryna Gurevych

Learning What's Easy: Fully Differentiable Neural Easy-First Taggers
André F. T. Martins and Julia Kreutzer

Incremental Skip-gram Model with Negative Sampling
Nobuhiro Kaji and Hayato Kobayashi

Learning to select data for transfer learning with Bayesian Optimization
Sebastian Ruder and Barbara Plank

Unsupervised Pretraining for Sequence to Sequence Learning
Prajit Ramachandran, Peter Liu and Quoc Le

Efficient Attention using a Fixed-Size Memory Representation
Denny Britz, Melody Guan and Minh-Thang Luong

Rotated Word Vector Representations and their Interpretability
Sungjoon Park, JinYeong Bak and Alice Oh

A causal framework for explaining the predictions of black-box sequence-to-sequence models
David Alvarez-Melis and Tommi Jaakkola

Piecewise Latent Variables for Neural Variational Text Processing
Iulian Vlad Serban, Alexander G. Ororbia, Joelle Pineau and Aaron Courville

Saturday, September 9, 2017 (continued)

Learning the Structure of Variable-Order CRFs: a finite-state perspective
Thomas Lavergne and François Yvon

Sparse Communication for Distributed Gradient Descent
Alham Fikri Aji and Kenneth Heafield

Why ADAGRAD Fails for Online Topic Modeling
You Lu, Jeffrey Lund and Jordan Boyd-Graber

13:40–15:20 Session 2F: Poster Session. Sentiment Analysis 1

Recurrent Attention Network on Memory for Aspect Sentiment Analysis
Peng Chen, Zhongqian Sun, Lidong Bing and Wei Yang

A Cognition Based Attention Model for Sentiment Analysis
Yunfei Long, Lu Qin, Rong Xiang, Minglei Li and Chu-Ren Huang

Author-aware Aspect Topic Sentiment Model to Retrieve Supporting Opinions from Reviews
Lahari Poddar, Wynne Hsu and Mong Li Lee

Magnets for Sarcasm: Making Sarcasm Detection Timely, Contextual and Very Personal
Aniruddha Ghosh and Tony Veale

Identifying Humor in Reviews using Background Text Sources
Alex Morales and Chengxiang Zhai

Sentiment Lexicon Construction with Representation Learning Based on Hierarchical Sentiment Supervision
Leyi Wang and Rui Xia

Towards a Universal Sentiment Classifier in Multiple languages
Kui Xu and Xiaojun Wan

Capturing User and Product Information for Document Level Sentiment Analysis with Deep Memory Network
Zi-Yi Dou

Saturday, September 9, 2017 (continued)

Identifying and Tracking Sentiments and Topics from Social Media Texts during Natural Disasters
Min Yang, Jincheng Mei, Heng Ji, zhao wei, Zhou Zhao and Xiaojun Chen

Refining Word Embeddings for Sentiment Analysis
Liang-Chih Yu, Jin Wang, K. Robert Lai and Xuejie Zhang

A Multilayer Perceptron based Ensemble Technique for Fine-grained Financial Sentiment Analysis
Md Shad Akhtar, Abhishek Kumar, Deepanway Ghosal, Asif Ekbal and Pushpak Bhattacharyya

Sentiment Intensity Ranking among Adjectives Using Sentiment Bearing Word Embeddings
Raksha Sharma, Arpan Somani, Lakshya Kumar and Pushpak Bhattacharyya

Sentiment Lexicon Expansion Based on Neural PU Learning, Double Dictionary Lookup, and Polarity Association
Yasheng Wang, Yang Zhang and Bing Liu

15:20–15:50 ***Coffee Break***

15:50–17:30 **Session 3A: Machine Learning 2**

15:50–16:15 *DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning*
Wenhan Xiong, Thien Hoang and William Yang Wang

16:15–16:40 *Task-Oriented Query Reformulation with Reinforcement Learning*
Rodrigo Nogueira and Kyunghyun Cho

16:40–17:05 *Sentence Simplification with Deep Reinforcement Learning*
Xingxing Zhang and Mirella Lapata

17:05–17:30 *Learning how to Active Learn: A Deep Reinforcement Learning Approach*
Meng Fang, Yuan Li and Trevor Cohn

15:50–17:30	**Session 3B: Generation**
15:50–16:15	*Split and Rephrase* Shashi Narayan, Claire Gardent, Shay B. Cohen and Anastasia Shimorina
16:15–16:40	*Neural Response Generation via GAN with an Approximate Embedding Layer* Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie SUN, Xiaolong Wang, Zhuoran Wang and Chao Qi
16:40–17:05	*A Hybrid Convolutional Variational Autoencoder for Text Generation* Stanislau Semeniuta, Aliaksei Severyn and Erhardt Barth
17:05–17:30	*Filling the Blanks (hint: plural noun) for Mad Libs Humor* Nabil Hossain, John Krumm, Lucy Vanderwende, Eric Horvitz and Henry Kautz
15:50–17:30	**Session 3C: Semantics 1**
15:50–16:15	*Measuring Thematic Fit with Distributional Feature Overlap* Enrico Santus, Emmanuele Chersoni, Alessandro Lenci and Philippe Blache
16:15–16:40	*SCDV : Sparse Composite Document Vectors using soft clustering over distributional representations* Dheeraj Mekala, Vivek Gupta, Bhargavi Paranjape and Harish Karnick
16:40–17:05	*Supervised Learning of Universal Sentence Representations from Natural Language Inference Data* Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault and Antoine Bordes
17:05–17:30	*Determining Semantic Textual Similarity using Natural Deduction Proofs* Hitomi Yanaka, Koji Mineshima, Pascual Martínez-Gómez and Daisuke Bekki

15:50–17:30 Session 3D: Poster Session. Syntax 2

Multi-Grained Chinese Word Segmentation
Chen Gong, Zhenghua Li, Min Zhang and Xinzhou Jiang

Don't Throw Those Morphological Analyzers Away Just Yet: Neural Morphological Disambiguation for Arabic
Nasser Zalmout and Nizar Habash

Paradigm Completion for Derivational Morphology
Ryan Cotterell, Ekaterina Vylomova, Huda Khayrallah, Christo Kirov and David Yarowsky

A Sub-Character Architecture for Korean Language Processing
Karl Stratos

Do LSTMs really work so well for PoS tagging? – A replication study
Tobias Horsmann and Torsten Zesch

The Labeled Segmentation of Printed Books
Lara McConnaughey, Jennifer Dai and David Bamman

Cross-lingual Character-Level Neural Morphological Tagging
Ryan Cotterell and Georg Heigold

Word-Context Character Embeddings for Chinese Word Segmentation
Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, XIN-YU DAI and Jiajun Chen

Segmentation-Free Word Embedding for Unsegmented Languages
Takamasa Oshikiri

Saturday, September 9, 2017 (continued)

15:50–17:30 Session 3E: Poster Session. Question Answering and Machine Comprehension

From Textbooks to Knowledge: A Case Study in Harvesting Axiomatic Knowledge from Textbooks to Solve Geometry Problems
Mrinmaya Sachan, Kumar Dubey and Eric Xing

RACE: Large-scale ReAding Comprehension Dataset From Examinations
Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang and Eduard Hovy

Beyond Sentential Semantic Parsing: Tackling the Math SAT with a Cascade of Tree Transducers
Mark Hopkins, Cristian Petrescu-Prahova, Roie Levin, Ronan Le Bras, Alvaro Herrasti and Vidur Joshi

Learning Fine-Grained Expressions to Solve Math Word Problems
Danqing Huang, Shuming Shi, Chin-Yew Lin and Jian Yin

Structural Embedding of Syntactic Trees for Machine Comprehension
Rui Liu, Junjie Hu, Wei Wei, Zi Yang and Eric Nyberg

World Knowledge for Reading Comprehension: Rare Entity Prediction with Hierarchical LSTMs Using External Descriptions
Teng Long, Emmanuel Bengio, Ryan Lowe, Jackie Chi Kit Cheung and Doina Precup

Two-Stage Synthesis Networks for Transfer Learning in Machine Comprehension
David Golub, Po-Sen Huang, Xiaodong He and Li Deng

Deep Neural Solver for Math Word Problems
Yan Wang, Xiaojiang Liu and Shuming Shi

Latent Space Embedding for Retrieval in Question-Answer Archives
Deepak P, Dinesh Garg and Shirish Shevade

Question Generation for Question Answering
Nan Duan, Duyu Tang, Peng Chen and Ming Zhou

Learning to Paraphrase for Question Answering
Li Dong, Jonathan Mallinson, Siva Reddy and Mirella Lapata

Temporal Information Extraction for Question Answering Using Syntactic Dependencies in an LSTM-based Architecture
Yuanliang Meng, Anna Rumshisky and Alexey Romanov

Ranking Kernels for Structures and Embeddings: A Hybrid Preference and Classification Model
Kateryna Tymoshenko, Daniele Bonadiman and Alessandro Moschitti

Recovering Question Answering Errors via Query Revision
Semih Yavuz, Izzeddin Gur, Yu Su and Xifeng Yan

15:50–17:30 Session 3F: Poster Session. Multimodal NLP 1

An empirical study on the effectiveness of images in Multimodal Neural Machine Translation
Jean-Benoit Delbrouck and Stéphane Dupont

Sound-Word2Vec: Learning Word Representations Grounded in Sounds
Ashwin Vijayakumar, Ramakrishna Vedantam and Devi Parikh

The Promise of Premise: Harnessing Question Premises in Visual Question Answering
Aroma Mahendru, Viraj Prabhu, Akrit Mohapatra, Dhruv Batra and Stefan Lee

Guided Open Vocabulary Image Captioning with Constrained Beam Search
Peter Anderson, Basura Fernando, Mark Johnson and Stephen Gould

Zero-Shot Activity Recognition with Verb Attribute Induction
Rowan Zellers and Yejin Choi

Deriving continous grounded meaning representations from referentially structured multimodal contexts
Sina Zarrieß and David Schlangen

Hierarchically-Attentive RNN for Album Summarization and Storytelling
Licheng Yu, Mohit Bansal and Tamara Berg

Video Highlight Prediction Using Audience Chat Reactions
Cheng-Yang Fu, Joon Lee, Mohit Bansal and Alexander Berg

Saturday, September 9, 2017 (continued)

Reinforced Video Captioning with Entailment Rewards
Ramakanth Pasunuru and Mohit Bansal

Evaluating Hierarchies of Verb Argument Structure with Hierarchical Clustering
Jesse Mu, Joshua K. Hartshorne and Timothy O'Donnell

Incorporating Global Visual Features into Attention-based Neural Machine Translation.
Iacer Calixto and Qun Liu

Mapping Instructions and Visual Observations to Actions with Reinforcement Learning
Dipendra Misra, John Langford and Yoav Artzi

An analysis of eye-movements during reading for the detection of mild cognitive impairment
Kathleen C. Fraser, Kristina Lundholm Fors, Dimitrios Kokkinakis and Arto Nordlund

Evaluating Low-Level Speech Features Against Human Perceptual Data
Caitlin Richter, Naomi H Feldman, Harini Salgado and Aren Jansen

Sunday, September 10, 2017

07:30–17:30 **Registration Day 2**

08:00–09:00 ***Morning Coffee***

09:00–10:00 Plenary Session. Invited Talk by Sharon Goldwater

09:00–10:00 *Towards more universal language technology: unsupervised learning from speech*
Sharon Goldwater

10:00–10:30 *Coffee Break*

10:30–12:10 Session 4A: Reading and Retrieving

10:30–10:55 *A Structured Learning Approach to Temporal Relation Extraction*
Qiang Ning, Zhili Feng and Dan Roth

10:55–11:20 *Importance sampling for unbiased on-demand evaluation of knowledge base population*
Arun Chaganty, Ashwin Paranjape, Percy Liang and Christopher D. Manning

11:20–11:45 *PACRR: A Position-Aware Neural IR Model for Relevance Matching*
Kai Hui, Andrew Yates, Klaus Berberich and Gerard de Melo

11:45–12:10 *Globally Normalized Reader*
Jonathan Raiman and John Miller

10:30–12:10 Session 4B: Multimodal NLP 2

10:30–10:55 *Speech segmentation with a neural encoder model of working memory*
Micha Elsner and Cory Shain

10:55–11:20 *Speaking, Seeing, Understanding: Correlating semantic models with conceptual representation in the brain*
Luana Bulat, Stephen Clark and Ekaterina Shutova

11:20–11:45 *Multi-modal Summarization for Asynchronous Collection of Text, Image, Audio and Video*
Haoran Li, Junnan Zhu, Cong Ma, Jiajun Zhang and Chengqing Zong

11:45–12:10 *Tensor Fusion Network for Multimodal Sentiment Analysis*
Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria and Louis-Philippe Morency

10:30–12:10 Session 4C: Human Centered NLP and Linguistic Theory

10:30–10:55 *ConStance: Modeling Annotation Contexts to Improve Stance Classification*
Kenneth Joseph, Lisa Friedland, William Hobbs, David Lazer and Oren Tsur

10:55–11:20 *Deeper Attention to Abusive User Content Moderation*
John Pavlopoulos, Prodromos Malakasiotis and Ion Androutsopoulos

11:20–11:45 *Outta Control: Laws of Semantic Change and Inherent Biases in Word Representation Models*
Haim Dubossarsky, Daphna Weinshall and Eitan Grossman

11:45–12:10 *Human Centered NLP with User-Factor Adaptation*
Veronica Lynn, Youngseo Son, Vivek Kulkarni, Niranjan Balasubramanian and H. Andrew Schwartz

10:30–12:10 Session 4D: Poster Session. Semantics 2

Neural Sequence Learning Models for Word Sense Disambiguation
Alessandro Raganato, Claudio Delli Bovi and Roberto Navigli

Learning Word Relatedness over Time
Guy D. Rosin, Eytan Adar and Kira Radinsky

Inter-Weighted Alignment Network for Sentence Pair Modeling
Gehui Shen, Yunlun Yang and Zhi-Hong Deng

A Short Survey on Taxonomy Learning from Text Corpora: Issues, Resources and Recent Advances
Chengyu Wang, Xiaofeng He and Aoying Zhou

Idiom-Aware Compositional Distributed Semantics
Pengfei Liu, Kaiyu Qian, Xipeng Qiu and Xuanjing Huang

Macro Grammars and Holistic Triggering for Efficient Semantic Parsing
Yuchen Zhang, Panupong Pasupat and Percy Liang

A Continuously Growing Dataset of Sentential Paraphrases
Wuwei Lan, Siyu Qiu, Hua He and Wei Xu

Cross-domain Semantic Parsing via Paraphrasing
Yu Su and Xifeng Yan

A Joint Sequential and Relational Model for Frame-Semantic Parsing
Bishan Yang and Tom Mitchell

Getting the Most out of AMR Parsing
Chuan Wang and Nianwen Xue

AMR Parsing using Stack-LSTMs
Miguel Ballesteros and Yaser Al-Onaizan

An End-to-End Deep Framework for Answer Triggering with a Novel Group-Level Objective
Jie Zhao, Yu Su, Ziyu Guan and Huan Sun

Predicting Word Association Strengths
Andrew Cattle and Xiaojuan Ma

10:30–12:10 Session 4E: Poster Session. Discourse

Learning Contextually Informed Representations for Linear-Time Discourse Parsing
Yang Liu and Mirella Lapata

Multi-task Attention-based Neural Networks for Implicit Discourse Relationship Representation and Identification
Man Lan, Jianxiang Wang, Yuanbin Wu, Zheng-Yu Niu and Haifeng Wang

Chinese Zero Pronoun Resolution with Deep Memory Network
Qingyu Yin, Yu Zhang, Weinan Zhang and Ting Liu

How much progress have we made on RST discourse parsing? A replication study of recent results on the RST-DT
Mathieu Morey, Philippe Muller and Nicholas Asher

Sunday, September 10, 2017 (continued)

What is it? Disambiguating the different readings of the pronoun 'it'
Sharid Loáiciga, Liane Guillou and Christian Hardmeier

Revisiting Selectional Preferences for Coreference Resolution
Benjamin Heinzerling, Nafise Sadat Moosavi and Michael Strube

Learning to Rank Semantic Coherence for Topic Segmentation
Liang Wang, Sujian Li, Yajuan Lv and Houfeng WANG

GRASP: Rich Patterns for Argumentation Mining
Eyal Shnarch, Ran Levy, Vikas Raykar and Noam Slonim

Patterns of Argumentation Strategies across Topics
Khalid Al Khatib, Henning Wachsmuth, Matthias Hagen and Benno Stein

Using Argument-based Features to Predict and Analyse Review Helpfulness
Haijing Liu, Yang Gao, Pin Lv, Mengxue Li, Shiqiang Geng, Minglan Li and Hao Wang

Here's My Point: Joint Pointer Architecture for Argument Mining
Peter Potash, Alexey Romanov and Anna Rumshisky

Identifying attack and support argumentative relations using deep learning
Oana Cocarascu and Francesca Toni

10:30–12:10 Session 4F: Poster Session. Machine Translation and Multilingual NLP 1

Neural Lattice-to-Sequence Models for Uncertain Inputs
Matthias Sperber, Graham Neubig, Jan Niehues and Alex Waibel

Memory-augmented Neural Machine Translation
Yang Feng, Shiyue Zhang, Andi Zhang, Dong Wang and Andrew Abel

Dynamic Data Selection for Neural Machine Translation
Marlies van der Wees, Arianna Bisazza and Christof Monz

Neural Machine Translation Leveraging Phrase-based Models in a Hybrid Search
Leonard Dahlmann, Evgeny Matusov, Pavel Petrushkov and Shahram Khadivi

Translating Phrases in Neural Machine Translation
Xing Wang, Zhaopeng Tu, Deyi Xiong and Min Zhang

Towards Bidirectional Hierarchical Representations for Attention-based Neural Machine Translation
Baosong Yang, Derek F. Wong, Tong Xiao, Lidia S. Chao and Jingbo Zhu

Massive Exploration of Neural Machine Translation Architectures
Denny Britz, Anna Goldie, Minh-Thang Luong and Quoc Le

Learning Translations via Matrix Completion
Derry Tanti Wijaya, Brendan Callahan, John Hewitt, Jie Gao, Xiao Ling, Marianna Apidianaki and Chris Callison-Burch

Reinforcement Learning for Bandit Neural Machine Translation with Simulated Human Feedback
Khanh Nguyen, Hal Daumé III and Jordan Boyd-Graber

Towards Compact and Fast Neural Machine Translation Using a Combined Method
Xiaowei Zhang, Wei Chen, Feng Wang, Shuang Xu and Bo Xu

Instance Weighting for Neural Machine Translation Domain Adaptation
Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen and Eiichiro Sumita

Regularization techniques for fine-tuning in neural machine translation
Antonio Valerio Miceli Barone, Barry Haddow, Ulrich Germann and Rico Sennrich

Source-Side Left-to-Right or Target-Side Left-to-Right? An Empirical Comparison of Two Phrase-Based Decoding Algorithms
Yin-Wen Chang and Michael Collins

Using Target-side Monolingual Data for Neural Machine Translation through Multi-task Learning
Tobias Domhan and Felix Hieber

12:10–13:40 ***Lunch***

12:40–13:40 SIGDAT Business Meeting

13:40–15:20 Session 5A: Semantics 3

13:40–14:05 *Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling*
Diego Marcheggiani and Ivan Titov

14:05–14:30 *Neural Semantic Parsing with Type Constraints for Semi-Structured Tables*
Jayant Krishnamurthy, Pradeep Dasigi and Matt Gardner

14:30–14:55 *Joint Concept Learning and Semantic Parsing from Natural Language Explanations*
Shashank Srivastava, Igor Labutov and Tom Mitchell

14:55–15:20 *Grasping the Finer Point: A Supervised Similarity Network for Metaphor Detection*
Marek Rei, Luana Bulat, Douwe Kiela and Ekaterina Shutova

13:40–15:20 Session 5B: Computational Social Science 1

13:40–14:05 *Identifying civilians killed by police with distantly supervised entity-event extraction*
Katherine Keith, Abram Handler, Michael Pinkham, Cara Magliozzi, Joshua McDuffie and Brendan O'Connor

14:05–14:30 *Asking too much? The rhetorical role of questions in political discourse*
Justine Zhang, Arthur Spirling and Cristian Danescu-Niculescu-Mizil

14:30–14:55 *Detecting Perspectives in Political Debates*
David Vilares and Yulan He

14:55–15:20 *"i have a feeling trump will win..................": Forecasting Winners and Losers from User Predictions on Twitter*
Sandesh Swamy, Alan Ritter and Marie-Catherine de Marneffe

13:40–15:20 Session 5C: Sentiment Analysis 2

13:40–14:05 *A Question Answering Approach for Emotion Cause Extraction*
Lin Gui, Jiannan Hu, Yulan He, Ruifeng Xu, Lu Qin and Jiachen Du

14:05–14:30 *Story Comprehension for Predicting What Happens Next*
Snigdha Chaturvedi, Haoruo Peng and Dan Roth

14:30–14:55 *Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm*
Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan and Sune Lehmann

14:55–15:20 *Opinion Recommendation Using A Neural Model*
Zhongqing Wang and Yue Zhang

13:40–15:20 Session 5D: Poster Session. Syntax 3

CRF Autoencoder for Unsupervised Dependency Parsing
Jiong Cai, Yong Jiang and Kewei Tu

Efficient Discontinuous Phrase-Structure Parsing via the Generalized Maximum Spanning Arborescence
Caio Corro, Joseph Le Roux and Mathieu Lacroix

Incremental Graph-based Neural Dependency Parsing
Xiaoqing Zheng

Neural Discontinuous Constituency Parsing
Miloš Stanojević and Raquel Garrido Alhama

Stack-based Multi-layer Attention for Transition-based Dependency Parsing
Zhirui Zhang, Shujie Liu, Mu Li, Ming Zhou and Enhong Chen

Dependency Grammar Induction with Neural Lexicalization and Big Training Data
Wenjuan Han, Yong Jiang and Kewei Tu

Combining Generative and Discriminative Approaches to Unsupervised Dependency Parsing via Dual Decomposition
Yong Jiang, Wenjuan Han and Kewei Tu

Effective Inference for Generative Neural Parsing
Mitchell Stern, Daniel Fried and Dan Klein

Semi-supervised Structured Prediction with Neural CRF Autoencoder
Xiao Zhang, Yong Jiang, Hao Peng, Kewei Tu and Dan Goldwasser

TAG Parsing with Neural Networks and Vector Representations of Supertags
Jungo Kasai, Bob Frank, Tom McCoy, Owen Rambow and Alexis Nasr

13:40–15:20 Session 5E: Poster Session. Relations

Global Normalization of Convolutional Neural Networks for Joint Entity and Relation Classification
Heike Adel and Hinrich Schütze

End-to-End Neural Relation Extraction with Global Optimization
Meishan Zhang, Yue Zhang and Guohong Fu

KGEval: Accuracy Estimation of Automatically Constructed Knowledge Graphs
Prakhar Ojha and Partha Talukdar

Sparsity and Noise: Where Knowledge Graph Embeddings Fall Short
Jay Pujara, Eriq Augustine and Lise Getoor

Dual Tensor Model for Detecting Asymmetric Lexico-Semantic Relations
Goran Glavaš and Simone Paolo Ponzetto

Incorporating Relation Paths in Neural Relation Extraction
Wenyuan Zeng, Yankai Lin, Zhiyuan Liu and Maosong Sun

Adversarial Training for Relation Extraction
Yi Wu, David Bamman and Stuart Russell

Context-Aware Representations for Knowledge Base Relation Extraction
Daniil Sorokin and Iryna Gurevych

A Soft-label Method for Noise-tolerant Distantly Supervised Relation Extraction
Tianyu Liu, Kexiang Wang, Baobao Chang and Zhifang Sui

A Sequential Model for Classifying Temporal Relations between Intra-Sentence Events
Prafulla Kumar Choubey and Ruihong Huang

Deep Residual Learning for Weakly-Supervised Relation Extraction
YiYao Huang and William Yang Wang

Noise-Clustered Distant Supervision for Relation Extraction: A Nonparametric Bayesian Perspective
Qing Zhang and Houfeng Wang

Exploring Vector Spaces for Semantic Relations
Kata Gábor, Haifa Zargayouna, Isabelle Tellier, Davide Buscaldi and Thierry Charnois

Temporal dynamics of semantic relations in word embeddings: an application to predicting armed conflict participants
Andrey Kutuzov, Erik Velldal and Lilja Øvrelid

13:40–15:20 Session 5F: Poster Session. Language Models, Text Mining, and Crowd Sourcing

Dynamic Entity Representations in Neural Language Models
Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi and Noah A. Smith

Towards Quantum Language Models
Ivano Basile and Fabio Tamburini

Reference-Aware Language Models
Zichao Yang, Phil Blunsom, Chris Dyer and Wang Ling

A Simple Language Model based on PMI Matrix Approximations
Oren Melamud, Ido Dagan and Jacob Goldberger

Syllable-aware Neural Language Models: A Failure to Beat Character-aware Ones
Zhenisbek Assylbekov, Rustem Takhanov, Bagdat Myrzakhmetov and Jonathan N. Washington

Inducing Semantic Micro-Clusters from Deep Multi-View Representations of Novels
Lea Frermann and György Szarvas

Initializing Convolutional Filters with Semantic Features for Text Classification
Shen Li, Zhe Zhao, Tao Liu, Renfen Hu and Xiaoyong Du

Shortest-Path Graph Kernels for Document Similarity
Giannis Nikolentzos, Polykarpos Meladianos, Francois Rousseau, Yannis Stavrakas and Michalis Vazirgiannis

Adapting Topic Models using Lexical Associations with Tree Priors
Weiwei Yang, Jordan Boyd-Graber and Philip Resnik

Finding Patterns in Noisy Crowds: Regression-based Annotation Aggregation for Crowdsourced Data
Natalie Parde and Rodney Nielsen

CROWD-IN-THE-LOOP: A Hybrid Approach for Annotating Semantic Roles
Chenguang Wang, Alan Akbik, laura chiticariu, Yunyao Li, Fei Xia and Anbang Xu

A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks
Kazuma Hashimoto, caiming xiong, Yoshimasa Tsuruoka and Richard Socher

15:20–15:50 ***Coffee Break***

15:50–17:30 **Session 6A: Machine Translation 2**

15:50–16:15 *Earth Mover's Distance Minimization for Unsupervised Bilingual Lexicon Induction*
Meng Zhang, Yang Liu, Huanbo Luan and Maosong Sun

16:15–16:40 *Unfolding and Shrinking Neural Machine Translation Ensembles*
Felix Stahlberg and Bill Byrne

16:40–17:05 *Graph Convolutional Encoders for Syntax-aware Neural Machine Translation*
Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani and Khalil Simaan

17:05–17:30 *Trainable Greedy Decoding for Neural Machine Translation*
Jiatao Gu, Kyunghyun Cho and Victor O.K. Li

Sunday, September 10, 2017 (continued)

15:50–17:30 Session 6B: Text Mining and NLP applications

15:50–16:15 *Satirical News Detection and Analysis using Attention Mechanism and Linguistic Features*
Fan Yang, Arjun Mukherjee and Eduard Dragut

16:15–16:40 *Fine Grained Citation Span for References in Wikipedia*
Besnik Fetahu, Katja Markert and Avishek Anand

16:40–17:05 *Joint Modeling of Topics, Citations, and Topical Authority in Academic Corpora*
Jooyeon Kim, Dongwoo Kim and Alice Oh

17:05–17:30 *Identifying Semantic Edit Intentions from Revisions in Wikipedia*
Diyi Yang, Aaron Halfaker, Robert Kraut and Eduard Hovy

15:50–17:30 Session 6C: Machine Comprehension

15:50–16:15 *Accurate Supervised and Semi-Supervised Machine Reading for Long Documents*
Daniel Hewlett, Llion Jones, Alexandre Lacoste and izzeddin gur

16:15–16:40 *Adversarial Examples for Evaluating Reading Comprehension Systems*
Robin Jia and Percy Liang

16:40–17:05 *Reasoning with Heterogeneous Knowledge for Commonsense Machine Comprehension*
Hongyu Lin, Le Sun and Xianpei Han

17:05–17:30 *Document-Level Multi-Aspect Sentiment Classification as Machine Comprehension*
Yichun Yin, Yangqiu Song and Ming Zhang

15:50–17:30 Session 6D: Poster Session. Summarization, Generation, Dialog, and Discourse 1

What is the Essence of a Claim? Cross-Domain Claim Identification
Johannes Daxenberger, Steffen Eger, Ivan Habernal, Christian Stab and Iryna Gurevych

Identifying Where to Focus in Reading Comprehension for Neural Question Generation
Xinya Du and Claire Cardie

Break it Down for Me: A Study in Automated Lyric Annotation
Lucas Sterckx, Jason Naradowsky, Bill Byrne, Thomas Demeester and Chris Develder

Cascaded Attention based Unsupervised Information Distillation for Compressive Summarization
Piji Li, Wai Lam, Lidong Bing, Weiwei Guo and Hang Li

Deep Recurrent Generative Decoder for Abstractive Text Summarization
Piji Li, Wai Lam, Lidong Bing and Zihao Wang

Extractive Summarization Using Multi-Task Learning with Document Classification
Masaru Isonuma, Toru Fujino, Junichiro Mori, Yutaka Matsuo and Ichiro Sakata

Towards Automatic Construction of News Overview Articles by News Synthesis
Jianmin Zhang and Xiaojun Wan

Joint Syntacto-Discourse Parsing and the Syntacto-Discourse Treebank
Kai Zhao and Liang Huang

Event Coreference Resolution by Iteratively Unfolding Inter-dependencies among Events
Prafulla Kumar Choubey and Ruihong Huang

When to Finish? Optimal Beam Search for Neural Text Generation (modulo beam size)
Liang Huang, Kai Zhao and Mingbo Ma

Steering Output Style and Topic in Neural Response Generation
Di Wang, Nebojsa Jojic, Chris Brockett and Eric Nyberg

15:50–17:30 Session 6E: Poster Session. Summarization, Generation, Dialog, and Discourse 2

Preserving Distributional Information in Dialogue Act Classification
Quan Hung Tran, Ingrid Zukerman and Gholamreza Haffari

Adversarial Learning for Neural Dialogue Generation
Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter and Dan Jurafsky

Using Context Information for Dialog Act Classification in DNN Framework
Yang Liu, Kun Han, Zhao Tan and Yun Lei

Modeling Dialogue Acts with Content Word Filtering and Speaker Preferences
Yohan Jo, Michael Yoder, Hyeju Jang and Carolyn Rose

Towards Implicit Content-Introducing for Generative Short-Text Conversation Systems
Lili Yao, Yaoyuan Zhang, Yansong Feng, Dongyan Zhao and Rui Yan

Affordable On-line Dialogue Policy Learning
Cheng Chang, Runzhe Yang, Lu Chen, Xiang Zhou and Kai Yu

Generating High-Quality and Informative Conversation Responses with Sequence-to-Sequence Models
Yuanlong Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope and Ray Kurzweil

Bootstrapping incremental dialogue systems from minimal data: the generalisation power of dialogue grammars
Arash Eshghi, Igor Shalyminov and Oliver Lemon

Composite Task-Completion Dialogue Policy Learning via Hierarchical Deep Reinforcement Learning
Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee and Kam-Fai Wong

Why We Need New Evaluation Metrics for NLG
Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry and Verena Rieser

Challenges in Data-to-Document Generation
Sam Wiseman, Stuart Shieber and Alexander Rush

15:50–17:30 Session 6F: Poster Session. Computational Social Science 2

All that is English may be Hindi: Enhancing language identification through automatic ranking of the likeliness of word borrowing in social media
Jasabanta Patro, Bidisha Samanta, Saurabh Singh, Abhipsa Basu, Prithwish Mukherjee, Monojit Choudhury and Animesh Mukherjee

Multi-View Unsupervised User Feature Embedding for Social Media-based Substance Use Prediction
Tao Ding, Warren K. Bickel and Shimei Pan

Demographic-aware word associations
Aparna Garimella, Carmen Banea and Rada Mihalcea

A Factored Neural Network Model for Characterizing Online Discussions in Vector Space
Hao Cheng, Hao Fang and Mari Ostendorf

Dimensions of Interpersonal Relationships: Corpus and Experiments
Farzana Rashid and Eduardo Blanco

Argument Mining on Twitter: Arguments, Facts and Sources
Mihai Dusmanu, Elena Cabrio and Serena Villata

Distinguishing Japanese Non-standard Usages from Standard Ones
Tatsuya Aoki, Ryohei Sasano, Hiroya Takamura and Manabu Okumura

Connotation Frames of Power and Agency in Modern Films
Maarten Sap, Marcella Cindy Prasettio, Ari Holtzman, Hannah Rashkin and Yejin Choi

Controlling Human Perception of Basic User Traits
Daniel Preoţiuc-Pietro, Sharath Chandra Guntuku and Lyle Ungar

Topic Signatures in Political Campaign Speeches
Clément Gautrais, Peggy Cellier, René Quiniou and Alexandre Termier

Assessing Objective Recommendation Quality through Political Forecasting
H. Andrew Schwartz, Masoud Rouhizadeh, Michael Bishop, Philip Tetlock, Barbara Mellers and Lyle Ungar

Sunday, September 10, 2017 (continued)

Never Abandon Minorities: Exhaustive Extraction of Bursty Phrases on Microblogs Using Set Cover Problem
Masumi Shirakawa, Takahiro Hara and Takuya Maekawa

18:00–22:00 Social Event

Monday, September 11, 2017

07:30–17:30 Registration Day 3

08:00–09:00 *Morning Coffee*

09:00–10:00 Plenary Session. Invited Talk by Dan Jurafsky

09:00–10:00 *"Does This Vehicle Belong to You”? Processing the Language of Policing for Improving Police-Community Relations*
Dan Jurafsky

10:00–10:30 *Coffee Break*

10:30–12:10 Session 7A: Machine Learning 3

10:30–10:55 *Maximum Margin Reward Networks for Learning from Explicit and Implicit Supervision*
Haoruo Peng, Ming-Wei Chang and Wen-tau Yih

10:55–11:20 *The Impact of Modeling Overall Argumentation with Tree Kernels*
Henning Wachsmuth, Giovanni Da San Martino, Dora Kiesel and Benno Stein

11:20–11:45 *Learning Generic Sentence Representations Using Convolutional Neural Networks*
Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He and Lawrence Carin

11:45–12:10 *Repeat before Forgetting: Spaced Repetition for Efficient and Effective Training of Neural Networks*
Hadi Amiri, Timothy Miller and Guergana Savova

10:30–12:10 Session 7B: Syntax 4

10:30–10:55 *Part-of-Speech Tagging for Twitter with Adversarial Neural Networks*
Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng and Xuanjing Huang

10:55–11:20 *Investigating Different Syntactic Context Types and Context Representations for Learning Word Embeddings*
Bofang Li, Tao Liu, Zhe Zhao, Buzhou Tang, Aleksandr Drozd, Anna Rogers and Xiaoyong Du

11:20–11:45 *Does syntax help discourse segmentation? Not so much*
Chloé Braud, Ophélie Lacroix and Anders Søgaard

11:45–12:10 *Nonparametric Bayesian Semi-supervised Word Segmentation*
Ryo Fujii, Ryo Domoto and Daichi Mochihashi

10:30–12:10 Session 7C: Dialogue

10:30–10:55 *Deal or No Deal? End-to-End Learning of Negotiation Dialogues*
Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh and Dhruv Batra

10:55–11:20 *Agent-Aware Dropout DQN for Safe and Efficient On-line Dialogue Policy Learning*
Lu Chen, Xiang Zhou, Cheng Chang, Runzhe Yang and Kai Yu

11:20–11:45 *Towards Debate Automation: a Recurrent Model for Predicting Debate Winners*
Peter Potash and Anna Rumshisky

11:45–12:10 *Conversation Modeling on Reddit Using a Graph-Structured LSTM*
Victoria Zayats and Mari Ostendorf

10:30–12:10 Session 7D: Poster Session. Machine Translation and Multilingual NLP 2

Joint Prediction of Word Alignment with Alignment Types
Anahita Mansouri Bigvand, Te Bu and Anoop Sarkar

Further Investigation into Reference Bias in Monolingual Evaluation of Machine Translation
Qingsong Ma, Yvette Graham, Timothy Baldwin and Qun Liu

A Challenge Set Approach to Evaluating Machine Translation
Pierre Isabelle, Colin Cherry and George Foster

Knowledge Distillation for Bilingual Dictionary Induction
Ndapandula Nakashole and Raphael Flauger

Machine Translation, it's a question of style, innit? The case of English tag questions
Rachel Bawden

Deciphering Related Languages
Nima Pourdamghani and Kevin Knight

Identifying Cognate Sets Across Dictionaries of Related Languages
Adam St Arnaud, David Beck and Grzegorz Kondrak

Learning Language Representations for Typology Prediction
Chaitanya Malaviya, Graham Neubig and Patrick Littell

Cheap Translation for Cross-Lingual Named Entity Recognition
Stephen Mayhew, Chen-Tse Tsai and Dan Roth

Cross-Lingual Induction and Transfer of Verb Classes Based on Word Vector Space Specialisation
Ivan Vulić, Nikola Mrkšić and Anna Korhonen

Classification of telicity using cross-linguistic annotation projection
Annemarie Friedrich and Damyana Gateva

Semantic Specialisation of Distributional Word Vector Spaces using Monolingual and Cross-Lingual Constraints
Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen and Steve Young

Counterfactual Learning from Bandit Feedback under Deterministic Logging : A Case Study in Statistical Machine Translation
Carolin Lawrence, Artem Sokolov and Stefan Riezler

10:30–12:10 Session 7E: Poster Session. Information Extraction 2

Learning Fine-grained Relations from Chinese User Generated Categories
Chengyu Wang, Yan Fan, Xiaofeng He and Aoying Zhou

Improving Slot Filling Performance with Attentive Neural Networks on Dependency Structures
Lifu Huang, Avirup Sil, Heng Ji and Radu Florian

Identifying Products in Online Cybercrime Marketplaces: A Dataset for Fine-grained Domain Adaptation
Greg Durrett, Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick, Rebecca Portnoff, Sadia Afroz, Damon McCoy, Kirill Levchenko and Vern Paxson

Labeling Gaps Between Words: Recognizing Overlapping Mentions with Mention Separators
Aldrian Obaja Muis and Wei Lu

Deep Joint Entity Disambiguation with Local Neural Attention
Octavian-Eugen Ganea and Thomas Hofmann

MinIE: Minimizing Facts in Open Information Extraction
Kiril Gashteovski, Rainer Gemulla and Luciano Del Corro

Scientific Information Extraction with Semi-supervised Neural Tagging
Yi Luan, Mari Ostendorf and Hannaneh Hajishirzi

NITE: A Neural Inductive Teaching Framework for Domain Specific NER
Siliang Tang, Ning Zhang, Jinjiang Zhang, Fei Wu and Yueting Zhuang

Speeding up Reinforcement Learning-based Information Extraction Training using Asynchronous Methods
Aditya Sharma, Zarana Parekh and Partha Talukdar

Leveraging Linguistic Structures for Named Entity Recognition with Bidirectional Recursive Neural Networks
Peng-Hsuan Li, Ruo-Ping Dong, Yu-Siang Wang, Ju-Chieh Chou and Wei-Yun Ma

Fast and Accurate Entity Recognition with Iterated Dilated Convolutions
Emma Strubell, Patrick Verga, David Belanger and Andrew McCallum

Entity Linking via Joint Encoding of Types, Descriptions, and Context
Nitish Gupta, Sameer Singh and Dan Roth

An Insight Extraction System on BioMedical Literature with Deep Neural Networks
Hua He, Kris Ganjam, Navendu Jain, Jessica Lundin, Ryen White and Jimmy Lin

10:30–12:10 Session 7F: Poster Session. NLP Applications

Word Etymology as Native Language Interference
Vivi Nastase and Carlo Strapparava

A Simpler and More Generalizable Story Detector using Verb and Character Features
Joshua Eisenberg and Mark Finlayson

Multi-modular domain-tailored OCR post-correction
Sarah Schulz and Jonas Kuhn

Learning to Predict Charges for Criminal Cases with Legal Basis
Bingfeng Luo, Yansong Feng, Jianbo Xu, Xiang Zhang and Dongyan Zhao

Quantifying the Effects of Text Duplication on Semantic Models
Alexandra Schofield, Laure Thompson and David Mimno

Identifying Semantically Deviating Outlier Documents
Honglei Zhuang, Chi Wang, Fangbo Tao, Lance Kaplan and Jiawei Han

Detecting and Explaining Causes From Text For a Time Series Event
Dongyeop Kang, Varun Gangal, Ang Lu, Zheng Chen and Eduard Hovy

A Novel Cascade Model for Learning Latent Similarity from Heterogeneous Sequential Data of MOOC
Zhuoxuan Jiang, Shanshan Feng, Gao Cong, Chunyan Miao and Xiaoming Li

Identifying the Provision of Choices in Privacy Policy Text
Kanthashree Mysore Sathyendra, Shomir Wilson, Florian Schaub, Sebastian Zimmeck and Norman Sadeh

An Empirical Analysis of Edit Importance between Document Versions
Tanya Goyal, Sachin Kelkar, Manas Agarwal and Jeenu Grover

Transition-Based Disfluency Detection using LSTMs
Shaolei Wang, Wanxiang Che, Yue Zhang, Meishan Zhang and Ting Liu

Neural Sequence-Labelling Models for Grammatical Error Correction
Helen Yannakoudakis, Marek Rei, Øistein E. Andersen and Zheng Yuan

Adapting Sequence Models for Sentence Correction
Allen Schmaltz, Yoon Kim, Alexander Rush and Stuart Shieber

12:10–13:40 ***Lunch***

13:40–15:25 **Session 8A: Machine Translation and Multilingual/Multimodal NLP (Short)**

13:40–13:55 *A Study of Style in Machine Translation: Controlling the Formality of Machine Translation Output*
Xing Niu, Marianna Martindale and Marine Carpuat

13:55–14:10 *Sharp Models on Dull Hardware: Fast and Accurate Neural Machine Translation Decoding on the CPU*
Jacob Devlin

14:10–14:25 *Exploiting Cross-Sentence Context for Neural Machine Translation*
Longyue Wang, Zhaopeng Tu, Andy Way and Qun Liu

14:25–14:40 *Cross-Lingual Transfer Learning for POS Tagging without Cross-Lingual Resources*
Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya and Eric Fosler-Lussier

Monday, September 11, 2017 (continued)

14:40–14:55 *Image Pivoting for Learning Multilingual Multimodal Representations*
Spandana Gella, Rico Sennrich, Frank Keller and Mirella Lapata

14:55–15:10 *Neural Machine Translation with Source Dependency Representation*
Kehai Chen, Rui Wang, Masao Utiyama, Lemao Liu, Akihiro Tamura, Eiichiro Sumita and Tiejun Zhao

15:10–15:25 *Visual Denotations for Recognizing Textual Entailment*
Dan Han, Pascual Martínez-Gómez and Koji Mineshima

13:40–15:25 Session 8B: Machine Learning (Short)

13:40–13:55 *Sequence Effects in Crowdsourced Annotations*
Nitika Mathur, Timothy Baldwin and Trevor Cohn

13:55–14:10 *No Need to Pay Attention: Simple Recurrent Neural Networks Work!*
Ferhan Ture and Oliver Jojic

14:10–14:25 *The strange geometry of skip-gram with negative sampling*
David Mimno and Laure Thompson

14:25–14:40 *Natural Language Processing with Small Feed-Forward Networks*
Jan A. Botha, Emily Pitler, Ji Ma, Anton Bakalov, Alex Salcianu, David Weiss, Ryan McDonald and Slav Petrov

14:40–14:55 *Deep Multi-Task Learning for Aspect Term Extraction with Memory Interaction*
Xin Li and Wai Lam

14:55–15:10 *Analogs of Linguistic Structure in Deep Representations*
Jacob Andreas and Dan Klein

15:10–15:25 *A Simple Regularization-based Algorithm for Learning Cross-Domain Word Embeddings*
Wei Yang, Wei Lu and Vincent Zheng

Monday, September 11, 2017 (continued)

13:40–15:25 Session 8C: NLP Applications (Short)

13:40–13:55 *Learning what to read: Focused machine reading*
Enrique Noriega-Atala, Marco A. Valenzuela-Escárcega, Clayton Morrison and Mihai Surdeanu

13:55–14:10 *DOC: Deep Open Classification of Text Documents*
Lei Shu, Hu Xu and Bing Liu

14:10–14:25 *Charmanteau: Character Embedding Models For Portmanteau Creation*
Varun Gangal, Harsh Jhamtani, Graham Neubig, Eduard Hovy and Eric Nyberg

14:25–14:40 *Using Automated Metaphor Identification to Aid in Detection and Prediction of First-Episode Schizophrenia*
E. Dario Gutierrez, Guillermo Cecchi, Cheryl Corcoran and Philip Corlett

14:40–14:55 *Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking*
Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova and Yejin Choi

14:55–15:10 *Topic-Based Agreement and Disagreement in US Electoral Manifestos*
Stefano Menini, Federico Nanni, Simone Paolo Ponzetto and Sara Tonelli

15:10–15:25 *Zipporah: a Fast and Scalable Data Cleaning System for Noisy Web-Crawled Parallel Corpora*
Hainan Xu and Philipp Koehn

15:25–15:50 *Coffee Break*

15:50–17:25 Plenary Session. Best Paper

17:00–17:25 *Bringing Structure into Summaries: Crowdsourcing a Benchmark Corpus of Concept Maps*
Tobias Falke and Iryna Gurevych

16:20–16:35 *Natural Language Does Not Emerge 'Naturally' in Multi-Agent Dialog*
Satwik Kottur, José Moura, Stefan Lee and Dhruv Batra

16:35–17:00 *Depression and Self-Harm Risk Assessment in Online Forums*
Andrew Yates, Arman Cohan and Nazli Goharian

15:55–16:20 *Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints*
Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez and Kai-Wei Chang

17:25–17:45 Plenary Session. Closing Remarks

17:25–17:45 *Closing Remarks*
General Chair

Monolingual Phrase Alignment on Parse Forests

Yuki Arase[1⋆] and **Junichi Tsujii**[⋆2]
[1]Osaka University, Japan
[⋆]Artificial Intelligence Research Center (AIRC), AIST, Japan
[2]NaCTeM, School of Computer Science, University of Manchester, UK
arase@ist.osaka-u.ac.jp, j-tsujii@aist.go.jp

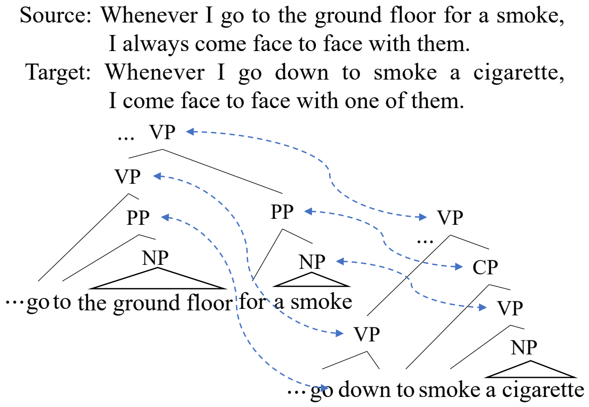
Abstract

We propose an efficient method to conduct phrase alignment on parse forests for paraphrase detection. Unlike previous studies, our method identifies syntactic paraphrases under linguistically motivated grammar. In addition, it allows phrases to non-compositionally align to handle paraphrases with non-homographic phrase correspondences. A dataset that provides gold parse trees and their phrase alignments is created. The experimental results confirm that the proposed method conducts highly accurate phrase alignment compared to human performance.

1 Introduction

Paraphrase detection is crucial in various applications, which has been actively studied for years. Due to difficulties caused by the non-homographic nature of phrase correspondences, the units of correspondence in previous studies are defined as sequences of words like in (Yao et al., 2013) and not syntactic phrases. On the other hand, syntactic structures are important in modeling sentences, *e.g.*, their sentiments and semantic similarities (Socher et al., 2013; Tai et al., 2015).

In this paper, we present an algorithm to align syntactic phrases in a paraphrased pair of sentences. We show that (1) the problem of identifying a legitimate set of syntactic paraphrases under linguistically motivated grammar is formalized, (2) dynamic programing a la CKY (Cocke, 1969; Kasami, 1965; Younger, 1967) makes phrase alignment computationally feasible, (3) alignment quality of phrases can be improved using n-best parse forests instead of 1-best trees, and (4) non-compositional alignment allows non-homographic correspondences of phrases. Motivated by recent findings that syntax is important for phrase embedding (Socher et al., 2013) in which phrasal paraphrases allow semantic similarity to be replicated (Wieting et al., 2016, 2015), we focus on the syntactic paraphrase alignment.



Source: Whenever I go to the ground floor for a smoke,
I always come face to face with them.
Target: Whenever I go down to smoke a cigarette,
I come face to face with one of them.

... VP
VP
PP
NP
PP
NP
...go to the ground floor for a smoke
VP
...
CP
VP
VP
NP
...go down to smoke a cigarette

Figure 1: Example of phrase alignments

Fig. 1 shows a real example of phrase alignments produced by our method. Alignment proceeds in a bottom-up manner using the compositional nature of phrase alignments. First, word alignments are given. Then, phrase alignments are recursively identified by supporting relations between phrase pairs. Non-compositional alignment is triggered when the compositionality is violated, which is common in paraphrasing.

For systematic research on syntactic phrase alignment in paraphrases, we constructed a gold standard dataset of paraphrase sentences with phrase alignment ($20,678$ phrases in 201 paraphrasal sentences). This dataset will be made public for future research on paraphrase alignment. The experiment results show that our method achieves 83.64% and 78.91% in recall and precision in terms of alignment pairs, which are 92% and 89% of human performance, respectively.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1–11
Copenhagen, Denmark, September 7–11, 2017.

2 Related Work

Due to the large amount of sentence-level paraphrases collected (Dolan et al., 2004; Cohn et al., 2008; Heilman and Smith, 2010; Yin and Schütze, 2015; Biran et al., 2016), researchers can identify phrasal correspondences for natural language inferences (MacCartney et al., 2008; Thadani et al., 2012; Yao et al., 2013). Current methods extend word alignments to phrases in accordance with the methods in statistical machine translation. However, phrases are defined as a simple sequence of words, which do not conform to syntactic phrases. PPDB (Ganitkevitch et al., 2013) provides syntactic paraphrases similar to synchronous context free grammar (SCFG). As discussed below, SCFG captures only a fraction of paraphrasing phenomenon.

In terms of our approach, parallel parsing is a relevant area. Smith and Smith (2004) related monolingual parses in different languages using word alignments, while Burkett and Klein (2008) employed phrase alignments. Moreover, Das and Smith (2009) proposed a model that generates a paraphrase of a given sentence using quasi-synchronous dependency grammar (Smith and Eisner, 2006). Since they used phrase alignments simply as features, there is no guarantee that the output alignments are legitimate.

Synchronous rewriting in parallel parsing (Kaeshammer, 2013; Maillette de Buy Wenniger and Sima'an, 2013) derives parse trees that conform to discontinuous word alignments. In contrast, our method respects parse trees derived by linguistically motivated grammar while handling non-monotonic phrase alignment.

The synchronous assumption in parallel parsing has been argued to be too rigid to handle parallel sentence pairs or even paraphrasal sentence pairs. Burkett et al. (2010) proposed weakly synchronized parallel parsing to tackle this problem. Although this model increases the flexibility, the obtainable alignments are restricted to conform to inversion transduction grammar (ITG) (Wu, 1997). Similarly, Choe and McClosky (2015) used dependency forests of paraphrasal sentence pairs and allowed disagreements to some extent. However, alignment quality was beyond their scope. Weese et al. (2014) extracted SCFG from paraphrase corpora. They showed that parsing was only successful in 9.1% of paraphrases, confirming that a significant amount of transformations in paraphrases do not conform to compositionality or ITG.

	Explanation
s, t	Source and target sentences
τ	Phrase in the parse tree
$\tau_R, \tau_\emptyset$	τ_R is a phrase of a root node; $\tau_\emptyset$ is a special phrase with the null span that exists in every parse tree
ϕ	Phrase aligned to $\tau_\emptyset$
$\langle \cdot, \cdot \rangle$	Pair of entities; a pair itself can be regarded as an entity
$\{\cdot\}$	Set of entities
$m(\cdot)$	Derive the mother node of a phrase
$l(\cdot)$, $r(\cdot)$	Derive the left and right child nodes, respectively
$ds(\cdot)$	Derive descendants of a node including self; $\tau \in ds(\tau)$
$lca(\cdot, \cdot)$	Derive the lowest common ancestor (LCA) of two phrases

Table 1: Notation summary

3 Formulation of Phrase Alignment

In this study, we formalize the problem of legitimate phrase alignment. For simplicity, we discuss tree alignment instead of forests using Fig. 2 as a running example.

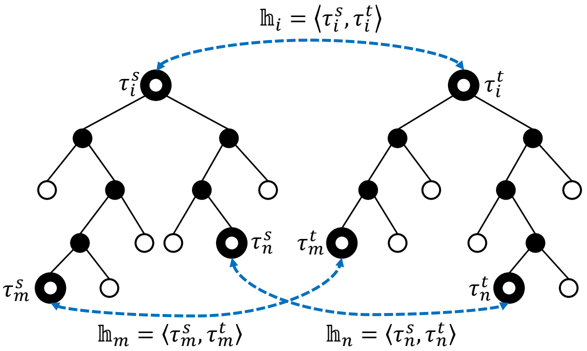
3.1 Notation

Table 1 describes the notation used in this paper. We call a paraphrased pair *source* sentence s and the other as *target* t. Superscripts of s and t represent the source and the target, respectively. Specifically, $\langle \tau^s, \tau^t \rangle$ is a pair of source and target phrases. We represent $f_1/f_2/\cdots/f_i(\cdot)$ to abbreviate $f_i(\cdots f_2(f_1(\cdot))\cdots)$ as an intuitive illustration. It should be noted that the order of the function symbols is reversed, *e.g.*, $l/r(\tau)$ $(= r(l(\tau)))$ derives the right-child of the left-child node of τ, and $l/ds(\tau)$ derives the left descendants of τ.

3.2 Definition of a Legitimate Alignment

A possible parse tree alignment of s and t is represented as a set of aligned pairs of phrases $\{\langle \tau_i^s, \tau_i^t \rangle\}$. τ_i^s and τ_i^t are the source and the target phrases that constitute the i-th alignment, respectively. Either τ_i^s or τ_i^t can be $\tau_\emptyset$ when a phrase does not correspond to another sentence, which is called a *null-alignment*. Each phrase alignment can have *support* relations as:

Definition 3.1. *A pair* $\mathbb{h}_i = \langle \tau_i^s, \tau_i^t \rangle$ *is supported by alignments of their descendant phrases when*



$\mathbb{h}_i = \langle \tau_i^s, \tau_i^t \rangle$
τ_i^s
τ_i^t
τ_n^s
τ_m^t
τ_m^s
τ_n^t
$\mathbb{h}_m = \langle \tau_m^s, \tau_m^t \rangle$
$\mathbb{h}_n = \langle \tau_n^s, \tau_n^t \rangle$

Figure 2: Alignment pair and its supports

$\langle\langle l/ds(\tau_i^s), l/ds(\tau_i^t)\rangle, \langle r/ds(\tau_i^s), r/ds(\tau_i^t)\rangle\rangle$ *or* $\langle\langle l/ds(\tau_i^s), r/ds(\tau_i^t)\rangle, \langle r/ds(\tau_i^s), l/ds(\tau_i^t)\rangle\rangle$ *exists. Pre-terminal phrases are supported by the corresponding word alignments.*

Support relations are denoted using $\Rightarrow$ or $\overset{R}{\Rightarrow}$ that represent the order of support phrases. Specifically, $\langle\langle l(\tau_i^s), l(\tau_i^t)\rangle, \langle r(\tau_i^s), r(\tau_i^t)\rangle\rangle \Rightarrow \mathbb{h}_i$ is straight while $\langle\langle l(\tau_i^s), r(\tau_i^t)\rangle, \langle r(\tau_i^s), l(\tau_i^t)\rangle\rangle \overset{R}{\Rightarrow} \mathbb{h}_i$ is inverted. In Fig. 2, $\langle\langle \tau_m^s, \tau_m^t\rangle, \langle \tau_n^s, \tau_n^t\rangle\rangle \Rightarrow \mathbb{h}_i$, where $\tau_m^s = l/ds(\tau_i^s)$ and $\tau_n^s = r/ds(\tau_i^s)$.

The number of all possible alignments in s and t, which is denoted as $\mathbb{H}$, is exponential to the length. However, only its fraction constitutes legitimate parse tree alignments. For example, a subset in which the same phrase in s is aligned with multiple phrases in t, called *competing* alignments, is not legitimate as a parse tree alignment. The relationships among phrases in parse trees impose constraints on a subset to provide legitimacy.

Given word alignments $\mathbb{W}$ that provide the basis for the phrase alignment, its legitimate set $\mathbb{W}_L \subset \mathbb{W}$ should be 1-to-1 alignments. Starting with $\mathbb{W}_L$, a legitimate set of phrase alignments $\mathbb{H}_L(\subset \mathbb{H})$ with an accompanying set of support relations, $\Delta_L(\subset \Delta)$ is constructed. A legitimate set of alignments $\langle \mathbb{H}_L, \Delta_L\rangle$ can be enlarged only by adding $\mathbb{h}_i$ to $\mathbb{H}_L$ with either the support relation $\Rightarrow$ or $\overset{R}{\Rightarrow}$ added to Δ_L. These assume competing alignments among the child phrases, thus cannot co-exist in the same legitimate set.

$\mathbb{h}_i$ can be supported by more than one pair of descendant alignments in Δ_L, *i.e.*, $\{\langle \mathbb{h}_m, \cdot\rangle\} \Rightarrow \mathbb{h}_i$ or $\{\langle \mathbb{h}_m, \cdot\rangle\} \overset{R}{\Rightarrow} \mathbb{h}_i$ exists. For $\mathbb{H}_m = \{\mathbb{h}_m\}$, we define the relationship $\leq$ for alignments, *i.e.*, $\mathbb{h}_p \leq \mathbb{h}_q$ meaning that $\tau_p^s \in ds(\tau_q^s) \wedge \tau_p^t \in ds(\tau_q^t)$. For example, in Fig. 2, $\mathbb{h}_m \leq \mathbb{h}_i$ and $\mathbb{h}_n \leq \mathbb{h}_i$.

Theorem 3.1. *There always exist the maximum pair* $\mathbb{h}_M \in \mathbb{H}_m$ *where* $\forall \mathbb{h}_m \in \mathbb{H}_m, \mathbb{h}_m \leq \mathbb{h}_M$.

$\langle \mathbb{H}_L, \Delta_L\rangle$ should satisfy the conditions in Definition 3.2 to be legitimate as a whole. We denote $\mathbb{h}_i \overset{*}{\mapsto} \mathbb{h}_j$ when a chain exists in Δ_L, which connects $\mathbb{h}_i$ to $\mathbb{h}_j$ regardless of straight or inverted directions of intermediate supports, e.g., $(\langle \mathbb{h}_i, \cdot\rangle \Rightarrow \mathbb{h}_{i+1}), (\langle \mathbb{h}_{i+1}, \cdot\rangle \overset{R}{\Rightarrow} \mathbb{h}_{i+2}), \ldots, (\langle \mathbb{h}_{j-1}, \cdot\rangle \Rightarrow \mathbb{h}_j)$. Note $\mathbb{h}_i \overset{*}{\mapsto} \mathbb{h}_i$ is always true.

Definition 3.2. $\langle \mathbb{H}_L, \Delta_L\rangle$ *should satisfy:*

1. *Root-Pair Containment:* $\langle \tau_R^s, \tau_R^t\rangle \in \mathbb{H}_L$
2. *Same-Tree:* $\{\tau_i^s \mid \langle \tau_i^s, \tau_i^t\rangle \in \mathbb{H}_L\}$ *are subsets of phrases in the same complete parse tree of* s *(same for* t*).*
3. *Relevance:* $\forall \mathbb{h}_i \in \mathbb{H}_L, \mathbb{h}_i \overset{*}{\mapsto} \langle \tau_R^s, \tau_R^t\rangle \in \Delta_L$
4. *Consistency: In* $\mathbb{H}_L$*, a phrase (*$\neq \tau_\emptyset$*) in the source tree is aligned with at most one phrase (*$\neq \tau_\emptyset$*) in the target tree, and vice versa.*
5. *Monotonous: For* $\langle \tau_i^s, \tau_i^t\rangle, \langle \tau_j^s, \tau_j^t\rangle \in \mathbb{H}_L$, $\tau_i^s \in ds(\tau_j^s)$ *iff* $\tau_i^t \in ds(\tau_j^t)$.
6. *Maximum Set:* $\mathbb{H}_L$ *is the maximum legitimate set, in the sense that* $\forall \langle \tau^s, \tau^t\rangle \in (\mathbb{H} \setminus \mathbb{H}_L), \{\langle \tau^s, \tau^t\rangle\} \cup \mathbb{H}_L$ *cannot be a legitimate set with any* Δ.

The *Same-Tree* condition is required to conduct an alignment on forests that consist of multiple trees in a packed representation. The *Consistency* condition excludes competing alignments. The *Monotonous* condition is a consequence of compositionality. The *Maximum Set* means if $\mathbb{h}_m, \mathbb{h}_n \in \mathbb{H}_L$ are in positions of a parse tree that can support $\mathbb{h}_i$, $\mathbb{h}_i$ and the support relation should be added to $\langle \mathbb{H}_L, \Delta_L\rangle$. Such a strict locality of compositionality is often violated in practice as discussed in Sec. 2. To tackle this issue, we add another operation to align phrases in a non-compositional way in Sec. 4.3.

3.3 Lowest Common Ancestor

The same aligned pair can have more than one support of descendant alignments because there are numerous descendant node combinations. However, the *Monotonous* and the *Maximum Set* conditions allow Δ_L to be further restricted so that each of aligned pairs in $\mathbb{H}_L$ has only one support.

Let us assume that alignment $\mathbb{h}_i$ is supported by more than one pair of descendant alignments

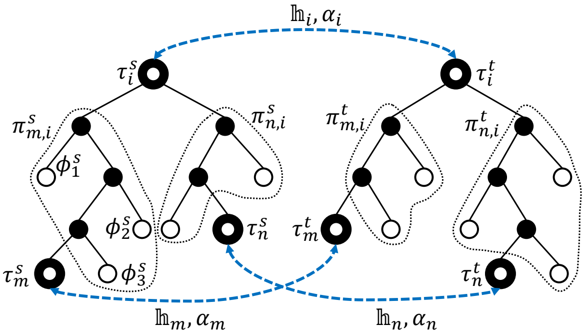


Figure 3: Inside probability depends on support alignments and paths to reach an LCA.

in Δ_L, *i.e.*, $\Delta_L \supseteq (\{\langle \mathbb{h}_m, \mathbb{h}_n \rangle\} \Rightarrow \mathbb{h}_i)$[1]. We denote $\mathbb{H}_m = \{\mathbb{h}_m\}$ and $\mathbb{H}_n = \{\mathbb{h}_n\}$. For each $\mathbb{h}_m \in \mathbb{H}_m$ and $\mathbb{h}_n \in \mathbb{H}_n$, we remove all support relations from Δ_L except for the maximum pairs or the pre-terminal alignments. The resultant set Δ'_L satisfies:

Theorem 3.2. *For all* $(\langle \mathbb{h}_m, \mathbb{h}_n \rangle \Rightarrow \mathbb{h}_i) \in \Delta'_L$, $\tau_i^s = lca(\tau_m^s, \tau_n^s)$ *and* $\tau_i^t = lca(\tau_m^t, \tau_n^t)$ *are true.*

In Fig. 2, τ_i^s is the lowest common ancestor (LCA) of τ_m^s and τ_n^s, and τ_i^t is the LCA of τ_m^t and τ_n^t. Theorem 3.2 constitutes the basis for the dynamic programming (DP) in our phrase alignment algorithm (Sec. 4.2).

4 Modeling of Phrase Alignment

We formally model the phrase alignment process as illustrated in Fig. 3, where $\mathbb{h}_i$ is aligned from descendant alignments, *i.e.*, $\mathbb{h}_m$ and $\mathbb{h}_n$.

4.1 Probabilistic Model

Similar to the probabilistic context free grammar (PCFG), the inside probability α_i of $\mathbb{h}_i$ is determined by the inside probabilities, α_m and α_n, of the support pairs, together with the probability of the rule, *i.e.*, the way by which $\mathbb{h}_m$ and $\mathbb{h}_n$ are combined to support $\mathbb{h}_i$ as shown in Fig. 3. It is characterized by four paths, $\pi_{m,i}^s$ (the path from τ_m^s to τ_i^s), $\pi_{n,i}^s$ (τ_n^s to τ_i^s), $\pi_{m,i}^t$ (τ_m^t to τ_i^t), and $\pi_{n,i}^t$ (τ_n^t to τ_i^t).

Each path consists of a set of null-aligned phrases $\phi \in \langle \phi, \tau_\emptyset \rangle$ and their mothers, *e.g.*, the path $\pi_{m,i}^s$ in Fig. 3 is a set of $\langle \phi_1^s, m(\phi_1^s) \rangle$, $\langle \phi_2^s, m(\phi_2^s) \rangle$, and $\langle \phi_3^s, m(\phi_3^s) \rangle$. We assume that each occurrence of a null-alignment is independent. Thus, its probability $\beta_{m,i}^s$ is computed as:

$$\beta_{m,i}^s = \Pi_{\phi_k^s \in \pi_{m,i}^s} P_r(\phi_k^s, \tau_\emptyset).$$

$\beta_{n,i}^s$, $\beta_{m,i}^t$, and $\beta_{n,i}^t$ are computed in the same manner. We abbreviate $\gamma_{m,n,i}^s = \beta_{m,i}^s \beta_{n,i}^s$, likewise $\gamma_{m,n,i}^t = \beta_{m,i}^t \beta_{n,i}^t$. Finally, α_i can be represented as a simple relation:

$$\alpha_i = \alpha_m \alpha_n P_r(\tau_i^s, \tau_i^t) \gamma_{m,n,i}^s \gamma_{m,n,i}^t. \qquad (1)$$

$P_r(\cdot,\cdot)$ is the alignment probability parameterized in Sec. 5. Since we assume that the structures of parse trees of s and t are determined by a parser, the values of $\gamma_{m,n,i}^s$ and $\gamma_{m,n,i}^t$ are fixed. Therefore, by traversing the parse tree in a bottom-up manner, we can identify an LCA (*i.e.*, τ_i) for phrases τ_m and τ_n while simultaneously computing $\gamma_{m,n,i}$.

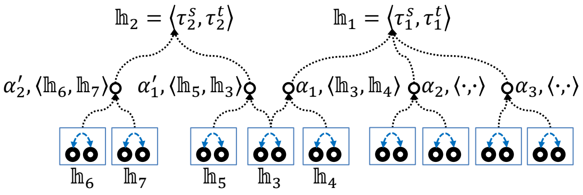

$\mathbb{h}_2 = \langle \tau_2^s, \tau_2^t \rangle$
$\mathbb{h}_1 = \langle \tau_1^s, \tau_1^t \rangle$
$\alpha'_2, \langle \mathbb{h}_6, \mathbb{h}_7 \rangle$
$\alpha'_1, \langle \mathbb{h}_5, \mathbb{h}_3 \rangle$
$\alpha_1, \langle \mathbb{h}_3, \mathbb{h}_4 \rangle$
$\alpha_2, \langle \cdot,\cdot \rangle$
$\alpha_3, \langle \cdot,\cdot \rangle$
$\mathbb{h}_6$
$\mathbb{h}_7$
$\mathbb{h}_5$
$\mathbb{h}_3$
$\mathbb{h}_4$

Figure 4: Alignment pairs and packed supports

4.2 Alignment Algorithm

Algorithm 4.1 depicts our algorithm. Given word alignments $\mathbb{W} = \{\langle w_i^s, w_i^t \rangle\}$, it constructs legitimate sets of aligned pairs in a bottom-up manner. Like the CKY algorithm, Algorithm 4.1 uses DP to efficiently compute all possible legitimate sets and their probabilities in parallel. In addition, null-alignments are allowed when aligning an LCA supported by aligned descendant nodes.

$A[\cdot]$ is indexed by phrases in the parse tree of s and maintains a list of all possible aligned pairs. Furthermore, to deal with non-monotonic alignment (Sec. 4.3), it keeps all competing hypotheses of support relations using packed representations. Specifically, $\mathbb{h}_i$ is accompanied by its packed support list as illustrated in Fig. 4; $\mathbb{h}_1 = \langle \tau_1^s, \tau_1^t \rangle$ is aligned with supports of $\{\langle \alpha_j, \langle \mathbb{h}_m, \mathbb{h}_n \rangle \rangle\}$ like $\langle \alpha_1, \langle \mathbb{h}_3, \mathbb{h}_4 \rangle \rangle$. Depending on the support alignments, $\mathbb{h}_i$ has different inside probabilities, *i.e.*, α_1, α_2, and α_3. Since the succeeding process of alignment only deals with the LCA's of τ_1^s and τ_1^t that are independent of the support alignment, all

[1] $\Rightarrow$ and $\overset{R}{\Rightarrow}$ are not distinguished here.

Algorithm 4.1 Phrase Alignment

```
LCAs and $\gamma$ in parse trees of $s$ and $t$ are computed and stored in $Lca^s[\cdot][\cdot]$ and $Lca^t[\cdot][\cdot]$.
set $A[\tau^s] \leftarrow \emptyset$ for all $\tau^s$
for all $\langle w^s, w^t \rangle \in \mathbb{W}$ do
    Find $\tau^s$ and $\tau^t$ covering $w^s$ and $w^t$
    Compute $\alpha_i$ of $\langle \tau^s, \tau^t \rangle$ using Eq. (1)
    PACK($\langle \tau^s, \tau^t \rangle, \langle \alpha_i, \emptyset \rangle, A$)
for all $\tau^s_m, \tau^s_n$ do   ▷ Trace the source tree from the bottom to top
    for all $\langle \tau^s_i, \gamma^s_{m,n,i} \rangle \in Lca^s[\tau^s_m][\tau^s_n]$ do
        ALIGN($\tau^s_m, \tau^s_n, \tau^s_i, \gamma^s_{m,n,i}, A$)
function ALIGN($\tau^s_m, \tau^s_n, \tau^s_i, \gamma^s, A$)
    for all $\mathbb{h}_m = \langle \tau^s_m, \tau^t_m \rangle \in A[\tau^s_m]$ do
        for all $\mathbb{h}_n = \langle \tau^s_n, \tau^t_n \rangle \in A[\tau^s_n]$ do
            $\langle \tau^t_i, \gamma^t \rangle \leftarrow Lca^t[\tau^t_m][\tau^t_n]$
            Compute $\alpha_i$ using Eq. (1)
            PACK($\langle \tau^s_i, \tau^t_i \rangle, \langle \alpha_i, \langle \mathbb{h}_m, \mathbb{h}_n \rangle \rangle, A$)
function PACK($\langle \tau^s, \tau^t \rangle, \langle \alpha, \langle \mathbb{h}_m, \mathbb{h}_n \rangle \rangle, A$)
    if $\langle \tau^s, \tau^t \rangle \in A[\tau^s]$ then
        $A[\tau^s] \leftarrow A[\tau^s] \cup \langle \alpha, \langle \mathbb{h}_m, \mathbb{h}_n \rangle \rangle$   ▷ Merge supports and their inside probability
    else
        $A[\tau^s] \leftarrow (\langle \tau^s, \tau^t \rangle, \langle \alpha, \langle \mathbb{h}_m, \mathbb{h}_n \rangle \rangle)$
```

support relations are packed as a support list[2] by the PACK function.

4.3 Non-Compositional Alignment

A monotonic alignment requires $\tau^t_m \in \mathbb{h}_m$ and $\tau^t_n \in \mathbb{h}_n$ to have an LCA, which adheres to the compositionality in language. However, previous studies declared that the compositionality is violated in a monolingual phrase alignment (Burkett et al., 2010; Weese et al., 2014). Heilman and Smith (2010) discuss complex phrase reordering is prevalent in paraphrases and entailed text.

A non-monotonic alignment occurs when corresponding phrases have largely different orders, *i.e.*, one of them (*e.g.*, τ^t_m) is an ancestor of another (*e.g.*, τ^t_n) or the same phrase. Such a case could be exceptionally compatible, when τ^t_m has null-alignments and all the aligned phrases of τ^t_n fit in these null-alignments. A new alignment $\langle \tau^s_i, \tau^t_i (= \tau^t_m) \rangle$ would be non-monotonically formed. Fig. 5 shows a real example of non-compositional alignment produced by our method. The target phrase τ^t_n ("through the spirit of teamwork") is null-alignment when aligning τ^s_m and τ^t_m, but then the alignment to τ^s_n ("Relying on team spirit") is allowed by non-compositional alignment of τ^s_i.

Algorithm 4.2 Non-Compositional Alignment

```
1:  function TRACE($\tau_n, \tau_m$)   ▷ $\tau_n \in ds(\tau_m)$
2:      $V \leftarrow \emptyset$
3:      for all $[\tau_m]^i$ do
4:          if $\tau_n \in ds(\phi)$ for $\exists \phi \in \Phi^{[\tau_m]^i}$ then
5:              $V \leftarrow V \cup \langle \Psi^{[\tau_m]^i} \cup \tau_n, (\Phi^{[\tau_m]^i} \setminus \phi) \cup$ GAP($\tau_n, \phi$) $\rangle$
6:          else if $\tau_n \in ds(\psi)$ for $\exists \psi \in \Psi^{[\tau_m]^i}$ then
7:              $V \leftarrow V \cup$ TRACE($\tau_n, \psi$)
8:          else
9:              for all $[\tau_n]^j$ do
10:                 $V \leftarrow V \cup$ DOWN($[\tau_n]^j, [\tau_m]^i$)
11:     return $V$;
```

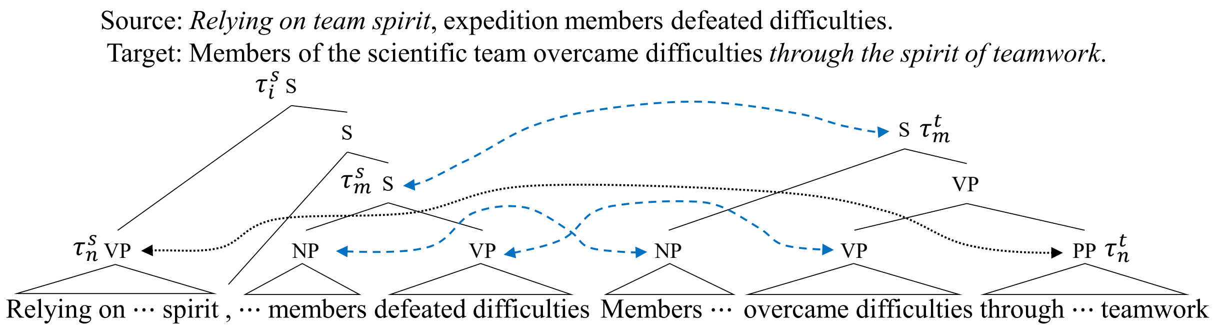
Unlike monotonous alignment, we have to verify whether the internal structures of τ^t_m and τ^t_n are compatible. Since the internal structures of τ^t_m and τ^t_n depend on their supporting alignments, their packed representations in A have to be unpacked, and each pair of supporting alignments for $\mathbb{h}_m$ and $\mathbb{h}_n$ must be checked to confirm compatibility. Furthermore, since the aligned phrases inside τ^t_m and τ^t_n have their own null-alignments, we need to unpack deeper supporting alignments as well.

Algorithm 4.2 checks if target phrases τ_m and $\tau_n \in ds(\tau_m)$ are compatible. We use the following notations: $[\tau_m]^i$ and $[\tau_n]^j$ represent the phrases of τ_m and τ_n with the i-th and j-th sets of supporting alignments, respectively. For τ^t_2 in Fig. 4, there are $[\tau^t_2]^1$ supported by $\langle \mathbb{h}_5, \mathbb{h}_3 \rangle$ and $[\tau^t_2]^2$ supported by $\langle \mathbb{h}_6, \mathbb{h}_7 \rangle$. $[\tau_m]^i$ consists of sets of aligned target phrases $\Psi^{[\tau_m]^i} = \{\psi_k^{[\tau_m]^i}\}$ and null-alignments $\Phi^{[\tau_m]^i} = \{\phi_l^{[\tau_m]^i}\}$ ($[\tau_n]^j$ is similar).

For each $[\tau_m]^i$, if τ_n fits in its null-alignment like in Fig. 5, the alignment information is updated at line 5, where GAP function takes two phrases and returns a set of null-alignments on a path between them. If τ_n is a descendant of a support of τ_m, the compatibility is recursively checked (line 7). Otherwise, the compatibility of the supports of τ_n and τ_m are recursively checked in DOWN function in a similar manner (line 10).

When TRACE function returns a set of $\{\langle \Psi^k, \Phi^k \rangle\}$, all $\psi \in \Psi^k$ are aligned with phrases in the source and their inside probabilities are stored in A. Thus we can compute the inside probability for each $\langle \Psi^k, \Phi^k \rangle$, which is stored in A to-

[2] This is true except for a non-compositional alignment where the packed representation must be unpacked.



Source: *Relying on team spirit*, expedition members defeated difficulties.

Target: Members of the scientific team overcame difficulties *through the spirit of teamwork*.

τ_i^s S
S
τ_m^s S
S τ_m^t
VP
τ_n^s VP
NP
VP
NP
VP
PP τ_n^t
Relying on ⋯ spirit , ⋯ members defeated difficulties Members ⋯ overcame difficulties through ⋯ teamwork

Figure 5: Example of a non-compositional alignment

gether with a new alignment pair $\langle \tau_i^s, \tau_i^t \rangle$ where $\tau_i^s = lca(\tau_m^s, \tau_n^s)$ and $\tau_i^t = \tau_m^t$.

4.4 Forest Alignment

Although we have discussed using trees for clarity, the alignment is conducted on forests. The alignment process is basically the same. The only difference is that the same pair has multiple LCAs. Hence, we need to verify if the sub-trees can be on the same tree when identifying their LCAs since multiple nodes may cover the same span with different derivations. This is critical for non-compositional alignment because whether the internal structures are on the same tree must be confirmed while unpacking them.

Our alignment process corresponds to reranking of forests and may derive a different tree from the 1-best, which may resolve ambiguity in parsing. We use a parser trained beforehand because joint parsing and alignment is computationally too expensive.

5 Parameterization

Next, we parameterize the alignment probability.

5.1 Feature-enhanced EM Algorithm

We apply the feature-enhanced EM (Berg-Kirkpatrick et al., 2010) due to its ability to use dependent features without an irrational independence assumption. This is preferable because the attributes of phrases largely depend on each other.

Our method is computationally heavy since it handles forests and involves unpacking in the non-compositional alignment process. Thus, we use Viterbi training (Brown et al., 1993) together with a beam search of size $\mu_b \in \mathbb{N}$ on the feature-enhanced EM. Also, mini-batch training (Liang and Klein, 2009) is applied. Such an approximation for efficiency is common in parallel parsing (Burkett and Klein, 2008; Burkett et al., 2010).

In addition, an alignment supported by distant descendants tends to fail to reach a root-pair alignment. Thus, we restrict the generation gap between a support alignment and its LCA to be less than or equal to $\mu_g \in \mathbb{N}$.

5.2 Features

In feature-enhanced EM, the alignment probability in Eq. (1) is parameterized using features:

$$P_r(\tau_i^s, \tau_i^t) \doteq \frac{\exp(\mathbf{w} \cdot \mathbb{F}(\boldsymbol{a}_i^s, \boldsymbol{a}_i^t))}{\sum_{\langle \tau_j^s, \tau_j^t \rangle, \tau_i^s = \tau_j^s} \exp(\mathbf{w} \cdot \mathbb{F}(\boldsymbol{a}_j^s, \boldsymbol{a}_j^t))},$$

where $\boldsymbol{a} \doteq (a_0, \cdots, a_n)$ consists of n attributes of τ. $\mathbb{F}(\cdot, \cdot)$ and $\mathbf{w}$ are vectors of feature functions and their weights, respectively.

In a parse tree, the head of a phrase determines its property. Hence, a lemmatized lexical head $a_{\text{lex}} \in \boldsymbol{a}$ combined with its syntactic category $a_{\text{cat}} \in \boldsymbol{a}$ is encoded as a feature[3] as shown below. We use semantic (instead of syntactic) heads to encode semantic relationships in paraphrases.

1: $\mathbb{1}(a_{\text{lex}}^s = \cdot, a_{\text{cat}}^s = \cdot, a_{\text{lex}}^t = \cdot, a_{\text{cat}}^t = \cdot)$
2: $\mathbb{1}(\text{SurfaceSim}(a_{\text{lex}}^s = \cdot, a_{\text{lex}}^t = \cdot))$
3: $\mathbb{1}(\text{WordnetSim}(a_{\text{lex}}^s = \cdot, a_{\text{lex}}^t = \cdot))$
4: $\mathbb{1}(\text{EmbeddingSim}(a_{\text{lex}}^s = \cdot, a_{\text{lex}}^t = \cdot))$
5: $\mathbb{1}(\text{IsPrepositionPair}(a_{\text{lex}}^s = \cdot, a_{\text{lex}}^t = \cdot))$
6: $\mathbb{1}(a_{\text{cat}}^s = \cdot, a_{\text{cat}}^t = \cdot)$
7: $\mathbb{1}(\text{IsSameCategory}(a_{\text{cat}}^s = \cdot, a_{\text{cat}}^t = \cdot))$

The first feature is an indicator invoked only at specific values. On the other hand, the rest of the

[3] We also tried features based on the configurations of the source and target sub-trees similar to (Das and Smith, 2009) as well as features based on the spans of null-alignments. However, none of them contributed to alignment quality.

features are invoked across multiple values, allowing general patterns to be learned. The second feature is invoked if two heads are identical or a head is a substring of another. The third feature is invoked if two heads are synonyms or derivations that are extracted from the WordNet[4]. The fourth feature is invoked if the cosine similarity between word embeddings of two heads is larger than a threshold. The fifth feature is invoked when the heads are both prepositions to capture their different natures from the content words. The last two features are for categories; the sixth one is invoked at each category pair, while the seventh feature is invoked if the input categories are the same.

To avoid generating a huge number of features, we reduce the number of syntactic categories; for contents (N, V, ADJ, and ADV), prepositions, coordinations, null (*i.e.*, for $\tau_\emptyset$), and others.

5.3 Penalty Function

Since our method allows null-alignments, it has a degenerate maximum likelihood solution (Liang and Klein, 2009) that makes every phrase null-alignment. Similarly, a degenerate solution overly conducts non-compositional alignment.

To avoid these issues, a penalty is incorporated:

$$P_e(\tau_i^s, \tau_i^t) = \begin{cases} \exp\{-(|\tau_i^s|_\phi + |\tau_i^t|_\phi + \mu_c + 1)^{\mu_n}\} \\ \qquad \text{(non-compositional alignment)} \\ \exp\{-(|\tau_i^s|_\phi + |\tau_i^t|_\phi + 1)^{\mu_n}\} \\ \qquad \text{(otherwise)} \end{cases}$$

where $|\cdot|_\phi$ computes the span of internal null-alignments, and $\mu_n \geq 1.0$ and $\mu_c \in \mathbb{R}_+$ control the strength of the penalties of the null-alignment and the non-compositional alignment, respectively. The penalty function is multiplied by Eq. (1) as a *soft-constraint* for re-ranking alignment pairs in Algorithm 4.1.

5.4 Combination with Parse Probability

Following the spirit of parallel parsing that simultaneously parses and aligns sentences, we linearly interpolate the alignment probability with the parsing probability once the parameters are tuned by EM. When aligning a node pair $\langle \tau_i^s, \tau_i^t \rangle$, the overall probability is computed as:

$$(1 - \mu_p)\alpha_i + \mu_p \varrho(\tau_i^s)\varrho(\tau_i^t),$$

where $\varrho(\cdot)$ gives the marginal probability in parsing and $\mu_p \in [0, 1]$ balances these probabilities.

6 Evaluation

As discussed in Sec. 2, previous studies have not conducted syntactic phrase alignment on parse trees. A direct metric does not exist to compare paraphrases that cover different spans, *i.e.*, our syntactic paraphrases and paraphrases of n-grams. Thus, we compared the alignment quality to that of humans as a realistic way to evaluate the performance of our method.

We also evaluated the parsing quality. Similar to the alignment quality, differences in phrase structures disturb the comparisons (Sagae et al., 2008). Our method applies an HPSG parser Enju (Miyao and Tsujii, 2008) to derive parse forests due to its state-of-the-art performance and ability to provide rich properties of phrases. Hence, we compared our parsing quality to the 1-best parses of Enju.

6.1 Language Resources

We used reference translations to evaluate machine translations[5] as sentential paraphrases (Weese et al., 2014). The reference translations of 10 to 30 words were extracted and paired, giving $41K$ pairs as a training corpus.

We use different kinds of dictionaries to obtain word alignments $\mathbb{W}$ as well as to compute feature functions. First, we extract synonyms and words with derivational relationship using WordNet. Then we handcraft derivation rules (*e.g.*, *create*, *creation*, *creator*) and extract potentially derivational words from the training corpus. Finally, we use prepositions defined in (Srikumar and Roth, 2013) as a preposition dictionary to compute the feature function.

In addition, we extend $\mathbb{W}$ using word embeddings; we use the MVLSA word embeddings (Rastogi et al., 2015) given the superior performance in word similarity tasks. Specifically, we compute the cosine similarity of embeddings; words with a higher similarity value than a threshold are determined as similar words. The threshold is empirically set as the 100th highest similarity value between words in the training corpus.

6.2 Gold-Standard Data

Since no annotated corpus provides phrase alignments on parse trees, we created one through two-phase manual annotation. First, a linguistic expert with rich experience on annotating HPSG trees

[4] http://wordnet.princeton.edu

[5] NIST OpenMT corpora: LDC2010T14, LDC2010T17, LDC2010T21, LDC2010T23, LDC2013T03

annotated gold-trees to paraphrasal sentence pairs sampled from the training corpus. To diversify the data, only one reference pair per sentence of a source language was annotated. Consequently, 201 paraphrased pairs with gold-trees (containing $20,678$ phrases) were obtained.

Next, three professional English translators identified paraphrased pairs including null-alignments given sets of phrases extracted from the gold-trees. These annotators independently annotated the same set, yielding $14,356$ phrase alignments where at least one annotator regarded as a paraphrase. All the annotators agreed that 77% of the phrases were paraphrases.

We used 50 sentence pairs for development and another 151 for testing. These pairs were excluded from the training corpus.

6.3 Evaluation Metric

Alignment Quality Alignment quality was evaluated by measuring the extent that the automatic alignment results agree with those of humans. Specifically, we evaluated how gold-alignments can be replicated by automatic alignment (called recall) and how automatic alignments overlap with alignments that at least an annotator aligned (called precision) as:

$$\text{Recall} = \frac{|\{\mathbb{h}|\mathbb{h} \in \mathbb{H}_a \wedge \mathbb{h} \in \mathbb{G} \cap \mathbb{G}'\}|}{|\mathbb{G} \cap \mathbb{G}'|},$$

$$\text{Precision} = \frac{|\{\mathbb{h}|\mathbb{h} \in \mathbb{H}a \wedge \mathbb{h} \in \mathbb{G} \cup \mathbb{G}'\}|}{|\mathbb{H}a|},$$

where $\mathbb{H}a$ is a set of alignments, while $\mathbb{G}$ and $\mathbb{G}'$ are the ones that two of annotators produce, respectively. The function of $|\cdot|$ counts the elements in a set. There are three combinations for $\mathbb{G}$ and $\mathbb{G}'$ because we had three annotators. The final precision and recall values are their averages.

Parsing Quality The parsing quality was evaluated using the CONLL-X (Buchholz and Marsi, 2006) standard. Dependencies were extracted from the output HPSG trees, and evaluated using the official script[6]. Due to this conversion, the accuracy on the relation labels is less important. Thus, we reported only the unlabeled attachment score (UAS)[7]. The development and test sets provide $2,371$ and $6,957$ dependencies, respectively.

	Roles of hyper-parameters
μ_n	Control penalty for null-alignment
μ_c	Control penalty for non-compositional alignment
μ_p	Balance alignment and parsing prob.
μ_b	Beam size at alignment
μ_g	Generation gap to reach an LCA

Table 2: Summary of the hyper-parameters

Method	Recall	Prec.	UAS	%
Human	90.65	88.21	–	–
Proposed	**83.64**	**78.91**	93.49	98
Monotonic	82.86*	77.97*	93.49	98
w/o EM	81.33*	75.09*	92.91*	86
1-best tree	80.11*	73.26*	93.56	100

Table 3: Evaluation results on the test set, where * represents p-value < 0.05 against our method.

Since all metrics were computed in a set, the approximate randomization (Noreen, 1989; Riezler and Maxwell, 2005) ($B = 10K$) was used for significance testing. It has been shown to be more conservative than using bootstrap resampling (Riezler and Maxwell, 2005).

6.4 Results and Discussion

Overall Results Table 2 summarizes the hyper-parameters, which were tuned to maximize UAS in the development set using the Bayesian optimization. For efficiency, we used $2K$ samples from the training corpus and set the mini-batch size in feature-enhanced EM to 200 similar to “rapid training” in (Burkett and Klein, 2008). We also set $\mu_b = 50$ during EM training to manage the training time.

Table 3 shows the performance on the test set for variations of our method and that of the human annotators. The last column shows the percentage of pairs where a root pair is reached to be aligned, called reachability. Our method is denoted as *Proposed*, while its variations include a method with only monotonic alignment (*monotonic*), without EM (*w/o EM*), and a method aligning only 1-best trees (1-*best tree*).

The performance of the human annotators was assessed by considering one annotator as the test and the other two as the gold-standard, and then taking the averages, which is the same setting as our method. We regard this as the pseudo inter-

[6] http://ilk.uvt.nl/conll/software.html

[7] Although omitted, the labeled attachment score showed the same tendency as UAS.

annotator agreement, since the conventional inter-annotator agreement is not directly applicable due to variations in aligned phrases.

Our method significantly outperforms the others as it achieved the highest recall and precision for alignment quality. Our recall and precision reach 92% and 89% of those of humans, respectively. Non-compositional alignment is shown to contribute to alignment quality, while the feature-enhanced EM is effective for both the alignment and parsing quality. Comparing our method and the one aligning only 1-best trees demonstrates that the alignment of parse *forests* largely contributes to the alignment quality. Although we confirmed that aligning larger forests slightly improved recall and precision, the improvements were statistically insignificant. The parsing quality was not much affected by phrase alignment, which is further investigated in the following.

Finally, our method achieved 98% reachability, where 2% of unreachable cases were due to the beam search. While understanding that the reachability depends on experimental data, ours is notably higher than that of SCFG, reported as 9.1% in (Weese et al., 2014). These results show the ability of our method to accurately align paraphrases with divergent phrase correspondences.

Effect of Mini-Batch Size We investigated the effect of the mini-batch size in EM training using the entire training corpus ($41K$ pairs). When increasing the mini-batch size from 200 to $2K$, recall, precision, and UAS values are fairly stable. In addition, they are insensitive against the amount of training corpus, showing the comparable values against the model trained on $2K$ samples. These results demonstrate that our method can be trained with a moderate amount of data.

Observations Previous studies show that parallel parsing improves parsing quality, while such an effect is insignificant here. We examine causes through manual observations.

The evaluation script indicated that our method corrected 34 errors while introducing 41 new errors[8]. We further analyzed these 75 cases; 12 cases are ambiguous as both the gold-standard and the output are correct. In addition, 8 cases are due to erroneous original sentences that should be disregarded, *e.g.*, " For two weeks ago,..." and "According to the source, will also meet...". Consequently, our method corrected 32 errors while introducing 23 errors in reality for 446 errors in 1-best trees, which achieves a 2.5% error reduction.

These are promising results for our method to improve parsing quality, especially on the PP-attachment (159 errors in 1-best), which contained 14 of the 32 corrected errors. Fig. 1 shows a real example; the phrase of "for a smoke" in the source was mistakenly attached to "ground floor" in the 1-best tree. This error was corrected as depicted.

Duan et al. (2016) showed that paraphrases artificially generated using n-best parses improved the parsing quality. One reason for limited improvement in our experiments may be because structural changes in our natural paraphrases are more dynamic than the level useful to resolve ambiguities. We will further investigate this in future.

7 Conclusion

We propose an efficient method for phrase alignment on parse forests of paraphrased sentences. To increase the amount of collected paraphrases, we plan to extend our method to align comparable paraphrases that are partially paraphrasal sentences. In addition, we will apply our method to parallel parsing and other grammar, *e.g.*, projective dependency trees. Furthermore, we will apply such syntactic paraphrases to phrase embedding.

Acknowledgments

We thank Professor Issei Sato for permission to use their package for Bayesian optimization. Special thanks also to Dr. Yuka Tateishi for her contribution to HPSG tree annotation. Advice and comments given by Professor Takuya Matsuzaki and Professor Yusuke Miyao have been a great help in applying Enju parser for this project. We appreciate the anonymous reviewers for their insightful comments and suggestions to improve the paper. This project is funded by Microsoft Research Asia and the Kayamori Foundation of Informational Science Advancement.

Supplemental Material

The supplemental material is available at our web site[9] that provides proofs of the theorems, pseudo-codes of the algorithms, and more experiment results with examples.

[8] Alignments were obtained by the model trained using the entire corpus with the $1K$ mini-batch size.

[9] http://www-bigdata.ist.osaka-u.ac.jp/arase/pj/phrase-alignment/

References

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 582–590, Los Angeles, California.

Or Biran, Terra Blevins, and Kathleen McKeown. 2016. Mining paraphrasal typed templates from a plain text corpus. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1913–1923, Berlin, Germany.

Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceesings of the Conference on Natural Language Learning (CoNLL)*, pages 149–164, New York City.

David Burkett, John Blitzer, and Dan Klein. 2010. Joint parsing and alignment with weakly synchronized grammars. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 127–135, Los Angeles, California.

David Burkett and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proceesings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 877–886, Honolulu, Hawaii.

Gideon Maillette de Buy Wenniger and Khalil Sima'an. 2013. A formal characterization of parsing word alignments by synchronous grammars with empirical evidence to the ITG hypothesis. In *Proceedings of the Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST)*, pages 58–67, Atlanta, Georgia.

Do Kook Choe and David McClosky. 2015. Parsing paraphrases with joint inference. In *Proceedings of the Joint Conference of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 1223–1233, Beijing, China.

John Cocke. 1969. *Programming Languages and Their Compilers: Preliminary Notes*. Courant Institute of Mathematical Sciences, New York University.

Trevor Cohn, Chris Callison-Burch, and Mirella Lapata. 2008. Constructing corpora for the development and evaluation of paraphrase systems. *Computational Linguistics*, 34(4):597–614.

Dipanjan Das and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 468–476, Suntec, Singapore.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceesings of the International Conference on Computational Linguistics (COLING)*, pages 350–356, Geneva, Switzerland.

Manjuan Duan, Ethan Hill, and Michael White. 2016. Generating disambiguating paraphrases for structurally ambiguous sentences. In *Proceedings of the Linguistic Annotation Workshop (LAW)*, pages 160–170, Berlin, Germany.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 758–764, Atlanta, Georgia.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1011–1019, Los Angeles, California.

Miriam Kaeshammer. 2013. Synchronous linear context-free rewriting systems for machine translation. In *Proceedings of the Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST)*, pages 68–77, Atlanta, Georgia.

Tadao Kasami. 1965. An efficient recognition and syntax-analysis algorithm for context-free languages. Scientific report AFCRL-65-758, Air Force Cambridge Research Lab.

Percy Liang and Dan Klein. 2009. Online EM for unsupervised models. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 611–619, Boulder, Colorado.

Bill MacCartney, Michel Galley, and Christopher D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceesings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 802–811, Honolulu, Hawaii.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1):35–80.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley.

Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. 2015. Multiview LSA: Representation learning via generalized CCA. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 556–566, Denver, Colorado.

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64, Ann Arbor, Michigan.

Kenji Sagae, Yusuke Miyao, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Challenges in mapping of syntactic representations for framework-independent parser evaluation. In *Proceedings of the Workshop on Automated Syntatic Annotations for Interoperable Language Resources*.

David Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the Workshop on Statistical Machine Translation (WMT)*, pages 23–30, New York City.

David A. Smith and Noah A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proceesings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 49–56, Barcelona, Spain.

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

Vivek Srikumar and Dan Roth. 2013. Modeling semantic relations expressed by prepositions. *Transactions of the Association of Computational Linguistics (TACL)*, 1:231–242.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. pages 1556–1566, Beijing, China.

Kapil Thadani, Scott Martin, and Michael White. 2012. A joint phrasal and dependency model for paraphrase alignment. In *Proceesings of the International Conference on Computational Linguistics (COLING)*, pages 1229–1238, Mumbai, India.

Jonathan Weese, Juri Ganitkevitch, and Chris Callison-Burch. 2014. PARADIGM: Paraphrase diagnostics through grammar matching. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 192–201, Gothenburg, Sweden.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association of Computational Linguistics (TACL)*, 3(1):345–358.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *Proceesings of the International Conference on Learning Representations (ICLR)*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Semi-Markov phrase-based monolingual alignment. In *Proceesings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 590–600, Seattle, Washington, USA.

Wenpeng Yin and Hinrich Schütze. 2015. Discriminative phrase embedding for paraphrase identification. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1368–1373, Denver, Colorado.

Daniel H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Fast(er) Exact Decoding and Global Training for Transition-Based Dependency Parsing via a Minimal Feature Set

Tianze Shi
Cornell University
tianze@cs.cornell.edu

Liang Huang
Oregon State University
liang.huang.sh@gmail.com

Lillian Lee
Cornell University
llee@cs.cornell.edu

Abstract

We first present a minimal feature set for transition-based dependency parsing, continuing a recent trend started by Kiperwasser and Goldberg (2016a) and Cross and Huang (2016a) of using bi-directional LSTM features. We plug our minimal feature set into the dynamic-programming framework of Huang and Sagae (2010) and Kuhlmann et al. (2011) to produce the first implementation of worst-case $O(n^3)$ exact decoders for arc-hybrid and arc-eager transition systems. With our minimal features, we also present $O(n^3)$ global training methods. Finally, using ensembles including our new parsers, we achieve the best unlabeled attachment score reported (to our knowledge) on the Chinese Treebank and the "second-best-in-class" result on the English Penn Treebank.

1 Introduction

It used to be the case that the most accurate dependency parsers made global decisions and employed exact decoding. But transition-based dependency parsers (TBDPs) have recently achieved state-of-the-art performance, despite the fact that for efficiency reasons, they are usually trained to make local, rather than global, decisions and the decoding process is done approximately, rather than exactly (Weiss et al., 2015; Dyer et al., 2015; Andor et al., 2016). The key efficiency issue for decoding is as follows. In order to make accurate (local) attachment decisions, historically, TBDPs have required a large set of features in order to access rich information about particular *positions* in the stack and buffer of the current parser configuration. But consulting many positions means that although polynomial-time exact-decoding algorithms do exist, having been introduced by Huang and Sagae (2010) and Kuhlmann et al. (2011), unfortunately, they are prohibitively costly in practice, since the number of positions considered can factor into the exponent of the running time. For instance, Huang and Sagae employ a fairly reduced set of nine positions, but the worst-case running time for the exact-decoding version of their algorithm is $O(n^6)$ (originally reported as $O(n^7)$) for a length-n sentence. As an extreme case, Dyer et al. (2015) use an LSTM to summarize *arbitrary* information on the stack, which completely rules out dynamic programming.

Recently, Kiperwasser and Goldberg (2016a) and Cross and Huang (2016a) applied bi-directional long short-term memory networks (Graves and Schmidhuber, 2005, bi-LSTMs) to derive feature representations for parsing, because these networks capture wide-window contextual information well. Collectively, these two sets of authors demonstrated that with bi-LSTMs, *four* positional features suffice for the arc-hybrid parsing system (K&G), and *three* suffice for arc-standard (C&H).[1]

Inspired by their work, we arrive at a *minimal* feature set for arc-hybrid and arc-eager: it contains *only two* positional bi-LSTM vectors, suffers almost no loss in performance in comparison to larger sets, and out-performs a single position. (Details regarding the situation with arc-standard can be found in §2.)

Our minimal feature set plugs into Huang and Sagae's and Kuhlmann et al.'s dynamic program-

[1] We note that K&G were not focused on minimizing positions, although they explicitly noted the implications of doing so: "While not explored in this work, [fewer positions] results in very compact state signatures, [which is] very appealing for use in transition-based parsers that employ dynamic-programming search" (pg. 319). C&H also noted in their follow-up (Cross and Huang, 2016b) the possibility of future work using dynamic programming thanks to simple features.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 12–23
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

ming framework to produce the first implementation of $O(n^3)$ *exact* decoders for arc-hybrid and arc-eager parsers. We also enable and implement $O(n^3)$ *global* training methods. Empirically, *ensembles* containing our minimal-feature, globally-trained and exactly-decoded models produce the best unlabeled attachment score (UAS) reported (to our knowledge) on the Chinese Treebank and the "second-best-in-class" result on the English Penn Treebank.[2]

Additionally, we provide a slight update to the theoretical connections previously drawn by Gómez-Rodríguez, Carroll, and Weir (2008, 2011) between TBDPs and the *graph-based* dependency parsing algorithms of Eisner (1996) and Eisner and Satta (1999), including results regarding the arc-eager parsing system.

2 A Minimal Feature Set

TBDPs incrementally process a sentence by making transitions through search states representing parser configurations. Three of the main transition systems in use today (formal introduction in §3.1) all maintain the following two data structures in their configurations: (1) a stack of partially parsed subtrees and (2) a buffer (mostly) of unprocessed sentence tokens.

To featurize configurations for use in a scoring function, it is common to have features that extract information about the first several elements on the stack and the buffer, such as their word forms and part-of-speech (POS) tags. We refer to these as *positional features*, as each feature relates to a particular position in the stack or buffer. Typically, millions of sparse indicator features (often developed via manual engineering) are used.

In contrast, Chen and Manning (2014) introduce a feature set consisting of *dense* word-, POS-, and dependency-label embeddings. While dense, these features are for the same 18 positions that have been typically used in prior work. Recently, Kiperwasser and Goldberg (2016a) and Cross and Huang (2016a) adopt bi-directional LSTMs, which have nice expressiveness and context-sensitivity properties, to reduce the number of positions considered down to four and three, for different transition systems, respectively.

Features	Arc-standard	Arc-hybrid	Arc-eager
$\{\overset{\rightarrow\leftarrow}{s}_2, \overset{\rightarrow\leftarrow}{s}_1, \overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$93.95_{\pm 0.12}$	$94.08_{\pm 0.13}$	$93.92_{\pm 0.04}$
$\{\overset{\rightarrow\leftarrow}{s}_1, \overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$94.13_{\pm 0.06}$	$94.08_{\pm 0.05}$	$93.91_{\pm 0.07}$
$\{\overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$54.47_{\pm 0.36}$	$94.03_{\pm 0.12}$	$93.92_{\pm 0.07}$
$\{\overset{\rightarrow\leftarrow}{b}_0\}$	$47.11_{\pm 0.44}$	$52.39_{\pm 0.23}$	$79.15_{\pm 0.06}$

Min positions	Arc-standard	Arc-hybrid	Arc-eager
K&G 2016a	-	4	-
C&H 2016a	3	-	-
our work	3	**2**	**2**

Table 1: *Top:* English PTB dev-set UAS% for progressively smaller sets of positional features, for greedy parsers with different transition systems. The "double-arrow" notation indicates vectors produced by a *bi*-directional LSTM. Internal lines highlight large performance drop-offs when a feature is deleted. *Bottom:* sizes of the minimal feature sets in Kiperwasser and Goldberg (2016a), Cross and Huang (2016a), and our work.

This naturally begs the question, what is the lower limit on the number of positional features necessary for a parser to perform well? Kiperwasser and Goldberg (2016a) reason that for the arc-hybrid system, the first and second items on the stack and the first buffer item — denoted by s_0, s_1, and b_0, respectively — are required; they additionally include the third stack item, s_2, because it may not be adjacent to the others in the original sentence. For arc-standard, Cross and Huang (2016a) argue for the necessity of s_0, s_1, and b_0.

We address the lower-limit question empirically, and find that, surprisingly, *two positions suffice* for the greedy arc-eager and arc-hybrid parsers. We also provide empirical support for Cross and Huang's argument for the necessity of three features for arc-standard. In the rest of this section, we explain our experiments, run only on an English development set, that support this conclusion; the results are depicted in Table 1. We later explore the *implementation implications* in §3-4 and then *test-set* parsing-accuracy in §6.

We employ the same model architecture as Kiperwasser and Goldberg (2016a). Specifically, we first use a bi-LSTM to encode an n-token sentence, treated as a sequence of per-token concatenations of word- and POS-tag embeddings, into a sequence of vectors $[\overset{\rightarrow\leftarrow}{w}_1, \ldots, \overset{\rightarrow\leftarrow}{w}_n]$, where each $\overset{\rightarrow\leftarrow}{w}_i$

[2] Our ideas were subsequently adapted to the *labeled* setting by Shi, Wu, Chen, and Cheng (2017) in their submission to the CoNLL 2017 shared task on Universal Dependencies parsing. Their team achieved the second-highest *labeled* attachment score in general and had the top average performance on the surprise languages.

is the output of the bi-LSTM at time step i. (The double-arrow notation for these vectors emphasizes the bi-directionality of their origin). Then, for a given parser configuration, stack positions are represented by $\overleftrightarrow{s}_j$, defined as $\overleftrightarrow{w}_{i(s_j)}$ where $i(s_j)$ gives the position in the sentence of the token that is the head of the tree in s_j. Similarly, buffer positions are represented by $\overleftrightarrow{b}_j$, defined as $\overleftrightarrow{w}_{i(b_j)}$ for the token at buffer position j. Finally, as in Chen and Manning (2014), we use a multilayer perceptron to score possible transitions from the given configuration, where the input is the concatenation of some selection of the $\overleftrightarrow{s}_j$s and $\overleftrightarrow{b}_k$s. We use greedy decoders, and train the models with dynamic oracles (Goldberg and Nivre, 2013).

Table 1 reports the parsing accuracy that results for feature sets of size four, three, two, and one for three commonly-used transition systems. The data is the development section of the English Penn Treebank (PTB), and experimental settings are as described in our other experimental section, §6. We see that we can go down to three or, in the arc-hybrid and arc-eager transition systems, even two positions with very little loss in performance, but not further. We therefore call $\{\overleftrightarrow{s}_0, \overleftrightarrow{b}_0\}$ our *minimal* feature set with respect to arc-hybrid and arc-eager, and empirically confirm that Cross and Huang's $\{\overleftrightarrow{s}_0, \overleftrightarrow{s}_1, \overleftrightarrow{b}_0\}$ is minimal for arc-standard; see Table 1 for a summary.[3]

3 Dynamic Programming for TBDPs

As stated in the introduction, our minimal feature set from §2 plugs into Huang and Sagae and Kuhlmann et al.'s dynamic programming (DP) framework. To help explain the connection, this section provides an overview of the DP framework. We draw heavily from the presentation of Kuhlmann et al. (2011).

3.1 Three Transition Systems

Transition-based parsing (Nivre, 2008; Kübler et al., 2009) is an incremental parsing framework based on transitions between parser configurations. For a sentence to be parsed, the system starts from a corresponding initial configuration, and attempts to sequentially apply transitions until a configuration corresponding to a full parse is produced. Formally, a transition system is defined as $\mathcal{S} = (C, T, c^s, C_\tau)$, where C is a nonempty set of configurations, each $t \in T : C \rightharpoonup C$ is a transition function between configurations, c^s is an initialization function that maps an input sentence to an initial configuration, and $C_\tau \subseteq C$ is a set of terminal configurations.

All systems we consider share a common tripartite representation for configurations: when we write $c = (\sigma, \beta, A)$ for some $c \in C$, we are referring to a stack σ of partially parsed subtrees; a buffer β of unprocessed tokens and, optionally, at its beginning, a subtree with only left descendants; and a set A of elements (h, m), each of which is an attachment (dependency arc) with head h and modifier m.[4] We write $m \curvearrowleft h$ to indicate that m left-modifies h, and $h \curvearrowright m$ to indicate that m right-modifies h. For a sentence $w = w_1, ..., w_n$, the initial configuration is (σ_0, β_0, A_0), where σ_0 and A_0 are empty and $\beta_0 = [\mathtt{ROOT}|w_1, ..., w_n]$; ROOT is a special node denoting the root of the parse tree[5] (vertical bars are a notational convenience for indicating different parts of the buffer or stack; our convention is to depict the buffer first element leftmost, and to depict the stack first element rightmost). All terminal configurations have an empty buffer and a stack containing only ROOT.

Arc-Standard The arc-standard system (Nivre, 2004) is motivated by bottom-up parsing: each dependent has to be complete before being attached. The three transitions, shift (sh, move a token from the buffer to the stack), right-reduce (re$_\curvearrowright$, reduce and attach a right modifier), and left-reduce (re$_\curvearrowleft$, reduce and attach a left modifier), are defined as:

$$\begin{aligned}
\mathsf{sh}[(\sigma, b_0|\beta, A)] &= (\sigma|b_0, \beta, A)\\
\mathsf{re}_\curvearrowright[(\sigma|s_1|s_0, \beta, A)] &= (\sigma|s_1, \beta, A \cup \{(s_1, s_0)\})\\
\mathsf{re}_\curvearrowleft[(\sigma|s_1|s_0, \beta, A)] &= (\sigma|s_0, \beta, A \cup \{(s_0, s_1)\})
\end{aligned}$$

Arc-Hybrid The arc-hybrid system (Yamada and Matsumoto, 2003; Gómez-Rodríguez et al., 2008; Kuhlmann et al., 2011) has the same definitions of sh and re$_\curvearrowright$ as arc-standard, but forces

[3] We tentatively conjecture that the following might explain the observed phenomena, but stress that we don't currently see a concrete way to test the following hypothesis. With $\{\overleftrightarrow{s}_0, \overleftrightarrow{b}_0\}$, in the arc-standard case, situations can arise where there are multiple possible transitions with missing information. In contrast, in the arc-hybrid case, there is only one possible transition with missing information (namely, re$_\curvearrowright$, introduced in §3.1); perhaps $\overleftrightarrow{s}_1$ is therefore not so crucial for arc-hybrid in practice?

[4] For simplicity, we only present unlabeled parsing here. See Shi et al. (2017) for labeled-parsing results.

[5] Other presentations place ROOT at the end of the buffer or omit it entirely (Ballesteros and Nivre, 2013).

the collection of left modifiers before right modifiers via its b_0-modifier $\mathsf{re}_{\curvearrowleft}$ transition. This contrasts with arc-standard, where the attachment of left and right modifiers can be interleaved on the stack.

$$\mathsf{sh}[(\sigma, b_0|\beta, A)] = (\sigma|b_0, \beta, A)$$
$$\mathsf{re}_{\curvearrowright}[(\sigma|s_1|s_0, \beta, A)] = (\sigma|s_1, \beta, A \cup \{(s_1, s_0)\})$$
$$\mathsf{re}_{\curvearrowleft}[(\sigma|s_0, b_0|\beta, A)] = (\sigma, b_0|\beta, A \cup \{(b_0, s_0)\})$$

Arc-Eager In contrast to the former two systems, the arc-eager system (Nivre, 2003) makes attachments as early as possible — even if a modifier has not yet received all of its own modifiers. This behavior is accomplished by decomposing the right-reduce transition into two independent transitions, one making the attachment (ra) and one reducing the right-attached child (re).

$$\mathsf{sh}[(\sigma, b_0|\beta, A)] = (\sigma|b_0, \beta, A)$$
$$\mathsf{re}_{\curvearrowleft}[(\sigma|s_0, b_0|\beta, A)] = (\sigma, b_0|\beta, A \cup \{(b_0, s_0)\})$$

(precondition: s_0 not attached to any word)

$$\mathsf{ra}[(\sigma|s_0, b_0|\beta, A)] = (\sigma|s_0|b_0, \beta, A \cup \{(s_0, b_0)\})$$
$$\mathsf{re}[(\sigma|s_0, \beta, A)] = (\sigma, \beta, A)$$

(precondition: s_0 has been attached to its head)

3.2 Deduction and Dynamic Programming

Kuhlmann et al. (2011) reformulate the three transition systems just discussed as deduction systems (Pereira and Warren, 1983; Shieber et al., 1995), wherein transitions serve as inference rules; these are given as the lefthand sides of the first three subfigures in Figure 1. For a given $w = w_1, ..., w_n$, assertions take the form $[i, j, k]$ (or, when applicable, a two-index shorthand to be discussed soon), meaning that there exists a sequence of transitions that, starting from a configuration wherein $head(s_0) = w_i$, results in an ending configuration wherein $head(s_0) = w_j$ and $head(b_0) = w_k$. If we define w_0 as ROOT and w_{n+1} as an end-of-sentence marker, then the goal theorem can be stated as $[0, 0, n + 1]$.

For arc-standard, we depict an assertion $[i, h, k]$ as a subtree whose root (head) is the token at h. Assertions of the form $[i, i, k]$ play an important role for arc-hybrid and arc-eager, and we employ the special shorthand $[i, k]$ for them in Figure 1. In that figure, we also graphically depict such situations as two consecutive half-trees with roots w_i and w_k, where all tokens between i and k are already attached. The superscript b in an arc-eager assertion $[i^b, j]$ is an indicator variable for whether w_i has been attached to its head ($b = 1$) or not ($b = 0$) after the transition sequence is applied.

Kuhlmann et al. (2011) show that all three deduction systems can be directly "tabularized" and dynamic programming (DP) can be applied, such that, *ignoring for the moment the issue of incorporating complex features* (we return to this later), time and space needs are low-order polynomial. Specifically, as the two-index shorthand $[i, j]$ suggests, arc-eager and arc-hybrid systems can be implemented to take $O(n^2)$ space and $O(n^3)$ time; the arc-standard system requires $O(n^3)$ space and $O(n^4)$ time (if one applies the so-called hook trick (Eisner and Satta, 1999)).

Since an $O(n^4)$ running time is not sufficiently practical even in the simple-feature case, *in the remainder of this paper we consider only the arc-hybrid and arc-eager systems, not arc-standard.*

4 Practical Optimal Algorithms Enabled By Our Minimal Feature Set

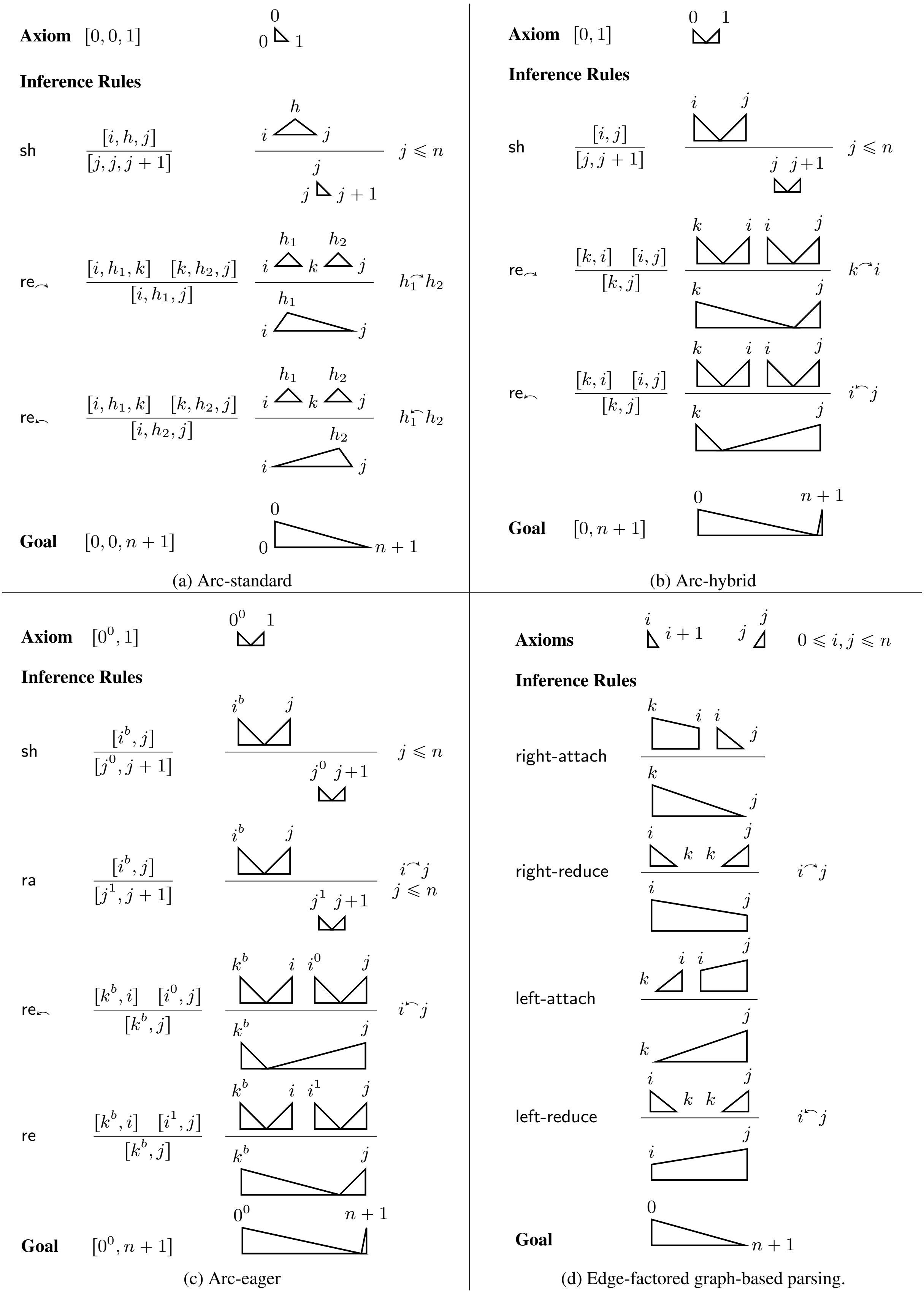
Until now, no one had suggested a set of positional features that was both information-rich enough for accurate parsing *and* small enough to obtain the $O(n^3)$ running-time promised above. Fortunately, our bi-LSTM-based $\{\overleftrightarrow{s}_0, \overleftrightarrow{b}_0\}$ feature set qualifies, and enables the fast optimal procedures described in this section.

4.1 Exact Decoding

Given an input sentence, a TBDP must choose among a potentially exponential number of corresponding transition sequences. We assume access to functions f_t that score individual configurations, where these functions are indexed by the transition functions $t \in T$. For a fixed transition sequence $\mathbf{t} = t_1, t_2, \ldots$, we use c_i to denote the configuration that results after applying t_i.

Typically, for efficiency reasons, greedy left-to-right decoding is employed: the next transition t_i^* out of c_{i-1} is $\arg\max_t f_t(c_{i-1})$, so that past and future decisions are not taken into account. The score $F(\mathbf{t})$ for the transition sequence is induced by summing the relevant $f_{t_i}(c_{i-1})$ values.

However, our use of minimal feature sets enables direct computation of an argmax over the entire space of transition sequences, $\arg\max_{\mathbf{t}} F(\mathbf{t})$, via dynamic programming, because our positions don't rely on any information "outside" the deduction rule indices, thus eliminating the need for ad-


Axiom $[0,0,1]$
Inference Rules
sh $\frac{[i,h,j]}{[j,j,j+1]}$ $j \leqslant n$
re$_{\curvearrowright}$ $\frac{[i,h_1,k] \quad [k,h_2,j]}{[i,h_1,j]}$ $h_1 \curvearrowright h_2$
re$_{\curvearrowleft}$ $\frac{[i,h_1,k] \quad [k,h_2,j]}{[i,h_2,j]}$ $h_1 \curvearrowleft h_2$
Goal $[0,0,n+1]$
Axiom $[0,1]$
Inference Rules
sh $\frac{[i,j]}{[j,j+1]}$ $j \leqslant n$
re$_{\curvearrowright}$ $\frac{[k,i] \quad [i,j]}{[k,j]}$ $k \curvearrowright i$
re$_{\curvearrowleft}$ $\frac{[k,i] \quad [i,j]}{[k,j]}$ $i \curvearrowleft j$
Goal $[0,n+1]$
Axiom $[0^0,1]$
Inference Rules
sh $\frac{[i^b,j]}{[j^0,j+1]}$ $j \leqslant n$
ra $\frac{[i^b,j]}{[j^1,j+1]}$ $i \curvearrowright j$ $j \leqslant n$
re$_{\curvearrowleft}$ $\frac{[k^b,i] \quad [i^0,j]}{[k^b,j]}$ $i \curvearrowleft j$
re $\frac{[k^b,i] \quad [i^1,j]}{[k^b,j]}$
Goal $[0^0,n+1]$
Axioms $0 \leqslant i,j \leqslant n$
Inference Rules
right-attach
right-reduce $i \curvearrowright j$
left-attach
left-reduce $i \curvearrowleft j$
Goal

(a) Arc-standard

(b) Arc-hybrid

(c) Arc-eager

(d) Edge-factored graph-based parsing.

Figure 1: 1a-1c: Kuhlmann et al.'s inference rules for three transition systems, together with CKY-style visualizations of the local structures involved and, to their right, conditions for the rule to apply. 1d: the edge-factored graph-based parsing algorithm (Eisner and Satta, 1999) discussed in §5.

ditional state-keeping.

We show how to integrate the scoring functions for the arc-eager system; the arc-hybrid system is handled similarly. The score-annotated rules are as follows:

$$\frac{[i^b, j] : v}{[j^0, j+1] : 0}(\mathsf{sh}) \quad \frac{[k^b, i] : v_1 \quad [i^0, j] : v_2}{[k^b, j] : v_1 + v_2 + \Delta}(\mathsf{re}_{\curvearrowleft})$$

where $\Delta = f_{\mathsf{sh}}(\overset{\to\leftarrow}{w}_k, \overset{\to\leftarrow}{w}_i) + f_{\mathsf{re}_{\curvearrowleft}}(\overset{\to\leftarrow}{w}_i, \overset{\to\leftarrow}{w}_j)$ — abusing notation by referring to configurations by their features. The left-reduce rule says that we can first take the sequence of transitions asserted by $[k^b, i]$, which has a score of v_1, and then a shift transition moving w_i from b_0 to s_0. This means that the initial condition for $[i^0, j]$ is met, so we can take the sequence of transitions asserted by $[i^0, j]$ — say it has score v_2 — and finally a left-reduce transition to finish composing the larger transition sequence. Notice that the scores for sh and ra are 0, as the scoring of these transitions is accounted for by reduce rules elsewhere in the sequence.

4.2 Global Training

We employ large-margin training that considers each transition sequence globally. Formally, for a training sentence $w = w_1, \ldots, w_n$ with gold transition sequence $\mathbf{t}^{\text{gold}}$, our loss function is

$$\max_{\mathbf{t}} \left(F(\mathbf{t}) + \mathit{cost}(\mathbf{t}^{\text{gold}}, \mathbf{t}) - F(\mathbf{t}^{\text{gold}}) \right)$$

where $\mathit{cost}(\mathbf{t}^{\text{gold}}, \mathbf{t})$ is a custom margin for taking $\mathbf{t}$ instead of $\mathbf{t}^{\text{gold}}$ — specifically, the number of mis-attached nodes. Computing this max can again be done efficiently with a slight modification to the scoring of reduce transitions:

$$\frac{[k^b, i] : v_1 \quad [i^0, j] : v_2}{[k^b, j] : v_1 + v_2 + \Delta'}(\mathsf{re}_{\curvearrowleft})$$

where $\Delta' = \Delta + \mathbf{1}\left(\mathit{head}(w_i) \neq w_j\right)$. This loss-augmented inference or cost-augmented decoding (Taskar et al., 2005; Smith, 2011) technique has previously been applied to graph-based parsing by Kiperwasser and Goldberg (2016a).

Efficiency Note The computation decomposes into two parts: scoring all feature combinations, and using DP to find a proof for the goal theorem in the deduction system. Time-complexity analysis is usually given in terms of the latter, but the former might have a large constant factor, such as 10^4 or worse for neural-network-based scoring functions. As a result, in practice, with a small n, scoring with the feature set $\{\overset{\to\leftarrow}{s}_0, \overset{\to\leftarrow}{b}_0\}$ ($O(n^2)$) can be as time-consuming as the decoding steps ($O(n^3)$) for the arc-hybrid and arc-eager systems.

5 Theoretical Connections

Our minimal feature set brings implementation of practical optimal algorithms to TBDPs, whereas previously only *graph-based* dependency parsers (GBDPs) — a radically different, non-incremental paradigm — enjoyed the ability to deploy them. Interestingly, for both the transition- and graph-based paradigms, the optimal algorithms build dependency trees bottom-up from local structures. It is thus natural to wonder if there are deeper, more formal connections between the two.

In previous work, Kuhlmann et al. (2011) related the arc-standard system to the classic CKY algorithm (Cocke, 1969; Kasami, 1965; Younger, 1967) in a manner clearly suggested by Figure 1a; CKY can be viewed as a very simple graph-based approach. Gómez-Rodríguez et al. (2008, 2011) formally prove that sequences of steps in the *edge-factored* GBDP (Eisner, 1996) can be used to emulate any individual step in the arc-hybrid system (Yamada and Matsumoto, 2003) and the Eisner and Satta (1999, Figure 1d) version. However, they did not draw an explicitly direct connection between Eisner and Satta (1999) and TBDPs.

Here, we provide an update to these previous findings, stated in terms of the expressiveness of scoring functions, considered as parameterization.

For the edge-factored GBDP, we write the score for an edge as $f_G(\overset{\to\leftarrow}{h}, \overset{\to\leftarrow}{m})$, where h is the head and m the modifier. A tree's score is the sum of its edge scores. We say that a parameterized dependency parsing model A *contains* model B if for every instance of parameterization in model B, there exists an instance of model A such that the two models assign the same score to every parse tree. We claim:

Lemma 1. *The arc-eager model presented in §4.1 contains the edge-factored model.*

Proof Sketch. Consider a given edge-factored GBDP parameterized by f_G. For any parse tree, every edge $i \curvearrowleft j$ involves two deduction rules, and their contribution to the score of the final proof is $f_{\mathsf{sh}}(\overset{\to\leftarrow}{w}_k, \overset{\to\leftarrow}{w}_i) + f_{\mathsf{re}_{\curvearrowleft}}(\overset{\to\leftarrow}{w}_i, \overset{\to\leftarrow}{w}_j)$. We set $f_{\mathsf{sh}}(\overset{\to\leftarrow}{w}_k, \overset{\to\leftarrow}{w}_i) = 0$ and $f_{\mathsf{re}_{\curvearrowleft}}(\overset{\to\leftarrow}{w}_i, \overset{\to\leftarrow}{w}_j) = f_G(\overset{\to\leftarrow}{w}_j, \overset{\to\leftarrow}{w}_i)$. Similarly, for edges $k \curvearrowright i$ in the other direction, we set

Model	Training	Features	PTB		CTB	
			UAS (%)	UEM (%)	UAS (%)	UEM (%)
Arc-standard	Local	$\{\overset{\rightarrow\leftarrow}{s}_2, \overset{\rightarrow\leftarrow}{s}_1, \overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$93.95_{\pm 0.12}$	$52.29_{\pm 0.66}$	$88.01_{\pm 0.26}$	$36.87_{\pm 0.53}$
Arc-hybrid	Local	$\{\overset{\rightarrow\leftarrow}{s}_2, \overset{\rightarrow\leftarrow}{s}_1, \overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$93.89_{\pm 0.10}$	$50.82_{\pm 0.75}$	$87.87_{\pm 0.17}$	$35.47_{\pm 0.48}$
	Local	$\{\overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$93.80_{\pm 0.12}$	$49.66_{\pm 0.43}$	$87.78_{\pm 0.09}$	$35.09_{\pm 0.40}$
	Global	$\{\overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$94.43_{\pm 0.08}$	$53.03_{\pm 0.71}$	$88.38_{\pm 0.11}$	$36.59_{\pm 0.27}$
Arc-eager	Local	$\{\overset{\rightarrow\leftarrow}{s}_2, \overset{\rightarrow\leftarrow}{s}_1, \overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$93.80_{\pm 0.12}$	$49.66_{\pm 0.43}$	$87.49_{\pm 0.20}$	$33.15_{\pm 0.72}$
	Local	$\{\overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$93.77_{\pm 0.08}$	$49.71_{\pm 0.24}$	$87.33_{\pm 0.11}$	$34.17_{\pm 0.41}$
	Global	$\{\overset{\rightarrow\leftarrow}{s}_0, \overset{\rightarrow\leftarrow}{b}_0\}$	$\mathbf{94.53}_{\pm 0.05}$	$53.77_{\pm 0.46}$	$\mathbf{88.62}_{\pm 0.09}$	$\mathbf{37.75}_{\pm 0.87}$
Edge-factored	Global	$\{\overset{\rightarrow\leftarrow}{h}, \overset{\rightarrow\leftarrow}{m}\}$	$94.50_{\pm 0.13}$	$\mathbf{53.86}_{\pm 0.78}$	$88.25_{\pm 0.12}$	$36.42_{\pm 0.52}$

Table 2: Test set performance for different training regimes and feature sets. The models use the same decoders for testing and training. For each setting, the average and standard deviation across 5 runs with different random initializations are reported. Boldface: best (averaged) result per dataset/measure.

$f_{\mathsf{ra}}(\overset{\rightarrow\leftarrow}{w}_k, \overset{\rightarrow\leftarrow}{w}_i) = f_G(\overset{\rightarrow\leftarrow}{w}_k, \overset{\rightarrow\leftarrow}{w}_i)$ and $f_{\mathsf{re}}(\overset{\rightarrow\leftarrow}{w}_i, \overset{\rightarrow\leftarrow}{w}_j) = 0$. The parameterization we arrive at emulates exactly the scoring model of f_G. □

We further claim that the arc-eager model is more expressive than not only the edge-factored GBDP, but also the arc-hybrid model in our paper.

Lemma 2. *The arc-eager model contains the arc-hybrid model.*

Proof Sketch. We leverage the fact that the arc-eager model divides the sh transition in the arc-hybrid model into two separate transitions, sh and ra. When we constrain the parameters $f_{\mathsf{sh}} = f_{\mathsf{ra}}$ in the arc-eager model, the model hypothesis space becomes exactly the same as arc-hybrid's. □

The extra expressiveness of the arc-eager model comes from the scoring functions f_{sh} and f_{re} that capture structural contexts other than head-modifier relations. Unlike traditional higher-order graph-based parsing that directly models relations such as siblinghood (McDonald and Pereira, 2006) or grandparenthood (Carreras, 2007), however, the arguments in those two functions do not have any fixed type of structural interactions.

6 Experiments

Data and Evaluation We experimented with English and Chinese. For English, we used the Stanford Dependencies (de Marneffe and Manning, 2008) conversion (via the Stanford parser 3.3.0) of the Penn Treebank (Marcus et al., 1993, PTB). As is standard, we used §2-21 of the Wall Street Journal for training, §22 for development, and §23 for testing; POS tags were predicted using 10-way jackknifing with the Stanford max entropy tagger (Toutanova et al., 2003). For Chinese, we used the Penn Chinese Treebank 5.1 (Xue et al., 2002, CTB), with the same splits and head-finding rules for conversion to dependencies as Zhang and Clark (2008). We adopted the CTB's gold-standard tokenization and POS tags. We report unlabeled attachment score (UAS) and sentence-level unlabeled exact match (UEM). Following prior work, all punctuation is excluded from evaluation. For each model, we initialized the network parameters with 5 different random seeds and report performance average and standard deviation.

Implementation Details Our model structures reproduce those of Kiperwasser and Goldberg (2016a). We use 2-layer bi-directional LSTMs with 256 hidden cell units. Inputs are concatenations of 28-dimensional randomly-initialized part-of-speech embeddings and 100-dimensional word vectors initialized from GloVe vectors (Pennington et al., 2014) (English) and pre-trained skip-gram-model vectors (Mikolov et al., 2013) (Chinese). The concatenation of the bi-LSTM feature vectors is passed through a multi-layer perceptron (MLP) with 1 hidden layer which has 256 hidden units and activation function tanh. We set the dropout rate for the bi-LSTM (Gal and Ghahramani, 2016) and MLP (Srivastava et al., 2014) for each model according to development-set performance.[6] All parameters except the word embed-

[6] For bi-LSTM input and recurrent connections, we consider dropout rates in $\{0., 0.2\}$, and for MLP, $\{0., 0.4\}$.

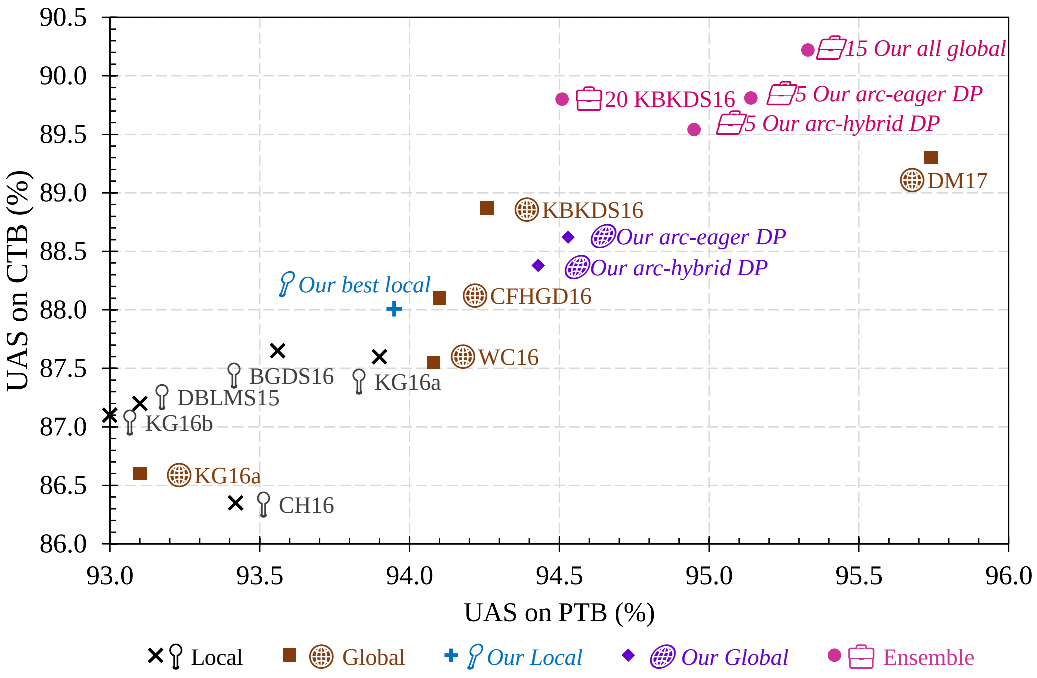


Figure 2: Comparing our UAS results with results from the literature. x-axis: PTB; y-axis: CTB. Most datapoint labels give author initials and publication year; citations are in the bibliography. Ensemble datapoints are annotated with ensemble size. Weiss et al. (2015) and Andor et al. (2016) achieve UAS of 94.26 and 94.61 on PTB with beam search, but did not report CTB results, and are therefore omitted.

dings are initialized uniformly (Glorot and Bengio, 2010). Approximately 1,000 tokens form a mini-batch for sub-gradient computation. We train each model for 20 epochs and perform model selection based on development UAS. The proposed structured loss function is optimized via Adam (Kingma and Ba, 2015). The neural network computation is based on the python interface to DyNet (Neubig et al., 2017), and the exact decoding algorithms are implemented in Cython.[7]

Main Results We implement exact decoders for the arc-hybrid and arc-eager systems, and present the test performance of different model configurations in Table 2, comparing global models with local models. All models use the same decoder for testing as during the training process. Though no global decoder for the arc-standard system has been explored in this paper, its local models are listed for comparison. We also include an edge-factored graph-based model, which is conventionally trained globally. The edge-factored model scores bi-LSTM features for each head-modifier pair; a maximum spanning tree algorithm is used to find the tree with the highest sum of edge scores. For this model, we use Dozat and Manning's (2017) biaffine scoring model, although in our case the model size is smaller.[8]

Analogously to the dev-set results given in §2, on the test data, the minimal feature sets perform as well as larger ones in locally-trained models. And there exists a clear trend of global models outperforming local models for the two different transition systems on both datasets. This illustrates the effectiveness of exact decoding and global training. Of the three types of global models, the arc-eager arguably has the edge, an empirical finding resonating with our theoretical comparison of their model expressiveness.

Comparison with State-of-the-Art Models Figure 2 compares our algorithms' results with those of the state-of-the-art.[9] Our models are competitive and an ensemble of 15 globally-trained models (5 models each for arc-eager DP, arc-hybrid DP and edge-factored) achieves 95.33 and 90.22 on PTB and CTB, respectively, reach-

[7] See https://github.com/tzshi/dp-parser-emnlp17.

[8] The same architecture and model size as other transition-based global models is used for fair comparison.

[9] We exclude Choe and Charniak (2016), Kuncoro et al. (2017) and Liu and Zhang (2017), which convert constituent-based parses to dependency parses. They produce higher PTB UAS, but access more training information and do not directly apply to datasets without constituency annotation.

ing the highest reported UAS on the CTB dataset, and the second highest reported on the PTB dataset among dependency-based approaches.

7 Related Work Not Yet Mentioned

Approximate Optimal Decoding/Training Besides dynamic programming (Huang and Sagae, 2010; Kuhlmann et al., 2011), various other approaches have been proposed for approaching global training and exact decoding. Best-first and A* search (Klein and Manning, 2003; Sagae and Lavie, 2006; Sagae and Tsujii, 2007; Zhao et al., 2013; Thang et al., 2015; Lee et al., 2016) give optimality certificates when solutions are found, but have the same worst-case time complexity as the original search framework. Other common approaches to search a larger space at training or test time include beam search (Zhang and Clark, 2011), dynamic oracles (Goldberg and Nivre, 2012, 2013; Cross and Huang, 2016b) and error states (Vaswani and Sagae, 2016). Beam search records the k best-scoring transition prefixes to delay local hard decisions, while the latter two leverage configurations deviating from the gold transition path during training to better simulate the test-time environment.

Neural Parsing Neural-network-based models are widely used in state-of-the-art dependency parsers (Henderson, 2003, 2004; Chen and Manning, 2014; Weiss et al., 2015; Andor et al., 2016; Dozat and Manning, 2017) because of their expressive representation power. Recently, Stern et al. (2017) have proposed minimal span-based features for constituency parsing.

Recurrent and recursive neural networks can be used to build representations that encode complete configuration information or the entire parse tree (Le and Zuidema, 2014; Dyer et al., 2015; Kiperwasser and Goldberg, 2016b), but these models cannot be readily combined with DP approaches, because their state spaces cannot be merged into smaller sets and thus remain exponentially large.

8 Concluding Remarks

In this paper, we have shown the following.

- The bi-LSTM-powered feature set $\{\overleftrightarrow{s}_0, \overleftrightarrow{b}_0\}$ is minimal yet highly effective for arc-hybrid and arc-eager transition-based parsing.

- Since DP algorithms for exact decoding (Huang and Sagae, 2010; Kuhlmann et al., 2011) have a run-time dependence on the number of positional features, using our mere two effective positional features results in a running time of $O(n^3)$, feasible for practice.

- Combining exact decoding with global training — which is also enabled by our minimal feature set — with an ensemble of parsers achieves 90.22 UAS on the Chinese Treebank and 95.33 UAS on the Penn Treebank: these are, to our knowledge, the best and second-best results to date on these data sets among "purely" dependency-based approaches.

There are many directions for further exploration. Two possibilities are to create even better training methods, and to find some way to extend our run-time improvements to other transition systems. It would also be interesting to further investigate relationships between graph-based and dependency-based parsing. In §5 we have mentioned important earlier work in this regard, and provided an update to those formal findings.

In our work, we have brought exact decoding, which was formerly the province solely of graph-based parsing, to the transition-based paradigm. We hope that the future will bring more inspiration from an integration of the two perspectives.

Acknowledgments: an author-reviewer success story We sincerely thank all the reviewers for their extraordinarily careful and helpful comments. Indeed, this paper originated as a short paper submission by TS&LL to ACL 2017, where an anonymous reviewer explained in the review comments how, among other things, the DP run-time could be improved from $O(n^4)$ to $O(n^3)$. In their author response, TS&LL invited the reviewer to co-author, suggesting that they ask the conference organizers to make the connection between anonymous reviewer and anonymous authors. All three of us are truly grateful to PC co-chair Regina Barzilay for implementing this idea, bringing us together!

We also thank Kai Sun for help with Chinese word vectors, and Xilun Chen, Yao Cheng, Dezhong Deng, Juneki Hong, Jon Kleinberg, Ryan McDonald, Ashudeep Singh, and Kai Zhao for discussions and suggestions. TS and LL were supported in part by a Google focused research grant to Cornell University. LH was supported in part by NSF IIS-1656051, DARPA N66001-17-2-4030, and a Google Faculty Research Award.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack LSTM parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2005–2010.

Miguel Ballesteros and Joakim Nivre. 2013. Going to the roots of dependency parsing. *Computational Linguistics*, 39(1):5–13.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 957–961.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 740–750, Doha, Qatar.

Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional attention with agreement for dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2204–2214.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas.

John Cocke. 1969. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York University.

James Cross and Liang Huang. 2016a. Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37.

James Cross and Liang Huang. 2016b. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1–11.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 340–345.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 457–464.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1019–1027.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 959–976.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics*, 1:403–414.

Carlos Gómez-Rodríguez, John Carroll, and David Weir. 2008. A deductive approach to dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technology*, pages 968–976.

Carlos Gómez-Rodríguez, John Carroll, and David Weir. 2011. Dependency parsing schemata and mildly non-projective dependency parsing. *Computational Linguistics*, 37(3):541–586.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 24–31.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 95–102.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086.

Tadao Kasami. 1965. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical report, Hawaii University Honolulu Department of Electrical Engineering.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 4th International Conference on Learning Representations*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016a. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Eliyahu Kiperwasser and Yoav Goldberg. 2016b. Easy-first dependency parsing with hierarchical tree LSTMs. *Transactions of the Association for Computational Linguistics*, 4:445–461.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency parsing*, volume 2 of *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 673–682.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258, Valencia, Spain.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one MST parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1744–1753.

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 729–739.

Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2016. Global neural CCG parsing with optimality guarantees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2366–2376.

Jiangming Liu and Yue Zhang. 2017. In-order transition-based constituent parsing. *Transactions of the Association for Computational Linguistics*. To appear.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. Stanford typed dependencies manual. Technical report, Stanford University.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pages 149–160.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

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

Fernando C. N. Pereira and David H. D. Warren. 1983. Parsing as deduction. In *Proceedings of the 21st Annual Meeting on Association for Computational Linguistics*, pages 137–144.

Kenji Sagae and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 691–698.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1044–1050.

Tianze Shi, Felix G. Wu, Xilun Chen, and Yao Cheng. 2017. Combining global models for parsing Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 31–39, Vancouver, Canada.

Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of Logic Programming*, 24(1):3–36.

Noah A. Smith. 2011. *Linguistic Structure Prediction*, volume 13 of *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool Publishers.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituent parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. To appear.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 896–903.

Quang Le Thang, Hiroshi Noji, and Yusuke Miyao. 2015. Optimal shift-reduce constituent parsing with structured perceptron. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1534–1544.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 173–180.

Ashish Vaswani and Kenji Sagae. 2016. Efficient structured inference for transition-based parsing with neural networks and error states. *Transactions of the Association for Computational Linguistics*, 4:183–196.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2306–2315.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333, Beijing, China.

Nianwen Xue, Fu-Dong Chiou, and Martha Palmer. 2002. Building a large-scale annotated Chinese corpus. In *Proceedings of the 19th International Conference on Computational Linguistics*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pages 195–206.

Daniel H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189 – 208.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 562–571.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.

Kai Zhao, James Cross, and Liang Huang. 2013. Optimal incremental parsing via best-first dynamic programming. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 758–768.

Quasi-Second-Order Parsing for 1-Endpoint-Crossing, Pagenumber-2 Graphs

Junjie Cao, Sheng Huang, Weiwei Sun and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{junjie.cao,huangsheng,ws,wanxiaojun}@pku.edu.cn

Abstract

We propose a new Maximum Subgraph algorithm for first-order parsing to 1-endpoint-crossing, pagenumber-2 graphs. Our algorithm has two characteristics: (1) it separates the construction for noncrossing edges and crossing edges; (2) in a single construction step, whether to create a new arc is deterministic. These two characteristics make our algorithm relatively easy to be extended to incorporiate crossing-sensitive second-order features. We then introduce a new algorithm for quasi-second-order parsing. Experiments demonstrate that second-order features are helpful for Maximum Subgraph parsing.

1 Introduction

Previous work showed that treating semantic dependency parsing as the search for Maximum Subgraphs is not only elegant in theory but also effective in practice (Kuhlmann and Jonsson, 2015; Cao et al., 2017). In particular, our previous work showed that 1-endpoint-crossing, pagenumber-2 (1EC/P2) graphs are an appropriate graph class for modelling semantic dependency structures (Cao et al., 2017). On the one hand, it is highly expressive to cover a majority of semantic analysis. On the other hand, the corresponding Maximum Subgraph problem with an arc-factored disambiguation model can be solved in low-degree polynomial time.

Defining disambiguation models on wider contexts than individual bi-lexical dependencies improves various syntactic parsers in different architectures. This paper studies exact algorithms for second-order parsing for 1EC/P2 graphs. The existing algorithm, viz. our previous algorithm (GCHSW, hereafter), has two properties that make it hard to incorporate higher-order features in a principled way. First, GCHSW does not explicitly consider the construction of noncrossing arcs. We will show that incorporiating higher-order factors containing crossing arcs without increasing time and space complexity is extremely hard. An effective strategy is to only include higher-order factors containing only noncrossing arcs (Pitler, 2014). But this *crossing-sensitive* strategy is incompatible with GCHSW. Second, all existing higher-order parsing algorithms for projective trees, including (McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010), require that which arcs are created in a construction step be deterministic. This design is also incompatible with GCHSW. In summary, it is not convenient to extend GCHSW to incorporate higher-order features while keeping the same time complexity.

In this paper, we introduce an alternative Maximum Subgraph algorithm for first-order parsing to 1EC/P2 graphs. while keeping the same time and space complexity to GCHSW, our new algorithm has two characteristics that make it relatively easy to be extended to incorporate crossing-sensitive, second-order features: (1) it separates the construction for noncrossing edges and possible crossing edges; (2) whether an edge is created is deterministic in each construction rule. We then introduce a new algorithm to perform second-order parsing. When all second-order scores are greater than or equal to 0, it exactly solves the corresponding optimization problem.

We implement a practical parser with a statistical disambiguation model and evaluate it on four data sets: those used in SemEval 2014 Task 8 (Oepen et al., 2014), and the dependency graphs extracted from CCGbank (Hockenmaier and Steedman, 2007). On all data sets, we find that our second-order parsing models are more ac-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 24–34
Copenhagen, Denmark, September 7–11, 2017.

curate than the first-order baseline. If we do not use features derived from syntactic trees, we get an absolute unlabeled F-score improvement of 1.3 on average. When syntactic analysis is used, we get an improvement of 0.4 on average.

2 Preliminaries

2.1 Maximum Subgraph Parsing

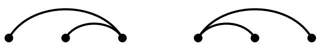
Semantic dependency parsing can be formulated as the search for Maximum Subgraph for graph class $\mathcal{G}$: Given a graph $G = (V, A)$, find a subset $A' \subseteq A$ with maximum total score such that the induced subgraph $G' = (V, A')$ belongs to $\mathcal{G}$. Formally, we have the following optimization problem:

$$\arg \max_{G^* \in \mathcal{G}(s,G)} \sum_{p \text{ in } G^*} s_{\text{part}}(s, p)$$

$\mathcal{G}(s, G)$ denotes the set of all graphs that belong to $\mathcal{G}$ and are compatible with s and G. G is usually a complete digraph. $s_{\text{part}}(s, p)$ evaluates the event that part p (from a candidate graph G^*) is good. We define the *order* of p according to the number of arcs it contains, in analogy with tree parsing in terminology. Previous work only discussed the first-order case:

$$\arg \max_{G^* \in \mathcal{G}(G)} \sum_{d \in \textsc{Arc}(G^*)} s_{\text{arc}}(d)$$

If $\mathcal{G}$ is the set of noncrossing or 1EC/P2 graphs, the above optimization problem can be solved in cubic-time (Kuhlmann and Jonsson, 2015) and quintic-time (Cao et al., 2017) respectively. Furthermore, ignoring one linguistically-rare structure in 1EC/P2 graphs descreases the complexity to $O(n^4)$. This paper is concerned with second-order parsing, with a special focus on the following factorizations:



And the objective function turns to be:

$$\sum_{d \in \textsc{Arc}(G^*)} s_{\text{arc}}(d) + \sum_{s \in \textsc{Sib}(G^*)} s_{\text{sib}}(s)$$

Sun et al. (2017) introduced a dynamic programming algorithm for second-order planar parsing. Their empirical evaluation showed that second-order features are effective to improve parsing accuracy. It is still unknown how to incorporate such features for 1EC/P2 parsing.

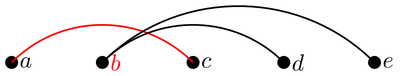


Figure 1: $e_{(a,c)}$'s crossing edges $e_{(b,d)}$ and $e_{(b,e)}$ share an endpoint b.

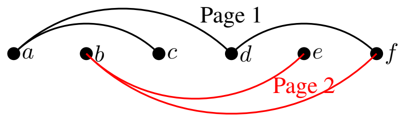


Figure 2: A pagenumber-2 graph. The upper and the lower figures represent two half-planes respectively.

2.2 1-Endpoint-Crossing, Pagenumber-2 Graphs

The formal description of the 1-endpoint-crossing property is adopted from (Pitler et al., 2013).

Definition 1. *Edges e_1 and e_2 cross if e_1 and e_2 have distinct endpoints and exactly one of the endpoints of e_1 lies between the endpoints of e_2.*

Definition 2. *A dependency graph is 1-Endpoint-Crossing if for any edge e, all edges that cross e share an endpoint p named pencil point.*

Given a sentence $s = w_0 w_1 \cdots w_{n-1}$ of length n, the vertices, i.e. words, are indexed with integers, an arc from w_i to w_j as $a_{(i,j)}$, and the common endpoint, namely pencil point, of all edges crossed with $a_{(i,j)}$ or $a_{(j,i)}$ as $pt(i, j)$. We denote an edge as $e_{(i,j)}$, if we do not consider its direction. Figure 1 is an example.

Definition 3. *A pagenumber-k graph means it consists at most k half-planes, and arcs on each half-plane are noncrossing.*

These half-planes may be thought of as the pages of a book, with the vertex line corresponding to the books spine, and the embedding of a graph into such a structure is known as a book embedding. Figure 2 is an example.

(Pitler et al., 2013) proved that 1-endpoint-crossing trees are a subclass of graphs whose pagenumber is at most 2. In Cao et al. (2017), we studied graphs that are constrained to be both 1-endpoint-crossing and pagenumber-2. In this paper, we ignored a complex and linguistic-rare

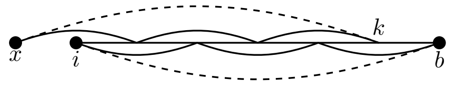


Figure 3: C structure has two crossing chains.

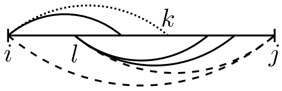


Figure 4: A prototype backbone of 1EC/P2 graphs. To decompose this structure, GCHSW focuses on $e_{(i,j)}$ and $e_{(l,j)}$, because these two edges can be optionally created without violation of both 1EC and P2 restrictions. Our algorithm focuses on the existence of $e_{(i,k)}$, and makes it the only edge that is constructed by applying a corresponding rule.

structure and studied a subset of 1EC/P2 graphs. The complex structure is named as C structures in our previous paper, and Figure 3 is the prototype of C structures. In this paper, we present new algorithms for finding optimal 1EC/P2, C-free graphs.

2.3 The GCHSWAlgorithm

Cao et al. (2017) designed a polynomial time Maximum Subgraph algorithm, viz. GCHSW, for 1EC/P2 graphs by exploring the following property: Every subgraph of a 1EC/P2 graph is also a 1EC/P2 graph. GCHSW defines a number of prototype backbones for decomposing a 1EC/P2 graph in a principled way. In each decomposition step, GCHSW focuses on the edges that can be created without violating either the 1EC nor P2 restriction. Sometimes, multiple edges can be created simultaneously in one single step. Figure 4 is an example.

There is an important difference between GCHSW and Eisner-style Maximum Spanning Tree algorithms (MST; Eisner, 1996; McDonald and Pereira, 2006; Koo and Collins, 2010). In each construction step, GCHSW allows multiple arcs to be constructed, but whether or not such arcs are added to the target graph depends on their arc-weights. If all arcs are assigned scores that are greater than 0, the output of our algorithm includes the most complicated 1EC/P2 graphs. For the higher-order MST algorithms, in a single construction step, it is clear whether adding a new arc, and which one. There is no local search. This deterministic strategy is also followed by Kuhlmann and Jonsson's Maximum Subgraph algorithm for noncrossing graphs. Higher-order MST models associate higher-order score functions with the construction of individual dependencies. Therefore the deterministic strategy is a prerequisite to incorporate higher-order features. The design of GCHSW is incompatible with this strategy.

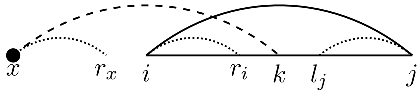


Figure 5: A typical structure of crossing arcs.

2.4 Challenge of Second-Order Decoding

It is very difficult to enumerate all high-order features for crossing arcs. Figure 5 illustrates the idea. There is a pair of corssing arcs, viz. $e_{(x,k)}$ and $e_{(i,j)}$. The key strategy to develop a dynamic programming algorithm to generate such crossing structure is to treat parts of this structures as intervals/spans together with an external vertex (Pitler et al., 2013; Cao et al., 2017). Without loss of generality, we assume $[i, j]$ makes up such an interval and x is the corresponding external vertex. When we consider $e_{(i,j)}$, its neighboring edges can be $e_{(i,r_i)}$ and $e_{(l_j,j)}$, and therefore we need to consider searching the best positions of both r_i and l_j. Because we have already taken into account three vertices, viz. x, i and j, the two new positions increase the time complexity to be at least quintic.

Now consider $e_{(x,k)}$. When we decompose the whole graph into inverval $[i, j]$ plus x and remaining part, we will factor out $e_{(x,k)}$ in a successive decomposition for resolving $[i, j]$ plus x. We cannot capture the second features associated to $e_{(x,k)}$ and $e_{(x,r_x)}$, because they are in different intervals, and when these intervals are combined, we have already hidden the position information of k. Explicitly encoding k increases the time complexity to be at least quintic too.

Pitler (2014) showed that it is still possible to build accurate tree parsers by considering only higher-order features of noncrossing arcs. This is in part because only a tiny fraction of neighboring arcs involve crossing arcs. However, this strategy is not easy to by applied to GCHSW, because GCHSW does not explicitly analyze sub-graphs of noncrossing arcs.

3 A New Maximum Subgraph Algorithm

Based on the discussion of Section 2.3 and 2.4, we can see that it is not easy to extend the existing algorithm, viz. GCHSW, to handle second-order features. In this paper, we propose an alternative first-order dynamic programming algorithm. Because ignoring one linguistically-rare structure associated with the **C** problem in GCHSW descreases the complexity, we exclude this structure in our algorithm. Formally, we introduce a new algorithm

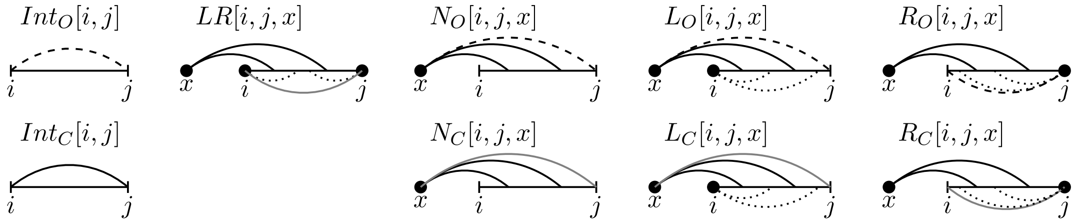


Figure 6: Graphical representations of sub-problems. Gray curves mean the corresponding edge in this sub-problem, but should be included in the final generated graph.

$$Int_O(i,j) \leftarrow \max \begin{cases} Int_O(i+1,j) \\ Int_C(i,j) \\ Int_C(i,k) + Int_O(k,j) \\ R_C(i,k,x) + Int_O(k,x) + L_O(x,j,k) + \mathrm{s}_{\mathrm{arc}}(i,k) \\ LR(i,k,x) + Int_O(k,x) + Int_O(x,j,k) + \mathrm{s}_{\mathrm{arc}}(i,k) \\ Int_O[i,x] + L_C[x,k,i] + N_O[k,j,x] + \mathrm{s}_{\mathrm{arc}}(i,k) \\ R_O[i,x,k] + Int_O[x,k] + L_O[k,j,x] + \mathrm{s}_{\mathrm{arc}}(i,k) \end{cases}$$

$$Int_C(i,j) \leftarrow \mathrm{s}_{\mathrm{arc}}(i,j) + \max \begin{cases} Int_O(i+1,j) \\ Int_C(i,k) + Int_O(k,j) \\ R_C(i,k,x) + Int_O(k,x) + L_O(x,j,k) + \mathrm{s}_{\mathrm{arc}}(i,k) \\ LR(i,k,x) + Int_O(k,x) + Int_O(x,j,k) + \mathrm{s}_{\mathrm{arc}}(i,k) \\ Int_O[i,x] + L_C[x,k,i] + N_O[k,j,x] + \mathrm{s}_{\mathrm{arc}}(i,k) \\ R_O[i,x,k] + Int_O[x,k] + L_O[k,j,x] + \mathrm{s}_{\mathrm{arc}}(i,k) \end{cases}$$

$$N_O(i,j,x) \leftarrow \max \begin{cases} Int_O(i,j) \\ N_C(i,j,x) + \mathrm{s}_{\mathrm{arc}}(x,j) \\ N_C(i,k,x) + Int_O(k,j) + \mathrm{s}_{\mathrm{arc}}(x,k) \end{cases}$$

$$N_C(i,j,x) \leftarrow \max \begin{cases} Int_O(i,j) \\ N_C(i,k,x) + Int_O(k,j) + \mathrm{s}_{\mathrm{arc}}(x,k) \end{cases}$$

$$LR(i,j,x) \leftarrow \max \left\{ L_O(i,k,x) + R_O(k,j,x) \right.$$

$$L_O(i,j,x) \leftarrow \max \begin{cases} Int_O(i,j) \\ L_C(i,j,x) + \mathrm{s}_{\mathrm{arc}}(x,j) \\ L_C(i,k,x) + N_O(k,j) + \mathrm{s}_{\mathrm{arc}}(x,k) \\ Int_O(i,k,x) + L_O(k,j) + \mathrm{s}_{\mathrm{arc}}(x,k) \end{cases}$$

$$L_C(i,j,x) \leftarrow \max \begin{cases} Int_O(i,j) \\ L_C(i,j,x) + \mathrm{s}_{\mathrm{arc}}(x,j) \\ L_C(i,k,x) + N_O(k,j,i) + \mathrm{s}_{\mathrm{arc}}(x,k) \\ Int_O(i,k) + L_O(k,j,i) + \mathrm{s}_{\mathrm{arc}}(x,k) \end{cases}$$

$$R_O(i,j,x) \leftarrow \max \begin{cases} Int_O(i,j) \\ R_C(i,j,x) + \mathrm{s}_{\mathrm{arc}}(x,j) \\ N_C(i,k,j) + R_O(k,j,x) + \mathrm{s}_{\mathrm{arc}}(x,k) \\ R_O(i,k,x) + Int_O(k,j) + \mathrm{s}_{\mathrm{arc}}(x,k) \end{cases}$$

$$R_C(i,j,x) \leftarrow \max \begin{cases} N_C(i,k,j) + R_O(k,j,x) + \mathrm{s}_{\mathrm{arc}}(x,k) \\ R_O(i,k,x) + Int_O(k,j) + \mathrm{s}_{\mathrm{arc}}(x,k) \end{cases}$$

Figure 7: A dynamic program to find optimal 1EC/P2, **C**-free graphs with arc-factored weights.

to solve the following optimization problem:

$$\arg \max_{G^* \in \mathcal{G}(G)} \sum_{d \in \textsc{Arc}(G^*)} \mathrm{s}_{\mathrm{arc}}(d)$$

where $\mathcal{G}$ means 1EC/P2, **C**-free graphs. Our algorithm has the same time and space complexity to the degenerated version of GCHSW. We represent our algorithm using undirected graphs.

3.1 Sub-problems

Following GCHSW, we consider five sub-problems when we construct a maximum dependency graph on a given interval $[i,k]$. Though the sub-problems introduced by GCHSW and us handle similar structures, their definitions are quite different. The sub-problems are explained as follows:

Int $Int[i,j]$ represents an interval from i to j inclusively. And there is no edge $e_{(i',j')}$ such that $i' \in [i,j]$ and $j' \notin [i,j]$. We distinguish two sub-types for **Int**. $Int_O[i,j]$ may or may not contain $e_{(i,j)}$, while $Int_C[i,j]$ contains $e_{(i,j)}$.

LR $LR[i,j,x]$ represents an interval from i to j inclusively and an external vertex x. $\forall p \in [i,j], pt(x,p) = i$ or j. $LR[i,j,x]$ implies the existence of $e_{(i,j)}$, but does not contain $e_{(i,j)}$. When $LR[i,j,x]$ is combined with other DP sub-structures, $e_{(i,j)}$ is immediately created. $LR[i,j,x]$ disallows neither $e_{(x,i)}$ nor $e_{(x,j)}$.

N $N[i,j,x]$ represents an interval from i to j inclusively and an external vertex x. $\forall p \in [i,j], pt(x,p) \notin [i,j]$. $N[i,j,x]$ could contain $e_{(i,j)}$ but disallows $e_{(x,i)}$. We distinguish two sub-types. $N_O[i,j,x]$ may or may not contain $e_{(x,j)}$. $N_C[i,j,x]$ implies the existence of but does not contain $e_{(x,j)}$. When $N[i,j,x]$ is combined with others, $e_{(x,j)}$ is immediately created.

L $L[i,j,x]$ represents an interval from i to j inclusively as well as an external vertex x. $\forall p \in [i,j], pt(x,p) = i$. $L[i,j,x]$ could contain $e_{(i,j)}$ but disallows $e_{(x,i)}$. We distinguish sub-two types for **L**. $L_O[i,j,x]$ may or may not contain $e_{(x,j)}$. $L_C[i,j,x]$ implies the existence of but does not contain $e_{(x,j)}$. When it is combined with others, $e_{(x,j)}$ is immediately created.

R $R[i,j,x]$ represents an interval from i to j inclusively as well as an external vertex x. $\forall p \in [i,j], pt(x,p) = j$. $R[i,j,x]$ disallows $e_{(x,j)}$ and $e_{(x,i)}$. We distinguish two sub-types for **R**. $R_O[i,j,x]$ may or may not contain $e_{(i,j)}$. $R_C[i,j,x]$ implies the existence of but does not contain $e_{(i,j)}$. When it is combined with others, $e_{(i,j)}$ is immediately created.

3.2 Decomposing Sub-problems

Figure 7 gives a sketch of our dynamic programming algorithm. We give a detailed illustration for **Int**, a rough idea for **L** and **LR**, and omit other sub-problems. More details about the whole algorithm can be found in the supplementary note.

3.2.1 Decomposing an Int Sub-problem

Consider $Int_O[i,j]$ and $Int_C[i,j]$ sub-problem. Because the decomposition for $Int_C[i,j]$ is very similar to $Int_O[i,j]$ and needs to be modified by our second-order parsing algorithm, we only show the decomposition of $Int_C[i,j]$. Assume that $k (k \in (i,j))$ is the **farthest** vertex that is adjacent to i, and $x = pt(i,k)$. If there is no such k (i.e. there no arc from i to some other node in this interval), then we denote k as $\emptyset$. So it is to x. We illustrate different cases as following and give a graphical representation in Figure 8.

Case a: $k = \emptyset$**.** We can directly consider interval $[i+1,j]$. Because there is no edge from i to any node in $[i+1,j]$, $[i+1,j]$ is an $\mathbf{Int_O}$.

Case b: $x = \emptyset$**.** $x = \emptyset$ means that $e_{(i,k)}$ does not cross other arcs. So $[i,k]$ and $[k,j]$ are **Int**.

Case c: $x \in (k,j]$**.** $e_{(i,k)}$ is taken as a possible crossing edge. k and x divide the interval $[i,j]$ into three parts: $[i,k]$, $[k,x]$, $[x,j]$. Because x may be j, interval $[x,j]$ may only contain j and become an empty interval. We define x' as the pencil point of all edges from (i,k) to x, and distinguish two sub-problems as follows.

c.1 Assume that there exists an edge from k to some node r in $(x,j]$, so x' can only be k and pencil point of edges from k to $(x,j]$ is x. Thus interval $[i,k,x]$ is an **R**. Due to the existence of $e_{(i,k)}$, its sub-type is $\mathbf{R_C}$. The $e_{(i,k)}$ is created in this construction and thus not contained by $R_C[i,k,x]$. An edge from within $[k,x]$ to outside violates the 1EC restriction, so $[k,x]$ is an **Int**. Since x is endpoint of edge from k to $[x,r]$, interval $[k,j]$ is an $\boldsymbol{L_O}$ with external vertex k.

c.2 We assume no edge from k to any node in $[x,j]$, x' thus can be i or k. As a result, $[x,j]$ is an **Int** and $[i,k,x]$ is an **LR**.

Case d: $x \in (i,k)$**.**

d.1 Assume that there exist edges from i to (x,k), so the pencil point of edges from x to $(k,j]$ is i. Therefore $[k,j]$ is an **N**. Because x is pencil point of edges from i to $(x,k]$, $[x,k]$ is an **L**. Furthmore, it is an $\mathbf{L_C}$ because we generate $e_{(i,k)}$ in this step. It is obvious that $[i,x]$ is an **Int**.

d.2 Assume that there exists edges from k to (i,x), and the pencil point of edges from x to $(k,j]$ is thus k. Similar to the above analysis, we reach $R_O[i,x,k] + Int_O[x,k] + L_O[k,j,x] + e_{(i,k)} + e_{(i,j)}$.

For $Int_O[i,j]$, because there may be $e_{(i,j)}$, we add one more rule: $Int_O[i,j] = Int_C[i,j]$. And we do not need to create $e_{(i,j)}$ in all cases.

3.2.2 Decomposing an L Sub-problem

Without loss of generality, we show the decomposition of $L_O[i,j,x]$ as follows. For $L_C[i,j,x]$, we ignore Case b but follow the others.

Case a. If there is no more edge from x to $(i,j]$, then it will degenerate to $Int_O[i,j]$.

Case b. If there exists $e_{(x,j)}$, then it will degenerate to $L_C[i,j,x] + e_{(x,j)}$.

Case c. Assume that there are edges from x to (i,j) and $e_{(x,k)}$ is the farthest one. It divides $[i,j]$ into $[i,k]$ and $[k,j]$.

c.1 If there is an edge from x to (i,k), $[i,k]$ and $[k,j]$ are $L_C[i,k,x]$ and $N_O[k,j,i]$.

c.2 If there is no edge from x to (i,k), $[i,k]$ and $[k,j]$ are $Int_O[i,k]$ and $L_O[k,j,i]$.

Figure 8 is a graphical representation.

3.2.3 Decomposing an LR Sub-problem

$LR[i,j,x]$ means i or j is the pencil point of edges from x to (i,j). We show the decomposition of $LR[i,j,x]$ as follows:

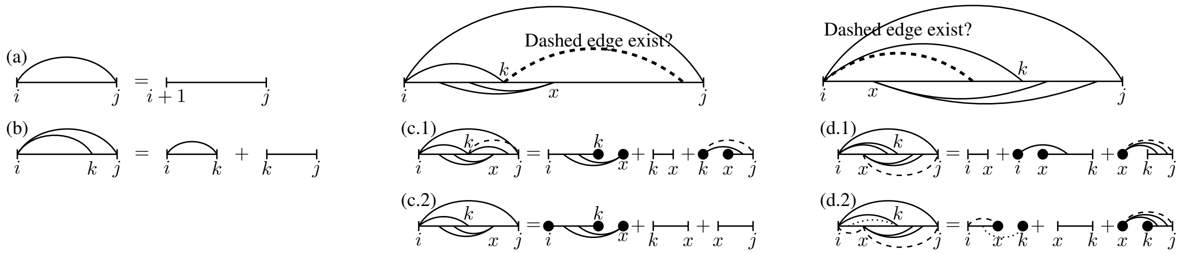


Figure 8: Decomposition for $Int_C[i,j]$ in the first-order parsing algorithm. $pt(i,k) = x$.

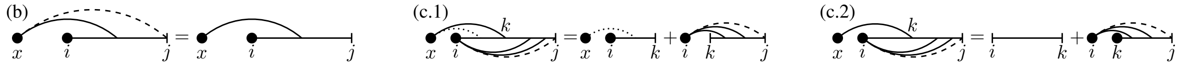


Figure 9: Decomposition for $L_O[i,j,x]$.

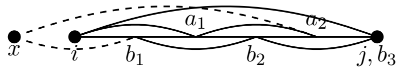


Figure 10: $b_3 = j$, Not both $e_{(x,b_1)}$ and $e_{(x,a_2)}$ exist.

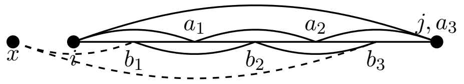


Figure 11: $a_3 = j$. Both $e_{(x,b_1)}$ and $e_{(x,b_3)}$ exist.

Case a. If there is a vertex k within (i,j), which divides $[i,j]$ into $[i,k]$ and $[k,j]$. And it guarantees no edge from $[i,k)$ to $(k,j]$. i is the pencil point of edges from x to $(i,k]$ because no edge from j to (i,k) can cross these edges. Similarly j has to be the pencil point of edges from x to (k,j). Obviously, $[i,k]$ is an L_O and $[k,j]$ is an R_O with external x. Thus the problem is decomposed as $L_O[i,k,x] + R_O[k,j,x]$.

Case b. If there is no such vertex k, there must be edges from $[i,k')$ to $(k',j]$ for every k' in (i,j) without considering $e_{(i,j)}$. For $i+1$, we assume $e_{(i,a_1)}$ is the farthest edge that goes from i. For a_1, we assume $e_{(b_1,b_2)}$ is the farthest edge from b_1 where b_1 is in (i,a_1) and b_2 is in (a_1,j). For b_2, we assume $e_{(a_1,a_3)}$ is the farthest edge from a_1 where a_3 is in (b_2,j) and a_1 is the pencil point. We then get the series $\{a_1, a_2, a_3 ... a_n\}$ and $\{b_1, b_2 ... b_m\}$ which guarantees $b_i < a_i$, $a_i < b_{i+1}$ and $max(a_n, b_m) = j$.

If $b_m = j$, we will get a graph like Figure 10. If $e_{(x,b_1)}$ exists, this LR subproblem degenerates to an L subproblem. If $e_{(x,a_n)}$ exists, this subproblem degenerates to an R subproblem.

If $a_m = j$, we will get a graph like Figure 11. If there exists only $e_{(x,b_1)}$ or $e_{(x,b_m)}$, we can solve it like $b_m = j$. If both exist, this is a typical C-structure like Figure 3 and we cannot get it through other decompostion.

The above discussion gives the rough idea of the correctness of the following conclusion.

Theorem 1. *Our new algorithm is sound and complete with respect to* 1EC/P2*, C-free graphs.*

3.3 Spurious Ambiguity

An **LR**, **L**, **R** or **N** sub-problem allows to build crossing arcs, but does not necessarily create crossing arcs. For example, $L_C[i,j,x]$ allows $e_{(i,j)}$ to cross with $e_{(x,y)}$ $(y \in (i,j))$. Because every subgraph of a 1EC/P2 graph is also a 1EC/P2 graph, we allow an $L_C[i,j,x]$ to be directly degenerated to $I_O[i,j]$. In this way, we can make sure that all subgraphs can be constructed by our algorithm. Figure 12 shows the rough idea. To generate the same graph, we have different derivations. The spurious ambiguity in our algorithm does not affect the correctness of first-order parsing, because scores are assigned to individual dependencies, rather than derivation processes. There is no need to distinguish one special derivation here.

4 Quasi-Second-Order Extension

We propose a second-order extension of our new algorithm. We focus on factorizations introduced in Section 2.1. Especially, the two arcs in a factor should not cross other arcs. Formally, we introduce a new algorithm to solve the optimization problem with the following objective:

$$\sum_{d \in \textsc{Arc}(G^*)} s_{\text{arc}}(d) + \sum_{s \in \textsc{Sib}(G^*)} \max(s_{\text{sib}}(s), 0)$$

In the first-order algorithm, all noncrossing edges can be constructed as the frontier edge of an $\mathbf{Int_C}$.

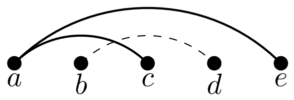


Figure 12: Illustration of spurious ambiguity. The two solid curves represent two arcs in the target graph, but not the dashed one. Excluding crossing edges leads to the first derivation: $Int_C[a,e] \Rightarrow e_{(a,e)} + Int_C[a,c] + Int_O[c,e] + e_{(a,c)}$. Assuming that a pair of crossing arcs may exist yields another derivation: $Int_C[a,e] \Rightarrow e_{(a,e)} + LR[a,c,d] + Int_O[k,d] + L_O[d,e,c] + e_{(a,c)}$; Then $LR[a,c,d] \Rightarrow L_O[a,b,d] + R_O[b,c,d] \Rightarrow Int_O[a,b] + Int_O[b,c]$.

So we can develop an exact decoding algorithm by modifying the composition for $\mathbf{Int_C}$ while keeping intact the decomposition for **LR**, **N**, **L**, **R**.

4.1 New Decomposition for Int_C

In order to capture the second-order features from noncrossing neighbors, we need to find the rightmost node adjacent to i, denoted as r_i, and the leftmost node adjacent to j, denoted as l_j,while $i < r_i \leq l_j < j$. To do this, we split $Int_C[i,j]$ into at most three parts to capture the sibling factors. Denote the score of adjacent edges $e_{(i,j_1)}$ and $e_{(i,j_2)}$ as $\mathrm{s}_2(i,j_1,j_2)$. When j is the inner most node adjacent to i, we denote the score as $\mathrm{s}_2(i,\emptyset,j)$. We give a sketch of the decomposition in Figure 14 and a graphical representation in Figure 13. The following is a rough illustration.

Case a: $r_i = \emptyset$**.** We further distinguish three sub-problems:

a.1 If $l_j = \emptyset$ too, both sides are the inner most second-order factor.

a.2 There is a crossing arc from j. This case is handled in the same way as the first-order algorithm.

a.3 $l_j \neq \emptyset$. We introduce a new decomposition rule.

Case b: There is a crossing arc from i**.**

b.1 $l_j = \emptyset$. Similar case to (a.2).

b.2 There is a crossing arc from j. Similar case to (a.2).

b.3 There is a noncrossing arc from j. We introduce a new rule to calculate $\textsc{Sib}(j,l_j,i)$.

Case c: There is a noncrossing arc from i**.**

c.1 $l_j = \emptyset$. Similar to (a.3).

c.2 There is a crossing arc from j. Similar to (b.3).

c.3 There is a noncrossing arc from j too. We introduce a new rule to calculate $\textsc{Sib}(i,r_i,j)$ and $\textsc{Sib}(j,l_j,i)$.

4.2 Complexity

The complexity of both first- and second-order algorithms can be analyzed in the same way. The sub-problem **Int** is of size $O(n^2)$, with a calculating time of order $O(n^2)$ at most. For sub-problems **L**, **R**, **LR**, and **N**, each has $O(n^3)$ elements, with a unit calculating time $O(n)$. Therefore both algorithms run in time of $O(n^4)$ with a space requirement of $O(n^3)$.

4.3 Discussion

A traditional second-order model takes as the objective function $\sum_{s \in \textsc{Sib}(G^*)} \mathrm{s_{sib}}(s)$. Our model instead tries to optimize $\sum_{s \in \textsc{Sib}(G^*)} \max(\mathrm{s_{sib}}(s), 0)$. This model is somehow inadequate given that the second-order score function cannot penalize a *bad* factor. When a negative score is assigned to a second-order factor, it will be taken as 0 by our algorithm.

This inadequacy is due to the spurious ambiguity problem that is illustrated in Section 3.3. Take the two derivations in Figure 12 for example. The derivation that starts from $Int_C[a,e] \Rightarrow Int_C[a,c] + Int_O[c,e]$ incorporates the second-order score $\mathrm{s_{sib}}(a,c,e)$. This is different when we consider the derivation that starts from $Int_C[a,e] \Rightarrow LR[a,c,d] + Int_O[k,d] + L_O[d,e,c]$. Because we assume temporarily that $e_{(a,c)}$ crosses others, we do not consider $\mathrm{s_{sib}}(a,c,e)$. We can see from this example that second-order scores not only depend on the derived graphs but also sensitive to the derivation processes.

If a second-order score is greater than 0, our algorithm selects the derivation that takes it into account since it increases the total score. If a second-order score is negative, our algorithm avoids including it by selecting other paths. In other words, our algorithm treats this score as 0.

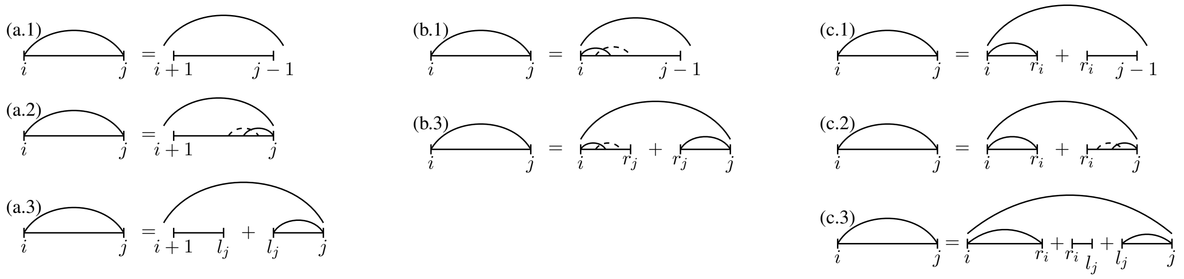


Figure 13: Decomposition for $Int_C[i,j]$ in the second-order parsing algorithm.

$$Int_C(i,j) \leftarrow \mathrm{s}_{\mathrm{arc}}(i,j) + \max \begin{cases} Int_O(i+1,j-1) + \mathrm{s}_{\mathrm{sib}}(i,\emptyset,j) + \mathrm{s}_{\mathrm{sib}}(j,\emptyset,i) \\ Int_O(i+1,j) + \mathrm{s}_{\mathrm{sib}}(i,\emptyset,j) \\ Int_O(i+1,l_j) + Int_C(l_j,j) + \mathrm{s}_{\mathrm{sib}}(i,\emptyset,j) + \\ \qquad \mathrm{s}_{\mathrm{sib}}(j,l_j,i) \\ Int_O(i,j-1) + \mathrm{s}_{\mathrm{sib}}(j,\emptyset,i) \\ Int_O(i,l_j) + Int_C(l_j,j) + \mathrm{s}_{\mathrm{sib}}(j,l_j,i) \\ Int_C(i,r_i) + Int_O[r_i,j-1] + \mathrm{s}_{\mathrm{sib}}(i,r_i,j) + \\ \qquad \mathrm{s}_{\mathrm{sib}}(j,\emptyset,i) \\ Int_C(i,r_i) + Int_O[r_i,j] + \mathrm{s}_{\mathrm{sib}}(i,r_i,j) \\ Int_C(i,r_i) + Int_O[r_i,l_j] + Int_C(l_j,j) + \\ \qquad \mathrm{s}_{\mathrm{sib}}(i,r_i,j) + \mathrm{s}_{\mathrm{sib}}(j,l_j,i) \\ R_C(i,k,x) + Int_O(k,x) + L_O(x,j,k) + e_{(i,k)} \\ LR(i,k,x) + Int_O(k,x) + Int_O(x,j,k) + e_{(i,k)} \\ Int_O[i,x] + L_C[x,k,i] + N_O[k,j,x] + e_{(i,k)} \\ R_O[i,x,k] + Int_O[x,k] + L_O[k,j,x] + e_{(i,k)} \end{cases}$$

Figure 14: Decomposition for $Int_C[i,j,x]$.

5 Practical Parsing

5.1 Derivation-Sensitive Training

We extend our quartic-time parsing algorithm into a practical parser. In the context of data-driven parsing, this requires an extra disambiguation model. As with many other parsers, we employ a global linear model. Following Zhang et al. (2016)'s experience, we define rich features extracted from word, POS-tags and pseudo trees. To estimate parameters, we utilize the averaged perceptron algorithm (Collins, 2002).

Our training proceudre is sensitive to derivation rather then derived graphs. For each sentence, we first apply our algorithm to find the optimal prediction derivation. The we collect all first- and second-order factors from this derivation to update parameters. To train a first-order model, because our algorithm includes all factors, viz. dependencies, there is no difference between our derivation-based method and a traditional derived structure-based method. For the second-order model, our method increases the second-order scores somehow.

5.2 Data and Preprocessing

We evaluate first- and second-order models on four representative data sets: CCGBank (Hockenmaier and Steedman, 2007), DeepBank (Flickinger et al., 2012), Enju HPSGBank (Miyao et al., 2005) and Prague Dependency TreeBank (Hajic et al., 2012). We use "standard" training, validation, and test splits to facilitate comparisons.

- Following previous experimental setup for English CCG parsing, we use section 02-21 as training data, section 00 as the development data, and section 23 for testing.

- The DeepBank, Enju HPSGBank and Prague Dependency TreeBank are from SemEval 2014 Task 8 (Oepen et al., 2014), and the data splitting policy follows the shared task.

Experiments for CCG-grounded analysis were performed using automatically assigned POS-tags that are generated by a symbol-refined HMM tagger (Huang et al., 2010). Experiments for the other three data sets used POS-tags provided by the shared task. We also use features extracted from pseudo trees. We utilize the Mate parser (Bohnet, 2010) to generate pseudo trees. All experimental results consider directed dependencies in a standard way. We report Unlabeled Precision (UP), Recall (UR) and F-score (UF), which are calculated using the official evaluation tool provided by SDP2014 shared task.

5.3 Accuracy

Table 1 lists the accuracy of our system. The output of our parser was evaluated against each dependency in the corpus. We can see that the first-order parser obtains a considerably good accuracy, with rich syntactic features. Furthermore, we can see that the introduction of higher-order features improves parsing substantially for all data sets, as expected. When syntactic trees are utilized, the

		DeepBank			EnjuBank			CCGBank			PCEDT		
Tree		UP	UR	UF	UP	UR	UF	UP	UR	UF	UP	UR	UF
No	1or	89.43	83.03	86.11	90.10	87.10	88.58	91.63	88.07	89.82	88.13	81.53	84.70
	2or	89.23	85.98	87.57	90.88	89.90	90.39	91.96	89.54	90.74	88.56	84.57	86.52
Syn	1or	91.24	87.14	89.14	92.72	90.96	91.83	94.28	91.79	93.02	91.53	86.95	89.18
	2or	90.93	88.79	89.85	92.73	92.11	92.42	93.99	92.27	93.13	91.02	88.20	89.59

Table 1: Parsing accuracy evaluated on the development sets.

		DeepBank			EnjuBank			CCGBank			PCEDT		
Tree		UP	UR	UF	UP	UR	UF	UP	UR	UF	UP	UR	UF
No	1or	88.87	82.50	85.57	90.12	86.76	88.41	91.95	88.29	90.08	86.87	80.45	83.54
	2or	88.77	85.61	87.16	91.06	89.50	90.27	92.25	89.80	91.01	87.07	83.45	85.22
Syn	1or	90.68	86.57	88.58	92.82	90.62	91.71	94.32	91.88	93.09	90.11	85.83	87.97
	2or	90.13	88.21	89.16	92.84	91.50	92.17	94.09	92.27	93.17	89.73	87.13	88.41
SJW (2or)		89.99	87.77	88.87	92.87	92.04	92.46	93.45	92.51	92.98	89.58	87.73	88.65

Table 2: Parsing accuracy evaluated on the test sets. "SJW" denotes the book embedding parser introduced in (Sun et al., 2017).

improvement is smaller but still significant on the three SemEval data sets.

Table 2 lists the parsing results on the test data together with the result obtained by Sun et al. (SJW; 2017)'s system. The building architectures of both systems are comparable.

1. Both systems have explicit control of the output structures. While Sun et al.'s system constrain the output graph to be P2 only, our system adds an additional 1EC restriction.
2. Their system's second-order features also includes both-side neighboring features.
3. Their system uses beam search and dual decomposition and therefore approximate, while ours perform exact decoding.

We can see that while our purely Maximum Subgraph parser obtains better results on DeepBank and CCGBank; while the book embedding parser is better on the other two data sets.

5.4 Analysis

Our algorithm is sensitive to the derivation process and may exclude a couple of negative second-order scores by selecting misleading derivations. Neverthess, our algorithm works in an exact way to include all positive second-order scores. Table 3 shows the coverage of all second-order factors. On average, 99.67% second-order factors are calculated by our algorithm. This relatively satisfactory coverage suggests that our algorithm is very effective to include second-order features. Only a very small portion is dropped.

	DeepBank	EnjuBank	CCGBank	PCEDT
No	99.08	99.52	99.67	98.32
Syn	99.77	99.69	99.88	99.33

Table 3: Coverage of second-order factors on the developmenet data.

6 Conclusion

This paper proposed two exact, graph-based algorithms for 1EC/P2 parsing with first-order and quasi-second-order scores. The resulting parser has the same asymptotic run time as Cao et al. (2017)'s algorithm. An exploration of other factorizations that facilitate semantic dependency parsing may be an interesting avenue for future work. Recent work has investigated faster decoding for higher-order graph-based projective parsing e.g. vine pruning (Rush and Petrov, 2012) and cube pruning (Zhang and McDonald, 2012). It would be interesting to extend these lines of work to decrease the complexity of our quartic algorithm.

Acknowledgments

This work was supported by 863 Program of China (2015AA015403), NSFC (61331011), and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). Weiwei Sun is the corresponding author.

References

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, Beijing, China, pages 89–97. http://www.aclweb.org/anthology/C10-1011.

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. 2017. Parsing to 1-endpoint-crossing, pagenumber-2 graphs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *In Proc. EMNLP-CoNLL*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1–8. https://doi.org/10.3115/1118693.1118694.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proceedings of the 16th conference on Computational linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 340–345.

Daniel Flickinger, Yi Zhang, and Valia Kordoni. 2012. Deepbank: A dynamically annotated treebank of the wall street journal. In *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories*. pages 85–96.

Jan Hajic, Eva Hajicová, Jarmila Panevová, Petr Sgall, Ondej Bojar, Silvie Cinková, Eva Fucíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jirí Semecký, Jana Sindlerová, Jan Stepánek, Josef Toman, Zdenka Uresová, and Zdenek Zabokrtský. 2012. Announcing prague czech-english dependency treebank 2.0. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*. Istanbul, Turkey.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics* 33(3):355–396.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Cambridge, MA, pages 12–22. http://www.aclweb.org/anthology/D10-1002.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 1–11. http://www.aclweb.org/anthology/P10-1001.

Marco Kuhlmann and Peter Jonsson. 2015. Parsing to noncrossing dependency graphs. *Transactions of the Association for Computational Linguistics* 3:559–570.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006))*. volume 6, pages 81–88.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2005. Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the penn treebank. In *IJCNLP*. pages 684–693.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics and Dublin City University, Dublin, Ireland, pages 63–72. http://www.aclweb.org/anthology/S14-2008.

Emily Pitler. 2014. A crossing-sensitive third-order factorization for dependency parsing. *TACL* 2:41–54. http://www.transacl.org/wp-content/uploads/2014/02/39.pdf.

Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2013. Finding optimal 1-endpoint-crossing trees. *TACL* 1:13–24. http://www.transacl.org/wp-content/uploads/2013/03/paper13.pdf.

Alexander Rush and Slav Petrov. 2012. Vine pruning for efficient multi-pass dependency parsing. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Montréal, Canada, pages 498–507. http://www.aclweb.org/anthology/N12-1054.

Weiwei Sun, Junjie Cao, and Xiaojun Wan. 2017. Semantic dependency parsing via book embedding. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Jeju Island, Korea, pages 320–331. http://www.aclweb.org/anthology/D12-1030.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics* 42(3):353–389. http://aclweb.org/anthology/J16-3001.

Position-aware Attention and Supervised Data Improve Slot Filling

Yuhao Zhang, Victor Zhong, Danqi Chen,
Gabor Angeli, Christopher D. Manning
Stanford University
Stanford, CA 94305
{yuhao, vzhong, danqi}@cs.stanford.edu
{angeli, manning}@cs.stanford.edu

Abstract

Organized relational knowledge in the form of "knowledge graphs" is important for many applications. However, the ability to populate knowledge bases with facts automatically extracted from documents has improved frustratingly slowly. This paper simultaneously addresses two issues that have held back prior work. We first propose an effective new model, which combines an LSTM sequence model with a form of entity position-aware attention that is better suited to relation extraction. Then we build TACRED, a large (119,474 examples) supervised relation extraction dataset, obtained via crowdsourcing and targeted towards TAC KBP relations. The combination of better supervised data and a more appropriate high-capacity model enables much better relation extraction performance. When the model trained on this new dataset replaces the previous relation extraction component of the best TAC KBP 2015 slot filling system, its F_1 score increases markedly from 22.2% to 26.7%.

1 Introduction

A basic but highly important challenge in natural language understanding is being able to populate a knowledge base with relational facts contained in a piece of text. For the text shown in Figure 1, the system should extract triples, or equivalently, knowledge graph edges, such as ⟨*Penner*, *per:spouse*, *Lisa Dillman*⟩. Combining such extractions, a system can produce a knowledge graph of relational facts between persons, organizations, and locations in the text. This task involves entity recognition, mention coreference and/or entity linking, and relation extraction; we focus on the most challenging "slot filling" task of filling in the relations between entities in the text.

> ***Penner*** *is survived by his brother,* ***John****, a* ***copy editor*** *at the* ***Times****, and his former wife,* ***Times sportswriter Lisa Dillman****.*

Subject	**Relation**	**Object**
Mike Penner	per:spouse	Lisa Dillman
Mike Penner	per:siblings	John Penner
Lisa Dillman	per:title	Sportswriter
Lisa Dillman	per:employee_of	Los Angeles Times
John Penner	per:title	Copy Editor
John Penner	per:employee_of	Los Angeles Times

Figure 1: An example of relation extraction from the TAC KBP corpus.

Organized relational knowledge in the form of "knowledge graphs" has become an important knowledge resource. These graphs are now extensively used by search engine companies, both to provide information to end-users and internally to the system, as a way to understand relationships. However, up until now, automatic knowledge extraction has proven sufficiently difficult that most of the facts in these knowledge graphs have been built up by hand. It is therefore a key challenge to show that NLP technology can effectively contribute to this important problem.

Existing work on relation extraction (e.g., Zelenko et al., 2003; Mintz et al., 2009; Adel et al., 2016) has been unable to achieve sufficient recall or precision for the results to be usable versus hand-constructed knowledge bases. Supervised training data has been scarce and, while techniques like distant supervision appear to be a promising way to extend knowledge bases at low cost, in practice the training data has often been too noisy for reliable training of relation extraction systems (Angeli et al., 2015). As a result most systems fail to make correct extractions even in apparently straightforward cases like Figure 1,

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 35–45
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Example	**Entity Types & Label**
Carey will succeed **Cathleen P. Black**, who held the position for 15 years and will take on a new role as **chairwoman** of Hearst Magazines, the company said.	**Types:** PERSON/TITLE **Relation:** *per:title*
Irene Morgan Kirkaldy, who was born and reared in **Baltimore**, lived on Long Island and ran a child-care center in Queens with her second husband, Stanley Kirkaldy.	**Types:** PERSON/CITY **Relation:** *per:city_of_birth*
Pandit worked at the brokerage Morgan Stanley for about 11 years until 2005, when he and some Morgan Stanley colleagues quit and later founded the hedge fund **Old Lane Partners**.	**Types:** ORGANIZATION/PERSON **Relation:** *org:founded_by*
Baldwin declined further comment, and said JetBlue chief **executive** Dave Barger was unavailable.	**Types:** PERSON/TITLE **Relation:** *no_relation*

Table 1: Sampled examples from the TACRED dataset. Subject entities are highlighted in blue and object entities are highlighted in red.

where the best system at the NIST TAC Knowledge Base Population (TAC KBP) 2015 evaluation failed to recognize the relation between *Penner* and *Dillman*.[1] Consequently most automatic systems continue to make heavy use of hand-written rules or patterns because it has been hard for machine learning systems to achieve adequate precision or to generalize as well across text types. We believe machine learning approaches have suffered from two key problems: (1) the models used have been insufficiently tailored to relation extraction, and (2) there has been insufficient annotated data available to satisfy the training of data-hungry models, such as deep learning models.

This work addresses both of these problems. We propose a new, effective neural network sequence model for relation classification. Its architecture is better customized for the slot filling task: the word representations are augmented by extra distributed representations of word position relative to the subject and object of the putative relation. This means that the neural attention model can effectively exploit the combination of semantic similarity-based attention and position-based attention. Secondly, we markedly improve the availability of supervised training data by using Mechanical Turk crowd annotation to produce a large supervised training dataset (Table 1), suitable for the common relations between people, organizations and locations which are used in the TAC KBP evaluations. We name this dataset the **TAC R**elation **E**xtraction **D**ataset (TACRED), and will make it available through the Linguistic Data Consortium (LDC) in order to respect copyrights on the underlying text.

Combining these two gives a system with markedly better slot filling performance. This is shown not only for a relation classification task on the crowd-annotated data but also for the incorporation of the resulting classifiers into a complete cold start knowledge base population system. On TACRED, our system achieves a relation classification F_1 score that is 7.9% higher than that of a strong feature-based classifier, and 3.5% higher than that of the best previous neural architecture that we re-implemented. When this model is used in concert with a pattern-based system on the TAC KBP 2015 Cold Start Slot Filling evaluation data, the system achieves an F_1 score of 26.7%, which exceeds the previous state-of-the-art by 4.5% absolute. While this performance certainly does not solve the knowledge base population problem – achieving sufficient recall remains a formidable challenge – this is nevertheless notable progress.

2 A Position-aware Neural Sequence Model Suitable for Relation Extraction

Existing work on neural relation extraction (e.g., Zeng et al., 2014; Nguyen and Grishman, 2015; Zhou et al., 2016) has focused on convolutional neural networks (CNNs), recurrent neural networks (RNNs), or their combination. While these models generally work well on the datasets they are tested on, as we will show, they often fail to generalize to the longer sentences that are common in real-world text (such as in TAC KBP).

We believe that existing model architectures suffer from two problems: (1) Although modern sequence models such as Long Short-Term Memory (LSTM) networks have gating mechanisms to control the relative influence of each individual word to the final sentence representation (Hochreiter and Schmidhuber, 1997), these controls are not explicitly conditioned on the entire sentence being classified; (2) Most existing work either

[1] Note: former spouses count as spouses in the ontology.

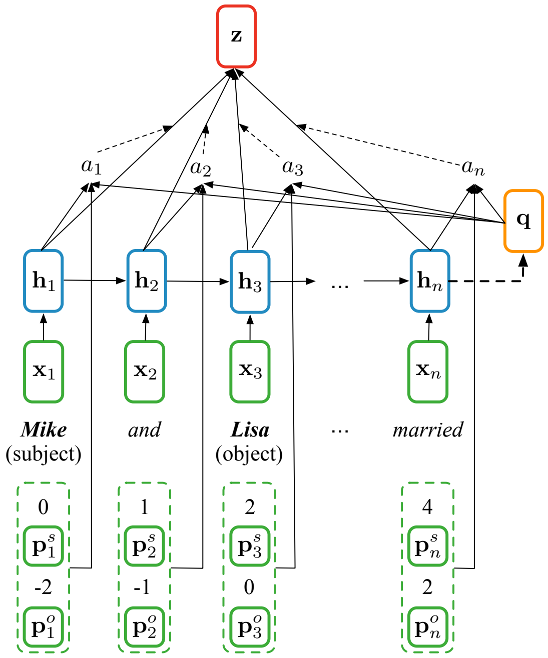


Figure 2: Our proposed position-aware neural sequence model. The model is shown with an example sentence *Mike and Lisa got married.*

does not explicitly model the positions of entities (i.e., subject and object) in the sequence, or models the positions only within a local region.

Here, we propose a new neural sequence model with a **position-aware attention mechanism** over an LSTM network to tackle these challenges. This model can (1) evaluate the relative contribution of each word after seeing the entire sequence, and (2) base this evaluation not only on the semantic information of the sequence, but also on the global positions of the entities within the sequence.

We formalize the relation extraction task as follows: Let $\mathcal{X} = [x_1, ..., x_n]$ denote a sentence, where x_i is the i-th token. A subject entity s and an object entity o are identified in the sentence, corresponding to two non-overlapping consecutive spans: $\mathcal{X}_s = [x_{s_1}, x_{s_1+1}, \dots, x_{s_2}]$ and $\mathcal{X}_o = [x_{o_1}, x_{o_1+1}, \dots, x_{o_2}]$. Given the sentence $\mathcal{X}$ and the positions of s and o, the goal is to predict a relation $r \in \mathcal{R}$ ($\mathcal{R}$ is the set of relations) that holds between s and o or no relation otherwise.

Inspired by the position encoding vectors used in Collobert et al. (2011) and Zeng et al. (2014), we define a position sequence relative to the subject entity $[p^s_1, ..., p^s_n]$, where

$$p^s_i = \begin{cases} i - s_1, & i < s_1 \\ 0, & s_1 \le i \le s_2 \\ i - s_2, & i > s_2 \end{cases} \tag{1}$$

Here s_1, s_2 are the starting and ending indices of the subject entity respectively, and $p^s_i \in \mathbb{Z}$ can be viewed as the relative distance of token x_i to the subject entity. Similarly, we obtain a position sequence $[p^o_1, ..., p^o_n]$ relative to the object entities.

Let $\mathbf{x} = [\mathbf{x}_1, ..., \mathbf{x}_n]$ be word embeddings of the sentence, obtained using an embedding matrix $\mathbf{E}$. Similarly, we obtain position embedding vectors $\mathbf{p}^s = [\mathbf{p}^s_1, ..., \mathbf{p}^s_n]$ and $\mathbf{p}^o = [\mathbf{p}^o_1, ..., \mathbf{p}^o_n]$ using a shared position embedding matrix $\mathbf{P}$ respectively. Next, as shown in Figure 2, we obtain hidden state representations of the sentence by feeding $\mathbf{x}$ into an LSTM:

$$\{\mathbf{h}_1, ..., \mathbf{h}_n\} = \text{LSTM}(\{\mathbf{x}_1, ..., \mathbf{x}_n\}) \tag{2}$$

We define a *summary* vector $\mathbf{q} = \mathbf{h}_n$ (i.e., the output state of the LSTM). This summary vector encodes information about the entire sentence. Then for each hidden state $\mathbf{h}_i$, we calculate an attention weight a_i as:

$$\begin{aligned} u_i = \mathbf{v}^\top \tanh(&\mathbf{W}_h\mathbf{h}_i + \mathbf{W}_q\mathbf{q} + \\ &\mathbf{W}_s\mathbf{p}^s_i + \mathbf{W}_o\mathbf{p}^o_i) \end{aligned} \tag{3}$$

$$a_i = \frac{\exp(u_i)}{\sum_{j=1}^n \exp(u_j)} \tag{4}$$

Here $\mathbf{W}_h, \mathbf{W}_q \in \mathbb{R}^{d_a \times d}$, $\mathbf{W}_s, \mathbf{W}_o \in \mathbb{R}^{d_a \times d_p}$ and $\mathbf{v} \in \mathbb{R}^{d_a}$ are learnable parameters of the network, where d is the dimension of hidden states, d_p is the dimension of position embeddings, and d_a is the size of attention layer. Additional parameters of the network include embedding matrices $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$ and $\mathbf{P} \in \mathbb{R}^{(2L-1) \times d_p}$, where $\mathcal{V}$ is the vocabulary and L is the maximum sentence length.

We regard attention weight a_i as the relative contribution of the specific word to the sentence representation. The final sentence representation $\mathbf{z}$ is computed as:

$$\mathbf{z} = \sum\nolimits_{i=1}^n a_i\mathbf{h}_i \tag{5}$$

$\mathbf{z}$ is later fed into a fully-connected layer followed by a softmax layer for relation classification.

Note that our model significantly differs from the attention mechanism in Bahdanau et al. (2015) and Zhou et al. (2016) in our use of the summary vector and position embeddings, and the way our attention weights are computed. An intuitive way to understand the model is to view the attention calculation as a selection process, where the goal is to select relevant contexts over irrelevant ones.

Dataset	# Rel.	# Ex.	% Neg.
SemEval-2010 Task 8	19	10,717	17.4%
ACE 2003–2004	24	16,771	N/A
TACRED	42	119,474	78.7%

Table 2: A comparison of existing datasets and our proposed TACRED dataset. % Neg. denotes the percentage of negative examples (no relation).

Here the summary vector ($\mathbf{q}$) helps the model to base this selection on the semantic information of the entire sentence (rather than on each word only), while the position vectors ($\mathbf{p}_i^s$ and $\mathbf{p}_i^o$) provides important spatial information between each word and the entities.

3 The TAC Relation Extraction Dataset

Previous research has shown that slot filling systems can greatly benefit from supervised data. For example, Angeli et al. (2014b) showed that even a small amount of supervised data can boost the end-to-end F_1 score by 3.9% on the TAC KBP tasks. However, existing relation extraction datasets such as the SemEval-2010 Task 8 dataset (Hendrickx et al., 2009) and the Automatic Content Extraction (ACE) (Strassel et al., 2008) dataset are less useful for this purpose. This is mainly because: (1) these datasets are relatively small for effectively training high-capacity models (see Table 2), and (2) they capture very different types of relations. For example, the SemEval dataset focuses on semantic relations (e.g., *Cause-Effect*, *Component-Whole*) between two nominals.

One can further argue that it is easy to obtain a large amount of training data using distant supervision (Mintz et al., 2009). In practice, however, due to the large amount of noise in the induced data, training relation extractors that perform well becomes very difficult. For example, Riedel et al. (2010) show that up to 31% of the distantly supervised labels are wrong when creating training data from aligning Freebase to newswire text.

To tackle these challenges, we collect a large supervised dataset TACRED, targeted towards the TAC KBP relations.

Data collection. We create TACRED based on query entities and annotated system responses in the yearly TAC KBP evaluations. In each year of the TAC KBP evaluation (2009–2015), 100 entities (people or organizations) are given as queries, for which participating systems should find associated relations and object entities. We make use of Mechanical Turk to annotate each sentence in the source corpus that contains one of these query entities. For each sentence, we ask crowd workers to annotate both the subject and object entity spans and the relation types.

Data Split	# Ex.	Years
Train	75,050	2009–2012
Dev	25,764	2013
Test	18,660	2014

Table 3: Statistics on TACRED: number of examples and the source of each portion.

Dataset stratification. In total we collect 119,474 examples. We stratify TACRED across years in which the TAC KBP challenge was run, and use examples corresponding to query entities from 2009 to 2012 as training split, 2013 as development split, and 2014 as test split. We reserve the TAC KBP 2015 evaluation data for running slot filling evaluations, as presented in Section 4. Detailed statistics are given in Table 3.

Discussion. Table 1 presents sampled examples from TACRED. Compared to existing datasets, TACRED has four advantages. First, it contains an order of magnitude more relation instances (Table 2), enabling the training of expressive models. Second, we reuse the entity and relation types of the TAC KBP tasks. We believe these relation types are of more interest to downstream applications. Third, we fully annotate all negative instances that appear in our data collection process, to ensure that models trained on TACRED are not biased towards predicting false positives on real-world text. Lastly, the average sentence length in TACRED is 36.2, compared to 19.1 in the SemEval dataset, reflecting the complexity of contexts in which relations occur in real-world text.

Due to space constraints, we describe the data collection and validation process, system interfaces, and more statistics and examples of TACRED in the supplementary material. We will make TACRED publicly available through the LDC.

4 Experiments

In this section we evaluate the effectiveness of our proposed model and TACRED on improving slot filling systems. Specifically, we run two sets of experiments: (1) we evaluate model performance on the relation extraction task using TACRED, and (2) we evaluate model performance on the TAC KBP 2015 cold start slot filling task, by training the models on TACRED.

4.1 Baseline Models

We compare our model against the following baseline models for relation extraction and slot filling:

TAC KBP 2015 winning system. To judge our proposed model against a strong baseline, we compare against Stanford's top performing system on the TAC KBP 2015 cold start slot filling task (Angeli et al., 2015). At the core of this system are two relation extractors: a pattern-based extractor and a logistic regression (LR) classifier. The pattern-based system uses a total of 4,528 surface patterns and 169 dependency patterns. The logistic regression model was trained on approximately 2 million bootstrapped examples (using a small annotated dataset and high-precision pattern system output) that are carefully tuned for TAC KBP slot filling evaluation. It uses a comprehensive feature set similar to the MIML-RE system for relation extraction (Surdeanu et al., 2012), including lemmatized n-grams, sequence NER tags and POS tags, positions of entities, and various features over dependency paths, etc.

Convolutional neural networks. We follow the 1-dimensional CNN architecture by Nguyen and Grishman (2015) for relation extraction. This model learns a representation of the input sentence, by first running a series of convolutional operations on the sentence with various filters, and then feeding the output into a max-pooling layer to reduce the dimension. The resulting representation is then fed into a fully-connected layer followed by a softmax layer for relation classification. As an extension, positional embeddings are also introduced into this model to better capture the relative position of each word to the subject and object entities and were shown to achieve improved results. We use "CNN-PE" to represent the CNN model with positional embeddings.

Dependency-based recurrent neural networks. In dependency-based neural models, shortest dependency paths between entities are often used as input to the neural networks. The intuition is to eliminate tokens that are potentially less relevant to the classification of the relation. For the example in Figure 1, the shortest dependency path between the two entities is:

$$\text{[Penner]} \leftarrow \text{survived} \rightarrow \text{brother}$$
$$\rightarrow \text{wife} \rightarrow \text{[Lisa Dillman]}$$

We follow the SDP-LSTM model proposed by Xu et al. (2015b). In this model, each shortest dependency path is divided into two separate sub-paths from the subject entity and the object entity to the lowest common ancestor node. Each sub-path is fed into an LSTM network, and the resulting hidden units at each word position are passed into a max-over-time pooling layer to form the output of this sub-path. Outputs from the two sub-paths are then concatenated to form the final representation.

In addition to the above models, we also compare our proposed model against an LSTM sequence model without attention mechanism.

4.2 Implementation Details

We map words that occur less than 2 times in the training set to a special <*UNK*> token. We use the pre-trained GloVe vectors (Pennington et al., 2014) to initialize word embeddings. For all the LSTM layers, we find that 2-layer stacked LSTMs generally work better than one-layer LSTMs. We minimize cross-entropy loss over all 42 relations using *AdaGrad* (Duchi et al., 2011). We apply Dropout with $p = 0.5$ to CNNs and LSTMs. During training we also find a word dropout strategy to be very effective: we randomly set a token to be <*UNK*> with a probability p. We set p to be 0.06 for the SDP-LSTM model and 0.04 for all other models.

Entity masking. We replace each subject entity in the original sentence with a special <*NER*>-*SUBJ* token where <*NER*> is the corresponding NER signature of the subject as provided in TACRED. We do the same processing for object entities. This processing step helps (1) provide a model with entity type information, and (2) prevent a model from overfitting its predictions to specific entities.

Multi-channel augmentation. Instead of using only word vectors as input to the network, we augment the input with part-of-speech (POS) and named entity recognition (NER) embeddings. We run Stanford CoreNLP (Manning et al., 2014) to obtain the POS and NER annotations.

	Model	P	R	F_1
Traditional	Patterns	**85.3**	23.4	36.8
	LR	72.0	47.8	57.5
	LR + Patterns	71.4	50.1	58.9
Neural	CNN	72.1	50.3	59.2
	CNN-PE	68.2	55.4	61.1
	SDP-LSTM	62.0	54.8	58.2
	LSTM	61.4	61.7	61.5
	Our model	67.7	**63.2**	**65.4**
	Ensemble	69.4	**64.8**	**67.0**

Table 4: Model performance on the test set of TACRED, micro-averaged over instances. LR = Logistic Regression.

We describe our model hyperparameters and training in detail in the supplementary material.

4.3 Evaluation on TACRED

We first evaluate all models on TACRED. We train each model for 5 separate runs with independent random initializations. For each run we perform early stopping using the dev set. We then select the run (among 5) that achieves the *median* F_1 score on the dev set, and report its test set performance.

Table 4 summarizes our results. We observe that all neural models achieve higher F_1 scores than the logistic regression and patterns systems, which demonstrates the effectiveness of neural models for relation extraction. Although positional embeddings help increase the F_1 by around 2% over the plain CNN model, a simple (2-layer) LSTM model performs surprisingly better than CNN and dependency-based models. Lastly, our proposed position-aware mechanism is very effective and achieves an F_1 score of 65.4%, with an absolute increase of 3.9% over the best baseline neural model (LSTM) and 7.9% over the baseline logistic regression system. We also run an ensemble of our position-aware attention model which takes majority votes from 5 runs with random initializations and it further pushes the F_1 score up by 1.6%.

We find that different neural architectures show a different balance between precision and recall. CNN-based models tend to have higher precision; RNN-based models have better recall. This can be explained by noting that the filters in CNNs are essentially a form of "fuzzy n-gram patterns".

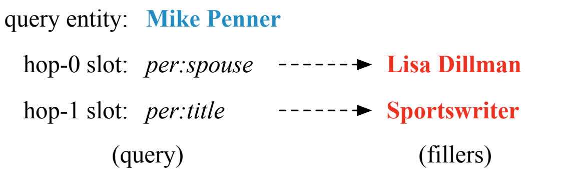


Figure 3: An example query and corresponding fillers in the TAC KBP cold start slot filling task.

4.4 Evaluation on TAC KBP Slot Filling

Second, we evaluate the slot filling performance of all models using the TAC KBP 2015 cold start slot filling task (Ellis et al., 2015). In this task, about 50k newswire and Web forum documents are selected as the evaluation corpus. A slot filling system is asked to answer a series of queries with two-hop slots (Figure 3): The first slot asks about fillers of a relation with the query entity as the subject (Mike Penner), and we term this a **hop-0** slot; the second slot asks about fillers with the system's hop-0 output as the subject, and we term this a **hop-1** slot. System predictions are then evaluated against gold annotations, and micro-averaged precision, recall and F_1 scores are calculated at the hop-0 and hop-1 levels. Lastly **hop-all** scores are calculated by combining hop-0 and hop-1 scores.[2]

Evaluating relation extraction systems on slot filling is particularly challenging in that: (1) End-to-end cold start slot filling scores conflate the performance of all modules in the system (i.e., entity recognizer, entity linker and relation extractor). (2) Errors in hop-0 predictions can easily propagate to hop-1 predictions. To fairly evaluate each relation extraction model on this task, we use Stanford's 2015 slot filling system as our basic pipeline.[3] It is a very strong baseline specifically tuned for TAC KBP evaluation and ranked top in the 2015 evaluation. We then plug in the corresponding relation extractor trained on TACRED, keeping all other modules unchanged.

Table 5 presents our results. We find that: (1) by only training our logistic regression model on TACRED (in contrast to on the 2 million bootstrapped examples used in the 2015 Stanford system) and combining it with patterns, we obtain a higher hop-0 F_1 score than the 2015 Stanford sys-

[2] In the TAC KBP cold start slot filling evaluation, a hop-1 slot is transferred to a pseudo-slot which is treated equally as a hop-0 slot. Hop-all precision, recall and F1 are then calculated by combining these pseudo-slot predictions and hop-0 predictions.

[3] This system uses the fine-grained NER system in Stanford CoreNLP (Manning et al., 2014) for entity detection and the Illinois Wikifier (Ratinov et al., 2011) for entity linking.

Model	Hop-0			Hop-1			Hop-all		
	P	R	F_1	P	R	F_1	P	R	F_1
Patterns	**63.8**	17.7	27.7	**49.3**	8.6	14.7	**58.9**	13.3	21.8
LR	36.6	21.9	27.4	15.1	10.1	12.2	25.6	16.3	19.9
+ Patterns (2015 winning system)	37.5	24.5	29.7	16.5	12.8	14.4	26.6	19.0	22.2
LR trained on TACRED	32.7	20.6	25.3	7.9	9.5	8.6	16.8	15.3	16.0
+ Patterns	36.5	26.5	30.7	11.0	15.3	12.8	20.1	21.2	20.6
Our model	39.0	28.9	33.2	17.7	13.9	15.6	28.2	21.5	24.4
+ Patterns	40.2	**31.5**	**35.3**	19.4	**16.5**	**17.8**	29.7	**24.2**	**26.7**

Table 5: Model performance on TAC KBP 2015 slot filling evaluation, micro-averaged over queries. Hop-0 scores are calculated on the simple single-hop slot filling results; hop-1 scores are calculated on slot filling results chained on systems' hop-0 predictions; hop-all scores are calculated based on the combination of the two. LR = logistic regression.

Model	Dev F_1
Final Model	**66.22**
– Position-aware attention	65.12
– Attention	64.71
– Pre-trained embeddings	65.34
– Word dropout	65.69
– All above	63.60

Table 6: An ablation test of our position-aware attention model, evaluated on TACRED dev set. Scores are median of 5 models.

tem, and a similar hop-all F_1; (2) our proposed position-aware attention model substantially outperforms the 2015 Stanford system on all hop-0, hop-1 and hop-all F_1 scores. Combining it with the patterns, we achieve a hop-all F_1 of 26.7%, an absolute improvement of 4.5% over the previous state-of-the-art result.

4.5 Analysis

Model ablation. Table 6 presents the results of an ablation test of our position-aware attention model on the development set of TACRED. The entire attention mechanism contributes about 1.5% F_1, where the position-aware term in Eq. (3) alone contributes about 1% F_1 score.

Impact of negative examples. Figure 4 shows how the slot filling evaluation scores change as we change the amount of negative (i.e., *no_relation*) training data provided to our proposed model. We find that: (1) At hop-0 level, precision increases as we provide more negative examples, while recall stays almost unchanged. F_1 score keeps increasing. (2) At hop-all level, F_1 score increases by about 10% as we change the amount of negative examples from 20% to 100%.

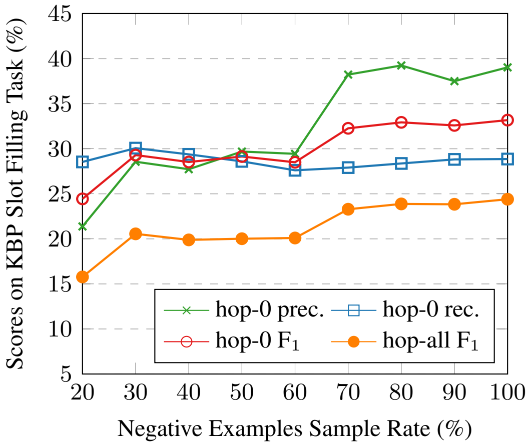


Figure 4: Change of slot filling hop-0 and hop-all scores as number of negative training examples changes. 100% is with all the negative examples included in the training set; the left side scores have positives and negatives roughly balanced.

Performance by sentence length. Figure 5 shows performance on varying sentence lengths. We find that: (1) Performance of all models degrades substantially as the sentences get longer. (2) Compared to the baseline Logistic Regression model, all neural models handle long sentences better. (3) Compared to CNN-PE model, RNN-based models are more robust on long sentences, and notably SDP-LSTM model is least sensitive to sentence length. (4) Our proposed model achieves equal or better results on sentences of all lengths, except for sentences with more than 60 tokens where SDP-LSTM model achieves the best result.

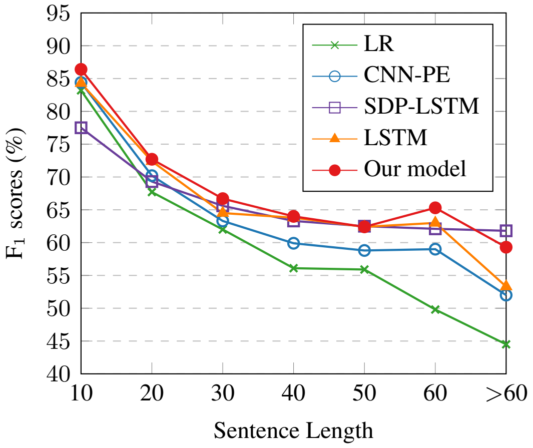


Figure 5: TACRED development set F_1 scores for sentences of varying lengths.

Improvement by slot types. We calculate the F_1 score for each slot type and compare the improvement from using our proposed model across slot types. When compared with the CNN-PE model, our position-aware attention model achieves improved F_1 scores on 30 out of the 41 slot types, with the top 5 slot types being *org:members*, *per:country_of_death*, *org:shareholders*, *per:children* and *per:religion*. When compared with SDP-LSTM model, our model achieves improved F_1 scores on 26 out of the 41 slot types, with the top 5 slot types being *org:political/religious_affiliation*, *per:country_of_death*, *org:alternate_names*, *per:religion* and *per:alternate_names*. We observe that slot types with relatively sparse training examples tend to be improved by using the position-aware attention model.

Attention visualization. Lastly, Figure 6 shows the visualization of attention weights assigned by our model on sampled sentences from the development set. We find that the model learns to pay more attention to words that are informative for the relation (e.g., "graduated from", "niece" and "chairman"), though it still makes mistakes (e.g., "refused to name the three"). We also observe that the model tends to put a lot of weight onto object entities, as the object NER signatures are very informative to the classification of relations.

5 Related Work

Relation extraction. There are broadly three main lines of work on relation extraction: first, fully-supervised approaches (Zelenko et al., 2003; Bunescu and Mooney, 2005), where a statistical classifier is trained on an annotated dataset; second, distant supervision (Mintz et al., 2009; Surdeanu et al., 2012), where a training set is formed by projecting the relations in an existing knowledge base onto textual instances that contain the entities that the relation connects; and third, Open IE (Fader et al., 2011; Mausam et al., 2012), which views its goal as producing subject-relation-object triples and expressing the relation in text.

Slot filling and knowledge base population. The most widely-known effort to evaluate slot filling and KBP systems is the yearly TAC KBP slot filling tasks, starting from 2009 (McNamee and Dang, 2009). Participants in slot filling tasks usually make use of hybrid systems that combine patterns, Open IE, distant supervision and supervised systems for relation extraction (Kisiel et al., 2015; Finin et al., 2015; Zhang et al., 2016).

Datasets for relation extraction. Popular general-domain datasets include the ACE dataset (Strassel et al., 2008) and the SemEval-2010 task 8 dataset (Hendrickx et al., 2009). In addition, the BioNLP Shared Tasks (Kim et al., 2009) are yearly efforts on creating datasets and evaluations for biomedical information extraction systems.

Deep learning models for relation extraction. Many deep learning models have been proposed for relation extraction, with a focus on end-to-end training using CNNs (Zeng et al., 2014; Nguyen and Grishman, 2015) and RNNs (Zhang et al., 2015). Other popular approaches include using CNN or RNN over dependency paths between entities (Xu et al., 2015a,b), augmenting RNNs with different components (Xu et al., 2016; Zhou et al., 2016), and combining RNNs and CNNs (Vu et al., 2016; Wang et al., 2016). Adel et al. (2016) compares the performance of CNN models against traditional approaches on slot filling using a portion of the TAC KBP evaluation data.

6 Conclusion

We introduce a state-of-the-art position-aware neural sequence model for relation extraction, as well as TACRED, a large-scale, crowd-sourced dataset that is orders of magnitude larger than previous relation extraction datasets. Our proposed model outperforms a strong feature-based classifier and all baseline neural models. In combination with the new dataset, it improves the state-of-the-

Sampled Sentences	Predicted Labels
PER-SUBJ graduated from North Korea 's elite Kim Il Sung University and ORG-OBJ ORG-OBJ .	*per:schools_attended*
The cause was a heart attack following a case of pneumonia , said PER-SUBJ 's niece , PER-OBJ PER-OBJ .	*per:other_family*
Independent ORG-SUBJ ORG-SUBJ ORG-SUBJ (ECC) chairman PER-OBJ PER-OBJ refused to name the three , saying they would be identified when the final list of candidates for the august 20 polls is published on Friday .	*org:top_members/employees*

Figure 6: Sampled sentences from the TACRED development set, with words highlighted according to the attention weights produced by our best model.

art hop-all F_1 on the TAC KBP 2015 slot filling task by 4.5% absolute.

Acknowledgments

We thank the anonymous reviewers for their helpful suggestions. We gratefully acknowledge the support of the Allen Institute for Artificial Intelligence and the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract No. FA8750-13-2-0040. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Heike Adel, Benjamin Roth, and Hinrich Schütze. 2016. Comparing convolutional neural networks to traditional models for slot filling. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT)*.

Gabor Angeli, Sonal Gupta, Melvin Johnson Premkumar, Christopher D. Manning, Christopher Ré, Julie Tibshirani, Jean Y. Wu, Sen Wu, and Ce Zhang. 2014a. Stanford's distantly supervised slot filling systems for KBP 2014. In *Text Analysis Conference (TAC) Proceedings 2014*.

Gabor Angeli, Julie Tibshirani, Jean Y. Wu, and Christopher D. Manning. 2014b. Combining distant and partial supervision for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Gabor Angeli, Victor Zhong, Danqi Chen, Arun Chaganty, Jason Bolton, Melvin Johnson Premkumar, Panupong Pasupat, Sonal Gupta, and Christopher D Manning. 2015. Bootstrapped self training for knowledge base population. In *Text Analysis Conference (TAC) Proceedings 2015*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (EMNLP 2005)*, pages 724–731.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.

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

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Joe Ellis, Jeremy Getman, Dana Fore, Neil Kuster, Zhiyi Song, Ann Bies, and Stephanie Strassel. 2015. Overview of linguistic resources for the TAC KBP 2015 evaluations: Methodologies and results. In *Text Analysis Conference (TAC) Proceedings 2015*, pages 16–17.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pages 1535–1545.

Tim Finin, Dawn Lawrie, Paul McNamee, James Mayfield, Douglas Oard, Nanyun Peng, Ning Gao, Yiu-Chang Lin, Joshi MacKin, and Tim Dowd. 2015.

HLTCOE participation in TAC KBP 2015: Cold start and TEDL. In *Text Analysis Conference (TAC) Proceedings 2015*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99.

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

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of BioNLP'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pages 1–9.

Bryan Kisiel, Bill McDowell, Matt Gardner, Ndapandula Nakashole, Emmanouil A Platanios, Abulhair Saparov, Shashank Srivastava, Derry Wijaya, and Tom Mitchell. 2015. CMUML System for KBP 2015 cold start slot filling. In *Text Analysis Conference (TAC) Proceedings 2015*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the TAC 2009 knowledge base population track. In *Text Analysis Conference (TAC) Proceedings 2009*, volume 17, pages 111–113.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT)*, pages 39–48.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, volume 14, pages 1532–1543.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL 2011)*, pages 1375–1384.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. *Machine learning and knowledge discovery in databases*, pages 148–163.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Stephanie Strassel, Mark A Przybocki, Kay Peterson, Zhiyi Song, and Kazuaki Maeda. 2008. Linguistic resources and evaluation techniques for evaluation of cross-document automatic content extraction. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT)*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention CNNs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with data augmentation. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1785–1794.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3:1083–1106.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2014)*, pages 2335–2344.

Dongxu Zhang, Dong Wang, and Rong Liu. 2015. Relation classification via recurrent neural network. Technical report, CSLT 20150024, Tsinghua University.

Yuhao Zhang, Arun Chaganty, Ashwin Paranjape, Danqi Chen, Jason Bolton, Peng Qi, and Christopher D. Manning. 2016. Stanford at TAC KBP 2016: Sealing pipeline leaks and understanding chinese. In *Text Analysis Conference (TAC) Proceedings 2016*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, page 207.

Heterogeneous Supervision for Relation Extraction: A Representation Learning Approach

Liyuan Liu[†*] Xiang Ren[†*] Qi Zhu[†] Huan Gui[†] Shi Zhi[†] Heng Ji[♯] Jiawei Han[†]
[†] University of Illinois at Urbana-Champaign, Urbana, IL, USA
[♯] Computer Science Department, Rensselaer Polytechnic Institute, USA
[†]{ll2, xren7, qiz3, huangui2, shizhi2, hanj}@illinois.edu [♯]{jih}@rpi.edu

Abstract

Relation extraction is a fundamental task in information extraction. Most existing methods have heavy reliance on annotations labeled by human experts, which are costly and time-consuming. To overcome this drawback, we propose a novel framework, REHESSION, to conduct relation extractor learning using annotations from heterogeneous information source, e.g., knowledge base and domain heuristics. These annotations, referred as *heterogeneous supervision*, often conflict with each other, which brings a new challenge to the original relation extraction task: how to infer the true label from noisy labels for a given instance. Identifying context information as the backbone of both relation extraction and true label discovery, we adopt embedding techniques to learn the distributed representations of context, which bridges all components with mutual enhancement in an iterative fashion. Extensive experimental results demonstrate the superiority of REHESSION over the state-of-the-art.

1 Introduction

One of the most important tasks towards text understanding is to detect and categorize semantic relations between two entities in a given context. For example, in Fig. 1, with regard to the sentence of c_1, relation between *Jesse James* and *Missouri* should be categorized as `died_in`. With accurate identification, relation extraction systems can provide essential support for many applications. One example is question answering, regarding a specific question, relation among entities can provide valuable information, which helps to seek better answers (Bao et al., 2014). Similarly, for medical science literature, relations like protein-protein interactions (Fundel et al., 2007) and gene disease associations (Chun et al., 2006) can be extracted and used in knowledge base population. Additionally, relation extractors can be used in ontology construction (Schutz and Buitelaar, 2005).

Typically, existing methods follow the supervised learning paradigm, and require extensive annotations from domain experts, which are costly and time-consuming. To alleviate such drawback, attempts have been made to build relation extractors with a small set of seed instances or human-crafted patterns (Nakashole et al., 2011; Carlson et al., 2010), based on which more patterns and instances will be iteratively generated by bootstrap learning. However, these methods often suffer from semantic drift (Mintz et al., 2009). Besides, knowledge bases like Freebase have been leveraged to automatically generate training data and provide *distant supervision* (Mintz et al., 2009). Nevertheless, for many domain-specific applications, distant supervision is either non-existent or insufficient (usually less than 25% of relation mentions are covered (Ren et al., 2015; Ling and Weld, 2012)).

Only recently have preliminary studies been developed to unite different supervisions, including knowledge bases and domain specific patterns, which are referred as *heterogeneous supervision*. As shown in Fig. 1, these supervisions often conflict with each other (Ratner et al., 2016). To address these conflicts, data programming (Ratner et al., 2016) employs a generative model, which encodes supervisions as *labeling functions*, and adopts the source consistency assumption: *a source is likely to provide true information with*

[*] Equal contribution.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 46–56
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

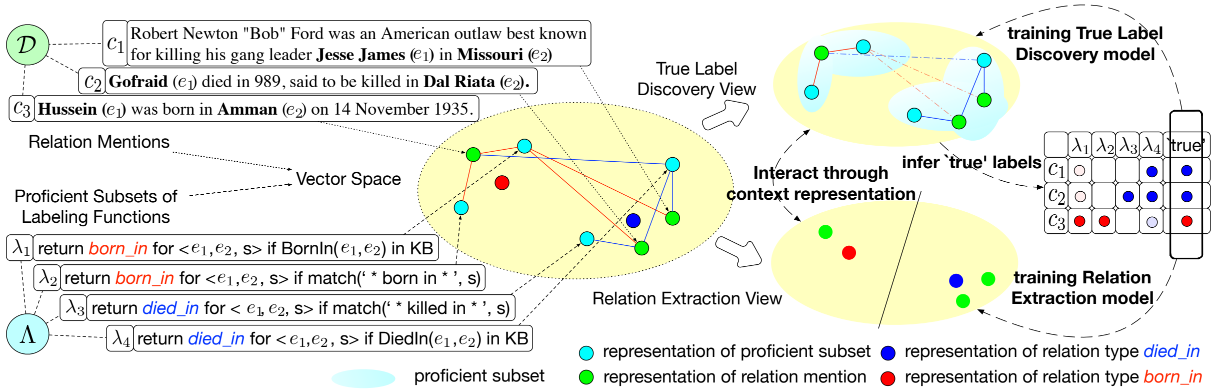

$\mathcal{D}$
c_1 Robert Newton "Bob" Ford was an American outlaw best known for killing his gang leader **Jesse James** (e_1) in **Missouri** (e_2)
c_2 **Gofraid** (e_1) died in 989, said to be killed in **Dal Riata** (e_2).
c_3 **Hussein** (e_1) was born in **Amman** (e_2) on 14 November 1935.
Relation Mentions
Vector Space
Proficient Subsets of Labeling Functions
λ_1 return born_in for <e_1,e_2, s> if BornIn(e_1,e_2) in KB
λ_2 return born_in for <e_1,e_2, s> if match(' * born in * ', s)
λ_3 return died_in for < e_1, e_2, s> if match(' * killed in * ', s)
λ_4 return died_in for <e_1,e_2, s> if DiedIn(e_1,e_2) in KB
Λ
True Label Discovery View
Relation Extraction View
Interact through context representation
infer 'true' labels
training True Label Discovery model
training Relation Extraction model
λ_1 λ_2 λ_3 λ_4 true
c_1 ○ ● ●
c_2 ○ ● ● ●
c_3 ● ● ○ ●
representation of proficient subset
representation of relation mention
representation of relation type died_in
representation of relation type born_in
proficient subset

Figure 1: REHESSION Framework except Extraction and Representation of Text Features

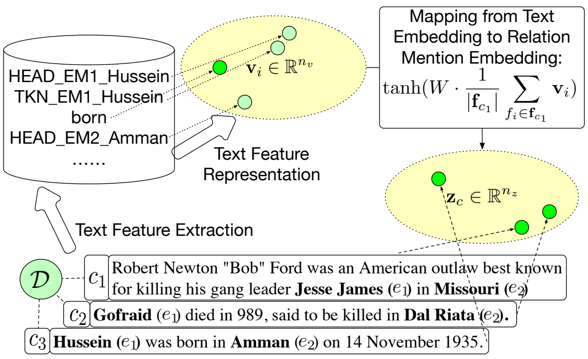

HEAD_EM1_Hussein
TKN_EM1_Hussein
born
HEAD_EM2_Amman
......
$\mathbf{v}_i \in \mathbb{R}^{n_v}$
Text Feature Representation
Mapping from Text Embedding to Relation Mention Embedding:
$\tanh(W \cdot \frac{1}{|\mathbf{f}_{c_1}|} \sum_{f_i \in \mathbf{f}_{c_1}} \mathbf{v}_i)$
$\mathbf{z}_c \in \mathbb{R}^{n_z}$
Text Feature Extraction
$\mathcal{D}$
c_1 Robert Newton "Bob" Ford was an American outlaw best known for killing his gang leader **Jesse James** (e_1) in **Missouri** (e_2)
c_2 **Gofraid** (e_1) died in 989, said to be killed in **Dal Riata** (e_2).
c_3 **Hussein** (e_1) was born in **Amman** (e_2) on 14 November 1935.

Figure 2: Relation Mention Representation

the same probability for all instances. This assumption is widely used in true label discovery literature (Li et al., 2016) to model reliabilities of information sources like crowdsourcing and infer the true label from noisy labels. Accordingly, most true label discovery methods would trust a human annotator on all instances to the same level.

However, labeling functions, unlike human annotators, do not make casual mistakes but follow certain "error routine". Thus, the reliability of a labeling function is not consistent among different pieces of instances. In particular, a labeling function could be more reliable for a certain subset (Varma et al., 2016) (also known as its *proficient subset*) comparing to the rest. We identify these proficient subsets based on context information, only trust labeling functions on these subsets and avoid assuming global source consistency.

Meanwhile, embedding methods have demonstrated great potential in capturing semantic meanings, which also reduce the dimension of overwhelming text features. Here, we present REHESSION, a novel framework capturing context's semantic meaning through representation learning, and conduct both relation extraction and true label discovery in a context-aware manner. Specifically, as depicted in Fig. 1, we embed relation mentions in a low-dimension vector space, where similar relation mentions tend to have similar relation types and annotations. 'True' labels are further inferred based on reliabilities of labeling functions, which are calculated with their proficient subsets' representations. Then, these inferred true labels would serve as supervision for all components, including context representation, true label discovery and relation extraction. Besides, the context representation bridges relation extraction with true label discovery, and allows them to enhance each other.

To the best of our knowledge, the framework proposed here is the first method that utilizes representation learning to provide heterogeneous supervision for relation extraction. The high-quality context representations serve as the backbone of true label discovery and relation extraction. Extensive experiments on benchmark datasets demonstrate significant improvements over the state-of-the-art.

The remaining of this paper is organized as follows. Section 2 gives the definition of relation extraction with heterogeneous supervision. We then present the REHESSION model and the learning algorithm in Section 3, and report our experimental evaluation in Section 4. Finally, we briefly survey related work in Section 5 and conclude this study in Section 6.

2 Preliminaries

In this section, we would formally define relation extraction and heterogeneous supervision, including the format of labeling functions.

2.1 Relation Extraction

Here we conduct relation extraction in *sentence-level* (Bao et al., 2014). For a sentence d, an entity mention is a token span in d which represents an entity, and a relation mention is a triple (e_1, e_2, d) which consists of an ordered entity pair (e_1, e_2) and d. And the relation extraction task is to categorize relation mentions into a given set of relation types $\mathcal{R}$, or Not-Target-Type (`None`) which means the type of the relation mention does not belong to $\mathcal{R}$.

2.2 Heterogeneous Supervision

Similar to (Ratner et al., 2016), we employ labeling functions as basic units to encode supervision information and generate annotations. Since different supervision information may have different proficient subsets, we require each labeling function to encode only one elementary supervision information. Specifically, in the relation extraction scenario, we require each labeling function to only annotate one relation type based on one elementary piece of information, e.g., four examples are listed in Fig. 1.

Notice that knowledge-based labeling functions are also considered to be noisy because relation extraction is conducted in sentence-level, e.g. although `president_of` (*Obama*, *USA*) exists in KB, it should not be assigned with "*Obama* was born in Honolulu, Hawaii, *USA*", since `president_of` is irrelevant to the context.

2.3 Problem Definition

For a POS-tagged corpus $\mathcal{D}$ with detected entities, we refer its relation mentions as $\mathcal{C} = \{c_i = (e_{i,1}, e_{i,2}, d), \forall d \in \mathcal{D}\}$. Our goal is to annotate entity mentions with relation types of interest ($\mathcal{R} = \{r_1, \ldots, r_K\}$) or `None`. We require users to provide heterogeneous supervision in the form of labeling function $\Lambda = \{\lambda_1, \ldots, \lambda_M\}$, and mark the annotations generated by Λ as $\mathcal{O} = \{o_{c,i} | \lambda_i$ generate annotation $o_{c,i}$ for $c \in \mathcal{C}\}$. We record relation mentions annotated by Λ as $\mathcal{C}_l$, and refer relation mentions without annotation as $\mathcal{C}_u$. Then, our task is to train a relation extractor based on $\mathcal{C}_l$ and categorize relation mentions in $\mathcal{C}_u$.

$\mathbf{f}_c$	c's text features set, where $c \in \mathcal{C}$
$\mathbf{v}_i$	text feature embedding for $f_i \in \mathcal{F}$
$\mathbf{z}_c$	relation mention embedding for $c \in \mathcal{C}$
$\mathbf{l}_i$	embedding for λ_i's proficient subset, $\lambda_i \in \Lambda$
$o_{c,i}$	annotation for c, generated by labeling function λ_i
o_c^*	underlying true label for c
$\rho_{c,i}$	identify whether $o_{c,i}$ is correct
$\mathcal{S}_i$	the proficient subset of labeling function λ_i
$s_{c,i}$	identify whether c belongs to λ_i's proficient subset
$\mathbf{t}_i$	relation type embedding for $r_i \in \mathcal{R}$

Table 1: Notation Table.

3 The REHession Framework

Here, we present REHession, a novel framework to infer true labels from automatically generated noisy labels, and categorize unlabeled instances into a set of relation types. Intuitively, errors of annotations ($\mathcal{O}$) come from mismatch of contexts, e.g., in Fig. 1, λ_1 annotates c_1 and c_2 with 'true' labels but for mismatched contexts 'killing' and 'killed'. Accordingly, we should only trust labeling functions on *matched* context, e.g., trust λ_1 on c_3 due to its context 'was born in', but not on c_1 and c_2. On the other hand, relation extraction can be viewed as *matching* appropriate relation type to a certain context. These two *matching* processes are closely related and can enhance each other, while context representation plays an important role in both of them.

Framework Overview. We propose a general framework to learn the relation extractor from automatically generated noisy labels. As plotted in Fig. 1, distributed representation of context bridges relation extraction with true label discovery, and allows them to enhance each other. Specifically, it follows the steps below:

1. After being extracted from context, text features are embedded in a low dimension space by representation learning (see Fig. 2);
2. Text feature embeddings are utilized to calculate relation mention embeddings (see Fig. 2);
3. With relation mention embeddings, true labels are inferred by calculating labeling functions' reliabilities in a context-aware manner (see Fig. 1);
4. Inferred true labels would 'supervise' all components to learn model parameters (see Fig. 1).

We now proceed by introducing these components of the model in further details.

3.1 Modeling Relation Mention

As shown in Table 2, we extract abundant lexical features (Ren et al., 2016; Mintz et al., 2009) to characterize relation mentions. However, this abundance also results in the gigantic dimension of original text features ($\sim 10^7$ in our case). In

Feature	**Description**	**Example**
Entity mention (EM) head	Syntactic head token of each entity mention	“*HEAD_EM1_Hussein*”, ...
Entity Mention Token	Tokens in each entity mention	“*TKN_EM1_Hussein*”, ...
Tokens between two EMs	Tokens between two EMs	“*was*”, “*born*”, “*in*”
Part-of-speech (POS) tag	POS tags of tokens between two EMs	“*VBD*”, “*VBN*”, “*IN*”
Collocations	Bigrams in left/right 3-word window of each EM	“*Hussein was*”, “*in Amman*”
Entity mention order	Whether EM 1 is before EM 2	“*EM1_BEFORE_EM2*”
Entity mention distance	Number of tokens between the two EMs	“*EM_DISTANCE_3*”
Body entity mentions numbers	Number of EMs between the two EMs	“*EM_NUMBER_0*”
Entity mention context	Unigrams before and after each EM	“*EM_AFTER_was*”, ...
Brown cluster (learned on $\mathcal{D}$)	Brown cluster ID for each token	“*BROWN_010011001*”, ...

Table 2: Text features $\mathcal{F}$ used in this paper. (“*Hussein*”, “*Amman*”,“***Hussein** was born in **Amman***”) is used as an example.

order to achieve better generalization ability, we represent relation mentions with low dimensional ($\sim 10^2$) vectors. In Fig. 2, for example, relation mention c_3 is first represented as bag-of-features. After learning text feature embeddings, we use the average of feature embedding vectors to derive the embedding vector for c_3.

Text Feature Representation. Similar to other principles of embedding learning, we assume text features occurring in the same contexts tend to have similar meanings (also known as distributional hypothesis(Harris, 1954)). Furthermore, we let each text feature's embedding vector to predict other text features occurred in the same relation mentions or context. Thus, text features with similar meaning should have similar embedding vectors. Formally, we mark text features as $\mathcal{F} = \{f_1, \cdots, f_{|\mathcal{F}|}\}$, record the feature set for $\forall c \in \mathcal{C}$ as $\mathbf{f}_c$, and represent the embedding vector for f_i as $\mathbf{v}_i \in \mathbb{R}^{n_v}$, and we aim to maximize the following log likelihood: $\sum_{c \in \mathcal{C}_l} \sum_{f_i, f_j \in \mathbf{f}_c} \log\, p(f_i|f_j)$, where $p(f_i|f_j) = \exp(\mathbf{v}_i^T \mathbf{v}_j^*) / \sum_{f_k \in \mathcal{F}} \exp(\mathbf{v}_i^T \mathbf{v}_k^*)$.

However, the optimization of this likelihood is impractical because the calculation of $\nabla p(f_i|f_j)$ requires summation over all text features, whose size exceeds 10^7 in our case. In order to perform efficient optimization, we adopt the negative sampling technique (Mikolov et al., 2013) to avoid this summation. Accordingly, we replace the log likelihood with Eq. 1 as below:

$$\mathcal{J}_E = \sum_{\substack{c \in \mathcal{C}_l \\ f_i, f_j \in \mathbf{f}_c}} (\log \sigma(\mathbf{v}_i^T \mathbf{v}_j^*) - \sum_{k=1}^{V} \mathbb{E}_{f_{k'} \sim \hat{P}} [\log \sigma(-\mathbf{v}_i^T \mathbf{v}_{k'}^*)]) \tag{1}$$

where $\hat{P}$ is noise distribution used in (Mikolov et al., 2013), σ is the sigmoid function and V is number of negative samples.

Relation Mention Representation. With text feature embeddings learned by Eq. 1, a naive method to represent relation mentions is to concatenate or average its text feature embeddings. However, text features embedding may be in a different semantic space with relation types. Thus, we directly learn a mapping g from text feature representations to relation mention representations (Van Gysel et al., 2016a,b) instead of simple heuristic rules like concatenate or average (see Fig. 2):

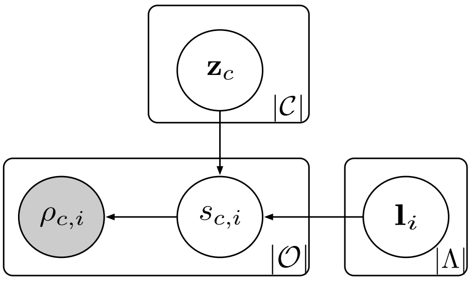


Figure 3: Graphical model of $o_{c,i}$'s correctness

$$\mathbf{z}_c = g(\mathbf{f}_c) = \tanh(W \cdot \frac{1}{|\mathbf{f}_c|} \sum_{f_i \in \mathbf{f}_c} \mathbf{v}_i) \tag{2}$$

where $\mathbf{z}_c$ is the representation of $c \in \mathcal{C}_l$, W is a $n_z \times n_v$ matrix, n_z is the dimension of relation mention embeddings and tanh is the element-wise hyperbolic tangent function.

In other words, we represent bag of text features with their average embedding, then apply linear map and hyperbolic tangent to transform the embedding from text feature semantic space to relation mention semantic space. The non-linear tanh function allows non-linear class boundaries in other components, and also regularize relation mention representation to range $[-1, 1]$ which avoids numerical instability issues.

3.2 True Label Discovery

Because heterogeneous supervision generates labels in a discriminative way, we suppose its errors follow certain underlying principles, i.e., if a

Datasets	NYT	Wiki-KBP
% of `None` in Training	0.6717	0.5552
% of `None` in Test	0.8972	0.8532

Table 3: Proportion of `None` in Training/Test Set

labeling function annotates a instance correctly / wrongly, it would annotate other similar instances correctly / wrongly. For example, λ_1 in Fig. 1 generates wrong annotations for two similar instances c_1, c_2 and would make the same errors on other similar instances. Since context representation captures the semantic meaning of relation mention and would be used to identify relation types, we also use it to identify the mismatch of context and labeling functions. Thus, we suppose for each labeling function λ_i, there exists an proficient subset $\mathcal{S}_i$ on $\mathcal{R}^{n_z}$, containing instances that λ_i can precisely annotate. In Fig. 1, for instance, c_3 is in the proficient subset of λ_1, while c_1 and c_2 are not. Moreover, the generation of annotations are not really random, and we propose a probabilistic model to describe the level of mismatch from labeling functions to real relation types instead of annotations' generation.

As shown in Fig. 3, we assume the indicator of whether c belongs to $\mathcal{S}_i$, $s_{c,i} = \delta(c \in \mathcal{S}_i)$, would first be generated based on context representation

$$p(s_{c,i} = 1|\mathbf{z}_c, \mathbf{l}_i) = p(c \in \mathcal{S}_i) = \sigma(\mathbf{z}_c^T \mathbf{l}_i) \quad (3)$$

Then the correctness of annotation $o_{c,i}$, $\rho_{c,i} = \delta(o_{c,i} = o_c^*)$, would be generated. Furthermore, we assume $p(\rho_{c,i} = 1|s_{c,i} = 1) = \phi_1$ and $p(\rho_{c,i} = 1|s_{c,i} = 0) = \phi_0$ to be constant for all relation mentions and labeling functions.

Because $s_{c,i}$ would not be used in other components of our framework, we integrate out $s_{c,i}$ and write the log likelihood as

$$\mathcal{J}_T = \sum_{o_{c,i} \in \mathcal{O}} \log(\sigma(\mathbf{z}_c^T \mathbf{l}_i)\phi_1^{\delta(o_{c,i}=o_c^*)}(1-\phi_1)^{\delta(o_{c,i}\neq o_c^*)} + (1-\sigma(\mathbf{z}_c^T \mathbf{l}_i))\phi_0^{\delta(o_{c,i}=o_c^*)}(1-\phi_0)^{\delta(o_{c,i}\neq o_c^*)}) \quad (4)$$

Note that o_c^* is a hidden variable but not a model parameter, and $\mathcal{J}_T$ is the likelihood of $\rho_{c,i} = \delta(o_{c,i} = o_c^*)$. Thus, we would first infer $o_c^* = \text{argmax}_{o_c^*} \mathcal{J}_T$, then train the true label discovery model by maximizing $\mathcal{J}_T$.

3.3 Modeling Relation Type

We now discuss the model for identifying relation types based on context representation. For each relation mention c, its representation $\mathbf{z}_c$ implies its relation type, and the distribution of relation type can be described by the soft-max function:

$$p(r_i|\mathbf{z}_c) = \frac{\exp(\mathbf{z}_c^T \mathbf{t}_i)}{\sum_{r_j \in \mathcal{R}\cup\{\texttt{None}\}} \exp(\mathbf{z}_c^T \mathbf{t}_j)} \quad (5)$$

where $\mathbf{t}_i \in \mathbb{R}^{v_z}$ is the representation for relation type r_i. Moreover, with the inferred true label o_c^*, the relation extraction model can be trained as a multi-class classifier. Specifically, we use Eq. 5 to approach the distribution

$$p(r_i|o_c^*) = \begin{cases} 1 & r_i = o_c^* \\ 0 & r_i \neq o_c^* \end{cases} \quad (6)$$

Moreover, we use KL-divergence to measure the dissimilarity between two distributions, and formulate model learning as maximizing $\mathcal{J}_R$:

$$\mathcal{J}_R = -\sum_{c \in \mathcal{C}_l} KL(p(.|\mathbf{z}_c)||p(.|o_c^*)) \quad (7)$$

where $KL(p(.|\mathbf{z}_c)||p(.|o_c^*))$ is the KL-divergence from $p(r_i|o_c^*)$ to $p(r_i|\mathbf{z}_c)$, $p(r_i|\mathbf{z}_c)$ and $p(r_i|o_c^*)$ has the form of Eq. 5 and Eq. 6.

3.4 Model Learning

Based on Eq. 1, Eq. 4 and Eq. 7, we form the joint optimization problem for model parameters as

$$\min_{W,\mathbf{v},\mathbf{v}^*,\mathbf{l},\mathbf{t},o^*} \mathcal{J} = -\mathcal{J}_R - \lambda_1 \mathcal{J}_E - \lambda_2 \mathcal{J}_T$$
$$\text{s.t. } \forall c \in \mathcal{C}_l, o_c^* = \underset{o_c^*}{\text{argmax}}\, \mathcal{J}_T, \mathbf{z}_c = g(\mathbf{f}_c) \quad (8)$$

Collectively optimizing Eq. 8 allows heterogeneous supervision guiding all three components, while these components would refine the context representation, and enhance each other.

In order to solve the joint optimization problem in Eq. 8 efficiently, we adopt the stochastic gradient descent algorithm to update $\{W, \mathbf{v}, \mathbf{v}^*, \mathbf{l}, \mathbf{t}\}$ iteratively, and o_c* is estimated by maximizing $\mathcal{J}_T$ after calculating $\mathbf{z}_c$. Additionally, we apply the widely used dropout techniques (Srivastava et al., 2014) to prevent overfitting and improve generalization performance.

The learning process of REHESSION is summarized as below. In each iteration, we would sample a relation mention c from $\mathcal{C}_l$, then sample c's text features and conduct the text features' representation learning. After calculating the representation of c, we would infer its true label o_c^* based on our true label discovery model, and finally update model parameters based on o_c^*.

3.5 Relation Type Inference

We now discuss the strategy of performing type inference for $\mathcal{C}_u$. As shown in Table 3, the proportion of None in $\mathcal{C}_u$ is usually much larger than in $\mathcal{C}_l$. Additionally, not like other relation types in $\mathcal{R}$, None does not have a coherent semantic meaning. Similar to (Ren et al., 2016), we introduce a heuristic rule: identifying a relation mention as None when (1) our relation extractor predict it as None, or (2) the entropy of $p(.|\mathbf{z}_c)$ over $\mathcal{R}$ exceeds a pre-defined threshold η. The entropy is calculated as $H(p(.|\mathbf{z}_c)) = -\sum_{r_i \in \mathcal{R}} p(r_i|\mathbf{z}_c) log(p(r_i|\mathbf{z}_c))$. And the second situation means based on relation extractor this relation mention is not likely belonging to any relation types in $\mathcal{R}$.

4 Experiments

In this section, we empirically validate our method by comparing to the state-of-the-art relation extraction methods on news and Wikipedia articles.

4.1 Datasets and settings

In the experiments, we conduct investigations on two benchmark datasets from different domains:[1]
NYT (Riedel et al., 2010) is a news corpus sampled from $\sim$ 294k 1989-2007 New York Times news articles. It consists of 1.18M sentences, while 395 of them are annotated by authors of (Hoffmann et al., 2011) and used as test data;
Wiki-KBP utilizes 1.5M sentences sampled from 780k Wikipedia articles (Ling and Weld, 2012) as training corpus, while test set consists of the 2k sentences manually annotated in 2013 KBP slot filling assessment results (Ellis et al., 2012).

For both datasets, the training and test sets partitions are maintained in our experiments. Furthermore, we create validation sets by randomly sampling 10% mentions from each test set and used the remaining part as evaluation sets.

Feature Generation. As summarized in Table 2, we use a 6-word window to extract context features for each entity mention, apply the Stanford CoreNLP tool (Manning et al., 2014) to generate entity mentions and get POS tags for both datasets. Brown clusters(Brown et al., 1992) are derived for each corpus using public implementation[2]. All these features are shared with all compared methods in our experiments.

Kind	Wiki-KBP		NYT	
	#Types	#LF	#Types	#LF
Pattern	13	147	16	115
KB	7	7	25	26

Table 4: Number of labeling functions and the relation types they can annotated w.r.t. two kinds of information

Labeling Functions. In our experiments, labeling functions are employed to encode two kinds of supervision information. One is knowledge base, the other is handcrafted domain-specific patterns. For domain-specific patterns, we manually design a number of labeling functions[3]; for knowledge base, annotations are generated following the procedure in (Ren et al., 2016; Riedel et al., 2010).

Regarding two kinds of supervision information, the statistics of the labeling functions are summarized in Table 4. We can observe that heuristic patterns can identify more relation types for KBP datasets, while for NYT datasets, knowledge base can provide supervision for more relation types. This observation aligns with our intuition that single kind of information might be insufficient while different kinds of information can complement each other.

We further summarize the statistics of annotations in Table 6. It can be observed that a large portion of instances is only annotated as None, while lots of conflicts exist among other instances. This phenomenon justifies the motivation to employ true label discovery model to resolve the conflicts among supervision. Also, we can observe most conflicts involve None type, accordingly, our proposed method should have more advantages over traditional true label discovery methods on the relation extraction task comparing to the relation classification task that excludes None type.

4.2 Compared Methods

We compare REHESSION with below methods:
FIGER (Ling and Weld, 2012) adopts multi-label

[1] Codes and datasets used in this paper can be downloaded at: https://github.com/LiyuanLucasLiu/ReHession.

[2] https://github.com/percyliang/brown-cluster

[3] pattern-based labeling functions can be accessed at: https://github.com/LiyuanLucasLiu/ReHession

Method	Relation Extraction						Relation Classification	
	NYT			Wiki-KBP			NYT	Wiki-KBP
	Prec	Rec	F1	Prec	Rec	F1	Accuracy	Accuracy
NL+FIGER	0.2364	0.2914	0.2606	0.2048	0.4489	0.2810	0.6598	0.6226
NL+BFK	0.1520	0.0508	0.0749	0.1504	0.3543	0.2101	0.6905	0.5000
NL+DSL	0.4150	0.5414	0.4690	0.3301	0.5446	0.4067	0.7954	0.6355
NL+MultiR	0.5196	0.2755	0.3594	0.3012	0.5296	0.3804	0.7059	0.6484
NL+FCM	0.4170	0.2890	0.3414	0.2523	0.5258	0.3410	0.7033	0.5419
NL+CoType-RM	0.3967	0.4049	0.3977	**0.3701**	0.4767	0.4122	0.6485	0.6935
TD+FIGER	0.3664	0.3350	0.3495	0.2650	**0.5666**	0.3582	0.7059	0.6355
TD+BFK	0.1011	0.0504	0.0670	0.1432	0.1935	0.1646	0.6292	0.5032
TD+DSL	0.3704	0.5025	0.4257	0.2950	0.5757	0.3849	0.7570	0.6452
TD+MultiR	**0.5232**	0.2736	0.3586	0.3045	0.5277	0.3810	0.6061	0.6613
TD+FCM	0.3394	0.3325	0.3360	0.1964	0.5645	0.2914	0.6803	0.5645
TD+CoType-RM	0.4516	0.3499	0.3923	0.3107	0.5368	0.3879	0.6409	0.6890
REHESSION	0.4122	**0.5726**	**0.4792**	0.3677	0.4933	**0.4208**	**0.8381**	**0.7277**

Table 5: Performance comparison of relation extraction and relation classification

Dataset	Wiki-KBP	NYT
Total Number of RM	225977	530767
RM annotated as None	100521	356497
RM with conflicts	32008	58198
Conflicts involving None	30559	38756

Table 6: Number of relation mentions (RM), relation mentions annotated as None, relation mentions with conflicting annotations and conflicts involving None

learning with Perceptron algorithm.

BFK (Bunescu and Mooney, 2005) applies bag-of-feature kernel to train a support vector machine;

DSL (Mintz et al., 2009) trains a multi-class logistic classifier[4] on the training data;

MultiR (Hoffmann et al., 2011) models training label noise by multi-instance multi-label learning;

FCM (Gormley et al., 2015) performs compositional embedding by neural language model.

CoType-RM (Ren et al., 2016) adopts partial-label loss to handle label noise and train the extractor.

Moreover, two different strategies are adopted to feed heterogeneous supervision to these methods. The first is to keep all noisy labels, marked as 'NL'. Alternatively, a true label discovery method, Investment (Pasternack and Roth, 2010), is applied to resolve conflicts, which is based on the source consistency assumption and iteratively updates inferred true labels and label functions' reliabilities. Then, the second strategy is to only feed the inferred true labels, referred as 'TD'.

Universal Schemas (Riedel et al., 2013) is proposed to unify different information by calculating a low-rank approximation of the annotations $\mathcal{O}$. It can serve as an alternative of the Investment method, i.e., selecting the relation type with highest score in the low-rank approximation as the true type. But it doesnt explicitly model noise and not fit our scenario very well. Due to the constraint of space, we only compared our method to Investment in most experiments, and Universal Schemas is listed as a baseline in Sec. 4.4. Indeed, it performs similarly to the Investment method.

Evaluation Metrics. For relation classification task, which excludes None type from training / testing, we use the classification accuracy (Acc) for evaluation, and for relation extraction task, precision (Prec), recall (Rec) and F1 score (Bunescu and Mooney, 2005; Bach and Badaskar, 2007) are employed. Notice that both relation extraction and relation classification are conducted and evaluated in *sentence-level* (Bao et al., 2014).

Parameter Settings. Based on the semantic meaning of proficient subset, we set ϕ_2 to $1/|\mathcal{R}\cup\{\texttt{None}\}|$, i.e., the probability of generating right label with random guess. Then we set ϕ_1 to $1-\phi_2$, $\lambda_1 = \lambda_2 = 1$, and the learning rate $\alpha = 0.025$. As for other parameters, they are tuned on the validation sets for each dataset. Similarly, all parameters of compared methods are tuned on validation set, and the parameters achieving highest F1 score are chosen for relation extraction.

[4] We use liblinear package from https://github.com/cjlin1/liblinear

Relation Mention	REHession	Investment & Universal Schemas
Ann Demeulemeester (**born** 1959 , Waregem , *Belgium*) is ...	born-in	None
Raila Odinga was **born** at ..., in *Maseno*, Kisumu District, ...	born-in	None
Ann Demeulemeester (**elected** 1959 , Waregem , *Belgium*) is ...	None	None
Raila Odinga was **examined** at ..., in *Maseno*, Kisumu District, ...	None	None

Table 7: Example output of true label discovery. The first two relation mentions come from Wiki-KBP, and their annotations are {born-in, None}. The last two are created by replacing key words of the first two. Key words are marked as bold and entity mentions are marked as Italics.

Dataset & Method		Prec	Rec	F1	Acc
Wiki-KBP	Ori	**0.3677**	0.4933	**0.4208**	**0.7277**
	TD	0.3032	**0.5279**	0.3850	0.7271
	US	0.3380	0.4779	0.3960	0.7268
NYT	Ori	**0.4122**	**0.5726**	**0.4792**	**0.8381**
	TD	0.3758	0.4887	0.4239	0.7387
	US	0.3573	0.5145	0.4223	0.7362

Table 8: Comparison among REHession (Ori), REHession-US (US) and REHession-TD (TD) on relation extraction and relation classification

4.3 Performance Comparison

Given the experimental setup described above, the averaged evaluation scores in 10 runs of relation classification and relation extraction on two datasets are summarized in Table 5.

From the comparison, it shows that NL strategy yields better performance than TD strategy, since the true labels inferred by Investment are actually wrong for many instances. On the other hand, as discussed in Sec. 4.4, our method introduces context-awareness to true label discovery, while the inferred true label guides the relation extractor achieving the best performance. This observation justifies the motivation of avoiding the source consistency assumption and the effectiveness of proposed true label discovery model.

One could also observe the difference between REHession and the compared methods is more significant on the NYT dataset than on the Wiki-KBP dataset. This observation accords with the fact that the NYT dataset contains more conflicts than KBP dataset (see Table 6), and the intuition is that our method would have more advantages on more conflicting labels.

Among four tasks, the relation classification of Wiki-KBP dataset has highest label quality, i.e. conflicting label ratio, but with least number of training instances. And CoType-RM and DSL reach relatively better performance among all compared methods. CoType-RM performs much better than DSL on Wiki-KBP relation classification task, while DSL gets better or similar performance with CoType-RM on other tasks. This may be because the representation learning method is able to generalize better, thus performs better when the training set size is small. However, it is rather vulnerable to the noisy labels compared to DSL. Our method employs embedding techniques, and also integrates context-aware true label discovery to de-noise labels, making the embedding method rather robust, thus achieves the best performance on all tasks.

4.4 Case Study

Context Awareness of True Label Discovery.

Although Universal Schemas does not adopted the source consistency assumption, but it's conducted in document-level, and is context-agnostic in our sentence-level setting. Similarly, most true label discovery methods adopt the source consistency assumption, which means if they trust a labeling function, they would trust it on all annotations. And our method infers true labels in a context-aware manner, which means we only trust labeling functions on matched contexts.

For example, Investment and Universal Schemas refer None as true type for all four instances in Table 7. And our method infers born-in as the true label for the first two relation mentions; after replacing the matched contexts (**born**) with other words (**elected** and **examined**), our method no longer trusts born-in since the modified contexts are no longer matched, then infers None as the true label. In other words, our proposed method infer the true label in a context aware manner.

Effectiveness of True Label Discovery. We explore the effectiveness of the proposed context-aware true label discovery component by comparing REHession to its variants REHession-TD and REHession-US, which uses Investment or Universal Schemas to resolve conflicts. The averaged evaluation scores are summarized in Table 8. We can observe that REHession significantly outperforms its variants. Since the only difference between REHession and its variants is the model employed to resolve conflicts, this gap verifies the effectiveness of the proposed context-aware true label discovery method.

5 Related Work

5.1 Relation Extraction

Relation extraction aims to detect and categorize semantic relations between a pair of entities. To alleviate the dependency of annotations given by human experts, weak supervision (Bunescu and Mooney, 2007; Etzioni et al., 2004) and distant supervision (Ren et al., 2016) have been employed to automatically generate annotations based on knowledge base (or seed patterns/instances). Universal Schemas (Riedel et al., 2013; Verga et al., 2015; Toutanova et al., 2015) has been proposed to unify patterns and knowledge base, but it's designed for document-level relation extraction, i.e., not to categorize relation types based on a specific context, but based on the whole corpus. Thus, it allows one relation mention to have multiple true relation types; and does not fit our scenario very well, which is sentence-level relation extraction and assumes one instance has only one relation type. Here we propose a more general framework to consolidate heterogeneous information and further refine the true label from noisy labels, which gives the relation extractor potential to detect more types of relations in a more precise way.

Word embedding has demonstrated great potential in capturing semantic meaning (Mikolov et al., 2013), and achieved great success in a wide range of NLP tasks like relation extraction (Zeng et al., 2014; Takase and Inui, 2016; Nguyen and Grishman, 2015). In our model, we employed the embedding techniques to represent context information, and reduce the dimension of text features, which allows our model to generalize better.

5.2 Truth Label Discovery

True label discovery methods have been developed to resolve conflicts among multi-source information under the assumption of source consistency (Li et al., 2016; Zhi et al., 2015). Specifically, in the *spammer-hammer* model (Karger et al., 2011), each source could either be a spammer, which annotates instances randomly; or a hammer, which annotates instances precisely. In this paper, we assume each labeling function would be a hammer on its proficient subset, and would be a spammer otherwise, while the proficient subsets are identified in the embedding space.

Besides data programming, socratic learning (Varma et al., 2016) has been developed to conduct binary classification under heterogeneous supervision. Its true label discovery module supervises the discriminative module in label level, while the discriminative module influences the true label discovery module by selecting a feature subset. Although delicately designed, it fails to make full use of the connection between these modules, i.e., not refine the context representation for classifier. Thus, its discriminative module might suffer from the overwhelming size of text features.

6 Conclusion and Future Work

In this paper, we propose REHESSION, an embedding framework to extract relation under heterogeneous supervision. When dealing with heterogeneous supervisions, one unique challenge is how to resolve conflicts generated by different labeling functions. Accordingly, we go beyond the "source consistency assumption" in prior works and leverage context-aware embeddings to induce proficient subsets. The resulting framework bridges true label discovery and relation extraction with context representation, and allows them to mutually enhance each other. Experimental evaluation justifies the necessity of involving context-awareness, the quality of inferred true label, and the effectiveness of the proposed framework on two real-world datasets.

There exist several directions for future work. One is to apply transfer learning techniques to handle label distributions' difference between training set and test set. Another is to incorporate OpenIE methods to automatically find domain-specific patterns and generate pattern-based labeling functions.

7 Acknowledgments

Research was sponsored in part by the U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), National Science Foundation IIS-1320617, IIS 16-18481, and NSF IIS 17-04532, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies of the U.S. Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

Nguyen Bach and Sameer Badaskar. 2007. A review of relation extraction. *Literature review for Language and Statistics II.*

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. *Cell*, 2(6).

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *ACL*.

Razvan Bunescu and Raymond J Mooney. 2005. Subsequence kernels for relation extraction. In *NIPS*, pages 171–178.

Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110. ACM.

Hong-Woo Chun, Yoshimasa Tsuruoka, Jin-Dong Kim, Rie Shiba, Naoki Nagata, Teruyoshi Hishiki, and Jun'ichi Tsujii. 2006. Extraction of gene-disease relations from medline using domain dictionaries and machine learning. In *Pacific Symposium on Biocomputing*, volume 11, pages 4–15.

Joe Ellis, Xuansong Li, Kira Griffitt, Stephanie Strassel, and Jonathan Wright. 2012. Linguistic resources for 2013 knowledge base population evaluations. In *TAC*.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall:(preliminary results). In *Proceedings of the 13th international conference on World Wide Web*, pages 100–110. ACM.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2007. Relexrelation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Matthew R Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. *arXiv preprint arXiv:1505.02419*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

David R Karger, Sewoong Oh, and Devavrat Shah. 2011. Iterative learning for reliable crowdsourcing systems. In *Advances in neural information processing systems*, pages 1953–1961.

Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2016. A survey on truth discovery. *SIGKDD Explor. Newsl.*, 17(2):1–16.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *AAAI*. Citeseer.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

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

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. 2011. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 227–236. ACM.

Thien Huu Nguyen and Ralph Grishman. 2015. Combining neural networks and log-linear models to improve relation extraction. *arXiv preprint arXiv:1511.05926*.

Jeff Pasternack and Dan Roth. 2010. Knowing what to believe (when you already know something). In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 877–885. Association for Computational Linguistics.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems*, pages 3567–3575.

Xiang Ren, Ahmed El-Kishky, Chi Wang, Fangbo Tao, Clare R Voss, and Jiawei Han. 2015. Clustype: Effective entity recognition and typing by relation phrase-based clustering. In *Proceedings of the 21th*

ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 995–1004. ACM.

Xiang Ren, Zeqiu Wu, Wenqi He, Meng Qu, Clare R Voss, Heng Ji, Tarek F Abdelzaher, and Jiawei Han. 2016. Cotype: Joint extraction of typed entities and relations with knowledge bases. *arXiv preprint arXiv:1610.08763*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *HLT-NAACL*, pages 74–84.

Alexander Schutz and Paul Buitelaar. 2005. Relext: A tool for relation extraction from text in ontology extension. In *International semantic web conference*, volume 2005, pages 593–606. Springer.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Sho Takase and Naoaki Okazaki Kentaro Inui. 2016. Composing distributed representations of relational patterns. In *Proceedings of ACL*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, volume 15, pages 1499–1509.

Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2016a. Learning latent vector spaces for product search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 165–174. ACM.

Christophe Van Gysel, Maarten de Rijke, and Marcel Worring. 2016b. Unsupervised, efficient and semantic expertise retrieval. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1069–1079. International World Wide Web Conferences Steering Committee.

Paroma Varma, Bryan He, Dan Iter, Peng Xu, Rose Yu, Christopher De Sa, and Christopher Ré. 2016. Socratic learning: Correcting misspecified generative models using discriminative models. *arXiv preprint arXiv:1610.08123*.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2015. Multilingual relation extraction using compositional universal schema. *arXiv preprint arXiv:1511.06396*.

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

Shi Zhi, Bo Zhao, Wenzhu Tong, Jing Gao, Dian Yu, Heng Ji, and Jiawei Han. 2015. Modeling truth existence in truth discovery. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1543–1552. ACM.

Integrating Order Information and Event Relation for Script Event Prediction[*]

Zhongqing Wang[†,‡], **Yue Zhang**[‡] and **Ching-Yun Chang**[‡]
[†]Soochow University, Suzhou, China
[‡]Singapore University of Technology and Design, Singapore
wangzq.antony@gmail.com, yue_zhang@sutd.edu.sg,
chang.frannie@gmail.com

Abstract

There has been a recent line of work automatically learning scripts from unstructured texts, by modeling narrative event chains. While the dominant approach group events using event pair relations, LSTMs have been used to encode full chains of narrative events. The latter has the advantage of learning long-range *temporal orders*[1], yet the former is more adaptive to partial orders. We propose a neural model that leverages the advantages of both methods, by using LSTM hidden states as features for event pair modelling. A dynamic memory network is utilized to automatically induce weights on existing events for inferring a subsequent event. Standard evaluation shows that our method significantly outperforms both methods above, giving the best results reported so far.

1 Introduction

Frequently recurring sequences of events in prototypical scenarios, such as visiting a restaurant and driving to work, are a useful source of world knowledge. Two examples are shown in Figure 1, which are different variations of the "restaurant visiting" scenario, where events are partially ordered and can be flexible. Such knowledge is useful for natural language understanding because texts typically do not include event details when mentioning a scenario. For example, the reader is expected to infer that the narrator could have been driving or cycling given the text "I got flat tire". Another typical use of event chain knowledge is to help infer what is likely to happen next given a previous event sequence in a scenario. We investigate the modeling of stereotypical event chains, which is remotely similar to language modeling, but with events being more sparse and flexibly ordered than words.

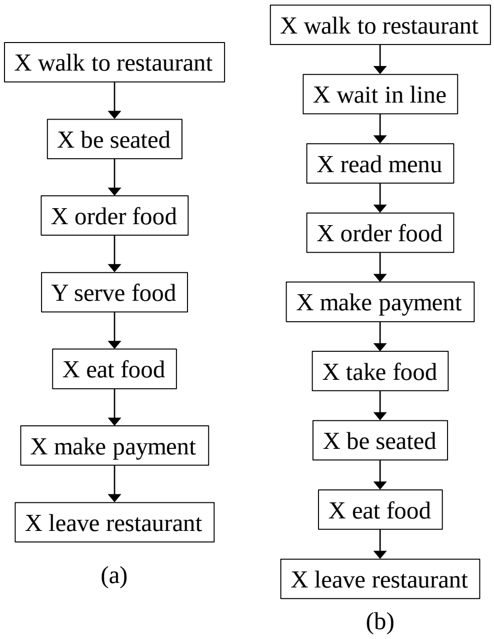


Figure 1: Event sequences for restaurant visiting.

Our work follows a recent line of NLP research on *script learning*. Stereotypical knowledge about partially-ordered *events*, together with their participant *roles* such as "customer", "waiter", and "table", is conventionally referred to as *scripts* (Schank et al., 1977). NLP algorithms have been investigated for automatically inducing scripts from unstructured texts (Mooney and DeJong, 1985; Chambers and Jurafsky, 2008). In particular, Chambers and Jurafsky (2008) made a first attempt to learn scripts from test inducing event

[*] This work has been done when the first author worked at SUTD.

[1] The term "temporal order" is used throughout this work to indicate the narrative order in texts, following Chambers and Jurafsky (2008). Strictly speaking, the event order we extract is the narrative order.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 57–67
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

chains by grouping events based on their narrative coherence, calculated based on Pairwise Mutual Information (PMI). Jans et al. (2012) showed that the method can be improved by calculating event relations using skip bi-gram probabilities, which explicitly model the temporal order of pairs event. Jans et al. (2012)'s model is adopted by a line of subsequent methods on inducing event chains from text (Orr et al., 2014; Pichotta and Mooney, 2014; Rudinger et al., 2015).

While the above methods are statistical, neural network models have recently been used for event sequence modeling. Granroth-Wilding and Clark (2016) used a Siamese Network instead of PMI to calculate the coherence between two events. Rudinger et al. (2015) extended the idea of Jans et al. (2012) by using a log-bilinear neural language model (Mnih and Hinton, 2007) to calculate event probabilities. By learning embeddings for reducing sparsity, the above models give much better results compared to the models of Chambers and Jurafsky (2008) and Jans et al. (2012). Similar in spirit, Modi (2016) predicted the probability of an event belonging to a certain event chain by modeling known events in the chain as a bag of vectors, showing that it outperforms discrete statistical methods. These neural methods are consistent with the earlier statistical models in leveraging event-pair relations.

Pichotta and Mooney (2016) experimented with LSTM for script learning, using an existing sequence of events to predict the probability of a next event, which outperformed strong discrete baselines. One advantage of LSTMs is that they can encode unbounded time sequences without losing long-term historical information. LSTMs capture significantly more order information compared to the methods of Granroth-Wilding and Clark (2016), Rudinger et al. (2015), and Modi (2016), which model the temporal order of only pairs of events. On the other hand, a strong-order LSTM model can also suffer the disadvantage of over-fitting, given the flexible order of event chains in a script, as demonstrated by the cases of Figure 1. In this aspect, event-pair models are more adaptive for flexible orders. However, no direct comparisons have been reported between LSTM and various existing neural network methods that model event-pairs.

We make such comparisons using the same benchmark, finding that the method of Pichotta and Mooney (2016) does not necessarily outperform event-pair models, such as Granroth-Wilding and Clark (2016). LSTM temporal ordering and event pair modeling have their respective strength. To leverage the advantages of both methods, we propose to integrate chain temporal order information into event relation measuring. In particular, we calculate event pair relations by representing events in a chain using LSTM hidden states, which encode temporal information. The LSTM over-fitting issue is mitigated by using the temporal-order in a chain as a *feature* for event pair modeling, rather than the direct model *output*. In addition, observing that the importance of existing events can vary for inferring a subsequent event, we use a dynamic memory network model to automatically induce event weights for each event for inferring the next event. In contrast, previous methods give equal weights to existing events (Chambers and Jurafsky, 2008; Modi, 2016; Granroth-Wilding and Clark, 2016).

Results on a multi-choice narrative cloze benchmark show that our model significantly outperforms both Granroth-Wilding and Clark (2016) and Pichotta and Mooney (2016), improving the state-of-the-art accuracy from 49.57% to 55.12%. Our contributions can be summarized as follows:

- We make a systematic comparison of LSTM and pair-based event sequence learning methods using the same benchmarks.
- We propose a novel dynamic memory network model, which combines the advantages of both LSTM temporal order learning and traditional event pair coherence learning.
- We obtain the best results in the standard multi-choice narrative cloze test.

Our code is released at `https://github.com/wangzq870305/event_chain`.

2 Related Work

Scripts have been a traditional subject in AI research (Schank et al., 1977), where event sequences are manually encoded in knowledge bases, and used for end tasks such as inference. They are also connected with research in linguistics and psychology, and sometimes referred to as *frames* (Minsky, 1975; Fillmore, 1982) and *schemata* (Rumelhart, 1975). The same concept is also studied as *templates* in information extraction (Sundheim, 1991). Chambers and Jurafsky (2008) pioneered the recent line of work on script induction (Jans et al., 2012; Pichotta and Mooney, 2016; Granroth-Wilding and Clark, 2016), where the focus is on modeling narrative event chains, a crucial subtask for script modeling from raw text. Below we summarize such investigations.

With respect to **event representation**, Chambers and Jurafsky (2008) casted narrative events as triples of the form $\langle event, dependency \rangle$, where the event is typically represented by a verb and the dependency represents typed dependency relations between the event and a protagonist, such as "subject" and "object". Chambers and Jurafsky (2008) organized narrative chains around a central actor, or protagonist, mining events that share a common protagonist from texts by using a syntactic parser and a coreference resolver. Balasubramanian et al. (2013) observed that the protagonist representation of event chains can suffer from weaknesses such as lack of coherence, and proposed to represent events as $\langle arg_1, relation, arg_2 \rangle$, where arg_1 and arg_2 represent the subject and object, respectively. Such representation is inspired by open information extraction (Mausam et al., 2012), and offers richer features for event pair modeling. Pichotta and Mooney (2014) adpoted a similar idea, using $v(e_s, e_o, e_p)$ to represent an event, where v is a verb lemma, e_s is the subject, e_o is the object, and e_p is an entity with prepositional relation to v. Their representation is used by subsequent work such as Modi (2016) and Granroth-Wilding and Clark (2016). We follow Pichotta and Mooney (2016) in our event representation form.

With respect to **modeling**, existing methods can be classified into two main categories, namely *weak-order* models, which calculate relations between pairs of events, and *strong-order* models, which consider the temporal order of events in a full sequence. Event-pair models have so far been the dominant method in the literature. Earlier work used discrete event representations and estimated event relations by statistical counting. As mentioned earlier, Chambers and Jurafsky (2008) used PMI to calculate event relations, and Jans et al. (2012) used skip bigram probabilites to the same end, which is order-sensitive. Most subsequent methods followed Jans et al. (2012) in using skip n-grams (Pichotta and Mooney, 2014; Rudinger et al., 2015).

Events being multi-argument structures, counting-based methods can suffer from sparsity issues. Recent work employed embeddings to address this disadvantage. Rudinger et al. (2015) learned event embeddings as a by-product of training a log-bilinear language model for events; Granroth-Wilding and Clark (2016) leveraged the skip-gram model of Mikolov et al. (2013) for training the embeddings of event and arguments by ordering them into a pseudo sentence. Modi (2016) utilized *word embeddings* of verbs and arguments directly, using a hidden layer to automatically consolidate word embedding into a single structured *event embeddings*. We follow Modi (2016) and use a hidden layer to learn event argument compositions given word embeddings, training the composition function as a part of the event chain learning process.

Mitigating the sparsity issue of event representations, neural methods can capture temporal orders between events beyond skip n-grams. Our model integrates the advantages of strong-order learning and event-pair learning by using LSTM hidden states as feature representation of existing events in the calculation of event pair relationships. In addition, we use a memory network model to weigh existing events, which gives better results compared to the equal weighting method of existing models.

With respect to **evaluation**, Chambers and Jurafsky (2008) proposed the *Narrative Cloze Test*, which asks for a missing event in a given event chain with a gap. The task has been adopted by various subsequent work for comparing results with Chambers and Jurafsky (2008) (Jans et al., 2012; Pichotta and Mooney, 2014; Rudinger et al., 2015). One issue of the narrative cloze test is that there can sometimes be multiple plausible answers, but only one gold-standard answer, which can make it overly expensive to manually evaluate system outputs. To address this issue, Modi (2016) proposed the *Adversarial Narrative Cloze* (ANC) task, which is to discriminate between pairs of real and corrupted event chains. Granroth-Wilding and Clark (2016) proposed the *Multi-Choice Narrative Cloze* (MCNC) task, which is to choose the most likely next event from a set of candidates given a chain of events. We choose MCNC for comparing different models.

Other related work includes learning temporal relations of events (Modi and Titov, 2014; Uz-

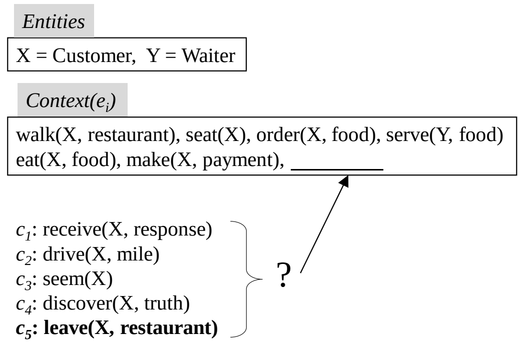


Figure 2: Multiple choice narrative cloze. The gold subsequent event is marked in bold.

Zaman et al., 2013; Abend et al., 2015), evaluated using different metrics. There has also been work using graph models to induce frames, which emphasize more on learning event structures and less on temporal orders (Chambers, 2013; Cheung et al., 2013). The above methods focus on one of the two subtasks we consider here. Frermann et al. (2014) used a Bayesian model to jointly cluster web collections of explicit event sequence and learn input event-pair temporal orders. However, their work is under a different input setting (Regneri et al., 2010), not learning event chains from texts. Mostafazadeh et al. (2016) proposed the story close task (SCT), which is to predict the ending given a unfinished story. Our narrative chain prediction task can be regarded as a sub task in the story close task, which can contribute as a major approach. On the other hand, information beyond event chains can be useful for the story close task.

3 Problem Definition

As shown in Figure 2, given a chain of narrative events e_1, e_2, ..., e_{n-1}, our work is to predict the likelihood of a next event candidate e_n. Formally, an event e is a structure $v(a_0, a_1, a_2)$, where v is a verb describing the event, a_0 and a_1 are its subject and direct object, respectively, and a_2 is a prepositional object. For example, given the sentence "John brought Marry to the restaurant", an event $bring\{John, Marry, to\ the\ restaurant\}$ can be extracted.

We follow the standard script induction setting (Chambers and Jurafsky, 2008; Granroth-Wilding and Clark, 2016), extracting events from a text corpus using a syntactic parser and a named entity resolver. A neural network is used to model chains of extracted events for script learning. In particular, we model the probability of a subsequent event given a chain of events. For evaluation, we solve the multi-choice narrative cloze task: given a chain of events and a set of candidate next events, the most likely candidate is chosen as the output.

4 Model

The overall structure of our model is shown in Figure 3, which has three main components. First, given an event $v(a_0, a_1, a_2)$, a representation layer is used to compose the embeddings of v, a_0, a_1, and a_2 into a single event vector e. Second, a LSTM is used to map a sequence of existing events e_1, e_2, ..., e_{n-1} into a sequence of hidden vectors h_1, h_2, ..., h_{n-1}, which encode the temporal order. Given a next event candidate e_c, the recurrent network takes one further step from h_{n-1} to derive its hidden vector h_c, which encodes e_c. Third, h_c is paired with h_1, h_2, ..., h_{n-1} individually, and passed to a dynamic memory network to learn the relatedness score s. s is used to denote the connectedness between the candidate subsequent event and the context event chain.

4.1 Event Representation

We learn vector representations of standard events by composing pre-trained word embeddings of its verb and arguments. The skipgram model (Mikolov et al., 2013) is used to train word vectors. For arguments that consist of more than one word, we use the averaged word for the representation. OOV words are represented simply using zero vectors. For events with less than 3 arguments, such as "John fell", where v = fall, a_0 = John, a_1 = NULL, and a_2 = NULL, the NULL arguments are represented using all-zero vectors.

Denoting the embeddings of v, a_0, a_1, and a_2 as $e(v)$, $e(a_0)$, $e(a_1)$, and $e(a_2)$, respectively, the embedding of e is calculated using a $\tanh$ composition layer

$$\begin{aligned} e(e) = &\tanh(W_e^v \cdot e(v) + W_e^0 \cdot e(a_0) + \\ &W_e^1 \cdot e(a_1) + W_e^2 \cdot e(a_2) + b_e) \end{aligned} \quad (1)$$

Here W_e^v, W_e^0, W_e^1, W_e^2, and b are model parameters, which are randomly initialized and tuned during the training of the main network.

4.2 Modeling Temporal Orders

Given the embeddings of the existing chain of events e_1, e_2, ..., e_{n-1}, we use a standard LSTM (Hochreiter and Schmidhuber, 1997) without

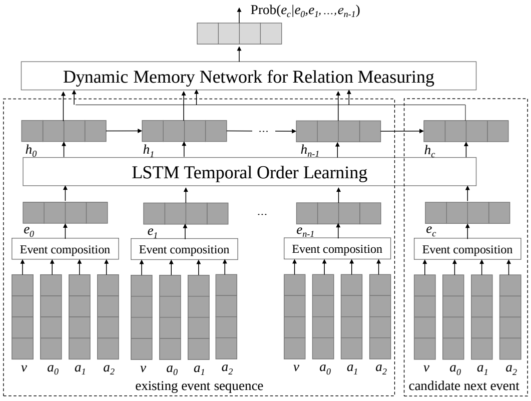


Figure 3: Overview of proposed model.

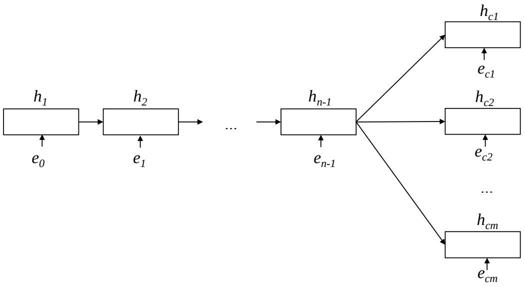


Figure 4: Temporal order modeling.

coupled input and forget gates or peephole connections to model the temporal order. We obtain a sequence of hidden state vectors h_1, h_2, ..., h_{n-1} by recurrently feeding $e(e_1)$, $e(e_2)$, ..., $e(e_{n-1})$ as inputs to the LSTM, where $h_i = \mathrm{LSTM}(e(e_i), h_{i-1})$. The initial state h_s and all stand LSTM parameters are randomly initialized and tuned during training.

Now for each candidate next event e_c, we obtain its vector representation $e(e_c)$ in the same way as for e_1 to e_{n-1}. $e(e_c)$ is then appended to the existing event chain to obtain a temporal-order-sensitive feature vector h_c, by advancing the recurrent encoding process for one step from h_{n-1}: $h_c = \mathrm{LSTM}(e(e_c), h_{n-1})$. With multiple next event candidates e_c^1, e_c^2, ..., e_c^m ($m \in [1, \infty]$), m feature vectors are obtained, as shown in Figure 4, each being used as a basis for estimating the probability of the corresponding event candidate.

4.3 Modeling Pairwise Event Relations

After obtaining the hidden states for events, we model event pair relations using these hidden state vectors. A straightforward approach to model the relation between two events is using a Siamese network (Granroth-Wilding and Clark, 2016). The order-sensitive LSTM features for existing events h_1, h_2, ..., h_{n-1} and the candidate event h_c are used as event representations. Given a pair of events h_i ($i \in [1..n-1]$) and h_c, the relatedness score is calculated by

$$s_i = \mathrm{sigmoid}(W_{si}h_i + W_{sc}h_c + b_s), \qquad (2)$$

where W_{si}, W_{sc} and b_s are model parameters.

Given the relation score s_i between h_c and each existing event h_i, the likelihood of e_c given e_1, e_2, ..., e_{n-1} can be calculated as the average of s_i:

$$s = \frac{\sum_{i=1}^{n-1} s_i}{n-1} \qquad (3)$$

Weighting existing events. The drawback of above approach is that it considers the contribution of each event on the chain is same. However, given a chain of existing events, some are more informative for inferring a subsequent event than others. For example, given the events "wait in queue", "getting seated" and "order food", "order food" is more relevant for inferring "eat food" compared with the other two given events. Given information over the full event chain, this link can be more evident since the scenario is likely restaurant visiting.

We use an attentional neural network to calculate the relative importance of each existing event according to the subsequent event candidate, using h_i ($i \in [1..n-1]$) and h_c for event representations:

$$u_i = \tanh(W_{ei}h_i + W_c h_c + b_u) \qquad (4)$$

$$\alpha_i = \frac{\exp(u_i)}{\sum_j \exp(u_j)} \qquad (5)$$

where $\alpha_i \in [0, 1]$ is the weight of h_i, and $\sum_i \alpha_i^t = 1$. W_{ei}, W_c, and b_u are model parameters.

After obtaining the weight α_i of each existing event h_i, the relatedness of e_c with the existing events can be calculated as:

$$s = \sum_{i=1}^{n-1} \alpha_i \cdot s_i \qquad (6)$$

Multi-layer attention using Deep memory network. Memory network (Weston et al., 2014; Mikolov et al., 2014) has been used for exploring deep semantic information for semantic tasks. Such as question answering (Sukhbaatar et al., 2015; Kumar et al., 2016) and reading comprehension (Hermann et al., 2015; Weston et al., 2015). Our task is analogous to such semantic tasks in

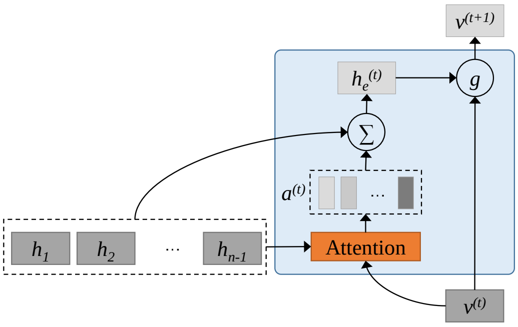


Figure 5: Memory network at hop t. h_i is the hidden variable of the existing event chain, v^t is the semantic representation between context events and candidate event. a^t is the weight of context events, and g is the gated recurrent network on Eq.10.

the sense that deep semantic information can be necessary for making the most rational inference. Hence, we are motivated to use a deep memory network model to refine event weight and event relation calculation by recurrently modeling more abstract representations of the scenario. Different from the previous researches, we use the memory network to model the event chain, refining the attention mechanism used to explore the pair-wise relation between events.

The memory model consists of multiple dynamic computational layers (hops). For the first layer (hop 1), the weights α for existing events e_1, e_2, ..., e_{n-1} can be calculated using the same attention mechanism as Eq.4 and Eq.5. Given the weights α, we build a consolidated representation of context event chain $e_1, e_2, ..., e_{n-1}$ as a weighted sum of $h_1, h_2, ..., h_{n-1}$:

$$h_e = \sum_{i-1}^{n-1} \alpha_i \cdot h_i \tag{7}$$

The event candidate h_c and the new representation of the existing chain h_e can be further integrated to deduce a deeper representation of the full event chain hypothesis to the next layer (hop 2), denoted as v. v contains deeper semantic information compared with h_c, which encode the temporal order of the event chain $[h_1, h_2, ..., h_{n-1}, h_c]$ without differentiating the weights of each event. As a result, in the next hop, better event weights can potentially be deduced by using v instead of h_c in the calculation of attention:

$$u_i^t = \tanh(W_{ei}h_i + W_v v^t + b_u) \tag{8}$$

$$\alpha_i^t = \frac{\exp(u_i^t)}{\sum_j \exp(u_j^t)} \tag{9}$$

In the same way, we stack multiple hops and repeat the steps multiple times, so that more abstract evidences can be extracted according to the chain of existing events. The above process can be performed recurrently, by taking h_c as an initial scenario representation v_0, and then repeatedly calculating h_e^t given h_1, h_2, ..., h_{n-1} and v^t, and using h_e^t and v_t to find a deeper scenario representation v^{t+1}. Following Chung et al. (2014) and Tran et al. (2016), a gated recurrent network is used to this end:

$$\begin{aligned} z &= \sigma(W_z h_e^t + U_z v^t) \\ r &= \sigma(W_r h_e^t + U_r v^t) \\ \hat{h} &= \tanh(W h_e^t + U(r \odot v^t)) \\ v^{t+1} &= (1-z) \odot v^t + z \odot \hat{h} \end{aligned} \tag{10}$$

At any step, if the value of $|v^{t+1} - v^t|$ is less than the threshold μ, we consider that the progress has reached convergence. Figure 5 shows an overview of the memory network at hop t.

4.4 Training

Given a set of event chains, each with a gold-standard subsequent event and a number of non-subsequent events, our training objective is to minimize the cross-entropy loss between the gold subsequent event and the set of non-subsequent events. The loss function of event chain prediction is that:

$$L(\Theta) = \sum_{i=1}^{N} (s_i - y_i)^2 + \frac{\lambda}{2}||\Theta||^2 \tag{11}$$

where s_i is the relation score, y_i is the label of the candidate ($y_i = 1$ for positive sample, and $y_i = 0$ for negative sample), Θ is the set of model parameters and λ is a parameter for L2 regularization. We apply online training, where model parameters are optimized by using AdaGrad (Duchi et al., 2011). We train word embedding using the *Skip-gram* algorithm (Mikolov et al., 2013)[2].

5 Experiments

5.1 Datasets

Following Granroth-Wilding and Clark (2016), we extract events from the NYT portion of the Gigaword corpus (Graff et al., 2003). The C&C

[2] https://code.google.com/p/word2vec/

tools (Curran et al., 2007) are used for POS tagging and dependency parsing, and OpenNLP[3] for phrase structure parsing and coreference resolution. The training set consists of 1,500,000 event chains. We follow Granroth-Wilding and Clark (2016) and use 10,000 event chains as the test set, and 1,000 event chains for development. There are 5 choices of output event for event input chain, which are given by Granroth-Wilding and Clark (2016). This dataset is referred to as **G&C16**.

We also adapt the Chambers and Jurafsky (2008)'s dataset to the multiple choice setting, and use this dataset as the second benchmark. The dataset contains 69 documents, with 346 multiple choice event chain samples. We randomly sample 4 negative subsequent events for each event chain to make multiple-choice candidates. This dataset is referred to as **C&J08**. For both datasets, accuracy (Acc.) of the chosen subsequent event is used to measure the performance of our model.

5.2 Hyper-parameters

There are several important hyper-parameters in our models, and we tune their values using the development dataset. We set the regularization weight $\lambda = 10^{-8}$ and the initial learning rate to 0.01. The size of word vectors is set to 300, and the size of hidden vectors in LSTM to 128. In order to avoid over-fitting, dropout (Hinton et al., 2012) is used for word embedding with a ratio of 0.2. The neighbor similarity threshold η is set to 0.25. The threshold μ of the memory network sets to 0.1.

5.3 Development Experiments

We conduct a set of development experiments on the G&C16 development set to study the influence of event argument representations and network configurations of the proposed *MemNet* model.

5.3.1 Influence of Event Structure

Existing literature discussed various structures to denote events, such as $v(a_0, a_1)$ and $v(a_0, a_1, a_2)$. We investigate the influence of integrating argument values of the subject a_0, object a_1 and preposition a_2, by doing ablation experiments on the development data. The results are shown in Table 1, where the system using all arguments gives a 54.36% accuracy. By removing a_2, which exists in 17.6% of the events in our developmental data, the accuracy drops to 54.02%. In contrast, by removing a_0 and a_1, which exist in 87.6% and 64.6% of the events in the development data, respectively, the accuracies drop to 53.43% and 53.57%, respectively, which demonstrates the relative importance of a_0 (i.e., the subject) and a_1 (i.e., the object) for event modelling. While most previous work (Chambers and Jurafsky, 2008; Balasubramanian et al., 2013; Pichotta and Mooney, 2014) modelled only a_0 and a_1, recent work (Pichotta and Mooney, 2016; Granroth-Wilding and Clark, 2016) modelled a_2 also.

Method	**Acc. (%)**
MemNet	**54.36**
-$verb$	42.63
-(a_0, a_1)	52.32
-(a_0)	53.43
-(a_1)	53.57
-(a_2)	54.02

Table 1: Influence of event arguments.

By removing both a_1 and a_2, the accuracy drops further to 53.32%. Interestingly, by removing the verb while keeping only the arguments, the accuracy drops to 42.63%. While this demonstrates the central value of the verb in denoting a event, it also suggests that the arguments themselves play a useful role in inferring the stereotypical scenario.

5.3.2 Influence of Network Configurations

We study the influence of various network configurations by performing ablation experiments, as shown in Table 2. *MemNet* is the full model of this paper; *-LSTM* denotes ablation of the LSTM layer, using $e(e_1)$, $e(e_2)$, ..., $e(e_{n-1})$ instead of h_1, h_2, ..., h_{n-1} to represent events; *-Hop* denotes ablation of the dynamic network model, using only attention mechanism to calculate the weights of each existing event; *-Attention* denotes ablation of the attention mechanism, using the same weight on each existing event when inferring e_c. The model "*-Attention, -LSTM*" is hence similar to the method of Granroth-Wilding and Clark (2016), although we used a different way of deriving event embeddings. The model "*LSTM-only*" shows a based by using LSTM hidden vector h_{n-1} to directly predict the next event, which is similar to the method of Pichotta and Mooney (2016).

Influence of Temporal Order. By comparing "MemNet" and "-LSTM", and comparing "-

[3] https://opennlp.apache.org/

Method	Acc. (%)
MemNet	**54.36**
-Hop	52.03
-Attention	50.76
-LSTM	51.72
-Hop,-LSTM	50.65
-Attention,-LSTM	48.26
LSTM-Only	46.72

Table 2: Analysis of network structure.

Attention" with "-Attention, -LSTM", one can find that temporal order information over the whole event chain does have significant influence on the results ($p - value < 0.01$ using t-test). On the other hand, using LSTM to directly predict the subsequent event ("LSTM-only") does not give better accuracies compared to model event pairs ("-Attention, -LSTM"). This confirms our intuition that strong-oder modelling and event-pair modelling each have their own strength.

Influence of Attention. Comparison between "-Attention" and "-Hop", and between "-Attention, -LSTM" and "-Hop, -LSTM" shows that giving different weights to different events does lead to improving results. Our analysis in Section 4.3 gives more intuitions to this observation. Finally, comparison between "-Hop" and "MemNet" and between "-Hop, -LSTM" and "-LSTM" shows that a multi-hop deep memory network can indeed enhance the model with single level attention by offering more effective semantic representation of the scenarios.

5.4 Final Results

Table 3 shows the final results on the C&C 16 and C&J08 datasets, respectively. We compare the results of our final model with the following baselines:

- **PMI** is the co-occurrence based model of Chambers and Jurafsky (2008), who calculate event pair relations based on Pointwise Mutual Information (PMI), scoring each candidate event e_c by the sum of PMI scores between the given events e_0, e_1, ..., e_{n-1} and the candidate.

- **Bigram** is the counting based model of Jans et al. (2012), calculating event pair relations based on skip bigram probabilities, trained using maximum likelihood estimation.

Method	G&C16	C&J08
PMI	30.52	30.92
Bigram	29.67	25.43
Event-Comp	49.57	43.28
RNN	45.74	43.17
MemNet	**55.12**	**46.67**

Table 3: Final results.

- **Event-Comp** is the neural event relation model proposed by Granroth-Wilding and Clark (2016). They learn event representations by calculating pair-wise event scores using a Siamese network.

- **RNN** is the method of Pichotta and Mooney (2016), who model event chains by directly using h_c in Section 4.2 to predict the output, rather than taking them as features for event pair relation modeling.

- **MemNet** is the proposed deep memory network model.

Our reimplementation of PMI and Bigrams follows (Granroth-Wilding and Clark, 2016). It can be seen from the table that the statistical counting-based models PMI and Bigram significantly underperform the neural network models Event-Comp, RNN and MemNet, which is largely due to their sparsity and lack of semantic representation power. Under our event representation, Bigram does not outperform PMI significantly either, although considering the order of event pairs. This is likely due to sparsity of events when all arguments are considered.

Direct comparison between Event-Comp and RNN shows that the event-pair model gives comparable results to the strong-order LSTM model. Although Granroth-Wilding and Clark (2016) and Pichotta and Mooney (2016) both compared with statistical baselines, they did not make direct comparisons between their methods, which represent two different approaches to the task. Our results show that they each have their unique advantages, which confirm our intuition in the introduction. By considering both pairwise relations and chain temporal orders, our method significantly outperform both Event-Comp and RNN ($p - value < 0.01$ using t-test), giving the best reported results on both datasets.

6 Conclusion

We proposed a dynamic memory network to integrate chain order information into event relation measuring, calculating event pair relations by representing events in a chain using LSTM hidden states, which encode temporal orders, and using a dynamic memory model to automatically induce event weights for each event. Standard evaluation showed that our method significantly outperforms state-of-the-art event pair models and event chain models, giving the best results reported so far.

Acknowledgments

The corresponding author is Yue Zhang. We are grateful for the help of Fei Dong for his initial discussion. We thank our anonymous reviewers for their constructive comments, which helped to improve the paper. This work is supported by the Temasek Lab grant IGDST1403012 at Singapore University of Technology and Design.

References

Omri Abend, Shay B. Cohen, and Mark Steedman. 2015. Lexical event ordering with an edge-factored model. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1161–1171.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1721–1731.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1797–1807.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 789–797.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 837–846.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

James R. Curran, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with c&c and boxer. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Charles Fillmore. 1982. Frame semantics. *Linguistics in the morning calm*, pages 111–137.

Lea Frermann, Ivan Titov, and Manfred Pinkal. 2014. A hierarchical bayesian model for unsupervised induction of script knowledge. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 49–57.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2727–2733.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

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

Bram Jans, Steven Bethard, Ivan Vulic, and Marie-Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *EACL 2012, 13th Conference of the European Chapter of* *the Association for Computational Linguistics, Avignon, France, April 23-27, 2012*, pages 336–344.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1378–1387.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 523–534.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michaël Mathieu, and Marc'Aurelio Ranzato. 2014. Learning longer memory in recurrent neural networks. *CoRR*, abs/1412.7753.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Marvin Minsky. 1975. A framework for representing knowledge.

Andriy Mnih and Geoffrey E. Hinton. 2007. Three new graphical models for statistical language modelling. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 641–648.

Ashutosh Modi. 2016. Event embeddings for semantic script modeling. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 75–83.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, CoNLL 2014, Baltimore, Maryland, USA, June 26-27, 2014*, pages 49–57.

Raymond Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. *Urbana*, 51:61801.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 839–849.

John Walker Orr, Prasad Tadepalli, Janardhan Rao Doppa, Xiaoli Fern, and Thomas G. Dietterich. 2014. Learning scripts as hidden markov models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1565–1571.

Karl Pichotta and Raymond J. Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 220–229.

Karl Pichotta and Raymond J. Mooney. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2800–2806.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 979–988.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1681–1686.

David E Rumelhart. 1975. Notes on a schema for stories. *Representation and understanding: Studies in cognitive science*, 211(236):45.

Roger Schank, Roger Schank, and Robert P Abelson. 1977. Scripts, plans, goals and understanding; an inquiry into human knowledge structures. Technical report.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448.

Beth Sundheim. 1991. Third message understanding evaluation and conference (muc-3): Phase 1 status report. In *HLT*.

Ke M. Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 321–331.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James F. Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Proceedings of the 7th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2013, Atlanta, Georgia, USA, June 14-15, 2013*, pages 1–9.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR*, abs/1410.3916.

Entity Linking for Queries by Searching Wikipedia Sentences

Chuanqi Tan[†] [*] **Furu Wei**[‡] **Pengjie Ren**[+] **Weifeng Lv**[†] **Ming Zhou**[‡]
[†]State Key Laboratory of Software Development Environment, Beihang University, China
[‡]Microsoft Research [+]Shandong University
[†]tanchuanqi@nlsde.buaa.edu.cn [+]jay.ren@outlook.com
[‡]{fuwei, mingzhou}@microsoft.com [†]lwf@buaa.edu.cn

Abstract

We present a simple yet effective approach for linking entities in queries. The key idea is to search sentences similar to a query from Wikipedia articles and directly use the human-annotated entities in the similar sentences as candidate entities for the query. Then, we employ a rich set of features, such as link-probability, context-matching, word embeddings, and relatedness among candidate entities as well as their related entities, to rank the candidates under a regression based framework. The advantages of our approach lie in two aspects, which contribute to the ranking process and final linking result. First, it can greatly reduce the number of candidate entities by filtering out irrelevant entities with the words in the query. Second, we can obtain the query sensitive prior probability in addition to the static link-probability derived from all Wikipedia articles. We conduct experiments on two benchmark datasets on entity linking for queries, namely the ERD14 dataset and the GERDAQ dataset. Experimental results show that our method outperforms state-of-the-art systems and yields 75.0% in F1 on the ERD14 dataset and 56.9% on the GERDAQ dataset.

1 Introduction

Query understanding has been an important research area in information retrieval and natural language processing (Croft et al., 2010). A key part of this problem is entity linking, which aims to annotate the entities in the query and link them to a knowledge base such as Freebase and Wikipedia. This problem has been extensively studied over the recent years (Ling et al., 2015; Usbeck et al., 2015; Cornolti et al., 2016).

The mainstream methods of entity linking for queries can be summed up in three steps: mention detection, candidate generation, and entity disambiguation. The first step is to recognize candidate mentions in the query. The most common method to detect mentions is to search a dictionary collected by the entity alias in a knowledge base and the human-maintained information in Wikipedia (such as anchors, titles and redirects) (Laclavik et al., 2014). The second step is to generate candidates by mapping mentions to entities. It usually uses all possible senses of detected mentions as candidates. Hereafter, we refer to these two steps of generating candidate entities as entity search. Finally, they disambiguate and prune candidate entities, which is usually implemented with a ranking framework.

There are two main issues in entity search. First, a mention may be linked to many entities. The methods using entity search usually leverage little context information in the query. Therefore it may generate many completely irrelevant entities for the query, which brings challenges to the ranking phase. For example, the mention “Austin” usually represents the capital of Texas in the United States. However, it can also be linked to “Austin, Western Australia”, “Austin, Quebec”, “Austin (name)”, “Austin College”, “Austin (song)” and 31 other entities in the Wikipedia page of “Austin (disambiguation)”. For the query “blake shelton austin lyrics”, Blake Shelton is a singer and made his debut with the song “Austin”. The entity search method detects the mention “austin” using the dictionary. However, while “Austin (song)” is most related to the context “blake shelton” and “lyrics”, the mention “austin” may be linked to all the above entities as candidates. Therefore candidate gener-

[*] Contribution during internship at Microsoft Research.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 68–77
Copenhagen, Denmark, September 7–11, 2017.

ation with entity search generates too many candidates especially for a common anchor text with a large number of corresponding entities. Second, it is hard to recognize entities with common surface names. The common methods usually define a feature called "link-probability" as the probability that a mention is annotated in all documents. There is an issue with this probability being static whatever the query is. We show an example with the query "her film". "Her (film)" is a film while its surface name is usually used as a possessive pronoun. Since the static link-probability of "her" from all Wikipedia articles is very low, "her" is usually not treated as a mention linked to the entity "Her (film)".

In this paper, we propose a novel approach to generating candidates by searching sentences from Wikipedia articles and directly using the human-annotated entities as the candidates. Our approach can greatly reduce the number of candidate entities and obtain the query sensitive prior probability. We take the query "blake shelton austin lyrics" as an example. Below we show a sentence in the Wikipedia page of "Austin (song)".

[[Austin (song)|Austin]] is the title of a debut song written by David Kent and Kirsti Manna, and performed by American country music artist [[Blake Shelton]].

Table 1: A sentence in the page "Austin (song)".

In the above sentence, the mentions "Austin" and "Blake Shelton" in square brackets are annotated to the entity "Austin (song)" and "Blake Shelton", respectively. We generate candidates by searching sentences and thus obtain "Blake Shelton" as well as "Austin (song)" from this example. We reduce the number of candidates because many irrelevant entities linked by "austin" do not occur in returned sentences. In addition, as previous methods generate candidates by searching entities without the query information, "austin" can be linked to "Austin, Texas" with much higher static link-probability than all other senses of "austin". However, the number of returned sentences that contain "Austin, Texas" is close to the number of sentences that contain "Austin (song)" in our system. We show another example with the query "her film" in Table 2. In this sentence, "Her", "romantic", "science fiction", "comedy-drama" and "Spike Jonze" are annotated to corresponding entities. As "Her" is annotated to "Her (film)" by humans in this example, we have strong evidence to annotate it even if it is usually used as a possessive pronoun with very low static link-probability.

[[Her (film)|Her]] is a 2013 American [[romantic]] [[science fiction]] [[comedy-drama]] film written, directed, and produced by [[Spike Jonze]].

Table 2: A sentence in the page "Her (film)".

We obtain the anchors as well as corresponding entities and map them to the query after searching similar sentences. Then we build a regression based framework to rank the candidates. We use a rich set of features, such as link-probability, context-matching, word embeddings, and relatedness among candidate entities as well as their related entities. We evaluate our method on the ERD14 and GERDAQ datasets. Experimental results show that our method outperforms state-of-the-art systems and yields 75.0% and 56.9% in terms of F1 metric on the ERD14 dataset and the GERDAQ dataset respectively.

2 Related Work

Recognizing entity mentions in text and linking them to the corresponding entries helps to understand documents and queries. Most work uses the knowledge base including Freebase (Chiu et al., 2014), YAGO (Yosef et al., 2011) and Dbpedia (Olieman et al., 2014). Wikify (Mihalcea and Csomai, 2007) is the very early work on linking anchor texts to Wikipedia pages. It extracts all n-grams that match Wikipedia concepts such as anchors and titles as candidates. They implement a voting scheme based on the knowledge-based and data-driven method to disambiguate candidates. Cucerzan (2007) uses four recourses to generate candidates, namely entity pages, redirecting pages, disambiguation pages, and list pages. Then they disambiguate candidates by calculating the similarity between the contextual information and the document as well as category tags on Wikipedia pages. Milne and Witten (2008) generate candidates by gathering all n-grams in the document, and retaining those whose probability exceeds a low threshold. Then they define commonness and relatedness on the hyper-link structure of Wikipedia to disambiguate candidates.

The work on linking entities in queries has been extensively studied in recent years. TagME (Ferragina and Scaiella, 2010) is a very early work on entity linking in queries. It generates candidates by searching Wikipedia page titles, anchors and redirects. Then disambiguation exploits the structure of the Wikipedia graph, according to a voting scheme based on a relatedness measure inspired by Milne and Witten (2008). The improved version of TagME, named WAT (Piccinno and Ferragina, 2014), uses Jaccard-similarity between two pages' in-links as a measure of relatedness and uses PageRank to rank the candidate entities. Moreover, Meij (2012) proposes a two step approach for linking tweets to Wikipedia articles. They first extract candidate concepts for each n-gram, and then use a supervised learning algorithm to classify relevant concepts.

Unlike the work which revolves around ranking entities for query spans, the Entity Recognition and Disambiguation (ERD) Challenge (Carmel et al., 2014) views entity linking in queries as the problem of finding multiple query interpretations. The SMAPH system (Cornolti et al., 2014) which wins the short-text track works in three phases: fetching, candidate-entity generation and pruning. First, they fetch the snippets returned by a commercial search engine. Next, snippets are parsed to identify candidate entities by looking at the boldfaced parts of the search snippets. Finally, they implement a binary classifier using a set of features such as the coherence and robustness of the annotation process and the ranking as well as composition of snippets. They further extend SMAPH-1 to SMAPH-2 (Cornolti et al., 2016). They use the annotator WAT to annotate the snippets of search results to generate candidates and joint the additionally link-back step as well as the pruning step in the ranking phase, which gets the state-of-the-art results on the ERD14 dataset and their released dataset GERDAQ. There is another work closed to SMAPH that uses information of query logs and anchor texts (Blanco et al., 2015), which gives a ranked list of entities and is evaluated by means of typical ranking metrics.

Our work is different from using search engines to generate candidates. We firstly propose to search Wikipedia sentences and take advantage of human annotations to generate candidates. The previous work, such as SMAPH, employs search engine for candidate generation, which puts queries in a larger context in which it is easier to make sense of them. However, it uses WAT, an entity search based tool, to pre-annotate the snippets for candidate generation, which falls back the issues of entity search.

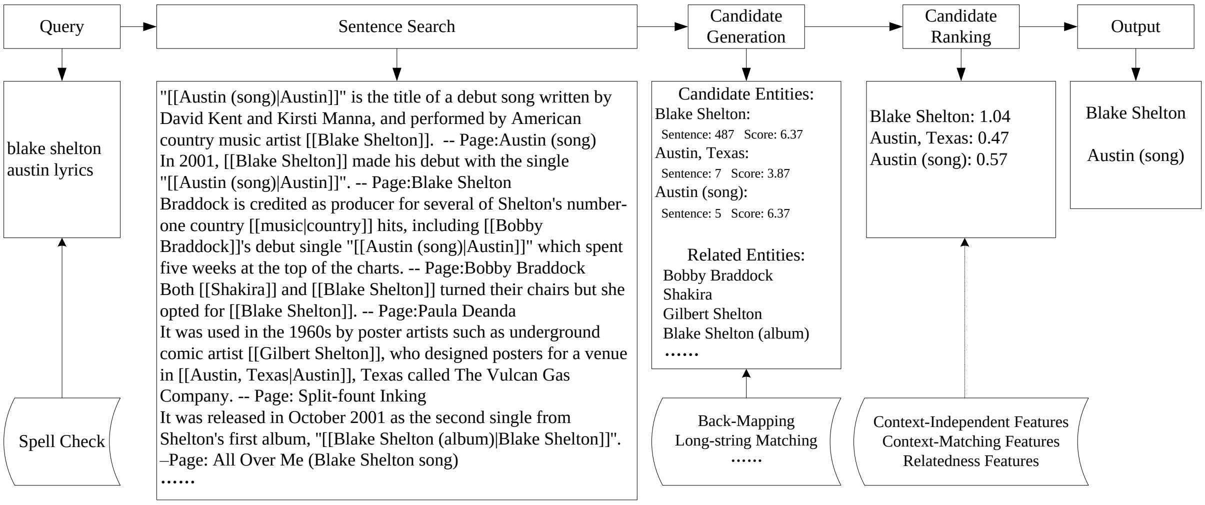
3 Our Approach

As shown in Figure 1, we introduce our approach with the query "blake shelton austin lyrics". Our approach consists of three main phases: sentence search, candidate generation, and candidate ranking. First, we search the query in all Wikipedia articles to obtain the similar sentences. Second, we extract human-annotated entities from these sentences. We keep the entities whose corresponding anchor texts occur in the query as candidates, and treat others as related entities. Specifically, we obtain three candidates in this example, namely "Blake Shelton", "Austin, Texas", and "Austin (song)". Finally, we use a regression based model to rank the candidate entities. We get the final annotations of "Blake Shelton" and "Austin (song)" whose scores are higher than the threshold selected on the development set. In the following sections, we describe these three phases in detail.

3.1 Sentence Search

Sentences in Wikipedia articles usually contain anchors linking to entities. We are therefore motivated to generate the candidate entities based on the sentence search instead of the common method using entity search. There are some issues in the original annotations because of the annotation regulation. First, entities in their own pages are usually not annotated. Thus we annotate these entities with matching between the text and the page title. Second, entities are usually annotated only in their first appearance. We annotate these entities if they are annotated in previous sentences in the page. Moreover, pronouns are widely used in Wikipedia sentences and are usually not annotated. We use the Stanford CoreNLP toolkit (Manning et al., 2014) to do the coreference resolution. In addition, we use the content in the disambiguation page and the infobox. Although these two kinds of information may have incomplete grammatical structure, it contains enough context information for the sentence search in our task.

We use the Wikipedia snapshot of May 1, 2016, which contains 4.45 million pages and 120 million sentences. We extract sentences that contain at least one anchor in the Wikipedia articles, and



"[[Austin (song)|Austin]]" is the title of a debut song written by David Kent and Kirsti Manna, and performed by American country music artist [[Blake Shelton]]. -- Page:Austin (song)
In 2001, [[Blake Shelton]] made his debut with the single "[[Austin (song)|Austin]]". -- Page:Blake Shelton
Braddock is credited as producer for several of Shelton's number-one country [[music|country]] hits, including [[Bobby Braddock]]'s debut single "[[Austin (song)|Austin]]" which spent five weeks at the top of the charts. -- Page:Bobby Braddock
Both [[Shakira]] and [[Blake Shelton]] turned their chairs but she opted for [[Blake Shelton]]. -- Page:Paula Deanda
It was used in the 1960s by poster artists such as underground comic artist [[Gilbert Shelton]], who designed posters for a venue in [[Austin, Texas|Austin]], Texas called The Vulcan Gas Company. -- Page: Split-fount Inking
It was released in October 2001 as the second single from Shelton's first album, "[[Blake Shelton (album)|Blake Shelton]]". –Page: All Over Me (Blake Shelton song)
……

Figure 1: Example of the linking process of the query "blake shelton austin lyrics"

extract human-annotated anchors as well as corresponding entities in the sentences. The original annotation contains 82.6 million anchors. We obtain 110 million annotated anchors in 48.4 million sentences after the incrementally annotation. All of above annotations are indexed by Lucene[1] by building documents consisting of two fields: the first one contains the sentence and the second one contains all anchors with their corresponding entities. For each query, we search it with Lucene using its default ranker[2] based on the vector space model and tf-idf to obtain the top K sentences (K is selected on the development set). We extract all entities as the related entities and use these sentences as their support sentences.

3.2 Candidate Generation

We back-map anchors and corresponding entities extracted in sentences to generate candidates. We use (a, e) to denote the pair of the anchor text and corresponding entity and use $w(a, e)$ to denote the number of sentences containing the pair (a, e). Then, we prune the candidate pairs according to following rules.

First, we only keep the pair whose corresponding anchor text a occurs in the query as a candidate, which has been used in previous work (Ferragina and Scaiella, 2010). Second, we follow the long-string match strategy. If we have two pairs (a_1, e_1) and (a_2, e_2) while a_1 is a substring of a_2, we drop (a_1, e_1) if $w(a_1, e_1) < w(a_2, e_2)$. This is because a_2 is typically less ambiguous than a_1. For example, for the query "mesa community college football", we can obtain the anchor "mesa", "college", "community college", and "mesa community college". We only keep "mesa community college" because it is longest and occurs most times in returned sentences. However, if $w(a_1, e_1) > w(a_2, e_2)$, we keep both candidate pairs because a_1 is more common in the query.

In addition, we keep the entity whose surface form is the same with the anchor text and prune others. If we have two pairs (a, e_1) and (a, e_2) with the same anchor, and only e_2 occurs in the query, we drop the pair (a, e_1) if $w(a, e_1) < w(a, e_2)$. For example, for the query "business day south africa", the anchor "south africa" can be linked to "south africa", "union of south africa", and "south africa cricket team". We only keep the entity "south africa".

3.3 Candidate Ranking

We build a regression based framework to rank the candidate entities. In the training phase, we treat the candidates that are equal to the ground truth as the positive samples and the others as negative samples. The regression object of the positive sample is set to the score 1.0. The negative sample is set to the maximum score of overlapping ratio of tokens between its text and each gold answer. The regression object of the negative sample is not simply set to 0 in order to give a small score if the candidate is very closed to the ground truth. We find it benefits the final results. We use LIBLIN-

[1] `http://lucene.apache.org`

[2] Details can be found in `https://lucene.apache.org/core/2_9_4/api/core/org/apache/lucene/search/Similarity.html`

EAR (Fan et al., 2008) with L2-regularized L2-loss support vector regression to train the regression model. The object function is to minimize

$$w^T w/2 + C \sum \max(0, |y_i - w^T x_i| - eps)^2 \quad (1)$$

where x_i is the feature set, y_i is the object score and w is the parameter to be learned. We follow the default setting that C is set to 1 and eps is set to 0.1.

In the test phase, each candidate gets a score of $w^T x_i$ and then we only output the candidate whose score is higher than the threshold selected on the development set.

We employ four different feature sets to capture the quality of a candidate from different aspects. All features are shown in Table 3.

Context-Independent Features This feature set measures each annotation pair (a, e) without context information. Feature 1-4 catch the syntactic properties of the candidate. Feature 5 is the number of returned sentences that contain (a, e). Feature 6 is the maximum search score (returned by Lucene) in its support sentences. Moreover, inspired by TagME (Ferragina and Scaiella, 2010), we denote $freq(a)$ as the number of times the text a occurs in Wikipedia. We use $link(a)$ to denote the number of times the text a occurs as an anchor. We use $lp(a) = link(a)/freq(a)$ to denote the static link-probability that an occurrence of a has been set as an anchor. We use $freq(a, e)$ to denote the number of times that the anchor text a links to the entity e, and use $pr(e|a) = freq(a, e)/link(a)$ to denote the static prior-probability that the anchor text a links to e. Features 7 and 8 are these two probabilities.

Context-Matching Features We treat the other words except for the anchor text as the context. This feature set measures the context matching to the query. Feature 9 is the context matching score calculated by tokens. We denote c as the set of context words. For each c_i in c, the $cm\_sc(c_i)$ is the ratio of times that c_i occurs in the support sentences, and $cm\_sc(c) = \frac{1}{N} \sum cm\_sc(c_i)$. Features 10 and 11 are the ratio of context words occurring in the first sentence in the entity page and the description of entity's disambiguation page (if existed), respectively. Moreover, we train a 300-dimensional word embeddings on all Wikipedia articles by word2vec (Mikolov et al., 2013) and use the average embedding of each word as the sentence representation. Feature 12 is the maximum cosine score between the query and each support sentence. Features 13 and 14 are calculated with the first sentence in the entity's page and the description in the disambiguation page.

ID	Name	Description
1	in_query	1 if e is in the query, 0 otherwise
2	is_pt	1 if e contains parenthesis, 0 otherwise
3	is_cm	1 if e contains comma, 0 otherwise
4	len	len(e) by tokens
5	$w(a, e)$	number of support sentences
6	$sc(a, e)$	maximum search score of support sentences
7	$lp(a)$	static link-probability that a is an anchor
8	$pr(a, e)$	static prior-probability that a links to e
9	cm_sc	context matching score to the support sentences
10	cm_fs	context matching score to the first sentence of e's page
11	cm_dd	context matching score to the description in e's disambiguation page
12	$embed\_sc$	maximum embedding similarity of the query and each support sentence
13	$embed\_fs$	embedding similarity of the query and the first sentence of e's page
14	$embed\_dd$	embedding similarity of the query and the description in e's disambiguation page
15	rel_cd_sc	number of candidates that occur in the support sentences
16	rel_cd_sp	number of candidates that occur in the same Wikipedia page
17	rel_re_sc	number of related entities that occur in the support sentences
18	rel_re_sp	number of related entities that occur in the same Wikipedia page

Table 3: Feature Set for Candidate Ranking

Relatedness Features of Candidate Entities This set of features measures how much an entity is supported by other candidates. Feature 15 is the number of other candidate entities occurring in the support sentences. Feature 16 is the number of candidate entities occurring in the same Wikipedia page with the current entity.

Relatedness Features to Related Entities This set of features measures the relatedness between candidates and related entities outside of queries. Related entities can provide useful signals for disambiguating the candidates. Features 17 and 18 are analogous features with features 15 and 16, which are calculated by the related entities.

4 Experiment

We conduct experiments on the ERD14 and GERDAQ datasets. We compare with several baseline annotators and experimental results show that our method outperforms the baseline on these two datasets. We also report the parameter selection on each dataset and analyze the quality of the candidates using different methods.

4.1 Dataset

ERD14[3] is a benchmark dataset in the ERD Challenge (Carmel et al., 2014), which contains both long-text track and short-text track. In this paper we only focus on the short-text track. It contains 500 queries as the development set and 500 queries as the test set. Due to the lack of training set, we use the development set to do the model training and tuning. This dataset can be evaluated by both Freebase and Wikipedia as the ERD Challenge Organizers provide the Freebase Wikipedia Mapping with one-to-one correspondence of entities between two knowledge bases. We use Wikipedia to evaluate our results.

GERDAQ[4] is a benchmark dataset to annotate entities to Wikipedia built by Cornolti et al. (2016). It contains 500 queries for training, 250 for development, and 250 for test. The query in this dataset is sampled from the KDD-Cup 2005 and then annotated manually. Both name entities and common concepts are annotated in this dataset.

4.2 Evaluation Metric

We use average F1 designed by ERD Challenge (Carmel et al., 2014) as the evaluation metrics. Specifically, given a query q, with labeled entities $\hat{A} = \{\hat{E_1}, \ldots, \hat{E_n}\}$. We define the F-measure of a set of hypothesized interpretations $A = \{E_1, \ldots, E_m\}$ as follows:

$$Precision = \frac{|\hat{A} \cap A|}{|A|}, Recall = \frac{|\hat{A} \cap A|}{|\hat{A}|} \quad (2)$$

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

The average F1 of the evaluation set is the average of the F1 for each query:

$$AverageF_1 = \frac{1}{N} \sum_{i=1}^{N} F_1(q_i) \quad (4)$$

Following the evaluation guideline in ERD14 and GERDAQ, we define recall to be 1.0 if the gold binding of a query is empty and define precision to be 1.0 if the hypothesized interpretation is empty.

4.3 Baseline Methods

We compare with several baselines and use the results reported by the ERD organizer and Cornolti et al. (2016).

AIDA (Hoffart et al., 2011) searches the mention using Stanford NER Tagger based on YAGO2. We select AIDA as a representative system aiming to entity linking for documents following the work in Cornolti et al. (2016).

WAT (Piccinno and Ferragina, 2014) is the improved version of TagME (Ferragina and Scaiella, 2010).

Magnetic IISAS (Laclavik et al., 2014) retrieves the index extracted from Wikipedia, Freebase and Dbpedia. Then it exploits Wikipedia link graph to assess the similarity of candidate entities for disambiguation and filtering.

Seznam (Eckhardt et al., 2014) uses Wikipedia and DBpedia to generate candidates. The disambiguation step is based on PageRank over the graph.

NTUNLP (Chiu et al., 2014) searches the query to match Freebase surface forms. The disambiguation step is built on top of TagME and Wikipedia.

SMAPH-1 (Cornolti et al., 2014) is the winner in the short-text track in the ERD14 Challenge.

SMAPH-2 (Cornolti et al., 2016) is the improved version of SMAPH-1. It generates candidates from the snippets of search results returned by the Bing search engine.

4.4 Result

We report results on the ERD datset and GERDAQ dataset in Table 4 and Table 5, respectively. On the ERD14 dataset, WAT is superior to AIDA but it is still up to 10% than SMAPH-1 that wins the ERD Challenge. SMAPH-2 improves 2% than SMAPH-1. Our system significantly outperforms the state-of-the-art annotator SMAPH-2 by 4.2%. On the GERDAQ dataset, our system is 2.5% superior to the state-of-the-art annotator SMAPH-2. The F1 score in this dataset is much lower than the ERD dataset because common concepts such as “Week” and “Game” that are not annotated in the ERD dataset are annotated in the GERDAQ dataset.

Spell checking has been widely used in the baseline annotators as it is not uncommon in queries (Laclavik et al., 2014). The SMAPH system that generates candidates by search results implicitly leverages the spell-checking embedded in

[3] http://web-ngram.research.microsoft.com/erd2014/Datasets.aspx

[4] http://acube.di.unipi.it/datasets

System	$F1_{avg}$
AIDA	22.1
WAT	58.6
Magnetic IISAS	65.6
Seznam	66.9
NTUNLP	68.0
SMAPH-1	68.8
SMAPH-2	70.8
Our work	**75.0***
w/o Spell Check	74.0
w/o Additional Annotation	74.4
w/o Context Feature	72.6
w/o Relatedness Feature	74.5

Table 4: Results on the ERD dataset. Results of the baseline systems are taken from Table 8 in Cornolti et al. (2016) and reported by the ERD organizer (Carmel et al., 2014). We only report the F1 score as precision and recall are not reported in previous work. *Significant improvement over state-of-the-art baselines (t-test, $p < 0.05$).

System	P_{avg}	R_{avg}	$F1_{avg}$
AIDA	94.0	12.2	12.6
TagME	60.4	51.2	44.7
WAT	49.6	57.0	46.0
SMAPH-1	77.4	54.3	52.1
SMAPH-2	72.1	55.3	54.4
Our work	71.5	58.5	**56.9**
w/o Spell Check	75.4	48.6	49.3
w/o Additional Annotation	70.3	58.2	55.8
w/o Context Feature	69.2	56.4	55.5
w/o Relatedness Feature	73.3	57.4	56.7

Table 5: Results on the GERDAQ dataset. Results of the baseline systems are taken from Table 10 in Cornolti et al. (2016).

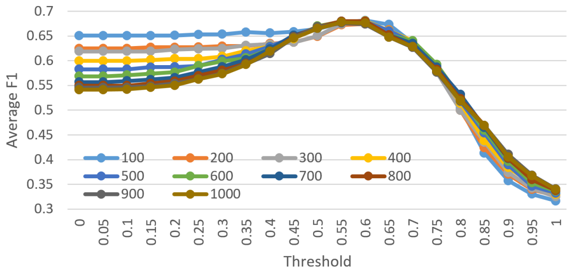


Figure 2: F1 scores with different search numbers and thresholds on the ERD development set

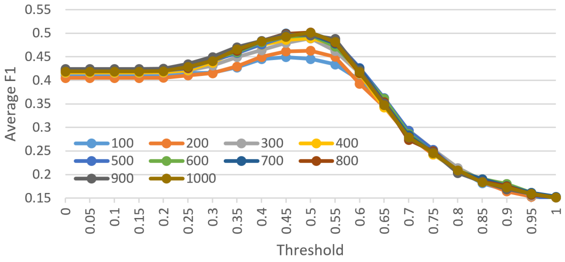


Figure 3: F1 scores with different search numbers and thresholds on the GERDAQ development set

search engines. In our experiments, spell checking improves 1.0% on the ERD dataset and 7.6% on the GERDAQ dataset. Furthermore, only 6.9% of queries in the ERD14 dataset have spelling mistakes, whereas the number in the GERDAQ dataset is 23.0%. Thus spell-checking is more important in the GERDAQ dataset.

The result decreases 0.6% on the ERD dataset and 1.1% on the GERDAQ dataset without the additional annotation. Furthermore, while the F1 score decreases 2.4% on the ERD dataset and 1.4% on the GERDAQ dataset without the context features, the score only decreases 0.5% on the ERD dataset and 0.2% on the GERDAQ dataset without the relatedness features. Unlike the work on entity linking for documents (Eckhardt et al., 2014; Witten and Milne, 2008) that features derived from entity relations get promising results, the context features play a more important role than the relatedness features on entity linking for queries as search queries are short and contain fewer entities than documents.

4.5 Parameter Selection

There are two parameters in our framework, namely the number of search sentences and the threshold for final output. We select these two parameters on the development set. We show the F1 score with different numbers of search sentences and thresholds in Figure 2 and Figure 3. On the ERD development set, better results occur in the search number between 600 and 800 as well as the threshold 0.55 and 0.6. On the GERDAQ development set, better results occur in the search number between 700 and 1000 as well as the threshold between 0.45 and 0.5. In our experiment, we set the number of sentences to 700 and the threshold to 0.56 on the ERD dataset as well as 800 and 0.48 on the GERDAQ dataset according to the F1 scores on the development set.

4.6 Model Analysis

The main difference between our method and most previous work is that we generate candidates by searching Wikipedia sentences instead of searching entities. For generating candidates with entity search, we build a dictionary containing all anchors, titles, and redirects in Wikipedia. Then we query the dictionary to get the mention and obtain corresponding entities as candidates. We use the

Method	Number of anchors	Number of candidates	$F1_{avg}$
Entity Search ES + RF	1.96	6.84	66.46 69.00
Sentence Search SS + RF	1.12	1.49	73.81 75.01

Table 6: Comparison with different candidate generation methods on the ERD dataset. +RF: integrating ranking features extracted by Sentence Search.

Method	C_{avg}	P_{avg}	R_{avg}
Entity Search	78.87	77.56	66.04
Sentence Search	74.42	89.61	69.08

Table 7: Results for the 398 queries which have at least one labeled entity on the ERD dataset using different candidate generation methods. C_{avg} is the average recall of candidates per query. P_{avg} and R_{avg} are calculated on the final results.

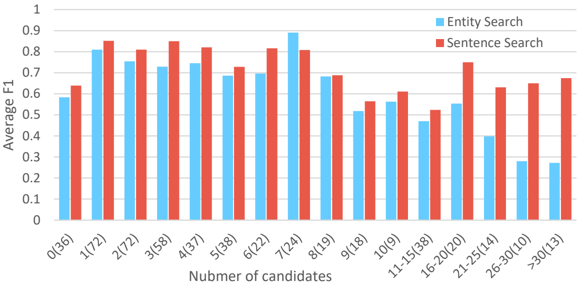


Figure 4: F1 scores with number of candidates using different methods on the ERD dataset. The number of queries is shown in the parentheses.

same pruning rules and ranking framework in our experiments, but exclude the features from support sentences because the entity search method does not contain the information. The F1 score is shown in Table 6. We achieve similar results in our implementation of the method using entity search on the ERD dataset as Magnetic IISAS (Laclavik et al., 2014) which uses a similar method and ranks 4th with the F1 of 65.57 in the ERD14 Challenge.

We compare the two candidate generation methods in several aspects. First, we show the overall results in Table 6. The average number of candidates from our method is much smaller. It is noted that the anchors from sentence search can also be found in entity search. However, we only extract the entities in the returned sentences while the methods by entity search use all entities linked by the anchors. In addition, features such as the number of sentences containing the entity from sentence search which provide query sensitive prior probability contribute to the ranking process. It improves the F1 score from 73.81 to 75.01 for sentence search and from 66.46 to 69.00 for entity search. More important, the result of "ES+RF" is still significantly worse than the result of both small candidate set and Wikipedia related features that prunes irrelevant candidates at the beginning, which proves that the high-quality candidate set is very important since the larger candidate set brings in lots of noise in training a ranking model. Moreover, there are 102 queries (20.4%) without labeled entities in the ERD dataset. We only give 7 incorrect annotations in these queries while the number is 13 from entity search. Furthermore, as shown in Table 7, the coverage of our method is lower in queries with at least one entity, but we obtain better results on precision, recall and F1 in the final stage.

Figure 4 illustrates the F1 score grouped by the number of candidates using entity search. In almost all columns the F1 score of our method is better than the baseline. In left columns (the number of candidates is less than 10), both methods generate few candidates. The F1 score of our method is higher, which proves that we train a better ranking model because of our small but quality candidate set. Moreover, the right columns (the number of candidates is more than 10) show that the F1 score using entity search gradually decreases with the incremental candidates. However, our method based on sentence search takes advantage of context information to keep a small set of candidates, which keeps a consistent result and outperforms the baseline.

5 Conclusion

In this paper we address the problem of entity linking for open-domain queries. We introduce a novel approach to generating candidate entities by searching sentences in the Wikipedia to the query, then we extract the human-annotated entities as the candidates. We implement a regression model to rank these candidates for the final output. Two experiments on the ERD dataset and the GERDAQ dataset show that our approach outperforms the baseline systems. In this work we directly use the default ranker in Lucene for similar sentences, which can be improved in future work.

Acknowledgments

We thank Ming-Wei Chang for sharing the ERD14 dataset. Chuanqi Tan and Weifeng Lv are supported by the National Natural Science Foundation of China (Grant No. 61421003). Weifeng Lv is the corresponding author of this paper.

References

Roi Blanco, Giuseppe Ottaviano, and Edgar Meij. 2015. Fast and space-efficient entity linking for queries. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 179–188. ACM.

David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June Paul Hsu, and Kuansan Wang. 2014. Erd'14: entity recognition and disambiguation challenge. In *ACM SIGIR Forum*, volume 48, pages 63–77. ACM.

Yen-Pin Chiu, Yong-Siang Shih, Yang-Yin Lee, Chih-Chieh Shao, Ming-Lun Cai, Sheng-Lun Wei, and Hsin-Hsi Chen. 2014. Ntunlp approaches to recognizing and disambiguating entities in long and short text at the erd challenge 2014. In *Proceedings of the first international workshop on Entity recognition & disambiguation*, pages 3–12. ACM.

Marco Cornolti, Paolo Ferragina, Massimiliano Ciaramita, Stefan Rüd, and Hinrich Schütze. 2016. A piggyback system for joint entity mention detection and linking in web queries. In *Proceedings of the 25th International Conference on World Wide Web*, pages 567–578. International World Wide Web Conferences Steering Committee.

Marco Cornolti, Paolo Ferragina, Massimiliano Ciaramita, Hinrich Schütze, and Stefan Rüd. 2014. The smaph system for query entity recognition and disambiguation. In *Proceedings of the first international workshop on Entity recognition & disambiguation*, pages 25–30. ACM.

W Bruce Croft, Michael Bendersky, Hang Li, and Gu Xu. 2010. Query representation and understanding workshop. In *SIGIR Forum*, volume 44, pages 48–53.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, volume 7, pages 708–716.

Alan Eckhardt, Juraj Hreško, Jan Procházka, and Otakar Smrf. 2014. Entity linking based on the cooccurrence graph and entity probability. In *Proceedings of the first international workshop on Entity recognition & disambiguation*, pages 37–44. ACM.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Michal Laclavik, Marek Ciglan, Alex Dorman, Stefan Dlugolinsky, Sam Steingold, and Martin Šeleng. 2014. A search based approach to entity recognition: magnetic and iisas team at erd challenge. In *Proceedings of the first international workshop on Entity recognition & disambiguation*, pages 63–68. ACM.

Xiao Ling, Sameer Singh, and S. Daniel Weld. 2015. Design challenges for entity linking. *Transactions of the Association of Computational Linguistics*, 3:315–328.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Edgar Meij, Wouter Weerkamp, and Maarten de Rijke. 2012. Adding semantics to microblog posts. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 563–572. ACM.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 233–242. ACM.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*.

David Milne and Ian H Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM.

Alex Olieman, Hosein Azarbonyad, Mostafa Dehghani, Jaap Kamps, and Maarten Marx. 2014. Entity linking by focusing dbpedia candidate entities. In *Proceedings of the first international workshop on Entity recognition & disambiguation*, pages 13–24. ACM.

Francesco Piccinno and Paolo Ferragina. 2014. From tagme to wat: a new entity annotator. In *Proceedings of the first international workshop on Entity recognition & disambiguation*, pages 55–62. ACM.

Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both, Martin Brümmer, Diego Ceccarelli, Marco Cornolti, Didier Cherix, Bernd Eickmann, et al. 2015. Gerbil: General entity annotator benchmarking framework. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1133–1143. ACM.

Ian Witten and David Milne. 2008. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, pages 25–30.

Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol, and Gerhard Weikum. 2011. Aida: An online tool for accurate disambiguation of named entities in text and tables. *Proceedings of the VLDB Endowment*, 4(12):1450–1453.

Train-O-Matic: Large-Scale Supervised Word Sense Disambiguation in Multiple Languages without Manual Training Data

Tommaso Pasini and **Roberto Navigli**
Department of Computer Science
Sapienza University of Rome
{pasini,navigli}@di.uniroma1.it

Abstract

Annotating large numbers of sentences with senses is the heaviest requirement of current Word Sense Disambiguation. We present Train-O-Matic, a language-independent method for generating millions of sense-annotated training instances for virtually all meanings of words in a language's vocabulary. The approach is fully automatic: no human intervention is required and the only type of human knowledge used is a WordNet-like resource. Train-O-Matic achieves consistently state-of-the-art performance across gold standard datasets and languages, while at the same time removing the burden of manual annotation. All the training data is available for research purposes at http://trainomatic.org.

1 Introduction

Word Sense Disambiguation (WSD) is a key task in computational lexical semantics, inasmuch as it addresses the lexical ambiguity of text by making explicit the meaning of words occurring in a given context (Navigli, 2009). Anyone who has struggled with frustratingly unintelligible translations from an automatic system, or with the meaning bias of search engines, can understand the importance for an intelligent system to go beyond the surface appearance of text.

There are two mainstream lines of research in WSD: supervised and knowledge-based WSD. Supervised WSD frames the problem as a classical machine learning task in which, first a training phase occurs aimed at learning a classification model from sentences annotated with word senses and, second the model is applied to previously-unseen sentences focused on a target word. A key difference from many other problems, however, is that the classes to choose from (i.e., the senses of a target word) vary for each word, therefore requiring a separate training process to be performed on a word by word basis. As a result, hundreds of training instances are needed for each ambiguous word in the vocabulary. This would necessitate a million-item training set to be manually created for each language of interest, an endeavour that is currently beyond reach even in resource-rich languages like English.

The second paradigm, i.e., knowledge-based WSD, takes a radically different approach: the idea is to exploit a general-purpose knowledge resource like WordNet (Fellbaum, 1998) to develop an algorithm which can take advantage of the structural and lexical-semantic information in the resource to choose among the possible senses of a target word occurring in context. For example, a PageRank-based algorithm can be developed to determine the probability of a given sense being reached starting from the senses of its context words. Recent approaches of this kind have been shown to obtain competitive results (Agirre et al., 2014; Moro et al., 2014). However, due to its inherent nature, knowledge-based WSD tends to adopt bag-of-word approaches which do not exploit the local lexical context of a target word, including function and collocation words, which limits this approach in some cases.

In this paper we get the best of both worlds and present Train-O-Matic, a novel method for generating huge high-quality training sets for all the words in a language's vocabulary. The approach is language-independent, thanks to its use of a multilingual knowledge resource, BabelNet (Navigli and Ponzetto, 2012), and it can be applied to any kind of corpus. The training sets produced with Train-O-Matic are shown to provide competitive performance with those of manually and semi-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 78–88
Copenhagen, Denmark, September 7–11, 2017.

automatically tagged corpora. Moreover, state-of-the-art performance is also reported for low resourced languages (i.e., Italian and Spanish) and domains, where manual training data is not available.

2 Building a Training Set from Scratch

In this Section we present Train-O-Matic, a language-independent approach to the automatic construction of a sense-tagged training set. Train-O-Matic takes as input a corpus C (e.g., Wikipedia) and a semantic network $G = (V, E)$. We assume a WordNet-like structure of G, i.e., V is the set of concepts (i.e., synsets) such that, for each word w in the vocabulary, $Senses(w)$ is the set of vertices in V that are expressed by w, e.g., the WordNet synsets that include w as one of their senses.

Train-O-Matic consists of three steps:

- **Lexical profiling:** for each vertex in the semantic network, we compute its Personalized PageRank vector, which provides its lexical-semantic profile (Section 2.1).
- **Sentence scoring:** For each sentence containing a word w, we compute a probability distribution over all the senses of w based on its context (Section 2.2).
- **Sentence ranking and selection:** for each sense s of a word w in the vocabulary, we select those sentences that are most likely to use w in the sense of s (Section 2.3).

2.1 Lexical profiling

In terms of semantic networks the probability of reaching a node v' starting from v can be interpreted as a measure of relatedness between the synsets v and v'. Thus we define the lexical profile of a vertex v in a graph $G = (V, E)$ as the probability distribution over all the vertices v' in the graph. Such distribution is computed by applying the Personalized PagaRank algorithm, a variant of the traditional PageRank (Brin and Page, 1998). While the latter is equivalent to performing random walks with uniform restart probability on every vertex at each step, PPR, on the other hand, makes the restart probability non-uniform, thereby concentrating more probability mass in the surroundings of those vertices having higher restart probability. Formally, (P)PR is computed as follows:

$$v^{(t+1)} = (1 - \alpha)v^{(0)} + \alpha M v^{(t)} \tag{1}$$

where M is the row-normalized adjacency matrix of the semantic network, the restart probability distribution is encoded by vector $v^{(0)}$, and α is the well-known damping factor usually set to 0.85 (Brin and Page, 1998). If we set $v^{(0)}$ to a unit probability vector $(0, \ldots, 0, 1, 0, \ldots, 0)$, i.e., restart is always on a given vertex, PPR outputs the probability of reaching every vertex starting from the restart vertex after a certain number of steps. This approach has been used in the literature to create semantic signatures (i.e., profiles) of individual concepts, i.e., vertices of the semantic network (Pilehvar et al., 2013), and then to determine the semantic similarity of concepts. As also done by Pilehvar and Collier (2016), we instead use the PPR vector as an estimate of the conditional probability of a word w' given the target sense[1] $s \in V$ of word w:

$$P(w'|s, w) = \frac{\max_{s' \in Senses(w')} v_s(s')}{Z} \tag{2}$$

where $Z = \sum_{w"} P(w"|s, w)$ is a normalization constant, v_s is the vector resulting from an adequate number of random walks used to calculate PPR, and $v_s(s')$ is the vector component corresponding to sense s'. To fix the number of iterations needed to have a sufficiently accurate vector, we follow Lofgren et al. (2014) and set the error $\delta = 0.00001$ and the number of iterations to $\frac{1}{\delta} = 100,000$.

As a result of this lexical profiling step we have a probability distribution over vocabulary words for each given word sense of interest.

2.2 Sentence scoring

The objective of the second step is to score the importance of word senses for each of the corpus sentences which contain the word of interest. Given a sentence $\sigma = w_1, w_2, \ldots, w_n$, for a given target word w in the sentence ($w \in \sigma$), and for each of its senses $s \in Senses(w)$, we compute the probability $P(s|\sigma, w)$. Thanks to Bayes' theorem we can determine the probability of sense s of w given the

[1] Note that we use senses and concepts (synsets) interchangeably, because – given a word – a word sense unambiguously determines a concept (i.e., the synset it is contained in) and vice versa.

sentence as follows:

$$P(s|\sigma, w) = \frac{P(\sigma|s, w)P(s|w)}{P(\sigma|w)} \quad (3)$$

$$= \frac{P(w_1, \dots, w_n|s, w)P(s|w)}{P(w_1, \dots, w_n|w)}$$

$$\propto P(w_1, \dots, w_n|s, w)P(s|w) \quad (4)$$

$$\approx P(w_1|s, w) \dots P(w_n|s, w)P(s|w) \quad (5)$$

where Formula 4 is proportional to the original probability (due to removing the constant in the denominator) and is approximated with Formula 5 due to the assumption of independence of the words in the sentence. $P(w_i|s, w)$ is calculated as in Formula 2 and $P(s|w)$ is set to $1/|Senses(w)|$ (recall that s is a sense of w). For example, given the sentence $\sigma =$"A match is a tool for starting a fire", the target word $w = match$ and its set of senses $S_{match} = \{s^1_{match}, s^2_{match}\}$, where s^1_{match} is the sense of *lighter* and s^2_{match} is the sense of *game match*, we want to calculate the probability of each $s^i_{match} \in S_{match}$ of being the correct sense of *match* in the sentence σ. Following Formula 5 we have:

$$\begin{aligned} P(s^1_{match}|\sigma, match) \approx{} & \\ & P(\text{tool}|s^1_{match}, match) \\ & \cdot P(\text{start}|s^1_{match}, match) \\ & \cdot P(\text{fire}|s^1_{match}, match) \\ & \cdot P(s^1_{match}|match) \\ & = 2.1 \cdot 10^{-4} \cdot 2 \cdot 10^{-3} \cdot 10^{-2} \cdot 5 \cdot 10^{-1} \\ & = 2.1 \cdot 10^{-9} \end{aligned}$$

$$\begin{aligned} P(s^2_{match}|\sigma, match) \approx{} & \\ & P(\text{tool}|s^2_{match}, match) \\ & \cdot P(\text{start}|s^2_{match}, match) \\ & \cdot P(\text{fire}|s^2_{match}, match) \\ & \cdot P(s^2_{match}|match) \\ & = 10^{-5} \cdot 2.9 \cdot 10^{-4} \cdot 10^{-6} \cdot 5 \cdot 10^{-1} \\ & = 1.45 \cdot 10^{-15} \end{aligned}$$

As can be seen, the first sense of *match* has a much higher probability due to its stronger relatedness to the other words in the context (i.e. *start*, *fire* and *tool*). Note also that all the probabilities for the second sense are at least one magnitude less than the probability of the first sense.

2.3 Sense-based sentence ranking and selection

Finally, for a given word w and a given sense $s_1 \in Senses(w)$, we score each sentence σ in which w appears and s_1 is its most likely sense according to a formula that takes into account the difference between the first (i.e., s_1) and the second most likely sense of w in σ:

$$\Delta_{s_1}(\sigma) = P(s_1|\sigma, w) - P(s_2|\sigma, w) \quad (6)$$

where $s_1 = \arg\max_{s \in Senses(w)} P(s|\sigma, w)$, and $s_2 = \arg\max_{s \in Senses(w) \setminus \{s_1\}} P(s|\sigma, w)$. We then sort all sentences based on $\Delta_{s_1}(\cdot)$ and return a ranked list of sentences where word w is most likely to be sense-annotated with s_1. Although we recognize that other scoring strategies could have been used, this was experimentally the most effective one when compared to alternative strategies, i.e., the sense probability, the number of words related to the target word w, the sentence length or a combination thereof.

3 Creating a Denser and Multilingual Semantic Network

In the previous Section we assumed that WordNet was our semantic network, with synsets as vertices and edges represented by its semantic relations. However, while its lexical coverage is high, with a rich set of fine-grained synsets, at the relation level WordNet provides mainly paradigmatic information, i.e., relations like hypernymy (is-a) and meronymy (part-of). It lacks, on the other hand, syntagmatic relations, such as those that connect verb synsets to their arguments (e.g., the appropriate senses of eat_v and $food_n$), or pairs of noun synsets (e.g., the appropriate senses of bus_n and $driver_n$).

Intuitively, Train-O-Matic would suffer from such a lack of syntagmatic relations, as the relevance of a sense for a given word in a sentence depends directly on the possibility of visiting senses of the other words in the same sentence (cf. Formula 5) via random walks as calculated with Formula 1. Such reachability depends on the connections available between synsets. Because syntagmatic relations are sparse in WordNet, if it was used on its own, we would end up with a poor ranking of sentences for any given word sense. Moreover, even though the methodology presented in Section 2 is language-independent, Train-O-Matic would lack informa-

mouse (animal)		mouse (device)	
WordNet	WordNet$_{BN}$	WordNet	WordNet$_{BN}$
mouse$_n^1$	mouse$_n^1$	mouse$_n^4$	mouse$_n^4$
tail$_n^1$	little$_a^1$	wheel$_n^1$	computer$_n^1$
hairless$_a^1$	rodent$_n^1$	electronic_device$_n^1$	pad$_n^4$
rodent$_n^1$	cheese$_n^1$	ball$_n^3$	cursor$_n^1$
trunk$_n^3$	cat$_n^1$	hand_operated$_n^1$	operating_system$_n^1$
elongate$_a^2$	rat$_n^1$	mouse_button$_n^1$	trackball$_n^1$
house_mouse$_n^1$	elephant$_n^1$	cursor$_n^1$	wheel$_n^1$
minuteness$_n^1$	pet$_n^1$	operate$_v^3$	joystick$_n^1$
nude_mouse$_n^1$	experiment$_n^1$	object$_n^1$	Windows$_n^1$

Table 1: Top-ranking synsets of the PPR vectors computed on WordNet (first and third columns) and WordNet$_{BN}$ (second and fourth columns) for *mouse* as animal (left) and as device (right).

tion (e.g. senses for a word in an arbitrary vocabulary) for languages other than English.

To cope with these issues, we exploit BabelNet,[2] a huge multilingual semantic network obtained from the automatic integration of WordNet, Wikipedia, Wiktionary and other resources (Navigli and Ponzetto, 2012), and create the BabelNet subgraph induced by the WordNet vertices. The result is a graph whose vertices are BabelNet synsets that contain at least one WordNet synset and whose edge set includes all those relations in BabelNet coming either from WordNet itself or from links in other resources mapped to WordNet (such as hyperlinks in a Wikipedia article connecting it to other articles). The greatest contribution of syntagmatic relations comes, indeed, from Wikipedia, as its articles are linked to related articles (e.g., the English Wikipedia *Bus* article[3] is linked to *Passenger*, *Tourism*, *Bus lane*, *Timetable*, *School*, and many more).

Because not all Wikipedia (and other resources') pages are connected with the same degree of relatedness (e.g., countries are often linked, but they are not necessarily closely related to the source article in which the link occurs), we apply the following weighting strategy to each edge $(s, s') \in E$ of our WordNet-induced subgraph of BabelNet $G = (V, E)$:

$$w(s, s') = \begin{cases} 1 & (s, s') \in E(\text{WordNet}) \\ WO(s, s') & \text{otherwise} \end{cases} \tag{7}$$

where E(WordNet) is the edge set of the original WordNet graph and $WO(s, s')$ is the weighted overlap measure which calculates the similarity between two synsets:

$$WO(s, s') = \frac{\sum_{i=1}^{|S|}(r_i^1 + r_i^2)^{-1}}{\sum_{i=1}^{|S|}(2i)^{-1}}$$

where r_i^1 and r_i^2 are the rankings of the i-th synsets in the set S of the components in common between the vectors associated with s and s', respectively. Because at this stage we still have to calculate our synset vector representation, we use the precomputed NASARI vectors (Camacho-Collados et al., 2015) to calculate WO. This choice is due to WO's higher performance over cosine similarity for vectors with explicit dimensions (Pilehvar et al., 2013).

As a result, each row of the original adjacency matrix M of G will be replaced with the weights calculated in Formula 7 and then normalized in order to be ready for PPR calculation (see Formula 1). An idea of why a denser semantic network has more useful connections and thus leads to better results is provided by the example in Table 1[4], where we show the highest-probability synsets in the PPR vectors calculated with Formula 1 for two different senses of *mouse* (its animal and device senses) when WordNet (left) and our WordNet-induced subgraph of BabelNet (WordNet$_{BN}$, right) are used as the underlying semantic network for PPR computation. Note that WordNet's top synsets are related to the target synset via paradigmatic (i.e., hypernymy and meronymy) relations, while WordNet$_{BN}$ includes many syntagmatically-related synsets (e.g., *exper-*

[2] `http://babelnet.org`

[3] Retrieved on February 3rd, 2017.

[4] We use the notation w_p^k introduced in (Navigli, 2009) to denote the k-th sense of word w with part-of-speech tag p.

iment for the animal, and *operating system* and *Windows* for the device sense, among others).

4 Experimental Setup

Corpora for sense annotation We used two different corpora to extract sentences: Wikipedia and the United Nations Parallel Corpus (Ziemski et al., 2016). The first is the largest and most up-to-date encyclopedic resource, containing definitional information, the second, on the other hand, is a public collection of parliamentary documents of the United Nations. The application of Train-O-Matic to the two corpora produced two sense-annotated datasets, which we named T-O-M$_{Wiki}$ and T-O-M$_{UN}$, respectively.

Semantic Network We created sense-annotated corpora with Train-O-Matic both when using PPR vectors computed from vanilla WordNet and when using WordNet$_{BN}$, our denser network obtained from the WordNet-induced subgraph of BabelNet (see Section 3).

Gold standard datasets We performed our evaluations using the framework made available by Raganato et al. (2017a) on five different all-words datasets, namely: the Senseval-2 (Edmonds and Cotton, 2001), Senseval-3 (Snyder and Palmer, 2004), SemEval-2007 (Pradhan et al., 2007), SemEval-2013 (Navigli et al., 2013) and SemEval-2015 (Moro and Navigli, 2015) WSD datasets. We focused on nouns only, given the fact that Wikipedia provides connections between nominal synsets only, and therefore contributes mainly to syntagmatic relations between nouns.

Comparison sense-annotated corpora To show the impact of our T-O-M corpora in WSD, we compared its performance on the above gold standard datasets, against training with:

- **SemCor** (Miller et al., 1993), a corpus containing about 226,000 words annotated manually with WordNet senses.

- **One Million Sense-Tagged Instances** (Taghipour and Ng, 2015, OMSTI), a sense-annotated dataset obtained via a semi-automatic approach based on the disambiguation of a parallel corpus, i.e., the United Nations Parallel Corpus, performed by exploiting manually translated word senses. Because OMSTI integrates SemCor to increase coverage, to keep a level playing field we excluded the latter from the corpus.

We note that T-O-M, instead, is fully automatic and does not require any WSD-specific human intervention nor any aligned corpus.

Reference system In all our experiments, we used It Makes Sense (Zhong and Ng, 2010, IMS), a state-of-the-art WSD system based on linear Support Vector Machines, as our reference system for comparing its performance when trained on T-O-M, against the same WSD system trained on other sense-annotated corpora (i.e., SemCor and OMSTI). Following the WSD literature, unless stated otherwise, we report performance in terms of F1, i.e., the harmonic mean of precision and recall.

We note that it is not the purpose of this paper to show that T-O-M, when integrated into IMS, beats all other configurations or alternative systems, but rather to fully automatize the WSD pipeline with performances which are competitive with the state of the art.

Baseline As a traditional baseline in WSD, we used the Most Frequent Sense (MFS) baseline given by the first sense in WordNet. The MFS is a very competitive baseline, due to the sense skewness phenomenon in language (Navigli, 2009).

Number of training sentences per sense Given a target word w, we sorted its senses $Senses(w)$ following the WordNet ordering and selected the top k_i training sentences for the i-th sense according to Formula 6, where:

$$k_i = \frac{1}{i^z} * K \tag{8}$$

with $K = 500$ and $z = 2$ which were tuned on a separate small in-house development dataset[5].

5 Results

5.1 Impact of syntagmatic relations

The first result we report regards the impact of vanilla WordNet vs. our WordNet-induced subgraph of BabelNet (WordNet$_{BN}$) when calculating PPR vectors. As can be seen from Table 2 – which shows the performance of the T-O-M$_{Wiki}$ corpora generated with the two semantic networks – using WordNet for PPR computation decreases

[5] 50 word-sense pairs annotated manually.

Dataset	T-O-M$_{Wiki}$ BN	T-O-M$_{Wiki}$ WN
Senseval-2	**70.5**	70.0
Senseval-3	**67.4**	63.1
SemEval-07	**59.8**	57.9
SemEval-13	**65.5**	63.7
SemEval-15	68.6	**69.5**
ALL	**67.3**	65.7

Table 2: F1 of IMS trained on T-O-M when PPR is obtained from the WordNet graph (WN) and from the WordNet-induced subgraph of BabelNet (BN).

the overall performance of IMS from 0.5 to around 4 points across the five datasets, with an overall loss of 1.6 F1 points. Similar performance losses were observed when using T-O-M$_{UN}$ (see Table 3). This corroborates our hunch discussed in Section 3 that a resource like BabelNet can contribute important syntagmatic relations that are beneficial for identifying (and ranking high) sentences which are semantically relevant for the target word sense. In the following experiments, we report only results using WordNet$_{BN}$.

5.2 Comparison against sense-annotated corpora

We now move to comparing the performance of T-O-M, which is fully automatic, against corpora which are annotated manually (SemCor) and semi-automatically (OMSTI). In Table 3 we show the F1-score of IMS on each gold standard dataset in the evaluation framework and on all datasets merged together (last row), when it is trained with the various corpora described above.

As can be seen, T-O-M$_{Wiki}$ and T-O-M$_{UN}$ obtain higher performance than OMSTI (up to 5.5 points above) on 3 out of 5 datasets, and, overall, T-O-M$_{Wiki}$ scores 1 point above OMSTI. The MFS is in the same ballpark as T-O-M$_{Wiki}$, performing better on some datasets and worse on others. We note that IMS trained on T-O-M$_{Wiki}$ succeeds in surpassing or obtaining the same results as IMS trained on SemCor on SemEval-15 and SemEval-13. We view this as a significant achievement given the total absence of manual effort involved in T-O-M. Because overall T-O-M$_{Wiki}$ outperforms T-O-M$_{UN}$, in what follows we report all the results with T-O-M$_{Wiki}$, except for the domain-oriented evaluation (see Section 5.4).

5.3 Performance without backoff strategy

IMS uses the MFS as a backoff strategy when no sense can be output for a target word in context (Zhong and Ng, 2010). Consequently, the performance of the MFS is mixed up with that of the SVM classifier. As shown in Table 4, OMSTI is able to provide annotated sentences for roughly half of the tokens in the datasets. Train-O-Matic, on the other hand, is able to cover almost all words in each dataset with at least one training sentence. This means that in around 50% of cases OMSTI gives an answer based on the IMS backoff strategy.

To determine the real impact of the different training data, we therefore decided to perform an additional analysis of the IMS performance when the MFS backoff strategy is disabled. Because we suspected the system would not always return a sense for each target word, in this experiment we measured precision, recall and their harmonic mean, i.e., F1. The results in Table 5 confirm our hunch, showing that OMSTI's recall drops heavily, thereby affecting F1 considerably. T-O-M performances, instead, remain high in terms of precision, recall and F1. This confirms that OMSTI relies heavily on data (those obtained for the MFS and from SemCor) that are produced manually, rather than semi-automatically.

5.4 Domain-oriented WSD

To further inspect the ability of T-O-M to enable disambiguation in different domains, we decided to evaluate on specific documents from the various gold standard datasets which could be clearly assigned a domain label. Specifically, we tested on 13 SemEval-13 documents from various domains[6] and 2 SemEval-15 documents (namely, maths & computers, and biomedicine) and carried out two separate tests and evaluations of T-O-M on each domain: once using the MFS backoff strategy, and once not using it. In Tables 6 and 7 we report the results of both T-O-M$_{Wiki}$ and T-O-M$_{UN}$ to determine the impact of the corpus type.

As can be seen in the tables, T-O-M$_{Wiki}$ systematically attains higher scores than OMSTI (except for the biology domain), and, in most cases, attains higher scores than MFS when the backoff is used, with a drastic, systematic increase over OMSTI with both Train-O-Matic configurations

[6] Namely biology, climate, finance, health care, politics, social issues and sport.

Dataset	Train-O-Matic$_{Wiki}$	Train-O-Matic$_{UN}$	OMSTI	SemCor	MFS
Senseval-2	70.5	69.0	74.1	**76.8**	72.1
Senseval-3	67.4	68.3	67.2	**73.8**	72.0
SemEval-07	59.8	57.9	62.3	**67.3**	65.4
SemEval-13	**65.5**	62.5	62.8	**65.5**	63.0
SemEval-15	**68.6**	63.5	63.1	66.1	66.3
ALL	67.3	65.3	66.4	**70.4**	67.6

Table 3: F1 of IMS trained on Train-O-Matic, OMSTI and SemCor, and MFS for the Senseval-2, Senseval-3, SemEval-07, SemEval-13 and SemEval-15 datasets.

Dataset	OMSTI	Train-O-Matic	Total
Senseval-2	469	1005	1066
Senseval-3	494	860	900
Semeval-07	89	159	159
Semeval-13	757	1428	1644
Semeval-15	249	494	531
ALL	2058	3946	4300

Table 4: Number of nominal tokens for which at least one training example is provided by OMSTI or Train-O-Matic for each dataset.

Dataset	OMSTI			Train-O-Matic		
	P	R	F1	P	R	F1
Senseval-2	64.8	28.5	39.6	69.5	65.5	**67.4**
Senseval-3	55.7	31.0	39.8	66.1	63.1	**64.6**
SemEval-07	64.1	35.9	46.0	59.8	59.8	**59.8**
SemEval-13	50.7	23.4	32.0	61.3	53.3	**57.0**
SemEval-15	57.0	26.7	36.4	67.0	62.3	**64.6**
ALL	56.5	27.0	36.5	65.1	59.7	**62.3**

Table 5: Precision, Recall and F1 of IMS trained on OMSTI and Train-O-Matic corpus without MFS backoff strategy for Senseval-2, Senseval-3, SemEval-07, SemEval-13 and SemEval-15.

in recall and F1 when the backoff strategy is disabled. This demonstrates the usefulness of the corpora annotated by Train-O-Matic not only on open text, but also on specific domains. We note that T-O-M$_{UN}$ obtains the best results in the politics domain, which is the closest domain to the UN corpus from which its training sentences are obtained.

6 Scaling up to Multiple Languages

Experimental Setup In this section we investigate the ability of Train-O-Matic to scale to low-resourced languages, such as Italian and Spanish, for which training data for WSD is not available.

Thanks to BabelNet, in fact, Train-O-Matic can be used to generate sense-annotated data for any language supported by the knowledge base. Thus, in order to build new training datasets for the two languages, we ran Train-O-Matic on their corresponding versions of Wikipedia, then we tuned the two parameters K and z on an in-house development dataset[7]. In contrast to the English setting, in order to calculate Formula 8 we sorted the senses of each word by vertex degree. Finally we used the output data to train IMS.

Results To perform our evaluation we chose the most recent multilingual task (SemEval 2015 task 13) which includes gold data for Italian and Spanish. As can be seen from Table 8 Train-O-Matic enabled IMS to perform better than the best participating system (Manion and Sainudiin, 2014, SUDOKU) in all three settings (All domains, Maths & Computer and Biomedicine). Its performance was in fact, 1 to 3 points higher, with a 6-point peak on Maths & Computer in Spanish and on Biomedicine in Italian. This demonstrates the ability of Train-O-Matic to enable supervised WSD systems to surpass state-of-the-art knowledge-based WSD approaches in low-resourced languages without relying on manually curated data for training.

7 Related Work

There are two mainstream approaches to Word Sense Disambiguation: supervised and knowledge-based approaches. Both suffer in different ways from the so-called knowledge acquisition bottleneck, that is, the difficulty in obtaining an adequate amount of lexical-semantic data: for training in the case of supervised systems, and for enriching semantic networks in the case of knowledge-based ones (Pilehvar and

[7] We set $K = 100$ and $z = 2.3$ for Spanish and $K = 100$ and $z = 2.5$ for Italian.

Domain	Backoff	T-O-M$_{Wiki}$			T-O-M$_{UN}$			OMSTI			SemCor	MFS	Size
		P	R	F1	P	R	F1	P	R	F1	F1	F1	
Biology	MFS	63.0	63.0	63.0	65.9	65.9	**65.9**	65.9	65.9	**65.9**	66.3	64.4	135
	-	59.0	53.3	56.0	62.3	56.3	**59.2**	48.1	18.5	26.7	-		
Climate	MFS	68.1	68.1	**68.1**	63.4	63.4	63.4	68.0	68.0	68.0	70.1	67.5	194
	-	63.4	50.0	**55.9**	57.5	45.4	50.7	58.0	24.2	34.2	-		
Finance	MFS	68.0	68.0	**68.0**	56.6	56.6	56.6	64.4	64.4	64.4	63.7	56.2	219
	-	62.1	51.6	**56.4**	48.4	40.2	43.9	57.4	28.3	37.9	-		
Health Care	MFS	65.2	65.2	**65.2**	60.1	60.1	60.1	52.9	52.9	52.9	62.7	56.5	138
	-	61.3	55.1	**58.0**	55.6	50.0	52.6	34.6	18.4	24.0	-		
Politics	MFS	65.2	65.2	65.2	66.3	66.3	**66.3**	63.4	63.4	63.4	69.5	67.7	279
	-	62.5	54.8	58.4	63.9	55.9	**59.6**	54.1	21.5	30.8	-		
Social Issues	MFS	68.5	68.5	**68.5**	63.6	63.6	63.6	65.6	65.6	65.6	66.8	67.6	349
	-	63.1	53.0	**57.6**	57.2	47.9	52.1	54.7	25.2	34.5	-		
Sport	MFS	60.3	60.3	60.3	60.9	60.9	**60.9**	58.8	58.8	58.8	60.4	57.6	330
	-	58.3	54.6	**56.4**	58.1	53.3	55.5	45.0	23.0	30.4	-		

Table 6: Performance comparison over SemEval-2013 domain-specific datasets.

Domain	Backoff	T-O-M$_{Wiki}$			T-O-M$_{UN}$			OMSTI			SemCor	MFS	Size
		P	R	F1	P	R	F1	P	R	F1	F1	F1	
Biomedicine	MFS	76.3	76.3	**76.3**	66.0	66.0	66.0	64.9	64.9	64.9	70.3	71.1	100
	-	76.1	72.2	**74.1**	64.4	59.8	62.0	60.5	26.8	37.2	-		
Maths & Computer	MFS	50.0	50.0	**50.0**	48.0	48.0	48.0	36.0	36.0	36.0	40.6	40.9	97
	-	50.0	47.0	**48.5**	47.8	44.0	45.8	21.2	11.0	14.5	-		

Table 7: Performance comparison over the Biomedical and Maths & Computer domains in SemEval-15.

Language	Dataset	Best System	Train-O-Matic		
		F1	P	R	F1
Italian	ALL	56.6	65.1	55.6	**59.9**
	Computers & Math	46.6	52.7	43.3	**47.6**
	Biomedicine	65.9	76.6	67.6	**71.8**
Spanish	ALL	56.3	61.3	54.8	**57.9**
	Computers & Math	42.4	53.3	44.4	**48.5**
	Biomedicine	65.5	71.8	65.5	**68.5**

Table 8: Performance comparison between T-O-M and SemEval-2015's best SUDOKU Run.

Navigli, 2014; Navigli, 2009).

State-of-the-art supervised systems include Support Vector Machines such as IMS (Zhong and Ng, 2010) and, more recently, LSTM neural networks with attention and multitask learning (Raganato et al., 2017b) as well as LSTMs paired with nearest neighbours classification (Melamud et al., 2016; Yuan et al., 2016). The latter also integrates a label propagation algorithm in order to enrich the sense annotated dataset. The main difference from our approach is its need for a manually annotated dataset to start the label propagation algorithm, whereas Train-O-Matic is fully automatic. An evaluation against this system would have been interesting, but neither the proprietary training data nor the code are available at the time of writing.

In order to generalize effectively, these supervised systems require large numbers of training instances annotated with senses for each target word occurrence. Overall, this amounts to millions of training instances for each language of interest, a number that is not within reach for any language. In fact, no supervised system has been submitted in major multilingual WSD competitions for languages other than English (Navigli et al., 2013; Moro and Navigli, 2015). To overcome this problem, new methodologies have recently been developed which aim to create sense-tagged corpora automatically. Raganato et al. (2016) developed 7 heuristics to grow the number of hyperlinks in Wikipedia pages. Otegi et al. (2016) applied a different disambiguation pipeline for each language to parallel text in Europarl (Koehn, 2005) and QTLeap (Agirre et al., 2015) in order to enrich them with semantic annotations. Taghipour and Ng (2015), the work closest to ours, exploits the alignment from English to Chinese sentences of the United Nation Parallel Corpus (Ziemski et al., 2016) to reduce the ambiguity of English words and sense-tag English sentences. The assumption is that the second language is less ambiguous than the first one and that hand-made translations of senses are available for each WordNet synset. This approach is, therefore, semi-automatic and relies on certain assumptions, in contrast to Train-O-Matic which is, instead, fully automatic and can be applied to any kind of corpus (and language) depending on the specific need. Earlier attempts at the automatic extraction of training samples were made by Agirre and De Lacalle (2004) and Fernández et al. (2004). Both exploited the monosemous relatives method (Leacock et al., 1998) in order to retrieve sentences from the Web which contained a given monosemous noun or a relative monosemous word (e.g., a synonym, a hypernym, etc.). As can be seen in (Fernández et al., 2004) this approach can lead to the retrieval of very accurate examples, but its main drawback lies in the number of senses covered. In fact, for all those synsets that do not have any monosemous relative, the system is unable to retrieve examples, thus heavily affecting the performance in terms of recall and F1.

Knowledge-based WSD, instead, bypasses the heavy requirement of sense-annotated corpora by applying algorithms that exploit a general-purpose semantic network, such as WordNet, which encodes the relational information that interconnects synsets via different kinds of relation. Approaches include variants of Personalized PageRank (Agirre et al., 2014) and densest subgraph approximation algorithms (Moro et al., 2014) which, thanks to the availability of multilingual resources such as BabelNet, can easily be extended to perform WSD in arbitrary languages. Other approaches to knowledge-based WSD exploit the definitional knowledge contained in a dictionary. The Lesk algorithm (Lesk, 1986) and its variants (Banerjee and Pedersen, 2002; Kilgarriff and Rosenzweig, 2000; Vasilescu et al., 2004) aim to determine the correct sense of a word by comparing each word-sense definition with the context in which the target word appears. The limit of knowledge-based WSD, however, lies in the absence of mechanisms that can take into account the very local context of a target word occurrence, including non-content words such as prepositions and articles. Furthermore, recent studies seem to suggest that such approaches are barely able to surpass supervised WSD systems when they enrich their networks starting from a comparable amount of annotated data (Pilehvar and Navigli, 2014). With T-O-M, rather than further enriching an existing semantic network, we exploit the information available in the network to annotate raw sentences with sense information and train a state-of-the-art supervised WSD system without task-specific human annotations.

8 Conclusion

In this paper we presented Train-O-Matic, a novel approach to the automatic construction of large training sets for supervised WSD in an arbitrary language. Train-O-Matic removes the burden of manual intervention by leveraging the structural semantic information available in the WordNet graph enriched with additional relational information from BabelNet, and achieves performance competitive to that of semi-automatic approaches and, in some cases, of manually-curated training data. T-O-M was shown to provide training data for virtually all the target ambiguous nouns, in marked contrast to alternatives like OMSTI, which covers in many cases around half of the tokens, resorting to the MFS otherwise. Moreover Train-O-Matic has proven to scale well to low-resourced languages, for which no manually annotated dataset exists, surpassing the current state of the art of knowledge-based systems.

We believe that the ability of T-O-M to overcome the current paucity of annotated data for WSD, coupled with video games with a purpose for validation purposes (Jurgens and Navigli, 2014; Vannella et al., 2014), paves the way for high-quality multilingual supervised WSD. All the training corpora, including approximately one million sentences which cover English, Italian and Spanish, are made available to the community at `http://trainomatic.org`.

As future work we plan to extend our approach to verbs, adjectives and adverbs. Following Bennett et al. (2016) we will also experiment on more realistic estimates of $P(s|w)$ in Formula 5 as well as other assumptions made in our work.

Acknowledgments

The authors gratefully acknowledge the support of the ERC Consolidator Grant MOUSSE No. 726487.

References

Eneko Agirre, António Branco, Martin Popel, and Kiril Simov. 2015. Europarl QTLeap WSD/NED corpus. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University in Prague.

Eneko Agirre and Oier Lopez De Lacalle. 2004. Publicly available topic signatures for all wordnet nominal senses. In *LREC*.

Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

Satanjeev Banerjee and Ted Pedersen. 2002. An adapted Lesk algorithm for word sense disambiguation using wordnet. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 136–145. Springer.

Andrew Bennett, Timothy Baldwin, Jey Han Lau, Diana McCarthy, and Francis Bond. 2016. Lexsemtm: A semantic dataset based on all-words unsupervised sense distribution learning. In *Proceedings of the 54nd Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1513 – 1524, Berlin, Germany.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Nasari: a novel approach to a semantically-aware representation of items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 567–577, Denver, Colorado. Association for Computational Linguistics.

Philip Edmonds and Scott Cotton. 2001. Senseval-2: overview. In *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5. Association for Computational Linguistics.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Database*. MIT Press, Cambridge, MA.

Juan Fernández, Mauro Castillo Valdés, German Rigau Claramunt, Jordi Atserias Batalla, and Jordi Tormo. 2004. Automatic acquisition of sense examples using exretriever. In *IBERAMIA Workshop on Lexical Resources and The Web for Word Sense Disambiguation*, pages 97–104.

David Jurgens and Roberto Navigli. 2014. It's All Fun and Games until Someone Annotates: Video Games with a Purpose for Linguistic Annotation. *Transactions of the Association for Computational Linguistics (TACL)*, 2:449–464.

Adam Kilgarriff and Joseph Rosenzweig. 2000. Framework and results for english SENSEVAL. *Computers and the Humanities*, 34(1–2):15–48.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Claudia Leacock, George A Miller, and Martin Chodorow. 1998. Using corpus statistics and wordnet relations for sense identification. *Computational Linguistics*, 24(1):147–165.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual Conference on Systems Documentation, Toronto, Ontario, Canada*, pages 24–26.

Peter A Lofgren, Siddhartha Banerjee, Ashish Goel, and C Seshadhri. 2014. Fast-ppr: Scaling personalized pagerank estimation for large graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1436–1445. ACM.

Steve L Manion and Raazesh Sainudiin. 2014. An iterative "sudoku style" approach to subgraph-based word sense disambiguation. In *In Proceedings of the Third Joint Conference on Lexical and Computational Se- mantics (*SEM 2014)*, pages 40–50, Dublin, Ireland.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of CONLL*, pages 51–61.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross Bunker. 1993. A semantic concordance. In *Proceedings of the 3rd DARPA Workshop on Human Language Technology*, pages 303–308, Plainsboro, N.J.

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

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics (TACL)*, 2:231–244.

Roberto Navigli. 2009. Word Sense Disambiguation: A survey. *ACM Computing Surveys*, 41(2):1–69.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. Semeval-2013 task 12: Multilingual word sense disambiguation. In *Proceedings of the* 7^{th} *International Workshop on Semantic Evaluation (SemEval 2013), in conjunction with the Second Joint Conference on Lexical and Computational Semantics (*SEM 2013)*, pages 222–231, Atlanta, USA.

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

Arantxa Otegi, Nora Aranberri, Antonio Branco, Jan Hajic, Steven Neale, Petya Osenova, Rita Pereira, Martin Popel, Joao Silva, Kiril Simov, et al. 2016. Qtleap wsd/ned corpora: Semantic annotation of parallel corpora in six languages. In *Proceedings of the 10th Language Resources and Evaluation Conference, LREC*, pages 3023–3030.

Mohammad Taher Pilehvar and Nigel Collier. 2016. De-conflated semantic representations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1680–1690, Austin, TX.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *Proceedings of ACL*, pages 1341–1351.

Mohammad Taher Pilehvar and Roberto Navigli. 2014. A large-scale pseudoword-based evaluation framework for state-of-the-art word sense disambiguation. *Computational Linguistics*, 40(4):837–881.

Sameer S Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. Semeval-2007 task 17: English lexical sample, srl and all words. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 87–92.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017a. Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison. In *Proceedings of EACL*, pages 99–110, Valencia, Spain.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2016. Automatic Construction and Evaluation of a Large Semantically Enriched Wikipedia. In *Proceedings of IJCAI*, pages 2894–2900, New York City, NY, USA.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017b. Neural sequence learning models for word sense disambiguation. In *Proceedings of Empirical Methods in Natural Language Processing*, Copenhagen, Denmark.

Benjamin Snyder and Martha Palmer. 2004. The english all-words task. In *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, Barcelona, Spain.

Kaveh Taghipour and Hwee Tou Ng. 2015. One million sense-tagged instances for word sense disambiguation and induction. *CoNLL 2015*, page 338.

Daniele Vannella, David Jurgens, Daniele Scarfini, Domenico Toscani, and Roberto Navigli. 2014. Validating and extending semantic knowledge bases using video games with a purpose. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 1294–1304, Baltimore, Maryland. Association for Computational Linguistics.

Florentina Vasilescu, Philippe Langlais, and Guy Lapalme. 2004. Evaluating variants of the lesk approach for disambiguating words. In *Proceedings of LREC*, Lisbon, Portugal.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. 2016. Semi-supervised word sense disambiguation with neural models. *Proceedings of COLING*, pages 1374–1385.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83, Uppsala, Sweden. Association for Computational Linguistics.

Micha Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The United Nations parallel corpus v1.0. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portoroz, Slovenia. European Language Resources Association (ELRA).

Universal Semantic Parsing

Siva Reddy[†*] **Oscar Täckström**[‡] **Slav Petrov**[‡] **Mark Steedman**[††] **Mirella Lapata**[††]
[†]Stanford University
[‡] Google Inc.
[††]University of Edinburgh
sivar@stanford.edu, {oscart, slav}@google.com, {steedman, mlap}@inf.ed.ac.uk

Abstract

Universal Dependencies (UD) offer a uniform cross-lingual syntactic representation, with the aim of advancing multilingual applications. Recent work shows that semantic parsing can be accomplished by transforming syntactic dependencies to logical forms. However, this work is limited to English, and cannot process dependency graphs, which allow handling complex phenomena such as control. In this work, we introduce UDEPLAMBDA, a semantic interface for UD, which maps natural language to logical forms in an almost language-independent fashion and can process dependency graphs. We perform experiments on question answering against Freebase and provide German and Spanish translations of the WebQuestions and GraphQuestions datasets to facilitate multilingual evaluation. Results show that UDEPLAMBDA outperforms strong baselines across languages and datasets. For English, it achieves a 4.9 F_1 point improvement over the state-of-the-art on GraphQuestions.

1 Introduction

The Universal Dependencies (UD) initiative seeks to develop cross-linguistically consistent annotation guidelines as well as a large number of uniformly annotated treebanks for many languages (Nivre et al., 2016). Such resources could advance multilingual applications of parsing, improve comparability of evaluation results, enable cross-lingual learning, and more generally support natural language understanding.

Seeking to exploit the benefits of UD for natural language understanding, we introduce UDEPLAMBDA, a semantic interface for UD that maps natural language to logical forms, representing underlying predicate-argument structures, in an almost language-independent manner. Our framework is based on DEPLAMBDA (Reddy et al., 2016) a recently developed method that converts English Stanford Dependencies (SD) to logical forms. The conversion process is illustrated in Figure 1 and discussed in more detail in Section 2. Whereas DEPLAMBDA works only for English, UDEPLAMBDA applies to any language for which UD annotations are available.[1] Moreover, DEPLAMBDA can only process tree-structured inputs whereas UDEPLAMBDA can also process dependency graphs, which allow to handle complex constructions such as control. The different treatments of various linguistic constructions in UD compared to SD also require different handling in UDEPLAMBDA (Section 3.3).

Our experiments focus on Freebase semantic parsing as a testbed for evaluating the framework's multilingual appeal. We convert natural language to logical forms which in turn are converted to machine interpretable formal meaning representations for retrieving answers to questions from Freebase. To facilitate multilingual evaluation, we provide translations of the English WebQuestions (Berant et al., 2013) and GraphQuestions (Su et al., 2016) datasets to German and Spanish. We demonstrate that UDEPLAMBDA can be used to derive logical forms for these languages using a minimal amount of language-specific knowledge. Aside from developing the first multilingual semantic parsing tool for Freebase, we also experimentally show that UDEPLAMBDA outperforms strong baselines across

[*] Work done at the University of Edinburgh

[1] As of v1.3, UD annotations are available for 47 languages at http://universaldependencies.org.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 89–101
Copenhagen, Denmark, September 7–11, 2017.

languages and datasets. For English, it achieves the strongest result to date on GraphQuestions, with competitive results on WebQuestions. Our implementation and translated datasets are publicly available at https://github.com/sivareddyg/udeplambda.

2 DepLambda

Before describing UDepLambda, we provide an overview of DepLambda (Reddy et al., 2016) on which our approach is based. DepLambda converts a dependency tree to its logical form in three steps: *binarization*, *substitution*, and *composition*, each of which is briefly outlined below. Algorithm 1 describes the steps of DepLambda in lines 4-6, whereas lines 2 and 3 are specific to UDepLambda.

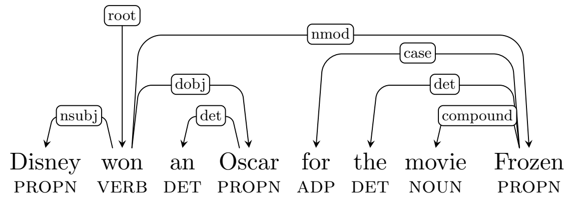
Binarization A dependency tree is first mapped to a Lisp-style s-expression indicating the order of semantic composition. Figure 1(b) shows the s-expression for the sentence *Disney won an Oscar for the movie Frozen*, derived from the dependency tree in Figure 1(a). Here, the sub-expression (dobj won (det Oscar an)) indicates that the logical form of the phrase *won an Oscar* is derived by composing the logical form of the label `dobj` with the logical form of the word *won* and the logical form of the phrase *an Oscar*, derived analogously. The s-expression can also be interpreted as a binarized tree with the dependency label as the root node, and the left and right expressions as subtrees.

A composition hierarchy is employed to impose a strict traversal ordering on the modifiers to each head in the dependency tree. As an example, *won* has three modifiers in Figure 1(a), which according to the composition hierarchy are composed in the order `dobj` > `nmod` > `nsubj`. In constructions like coordination, this ordering is crucial to arrive at the correct semantics. Lines 7-17 in Algorithm 1 describe the binarization step.

Substitution Each symbol in the s-expressions is substituted for a lambda expression encoding its semantics. Words and dependency labels are assigned different types of expressions. In general, words have expressions of the following kind:

ENTITY $\Rightarrow \lambda x.\,\text{word}(x_a)$; e.g. Oscar $\Rightarrow \lambda x.\,\text{Oscar}(x_a)$
EVENT $\Rightarrow \lambda x.\,\text{word}(x_e)$; e.g. won $\Rightarrow \lambda x.\,\text{won}(x_e)$
FUNCTIONAL $\Rightarrow \lambda x.\,\text{TRUE}$; e.g. an $\Rightarrow \lambda x.\,\text{TRUE}$

Here, the subscripts $\cdot_a$ and $\cdot_e$ denote the types of individuals (**Ind**) and events (**Event**), respectively, whereas x denotes a paired variable (x_a, x_e) of type **Ind** × **Event**. Roughly speaking, proper nouns and adjectives invoke ENTITY expressions, verbs and adverbs invoke EVENT expressions, and common nouns invoke both ENTITY and EVENT expressions (see Section 3.3), while remaining words invoke FUNCTIONAL expressions. DepLambda enforces the constraint that every s-expression is of the type $\eta = \textbf{Ind} \times \textbf{Event} \rightarrow \textbf{Bool}$, which simplifies the type system considerably.



(a) The dependency tree for *Disney won an Oscar for the movie Frozen* in the Universal Dependencies formalism.

(nsubj (nmod (dobj won (det Oscar an))
(case (det (comp. Frozen movie) the) for)) Disney)

(b) The binarized s-expression for the dependency tree.

$$\lambda x.\,\exists yzw.\,\text{won}(x_e) \wedge \text{Disney}(y_a) \wedge \text{Oscar}(z_a) \wedge \text{Frozen}(w_a) \wedge \text{movie}(w_a) \wedge \text{arg}_1(x_e, y_a) \wedge \text{arg}_2(x_e, z_a) \wedge \text{nmod.for}(x_e, w_a)$$

(c) The composed lambda-calculus expression.

Figure 1: The mapping of a dependency tree to its logical form with the intermediate s-expression.

Expressions for dependency labels glue the semantics of heads and modifiers to articulate predicate-argument structure. These expressions in general take one of the following forms:

COPY $\Rightarrow \lambda f g x.\,\exists y.\, f(x) \wedge g(y) \wedge \text{rel}(x, y)$
e.g. `nsubj`, `dobj`, `nmod`, `advmod`
INVERT $\Rightarrow \lambda f g x.\,\exists y.\, f(x) \wedge g(y) \wedge \text{rel}^{\text{i}}(y, x)$
e.g. `amod`, `acl`
MERGE $\Rightarrow \lambda f g x.\, f(x) \wedge g(x)$
e.g. `compound`, `appos`, `amod`, `acl`
HEAD $\Rightarrow \lambda f g x.\, f(x)$
e.g. `case`, `punct`, `aux`, `mark` .

As an example of COPY, consider the lambda expression for `dobj` in (dobj won (det Oscar an)): $\lambda f g x.\,\exists y.\, f(x) \wedge g(y) \wedge \text{arg}_2(x_e, y_a)$. This expression takes two functions f and g as input, where f represents the logical form of *won* and g represents the logical form of *an Oscar*. The predicate-argument structure $\text{arg}_2(x_e, y_a)$ indicates that the arg_2 of the event x_e, i.e. *won*, is the individual y_a, i.e. the entity *Oscar*. Since $\text{arg}_2(x_e, y_a)$ mimics the dependency structure dobj(won, Oscar), we refer to the expression kind evoked by `dobj` as COPY.

Expressions that invert the dependency direction are referred to as INVERT (e.g. `amod` in *running horse*); expressions that merge two subexpressions without introducing any relation predicates are referred to as MERGE (e.g. `compound` in *movie Frozen*); and expressions that simply return the parent expression semantics are referred to as HEAD (e.g. `case` in *for Frozen*). While this generalization applies to most dependency labels, several labels take a different logical form not listed here, some of which are discussed in Section 3.3. Sometimes the mapping of dependency label to lambda expression may depend on surrounding part-of-speech tags or dependency labels. For example, `amod` acts as INVERT when the modifier is a verb (e.g. in *running horse*), and as MERGE when the modifier is an adjective (e.g. in *beautiful horse*).[2] Lines 26-32 in Algorithm 1 describe the substitution procedure.

Composition The final logical form is computed by beta-reduction, treating expressions of the form (f x y) as the function f applied to the arguments x and y. For example, (dobj won (det Oscar an)) results in $\lambda x. \exists z. \text{won}(x_e) \wedge \text{Oscar}(z_a) \wedge \text{arg}_2(x_e, z_a)$ when the expression for `dobj` is applied to those for *won* and *(det Oscar an)*. Figure 1(c) shows the logical form for the s-expression in Figure 1(b). The binarized s-expression is recursively converted to a logical form as described in lines 18-25 in Algorithm 1.

3 UDepLambda

We now introduce UDepLambda, a semantic interface for Universal Dependencies.[3] Whereas DepLambda only applies to English Stanford Dependencies, UDepLambda takes advantage of the cross-lingual nature of UD to facilitate an (almost) language independent semantic interface. This is accomplished by restricting the binarization, substitution, and composition steps described above to rely solely on information encoded in the UD representation. As shown in Algorithm 1, lines 4-6 are common to both DepLambda and UDepLambda, whereas lines 2 and 3 applies only to UDepLambda. Importantly, UDepLambda is designed to not rely on lexical forms in a language to assign lambda expressions, but only on information contained in dependency labels and postags.

Algorithm 1: UDepLambda Steps

```
1  Function UDepLambda (depTree):
2      depGraph = Enhancement (depTree)
         #See Figure 2(a) for a depGraph.
3      bindedTree = SplitLongDistance (depGraph)
         #See Figure 2(b) for a bindedTree.
4      binarizedTree = Binarization (bindedTree)
         #See Figure 1(b) for a binarizedTree.
5      logicalForm = Composition (binarizedTree)
6      return logicalForm

7  Function Binarization (tree):
8      parent = GetRootNode (tree);
9      {(label1, child1), (label2, child2)...}
           = GetChildNodes (parent)
10     sortedChildren = SortUsingLabelHierarchy
           ({(label1, child1), (label2, child2)...})
11     binarziedTree.root = parent
12     for label, child ∈ sortedChildren:
13         temp.root = label
14         temp.left = binarziedTree
15         temp.right = Binarization(child)
16         binarziedTree = temp
17     return binarizedTree

18 Function Composition (binarizedTree):
19     mainLF = Substitution (binarizedTree.root)
20     if binarziedTree has left and right children:
21         leftLF = Composition (binarziedTree.left)
22         rightLF = Composition(binarziedTree.right)
23         mainLF = BetaReduce (mainLF, leftLF)
24         mainLF = BetaReduce (mainLF, rightLF)
25     return mainLF

26 Function Substitution (node):
27     logicalForms = [ ]
28     for tregexRule, template ∈ substitutionRules:
29         if tregexRule.match(node):
30             lf = GenLambdaExp (node, template)
31             logicalForms.add(lf)
32     return logicalForms
```

However, some linguistic phenomena are language specific (e.g. pronoun-dropping) or lexicalized (e.g. *every* and *the* in English have different semantics, despite being both determiners) and are not encoded in the UD schema. Furthermore, some cross-linguistic phenomena, such as long-distance dependencies, are not part of the core UD representation. To circumvent this limitation, a simple *enhancement* step enriches the original UD representation before binarization takes place (Section 3.1). This step adds to the dependency tree missing syntactic information and long-distance dependencies, thereby creating a graph. Whereas DepLambda is not able to handle graph-structured input, UDep-

[2] We use Tregex (Levy and Andrew, 2006) for substitution mappings and Cornell SPF (Artzi, 2013) as the lambda-calculus implementation. For example, in *running horse*, the tregex */label:amod/=target < /postag:verb/* matches `amod` to its INVERT expression $\lambda f g x. \exists y. f(x) \wedge g(y) \wedge \text{amod}^{\text{i}}(y_e, x_a)$.

[3] In what follows, all references to UD are to UD v1.3.

LAMBDA is designed to work with dependency graphs as well (Section 3.2). Finally, several constructions differ in structure between UD and SD, which requires different handling in the semantic interface (Section 3.3).

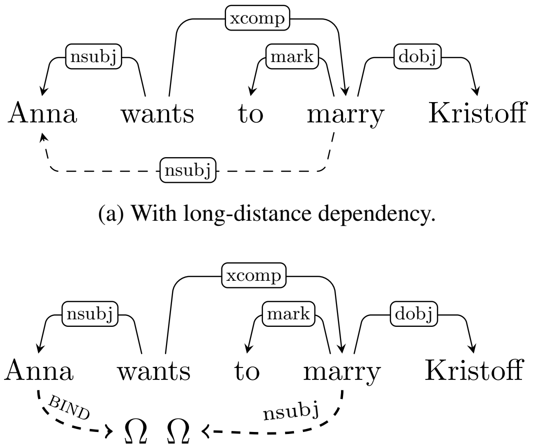
3.1 Enhancement

Both Schuster and Manning (2016) and Nivre et al. (2016) note the necessity of an enhanced UD representation to enable semantic applications. However, such enhancements are currently only available for a subset of languages in UD. Instead, we rely on a small number of enhancements for our main application—semantic parsing for question-answering—with the hope that this step can be replaced by an enhanced UD representation in the future. Specifically, we define three kinds of enhancements: (1) long-distance dependencies; (2) types of coordination; and (3) refined question word tags. These correspond to line 2 in Algorithm 1.

First, we identify long-distance dependencies in relative clauses and control constructions. We follow Schuster and Manning (2016) and find these using the labels `acl` (relative) and `xcomp` (control). Figure 2(a) shows the long-distance dependency in the sentence *Anna wants to marry Kristoff.* Here, *marry* is provided with its missing `nsubj` (dashed arc). Second, UD conflates all coordinating constructions to a single dependency label, `conj`. To obtain the correct coordination scope, we refine `conj` to `conj:verb`, `conj:vp`, `conj:sentence`, `conj:np`, and `conj:adj`, similar to Reddy et al. (2016). Finally, unlike the PTB tags (Marcus et al., 1993) used by SD, the UD part-of-speech tags do not distinguish question words. Since these are crucial to question-answering, we use a small lexicon to refine the tags for determiners (DET), adverbs (ADV) and pronouns (PRON) to DET:WH, ADV:WH and PRON:WH, respectively. Specifically, we use a list of 12 (English), 14 (Spanish) and 35 (German) words, respectively. This is the only part of UDEPLAMBDA that relies on language-specific information. We hope that, as the coverage of morphological features in UD improves, this refinement can be replaced by relying on morphological features, such as the interrogative feature (INT).

3.2 Graph Structures and BIND

To handle graph structures that may result from the enhancement step, such as those in Figure 2(a), we propose a variable-binding mechanism that differs from that of DEPLAMBDA. This is indicated in line 3 of Algorithm 1. First, each long-distance dependency is split into independent arcs as shown in Figure 2(b). Here, Ω is a placeholder for the subject of *marry*, which in turn corresponds to *Anna* as indicated by the binding of Ω via the pseudo-label BIND. We treat BIND like an ordinary dependency label with semantics MERGE and process the resulting tree as usual, via the s-expression:



(a) With long-distance dependency.

(b) With variable binding.

Figure 2: The original and enhanced dependency trees for *Anna wants to marry Kristoff.*

(nsubj (xcomp wants (nsubj (mark
(dobj marry Kristoff) to) Ω) (BIND Anna Ω)) ,

with the lambda-expression substitutions:

wants, *marry* ∈ EVENT; *to* ∈ FUNCTIONAL;
Anna, *Kristoff* ∈ ENTITY;
mark ∈ HEAD; BIND ∈ MERGE;
$\text{xcomp} = \lambda f g x. \exists y. f(x) \wedge g(y) \wedge \text{xcomp}(x_e, y_e)$.

These substitutions are based solely on unlexicalized context. For example, the part-of-speech tag PROPN of *Anna* invokes an ENTITY expression.

The placeholder Ω has semantics $\lambda x.\text{EQ}(x, \omega)$, where $\text{EQ}(u, \omega)$ is true iff u and ω are equal (have the same denotation), which unifies the subject variable of *wants* with the subject variable of *marry*.

After substitution and composition, we get:

$$\lambda z. \exists xywv. \text{wants}(z_e) \wedge \text{Anna}(x_a) \wedge \text{arg}_1(z_e, x_a) \wedge \text{EQ}(x, \omega) \\ \wedge \text{marry}(y_e) \wedge \text{xcomp}(z_e, y_e) \wedge \text{arg}_1(y_e, v_a) \wedge \text{EQ}(v, \omega) \\ \wedge \text{Kristoff}(w_a) \wedge \text{arg}_2(y_e, w_a) \,,$$

This expression may be simplified further by replacing all occurrences of v with x and removing the unification predicates EQ, which results in:

$$\lambda z. \exists xyw. \text{wants}(z_e) \wedge \text{Anna}(x_a) \wedge \text{arg}_1(z_e, x_a) \\ \wedge \text{marry}(y_e) \wedge \text{xcomp}(z_e, y_e) \wedge \text{arg}_1(y_e, x_a) \\ \wedge \text{Kristoff}(w_a) \wedge \text{arg}_2(y_e, w_a) \,.$$

This expression encodes the fact that *Anna* is the arg_1 of the *marry* event, as desired. DEPLAMBDA, in contrast, cannot handle graph-structured input, since it lacks a principled way of generating s-expressions from graphs. Even given the above s-expression, BIND in DEPLAMBDA is defined in a way such that the composition fails to unify v and x, which is crucial for the correct semantics. Moreover, the definition of BIND in DEPLAMBDA does not have a formal interpretation within the lambda calculus, unlike ours.

3.3 Linguistic Constructions

Below, we highlight the most pertinent differences between UDEPLAMBDA and DEPLAMBDA, stemming from the different treatment of various linguistic constructions in UD versus SD.

Prepositional Phrases UD uses a content-head analysis, in contrast to SD, which treats function words as heads of prepositional phrases, Accordingly, the s-expression for the phrase *president in 2009* is (nmod president (case 2009 in)) in UDEPLAMBDA and (prep president (pobj in 2009)) in DEPLAMBDA. To achieve the desired semantics,

$$\lambda x.\, \exists y.\, \text{president}(x_a) \wedge \text{president\_event}(x_e) \wedge \\ \text{arg}_1(x_e, x_a) \wedge \text{2009}(y_a) \wedge \text{prep.in}(x_e, y_a)\,,$$

DEPLAMBDA relies on an intermediate logical form that requires some post-processing, whereas UDEPLAMBDA obtains the desired logical form directly through the following entries:

$$\textit{in} \in \text{FUNCTIONAL};\ \textit{2009} \in \text{ENTITY};\ \text{case} \in \text{HEAD};\\ \textit{president} = \lambda x.\, \text{president}(x_a) \wedge \text{president\_event}(x_e) \\ \wedge\, \text{arg}_1(x_e, x_a)\,;\\ \text{nmod} = \lambda f g x.\, \exists y.\, f(x) \wedge g(y) \wedge \text{nmod.in}(x_e, y_a)\,.$$

Other `nmod` constructions, such as possessives (`nmod:poss`), temporal modifiers (`nmod:tmod`) and adverbial modifiers (`nmod:npmod`), are handled similarly. Note how the common noun *president*, evokes both entity and event predicates above.

Passives DEPLAMBDA gives special treatment to passive verbs, identified by the fine-grained part-of-speech tags in the PTB tag together with dependency context. For example, *An Oscar was won* is analyzed as $\lambda x.\, \text{won.pass}(x_e) \wedge \text{Oscar}(y_a) \wedge \text{arg}_1(x_e, y_a)$, where won.pass represents a passive event. However, UD does not distinguish between active and passive forms.[4] While the labels `nsubjpass` or `auxpass` indicate passive constructions, such clues are sometimes missing, such as in reduced relatives. We therefore opt to not have separate entries for passives, but aim to produce identical logical forms for active and passive forms when possible (for example, by treating `nsubjpass` as direct object). With the following entries,

$$\textit{won} \in \text{EVENT};\ \textit{an, was} \in \text{FUNCTIONAL};\ \text{auxpass} \in \text{HEAD};\\ \text{nsubjpass} = \lambda f g x.\, \exists y.\, f(x) \wedge g(y) \wedge \text{arg}_2(x_e, y_a)\,,$$

the lambda expression for *An Oscar was won* becomes $\lambda x.\, \text{won}(x_e) \wedge \text{Oscar}(y_a) \wedge \text{arg}_2(x_e, y_a)$, identical to that of its active form. However, not having a special entry for passive verbs may have undesirable side-effects. For example, in the reduced-relative construction *Pixar claimed the Oscar won for Frozen*, the phrase *the Oscar won ...* will receive the semantics $\lambda x.\, \text{Oscar}(y_a) \wedge \text{won}(x_e) \wedge \mathbf{arg_1}(x_e, y_a)$, which differs from that of *an Oscar was won*. We leave it to the target application to disambiguate the interpretation in such cases.

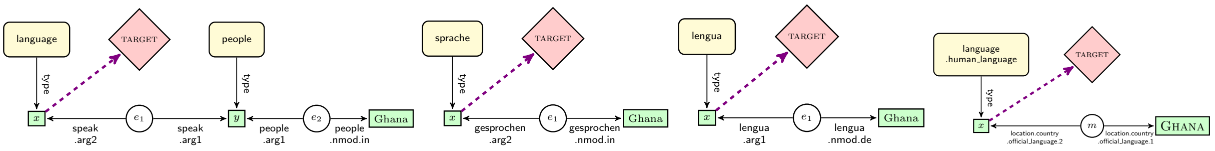
Long-Distance Dependencies As discussed in Section 3.2, we handle long-distance dependencies evoked by clausal modifiers (`acl`) and control verbs (`xcomp`) with the BIND mechanism, whereas DEPLAMBDA cannot handle control constructions. For `xcomp`, as seen earlier, we use the mapping $\lambda f g x.\, \exists y.\, f(x) \wedge g(y) \wedge \text{xcomp}(x_e, y_e)$. For `acl` we use $\lambda f g x.\, \exists y.\, f(x) \wedge g(y)$, to conjoin the main clause and the modifier clause. However, not all `acl` clauses evoke long-distance dependencies, e.g. in *the news that Disney won an Oscar*, the clause *that Disney won an Oscar* is a subordinating conjunction of *news*. In such cases, we instead assign `acl` the INVERT semantics.

Questions Question words are marked with the enhanced part-of-speech tags DET:WH, ADV:WH and PRON:WH, which are all assigned the semantics $\lambda x.\, \$\{\text{word}\}(x_a) \wedge \text{TARGET}(x_a)$. The predicate TARGET indicates that x_a represents the variable of interest, that is the answer to the question.

3.4 Limitations

In order to achieve language independence, UDEPLAMBDA has to sacrifice semantic specificity, since in many cases the semantics is carried by lexical information. Consider the sentences *John broke the window* and *The window broke*. Although it is the *window* that broke in both cases, our inferred logical forms do not canonicalize the relation between *broke* and *window*. To achieve this, we

[4] UD encodes voice as a morphological feature, but most syntactic analyzers do not produce this information yet.



(a) English (b) German (c) Spanish (d) Freebase

Figure 3: The ungrounded graphs for *What language do the people in Ghana speak?*, *Welche Sprache wird in Ghana gesprochen?* and *Cuál es la lengua de Ghana?*, and the corresponding grounded graph.

would have to make the substitution of `nsubj` depend on lexical context, such that when *window* occurs as `nsubj` with *broke*, the predicate arg_2 is invoked rather than arg_1. UDEPLAMBDA does not address this problem, and leave it to the target application to infer context-sensitive semantics of arg_1 and arg_2. To measure the impact of this limitation, we present UDEPLAMBDASRL in Section 4.4 which addresses this problem by relying on semantic roles from semantic role labeling (Palmer et al., 2010).

Other constructions that require lexical information are quantifiers like *every*, *some* and *most*, negation markers like *no* and *not*, and intentional verbs like *believe* and *said*. UD does not have special labels to indicate these. We discuss how to handle quantifiers in this framework in the supplementary material.

Although in the current setup UDEPLAMBDA rules are hand-coded, the number of rules are only proportional to the number of UD labels, making rule-writing manageable.[5] Moreover, we view UDEPLAMBDA as a first step towards learning rules for converting UD to richer semantic representations such as PropBank, AMR, or the Parallel Meaning Bank (Palmer et al., 2005; Banarescu et al., 2013; Abzianidze et al., 2017)..

4 Cross-lingual Semantic Parsing

To study the multilingual nature of UDEPLAMBDA, we conduct an empirical evaluation on question answering against Freebase in three different languages: English, Spanish, and German. Before discussing the details of this experiment, we briefly outline the semantic parsing framework employed.

4.1 Semantic Parsing as Graph Matching

UDEPLAMBDA generates *ungrounded* logical forms that are independent of any knowledge base, such as Freebase. We use GRAPHPARSER (Reddy et al., 2016) to map these logical forms to their grounded Freebase graphs, via corresponding ungrounded graphs. Figures 3(a) to 3(c) show the ungrounded graphs corresponding to logical forms from UDEPLAMBDA, each grounded to the same Freebase graph in Figure 3(d). Here, rectangles denote entities, circles denote events, rounded rectangles denote entity types, and edges between events and entities denote predicates or Freebase relations. Finally, the TARGET node represents the set of values of x that are consistent with the Freebase graph, that is the answer to the question.

GRAPHPARSER treats semantic parsing as a graph-matching problem with the goal of finding the Freebase graphs that are structurally isomorphic to an ungrounded graph and rank them according to a model. To account for structural mismatches, GRAPHPARSER uses two graph transformations: CONTRACT and EXPAND. In Figure 3(a) there are two edges between x and *Ghana*. CONTRACT collapses one of these edges to create a graph isomorphic to Freebase. EXPAND, in contrast, adds edges to connect the graph in the case of disconnected components. The search space is explored by beam search and model parameters are estimated with the averaged structured perceptron (Collins, 2002) from training data consisting of question-answer pairs, using answer F_1-score as the objective.

4.2 Datasets

We evaluate our approach on two public benchmarks of question answering against Freebase: WebQuestions (Berant et al., 2013), a widely used benchmark consisting of English questions and their answers, and GraphQuestions (Su et al., 2016), a recently released dataset of English questions with both their answers and grounded logical forms.

[5] UD v1.3 has 40 dependency labels, and the number of substitution rules in UDEPLAMBDA are 61, with some labels having multiple rules, and some representing lexical semantics.

While WebQuestions is dominated by simple entity-attribute questions, GraphQuestions contains a large number of compositional questions involving aggregation (e.g. *How many children of Eddard Stark were born in Winterfell?*) and comparison (e.g. *In which month does the average rainfall of New York City exceed 86 mm?*). The number of training, development and test questions is 2644, 1134, and 2032, respectively, for WebQuestions and 1794, 764, and 2608 for GraphQuestions.

To support multilingual evaluation, we created translations of WebQuestions and GraphQuestions to German and Spanish. For WebQuestions two professional annotators were hired per language, while for GraphQuestions we used a trusted pool of 20 annotators per language (with a single annotator per question). Examples of the original questions and their translations are provided in Table 1.

	WebQuestions
en	What language do the people in Ghana speak?
de	Welche Sprache wird in Ghana gesprochen?
es	¿Cuál es la lengua de Ghana?
en	Who was Vincent van Gogh inspired by?
de	Von wem wurde Vincent van Gogh inspiriert?
es	¿Qué inspiró a Van Gogh?
	GraphQuestions
en	NASA has how many launch sites?
de	Wie viele Abschussbasen besitzt NASA?
es	¿Cuántos sitios de despegue tiene NASA?
en	Which loudspeakers are heavier than 82.0 kg?
de	Welche Lautsprecher sind schwerer als 82.0 kg?
es	¿Qué altavoces pesan más de 82.0 kg?

Table 1: Example questions and their translations.

k	WebQuestions			GraphQuestions		
	en	de	es	en	de	es
1	89.6	82.8	86.7	47.2	39.9	39.5
10	95.7	91.2	94.0	56.9	48.4	51.6

Table 2: Structured perceptron *k*-best entity linking accuracies on the development sets.

4.3 Implementation Details

Here we provide details on the syntactic analyzers employed, our entity resolution algorithm, and the features used by the grounding model.

Dependency Parsing The English, Spanish, and German Universal Dependencies (UD) treebanks (v1.3; Nivre et al 2016) were used to train part of speech taggers and dependency parsers. We used a bidirectional LSTM tagger (Plank et al., 2016) and a bidirectional LSTM shift-reduce parser (Kiperwasser and Goldberg, 2016). Both the tagger and parser require word embeddings. For English, we used GloVe embeddings (Pennington et al., 2014) trained on Wikipedia and the Gigaword corpus. For German and Spanish, we used SENNA embeddings (Collobert et al., 2011; Al-Rfou et al., 2013) trained on Wikipedia corpora (589M words German; 397M words Spanish).[6] Measured on the UD test sets, the tagger accuracies are 94.5 (English), 92.2 (German), and 95.7 (Spanish), with corresponding labeled attachment parser scores of 81.8, 74.7, and 82.2.

Entity Resolution We follow Reddy et al. (2016) and resolve entities in three steps: (1) potential entity spans are identified using seven handcrafted part-of-speech patterns; (2) each span is associated with potential Freebase entities according to the Freebase/KG API; and (3) the 10-best entity linking lattices, scored by a structured perceptron, are input to GRAPHPARSER, leaving the final disambiguation to the semantic parsing problem. Table 2 shows the 1-best and 10-best entity disambiguation F_1-scores for each language and dataset.

Features We use features similar to Reddy et al. (2016): *basic* features of words and Freebase relations, and *graph* features crossing ungrounded events with grounded relations, ungrounded types with grounded relations, and ungrounded answer type crossed with a binary feature indicating if the answer is a number. In addition, we add features encoding the *semantic* similarity of ungrounded events and Freebase relations. Specifically, we used the cosine similarity of the translation-invariant embeddings of Huang et al. (2015).[7]

4.4 Comparison Systems

We compared UDEPLAMBDA to four versions of GRAPHPARSER that operate on different representations, in addition to prior work.

SINGLEEVENT This model resembles the learning-to-rank model of Bast and Haussmann (2015). An ungrounded graph is generated by connecting all entities in the question with the TARGET node, representing a single event. Note that this

[6] https://sites.google.com/site/rmyeid/projects/polyglot.

[7] http://128.2.220.95/multilingual/data/.

Method	WebQuestions en	de	es	GraphQuestions en	de	es
SINGLEEVENT	48.5	45.6	46.3	15.9	8.8	11.4
DEPTREE	48.8	45.9	46.4	16.0	8.3	11.3
CCGGRAPH	49.5	–	–	15.9	–	–
UDEPLAMBDA	49.5	46.1	47.5	17.7	9.5	12.8
UDEPLAMBDASRL	49.8	46.2	47.0	17.7	9.1	12.7

Table 3: F_1-scores on the test data.

baseline cannot handle compositional questions, or those with aggregation or comparison.

DEPTREE An ungrounded graph is obtained directly from the original dependency tree. An event is created for each parent and its dependents in the tree. Each dependent is linked to this event with an edge labeled with its dependency relation, while the parent is linked to the event with an edge labeled arg_0. If a word is a question word, an additional TARGET predicate is attached to its entity node.

CCGGRAPH This is the CCG-based semantic representation of Reddy et al. (2014). Note that this baseline exists only for English.

UDEPLAMBDASRL This is similar to UDEPLAMBDA except that instead of assuming `nsubj`, `dobj` and `nsubjpass` correspond to arg_1, arg_2 and arg_2, we employ semantic role labeling to identify the correct interpretation. We used the systems of Roth and Woodsend (2014) for English and German and Bjrkelund et al. (2009) for Spanish trained on the CoNLL-2009 dataset (Haji et al., 2009).[8]

4.5 Results

Table 3 shows the performance of GRAPHPARSER with these different representations. Here and in what follows, we use average F_1-score of predicted answers (Berant et al., 2013) as the evaluation metric. We first observe that UDEPLAMBDA consistently outperforms the SINGLEEVENT and DEPTREE representations in all languages.

For English, performance is on par with CCGGRAPH, which suggests that UDEPLAMBDA does not sacrifice too much specificity for universality. With both datasets, results are lower for German compared to Spanish. This agrees with the lower performance of the syntactic parser on the German portion of the UD treebank. While UDEPLAMBDASRL performs better than UDEPLAMBDA on WebQuestions for English, we do not see large performance gaps in other settings, suggesting that GRAPHPARSER is either able to learn context-sensitive semantics of ungrounded predicates or that the datasets do not contain ambiguous `nsubj`, `dobj` and `nsubjpass` mappings. Finally, while these results confirm that GraphQuestions is much harder compared to WebQuestions, we note that both datasets predominantly contain single-hop questions, as indicated by the competitive performance of SINGLEEVENT on both datasets.

Method	GraphQ.	WebQ.
SEMPRE (Berant et al., 2013)	10.8	35.7
JACANA (Yao and Van Durme, 2014)	5.1	33.0
PARASEMPRE (Berant and Liang, 2014)	12.8	39.9
QA (Yao, 2015)	–	44.3
AQQU (Bast and Haussmann, 2015)	–	49.4
AGENDAIL (Berant and Liang, 2015)	–	49.7
DEPLAMBDA (Reddy et al., 2016)	–	50.3
STAGG (Yih et al., 2015)	–	48.4 (52.5)
BILSTM (Türe and Jojic, 2016)	–	24.9 (52.2)
MCNN (Xu et al., 2016)	–	47.0 (53.3)
AGENDAIL-RANK (Yavuz et al., 2016)	–	51.6 (52.6)
UDEPLAMBDA	17.7	49.5

Table 4: F_1-scores on the English GraphQuestions and WebQuestions test sets (results with additional task-specific resources in parentheses).

Table 4 compares UDEPLAMBDA with previously published models which exist only for English and have been mainly evaluated on WebQuestions. These are either symbolic like ours (first block) or employ neural networks (second block). Results for models using additional task-specific training resources, such as ClueWeb09, Wikipedia, or SimpleQuestions (Bordes et al., 2015) are shown in parentheses. On GraphQuestions, we achieve a new state-of-the-art result with a gain of 4.8 F_1-points over the previously reported best result. On WebQuestions we are 2.1 points below the best model using comparable resources, and 3.8 points below the state of the art. Most related to our work is the English-specific system of Reddy et al. (2016). We attribute the 0.8 point difference in F_1-score to their use of the more fine-grained PTB tag set and Stanford Dependencies.

5 Related Work

Our work continues the long tradition of building logical forms from syntactic representations initiated by Montague (1973). The literature is rife with

[8] The parser accuracies (%) are 87.33, 81.38 and 79.91for English, German and Spanish respectively.

attempts to develop semantic interfaces for HPSG (Copestake et al., 2005), LFG (Kaplan and Bresnan, 1982; Dalrymple et al., 1995; Crouch and King, 2006), TAG (Kallmeyer and Joshi, 2003; Gardent and Kallmeyer, 2003; Nesson and Shieber, 2006), and CCG (Baldridge and Kruijff, 2002; Bos et al., 2004; Artzi et al., 2015). Unlike existing semantic interfaces, UDEPLAMBDA uses dependency syntax, a widely available syntactic resource.

A common trend in previous work on semantic interfaces is the reliance on rich typed feature structures or semantic types coupled with strong type constraints, which can be very informative but unavoidably language specific. Instead, UDEPLAMBDA relies on generic unlexicalized information present in dependency treebanks and uses a simple type system (one type for dependency labels, and one for words) along with a combinatory mechanism, which avoids type collisions. Earlier attempts at extracting semantic representations from dependencies have mainly focused on language-specific dependency representations (Spreyer and Frank, 2005; Simov and Osenova, 2011; Hahn and Meurers, 2011; Reddy et al., 2016; Falke et al., 2016; Beltagy, 2016), and multi-layered dependency annotations (Jakob et al., 2010; Bédaride and Gardent, 2011). In contrast, UDEPLAMBDA derives semantic representations for multiple languages in a common schema directly from Universal Dependencies. This work parallels a growing interest in creating other forms of multilingual semantic representations (Akbik et al., 2015; Vanderwende et al., 2015; White et al., 2016; Evang and Bos, 2016).

We evaluate UDEPLAMBDA on semantic parsing for question answering against a knowledge base. Here, the literature offers two main modeling paradigms: (1) learning of task-specific grammars that directly parse language to a grounded representation (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Berant et al., 2013; Flanigan et al., 2014; Pasupat and Liang, 2015; Groschwitz et al., 2015); and (2) converting language to a linguistically motivated task-independent representation that is then mapped to a grounded representation (Kwiatkowski et al., 2013; Reddy et al., 2014; Krishnamurthy and Mitchell, 2015; Gardner and Krishnamurthy, 2017). Our work belongs to the latter paradigm, as we map natural language to Freebase indirectly via logical forms. Capitalizing on natural-language syntax affords interpretability, scalability, and reduced duplication of effort across applications (Bender et al., 2015). Our work also relates to literature on parsing multiple languages to a common executable representation (Cimiano et al., 2013; Haas and Riezler, 2016). However, existing approaches still map to the target meaning representations (more or less) directly (Kwiatkowksi et al., 2010; Jones et al., 2012; Jie and Lu, 2014).

6 Conclusions

We introduced UDEPLAMBDA, a semantic interface for Universal Dependencies, and showed that the resulting semantic representation can be used for question-answering against a knowledge base in multiple languages. We provided translations of benchmark datasets in German and Spanish, in the hope to stimulate further multilingual research on semantic parsing and question answering in general. We have only scratched the surface when it comes to applying UDEPLAMBDA to natural language understanding tasks. In the future, we would like to explore how this framework can benefit applications such as summarization (Liu et al., 2015) and machine reading (Sachan and Xing, 2016).

Acknowledgements

This work greatly benefited from discussions with Michael Collins, Dipanjan Das, Federico Fancellu, Julia Hockenmaier, Tom Kwiatkowski, Adam Lopez, Valeria de Paiva, Martha Palmer, Fernando Pereira, Emily Pitler, Vijay Saraswat, Nathan Schneider, Bonnie Webber, Luke Zettlemoyer, and the members of ILCC Edinburgh University, the Microsoft Research Redmond NLP group, the Stanford NLP group, and the UW NLP and Linguistics group. We thank Reviewer 2 for useful feedback. The authors would also like to thank the Universal Dependencies community for the treebanks and documentation. This research is supported by a Google PhD Fellowship to the first author. We acknowledge the financial support of the European Research Council (Lapata; award number 681760).

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The Parallel Meaning Bank: Towards a Multilingual Corpus of Translations Annotated with Compositional Meaning Representations. In *Proceedings of the*

European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, Valencia, Spain, pages 242–247.

Alan Akbik, laura chiticariu, Marina Danilevsky, Yunyao Li, Shivakumar Vaithyanathan, and Huaiyu Zhu. 2015. Generating High Quality Proposition Banks for Multilingual Semantic Role Labeling. In *Proceedings of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Beijing, China, pages 397–407.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed Word Representations for Multilingual NLP. In *Proceedings of the Computational Natural Language Learning*. Sofia, Bulgaria, pages 183–192.

Yoav Artzi. 2013. Cornell SPF: Cornell Semantic Parsing Framework. *arXiv:1311.3011 [cs.CL]* .

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG Semantic Parsing with AMR. In *Proceedings of the Empirical Methods on Natural Language Processing*. pages 1699–1710.

Jason Baldridge and Geert-Jan Kruijff. 2002. Coupling CCG and Hybrid Logic Dependency Semantics. In *Proceedings of the Association for Computational Linguistics*. pages 319–326.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Linguistic Annotation Workshop and Interoperability with Discourse*. Sofia, Bulgaria, pages 178–186.

Hannah Bast and Elmar Haussmann. 2015. More Accurate Question Answering on Freebase. In *Proceedings of ACM International Conference on Information and Knowledge Management*. pages 1431–1440.

Paul Bédaride and Claire Gardent. 2011. Deep Semantics for Dependency Structures. In *Proceedings of Conference on Intelligent Text Processing and Computational Linguistics*. pages 277–288.

Islam Beltagy. 2016. *Natural Language Semantics Using Probabilistic Logic*. Ph.D. thesis, Department of Computer Science, The University of Texas at Austin.

Emily M. Bender, Dan Flickinger, Stephan Oepen, Woodley Packard, and Ann Copestake. 2015. Layers of Interpretation: On Grammar and Compositionality. In *Proceedings of the International Conference on Computational Semantics*. Association for Computational Linguistics, London, UK, pages 239–249.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the Empirical Methods on Natural Language Processing*. pages 1533–1544.

Jonathan Berant and Percy Liang. 2014. Semantic Parsing via Paraphrasing. In *Proceedings of the Association for Computational Linguistics*. pages 1415–1425.

Jonathan Berant and Percy Liang. 2015. Imitation Learning of Agenda-Based Semantic Parsers. *Transactions of the Association for Computational Linguistics* 3:545–558.

Anders Bjrkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual Semantic Role Labeling. In *Proceedings of Computational Natural Language Learning (CoNLL 2009): Shared Task*. Association for Computational Linguistics, Boulder, Colorado, pages 43–48.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. CoRR abs/1506.02075.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-Coverage Semantic Representations from a CCG Parser. In *Proceedings of the Conference on Computational Linguistics*. pages 1240–1246.

Philipp Cimiano, Vanessa Lopez, Christina Unger, Elena Cabrio, Axel-Cyrille Ngonga Ngomo, and Sebastian Walter. 2013. Multilingual question answering over linked data (QALD-3): Lab overview. In *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*. Springer, Valencia, Spain, volume 8138.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the Empirical Methods on Natural Language Processing*. pages 1–8.

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

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal Recursion Semantics: An Introduction. *Research on Language and Computation* 3(2-3):281–332.

Dick Crouch and Tracy Holloway King. 2006. Semantics via f-structure rewriting. In *Proceedings of the LFG'06 Conference*. CSLI Publications, page 145.

Mary Dalrymple, John Lamping, Fernando C. N. Pereira, and Vijay A. Saraswat. 1995. Linear Logic for Meaning Assembly. In *Proceedings of Computational Logic for Natural Language Processing*.

Kilian Evang and Johan Bos. 2016. Cross-lingual Learning of an Open-domain Semantic Parser. In *Proceedings of the Conference on Computational Linguistics*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 579–588.

Tobias Falke, Gabriel Stanovsky, Iryna Gurevych, and Ido Dagan. 2016. Porting an Open Information Extraction System from English to German. In *Proceedings of the Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 892–898.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In *Proceedings of the Association for Computational Linguistics*. pages 1426–1436.

Claire Gardent and Laura Kallmeyer. 2003. Semantic Construction in Feature-based TAG. In *Proceedings of European Chapter of the Association for Computational Linguistics*. pages 123–130.

Matt Gardner and Jayant Krishnamurthy. 2017. Open-Vocabulary Semantic Parsing with both Distributional Statistics and Formal Knowledge. In *Proceedings of Association for the Advancement of Artificial Intelligence*.

Jonas Groschwitz, Alexander Koller, and Christoph Teichmann. 2015. Graph parsing with s-graph grammars. In *Proceedings of the Association for Computational Linguistics*. pages 1481–1490.

Carolin Haas and Stefan Riezler. 2016. A Corpus and Semantic Parser for Multilingual Natural Language Querying of OpenStreetMap. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 740–750.

Michael Hahn and Detmar Meurers. 2011. On deriving semantic representations from dependencies: A practical approach for evaluating meaning in learner corpora. In *Proceedings of the Int. Conference on Dependency Linguistics (Depling 2011)*. Barcelona, pages 94–103.

Jan Haji, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antnia Mart, Llus Mrquez, Adam Meyers, Joakim Nivre, Sebastian Pad, Jan tpnek, and others. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, pages 1–18.

Kejun Huang, Matt Gardner, Evangelos Papalexakis, Christos Faloutsos, Nikos Sidiropoulos, Tom Mitchell, Partha P. Talukdar, and Xiao Fu. 2015. Translation Invariant Word Embeddings. In *Proceedings of the Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 1084–1088.

Max Jakob, Markéta Lopatková, and Valia Kordoni. 2010. Mapping between Dependency Structures and Compositional Semantic Representations. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*.

Zhanming Jie and Wei Lu. 2014. Multilingual Semantic Parsing : Parsing Multiple Languages into Semantic Representations. In *Proceedings of the Conference on Computational Linguistics*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 1291–1301.

Bevan Keeley Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic Parsing with Bayesian Tree Transducers. In *Proceedings of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 488–496.

Laura Kallmeyer and Aravind Joshi. 2003. Factoring predicate argument and scope semantics: Underspecified semantics with LTAG. *Research on Language and Computation* 1(1-2):3–58.

Ronald M Kaplan and Joan Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representation. *Formal Issues in Lexical-Functional Grammar* pages 29–130.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics* 4:313–327.

Jayant Krishnamurthy and Tom M. Mitchell. 2015. Learning a Compositional Semantics for Freebase with an Open Predicate Vocabulary. *Transactions of the Association for Computational Linguistics* 3:257–270.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing Probabilistic CCG Grammars from Logical Form with Higher-Order Unification. In *Proceedings of the Empirical Methods on Natural Language Processing*. pages 1223–1233.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling Semantic Parsers with On-the-Fly Ontology Matching. In *Proceedings of the Empirical Methods on Natural Language Processing*. pages 1545–1556.

Roger Levy and Galen Andrew. 2006. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of LREC*. pages 2231–2234.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward Abstractive Summarization Using Semantic Representations. In *Proceedings of North American Chapter of the Association for Computational Linguistics*. pages 1077–1086.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics* 19(2):313–330.

Richard Montague. 1973. The Proper Treatment of Quantification in Ordinary English. In K.J.J. Hintikka, J.M.E. Moravcsik, and P. Suppes, editors, *Approaches to Natural Language*, Springer Netherlands, volume 49 of *Synthese Library*, pages 221–242.

Rebecca Nesson and Stuart M. Shieber. 2006. Simpler TAG Semantics Through Synchronization. In *Proceedings of the 11th Conference on Formal Grammar*. Center for the Study of Language and Information, Malaga, Spain, pages 129–142.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*. European Language Resources Association (ELRA), Paris, France.

Joakim Nivre et al. 2016. Universal dependencies 1.3. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University in Prague.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics* 31(1):71–106.

Martha Palmer, Daniel Gildea, and Nianwen Xue. 2010. Semantic role labeling. *Synthesis Lectures on Human Language Technologies* 3(1):1–103.

Panupong Pasupat and Percy Liang. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *Proceedings of the Association for Computational Linguistics*. pages 1470–1480.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, pages 412–418.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale Semantic Parsing without Question-Answer Pairs. *Transactions of the Association for Computational Linguistics* 2:377–392.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics* 4:127–140.

Michael Roth and Kristian Woodsend. 2014. Composition of Word Representations Improves Semantic Role Labelling. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 407–413.

Mrinmaya Sachan and Eric Xing. 2016. Machine Comprehension using Rich Semantic Representations. In *Proceedings of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, pages 486–492.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An Improved Representation for Natural Language Understanding Tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*. European Language Resources Association (ELRA), Paris, France.

Kiril Simov and Petya Osenova. 2011. Towards Minimal Recursion Semantics over Bulgarian Dependency Parsing. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*. RANLP 2011 Organising Committee, Hissar, Bulgaria, pages 471–478.

Kathrin Spreyer and Anette Frank. 2005. Projecting RMRS from TIGER Dependencies. In *Proceedings of the HPSG 2005 Conference*. CSLI Publications.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On Generating Characteristic-rich Question Sets for QA Evaluation. In *Proceedings of the Empirical Methods in Natural Language Processing*. Austin, Texas, pages 562–572.

Ferhan Türe and Oliver Jojic. 2016. Simple and Effective Question Answering with Recurrent Neural Networks. *CoRR* abs/1606.05029.

Lucy Vanderwende, Arul Menezes, and Chris Quirk. 2015. An AMR parser for English, French, German, Spanish and Japanese and a new AMR-annotated corpus. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. Association for Computational Linguistics, Denver, Colorado, pages 26–30.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal Decompositional Semantics on Universal Dependencies. In *Proceedings of the Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 1713–1723.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question Answering on Freebase via Relation Extraction and Textual Evidence. In *Proceedings of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, pages 2326–2336.

Xuchen Yao. 2015. Lean Question Answering over Freebase from Scratch. In *Proceedings of North American Chapter of the Association for Computational Linguistics*. pages 66–70.

Xuchen Yao and Benjamin Van Durme. 2014. Information Extraction over Structured Data: Question Answering with Freebase. In *Proceedings of the Association for Computational Linguistics*. pages 956–966.

Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. 2016. Improving Semantic Parsing via Answer Type Inference. In *Proceedings of the Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 149–159.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *Proceedings of the Association for Computational Linguistics*. pages 1321–1331.

John M. Zelle and Raymond J. Mooney. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of Association for the Advancement of Artificial Intelligence*. pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *Proceedings of Uncertainty in Artificial Intelligence*. pages 658–666.

Mimicking Word Embeddings using Subword RNNs

Yuval Pinter **Robert Guthrie** **Jacob Eisenstein**

School of Interactive Computing
Georgia Institute of Technology
{uvp,rguthrie3,jacobe}@gatech.edu

Abstract

Word embeddings improve generalization over lexical features by placing each word in a lower-dimensional space, using distributional information obtained from unlabeled data. However, the effectiveness of word embeddings for downstream NLP tasks is limited by out-of-vocabulary (OOV) words, for which embeddings do not exist. In this paper, we present MIMICK, an approach to generating OOV word embeddings compositionally, by learning a function from spellings to distributional embeddings. Unlike prior work, MIMICK does not require re-training on the original word embedding corpus; instead, learning is performed at the type level. Intrinsic and extrinsic evaluations demonstrate the power of this simple approach. On 23 languages, MIMICK improves performance over a word-based baseline for tagging part-of-speech and morphosyntactic attributes. It is competitive with (and complementary to) a supervised character-based model in low-resource settings.

1 Introduction

One of the key advantages of word embeddings for natural language processing is that they enable generalization to words that are unseen in labeled training data, by embedding lexical features from large unlabeled datasets into a relatively low-dimensional Euclidean space. These low-dimensional embeddings are typically trained to capture distributional similarity, so that information can be shared among words that tend to appear in similar contexts.

However, it is not possible to enumerate the entire vocabulary of any language, and even large unlabeled datasets will miss terms that appear in later applications. The issue of how to handle these *out-of-vocabulary* (OOV) words poses challenges for embedding-based methods. These challenges are particularly acute when working with low-resource languages, where even unlabeled data may be difficult to obtain at scale. A typical solution is to abandon hope, by assigning a single OOV embedding to all terms that do not appear in the unlabeled data.

We approach this challenge from a quasi-generative perspective. Knowing nothing of a word except for its embedding and its written form, we attempt to learn the former from the latter. We train a recurrent neural network (RNN) on the character level with the embedding as the target, and use it later to predict vectors for OOV words in any downstream task. We call this model the MIMICK-RNN, for its ability to read a word's spelling and mimick its distributional embedding.

Through nearest-neighbor analysis, we show that vectors learned via this method capture both word-shape features and lexical features. As a result, we obtain reasonable near-neighbors for OOV abbreviations, names, novel compounds, and orthographic errors. Quantitative evaluation on the Stanford RareWord dataset (Luong et al., 2013) provides more evidence that these character-based embeddings capture word similarity for rare and unseen words.

As an extrinsic evaluation, we conduct experiments on joint prediction of part-of-speech tags and morphosyntactic attributes for a diverse set of 23 languages, as provided in the Universal Dependencies dataset (De Marneffe et al., 2014). Our model shows significant improvement

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 102–112
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

across the board against a single *UNK*-embedding backoff method, and obtains competitive results against a supervised character-embedding model, which is trained end-to-end on the target task. In low-resource settings, our approach is particularly effective, and is complementary to supervised character embeddings trained from labeled data. The MIMICK-RNN therefore provides a useful new tool for tagging tasks in settings where there is limited labeled data. Models and code are available at `www.github.com/yuvalpinter/mimick`.

2 Related Work

Compositional models for embedding rare and unseen words. Several studies make use of morphological or orthographic information when training word embeddings, enabling the prediction of embeddings for unseen words based on their internal structure. Botha and Blunsom (2014) compute word embeddings by summing over embeddings of the morphemes; Luong et al. (2013) construct a recursive neural network over each word's morphological parse; Bhatia et al. (2016) use morpheme embeddings as a prior distribution over probabilistic word embeddings. While morphology-based approaches make use of meaningful linguistic substructures, they struggle with names and foreign language words, which include out-of-vocabulary morphemes. Character-based approaches avoid these problems: for example, Kim et al. (2016) train a recurrent neural network over words, whose embeddings are constructed by convolution over character embeddings; Wieting et al. (2016) learn embeddings of character n-grams, and then sum them into word embeddings. In all of these cases, the model for composing embeddings of subword units into word embeddings is learned by optimizing an objective over a large unlabeled corpus. In contrast, our approach is a post-processing step that can be applied to any set of word embeddings, regardless of how they were trained. This is similar to the "retrofitting" approach of Faruqui et al. (2015), but rather than smoothing embeddings over a graph, we learn a function to build embeddings compositionally.

Supervised subword models. Another class of methods learn task-specific character-based word embeddings within end-to-end supervised systems. For example, Santos and Zadrozny (2014) build word embeddings by convolution over characters, and then perform part-of-speech (POS) tagging using a local classifier; the tagging objective drives the entire learning process. Ling et al. (2015) propose a multi-level long short-term memory (LSTM; Hochreiter and Schmidhuber, 1997), in which word embeddings are built compositionally from an LSTM over characters, and then tagging is performed by an LSTM over words. Plank et al. (2016) show that concatenating a character-level or bit-level LSTM network to a word representation helps immensely in POS tagging. Because these methods learn from labeled data, they can cover only as much of the lexicon as appears in their labeled training sets. As we show, they struggle in several settings: low-resource languages, where labeled training data is scarce; morphologically rich languages, where the number of morphemes is large, or where the mapping from form to meaning is complex; and in Chinese, where the number of characters is orders of magnitude larger than in non-logographic scripts. Furthermore, supervised subword models can be combined with MIMICK, offering additive improvements.

Morphosyntactic attribute tagging. We evaluate our method on the task of tagging word tokens for their morphosyntactic attributes, such as gender, number, case, and tense. The task of morpho-syntactic tagging dates back at least to the mid 1990s (Oflazer and Kuruöz, 1994; Hajič and Hladká, 1998), and interest has been rejuvenated by the availability of large-scale multilingual morphosyntactic annotations through the Universal Dependencies (UD) corpus (De Marneffe et al., 2014). For example, Faruqui et al. (2016) propose a graph-based technique for propagating type-level morphological information across a lexicon, improving token-level morphosyntactic tagging in 11 languages, using an SVM tagger. In contrast, we apply a neural sequence labeling approach, inspired by the POS tagger of Plank et al. (2016).

3 MIMICK Word Embeddings

We approach the problem of out-of-vocabulary (OOV) embeddings as a **generation** problem: regardless of how the original embeddings were created, we assume there is a generative wordform-based protocol for creating these embeddings. By training a model over the existing vocabulary, we can later use that model for predicting the embedding of an unseen word.

Formally: given a language $\mathcal{L}$, a vocabulary $\mathcal{V} \subseteq \mathcal{L}$ of size V, and a pre-trained embeddings table $\mathcal{W} \in \mathbb{R}^{V \times d}$ where each word $\{w_k\}_{k=1}^{V}$ is assigned a vector $\boldsymbol{e}_k$ of dimension d, our model is trained to find the function $f : \mathcal{L} \rightarrow \mathbb{R}^d$ such that the projected function $f|_{\mathcal{V}}$ approximates the assignments $f(w_k) \approx \boldsymbol{e}_k$. Given such a model, a new word $w_{k^*} \in \mathcal{L} \setminus \mathcal{V}$ can now be assigned an embedding $\boldsymbol{e}_{k^*} = f(w_{k^*})$.

Our predictive function of choice is a **Word Type Character Bi-LSTM**. Given a word with character sequence $w = \{c_i\}_1^n$, a forward-LSTM and a backward-LSTM are run over the corresponding character embeddings sequence $\{\boldsymbol{e}_i^{(c)}\}_1^n$. Let $\boldsymbol{h}_f^n$ represent the final hidden vector for the forward-LSTM, and let $\boldsymbol{h}_b^0$ represent the final hidden vector for the backward-LSTM. The word embedding is computed by a multilayer perceptron:

$$f(w) = \mathbf{O}_T \cdot g(\mathbf{T}_h \cdot [\boldsymbol{h}_f^n; \boldsymbol{h}_b^0] + \boldsymbol{b}_h) + \boldsymbol{b}_T, \quad (1)$$

where $\mathbf{T}_h, \boldsymbol{b}_h$ and $\mathbf{O}_T, \boldsymbol{b}_T$ are parameters of affine transformations, and g is a nonlinear elementwise function. The model is presented in Figure 1.

The training objective is similar to that of Yin and Schütze (2016). We match the predicted embeddings $f(w_k)$ to the pre-trained word embeddings e_{w_k}, by minimizing the squared Euclidean distance,

$$\mathcal{L} = \|f(w_k) - \boldsymbol{e}_{w_k}\|_2^2 . \quad (2)$$

By backpropagating from this loss, it is possible to obtain local gradients with respect to the parameters of the LSTMs, the character embeddings, and the output model. The ultimate output of the training phase is the character embeddings matrix $\mathbf{C}$ and the parameters of the neural network: $\mathcal{M} = \{\mathbf{C}, \mathbf{F}, \mathbf{B}, \mathbf{T}_h, \boldsymbol{b}_h, \mathbf{O}_T, \boldsymbol{b}_T\}$, where $\mathbf{F}, \mathbf{B}$ are the forward and backward LSTM component parameters, respectively.

3.1 MIMICK Polyglot Embeddings

The pretrained embeddings we use in our experiments are obtained from Polyglot (Al-Rfou et al., 2013), a multilingual word embedding effort. Available for dozens of languages, each dataset contains 64-dimension embeddings for the 100,000 most frequent words in a language's training corpus (of variable size), as well as an *UNK* embedding to be used for OOV words. Even with this vocabulary size, querying words from respective UD corpora (train + dev + test) yields high OOV rates: in at least half of the 23 languages in our experiments (see Section 5), 29.1% or more of the word types do not appear in the Polyglot vocabulary. The token-level median rate is 9.2%.[1]

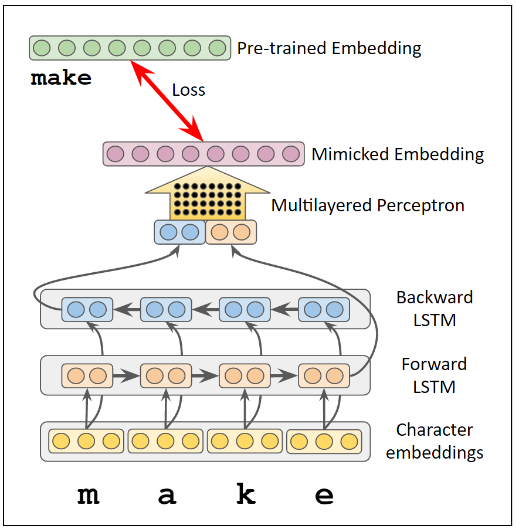


Figure 1: MIMICK model architecture.

Applying our MIMICK algorithm to Polyglot embeddings, we obtain a prediction model for each of the 23 languages. Based on preliminary testing on randomly selected held-out development sets of 1% from each Polyglot vocabulary (with error calculated as in Equation 2), we set the following hyper-parameters for the remainder of the experiments: character embedding dimension = 20; one LSTM layer with 50 hidden units; 60 training epochs with no dropout; nonlinearity function $g = \tanh$.[2] We initialize character embeddings randomly, and use DyNet to implement the model (Neubig et al., 2017).

Nearest-neighbor examination. As a preliminary sanity check for the validity of our protocol, we examined nearest-neighbor samples in languages for which speakers were available: English, Hebrew, Tamil, and Spanish. Table 1 presents selected English OOV words with

[1] Some OOV counts, and resulting model performance, may be adversely affected by tokenization differences between Polyglot and UD. Notably, some languages such as Spanish, Hebrew and Italian exhibit **relational synthesis** wherein words of separate grammatical phrases are joined into one form (e.g. Spanish *del* = *de* + *el*, 'from the-masc.-sg.'). For these languages, the UD annotations adhere to the sub-token level, while Polyglot does not perform sub-tokenization. As this is a real-world difficulty facing users of out-of-the-box embeddings, we do not patch it over in our implementations or evaluation.

[2] Other settings, described below, were tuned on the supervised downstream tasks.

OOV word	Nearest neighbors	OOV word	Nearest neighbors
MCT	AWS OTA APT PDM SMP	compartmentalize	formalize rationalize discern prioritize validate
McNeally	Howlett Gaughan McCallum Blaney	pesky	euphoric disagreeable horrid ghastly horrifying
Vercellotti	Martinelli Marini Sabatini Antonelli	lawnmower	tradesman bookmaker postman hairdresser
Secretive	Routine Niche Turnaround Themed	developiong	compromising inflating shrinking straining
corssing	slicing swaying pounding grasping	hurtling	splashing pounding swaying slicing rubbing
flatfish	slimy jerky watery glassy wrinkle	expectedly	legitimately profoundly strangely energetically

Table 1: Nearest-neighbor examples for the English MIMICK model.

their nearest in-vocabulary Polyglot words computed by cosine similarity. These examples demonstrate several properties: (a) word shape is learned well (acronyms, capitalizations, suffixes); (b) the model shows robustness to typos (e.g., *develop**iong***, *c**or**ssing*); (c) part-of-speech is learned across multiple suffixes (*pesky – euphoric, ghastly*); (d) word compounding is detected (e.g., *lawnmower – bookmaker, postman*); (e) semantics are not learned well (as is to be expected from the lack of context in training), but there are surprises (e.g., *flatfish – slimy, watery*). Table 2 presents examples from Hebrew that show learned properties can be extended to nominal morphosyntactic attributes (gender, number – first two examples) and even relational syntactic subword forms such as genetive markers (third example). Names are learned (fourth example) despite the lack of casing in the script. Spanish examples exhibit word-shape and part-of-speech learning patterns with some loose semantics: for example, the plural adjective form *prenatales* is similar to other family-related plural adjectives such as *patrimoniales* and *generacionales*. Tamil displays some semantic similarities as well: e.g. *enjineer* ('engineer') predicts similarity to other professional terms such as *kalviyiyal* ('education'), *thozhilnutpa* ('technical'), and *iraanuva* ('military').

Stanford RareWords. The Stanford RareWord evaluation corpus (Luong et al., 2013) focuses on predicting word similarity between pairs involving low-frequency English words, predominantly ones with common morphological affixes. As these words are unlikely to be above the cutoff threshold for standard word embedding models, they emphasize the performance on OOV words.

For evaluation of our MIMICK model on the RareWord corpus, we trained the Variational Embeddings algorithm (VarEmbed; Bhatia et al., 2016) on a 20-million-token, 100,000-type Wikipedia corpus, obtaining 128-dimension word embeddings for all words in the test corpus. VarEmbed estimates a prior distribution over word embeddings, conditional on the morphological composition. For in-vocabulary words, a posterior is estimated from unlabeled data; for out-of-vocabulary words, the expected embedding can be obtained from the prior alone. In addition, we compare to FastText (Bojanowski et al., 2016), a high-vocabulary, high-dimensionality embedding benchmark.

The results, shown in Table 3, demonstrate that the MIMICK RNN recovers about half of the loss in performance incurred by the original Polyglot training model due to out-of-vocabulary words in the "All pairs" condition. MIMICK also outperforms VarEmbed. FastText can be considered an upper bound: with a vocabulary that is 25 times larger than the other models, it was missing words from only 44 pairs on this data.

4 Joint Tagging of Parts-of-Speech and Morphosyntactic Attributes

The Universal Dependencies (UD) scheme (De Marneffe et al., 2014) features a minimal set of 17 POS tags (Petrov et al., 2012) and supports tagging further language-specific features using attribute-specific inventories. For example, a verb in Turkish could be assigned a value for the evidentiality attribute, one which is absent from Danish. These additional morphosyntactic attributes are marked in the UD dataset as optional per-token attribute-value pairs.

Our approach for tagging morphosyntactic attributes is similar to the part-of-speech tagging model of Ling et al. (2015), who attach a projection layer to the output of a sentence-level bidirectional LSTM. We extend this approach to morphosyntactic tagging by duplicating this projection layer for each attribute type. The input to our multilayer perceptron (MLP) projection network is the hidden state produced for each token in the sentence by an underlying LSTM, and the output is

OOV word	Nearest neighbors
TTGFM '(s/y) will come true',	TPTVR '(s/y) will solve', TBTL '(s/y) will cancel', TSIR '(s/y) will remove'
GIAVMTRIIM 'geometric(m-pl)'$_2$	ANTVMIIM 'anatomic(m-pl)', GAVMTRIIM 'geometric(m-pl)'$_1$
BQFTNV 'our request'	IVFBIHM 'their(m) residents', XTAIHM 'their(m) sins', IRVFTV 'his inheritance'
RIC'RDSVN 'Richardson'	AVISTRK 'Eustrach', QMINQA 'Kaminka', GVLDNBRG 'Goldenberg'

Table 2: Nearest-neighbor examples for Hebrew (Transcriptions per Sima'an et al. (2001)). 's/y' stands for 'she/you-m.sg.'; subscripts denote alternative spellings, standard form being 'X'$_1$.

	Emb. dim	Vocab size	Polyglot in-vocab $N = 862$	All pairs $N = 2034$
VarEmbed	128	100K	41.9	25.5
Polyglot	64	100K	40.8	8.7
MIMICK	64	0	17.9	17.5
Polyglot +MIMICK	64	100K	40.8	27.0
Fasttext	300	2.51M		47.3

Table 3: Similarity results on the RareWord set, measured as Spearman's $\rho \times 100$. VarEmbed was trained on a 20-million token dataset, Polyglot on a 1.7B-token dataset.

attribute-specific probability distributions over the possible values for each attribute on each token in the sequence. Formally, for a given attribute a with possible values $v \in V_a$, the tagging probability for the i'th word in a sentence is given by:

$$\Pr(a_{w_i} = v) = (\text{Softmax}(\phi(\boldsymbol{h}_i)))_v\ , \qquad (3)$$

with

$$\phi(\boldsymbol{h}_i) = \mathbf{O}_W^a \cdot \tanh(\mathbf{W}_h^a \cdot \boldsymbol{h}_i + \boldsymbol{b}_h^a) + \boldsymbol{b}_W^a, \qquad (4)$$

where $\boldsymbol{h}_i$ is the i'th hidden state in the underlying LSTM, and $\phi(\boldsymbol{h}_i)$ is a two-layer feedforward neural network, with weights $\mathbf{W}_h^a$ and $\mathbf{O}_W^a$. We apply a softmax transformation to the output; the value at position v is then equal to the probability of attribute v applying to token w_i. The input to the underlying LSTM is a sequence of word embeddings, which are initialized to the Polyglot vectors when possible, and to MIMICK vectors when necessary. Alternative initializations are considered in the evaluation, as described in Section 5.2.

Each tagged attribute sequence (including POS tags) produces a loss equal to the sum of negative log probabilities of the true tags. One way to combine these losses is to simply compute the **sum loss**. However, many languages have large differences in sparsity across morpho-syntactic attributes, as apparent from Table 4 (rightmost column). We therefore also compute a **weighted sum loss**, in which each attribute is weighted by the proportion of training corpus tokens on which it is assigned a non-*NONE* value. Preliminary experiments on development set data were inconclusive across languages and training set sizes, and so we kept the simpler sum loss objective for the remainder of our study. In all cases, part-of-speech tagging was less accurate when learned jointly with morphosyntactic attributes. This may be because the attribute loss acts as POS-unrelated "noise" affecting the common LSTM layer and the word embeddings.

5 Experimental Settings

The morphological complexity and compositionality of words varies greatly across languages. While a morphologically-rich agglutinative language such as Hungarian contains words that carry many attributes as fully separable morphemes, a sentence in an analytic language such as Vietnamese may have not a single polymorphemic or inflected word in it. To see whether this property is influential on our MIMICK model and its performance in the downstream tagging task, we select languages that comprise a sample of multiple morphological patterns. Language family and script type are other potentially influential factors in an orthography-based approach such as ours, and so we vary along these parameters as well. We also considered language selection recommendations from de Lhoneux and Nivre (2016) and Schluter and Agić (2017).

As stated above, our approach is built on the Polyglot word embeddings. The intersection of the Polyglot embeddings and the UD dataset (version 1.4) yields 44 languages. Of these, many are under-annotated for morphosyntactic attributes; we select twenty-three sufficiently-tagged languages, with the exception of Indonesian.[3] Table 4 presents the selected languages and their typological properties. As an additional proxy for mor-

[3] Vietnamese has no attributes by design; it is a pure analytic language.

	Language	Branch	Script type	Morpho.	Tokens w/ attr.
vi	Vietnamese	Vietic	alphabetic*	Analytic	00.0%
hu	Hungarian	Finno-Ugric	alphabetic	Agglutin.	83.6%
id	Indonesian	Malayic	alphabetic	Agglutin.	—
zh	Chinese	Sinitic	ideographic	Isolating	06.2%
tr	Turkish	Turkic	alphabetic	Agglutin.	68.4%
kk	Kazakh	Turkic	alphabetic	Agglutin.	20.9%
ar	Arabic	Semitic	consonantal	Fusional	60.6%
he	Hebrew	Semitic	consonantal	Fusional	62.9%
eu	Basque	Vasconic	alphabetic	Agglutin.	59.2%
ta	Tamil	Tamil	syllabic	Agglutin.	78.8%
fa	Persian	Iranian	consonantal	Agglutin.	65.4%
hi	Hindi	Indo-Aryan	alphasyllab.	Fusional	92.4%
lv	Latvian	Baltic	alphabetic	Fusional	69.2%
el	Greek	Hellenic	alphabetic	Fusional	64.8%
bg	Bulgarian	Slavic	alphabetic	Fusional	68.6%
ru	Russian	Slavic	alphabetic	Fusional	69.2%
cs	Czech	Slavic	alphabetic	Fusional	83.2%
es	Spanish	Romance	alphabetic	Fusional	67.1%
it	Italian	Romance	alphabetic	Fusional	67.3%
ro	Romanian	Romance	alphabetic	Fusional	87.1%
da	Danish	Germanic	alphabetic	Fusional	72.2%
en	English	Germanic	alphabetic	Analytic	72.8%
sv	Swedish	Germanic	alphabetic	Analytic	73.4%

Table 4: Languages used in tagging evaluation. Languages on the right are Indo-European. *In Vietnamese script, whitespace separates syllables rather than words.

phological expressiveness, the rightmost column shows the proportion of UD tokens which are annotated with any morphosyntactic attribute.

5.1 Metrics

As noted above, we use the UD datasets for testing our MIMICK algorithm on 23 languages[4] with the supplied train/dev/test division. We measure part-of-speech tagging by overall token-level accuracy.

For morphosyntactic attributes, there does not seem to be an agreed-upon metric for reporting performance. Dzeroski et al. (2000) report per-tag accuracies on a morphosyntactically tagged corpus of Slovene. Faruqui et al. (2016) report macro-averages of F1 scores of 11 languages from UD 1.1 for the various attributes (e.g., part-of-speech, case, gender, tense); recall and precision were calculated for the full set of each attribute's values, pooled together.[5] Agić et al. (2013) report separately on parts-of-speech and morphosyntactic attribute accuracies in Serbian and Croatian, as well as precision, recall, and F1 scores per tag. Georgiev et al. (2012) report token-level accuracy for exact all-attribute tags (e.g. 'Ncmsh' for "Noun short masculine singular definite") in Bulgarian, reaching a tagset of size 680. Müller et al. (2013) do the same for six other languages. We report **micro F1**: each token's value for each attribute is compared separately with the gold labeling, where a correct prediction is a matching non-*NONE* attribute/value assignment. Recall and precision are calculated over the entire set, with F1 defined as their harmonic mean.

5.2 Models

We implement and test the following models:

No-Char. Word embeddings are initialized from Polyglot models, with unseen words assigned the Polyglot-supplied *UNK* vector. Following tuning experiments on all languages with cased script, we found it beneficial to first back off to the lowercased form for an OOV word if its embedding exists, and only otherwise assign *UNK*.

MIMICK. Word embeddings are initialized from Polyglot, with OOV embeddings inferred from a MIMICK model (Section 3) trained on the Polyglot embeddings. Unlike the No-Char case, backing off to lowercased embeddings before using the MIMICK output did not yield conclusive benefits and thus we report results for the more straightforward no-backoff implementation.

CHAR→TAG. Word embeddings are initialized from Polyglot as in the No-Char model (with lowercase backoff), and appended with the output of a character-level LSTM updated during training (Plank et al., 2016). This additional module causes a threefold increase in training time.

Both. Word embeddings are initialized as in MIMICK, and appended with the CHAR→TAG LSTM.

Other models. Several non-Polyglot embedding models were examined, all performed substantially worse than Polyglot. Two of these

[4] When several datasets are available for a language, we use the unmarked corpus.

[5] Details were clarified in personal communication with the authors.

are notable: a random-initialization baseline, and a model initialized from FastText embeddings (tested on English). FastText supplies 300-dimension embeddings for 2.51 million lowercase-only forms, and no *UNK* vector.[6] Both of these embedding models were attempted with and without CHAR→TAG concatenation. Another model, initialized from only MIMICK output embeddings, performed well only on the language with smallest Polyglot training corpus (Latvian). A Polyglot model where OOVs were initialized using an averaged embedding of all Polyglot vectors, rather than the supplied *UNK* vector, performed worse than our No-Char baseline on a great majority of the languages.

Last, we do not employ type-based tagset restrictions. All tag inventories are computed from the training sets and each tag selection is performed over the full set.

5.3 Hyperparameters

Based on development set experiments, we set the following hyperparameters for all models on all languages: two LSTM layers of hidden size 128, MLP hidden layers of size equal to the number of each attribute's possible values; momentum stochastic gradient descent with 0.01 learning rate; 40 training epochs (80 for 5K settings) with a dropout rate of 0.5. The CHAR→TAG models use 20-dimension character embeddings and a single hidden layer of size 128.

6 Results

We report performance in both low-resource and full-resource settings. Low-resource training sets were obtained by randomly sampling training sentences, without replacement, until a predefined token limit was reached. We report the results on the full sets and on $N = 5000$ tokens in Table 5 (part-of-speech tagging accuracy) and Table 6 (morphosyntactic attribute tagging micro-F1). Results for additional training set sizes are shown in Figure 2; space constraints prevent us from showing figures for all languages.

MIMICK as OOV initialization. In nearly all experimental settings on both tasks, across languages and training corpus sizes, the MIMICK embeddings significantly improve over the Polyglot *UNK* embedding for OOV tokens on both POS and morphosyntactic tagging. For POS, the largest margins are in the Slavic languages (Russian, Czech, Bulgarian), where word order is relatively free and thus rich word representations are imperative. Chinese also exhibits impressive improvement across all settings, perhaps due to the large character inventory (> 12,000), for which a model such as MIMICK can learn well-informed embeddings using the large Polyglot vocabulary dataset, overcoming both word- and character-level sparsity in the UD corpus.[7] In morphosyntactic tagging, gains are apparent for Slavic languages and Chinese, but also for agglutinative languages — especially Tamil and Turkish — where the stable morpheme representation makes it easy for subword modeling to provide a type-level signal.[8] To examine the effects on Slavic and agglutinative languages in a more fine-grained view, we present results of multiple training-set size experiments for each model, averaged over five repetitions (with different corpus samples), in Figure 2.

MIMICK vs. CHAR→TAG. In several languages, the MIMICK algorithm fares better than the CHAR→TAG model on part-of-speech tagging in low-resource settings. Table 7 presents the POS tagging improvements that MIMICK achieves over the pre-trained Polyglot models, with and without CHAR→TAG concatenation, with 10,000 tokens of training data. We obtain statistically significant improvements in most languages, even when CHAR→TAG is included. These improvements are particularly substantial for test-set tokens outside the UD training set, as shown in the right two columns. While test set OOVs are a strength of the CHAR→TAG model (Plank et al., 2016), in many languages there are still considerable improvements to be obtained from the application of MIMICK initialization. This suggests that with limited training data, the end-to-end CHAR→TAG model is unable to learn a sufficiently accurate representational mapping from orthography.

7 Conclusion

We present a straightforward algorithm to infer OOV word embedding vectors from pre-trained,

[6] Vocabulary type-level coverage for the English UD corpus: 55.6% case-sensitive, 87.9% case-insensitive.

[7] Character coverage in Chinese Polyglot is surprisingly good: only eight characters from the UD dataset are unseen in Polyglot, across more than 10,000 unseen word types.

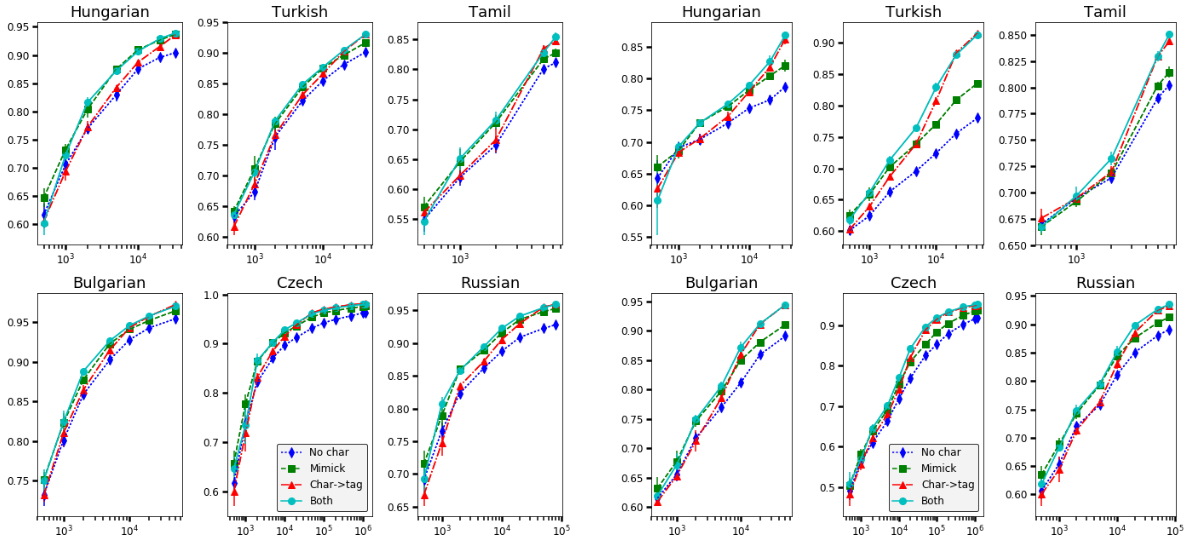
[8] Persian is officially classified as agglutinative but it is mostly so with respect to derivations. Its word-level inflections are rare and usually fusional.

	$N_{train} = 5000$				Full data					
	No-Char	MIMICK	CHAR →TAG	Both	N_{train}	No-Char	MIMICK	CHAR →TAG	Both	PSG 2016*
kk	—	—	—	—	4,949	81.94	83.95	83.64	84.88	
ta	82.30	81.55	84.97	85.22	6,329	80.44	**82.96**	84.11	84.46	
lv	80.44	**84.32**	84.49	**85.91**	13,781	85.77	**87.95**	89.55	89.99	
vi	85.67	*84.22*	84.85	85.43	31,800	89.94	90.34	90.50	90.19	
hu	82.88	**88.93**	85.83	**88.34**	33,017	91.52	**93.88**	94.07	93.74	
tr	83.69	**85.60**	84.23	**86.25**	41,748	90.19	**91.82**	93.11	92.68	
el	93.10	**93.63**	94.05	**94.64**	47,449	97.27	**98.08**	98.09	98.22	
bg	90.97	**93.16**	93.03	**93.52**	50,000	96.63	**97.29**	97.95	97.78	98.23
sv	90.87	**92.30**	92.27	**93.02**	66,645	95.26	**96.27**	96.69	96.87	96.60
eu	82.67	**84.44**	86.01	**86.93**	72,974	91.67	**93.16**	94.46	94.29	95.38
ru	87.40	**89.72**	88.65	**90.91**	79,772	92.59	**95.21**	95.98	95.84	
da	89.46	90.13	89.96	90.55	88,980	94.14	**95.04**	96.13	96.02	96.16
id	89.07	89.34	89.81	90.21	97,531	92.92	93.24	93.41	**93.70**	93.32
zh	80.84	**85.69**	81.84	**85.53**	98,608	90.91	**93.31**	93.36	93.72	
fa	93.50	93.58	93.53	93.71	121,064	96.77	**97.03**	97.20	97.16	97.60
he	90.73	**91.69**	91.93	91.70	135,496	95.65	**96.15**	96.59	96.37	96.62
ro	87.73	**89.18**	88.96	**89.38**	163,262	95.68	**96.72**	97.07	97.09	
en	87.48	**88.45**	88.89	88.89	204,587	93.39	**94.04**	94.90	94.70	95.17
ar	89.01	**90.58**	90.49	90.62	225,853	95.51	**95.72**	96.37	96.24	98.87
hi	87.89	87.77	87.92	88.09	281,057	96.31	96.45	96.64	96.61	96.97
it	91.35	**92.50**	92.45	**93.01**	289,440	97.22	97.47	97.76	97.69	97.90
es	90.54	**91.41**	91.71	91.78	382,436	94.68	94.84	95.08	95.05	95.67
cs	87.97	**90.81**	90.17	**91.29**	1,173,282	96.34	**97.62**	98.18	*97.93*	98.02

Table 5: POS tagging accuracy (UD 1.4 Test). **Bold** (*Italic*) indicates significant improvement (degradation) by McNemar's test, $p < .01$, comparing MIMICK to "No-Char", and "Both" to CHAR→TAG.
* For reference, we copy the reported results of Plank et al. (2016)'s analog to CHAR→TAG. Note that these were obtained on UD 1.2, and without jointly tagging morphosyntactic attributes.

	$N_{train} = 5000$				Full data			
	No-Char	MIMICK	CHAR →TAG	Both	No-Char	MIMICK	CHAR →TAG	Both
kk	—	—	—	—	21.48	20.07	28.47	20.98
ta	80.68	**81.96**	84.26	**85.63**	79.90	**81.93**	84.55	85.01
lv	56.98	**59.86**	64.81	**65.82**	66.16	66.61	76.11	75.44
hu	73.13	**76.30**	73.62	**76.85**	80.04	80.64	86.43	84.12
tr	69.58	**75.21**	75.81	**78.93**	78.31	**83.32**	91.51	90.86
el	86.87	*86.07*	86.40	**87.50**	94.64	**94.96**	96.55	**96.76**
bg	78.26	**81.77**	82.74	**84.93**	91.98	**93.48**	96.12	95.96
sv	82.09	**84.12**	85.26	**88.16**	92.45	**94.20**	96.37	**96.57**
eu	65.29	**66.00**	70.67	*70.27*	82.75	**84.74**	90.58	**91.39**
ru	77.31	**81.84**	79.83	**83.53**	88.80	**91.24**	93.54	93.56
da	80.26	**82.74**	83.59	82.65	92.06	**94.14**	96.05	95.96
zh	63.29	**71.44**	63.50	**74.66**	84.95	85.70	84.86	85.87
fa	84.73	**86.07**	85.94	81.75	95.30	**95.55**	96.90	96.80
he	75.35	68.57	81.06	75.24	90.25	**90.99**	93.35	93.63
ro	84.20	**85.64**	85.61	**87.31**	94.97	**96.10**	97.18	97.14
en	86.71	**87.99**	88.50	**89.61**	95.30	**95.59**	96.40	96.30
ar	84.14	84.17	81.41	*81.11*	94.43	**94.85**	95.50	95.37
hi	83.45	**86.89**	85.64	85.27	96.15	96.21	96.59	**96.67**
it	89.96	**92.07**	91.27	**92.62**	97.32	**97.80**	98.18	98.31
es	88.11	**89.81**	88.58	**89.63**	94.84	**95.44**	96.21	**96.84**
cs	68.66	**72.65**	71.02	**73.61**	91.75	**93.71**	95.29	95.31

Table 6: Micro-F1 for morphosyntactic attributes (UD 1.4 Test). **Bold** (*Italic*) type indicates significant improvement (degradation) by a bootstrapped Z-test, $p < .01$, comparing models as in Table 5. Note that the Kazakh (*kk*) test set has only 78 morphologically tagged tokens.



Part-of-speech tagging (accuracy)

Morpho-syntactic attribute tagging (micro-F1)

Figure 2: Results on agglutinative languages (top) and on Slavic languages (bottom). X-axis is number of training tokens, starting at 500. Error bars are the standard deviations over five random training data subsamples.

Test set	Missing embeddings	Full vocabulary		OOV (UD)	
CHAR→TAG		w/o	with	w/o	with
Persian	2.2%	0.03	**0.41**	**0.83**	**0.81**
Hindi	3.8%	**0.59**	0.21	**3.61**	0.36
English	4.5%	**0.83**	0.25	**3.26**	0.49
Spanish	5.2%	0.33	-0.26	1.03	-0.66
Italian	6.6%	**0.84**	0.28	**1.83**	0.21
Danish	7.8%	0.65	**0.99**	**2.41**	**1.72**
Hebrew	9.2%	**1.25**	**0.40**	**3.03**	0.06
Swedish	9.2%	**1.50**	**0.55**	**4.75**	**1.79**
Bulgarian	9.4%	**0.96**	0.12	**1.83**	-0.11
Czech	10.6%	**2.24**	**1.32**	**5.84**	**2.20**
Latvian	11.1%	**2.87**	**1.03**	**7.29**	**2.71**
Hungarian	11.6%	**2.62**	**2.01**	**5.76**	**4.85**
Turkish	14.5%	**1.73**	**1.69**	**3.58**	**2.71**
Tamil*	16.2%	**2.52**	0.35	2.09	1.35
Russian	16.5%	**2.17**	**1.82**	**4.55**	**3.52**
Greek	17.5%	**1.07**	0.34	**3.30**	1.17
Indonesian	19.1%	**0.46**	0.25	**1.19**	0.75
Kazakh*	21.0%	2.01	1.24	**5.34**	**4.20**
Vietnamese	21.9%	0.53	**1.18**	1.07	**5.73**
Romanian	27.1%	**1.49**	**0.47**	**4.22**	**1.24**
Arabic	27.1%	**1.23**	**0.32**	**2.15**	0.22
Basque	35.3%	**2.39**	**1.06**	**5.42**	**1.68**
Chinese	69.9%	**4.19**	**2.57**	**9.52**	**5.24**

Table 7: Absolute gain in POS tagging accuracy from using MIMICK for 10,000-token datasets (all tokens for Tamil and Kazakh). **Bold** denotes statistical significance (McNemar's test,$p < 0.01$).

limited-vocabulary models, without need to access the originating corpus. This method is particularly useful for low-resource languages and tasks with little labeled data available, and in fact is task-agnostic. Our method improves performance over word-based models on annotated sequence-tagging tasks for a large variety of languages across dimensions of family, orthography, and morphology. In addition, we present a Bi-LSTM approach for tagging morphosyntactic attributes at the token level. In this paper, the MIMICK model was trained using characters as input, but future work may consider the use of other subword units, such as morphemes, phonemes, or even bitmap representations of ideographic characters (Costa-jussà et al., 2017).

8 Acknowledgments

We thank Umashanthi Pavalanathan, Sandeep Soni, Roi Reichart, and our anonymous reviewers for their valuable input. We thank Manaal Faruqui and Ryan McDonald for their help in understanding the metrics for morphosyntactic tagging. The project was supported by project HDTRA1-15-1-0019 from the Defense Threat Reduction Agency.

References

Željko Agić, Nikola Ljubešić, and Danijela Merkler. 2013. Lemmatization and morphosyntactic tagging of Croatian and Serbian. In *4th Biennial International Workshop on Balto-Slavic Natural Language Processing (BSNLP 2013)*.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, pages 183–192, Sofia, Bulgaria.

Parminder Bhatia, Robert Guthrie, and Jacob Eisenstein. 2016. Morphological priors for probabilistic neural word embeddings. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Jan A Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Marta R Costa-jussà, David Aldón, and José AR Fonollosa. 2017. Chinese–spanish neural machine translation enhanced with character and word bitmap fonts. *Machine Translation*, pages 1–13.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. 2014. Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, pages 4585–4592.

Saso Dzeroski, Tomaz Erjavec, and Jakub Zavrel. 2000. Morphosyntactic tagging of slovene: Evaluating taggers and tagsets. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Denver, CO.

Manaal Faruqui, Ryan McDonald, and Radu Soricut. 2016. Morpho-syntactic lexicon generation using graph-based semi-supervised learning. *Transactions of the Association for Computational Linguistics*, 4:1–16.

Georgi Georgiev, Valentin Zhikov, Petya Osenova, Kiril Simov, and Preslav Nakov. 2012. Feature-rich part-of-speech tagging for morphologically complex languages: Application to Bulgarian. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 492–502, Avignon, France.

Jan Hajič and Barbora Hladká. 1998. Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 483–490.

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

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Miryam de Lhoneux and Joakim Nivre. 2016. Ud treebank sampling for comparative parser evaluation. In *The Sixth Swedish Language Technology Conference (SLTC)*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of Empirical Methods for Natural Language Processing (EMNLP)*, pages 322–332.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Kemal Oflazer and Ìlker Kuruöz. 1994. Tagging and morphological disambiguation of turkish text. In *Proceedings of the fourth conference on Applied natural language processing*, pages 144–149. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the Association for Computational Linguistics (ACL)*, Berlin.

Cicero D. Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1818–1826.

Natalie Schluter and Željko Agić. 2017. Empirically sampling universal dependencies. In *The NoDaLiDa Workshop on Universal Dependencies (UDW 2017)*.

Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and Noa Nativ. 2001. Building a tree-bank of modern hebrew text. *Traitement Automatique des Langues*, 42(2):247–380.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. *arXiv preprint arXiv:1607.02789*.

Wenpeng Yin and Hinrich Schütze. 2016. Learning word meta-embeddings. In *Proceedings of the Association for Computational Linguistics (ACL)*, Berlin.

Past, Present, Future: A Computational Investigation of the Typology of Tense in 1000 Languages

Ehsaneddin Asgari[1,2] and Hinrich Schütze[1]

[1]Center for Information and Language Processing, LMU Munich, Germany
[2]Applied Science and Technology, University of California, Berkeley, CA, USA,
asgari@berkeley.edu inquiries@cislmu.org

Abstract

We present SuperPivot, an analysis method for low-resource languages that occur in a superparallel corpus, i.e., in a corpus that contains an order of magnitude more languages than parallel corpora currently in use. We show that SuperPivot performs well for the crosslingual analysis of the linguistic phenomenon of tense. We produce analysis results for more than 1000 languages, conducting – to the best of our knowledge – the largest crosslingual computational study performed to date. We extend existing methodology for leveraging parallel corpora for typological analysis by overcoming a limiting assumption of earlier work: We only require that a linguistic feature is overtly marked in a *few* of thousands of languages as opposed to requiring that it be marked in *all* languages under investigation.

1 Introduction

Significant linguistic resources such as machine-readable lexicons and part-of-speech (POS) taggers are available for at most a few hundred languages. This means that the majority of the languages of the world are low-resource. Low-resource languages like Fulani are spoken by tens of millions of people and are politically and economically important; e.g., to manage a sudden refugee crisis, NLP tools would be of great benefit. Even "small" languages are important for the preservation of the common heritage of humankind that includes natural remedies and linguistic and cultural diversity that can potentially enrich everybody. Thus, developing analysis methods for low-resource languages is one of the most important challenges of NLP today.

We address this challenge by proposing a new method for analyzing what we call *superparallel corpora*, corpora that are by an order of magnitude more parallel than corpora that have been available in NLP to date. The corpus we work with in this paper is the Parallel Bible Corpus (PBC) that consists of translations of the New Testament in 1169 languages. Given that no NLP analysis tools are available for most of these 1169 languages, how can we extract the rich information that is potentially hidden in such superparallel corpora?

The method we propose is based on two hypotheses. **H1 Existence of overt encoding.** For any important linguistic distinction f that is frequently encoded across languages in the world, there are a few languages that encode f overtly on the surface. **H2 Overt-to-overt and overt-to-non-overt projection.** For a language l that encodes f, a projection of f from the "overt languages" to l in the superparallel corpus will identify the encoding that l uses for f, both in cases in which the encoding that l uses is overt and in cases in which the encoding that l uses is non-overt. Based on these two hypotheses, our method proceeds in 5 steps.

1. Selection of a linguistic feature. We select a linguistic feature f of interest. Running example: We select past tense as feature f.

2. Heuristic search for head pivot. Through a heuristic search, we find a language l^h that contains a *head pivot* p^h that is highly correlated with the linguistic feature of interest.

Running example: "ti" in Seychelles Creole (CRS). CRS "ti" meets our requirements for a head pivot well as will be verified empirically in §3. First, "ti" is a surface marker: it is easily identifable through whitespace tokenization and it is not ambiguous, e.g., it does not have a second meaning apart from being a grammatical marker. Second, "ti" is a good marker for past tense in

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 113–124
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

terms of both "precision" and "recall". CRS has mandatory past tense marking (as opposed to languages in which tense marking is facultative) and "ti" is highly correlated with the general notion of past tense.

This does not mean that every clause that a linguist would regard as past tense is marked with "ti" in CRS. For example, some tense-aspect configurations that are similar to English present perfect are marked with "in" in CRS, not with "ti" (e.g., ENG "has commanded" is translated as "in ordonn").

Our goal is not to find a head language and a head pivot that is a perfect marker of f. Such a head pivot probably does not exist; or, more precisely, linguistic features are not completely rigorously defined. In a sense, one of the contributions of this work is that we provide more rigorous definitions of past tense across languages; e.g., "ti" in CRS is one such rigorous definition of past tense and it automatically extends (through projection) to 1000 languages in the superparallel corpus.

3. Projection of head pivot to larger pivot set. Based on an alignment of the head language to the other languages in the superparallel corpus, we project the head pivot to all other languages and search for highly correlated surface markers, i.e., we search for additional pivots in other languages. This projection to more pivots achieves three goals. First, it makes the method more *robust*. Relying on a single pivot would result in many errors due to the inherent noisiness of linguistic data and because several components we use (e.g., alignment of the languages in the superparallel corpus) are imperfect. Second, as we discussed above, the head pivot does not necessarily have high "recall"; our example was that CRS "ti" is not applied to certain clauses that would be translated using present perfect in English. Thus, moving to a larger pivot set *increases recall*. Third, as we will see below, the pivot set can be leveraged to create a *fine-grained map of the linguistic feature*. Consider clauses referring to eventualities in the past that English speakers would render in past progressive, present perfect and simple past tense. Our hope is that the pivot set will cover these distinctions, i.e., one of the pivots marks past progressive, but not present prefect and simple past, another pivot marks present perfect, but not the other two and so on. An example of this type of map, including distinctions like progressive and perfective aspect, is given in §4.

Running example: We compute the correlation of "ti" with words in other languages and select the 100 most highly correlated words as pivots. Examples of pivots we find this way are Torres Strait Creole "bin" (from English "been") and Tzotzil "laj". "laj" is a perfective marker, e.g., "Laj meltzaj -uk" 'LAJ be-made subj' means "It's done being built" (Aissen, 1987).

4. Projection of pivot set to all languages. Now that we have a large pivot set, we project the pivots to all other languages to search for linguistic devices that express the linguistic feature f. Up to this point, we have made the assumption that it is easy to segment text in all languages into pieces of a size that is not too small (individual characters of the Latin alphabet would be too small) and not too large (entire sentences as tokens would be too large). Segmentation on standard delimiters is a good approximation for the majority of languages – but not for all: it undersegments some (e.g., the polysynthetic language Inuit) and oversegments others (e.g., languages that use punctuation marks as regular characters).

For this reason, we do not employ tokenization in this step. Rather we search for character n-grams ($2 \leq n \leq 6$) to find linguistic devices that express f. This implementation of the search procedure is a limitation – there are many linguistic devices that cannot be found using it, e.g., templates in templatic morphology. We leave addressing this for future work (§7).

Running example: We find "-ed" for English and "-te" for German as surface features that are highly correlated with the 100 past tense pivots.

5. Linguistic analysis. The result of the previous steps is a superparallel corpus that is richly annotated with information about linguistic feature f. This structure can be exploited for *the analysis of a single language* l^i that may be the focus of a linguistic investigation. Starting with the character n-grams that were found in the step "projection of pivot set to all languages", we can explore their use and function, e.g, for the mined n-gram "-ed" in English (assuming English is the language l^i and it is unfamiliar to us). Many of the other 1000 languages provide annotations of linguistic feature f for l^i: both the languages that are part of the pivot set (e.g., Tzotzil "laj") and the mined n-grams in other languages that we may have some knowledge about (e.g., "-te" in German).

We can also use the structure we have generated for *typological analysis across languages* following the work of Michael Cysouw ((Cysouw, 2014), §5). Our method is an advancement computationally over Cysouw's work because our method scales to thousands of languages as we demonstrate below.

Running example: We sketch the type of analysis that our new method makes possible in §4.

The above steps "1. heuristic search for head pivot" and "2. projection of head pivot to larger pivot set" are based on H1: we assume the **existence of overt coding** in a subset of languages.

The above steps "2. projection of head pivot to larger pivot set" and "3. projection of pivot set to all languages" are based on H2: we assume that **overt-to-overt and overt-to-non-overt projection** is possible.

In the rest of the paper, we will refer to the method that consists of steps 1 to 5 as *SuperPivot*: "linguistic analysis of SUPERparallel corpora using surface PIVOTs".

We make three contributions. (i) Our basic hypotheses are H1 and H2. (H1) For an important linguistic feature, there exist a few languages that mark it overtly and easily recognizably. (H2) It is possible to project overt markers to overt and non-overt markers in other languages. Based on these two hypotheses we design SuperPivot, a new method for analyzing highly parallel corpora, and show that it performs well for the crosslingual analysis of the linguistic phenomenon of tense. (ii) Given a superparallel corpus, SuperPivot can be used for the analysis of *any low-resource language* represented in that corpus. In the supplementary material, we present results of our analysis for three tenses (past, present, future) for 1163[1] languages. An evaluation of accuracy is presented in Table 3.2. (iii) We extend Michael Cysouw's method of typological analysis using parallel corpora by overcoming several limiting factors. The most important is that Cysouw's method is only applicable if markers of the relevant linguistic feature are recognizable on the surface in *all* languages. In contrast, we only assume that markers of the relevant linguistic feature are recognizable on the surface in *a small number of* languages.

2 SuperPivot: Description of method

1. Selection of a linguistic feature. The linguistic feature of interest f is selected by the person who performs a SuperPivot analysis, i.e., by a linguist, NLP researcher or data scientist. Henceforth, we will refer to this person as the linguist.

In this paper, $f \in F = \{\text{past}, \text{present}, \text{future}\}$.

2. Heuristic search for head pivot. There are several ways for finding the head language and the head pivot. Perhaps the linguist knows a language that has a good head pivot. Or she is a trained typologist and can find the head pivot by consulting the typological literature.

In this paper, we use our knowledge of English and an alignment from English to all other languages to find head pivots. (See below for details on alignment.) We define a "query" in English and search for words that are highly correlated to the query in other languages. For future tense, the query is simply the word "will", so we search for words in other languages that are highly correlated with "will". For present tense, the query is the union of "is", "are" and "am". So we search for words in other languages that are highly correlated with the "merger" of these three words. For past tense, we POS tag the English part of PBC and merge all words tagged as past tense into one past tense word.[2] We then search for words in other languages that are highly correlated with this artificial past tense word.

As an additional constraint, we do not select the most highly correlated word as the head pivot, but the most highly correlated word in a Creole language. Our rationale is that Creole languages are more regular than other languages because they are young and have not accumulated "historical baggage" that may make computational analysis more difficult.

Table 1 lists the three head pivots for F.

3. Projection of head pivot to larger pivot set. We first use fast_align (Dyer et al., 2013) to align the head language to all other languages in the corpus. This alignment is on the word level.

We compute a score for each word in each language based on the number of times it is aligned to the head pivot, the number of times it is aligned to another word and the total frequencies of head pivot and word. We use χ^2 (Casella and Berger, 2008) as the score throughout this paper. Finally,

[1] We exclude six of the 1169 languages because they do not share enough verses with the rest.

[2] Past tense is defined as tags BED, BED*, BEDZ, BEDZ*, DOD*, VBD, DOD. We use NLTK (Bird, 2006).

we select the k words as pivots that have the highest association score with the head pivot.

We impose the constraint that we only select one pivot per language. So as we go down the list, we skip pivots from languages for which we already have found a pivot. We set $k = 100$ in this paper. Table 1 gives the top 10 pivots.

4. Projection of pivot set to all languages. As discussed above, the process so far has been based on tokenization. To be able to find markers that cannot be easily detected on the surface (like "-ed" in English), we identify non-tokenization-based character n-gram features in step 4.

The immediate challenge is that without tokens, we have no alignment between the languages anymore. We could simply assume that the occurrence of a pivot has scope over the entire verse. But this is clearly inadequate, e.g., for the sentence "I arrived yesterday, I'm staying today, and I will leave tomorrow", it is incorrect to say that it is marked as past tense (or future tense) in its entirety. Fortunately, the verses in the New Testament mostly have a simple structure that limits the variation in where a particular piece of content occurs in the verse. We therefore make the assumption that a particular relative position in language l_1 (e.g., the character at relative position 0.62) is aligned with the same relative position in l_2 (i.e., the character at relative position 0.62). This is likely to work for a simple example like "I arrived yesterday, I'm staying today, and I will leave tomorrow" across languages.

In our analysis of errors, we found many cases where this assumption breaks down. A well-known problematic phenomenon for our method is the difference between, say, VSO and SOV languages: the first class puts the verb at the beginning, the second at the end. However, keep in mind that we accumulate evidence over $k = 100$ pivots and then compute aggregate statistics over the entire corpus. As our evaluation below shows, the "linear alignment" assumption does not seem to do much harm given the general robustness of our method.

One design element that increases robustness is that we find the two positions in each verse that are most highly (resp. least highly) correlated with the linguistic feature f. Specifically, we compute the relative position x of each pivot that occurs in the verse and apply a Gaussian filter ($\sigma = 6$ where the unit of length is the character), i.e., we set $p(x) \approx 0.066$ (0.066 is the density of a Gaussian with $\sigma = 6$ at $x = 0$) and center a bell curve around x. The total score for a position x is then the sum of the filter values at x summed over all occurring pivots. Finally, we select the positions x_{min} and x_{max} with lowest and highest values for each verse.

χ^2 is then computed based on the number of times a character n-gram occurs in a window of size w around x_{max} (positive count) and in a window of size w around x_{min} (negative count). Verses in which no pivot occurs are used for the negative count in their entirety. The top-ranked character n-grams are then output for analysis by the linguist. We set $w = 20$.

5. Linguistic analysis. We now have created a structure that contains rich information about the linguistic feature: for each verse we have relative positions of pivots that can be projected across languages. We also have maximum positions within a verse that allow us to pinpoint the most likely place in the vicinity of which linguistic feature f is marked in all languages. This structure can be used for the analysis of individual low-resource languages as well as for typological analysis. We will give an example of such an analysis in §4.

3 Data, experiments and results

3.1 Data

We use a New Testament subset of the Parallel Bible Corpus (PBS) (Mayer and Cysouw, 2014) that consists of 1556 translations of the Bible in 1169 unique languages. We consider two languages to be different if they have different ISO 639-3 codes.

The translations are aligned on the verse level. However, many translations do not have complete coverage, so that most verses are not present in at least one translation. One reason for this is that sometimes several consecutive verses are merged, so that one verse contains material that is in reality not part of it and the merged verses may then be missing from the translation. Thus, there is a trade-off between number of parallel translations and number of verses they have in common. Although some preprocessing was done by the authors of the resource, many translations are not preprocessed. For example, Japanese is not tokenized. We also observed some incorrectness and sparseness in the metadata. One example is that one Fijian translation (see §4) is tagged fij_hindi, but it is Fijian, not Fiji Hindi.

We use the 7958 verses with the best coverage across languages.

3.2 Experiments

1. Selection of a linguistic feature. We conduct three experiments for the linguistic features past tense, present tense and future tense.

2. Heuristic search for head pivot. We use the queries described in §2 for finding the following three head pivots. (i) Past tense head pivot: "ti" in Seychellois Creole (CRS) (McWhorter, 2005). (ii) Present tense head pivot: "ta" in Papiamentu (PAP) (Andersen, 1990). (iii) Future tense head pivot: "bai" in Tok Pisin (TPI) (Traugott, 1978; Sankoff, 1990).

3. Projection of head pivot to larger pivot set. Using the method described in §2, we project each head pivot to a set of $k = 100$ pivots. Table 1 gives the top 10 pivots for each tense.

4. Projection of pivot set to all languages. Using the method described in §2, we compute highly correlated character n-gram features, $2 \leq n \leq 6$, for all 1163 languages.

See §4 for the last step of SuperPivot: **5. Linguistic analysis.**

3.3 Evaluation

We rank n-gram features and retain the top 10, for each linguistic feature, for each language and for each n-gram size. We process 1556 translations. Thus, in total, we extract 1556 × 5 × 10 n-grams.

Table 3.2 shows Mean Reciprocal Rank (MRR) for 10 languages. The rank for a particular ranking of n-grams is the first n-gram that is highly correlated with the relevant tense; e.g., character subsequences of the name "Paulus" are evaluated as incorrect, the subsequence "-ed" in English as correct for past. MRR is averaged over all n-gram sizes, $2 \leq n \leq 6$. Chinese has consistent tense marking only for future, so results are poor. Russian and Polish perform poorly because their central grammatical category is aspect, not tense. The poor performance on Arabic is due to the limits of character n-gram features for a "templatic" language.

During this evaluation, we noticed a surprising amount of variation within translations of one language; e.g., top-ranked n-grams for some German translations include names like "Paulus". We suspect that for literal translations, linear alignment (§2) yields good n-grams. But many translations are free, e.g., they change the sequence of clauses. This deteriorates mined n-grams. See §7.

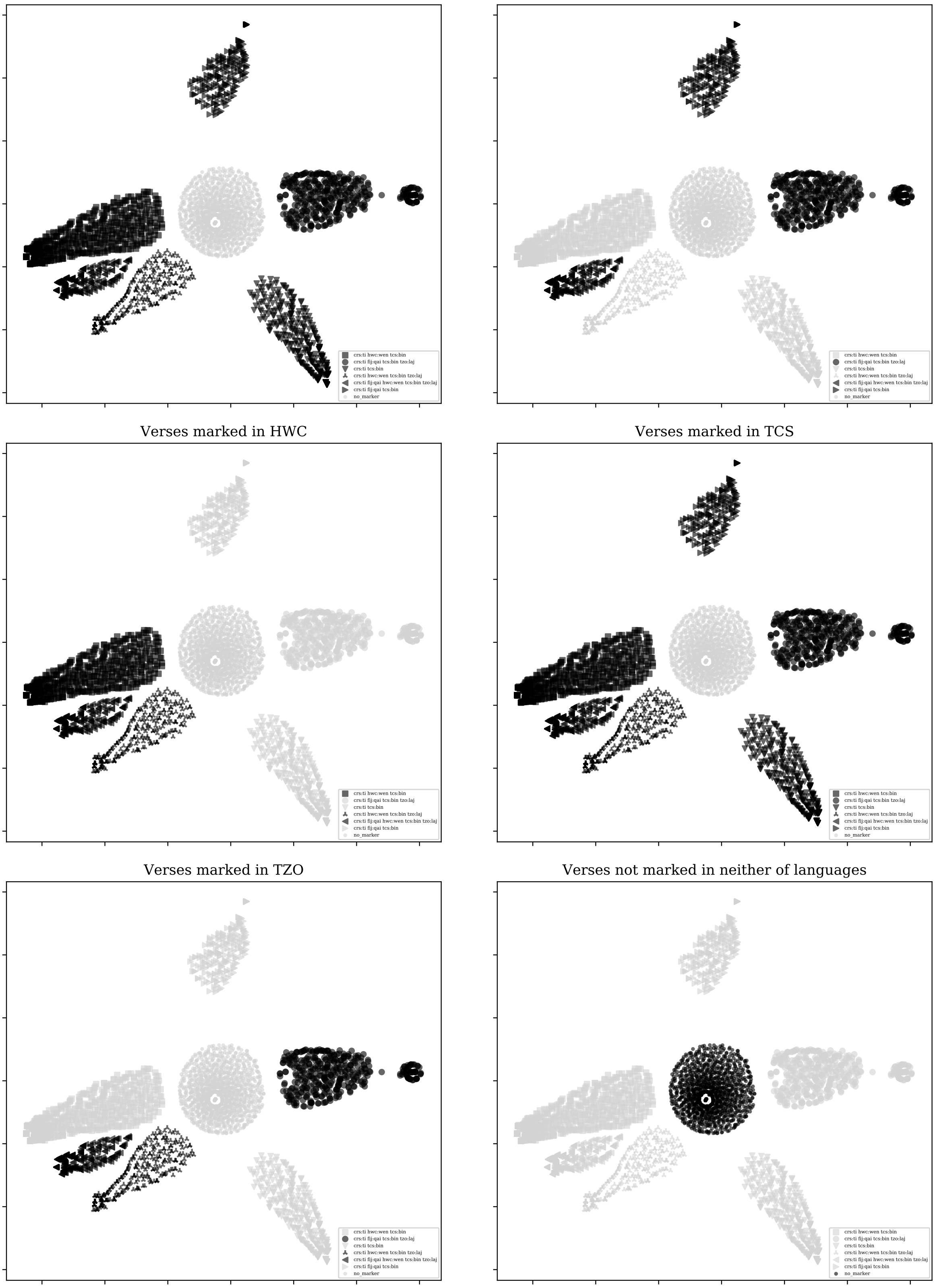
A reviewer points out that simple baselines may be available if all we want to do is compute features highly associated with past tense as evaluated in Table 3.2. As one such baseline, they suggested to first perform a word alignment with the head pivot and then search for highly associated features in the words that were aligned with the head pivot. We implemented this baseline and measured its performance. Indeed, the results were roughly comparable to the more complex method that we evaluate in Table 3.2.

However, our evaluation was not designed to be a direct evaluation of our method, but only meant as a relatively easy way of getting a quantitative sense of the accuracy of our results. The core result of our method is a corpus in which each language annotates each other language. This is only meaningful on the token or context level, not on the word level. For example, recognizing "-ed" as a possible past tense marker in English and applying it uniformly throughout the corpus would result in the incorrect annotation of the adjective "red" as a past tense form. In our proposed method, this will not happen since the annotation proceeds from reliable pivots to less reliable features, not the other way round. Nevertheless, we agree with the reviewer that we do not make enough use of "type-level" features in our method (type-level features of non-pivot languages) and this is something we plan to address in the future.

4 A map of past tense

To illustrate the potential of our method we select five out of the 100 past tense pivots that give rise to large clusters of distinct combinations. Specifically, starting with CRS, we find other pivots that "split" the set of verses that contain the CRS past tense pivot "ti" into two parts that have about the same size. This gives us two sets. We now look for a pivot that splits one of these two sets about evenly and so on. After iterating four times, we arrive at five pivots: CRS "ti", Fijian (FIJ) "qai", Hawaiian Creole (HWC) "wen", Torres Strait Creole (TCS) "bin" and Tzotzil (TZO) "laj".

Figure 1 shows a t-SNE (Maaten and Hinton, 2008) visualization of the large clusters of combinations that are found for these five languages, including one cluster of verses that do not contain any of the five pivots.



Verses marked in HWC

Verses marked in TCS

Verses marked in TZO

Verses not marked in neither of languages

Figure 1: A map of past tense based on the largest clusters of verses with particular combinations of the past tense pivots from Seychellois Creole (CRS), Fijian (FIJ), Hawaiian Creole (HWC), Torres Strait Creole (TCS) and Tzotzil (TZO). For each of the five languages, we present a subfigure that highlights the subset of verse clusters that are marked by the pivot of that language. The sixth subfigure highlights verses not marked by any of the five pivots.

		past			present			future	
	code	language	pivot	code	language	pivot	code	language	pivot
HPs	CRS	Seychelles C.	*ti*	PAP	Papiamentu	*ta*	TPI	Tok Pisin	*bai*
	GUX	Gourmanchéma	*den*	NOB	Norwegian Bokmål	*er*	LID	Nyindrou	*kameh*
	MAW	Mampruli	*daa*	HIF	Fiji Hindi	*hei*	GUL	Sea Island C.	*gwine*
	GFK	Patpatar	*ga*	AFR	Afrikaans	*is*	TGP	Tangoa	*pa*
	YAL	Yalunka	*yi*	DAN	Danish	*er*	BUK	Bugawac	*oc*
	TOH	Gitonga	*di*	SWE	Swedish	*är*	BIS	Bislama	*bambae*
	DGI	Northern Dagara	*tι*	EPO	Esperanto	*estas*	PIS	Pijin	*bae*
	BUM	Bulu (Cameroon)	*nga*	ELL	Greek	*ϵίναι*	APE	Bukiyip	*eke*
	TCS	Torres Strait C.	*bin*	HIN	Hindi	*haai*	HWC	Hawaiian C.	*goin*
	NDZ	Ndogo	*giì*	NAQ	Khoekhoe	*ra*	NHR	Nharo	*gha*

Table 1: Top ten past, present, and future tense pivots extracted from 1163 languages. HPs = head pivots. C. = Creole

language	past	present	future	all
Arabic	1.00	0.39	0.77	0.72
Chinese	0.00	0.00	0.87	0.29
English	1.00	1.00	1.00	1.00
French	1.00	1.00	1.00	1.00
German	1.00	1.00	1.00	1.00
Italian	1.00	1.00	1.00	1.00
Persian	0.77	1.00	1.00	0.92
Polish	1.00	1.00	0.58	0.86
Russian	0.90	0.50	0.62	0.67
Spanish	1.00	1.00	1.00	1.00
all	0.88	0.79	0.88	0.85

Table 2: MRR results for step 4. See text for details.

This figure is a map of past tense for all 1163 languages, not just for CRS, FIJ, HWC, TCS and TZO: once the interpretation of a particular cluster has been established based on CRS, FIJ, HWC, TCS and TZO, we can investigate this cluster in the 1164 other languages by looking at the verses that are members of this cluster. This methodology supports the empirical investigation of questions like "how is progressive past tense expressed in language X"? We just need to look up the cluster(s) that correspond to progressive past tense, look up the verses that are members and retrieve the text of these verses in language X.

To give the reader a flavor of the distinctions that are reflected in these clusters, we now list phenomena that are characteristic of verses that contain only one of the five pivots; these phenomena identify properties of one language that the other four do not have.

CRS "ti". CRS has a set of markers that can be systematically combined, in particular, a progressive marker "pe" that can be combined with the past tense marker "ti". As a result, past progressive sentences in CRS are generally marked with "ti". Example: "43004031 Meanwhile, the disciples were urging Jesus, 'Rabbi, eat something.'" "crs_bible 43004031 Pandan sa letan, bann disip ti pe sipliy Zezi, 'Met! Manz en pe.'"

The other four languages do not consistently use the pivot for marking the past progressive; e.g., HWC uses "was begging" in 43004031 (instead of "wen") and TCS uses "kip tok strongwan" 'keep talking strongly' in 43004031 (instead of "bin").

FIJ "qai". This pivot means "and then". It is highly correlated with past tense in the New Testament because most sequential descriptions of events are descriptions of past events. But there are also some non-past sequences. Example: "eng_newliving 44009016 And I will show him how much he must suffer for my name's sake." "fij_hindi 44009016 Au na qai vakatakila vua na levu ni ka e na sota kaya e na vukuqu." This verse is future tense, but it continues a temporal sequence (it starts in the preceding verse) and therefore FIJ uses "qai". The pivots of the other four languages are not general markers of temporal sequentiality, so they are not used for the future.

HWC "wen". HWC is less explicit than the other four languages in some respects and more explicit in others. It is less explicit in that not all sentences in a sequence of past tense sentences need to be marked explicitly with "wen", resulting in some sentences that are indistinguishable from present tense. On the other hand, we found many cases of noun phrases in the other four languages that refer implicitly to the past, but are translated as a verb with explicit past tense marking in HWC. Examples: "hwc_2000 40026046 Da guy who wen set me up . . . " 'the guy who WEN set me up', "eng_newliving 40026046 . . . my betrayer . . . "; "hwc_2000 43008005 . . . Moses wen tell us in da Rules . . . " 'Moses WEN tell us in the rules', "eng_newliving 43008005 The law of Moses says . . . "; "hwc_2000 47006012 We wen give you guys our love . . . ", "eng_newliving 47006012 There is no lack of love on our part . . . ". In these cases, the other four languages (and English too) use a noun phrase with no tense marking that is translated as a tense-marked clause in HWC.

While preparing this analysis, we realized that HWC "wen" unfortunately does not meet one of the criteria we set out for pivots: it is not unambiguous. In addition to being a past tense marker (derived from standard English "went"), it can also be a conjunction, derived from "when". This ambiguity is the cause for some noise in the clusters marked for presence of HWC "wen" in the figure.

TCS "bin". Conditionals is one pattern we found in verses that are marked with TCS "bin", but are not marked for past tense in the other four languages. Example: "tcs_bible 46015046 Wanem i bin kam pas i da nomal bodi ane den da spiritbodi i bin kam apta." 'what came first is the normal body and then the spirit body came after', "eng_newliving 46015046 What comes first is the natural body, then the spiritual body comes later." Apparently, "bin" also has a modal aspect in TCS: generic statements that do not refer to specific events are rendered using "bin" in TCS whereas the other four languages (and also English) use the default unmarked tense, i.e., present tense.

TZO "laj". This pivot indicates perfective aspect. The other four past tense pivots are not perfective markers, so that there are verses that are marked with "laj", but not marked with the past tense pivots of the other four languages. Example: "tzo_huixtan 40010042 . . . ja'ch-ac'bat bendición yu'un hech laj spas . . . " (literally "a blessing . . . LAJ make"), "eng_newliving 40010042 . . . you will surely be rewarded." Perfective aspect and past are correlated in the real world since most events that are viewed as simple wholes are in the past. But future events can also be viewed this way as the example shows.

Similar maps for present and future tenses are presented in the supplementary material.

5 Related work

Our work is inspired by (Cysouw, 2014; Cysouw and Wälchli, 2007); see also (Dahl, 2007; Wälchli, 2010). Cysouw creates maps like Figure 1 by manually identifying occurrences of the proper noun "Bible" in a parallel corpus of Jehovah's Witnesses' texts. Areas of the map correspond to semantic roles, e.g., the Bible as actor (it tells you to do something) or as object (it was printed). This is a definition of semantic roles that is complementary to and different from prior typological research because it is empirically grounded in real language use across a large number of languages. It allows typologists to investigate traditional questions from a new perspective.

The field of **typology** is important for both theoretical (Greenberg, 1960; Whaley, 1996; Croft, 2002) and computational (Heiden et al., 2000; Santaholma, 2007; Bender, 2009, 2011) linguistics. Typology is concerned with all areas of linguistics: morphology (Song, 2014), syntax (Comrie, 1989; Croft, 2001; Croft and Poole, 2008; Song, 2014), semantic roles (Hartmann et al., 2014; Cysouw, 2014), semantics (Koptjevskaja-Tamm et al., 2007; Dahl, 2014; Wälchli and Cysouw, 2012; Sharma, 2009), etc. Typological information is important for many NLP tasks including discourse analysis (Myhill and Myhill, 1992), information retrieval (Pirkola, 2001), POS tagging (Bohnet and Nivre, 2012), parsing (Bohnet and Nivre, 2012; McDonald et al., 2013), machine translation (Hajič et al., 2000; Kunchukuttan and Bhattacharyya, 2016) and morphology (Bohnet et al., 2013).

Tense is a central phenomenon in linguistics and the languages of the world differ greatly in whether and how they express tense (Traugott, 1978; Bybee and Dahl, 1989; Dahl, 2000, 1985; Santos, 2004; Dahl, 2007; Santos, 2004; Dahl, 2014).

Low resource. Even resources with the widest coverage like World Atlas of Linguistic Structures (WALS) (Dryer et al., 2005) have little information for hundreds of languages. Many researchers have taken advantage of parallel information for extracting linguistic knowledge in low-resource settings (Resnik et al., 1997; Resnik, 2004; Mihalcea and Simard, 2005; Mayer and Cysouw, 2014;

Christodouloupoulos and Steedman, 2015; Lison and Tiedemann, 2016).

Parallel projection. Parallel projection across languages has been used for a variety of NLP tasks. Machine translation aside, which is the most natural task on parallel corpora (Brown et al., 1993), parallel projection has been used for sense disambiguation (Ide, 2000), parsing (Hwa et al., 2005), paraphrasing (Bannard and Callison-Burch, 2005), part-of-speech tagging (Mukerjee et al., 2006), coreference resolution (de Souza and Orăsan, 2011), event marking (Nordrum, 2015), morphological segmentation (Chung et al., 2016), bilingual analysis of linguistic marking (McEnery and Xiao, 1999; Xiao and McEnery, 2002), as well as language classification (Asgari and Mofrad, 2016; Östling and Tiedemann, 2017).

6 Discussion

Our motivation is not to develop a method that can then be applied to many other corpora. Rather, our motivation is that many of the more than 1000 languages in the Parallel Bible Corpus are low-resource and that providing a method for creating the first richly annotated corpus (through the projection of annotation we propose) for many of these languages is a significant contribution.

The original motivation for our approach is provided by the work of the typologist Michael Cysouw. He created the same type of annotation as we, but he produced it manually whereas we use automatic methods. But the structure of the annotation and its use in linguistic analysis is the same as what we provide.

The basic idea of the utility of the final outcome of SuperPivot is that the 1163 languages all richly annotate each other. As long as there are a few among the 1163 languages that have a clear marker for linguistic feature f, then this marker can be projected to all other languages to richly annotate them. For any linguistic feature, there is a good chance that a few languages clearly mark it. Of course, this small subset of languages will be different for every linguistic feature.

Thus, even for extremely resource-poor languages for which at present no annotated resources exist, SuperPivot will make available richly annotated corpora that should advance linguistic research on these languages.

7 Conclusion

We presented SuperPivot, an analysis method for low-resource languages that occur in a superparallel corpus, i.e., in a corpus that contains an order of magnitude more languages than parallel corpora currently in use. We showed that SuperPivot performs well for the crosslingual analysis of the linguistic phenomenon of tense. We produced analysis results for more than 1000 languages, conducting – to the best of our knowledge – the largest crosslingual computational study performed to date. We extended existing methodology for leveraging parallel corpora for typological analysis by overcoming a limiting assumption of earlier work. We only require that a linguistic feature is overtly marked in a *few* of thousands of languages as opposed to requiring that it be marked in *all* languages under investigation.

8 Future directions

There are at least two future directions that seem promising to us.

- Creating a common map of tense along the lines of Figure 1, but unifying the three tenses
- Addressing shortcomings of the way we compute alignments: (i) generalizing character n-grams to more general features, so that templates in templatic morphology, reduplication and other more complex manifestations of linguistic features can be captured; (ii) use n-gram features of different lengths to account for differences among languages, e.g., shorter ones for Chinese, longer ones for English; (iii) segmenting verses into clauses and performing alignment not on the verse level (which caused many errors in our experiments), but on the clause level instead; (iv) using global information more effectively, e.g., by extracting alignment features from automatically induced bi- or multilingual lexicons.

Acknowledgments

We gratefully acknowledge financial support from Volkswagenstiftung and fruitful discussions with Fabienne Braune, Michael Cysouw, Alexander Fraser, Annemarie Friedrich, Mohsen Mahdavi Mazdeh, Mohammad R.K. Mofrad, and Benjamin Roth. We are indebted to Michael Cysouw for the Parallel Bible Corpus.

References

Judith L. Aissen. 1987. *Tzotzil Clause Structure*. Springer.

Roger W Andersen. 1990. Papiamentu tense-aspect, with special attention to discourse. *Pidgin and creole tense-mood-aspect systems*, pages 59–96.

Ehsaneddin Asgari and Mohammad R. K. Mofrad. 2016. Comparing fifty natural languages and twelve genetic languages using word embedding language divergence (WELD) as a quantitative measure of language distance. In *Proceedings of the NAACL-HLT Workshop on Multilingual and Cross-lingual Methods in NLP*, pages 65–74.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604. Association for Computational Linguistics.

Emily M Bender. 2009. Linguistically naïve!= language independent: why nlp needs linguistic typology. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, pages 26–32. Association for Computational Linguistics.

Emily M Bender. 2011. On achieving and evaluating language-independence in nlp. *Linguistic Issues in Language Technology*, 6(3):1–26.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465. Association for Computational Linguistics.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajič. 2013. Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, 1:415–428.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Joan L Bybee and Östen Dahl. 1989. *The creation of tense and aspect systems in the languages of the world*. John Benjamins Amsterdam.

George Casella and Roger L. Berger. 2008. *Statistical Inference*. Thomson.

Christos Christodouloupoulos and Mark Steedman. 2015. A massively parallel corpus: the bible in 100 languages. *Language resources and evaluation*, 49(2):375–395.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

Bernard Comrie. 1989. *Language universals and linguistic typology: Syntax and morphology*. University of Chicago press.

William Croft. 2001. *Radical construction grammar: Syntactic theory in typological perspective*. Oxford University Press on Demand.

William Croft. 2002. *Typology and universals*. Cambridge University Press.

William Croft and Keith T Poole. 2008. Inferring universals from grammatical variation: Multidimensional scaling for typological analysis. *Theoretical linguistics*, 34(1):1–37.

Michael Cysouw. 2014. Inducing semantic roles. *Perspectives on semantic roles*, pages 23–68.

Michael Cysouw and Bernhard Wälchli. 2007. Parallel texts: using translational equivalents in linguistic typology. *STUF-Sprachtypologie und Universalienforschung*, 60(2):95–99.

Östen Dahl. 1985. *Tense and aspect systems*. Basil Blackwell.

Östen Dahl. 2000. *Tense and Aspect in the Languages of Europe*. Walter de Gruyter.

Östen Dahl. 2007. From questionnaires to parallel corpora in typology. *STUF-Sprachtypologie und Universalienforschung*, 60(2):172–181.

Östen Dahl. 2014. The perfect map: Investigating the cross-linguistic distribution of tame categories in a parallel corpus. *Aggregating Dialectology, Typology, and Register Contents Analysis. Linguistic Variation in Text and Speech. Linguae & litterae*, 28:268–289.

Matthew S Dryer, David Gil, Bernard Comrie, Hagen Jung, Claudia Schmidt, et al. 2005. *The world atlas of language structures*. Oxford University Press.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 644–648.

Joseph H Greenberg. 1960. A quantitative approach to the morphological typology of language. *International journal of American linguistics*, 26(3):178–194.

Jan Hajič, Jan Hric, and Vladislav Kuboň. 2000. Machine translation of very close languages. In *Proceedings of the sixth conference on Applied natural language processing*, pages 7–12. Association for Computational Linguistics.

Iren Hartmann, Martin Haspelmath, and Michael Cysouw. 2014. Identifying semantic role clusters and alignment types via microrole coexpression tendencies. *Studies in Language. International Journal sponsored by the Foundation "Foundations of Language"*, 38(3):463–484.

Serge Heiden, Sophie Prévost, Benoit Habert, Helka Folch, Serge Fleury, Gabriel Illouz, Pierre Lafon, and Julien Nioche. 2000. Typtex: Inductive typological text classification by multivariate statistical analysis for nlp systems tuning/evaluation. In *Maria Gavrilidou, George Carayannis, Stella Markantonatou, Stelios Piperidis, Gregory Stainhaouer (éds) Second International Conference on Language Resources and Evaluation*, pages p–141.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(03):311–325.

Nancy Ide. 2000. Cross-lingual sense determination: Can it work? *Computers and the Humanities*, 34(1):223–234.

Maria Koptjevskaja-Tamm, Martine Vanhove, and Peter Koch. 2007. Typological approaches to lexical semantics. *Linguistic typology*, 11(1):159–185.

Anoop Kunchukuttan and Pushpak Bhattacharyya. 2016. Faster decoding for subword level phrase-based smt between related languages. *arXiv preprint arXiv:1611.00354*.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Thomas Mayer and Michael Cysouw. 2014. Creating a massively parallel bible corpus. *Oceania*, 135(273):40.

Ryan T McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *ACL (2)*, pages 92–97.

Tony McEnery and Richard Xiao. 1999. Domains, text types, aspect marking and english-chinese translation. *Languages in Contrast*, 2(2):211–229.

John H McWhorter. 2005. *Defining creole*. Oxford University Press.

Rada Mihalcea and Michel Simard. 2005. Parallel texts. *Natural Language Engineering*, 11(03):239–246.

Amitabha Mukerjee, Ankit Soni, and Achla M Raina. 2006. Detecting complex predicates in hindi using pos projection across parallel corpora. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 28–35. Association for Computational Linguistics.

John Myhill and Myhill. 1992. *Typological discourse analysis: Quantitative approaches to the study of linguistic function*. Blackwell Oxford.

Lene Nordrum. 2015. Exploring spontaneous-event marking though parallel corpora: Translating english ergative intransitive constructions into norwegian and swedish. *Languages in Contrast*, 15(2):230–250.

Robert Östling and Jörg Tiedemann. 2017. Continuous multilinguality with language vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, page 644–649. Association for Computational Linguistics.

Ari Pirkola. 2001. Morphological typology of languages for ir. *Journal of Documentation*, 57(3):330–348.

Philip Resnik. 2004. Exploiting hidden meanings: Using bilingual text for monolingual annotation. *Computational Linguistics and Intelligent Text Processing*, pages 283–299.

Philip Resnik, Mari Broman Olsen, and Mona Diab. 1997. Creating a parallel corpus from the book of 2000 tongues. In *Proceedings of the Text Encoding Initiative 10th Anniversary User Conference (TEI-10)*. Citeseer.

Gillian Sankoff. 1990. The grammaticalization of tense and aspect in tok pisin and sranan. *Language Variation and Change*, 2(03):295–312.

Marianne Elina Santaholma. 2007. Grammar sharing techniques for rule-based multilingual nlp systems. *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA)*.

Diana Santos. 2004. *Translation-based corpus studies: Contrasting English and Portuguese tense and aspect systems*. 50. Rodopi.

Devyani Sharma. 2009. Typological diversity in new englishes. *English World-Wide*, 30(2):170–195.

Jae Jung Song. 2014. *Linguistic typology: Morphology and syntax*. Routledge.

José Guilherme Camargo de Souza and Constantin Orăsan. 2011. Can projected chains in parallel corpora help coreference resolution? In *Discourse Anaphora and Anaphor Resolution Colloquium*, pages 59–69. Springer.

Elizabeth Closs Traugott. 1978. On the expression of spatio-temporal relations in language. *Universals of human language*, 3:369–400.

Bernhard Wälchli. 2010. The consonant template in synchrony and diachrony. *Baltic linguistics*, 1.

Bernhard Wälchli and Michael Cysouw. 2012. Lexical typology through similarity semantics: Toward a semantic map of motion verbs. *Linguistics*, 50(3):671–710.

Lindsay J Whaley. 1996. *Introduction to typology: the unity and diversity of language*. Sage Publications.

RZ Xiao and AM McEnery. 2002. A corpus-based approach to tense and aspect in english-chinese translation. In *The 1st International Symposium on Contrastive and Translation Studies between Chinese and English*.

Neural Machine Translation with Source-Side Latent Graph Parsing

Kazuma Hashimoto and Yoshimasa Tsuruoka
The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan
{hassy,tsuruoka}@logos.t.u-tokyo.ac.jp

Abstract

This paper presents a novel neural machine translation model which jointly learns translation and source-side latent graph representations of sentences. Unlike existing pipelined approaches using syntactic parsers, our end-to-end model learns a latent graph parser as part of the encoder of an attention-based neural machine translation model, and thus the parser is optimized according to the translation objective. In experiments, we first show that our model compares favorably with state-of-the-art sequential and pipelined syntax-based NMT models. We also show that the performance of our model can be further improved by pretraining it with a small amount of treebank annotations. Our final ensemble model significantly outperforms the previous best models on the standard English-to-Japanese translation dataset.

1 Introduction

Neural Machine Translation (NMT) is an active area of research due to its outstanding empirical results (Bahdanau et al., 2015; Luong et al., 2015; Sutskever et al., 2014). Most of the existing NMT models treat each sentence as a sequence of tokens, but recent studies suggest that syntactic information can help improve translation accuracy (Eriguchi et al., 2016b, 2017; Sennrich and Haddow, 2016; Stahlberg et al., 2016). The existing syntax-based NMT models employ a syntactic parser trained by supervised learning in advance, and hence the parser is not adapted to the translation tasks. An alternative approach for leveraging syntactic structure in a language processing task is to jointly learn syntactic trees of the sentences along with the target task (Socher et al., 2011; Yogatama et al., 2017).

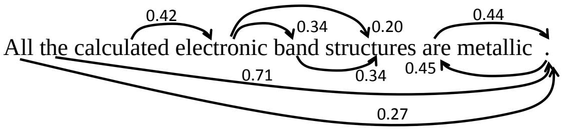

All the calculated electronic band structures are metallic .
0.42
0.34
0.20
0.44
0.71
0.34
0.45
0.27

Figure 1: An example of the learned latent graphs. Edges with a small weight are omitted.

Motivated by the promising results of recent joint learning approaches, we present a novel NMT model that can learn a task-specific latent graph structure for each source-side sentence. The graph structure is similar to the dependency structure of the sentence, but it can have cycles and is learned specifically for the translation task. Unlike the aforementioned approach of learning single syntactic trees, our latent graphs are composed of "soft" connections, i.e., the edges have real-valued weights (Figure 1). Our model consists of two parts: one is a task-independent parsing component, which we call a *latent graph parser*, and the other is an attention-based NMT model. The latent parser can be independently pre-trained with human-annotated treebanks and is then adapted to the translation task.

In experiments, we demonstrate that our model can be effectively pre-trained by the treebank annotations, outperforming a state-of-the-art sequential counterpart and a pipelined syntax-based model. Our final ensemble model outperforms the previous best results by a large margin on the WAT English-to-Japanese dataset.

2 Latent Graph Parser

We model the latent graph parser based on dependency parsing. In dependency parsing, a sentence is represented as a tree structure where each node corresponds to a word in the sentence and

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 125–135
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

a unique *root* node (ROOT) is added. Given a sentence of length N, the parent node $H_{w_i} \in \{w_1, \dots, w_N, \text{ROOT}\}$ ($H_{w_i} \neq w_i$) of each word w_i ($1 \leq i \leq N$) is called its *head*. The sentence is thus represented as a set of tuples $(w_i, H_{w_i}, \ell_{w_i})$, where ℓ_{w_i} is a dependency label.

In this paper, we remove the constraint of using the tree structure and represent a sentence as a set of tuples $(w_i, p(H_{w_i}|w_i), p(\ell_{w_i}|w_i))$, where $p(H_{w_i}|w_i)$ is the probability distribution of w_i's parent nodes, and $p(\ell_{w_i}|w_i)$ is the probability distribution of the dependency labels. For example, $p(H_{w_i} = w_j|w_i)$ is the probability that w_j is the parent node of w_i. Here, we assume that a special token ⟨EOS⟩ is appended to the end of the sentence, and we treat the ⟨EOS⟩ token as ROOT. This approach is similar to that of graph-based dependency parsing (McDonald et al., 2005) in that a sentence is represented with a set of weighted arcs between the words. To obtain the *latent graph* representation of the sentence, we use a dependency parsing model based on multi-task learning proposed by Hashimoto et al. (2017).

2.1 Word Representation

The i-th input word w_i is represented with the concatenation of its d_1-dimensional word embedding $v_{dp}(w_i) \in \mathbb{R}^{d_1}$ and its character n-gram embedding $c(w_i) \in \mathbb{R}^{d_1}$: $x(w_i) = [v_{dp}(w_i); c(w_i)]$. $c(w_i)$ is computed as the average of the embeddings of the character n-grams in w_i.

2.2 POS Tagging Layer

Our latent graph parser builds upon multi-layer bi-directional Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) units (Graves and Schmidhuber, 2005). In the first layer, POS tagging is handled by computing a hidden state $h_i^{(1)} = [\overrightarrow{h}_i^{(1)}; \overleftarrow{h}_i^{(1)}] \in \mathbb{R}^{2d_1}$ for w_i, where $\overrightarrow{h}_i^{(1)} = \text{LSTM}(\overrightarrow{h}_{i-1}^{(1)}, x(w_i)) \in \mathbb{R}^{d_1}$ and $\overleftarrow{h}_i^{(1)} = \text{LSTM}(\overleftarrow{h}_{i+1}^{(1)}, x(w_i)) \in \mathbb{R}^{d_1}$ are hidden states of the forward and backward LSTMs, respectively. $h_i^{(1)}$ is then fed into a softmax classifier to predict a probability distribution $p_i^{(1)} \in \mathbb{R}^{C^{(1)}}$ for word-level tags, where $C^{(1)}$ is the number of POS classes. The model parameters of this layer can be learned not only by human-annotated data, but also by backpropagation from higher layers, which are described in the next section.

2.3 Dependency Parsing Layer

Dependency parsing is performed in the second layer. A hidden state $h_i^{(2)} \in \mathbb{R}^{2d_1}$ is computed by $\overrightarrow{h}_i^{(2)} = \text{LSTM}(\overrightarrow{h}_{i-1}^{(2)}, [x(w_i); y(w_i); \overrightarrow{h}_i^{(1)}])$ and $\overleftarrow{h}_i^{(2)} = \text{LSTM}(\overleftarrow{h}_{i+1}^{(2)}, [x(w_i); y(w_i); \overleftarrow{h}_i^{(1)}])$, where $y(w_i) = W_\ell^{(1)} p_i^{(1)} \in \mathbb{R}^{d_2}$ is the POS information output from the first layer, and $W_\ell^{(1)} \in \mathbb{R}^{d_2 \times C^{(1)}}$ is a weight matrix.

Then, (soft) edges of our latent graph representation are obtained by computing the probabilities

$$p(H_{w_i} = w_j|w_i) = \frac{\exp(m(i,j))}{\sum_{k \neq i} \exp(m(i,k))}, \quad (1)$$

where $m(i,k) = h_k^{(2)\mathrm{T}} W_{dp} h_i^{(2)}$ ($1 \leq k \leq N+1, k \neq i$) is a scoring function with a weight matrix $W_{dp} \in \mathbb{R}^{2d_1 \times 2d_1}$. While the models of Hashimoto et al. (2017), Zhang et al. (2017), and Dozat and Manning (2017) learn the model parameters of their parsing models only by human-annotated data, we allow the model parameters to be learned by the translation task.

Next, $[h_i^{(2)}; z(H_{w_i})]$ is fed into a softmax classifier to predict the probability distribution $p(\ell_{w_i}|w_i)$, where $z(H_{w_i}) \in \mathbb{R}^{2d_1}$ is the weighted average of the hidden states of the parent nodes: $\sum_{j \neq i} p(H_{w_i} = w_j|w_i) h_j^{(2)}$. This results in the latent graph representation $(w_i, p(H_{w_i}|w_i), p(\ell_{w_i}|w_i))$ of the input sentence.

3 NMT with Latent Graph Parser

The latent graph representation described in Section 2 can be used for any sentence-level tasks, and here we apply it to an Attention-based NMT (ANMT) model (Luong et al., 2015). We modify the encoder and the decoder in the ANMT model to learn the latent graph representation.

3.1 Encoder with Dependency Composition

The ANMT model first encodes the information about the input sentence and then generates a sentence in another language. The encoder represents the word w_i with a word embedding $v_{enc}(w_i) \in \mathbb{R}^{d_3}$. It should be noted that $v_{enc}(w_i)$ is different from $v_{dp}(w_i)$ because each component is separately modeled. The encoder then takes the word embedding $v_{enc}(w_i)$ and the hidden state $h_i^{(2)}$ as the input to a uni-directional LSMT:

$$h_i^{(enc)} = \text{LSTM}(h_{i-1}^{(enc)}, [v_{enc}(w_i); h_i^{(2)}]), \quad (2)$$

where $h_i^{(enc)} \in \mathbb{R}^{d_3}$ is the hidden state corresponding to w_i. That is, the encoder of our model is a three-layer LSTM network, where the first two layers are bi-directional.

In the sequential LSTMs, relationships between words in distant positions are not *explicitly* considered. In our model, we explicitly incorporate such relationships into the encoder by defining a dependency composition function:

$$dep(w_i) = \tanh(W_{dep}[h_i^{enc}; \overline{h}(H_{w_i}); p(\ell_{w_i}|w_i)]), \tag{3}$$

where $\overline{h}(H_{w_i}) = \sum_{j \neq i} p(H_{w_i} = w_j|w_i) h_j^{(enc)}$ is the weighted average of the hidden states of the parent nodes.

Note on character n-gram embeddings In NMT models, sub-word units are widely used to address rare or unknown word problems (Sennrich et al., 2016). In our model, the character n-gram embeddings are fed through the latent graph parsing component. To the best of our knowledge, the character n-gram embeddings have never been used in NMT models. Wieting et al. (2016), Bojanowski et al. (2017), and Hashimoto et al. (2017) have reported that the character n-gram embeddings are useful in improving several NLP tasks by better handling unknown words.

3.2 Decoder with Attention Mechanism

The decoder of our model is a single-layer LSTM network, and the initial state is set with $h_{N+1}^{(enc)}$ and its corresponding memory cell. Given the t-th hidden state $h_t^{(dec)} \in \mathbb{R}^{d_3}$, the decoder predicts the t-th word in the target language using an attention mechanism. The attention mechanism in Luong et al. (2015) computes the weighted average of the hidden states $h_i^{(enc)}$ of the encoder:

$$s(i,t) = \frac{\exp(h_t^{(dec)} \cdot h_i^{(enc)})}{\sum_{j=1}^{N+1} \exp(h_t^{(dec)} \cdot h_j^{(enc)})}, \tag{4}$$

$$a_t = \sum_{i=1}^{N+1} s(i,t) h_i^{(enc)}, \tag{5}$$

where $s(i,t)$ is a scoring function which specifies how much each source-side hidden state contributes to the word prediction.

In addition, like the attention mechanism over constituency tree nodes (Eriguchi et al., 2016b), our model uses attention to the dependency composition vectors:

$$s'(i,t) = \frac{\exp(h_t^{(dec)} \cdot dep(w_i))}{\sum_{j=1}^{N} \exp(h_t^{(dec)} \cdot dep(w_j))}, \tag{6}$$

$$a'_t = \sum_{i=1}^{N} s'(i,t) dep(w_i), \tag{7}$$

To predict the target word, a hidden state $\tilde{h}_t^{(dec)} \in \mathbb{R}^{d_3}$ is then computed as follows:

$$\tilde{h}_t^{(dec)} = \tanh(\tilde{W}[h_t^{(dec)}; a_t; a'_t]), \tag{8}$$

where $\tilde{W} \in \mathbb{R}^{d_3 \times 3d_3}$ is a weight matrix. $\tilde{h}_t^{(dec)}$ is fed into a softmax classifier to predict a target word distribution. $\tilde{h}_t^{(dec)}$ is also used in the transition of the decoder LSTMs along with a word embedding $v_{dec}(w_t) \in \mathbb{R}^{d_3}$ of the target word w_t:

$$h_{t+1}^{(dec)} = \mathrm{LSTM}(h_t^{(dec)}, [v_{dec}(w_t); \tilde{h}_t^{(dec)}]), \tag{9}$$

where the use of $\tilde{h}_t^{(dec)}$ is called *input feeding* proposed by Luong et al. (2015).

The overall model parameters, including those of the latent graph parser, are jointly learned by minimizing the negative log-likelihood of the prediction probabilities of the target words in the training data. To speed up the training, we use BlackOut sampling (Ji et al., 2016). By this joint learning using Equation (3) and (7), the latent graph representations are automatically learned according to the target task.

Implementation Tips Inspired by Zoph et al. (2016), we further speed up BlackOut sampling by sharing noise samples across words in the same sentences. This technique has proven to be effective in RNN language modeling, and we have found that it is also effective in the NMT model. We have also found it effective to share the model parameters of the target word embeddings and the softmax weight matrix for word prediction (Inan et al., 2016; Press and Wolf, 2017). Also, we have found that a parameter averaging technique (Hashimoto et al., 2013) is helpful in improving translation accuracy.

Translation At test time, we use a novel beam search algorithm which combines statistics of sentence lengths (Eriguchi et al., 2016b) and length normalization (Cho et al., 2014). During the beam search step, we use the following scoring function for a generated word sequence $y = (y_1, y_2, \dots, y_{L_y})$ given a source word sequence $x = (x_1, x_2, \dots, x_{L_x})$:

$$\frac{1}{L_y}\left(\sum_{i=1}^{L_y} \log p(y_i|x, y_{1:i-1}) + \log p(L_y|L_x)\right), \tag{10}$$

where $p(L_y|L_x)$ is the probability that sentences of length L_y are generated given source-side sentences of length L_x. The statistics are taken by using the training data in advance. In our experiments, we have empirically found that this beam search algorithm helps the NMT models to avoid generating translation sentences that are too short.

4 Experimental Settings

4.1 Data

We used an English-to-Japanese translation task of the Asian Scientific Paper Excerpt Corpus (ASPEC) (Nakazawa et al., 2016b) used in the Workshop on Asian Translation (WAT), since it has been shown that syntactic information is useful in English-to-Japanese translation (Eriguchi et al., 2016b; Neubig et al., 2015). We followed the data preprocessing instruction for the English-to-Japanese task in Eriguchi et al. (2016b). The English sentences were tokenized by the tokenizer in the Enju parser (Miyao and Tsujii, 2008), and the Japanese sentences were segmented by the KyTea tool[1]. Among the first 1,500,000 translation pairs in the training data, we selected 1,346,946 pairs where the maximum sentence length is 50. In what follows, we call this dataset the *large* training dataset. We further selected the first 20,000 and 100,000 pairs to construct the *small* and *medium* training datasets, respectively. The development data include 1,790 pairs, and the test data 1,812 pairs.

For the small and medium datasets, we built the vocabulary with words whose minimum frequency is two, and for the large dataset, we used words whose minimum frequency is three for English and five for Japanese. As a result, the vocabulary of the target language was 8,593 for the small dataset, 23,532 for the medium dataset, and 65,680 for the large dataset. A special token $\langle$UNK$\rangle$ was used to replace words which were not included in the vocabularies. The character n-grams ($n = 2, 3, 4$) were also constructed from each training dataset with the same frequency settings.

4.2 Parameter Optimization and Translation

We turned hyper-parameters of the model using development data. We set $(d_1, d_2) = (100, 50)$ for the latent graph parser. The word and character n-gram embeddings of the latent graph parser were initialized with the pre-trained embeddings in Hashimoto et al. (2017).[2] The weight matrices in the latent graph parser were initialized with uniform random values in $[-\frac{\sqrt{6}}{\sqrt{row+col}}, +\frac{\sqrt{6}}{\sqrt{row+col}}]$, where row and col are the number of rows and columns of the matrices, respectively. All the bias vectors and the weight matrices in the softmax layers were initialized with zeros, and the bias vectors of the forget gates in the LSTMs were initialized by ones (Jozefowicz et al., 2015).

We set $d_3 = 128$ for the small training dataset, $d_3 = 256$ for the medium training dataset, and $d_3 = 512$ for the large training dataset. The word embeddings and the weight matrices of the NMT model were initialized with uniform random values in $[-0.1, +0.1]$. The training was performed by mini-batch stochastic gradient descent with momentum. For the BlackOut objective (Ji et al., 2016), the number of the negative samples was set to 2,000 for the small and medium training datasets, and 2,500 for the large training dataset. The mini-batch size was set to 128, and the momentum rate was set to 0.75 for the small and medium training datasets and 0.70 for the large training dataset. A gradient clipping technique was used with a clipping value of 1.0. The initial learning rate was set to 1.0, and the learning rate was halved when translation accuracy decreased. We used the BLEU scores obtained by greedy translation as the translation accuracy and checked it at every half epoch of the model training. We saved the model parameters at every half epoch and used the saved model parameters for the parameter averaging technique. For regularization, we used L2-norm regularization with a coefficient of 10^{-6} and applied dropout (Hinton et al., 2012) to Equation (8) with a dropout rate of 0.2.

The beam size for the beam search algorithm was 12 for the small and medium training datasets, and 50 for the large training dataset. We used BLEU (Papineni et al., 2002), RIBES (Isozaki et al., 2010), and perplexity scores as our evaluation metrics. Note that lower perplexity scores indicate better accuracy.

4.3 Pre-Training of Latent Graph Parser

The latent graph parser in our model can be optionally pre-trained by using human annotations for dependency parsing. In this paper we used

[1] http://www.phontron.com/kytea/.

[2] The pre-trained embeddings can be found at https://github.com/hassyGo/charNgram2vec.

the widely-used Wall Street Journal (WSJ) training data to jointly train the POS tagging and dependency parsing components. We used the standard training split (Section 0-18) for POS tagging. We followed Chen and Manning (2014) to generate the training data (Section 2-21) for dependency parsing. From each training dataset, we selected the first K sentences to pre-train our model. The training dataset for POS tagging includes 38,219 sentences, and that for dependency parsing includes 39,832 sentences.

The parser including the POS tagger was first trained for 10 epochs in advance according to the multi-task learning procedure of Hashimoto et al. (2017), and then the overall NMT model was trained. When pre-training the POS tagging and dependency parsing components, we did not apply dropout to the model and did not fine-tune the word and character n-gram embeddings to avoid strong overfitting.

4.4 Model Configurations

LGP-NMT is our proposed model that learns the Latent Graph Parsing for NMT.

LGP-NMT+ is constructed by pre-training the latent parser in LGP-NMT as described in Section 4.3.

SEQ is constructed by removing the dependency composition in Equation (3), forming a sequential NMT model with the multi-layer encoder.

DEP is constructed by using pre-trained dependency relations rather than learning them. That is, $p(H_{w_i} = w_j|w_i)$ is fixed to 1.0 such that w_j is the head of w_i. The dependency labels are also given by the parser which was trained by using all the training samples for parsing and tagging.

UNI is constructed by fixing $p(H_{w_i} = w_j|w_i)$ to $\frac{1}{N}$ for all the words in the same sentence. That is, the uniform probability distributions are used for equally connecting all the words.

5 Results on Small and Medium Datasets

We first show our translation results using the small and medium training datasets. We report averaged scores with standard deviations across five different runs of the model training.

5.1 Small Training Dataset

Table 1 shows the results of using the small training dataset. LGP-NMT performs worse than SEQ and UNI, which shows that the small training dataset is not enough to learn useful latent graph structures from scratch. However, LGP-NMT+ (K = 10,000) outperforms SEQ and UNI, and the standard deviations are the smallest. Therefore, the results suggest that pre-training the parsing and tagging components can improve the translation accuracy of our proposed model. We can also see that DEP performs the worst. This is not surprising because previous studies, e.g., Li et al. (2015), have reported that using syntactic structures do not always outperform competitive sequential models in several NLP tasks.

	BLEU	RIBES	Perplexity
LGP-NMT	14.31±1.49	65.96±1.86	41.13±2.66
LGP-NMT+	16.81±0.31	69.03±0.28	38.33±1.18
SEQ	15.37±1.18	67.01±1.55	38.12±2.52
UNI	15.13±1.67	66.95±1.94	39.25±2.98
DEP	13.34±0.67	64.95±0.75	43.89±1.52

Table 1: Evaluation on the development data using the small training dataset (20,000 pairs).

K	BLEU	RIBES	Perplexity
0	14.31±1.49	65.96±1.86	41.13±2.66
5,000	16.99±1.00	69.03±0.93	37.14±1.96
10,000	16.81±0.31	69.03±0.28	38.33±1.18
All	16.09±0.56	68.19±0.59	39.24±1.88

Table 2: Effects of the size K of the training datasets for POS tagging and dependency parsing.

Now that we have observed the effectiveness of pre-training our model, one question arises naturally:

> how many training samples for parsing and tagging are necessary for improving the translation accuracy?

Table 2 shows the results of using different numbers of training samples for parsing and tagging. The results of K= 0 and K= 10,000 correspond to those of LGP-NMT and LGP-NMT+ in Table 1, respectively. We can see that using the small amount of the training samples performs better than using all the training samples.[3] One possible reason is that the domains of the translation dataset and the parsing (tagging) dataset are considerably different. The parsing and tagging datasets come from WSJ, whereas the translation dataset comes from abstract text of scientific papers in a wide range of domains, such as

[3] We did not observe such significant difference when using the larger datasets, and we used all the training samples in the remaining part of this paper.

	BLEU	RIBES	Perplexity
LGP-NMT	28.70±0.27	77.51±0.13	12.10±0.16
LGP-NMT+	29.06±0.25	77.57±0.24	12.09±0.27
SEQ	28.60±0.24	77.39±0.15	12.15±0.12
UNI	28.25±0.35	77.13±0.20	12.37±0.08
DEP	26.83±0.38	76.05±0.22	13.33±0.23

Table 3: Evaluation on the development data using the medium training dataset (100,000 pairs).

biomedicine and computer science. These results suggest that our model can be improved by a small amount of parsing and tagging datasets in different domains. Considering the recent universal dependency project[4] which covers more than 50 languages, our model has the potential of being applied to a variety of language pairs.

5.2 Medium Training Dataset

Table 3 shows the results of using the medium training dataset. In contrast with using the small training dataset, LGP-NMT is slightly better than SEQ. LGP-NMT significantly outperforms UNI, which shows that our adaptive learning is more effective than using the uniform graph weights. By pre-training our model, LGP-NMT+ significantly outperforms SEQ in terms of the BLEU score. Again, DEP performs the worst among all the models.

By using our beam search strategy, the Brevity Penalty (BP) values of our translation results are equal to or close to 1.0, which is important when evaluating the translation results using the BLEU scores. A BP value ranges from 0.0 to 1.0, and larger values mean that the translated sentences have relevant lengths compared with the reference translations. As a result, our BLEU evaluation results are affected only by the word n-gram precision scores. BLEU scores are sensitive to the BP values, and thus our beam search strategy leads to more solid evaluation for NMT models.

6 Results on Large Dataset

Table 4 shows the BLEU and RIBES scores on the development data achieved with the large training dataset. Here we focus on our models and SEQ because UNI and DEP consistently perform worse than the other models as shown in Table 1 and 3. The averaging technique and attention-based unknown word replacement (Jean et al., 2015; Hashimoto et al., 2016) improve the scores. Again, we see that the translation scores of our model can be further improved by pre-training the model.

B./R.	Single	+Averaging	+UnkRep
LGP-NMT	38.05/81.98	38.44/82.23	38.77/82.29
LGP-NMT+	38.75/82.13	39.01/82.40	39.37/82.48
SEQ	38.24/81.84	38.26/82.14	38.61/82.18

Table 4: BLEU (B.) and RIBES (R.) scores on the development data using the large training dataset.

	BLEU	RIBES
LGP-NMT	39.19	82.66
LGP-NMT+	39.42	82.83
SEQ	38.96	82.18
Ensemble of the above three models	41.18	83.40
Cromieres et al. (2016)	38.20	82.39
Neubig et al. (2015)	38.17	81.38
Eriguchi et al. (2016a)	36.95	82.45
Neubig and Duh (2014)	36.58	79.65
Zhu (2015)	36.21	80.91
Lee et al. (2015)	35.75	81.15

Table 5: BLEU and RIBES scores on the test data.

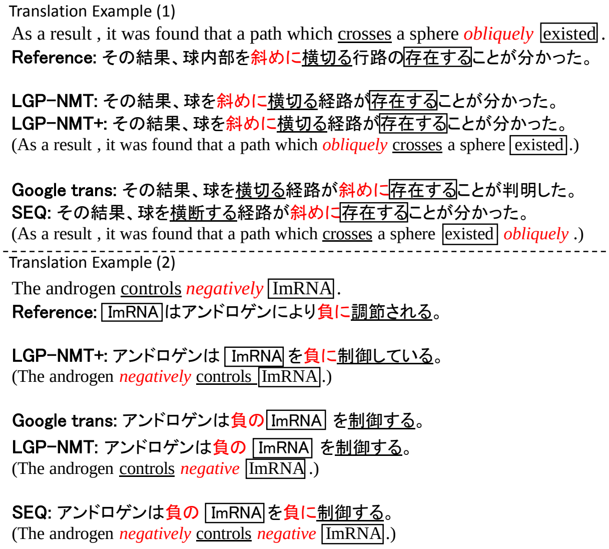
Table 5 shows our results on the test data, and the previous best results summarized in Nakazawa et al. (2016a) and the WAT website[5] are also shown. Our proposed models, LGP-NMT and LGP-NMT+, outperform not only SEQ but also all of the previous best results. Notice also that our implementation of the sequential model (SEQ) provides a very strong baseline, the performance of which is already comparable to the previous state of the art, even without using ensemble techniques. The confidence interval ($p \leq 0.05$) of the RIBES score of LGP-NMT+ estimated by bootstrap resampling (Noreen, 1989) is $(82.27, 83.37)$, and thus the RIBES score of LGP-NMT+ is significantly better than that of SEQ, which shows that our latent parser can be effectively pre-trained with the human-annotated treebank.

The sequential NMT model in Cromieres et al. (2016) and the tree-to-sequence NMT model in Eriguchi et al. (2016b) rely on ensemble techniques while our results mentioned above are obtained using single models. Moreover, our model is more compact[6] than the previous best NMT model in Cromieres et al. (2016). By applying the ensemble technique to LGP-NMT, LGP-NMT+,

[4] http://universaldependencies.org/.

[5] http://lotus.kuee.kyoto-u.ac.jp/WAT/evaluation/list.php?t=1&o=1.

[6] Our training time is within five days on a c4.8xlarge machine of Amazon Web Service by our CPU-based C++ code, while it is reported that the training time is more than two weeks in Cromieres et al. (2016) by their GPU code.



Translation Example (1)

As a result , it was found that a path which crosses a sphere *obliquely* existed .

Reference: その結果、球内部を斜めに横切る行路の存在することが分かった。

LGP-NMT: その結果、球を斜めに横切る経路が存在することが分かった。

LGP-NMT+: その結果、球を斜めに横切る経路が存在することが分かった。

(As a result , it was found that a path which *obliquely* crosses a sphere existed.)

Google trans: その結果、球を横切る経路が斜めに存在することが判明した。

SEQ: その結果、球を横断する経路が斜めに存在することが分かった。

(As a result , it was found that a path which crosses a sphere existed *obliquely* .)

Translation Example (2)

The androgen controls *negatively* ImRNA.

Reference: ImRNAはアンドロゲンにより負に調節される。

LGP-NMT+: アンドロゲンは ImRNA を負に制御している。

(The androgen *negatively* controls ImRNA.)

Google trans: アンドロゲンは負のImRNA を制御する。

LGP-NMT: アンドロゲンは負の ImRNA を制御する。

(The androgen controls *negative* ImRNA.)

SEQ: アンドロゲンは負の ImRNAを負に制御する。

(The androgen *negatively* controls *negative* ImRNA.)

Figure 2: English-to-Japanese translation examples for focusing on the usage of adverbs.

and SEQ, the BLEU and RIBES scores are further improved, and both of the scores are significantly better than the previous best scores.

6.1 Analysis on Translation Examples

Figure 2 shows two translation examples[7] to see how the proposed model works and what is missing in the state-of-the-art sequential NMT model, SEQ. Besides the reference translation, the outputs of our models with and without pre-training, SEQ, and Google Translation[8] are shown.

Selectional Preference In the translation example (1) in Figure 2, we see that the adverb "obliquely" is interpreted differently across the systems. As in the reference translation, "obliquely" is a modifier of the verb "crosses". Our models correctly capture the relationship between the two words, whereas Google Translation and SEQ treat "obliquely" as a modifier of the verb "existed". This error is not a surprise since the verb "existed" is located closer to "obliquely" than the verb "crosses". A possible reason for the correct interpretation by our models is that they can better capture long-distance dependencies and are less susceptible to surface word distances. This is an indication of our models' ability of capturing domain-specific selectional preference that cannot be captured by purely sequential models. It should be noted that simply using standard treebank-based parsers does not necessarily address this error, because our pre-trained dependency parser interprets that "obliquely" is a modifier of the verb "existed".

Adverb or Adjective The translation example (2) in Figure 2 shows another example where the adverb "negatively" is interpreted as an adverb or an adjective. As in the reference translation, "negatively" is a modifier of the verb "controls". Only LGP-NMT+ correctly captures the adverb-verb relationship, whereas "negatively" is interpreted as the adjective "negative" to modify the noun "ImRNA" in the translation results from Google Translation and LGP-NMT. SEQ interprets "negatively" as both an adverb and an adjective, which leads to the repeated translations. This error suggests that the state-of-the-art NMT models are strongly affected by the word order. By contrast, the pre-training strategy effectively embeds the information about the POS tags and the dependency relations into our model.

6.2 Analysis on Learned Latent Graphs

Without Pre-Training We inspected the latent graphs learned by LGP-NMT. Figure 1 shows an example of the learned latent graph obtained for a sentence taken from the development data of the translation task. It has long-range dependencies and cycles as well as ordinary left-to-right dependencies. We have observed that the punctuation mark "." is often pointed to by other words with large weights. This is primarily because the hidden state corresponding to the mark in each sentence has rich information about the sentence.

To measure the correlation between the latent graphs and human-defined dependencies, we parsed the sentences on the development data of the WSJ corpus and converted the graphs into dependency trees by Eisner's algorithm (Eisner, 1996). For evaluation, we followed Chen and Manning (2014) and measured Unlabeled Attachment Score (UAS). The UAS is 24.52%, which shows that the implicitly-learned latent graphs are partially consistent with the human-defined syntactic structures. Similar trends have been reported by Yogatama et al. (2017) in the case of binary constituency parsing. We checked the most dominant gold dependency labels which were assigned for the dependencies detected by LGP-NMT. The labels whose ratio is more than 3% are

[7] These English sentences were created by manual simplification of sentences in the development data.

[8] The translations were obtained at https://translate.google.com in Feb. and Mar. 2017.

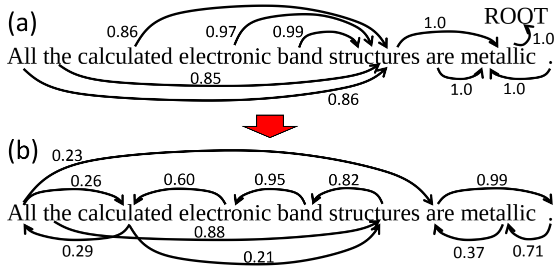

(a)
0.86
0.97
0.99
1.0
ROOT
1.0
All the calculated electronic band structures are metallic .
0.85
0.86
1.0
1.0
(b)
0.23
0.26
0.60
0.95
0.82
0.99
All the calculated electronic band structures are metallic .
0.88
0.29
0.21
0.37
0.71

Figure 3: An example of the pre-trained dependency structures (a) and its corresponding latent graph adapted by our model (b).

nn, amod, prep, pobj, dobj, nsubj, num, det, advmod, and poss. We see that dependencies between words in distant positions, such as subject-verb-object relations, can be captured.

With Pre-Training We also inspected the pre-trained latent graphs. Figure 3-(a) shows the dependency structure output by the pre-trained latent parser for the same sentence in Figure 1. This is an ordinary dependency tree, and the head selection is almost deterministic; that is, for each word, the largest weight of the head selection is close to 1.0. By contrast, the weight values are more evenly distributed in the case of LGP-NMT as shown in Figure 1. After the overall NMT model training, the latent parser is adapted to the translation task, and Figure 3-(b) shows the adapted latent graph. Again, we can see that the adapted weight values are also distributed and different from the original pre-trained weight values, which suggests that human-defined syntax is not always optimal for the target task.

The UAS of the pre-trained dependency trees is 92.52%[9], and that of the adapted latent graphs is 18.94%. Surprisingly, the resulting UAS (18.94%) is lower than the UAS of our model without pre-training (24.52%). However, in terms of the translation accuracy, our model with pre-training is better than that without pre-training. These results suggest that human-annotated treebanks can provide useful prior knowledge to guide the overall model training by pre-training, but the resulting sentence structures adapted to the target task do not need to highly correlate with the treebanks.

7 Related Work

While initial studies on NMT treat each sentence as a sequence of words (Bahdanau et al., 2015; Luong et al., 2015; Sutskever et al., 2014), researchers have recently started investigating into the use of syntactic structures in NMT models (Bastings et al., 2017; Chen et al., 2017; Eriguchi et al., 2016a,b, 2017; Li et al., 2017; Sennrich and Haddow, 2016; Stahlberg et al., 2016; Yang et al., 2017). In particular, Eriguchi et al. (2016b) introduced a tree-to-sequence NMT model by building a tree-structured encoder on top of a standard sequential encoder, which motivated the use of the dependency composition vectors in our proposed model. Prior to the advent of NMT, the syntactic structures had been successfully used in statistical machine translation systems (Neubig and Duh, 2014; Yamada and Knight, 2001). These syntax-based approaches are pipelined; a syntactic parser is first trained by supervised learning using a treebank such as the WSJ dataset, and then the parser is used to automatically extract syntactic information for machine translation. They rely on the output from the parser, and therefore parsing errors are propagated through the whole systems. By contrast, our model allows the parser to be adapted to the translation task, thereby providing a first step towards addressing ambiguous syntactic and semantic problems, such as domain-specific selectional preference and PP attachments, in a task-oriented fashion.

Our model learns latent graph structures in a source-side language. Eriguchi et al. (2017) have proposed a model which learns to parse and translate by using automatically-parsed data. Thus, it is also an interesting direction to learn latent structures in a target-side language.

As for the learning of latent syntactic structure, there are several studies on learning task-oriented syntactic structures. Yogatama et al. (2017) used a reinforcement learning method on shift-reduce action sequences to learn task-oriented binary constituency trees. They have shown that the learned trees do not necessarily highly correlate with the human-annotated treebanks, which is consistent with our experimental results. Socher et al. (2011) used a recursive autoencoder model to greedily construct a binary constituency tree for each sentence. The autoencoder objective works as a regularization term for sentiment classification tasks. Prior to these deep learning approaches,

[9] The UAS is significantly lower than the reported score in Hashimoto et al. (2017). The reason is described in Section 4.3.

Wu (1997) presented a method for *bilingual parsing*. One of the characteristics of our model is directly using the soft connections of the graph edges with the real-valued weights, whereas all of the above-mentioned methods use one best structure for each sentence. Our model is based on dependency structures, and it is a promising future direction to jointly learn dependency and constituency structures in a task-oriented fashion.

Finally, more related to our model, Kim et al. (2017) applied their *structured attention networks* to a Natural Language Inference (NLI) task for learning dependency-like structures. They showed that pre-training their model by a parsing dataset did not improve accuracy on the NLI task. By contrast, our experiments show that such a parsing dataset can be effectively used to improve translation accuracy by varying the size of the dataset and by avoiding strong overfitting. Moreover, our translation examples show the concrete benefit of learning task-oriented latent graph structures.

8 Conclusion and Future Work

We have presented an end-to-end NMT model by jointly learning translation and source-side latent graph representations. By pre-training our model using treebank annotations, our model significantly outperforms both a pipelined syntax-based model and a state-of-the-art sequential model. On English-to-Japanese translation, our model outperforms the previous best models by a large margin. In future work, we investigate the effectiveness of our approach in different types of target tasks.

Acknowledgments

We thank the anonymous reviewers and Akiko Eriguchi for their helpful comments and suggestions. We also thank Yuchen Qiao and Kenjiro Taura for their help in speeding up our training code. This work was supported by CREST, JST, and JSPS KAKENHI Grant Number 17J09620.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. *arXiv*, cs.CL 1704.04675.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Danqi Chen and Christopher Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 740–750.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved Neural Machine Translation with a Syntax-Aware Encoder and Decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. To appear.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Fabien Cromieres, Chenhui Chu, Toshiaki Nakazawa, and Sadao Kurohashi. 2016. Kyoto University Participation to WAT 2016. In *Proceedings of the 3rd Workshop on Asian Translation*, pages 166–174.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of the 5th International Conference on Learning Representations*.

Jason Eisner. 1996. Efficient Normal-Form Parsing for Combinatory Categorial Grammar. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 79–86.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016a. Character-based Decoding in Tree-to-Sequence Attention-based Neural Machine Translation. In *Proceedings of the 3rd Workshop on Asian Translation*, pages 175–183.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016b. Tree-to-Sequence Attentional Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to Parse and Translate Improves Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. To appear.

Alex Graves and Jurgen Schmidhuber. 2005. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks*, 18(5):602–610.

Kazuma Hashimoto, Akiko Eriguchi, and Yoshimasa Tsuruoka. 2016. Domain Adaptation and Attention-Based Unknown Word Replacement in Chinese-to-Japanese Neural Machine Translation. In *Proceedings of the 3rd Workshop on Asian Translation*, pages 75–83.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple Customization of Recursive Neural Networks for Semantic Relation Classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. To appear.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. *arXiv*, cs.CL 1611.01462.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic Evaluation of Translation Quality for Distant Language Pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal Neural Machine Translation Systems for WMTf15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140.

Shihao Ji, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, and Pradeep Dubey. 2016. BlackOut: Speeding up Recurrent Neural Network Language Models With Very Large Vocabularies. In *Proceedings of the 4th International Conference on Learning Representations*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An Empirical Exploration of Recurrent Network Architectures. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2342–2350.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of the 5th International Conference on Learning Representations*.

Hyoung-Gyu Lee, JaeSong Lee, Jun-Seok Kim, and Chang-Ki Lee. 2015. NAVER Machine Translation System for WAT 2015. In *Proceedings of the 2nd Workshop on Asian Translation*, pages 69–73.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When Are Tree Structures Necessary for Deep Learning of Representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2304–2314.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling Source Syntax for Neural Machine Translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. To appear.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online Large-Margin Training of Dependency Parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 91–98.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature Forest Models for Probabilistic HPSG Parsing. *Computational Linguistics*, 34(1):35–80.

Toshiaki Nakazawa, Hideya Mino, Chenchen Ding, Isao Goto, Graham Neubig, Sadao Kurohashi, and Eiichiro Sumita. 2016a. Overview of the 3rd Workshop on Asian Translation. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016b. ASPEC: Asian Scientific Paper Excerpt Corpus. In *Proceedings of the 10th Conference on International Language Resources and Evaluation*.

Graham Neubig and Kevin Duh. 2014. On the Elements of an Accurate Tree-to-String Machine Translation System. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 143–149.

Graham Neubig, Makoto Morishita, and Satoshi Nakamura. 2015. Neural Reranking Improves Subjective Quality of Machine Translation: NAIST at WAT2015. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 35–41.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley-Interscience.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings* *of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318.

Ofir Press and Lior Wolf. 2017. Using the Output Embedding to Improve Language Models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163.

Rico Sennrich and Barry Haddow. 2016. Linguistic Input Features Improve Neural Machine Translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016. Syntactically Guided Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 299–305.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding Words and Sentences via Character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515.

Dekai Wu. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377–404.

Kenji Yamada and Kevin Knight. 2001. A Syntax-based Statistical Translation Model. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 523–530.

Baosong Yang, Derek F. Wong, Tong Xiao, Lidia S. Chao, and Jingbo Zhu. 2017. Towards Bidirectional Hierarchical Representations for Attention-Based Neural Machine Translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. To appear.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2017. Learning to Compose Words into Sentences with Reinforcement Learning. In *Proceedings of the 5th International Conference on Learning Representations*.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency Parsing as Head Selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 665–676.

Zhongyuan Zhu. 2015. Evaluating Neural Machine Translation in English-Japanese Task. In *Proceedings of the 2nd Workshop on Asian Translation*, pages 61–68.

Barret Zoph, Ashish Vaswani, Jonathan May, and Kevin Knight. 2016. Simple, Fast Noise-Contrastive Estimation for Large RNN Vocabularies. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1217–1222.

Neural Machine Translation with Word Predictions

Rongxiang Weng, Shujian Huang, Zaixiang Zheng, Xinyu Dai and **Jiajun Chen**
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing 210023, China
{wengrx, huangsj, zhengzx, daixy, chenjj}@nlp.nju.edu.cn

Abstract

In the encoder-decoder architecture for neural machine translation (NMT), the hidden states of the recurrent structures in the encoder and decoder carry the crucial information about the sentence.These vectors are generated by parameters which are updated by back-propagation of translation errors through time. We argue that propagating errors through the end-to-end recurrent structures are not a direct way of control the hidden vectors. In this paper, we propose to use word predictions as a mechanism for direct supervision. More specifically, we require these vectors to be able to predict the vocabulary in target sentence. Our simple mechanism ensures better representations in the encoder and decoder without using any extra data or annotation. It is also helpful in reducing the target side vocabulary and improving the decoding efficiency. Experiments on Chinese-English and German-English machine translation tasks show BLEU improvements by 4.53 and 1.3, respectively.

1 Introduction

The encoder-decoder based neural machine translation (NMT) models (Sutskever et al., 2014; Cho et al., 2014) have been developing rapidly. Sutskever et al. (2014) propose to encode the source sentence as a fixed-length vector representation, based on which the decoder generates the target sequence, where both the encoder and decoder are recurrent neural networks (RNN) (Sutskever et al., 2014) or their variants (Cho et al., 2014; Chung et al., 2014; Bahdanau et al., 2014). In this framework, the fixed-length vector plays the crucial role of transitioning the information of the sentence from the source side to the target side.

Later, attention mechanisms are proposed to enhance the source side representations (Bahdanau et al., 2014; Luong et al., 2015b). The source side context is computed at each time-step of decoding, based on the attention weights between the source side representations and the current hidden state of the decoder. However, the hidden states in the recurrent decoder still originate from the single fixed-length representation (Luong et al., 2015b), or the average of the bi-directional representations (Bahdanau et al., 2014). Here we refer to the representation as *initial state*.

Interestingly, Britz et al. (2017) find that the value of initial state does not affect the translation performance, and prefer to set the initial state to be a zero vector. On the contrary, we argue that initial state still plays an important role of translation, which is currently neglected. We notice that beside the end-to-end error back propagation for the initial and transition parameters, there is no direct control of the initial state in the current NMT architectures. Due to the large number of parameters, it may be difficult for the NMT system to learn the proper sentence representation as the initial state. Thus, the model is very likely to get stuck in local minimums, making the translation process arbitrary and unstable.

In this paper, we propose to augment the current NMT architecture with a word prediction mechanism. More specifically, we require the initial state of the decoder to be able to predict all the words in the target sentence. In this way, there is a specific objective for learning the initial state. Thus the learnt source side representation will be better constrained. We further extend this idea by applying the word predictions mechanism to all the hidden states of the decoder. So the transition between different decoder states could be controlled

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 136–145
Copenhagen, Denmark, September 7–11, 2017.

as well.

Our mechanism is simple and requires no additional data or annotation. The proposed word predictions mechanism could be used as a training method and brings no extra computing cost during decoding.

Experiments on the Chinese-English and German-English translation tasks show that both the constraining of the initial state and the decoder hidden states bring significant improvement over the baseline systems. Furthermore, using the word prediction mechanism on the initial state as a word predictor to reduce the target side vocabulary could greatly improve the decoding efficiency, without a significant loss on the translation quality.

2 Related Work

Many previous works have noticed the problem of training an NMT system with lots of parameters. Some of them prefer to use the dropout technique (Srivastava et al., 2014; Luong et al., 2015b; Meng et al., 2016). Another possible choice is to ensemble several models with random starting points (Sutskever et al., 2014; Jean et al., 2015; Luong and Manning, 2016). Both techniques could bring more stable and better results. But they are general training techniques of neural networks, which are not specifically targeting the modeling of the translation process like ours. We will make empirical comparison with them in the experiments.

The way we add the word prediction is similar to the research of multi-task learning. Dong et al. (2015) propose to share an encoder between different translation tasks. Luong et al. (2015a) propose to jointly learn the translation task for different languages, the parsing task and the image captioning task, with a shared encoder or decoder. Zhang and Zong (2016) propose to use multitask learning for incorporating source side monolingual data. Different from these attempts, our method focuses solely on the current translation task, and does not require any extra data or annotation.

In the other sequence to sequence tasks, Suzuki and Nagata (2017) propose the idea for predicting words by using encoder information. However, the purpose and the way of our mechanism are different from them.

The word prediction technique has been applied in the research of both statistical machine translation (SMT) (Bangalore et al., 2007; Mauser et al., 2009; Jeong et al., 2010; Tran et al., 2014) and NMT (Mi et al., 2016; L'Hostis et al., 2016). In these research, word prediction mechanisms are employed to decide the selection of words or constrain the target vocabulary, while in this paper, we use word prediction as a control mechanism for neural model training.

3 Notations and Backgrounds

We present a popular NMT framework with the encoder-decoder architecture (Cho et al., 2014; Bahdanau et al., 2014) and the attention networks (Luong et al., 2015b), based on which we propose our word prediction mechanism.

Denote a source-target sentence pair as $\{\mathbf{x}, \mathbf{y}\}$ from the training set, where $\mathbf{x}$ is the source word sequence $(x_1, x_2, \cdots, x_{|\mathbf{x}|})$ and $\mathbf{y}$ is the target word sequence $(y_1, y_2, \cdots, y_{|\mathbf{y}|})$, $|\mathbf{x}|$ and $|\mathbf{y}|$ are the length of $\mathbf{x}$ and $\mathbf{y}$, respectively.

In the encoding stage, a bi-directional recurrent neural network is used (Bahdanau et al., 2014) to encode $\mathbf{x}$ into a sequence of vectors $(\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{|\mathbf{x}|})$. For each x_i, the representation $\mathbf{h}_i$ is:

$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i] \tag{1}$$

where $[\cdot;\cdot]$ denotes the concatenation of column vectors; $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ denote the hidden vectors for the word x_i in the forward and backward RNNs, respectively.

The gated recurrent unit (GRU) is used as the recurrent unit in each RNN, which is shown to have promising results in speech recognition and machine translation (Cho et al., 2014). Formally, the hidden state $\mathbf{h}_i$ at time step i of the forward RNN encoder is defined by the GRU function $g_{\overrightarrow{e}}(\cdot,\cdot)$, as follows:

$$\begin{aligned}\overrightarrow{\mathbf{h}}_i &= g_{\overrightarrow{e}}(\overrightarrow{\mathbf{h}}_{i-1}, \mathbf{emb}_{x_i}) \\ &= (\mathbf{1} - \overrightarrow{\mathbf{z}}_i) \odot \overrightarrow{\mathbf{h}}_{i-1} + \overrightarrow{\mathbf{z}}_i \odot \overrightarrow{\mathbf{h}'}_i\end{aligned} \tag{2}$$

$$\overrightarrow{\mathbf{z}}_i = \sigma(\overrightarrow{\mathbf{W}}_z[\mathbf{emb}_{x_i}; \overrightarrow{\mathbf{h}}_{i-1}]) \tag{3}$$

$$\overrightarrow{\mathbf{h}'}_i = \tanh(\overrightarrow{\mathbf{W}}[\mathbf{emb}_{x_i}; (\overrightarrow{\mathbf{r}}_i \odot \overrightarrow{\mathbf{h}}_{i-1})]) \tag{4}$$

$$\overrightarrow{\mathbf{r}}_i = \sigma(\overrightarrow{\mathbf{W}}_r[\mathbf{emb}_{x_i}; \overrightarrow{\mathbf{h}}_{i-1}]) \tag{5}$$

where $\odot$ denotes element-wise product between vectors and $\mathbf{emb}_{x_i}$ is the word embedding of the x_i. $\tanh(\cdot)$ and $\sigma(\cdot)$ are the tanh and sigmoid transformation functions that can be applied element-wise on vectors, respectively. For simplicity, we

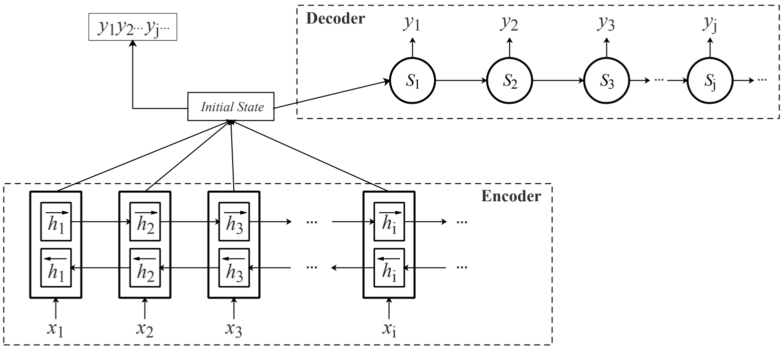


Figure 1: The NMT model with word prediction for the initial state.

omit the bias term in each network layer. The backward RNN encoder is defined likewise.

In the decoding stage, the decoder starts with the initial state $\mathbf{s}_0$, which is the average of source representations (Bahdanau et al., 2014).

$$\mathbf{s}_0 = \sigma(\mathbf{W}_s \frac{1}{|\mathbf{x}|} \sum_{i=1}^{|\mathbf{x}|} \mathbf{h}_i) \tag{6}$$

At each time step j, the decoder maximizes the conditional probability of generating the jth target word, which is defined as:

$$P(y_j|y_{<j}, \mathbf{x}) = f_d(t_d([\mathbf{emb}_{y_{j-1}}; \mathbf{s}_j; \mathbf{c}_j])) \tag{7}$$

$$f_d(\mathbf{u}) = \text{softmax}(\mathbf{W}_f \mathbf{u}) \tag{8}$$

$$t_d(\mathbf{v}) = \tanh(\mathbf{W}_t \mathbf{v}) \tag{9}$$

where $\mathbf{s}_j$ is the decoder's hidden state, which is computed by another GRU (as in Equation 2):

$$\mathbf{s}_j = g_d(\mathbf{s}_{j-1}, [\mathbf{emb}_{y_{j-1}}; \mathbf{c}_j]) \tag{10}$$

and the context vector $\mathbf{c}_j$ is from the attention mechanism (Luong et al., 2015b):

$$\mathbf{c}_j = \sum_{i=1}^{|\mathbf{x}|} a_{ji} \mathbf{h}_i \tag{11}$$

$$a_{ji} = \frac{\exp(e_{ji})}{\sum_{k=1}^{|\mathbf{x}|} \exp(e_{jk})} \tag{12}$$

$$e_{ji} = \tanh(\mathbf{W}_{att_d}[\mathbf{s}_{j-1}; \mathbf{h}_i]). \tag{13}$$

4 NMT with Word Predictions

4.1 Word Prediction for the Initial State

The decoder starts the generation of target sentence from the initial state $\mathbf{s}_0$ (Equation 6) generated by the encoder. Currently, the update for the encoder only happens when a translation error occurs in the decoder. The error is propagated through multiple time steps in the recurrent structure until it reaches the encoder. As there are hundreds of millions of parameters in the NMT system, it is hard for the model to learn the exact representation of source sentences. As a result, the values of initial state may not be exact during the translation process, leading to poor translation performances.

We propose word prediction as a mechanism to control the values of initial state. The intuition is that since the initial state is responsible for the translation of whole target sentence, it should at least contain information of each word in the target sentence. Thus, we optimize the initial state by making prediction for all target words. For simplicity, we assume each target word is independent of each other.

Here the word prediction mechanism is a simpler sub-task of translation, where the order of words is not considered. The prediction task could be trained jointly with the translation task in a multi-task learning way (Luong et al., 2015a; Dong et al., 2015; Zhang and Zong, 2016), where both tasks share the same encoder. In other words, word prediction for the initial state could be interpreted as an improvement for the encoder. We denote this mechanism as WP_E .

As shown in Figure 1, a prediction network is added to the initial state. We define the conditional probability of WP_E as follows:

$$P_{\text{WP}_\text{E}}(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^{|\mathbf{y}|} P_{\text{WP}_\text{E}}(y_j|\mathbf{x}) \tag{14}$$

$$P_{\text{WP}_\text{E}}(y_j|\mathbf{x}) = f_p(t_p([\mathbf{s}_0; \mathbf{c}_p])) \tag{15}$$

where $f_p(\cdot)$ and $t_p(\cdot)$ are the softmax layer and non-linear layer as defined in Equation 8-9, with different parameters; $\mathbf{c}_p$ is defined similar as the

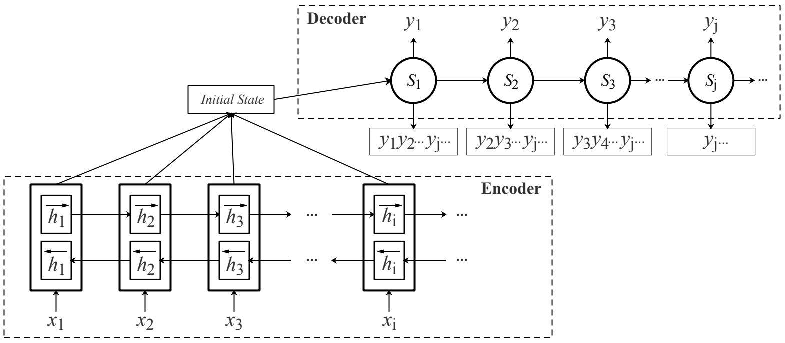


Figure 2: The NMT model with word predictions for the decoder's hidden states.

attention network, so the source side information could be enhanced.

$$\mathbf{c}_p = \sum_{i=1}^{|\mathbf{x}|} a_i \mathbf{h}_i \tag{16}$$

$$a_i = \frac{\exp(e_i)}{\sum_{k=1}^{|\mathbf{x}|} \exp(e_k)} \tag{17}$$

$$e_i = \tanh(\mathbf{W}_{att_p}[\mathbf{s}_0, \mathbf{h}_i]). \tag{18}$$

4.2 Word Predictions for Decoder's Hidden States

Similar intuition is also applied for the decoder. Because the hidden states of the decoder are responsible for the translation of target words, they should be able to predict the target words as well. The only difference is that we remove the already generated words from the prediction task. So each hidden state in the decoder is required to predict the target words which remain untranslated.

For the first state $\mathbf{s}_1$ of the decoder, the prediction task is similar with the task for the initial state. Since then, the prediction is no longer a separate training task, but integrated into each time step of the training process. We denote this mechanism as $\mathrm{WP_D}$.

As shown in Figure 2, for each time step j in the decoder, the hidden state $\mathbf{s}_j$ is used for the prediction of $(y_j, y_{j+1}, \cdots, y_{|\mathbf{y}|})$. The conditional probability of $\mathrm{WP_D}$ is defined as:

$$\begin{aligned} P_{\mathrm{WP_D}}(y_j, y_{j+1}, \cdots, y_{|\mathbf{y}|} | y_{<j}, \mathbf{x}) \\ = \prod_{k=j}^{|\mathbf{y}|} P_{\mathrm{WP_D}}(y_k | y_{<j}, \mathbf{x}) \end{aligned} \tag{19}$$

$$P_{\mathrm{WP_D}}(y_k | y_{<j}, \mathbf{x}) = f_d(p(t_d([\mathbf{emb}_{y_{j-1}}; \mathbf{s}_j; \mathbf{c}_j]))) \tag{20}$$

where $f_d(\cdot)$ and $t_d(\cdot)$ are the softmax layer and non-linear layer as defined in Equation 8-9; $p(\cdot)$ is another non-linear transformation layer, which prepares the current state for the prediction:

$$p(\mathbf{u}) = \tanh(\mathbf{W}_p \mathbf{u}). \tag{21}$$

4.3 Training

NMT models optimize the networks by maximizing the likelihood of the target translation $\mathbf{y}$ given source sentence $\mathbf{x}$, denoted by L_T.

$$L_\mathrm{T} = \frac{1}{|\mathbf{y}|} \sum_{j=1}^{|\mathbf{y}|} \log P(y_j | y_{<j}, \mathbf{x}) \tag{22}$$

where $P(y_j | y_{<j}, \mathbf{x})$ is defined in Equation 7.

To optimize the word prediction mechanism, we propose to add extra likelihood functions $L_{\mathrm{WP_E}}$ and $L_{\mathrm{WP_D}}$ into the training procedure.

For the $\mathrm{WP_E}$, we directly optimize the likelihood of translation and word prediction:

$$L_1 = L_\mathrm{T} + L_{\mathrm{WP_E}} \tag{23}$$

$$L_{\mathrm{WP_E}} = \log P_{\mathrm{WP_E}} \tag{24}$$

where $P_{\mathrm{WP_E}}$ is defined in Equation 14.

For the $\mathrm{WP_D}$, we optimize the likelihood as:

$$L_2 = L_\mathrm{T} + L_{\mathrm{WP_D}} \tag{25}$$

$$L_{\mathrm{WP_D}} = \sum_{j=1}^{|\mathbf{y}|} \frac{1}{|\mathbf{y}| - j + 1} \log P_{\mathrm{WP_D}} \tag{26}$$

where $P_{\mathrm{WP_D}}$ is defined in Equation 19; the coefficient of the logarithm is used to calculate the average probability of each prediction.

The two mechanisms could also work together, so that both the encoder and the decoder could be improved:

$$L_3 = L_\mathrm{T} + L_{\mathrm{WP_E}} + L_{\mathrm{WP_D}}. \tag{27}$$

4.4 Making Use of the Word Predictor

The previously proposed word prediction mechanism could be used only as a extra training objective, which will not be computed during the translation. Thus the computational complexity of our models for translation stays exactly the same.

On the other hand, using a smaller and specific vocabulary for each sentence or batch will improve translation efficiency. If the vocabulary is accurate enough, there is also a chance to improve the translation quality (Jean et al., 2015; Mi et al., 2016; L'Hostis et al., 2016). Our word prediction mechanism $\mathrm{WP_E}$ provides a natural solution for generating a possible set of target words at sentence level. The prediction could be made from the initial state $\mathbf{s}_0$, without using extra resources such as word dictionaries, extracted phrases or frequent word lists, as in Mi et al. (2016).

5 Experiments

5.1 Data

We perform experiments on the Chinese-English (CH-EN) and German-English (DE-EN) machine translation tasks. For the CH-EN, the training data consists of about 8 million sentence pairs [1]. We use NIST MT02 as our validation set, and the NIST MT03, MT04 and MT05 as our test sets. These sets have 878, 919, 1597 and 1082 source sentences, respectively, with 4 references for each sentence. For the DE-EN, the experiments trained on the standard benchmark WMT14, and it has about 4.5 million sentence pairs. We use newstest 2013 (NST13) as validation set, and newstest 2014(NST14) as test set. These sets have 3000 and 2737 source sentences, respectively, with 1 reference for each sentence. Sentences were encoded using byte-pair encoding (BPE) (Britz et al., 2017).

5.2 Systems and Techniques

We implement a baseline system with the bi-directional encoder (Bahdanau et al., 2014) and the attention mechanism (Luong et al., 2015b) as described in Section 3, denoted as baseNMT. Then our proposed word prediction mechanism on initial state and hidden states of decoder are implemented on the baseNMT system, denoted as $\mathrm{WP_E}$ and $\mathrm{WP_D}$, respectively. We denote the system use both techniques as $\mathrm{WP_{ED}}$. We implement systems with variable-sized vocabulary following (Mi et al., 2016). For comparison, we also implement systems with dropout (with dropout rate 0.5 on the output layer) and ensemble (ensemble of 4 systems at the output layer) techniques.

5.3 Implementation Details

Both our CH-EN and DE-EN experiments are implemented on the open source toolkit dl4mt [2], with most default parameter settings kept the same. We train the NMT systems with the sentences of length up to 50 words. The source and target vocabularies are limited to the most frequent 30K words for both Chinese and English, respectively, with the out-of-vocabulary words mapped to a special token UNK.

The dimension of word embedding is set to 512 and the size of the hidden layer is 1024. The recurrent weight matrices are initialized as random orthogonal matrices, and all the bias vectors as zero. Other parameters are initialized by sampling from the Gaussian distribution $\mathcal{N}(0, 0.01)$.

We use the mini-batch stochastic gradient descent (SGD) approach to update the parameters, with a batch size of 32. The learning rate is controlled by AdaDelta (Zeiler, 2012).

For efficient training of our system, we adopt a simple pre-train strategy. Firstly, the baseNMT system is trained. The training results are used as the initial parameters for pre-training our proposed models with word predictions.

For decoding during test time, we simply decode until the end-of-sentence symbol eos occurs, using a beam search with a beam width of 5.

5.4 Translation Experiments

To see the effect of word predictions in translation, we evaluate these systems in case-insensitive IBM-BLEU (Papineni et al., 2002) on both CH-EN and DE-EN tasks.

The detailed results are show in the Table 1 and Table 2. Compared to the baseNMT system, all of our models achieve significant improvements. On the CH-EN experiments, simply adding word predictions to the initial state ($\mathrm{WP_E}$) already brings considerable improvements. The average improvement on test set is 2.53 BLEU, showing that constraining the initial state does lead to a higher translation quality. Adding word predic-

[1] includes LDC2002E18, LDC2003E07, LDC2003E14, LDC2004E12, LDC2004T08, LDC2005T06, LDC2005T10, LDC2006E26 and LDC2007T09

[2] https://github.com/nyu-dl/dl4mt-tutorial

Models	MT02(dev)	MT03	MT04	MT05	Test Ave.	IMP
baseNMT	34.04	34.92	36.08	33.88	34.96	–
WP$_{\mathrm{E}}$	39.36	37.17	39.11	36.20	37.49	**+2.53**
WP$_{\mathrm{D}}$	40.28	38.45	40.99	37.90	39.11	**+4.15**
WP$_{\mathrm{ED}}$	40.25	39.50	40.91	38.05	39.49	**+4.53**

Table 1: Case-insensitive 4-gram BLEU scores of baseNMT, WP$_{\mathrm{E}}$, WP$_{\mathrm{D}}$, WP$_{\mathrm{ED}}$ systems on the CH-EN experiments. (The "IMP" column presents the improvement of test average compared to the baseNMT.)

Models	NST13(dev)	NST14	IMP
baseNMT	23.56	20.68	–
WP$_{\mathrm{E}}$	24.44	21.09	**+0.41**
WP$_{\mathrm{D}}$	25.31	21.54	**+0.86**
WP$_{\mathrm{ED}}$	25.97	21.98	**+1.3**

Table 2: Case-insensitive 4-gram BLEU scores of baseNMT, WP$_{\mathrm{E}}$, WP$_{\mathrm{D}}$, WP$_{\mathrm{ED}}$ systems on the DE-EN experiments.

Models	Test	IMP
baseNMT	34.86	–
WP$_{\mathrm{ED}}$	39.49	+4.53
baseNMT-dropout	37.02	+2.06
WP$_{\mathrm{ED}}$-dropout	39.25	+4.29
baseNMT-ensemble(4)	37.71	+2.75
WP$_{\mathrm{ED}}$-ensemble(4)	40.75	+5.79

Table 3: Average case-insensitive 4-gram BLEU scores on the CH-EN experiments for baseNMT and WP$_{\mathrm{ED}}$ systems, with the dropout and ensemble techniques.

Models	Test	IMP
baseNMT	20.68	–
WP$_{\mathrm{ED}}$	21.98	+1.3
baseNMT-dropout	21.62	+0.94
WP$_{\mathrm{ED}}$-dropout	21.71	+1.03
baseNMT-ensemble(4)	21.58	+0.9
WP$_{\mathrm{ED}}$-ensemble(4)	22.47	+1.79

Table 4: Case-insensitive 4-gram BLEU scores on the DE-EN experiments for baseNMT and WP$_{\mathrm{ED}}$ systems, with the dropout and ensemble techniques.

tions to the hidden states in the decoder (WP$_{\mathrm{D}}$) leads to further improvements against baseNMT (4.15 BLEU), because WP$_{\mathrm{D}}$ adds constraints to the state transitions through different time steps in the decoder. Using both techniques improves the baseline by 4.53 BLEU. On the DE-EN experiments, the improvement of WP$_{\mathrm{E}}$ model is 0.41 BLEU and WP$_{\mathrm{D}}$ model is 0.86 BLEU on test set. When use both techniques, the WP$_{\mathrm{ED}}$ improves on the test set is 1.3 BLEU.

We compare our models with systems using dropout and ensemble techniques. The results show in Table 3 and 4. On the CH-EN experiments, the dropout method successfully improves the baseNMT system by 2.06 BLEU. However, it does not work on our WP$_{\mathrm{ED}}$ system. The ensemble technique improves the baseNMT system by 2.75 BLEU. It still improves WP$_{\mathrm{ED}}$ by 1.26 BLEU, but the improvement is smaller than on the baseNMT. On the DE-EN experiments, the phenomenon of experiments is similar to CH-EN experiments. The baseNMT system improves 0.94 through dropout method and 0.9 BLEU through ensemble method. The dropout technique also does not work on WP$_{\mathrm{ED}}$ and the ensemble technique improves 1.79 BLEU. These comparisons suggests that our system already learns better and stable values for the parameters, enjoying some of the benefits of general training techniques like dropout and ensemble. Compared to dropout and ensemble, our method WP$_{\mathrm{ED}}$ achieves the highest improvement against the baseline system on both CH-EN and DE-EN experiments. Along with ensemble method, the improvement could be up to 5.79 BLEU and 1.79 BLEU respectively.

5.5 Word Prediction Experiments

Since we include an explicit word prediction mechanism during the training of NMT systems, we also evaluate the prediction performance on the CH-EN experiments to see how the training is improved.

For each sentence in the test set, we use the initial state of the given model to make prediction about the possible words. We denote the set of top n words as T_n, the set of words in all the references

top-n	baseNMT		WP$_\mathrm{E}$	
	Prec.	Recall	Prec.	Recall
top-10	45%	17%	73%	30%
top-20	33%	21%	63%	43%
top-50	21%	30%	41%	55%
top-100	14%	39%	28%	68%
top-1k	2%	67%	4%	89%
top-5k	0.7%	84%	0.9%	95%
top-10k	0.4%	90%	0.5%	97%

Table 5: Comparison between baseNMT and WP$_\mathrm{E}$ in precision and recall for the different prediction size on the CH-EN experiments.

as R. We define the precision, recall of the word prediction as follows:

$$\text{precision} = \frac{|T_n \cap R|}{|T_n|} * 100\% \tag{28}$$

$$\text{recall} = \frac{|T_n \cap R|}{|R|} * 100\% \tag{29}$$

We compare the prediction performance of baseNMT and WP$_\mathrm{E}$. WP$_\mathrm{ED}$ has similar prediction results with WP$_\mathrm{E}$, so we omit its results. As shown in Table 5, baseNMT system has a relatively lower prediction precision, for example, 45% in top 10 prediction. With an explicit training, the WP$_\mathrm{E}$ could achieve a much higher precision in all conditions. Specifically, the precision reaches 73% in top 10. This indicates that the initial state in WP$_\mathrm{E}$ contains more specific information about the prediction of the target words, which may be a step towards better semantic representation, and leads to better translation quality.

Because the total words in the references are limited (around 50), the precision goes down, as expected, when a larger prediction set is considered. On the other hand, the recall of WP$_\mathrm{E}$ is also much higher than baseNMT. When given 1k predictions, WP$_\mathrm{E}$ could successfully predict 89% of the words in the reference. The recall goes up to 95% with 5k predictions, which is only 1/6 of the current vocabulary.

To analyze the process of word prediction, we draw the attention heatmap (Equation 16) between the initial state $\mathbf{s}_0$ and the bi-directional representation of each source side word $\mathbf{h}_i$ for an example sentence. As shown in Figure 3, both examples show that the initial states have a very strong attention with all the content words in the source sentence. The blank cells are mostly functions words or high frequent tokens such as “的 ('s)”, “是 (is)”, “而 (and)”, “它 (it)”, comma and period. This indicates that the initial state successfully encodes information about most of the content words in the source sentence, which contributes for a high prediction performance and leads to better translation.

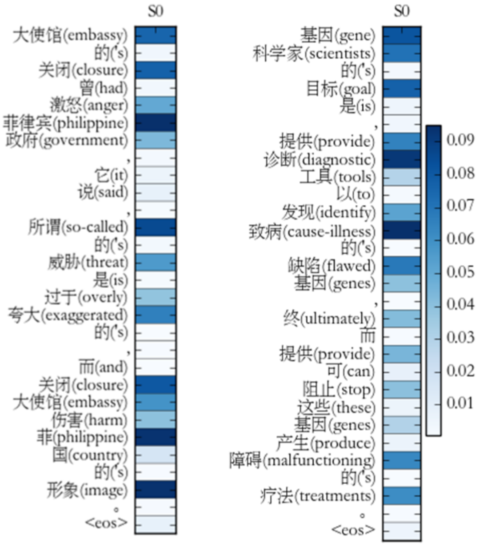


Figure 3: Two examples of the attention heatmap between the initial state $\mathbf{s}_0$ and the bi-directional representation of each source side word $\mathbf{h}_i$ from the CH-EN test sets. (The English translation of each source word is annotated in the parentheses after it.)

5.6 Improving Decoding Efficiency

To make use of the word prediction, we conduct experiments using the predicted vocabulary, with different vocabulary size (1k to 10k) on the CH-EN experiments, denoted as WP$_\mathrm{E}$-V and WP$_\mathrm{ED}$-V. The comparison is made in both translation quality and decoding time. As all our models with fixed vocabulary size have exactly the same number of parameters for decoding (extra mechanism is used only for training), we only plot the decoding time of the WP$_\mathrm{ED}$ for comparison. Figure 4 and 5 show the results.

When we start the experiments with top 1k vocabulary (1/30 of the baseline settings), the translation quality of both WP$_\mathrm{E}$-V and WP$_\mathrm{ED}$-V are already higher than the baseNMT; while their decoding time is less than 1/3 of an NMT system with 30k vocabulary. When the size of vocabulary increases, the translation quality improves as well. With a 6k predicted vocabulary (1/5 of the baseline settings), the decoding time is about 60% of a full-

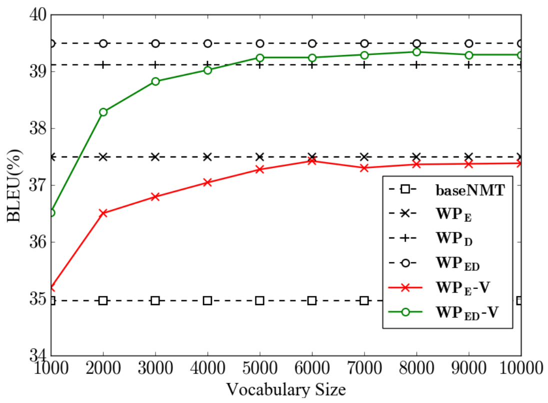


Figure 4: BLEU scores with different vocabulary sizes for each sentence on the CH-EN experiments. (The performance of baseNMT, WP$_E$, WP$_D$, WP$_{ED}$ are plotted as horizontal lines for comparison.)

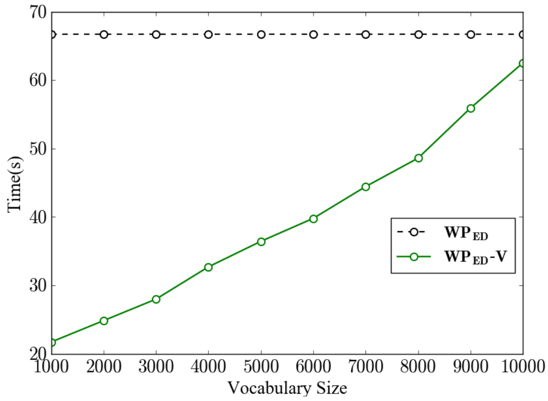


Figure 5: Decoding time with different vocabulary sizes for each sentence on the CH-EN experiments. (The horizontal line shows the decoding time for the systems with fixed vocabulary.)

vocabulary system; the performances of both systems with variable size vocabulary are comparable their corresponding fixed-vocabulary systems, which is higher than the baseNMT by 2.53 and 4.53 BLEU, respectively.

Although the comparison may not be fair enough due to the language pair and training conditions, the above relative improvements (e.g. WP$_{ED}$-V v.s. baseNMT) is much higher than previous research of manipulating the vocabularies (Jean et al., 2015; Mi et al., 2016; L'Hostis et al., 2016). This is because our mechanism is not only about reducing the vocabulary itself for each sentence or batch, it also brings improvement to the overall translation model. Please note that unlike these research, we keep the target vocabulary to be 30k in all our experiments, because we are not focusing on increasing the vocabulary size in this paper. It will be interesting to combine our mechanism with larger vocabulary to further enhance the translation performance. Again, our mechanism requires no extra annotation, dictionary, alignment or separate discriminative predictor, etc.

5.7 Translation Analysis

We also analyze real-case translations to see the difference between different systems (Table 6).

It is easy to see that the baseNMT system misses the translations of several important words, such as "advertising", "1.5", which are marked with underline in the reference. It also wrongly translates the company name "time warner inc." as the redundant information "internet company"; "america online" as "us line".

The results of dropout or ensemble show improvement compared to the baseNMT. But they still make mistakes about the translation of "online" and the company name "time warner inc.".

With WP$_{ED}$, most of these errors no longer exist, because we force the encoder and decoder to carry the exact information during translation.

6 Conclusions

The encoder-decoder architecture provides a general paradigm for learning machine translation from the source language to the target language. However, due to the large amount of parameters and relatively small training data set, the end-to-end learning of an NMT model may not be able to learn the best solution. We argue that at least part of the problem is caused by the long error backpropagation pipeline of the recurrent structures in multiple time steps, which provides no direct control of the information carried by the hidden states in both the encoder and decoder.

Instead of looking for other annotated data, we notice that the words in the target language sentence could be viewed as a natural annotation. We propose to use the word prediction mechanism to enhance the initial state generated by the encoder and extend the mechanism to control the hidden states of decoder as well. Experiments show promising results on the Chinese-English and German-English translation tasks. As a byproduct, the word predictor could be used to improve the efficiency of decoding, which may be

source	时代华纳公司的网络公司美国线上说, 它预期二○○二年的广告与商业销售将由二○○一年的二十七亿美元减少到十五亿美元。
reference	america online , the internet arm of time warner conglomerate , said it expects advertising and commerce revenue to decline from us \$ 2.7 billion in 2001 to us \$ 1.5 in 2002 .
baseNMT	in the *us line* , *the internet company 's internet company said on the internet* that it expected that the business sales in 2002 would fall from \$ UNK billion to \$ UNK billion in 2001 .
baseNMT +dropout	on *the united states line* , *UNK 's* internet company said *on the internet* that it expects to reduce the annual **advertising and commercial** sales from \$ UNK billion in 2001 to \$ **1.5** billion .
baseNMT +ensemble	in the *us line* , *the internet company 's internet company* said that it expected that the **advertising and commercial** sales volume for 2002 would be reduced from us \$ UNK billion to us \$ **1.5** billion in 2001 .
WP_{ED}	**the internet company** *of* **time warner inc.** , the *us* online , said that it expects that **the advertising and commercial** sales in 2002 will decrease from \$ UNK billion in 2001 to us \$ **1.5** billion .

Table 6: Comparisons of different systems in translating the same example sentence, which from CH-EN test sets. ("source" indicates the source sentence; "reference" indicates the human translation; the translation results are indicated by their system names, including our best "WP_{ED}" systems. The underline words in the reference are missed in the baseNMT output; the bold font indicates improvements over the baseNMT system; and the italic font indicates remaining translation errors.)

crucial for large scale applications.

Our attempts demonstrate that the learning of the large scale neural network systems is still not good enough. In the future, it might be helpful to analyze the benefits of jointly learning other related tasks together with machine translation, to provide further control of the learning process. It is interesting to demonstrate the effectiveness of the proposed mechanism on other sequence to sequence learning tasks as well.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. Shujian Huang is the corresponding author. This work is supported by the National Science Foundation of China (No. 61672277, 61472183), the Jiangsu Provincial Research Foundation for Basic Research (No. BK20170074).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Srinivas Bangalore, Patrick Haffner, and Stephan Kanthak. 2007. Statistical machine translation through global lexical selection and sentence reconstruction. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, pages 152–159. http://aclweb.org/anthology/P07-1020.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. 2017. Massive Exploration of Neural Machine Translation Architectures. *ArXiv e-prints* .

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1724–1734. https://doi.org/10.3115/v1/D14-1179.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* abs/1412.3555. http://arxiv.org/abs/1412.3555.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1723–1732. https://doi.org/10.3115/v1/P15-1166.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1–10. https://doi.org/10.3115/v1/P15-1001.

Minwoo Jeong, Kristina Toutanova, Hisami Suzuki, and Chris Quirk. 2010. A discriminative lexicon model for complex morphology. In *Proceedings of the Ninth Conference of the Association for Machine Translation in the Americas (AMTA 2010)*.

Gurvan L'Hostis, David Grangier, and Michael Auli. 2016. Vocabulary selection strategies for neural machine translation. *CoRR* abs/1610.00072. http://arxiv.org/abs/1610.00072.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task sequence to sequence learning. *CoRR* abs/1511.06114. http://arxiv.org/abs/1511.06114.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *CoRR* abs/1604.00788. http://arxiv.org/abs/1604.00788.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015b. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1412–1421. https://doi.org/10.18653/v1/D15-1166.

Arne Mauser, Saša Hasan, and Hermann Ney. 2009. Extending statistical machine translation with discriminative and trigger-based lexicon models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 210–218. http://aclweb.org/anthology/D09-1022.

Fandong Meng, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Interactive attention for neural machine translation. *CoRR* abs/1610.05011. http://arxiv.org/abs/1610.05011.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Vocabulary manipulation for neural machine translation. *CoRR* abs/1605.03209. http://arxiv.org/abs/1605.03209.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '02, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15(1):1929–1958. http://dl.acm.org/citation.cfm?id=2627435.2670313.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Jun Suzuki and Masaaki Nagata. 2017. Rnn-based encoder-decoder approach with word frequency estimation. *CoRR* abs/1701.00138. http://arxiv.org/abs/1701.00138.

Ke Tran, Arianna Bisazza, and Christof Monz. 2014. Word translation prediction for morphologically rich languages with bilingual neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1676–1688. https://doi.org/10.3115/v1/D14-1175.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1535–1545. http://aclweb.org/anthology/D16-1160.

Towards Decoding as Continuous Optimisation in Neural Machine Translation

Cong Duy Vu Hoang[†] and **Gholamreza Haffari**[‡] and **Trevor Cohn**[†]

† University of Melbourne
Melbourne, VIC, Australia

‡ Monash University
Clayton, VIC, Australia

vhoang2@student.unimelb.edu.au, gholamreza.haffari@monash.edu,
t.cohn@unimelb.edu.au

Abstract

We propose a novel decoding approach for neural machine translation (NMT) based on continuous optimisation. We reformulate decoding, a discrete optimization problem, into a continuous problem, such that optimization can make use of efficient gradient-based techniques. Our powerful decoding framework allows for more accurate decoding for standard neural machine translation models, as well as enabling decoding in intractable models such as intersection of several different NMT models. Our empirical results show that our decoding framework is effective, and can leads to substantial improvements in translations, especially in situations where greedy search and beam search are not feasible. Finally, we show how the technique is highly competitive with, and complementary to, reranking.

1 Introduction

Sequence to sequence learning with neural networks (Graves, 2013; Sutskever et al., 2014; Lipton et al., 2015) is typically associated with two phases: training and decoding (*a.k.a.* inference). Model parameters are learned by optimising the training objective, in order that the model can produce good translations when decoding unseen sentences. The majority of research has focused on the training paradigm or network architecture, however effective means of decoding have been under-investigated. Conventional heuristic-based approaches for approximate inference include greedy, beam, and stochastic search. Greedy and beam search have been empirically proved to be adequate for many sequence to sequence tasks, and are the standard methods for NMT decoding.

However, these inference approaches have several drawbacks. Firstly, although NMT models use a left-to-right generation which would appear to facilitate efficient search, the models themselves use a recurrent architecture, and accordingly are non-Markov. This prevents exact dynamic programming solutions, and moreover, limits the potential to incorporate additional global features or constraints. Global factors can be highly useful in producing better and more diverse translations. Secondly, the sequential decoding of symbols in the target sequence, the inter-dependencies among the target symbols are not fully exploited. For example, when decoding the words of the target sentence in a left-to-right manner, the right context is not exploited leading potentially to inferior performance (see Watanabe and Sumita (2002a) who apply this idea in traditional statistical MT). A natural way to capture this is to intersect left-to-right and right-to-left models, however the resulting model has no natural generation order, and thus standard decoding methods are unsuitable.

We introduce a novel decoding framework (§ 3) that relaxes this *discrete* optimisation problem into a *continuous* optimisation problem. This is akin to linear programming relaxation approach for approximate inference in graphical models with discrete random variables, where the exact inference is NP-hard (Sontag, 2010; Belanger and McCallum, 2016). The resulting continuous optimisation problem is challenging due to the non-linearity and non-convexity of the relaxed decoding objective. We make use of stochastic gradient descent (SGD) and exponentiated gradient (EG) algorithms for decoding based on our relaxation approach.[1] Our decoding framework is powerful and flexible, as it enables us to decode with global constraints involving intersection of multiple NMT

[1] Both methods are mainly used for training in prior work.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 146–156
Copenhagen, Denmark, September 7–11, 2017.

models (§4). We present experimental results on Chinese-English and German-English translation tasks, confirming the effectiveness of our relaxed optimisation method for decoding (§5).

2 Neural Machine Translation

We briefly review the attentional neural translation model proposed by Bahdanau et al. (2015) as a sequence-to-sequence neural model onto which we apply our decoding framework.

In neural machine translation (NMT), the probability of the target sentence $\boldsymbol{y}$ given a source sentence $\boldsymbol{x}$ is written as:

$$\begin{aligned} \mathrm{P}_{\boldsymbol{\Theta}}(\boldsymbol{y}|\boldsymbol{x}) &= \sum_{i=1}^{|\boldsymbol{y}|} \log \mathrm{P}_{\boldsymbol{\Theta}}(y_i|\boldsymbol{y}_{<i}, \boldsymbol{x}) \qquad (1) \\ y_i|\boldsymbol{y}_{<i}, \boldsymbol{x} &\sim \mathrm{softmax}\left(\boldsymbol{f}(\boldsymbol{\Theta}, \boldsymbol{y}_{<i}, \boldsymbol{x})\right) \end{aligned}$$

where f is a non-linear function of the previously generated sequence of words $\boldsymbol{y}_{<i}$, the source sentence $\boldsymbol{x}$, and the model parameters $\boldsymbol{\Theta}$. In this paper, we realise $\boldsymbol{f}$ as follows:

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{\Theta}, \boldsymbol{y}_{<i}, \boldsymbol{x}) &= \boldsymbol{W}_o \cdot \mathrm{MLP}\left(\boldsymbol{c}_i, \boldsymbol{E}_T^{y_{i-1}}, \boldsymbol{g}_i\right) + \boldsymbol{b}_o \\ \boldsymbol{g}_i &= \mathrm{RNN}_{dec}^{\phi}\left(\boldsymbol{c}_i, \boldsymbol{E}_T^{y_{i-1}}, \boldsymbol{g}_{i-1}\right) \end{aligned}$$

where MLP is a single hidden layer neural network with $\tanh$ activation function, and $\boldsymbol{E}_T^{y_{i-1}}$ is the embedding of the target word y_{i-1} in the embedding matrix $\boldsymbol{E}_T \in \mathbb{R}^{n_e \times |V_T|}$ of the target language vocabulary V_T and n_e is the embedding dimension. The state $\boldsymbol{g}_i$ of the decoder RNN is a function of y_{i-1}, its previous state $\boldsymbol{g}_{i-1}$, and the *context* $\boldsymbol{c}_i = \sum_{j=1}^{|\boldsymbol{x}|} \alpha_{ij} \boldsymbol{h}_j$ summarises parts of the source sentence which are *attended* to, where

$$\begin{aligned} \boldsymbol{\alpha}_i &= \mathrm{softmax}(\boldsymbol{e}_i) \quad ; \quad e_{ij} = \mathrm{MLP}\left(\boldsymbol{g}_{i-1}, \boldsymbol{h}_j\right) \\ \boldsymbol{h}_j &= \mathrm{biRNN}_{enc}^{\theta}\left(\boldsymbol{E}_S^{x_j}, \overrightarrow{\boldsymbol{h}}_{j-1}, \overleftarrow{\boldsymbol{h}}_{j+1}\right) \end{aligned}$$

In above, $\overrightarrow{\boldsymbol{h}}_i$ and $\overleftarrow{\boldsymbol{h}}_i$ are the states of the left-to-right and right-to-left RNNs encoding the source sentence, and $\boldsymbol{E}_S^{x_j}$ is the embedding of the source word x_j in the embedding matrix $\boldsymbol{E}_S \in \mathbb{R}^{n'_e \times |V_S|}$ of the source language vocabulary V_S and n'_e is the embedding dimension.

Given a bilingual corpus $\mathcal{D}$, the model parameters are learned by maximizing the conditional log-likelihood,

$$\boldsymbol{\Theta}^* := \mathrm{argmax}_{\boldsymbol{\Theta}} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{D}} \log \mathrm{P}_{\boldsymbol{\Theta}}(\boldsymbol{y} \mid \boldsymbol{x}). \qquad (2)$$

The model parameters $\boldsymbol{\Theta}$ include the weight matrix $\boldsymbol{W}_o \in \mathbb{R}^{|V_T| \times n_h}$ and the bias $\boldsymbol{b}_o \in \mathbb{R}^{|V_T|}$ – with n_H denoting the hidden dimension size – as well as the RNN encoder $\mathrm{biRNN}_{enc}^{\theta}$ / decoder $\mathrm{RNN}_{dec}^{\phi}$ parameters, word embedding matrices, and the parameters of the attention mechanism. The model is trained end-to-end by optimising the training objective using stochastic gradient descent (SGD) or its variants. In this paper, we focus on the decoding problem, which we turn to in the next section.

3 Decoding as Continuous Optimisation

In decoding, we are interested in finding the highest probability translation for a given source sentence:

$$\mathrm{minimise}_{\boldsymbol{y}} \quad -P_{\boldsymbol{\Theta}}(\boldsymbol{y} \mid \boldsymbol{x}) \quad \text{s.t.} \quad \boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}} \qquad (3)$$

where $\mathcal{Y}_{\boldsymbol{x}}$ is the space of possible translations for the source sentence $\boldsymbol{x}$. In general, searching $\mathcal{Y}_{\boldsymbol{x}}$ to find the highest probability translation is intractable due to the recurrent nature of eqn (1) which prevents dynamic programming for efficient search. This is problematic, as the space of translations is exponentially large with respect to the output length $|\boldsymbol{y}|$.

We now formulate this discrete optimisation problem as a continuous one, and then use standard algorithms for continuous optimisation for decoding. Let us assume that the maximum length of a possible translation for a source sentence is known and denote it as ℓ. The best translation for a given source sentence solves the following optimisation problem:

$$\begin{aligned} \boldsymbol{y}^* = &\operatorname*{arg\,min}_{y_1,\ldots,y_\ell} \sum_{i=1}^{\ell} -\log \mathrm{P}_{\boldsymbol{\Theta}}(y_i \mid \boldsymbol{y}_{<i}, \boldsymbol{x}) \qquad (4) \\ &\text{s.t.} \quad \forall i \in \{1 \ldots \ell\} : y_i \in V_T. \end{aligned}$$

where we allow the translation to be padded with sentinel symbols to the right, which are ignored in computing the model probability. Equivalently, we can rewrite the above discrete optimisation problem as follows:

$$\begin{aligned} &\operatorname*{arg\,min}_{\tilde{\boldsymbol{y}}_1,\ldots,\tilde{\boldsymbol{y}}_\ell} -\sum_{i=1}^{\ell} \tilde{\boldsymbol{y}}_i \cdot \log \mathrm{softmax}\left(\boldsymbol{f}\left(\boldsymbol{\Theta}, \tilde{\boldsymbol{y}}_{<i}, \boldsymbol{x}\right)\right) \\ &\text{s.t.} \quad \forall i \in \{1 \ldots \ell\} : \tilde{\boldsymbol{y}}_i \in \mathbb{I}^{|V_T|} \qquad (5) \end{aligned}$$

where $\tilde{\boldsymbol{y}}_i$ are vectors using the one-hot representation of the target words $\mathbb{I}^{|V_T|}$.

Algorithm 1 The EG Algorithm for Decoding by Optimisation

```
For all $i$ initialise $\hat{\boldsymbol{y}}_i^0 \in \Delta_{|V_T|}$
**for** $t = 1, \dots, \text{MaxIter}$ **do**                                   ▷ $Q(.)$ is defined as eqn (6)
    For all $i, w$ : calculate $\nabla_{i,w}^{t-1} = \frac{\partial Q(\hat{\boldsymbol{y}}_1^{t-1}, \dots, \hat{\boldsymbol{y}}_\ell^{t-1})}{\partial \hat{\boldsymbol{y}}_i(w)}$     ▷ using back-propagation
    For all $i, w$ : update $\hat{\boldsymbol{y}}_i^t(w) \propto \hat{\boldsymbol{y}}_i^{t-1}(w) \cdot \exp\left(-\eta \nabla_{i,w}^{t-1}\right)$     ▷ $\eta$ is the step size
**return** $\arg\min_t Q(\hat{\boldsymbol{y}}_1^t, \dots, \hat{\boldsymbol{y}}_\ell^t)$
```

We now convert the optimisation problem (5) to a continuous one by dropping the integrality constraints $\tilde{\boldsymbol{y}}_i \in \mathbb{I}^{|V|}$ and require the variables to take values from the probability simplex:

$$\boxed{\begin{aligned} &\arg\min_{\hat{\boldsymbol{y}}_1, \dots, \hat{\boldsymbol{y}}_\ell} - \sum_{i=1}^{\ell} \hat{\boldsymbol{y}}_i \cdot \log \operatorname{softmax}\left(\boldsymbol{f}\left(\boldsymbol{\Theta}, \hat{\boldsymbol{y}}_{<i}, \boldsymbol{x}\right)\right) \\ &\qquad\text{s.t.} \quad \forall i \in \{1 \dots \ell\} : \hat{\boldsymbol{y}}_i \in \Delta_{|V_T|} \end{aligned}}$$

where $\Delta_{|V_T|}$ is the $|V_T|$-dimensional probability simplex, i.e., $\{\hat{\boldsymbol{y}}_i \in [0, 1]^{|V_T|} : \|\hat{\boldsymbol{y}}_i\|_1 = 1\}$. Intuitively, this amounts to replacing $\boldsymbol{E}_T^{y_i}$ with the *expected* embedding of target language words $\mathbb{E}_{\hat{\boldsymbol{y}}_i(w)}[\boldsymbol{E}_T^w]$ under the distribution $\hat{\boldsymbol{y}}_i$.

After solving the above constrained continuous optimisation problem, there is no guarantee that the resulting solution $\{\hat{\boldsymbol{y}}_i^*\}_{i=1}^{\ell}$ will comprise one-hot vectors, i.,e., target language words. Instead it can find *fractional* solutions, that require 'rounding' in order to to resolve them to lexical items. To solve this problem, we take the $\arg\max$,[2] i.e., take the highest scoring word for each position $\hat{\boldsymbol{y}}_i^*$. We leave exploration of more elaborate projection techniques to the future work.

In the context of graphical models, the above relaxation technique gives rise to linear programming for approximate inference (Sontag, 2010; Belanger and McCallum, 2016). However, our decoding problem is much harder due to the non-linearity and non-convexity of the objective function operating on high dimensional space for deep models. We now turn our attention to optimisation algorithms to effectively solve the decoding optimisation problem.

3.1 Exponentiated Gradient (EG)

Exponentiated gradient (Kivinen and Warmuth, 1997) is an elegant algorithm for solving optimisation problems involving simplex constraints. Recall our constrained optimisation problem:

$$\begin{aligned} &\arg\min_{\hat{\boldsymbol{y}}_1, \dots, \hat{\boldsymbol{y}}_\ell} Q(\hat{\boldsymbol{y}}_1, \dots, \hat{\boldsymbol{y}}_\ell) \\ &\text{s.t.} \quad \forall i \in \{1 \dots \ell\} : \hat{\boldsymbol{y}}_i \in \Delta_{|V_T|} \end{aligned}$$

where $Q(\hat{\boldsymbol{y}}_1, \dots, \hat{\boldsymbol{y}}_\ell)$ is defined as

$$-\sum_{i=1}^{\ell} \hat{\boldsymbol{y}}_i \cdot \log \operatorname{softmax}\left(\boldsymbol{f}\left(\boldsymbol{\Theta}, \hat{\boldsymbol{y}}_{<i}, \boldsymbol{x}\right)\right). \qquad (6)$$

EG is an iterative algorithm, which updates each distribution $\hat{\boldsymbol{y}}_i^t$ in the current time-step t based on the distributions of the previous time-step as follows:

$$\forall w \in V_T : \quad \hat{\boldsymbol{y}}_i^t(w) = \frac{1}{Z_i^t} \hat{\boldsymbol{y}}_i^{t-1}(w) \exp\left(-\eta \nabla_{i,w}^{t-1}\right)$$

where η is the step size, $\nabla_{i,w}^{t-1} = \frac{\partial Q(\hat{\boldsymbol{y}}_1^{t-1}, \dots, \hat{\boldsymbol{y}}_\ell^{t-1})}{\partial \hat{\boldsymbol{y}}_i(w)}$ and Z_i^t is the normalisation constant

$$Z_i^t = \sum_{w \in V_T} \hat{\boldsymbol{y}}_i^{t-1}(w) \exp\left(-\eta \nabla_{i,w}^{t-1}\right).$$

The partial derivatives $\nabla_{i,w}$ are calculated using the back propagation algorithm treating $\{\hat{\boldsymbol{y}}_i\}_{i=1}^{\ell}$ as *parameters* and the original parameters of the model $\boldsymbol{\Theta}$ as constants. Adapting EG to our decoding problem leads to Algorithm 1.

It can be shown that the EG algorithm is a gradient descent algorithm for minimising the following objective function subject to the simplex constraints:

$$\begin{aligned} &Q(\hat{\boldsymbol{y}}_1, \dots, \hat{\boldsymbol{y}}_\ell) - \gamma \sum_{i=1}^{\ell} \sum_{w \in V_T} \hat{\boldsymbol{y}}_i(w) \log \frac{1}{\hat{\boldsymbol{y}}_i(w)} \\ &= Q(\hat{\boldsymbol{y}}_1, \dots, \hat{\boldsymbol{y}}_\ell) - \gamma \sum_{i=1}^{\ell} \text{Entropy}(\hat{\boldsymbol{y}}_i) \qquad (7) \end{aligned}$$

In other words, the algorithm looks for the maximum entropy solution which also maximizes the

[2] Ties are broken arbitrarily.

log likelihood under the model. There are intriguing parallels with the maximum entropy formulation of log-linear models (Berger et al., 1996). In our setting, the entropy term acts as a prior which discourages overly-confident estimates in the absence of sufficient evidence.

3.2 Stochastic Gradient Descent (SGD)

To be able to apply SGD to our optimisation problem, we need to make sure that the simplex constraints are enforced. One way to achieve this is by reparameterising using the softmax transformation, i.e. $\hat{\boldsymbol{y}}_i = \text{softmax}\left(\hat{\boldsymbol{r}}_i\right)$. The resulting *unconstrained* optimisation problem, now over $\hat{\boldsymbol{r}}_i$, becomes

$$\underset{\hat{\boldsymbol{r}}_1,\ldots,\hat{\boldsymbol{r}}_\ell}{\arg\min} - \sum_{i=1}^{\ell} \text{softmax}\left(\hat{\boldsymbol{r}}_i\right) \cdot \log \text{softmax}\left(\mathrm{f}\left(\boldsymbol{\Theta}, \hat{\boldsymbol{y}}_{<i}, \boldsymbol{x}\right)\right)$$

where $\boldsymbol{E}_T^{y_i}$ is replaced with the expected embedding of the target words under the distribution resulted from the $\mathbb{E}_{\text{softmax}(\hat{\boldsymbol{r}}_i)}\left[\boldsymbol{E}_T^{w}\right]$ in the model.

To apply SGD updates, we need the gradient of the objective function with respect to the new variables $\hat{\boldsymbol{r}}_i$ which can be derived with the back-propagation algorithm based on the chain rule:

$$\frac{\partial Q}{\partial \hat{\boldsymbol{r}}_i(w)} = \sum_{w' \in V_T} \frac{\partial Q(.)}{\partial \hat{\boldsymbol{y}}_i(w')} \frac{\partial \hat{\boldsymbol{y}}_i(w')}{\partial \hat{\boldsymbol{r}}_i(w)}$$

The resulting SGD algorithm is summarized in Algorithm 2.

4 Decoding in Extended NMT

Our decoding framework allows us to effectively and flexibly add additional global factors over the output symbols during inference. This enables decoding for richer global models, for which there is no effective means of greedy decoding or beam search. We outline several such models, and their corresponding relaxed objective functions for optimisation-based decoding.

Bidirectional Ensemble. Standard NMT generates the translation in a left-to-right manner, conditioning each target word on its left context. However, the joint probability of the translation can be decomposed in a myriad of different orders; one compelling alternative would be to condition each target word on its right context, i.e., generating the target sentence from right-to-left. We would not expect a right-to-left model to outperform a left-to-right, however, as the left-to-right ordering reflects the natural temporal order of spoken language. However, the right-to-left model is likely to provide a complementary signal in translation, as it will be bringing different biases and making largely independent prediction errors to those of the left-to-right model. For this reason, we propose to use both models, and seek to find translations that have high probability according both models (this mirrors work on bidirectional decoding in classical statistical machine translation by Watanabe and Sumita (2002b).) Decoding under the ensemble of these models leads to an intractable search problem, not well suited to traditional greedy or beam search algorithms, which require a fixed generation order of the target words. This ensemble decoding problem can be formulated simply in our linear relaxation approach, using the following objective function:

$$\begin{aligned} \mathcal{C}_{+\text{bidir}} := & -\alpha \log \mathrm{P}_{\boldsymbol{\Theta}_{\leftarrow}}\left(\boldsymbol{y} \mid \boldsymbol{x}\right) \\ & -(1-\alpha) \log \mathrm{P}_{\boldsymbol{\Theta}_{\rightarrow}}\left(\boldsymbol{y} \mid \boldsymbol{x}\right); \end{aligned} \qquad (8)$$

where α is an interpolation hyper-parameter, which we set to 0.5; $\boldsymbol{\Theta}_{\rightarrow}$ and $\boldsymbol{\Theta}_{\leftarrow}$ are the pre-trained left-to-right and right-to-left models, respectively. This bidirectional agreement may also lead to improvement in translation diversity, as shown in Li and Jurafsky (2016) in a re-ranking evaluation.

Bilingual Ensemble. Another source of complementary information is in terms of the translation direction, that is forward translation from the source to the target language, and reverse translation in the target to source direction. Decoding must find a translation which scores well under both the forward and reverse translation models. This is inspired by the direct and reverse feature functions commonly used in classical discriminative SMT (Och and Ney, 2002) which have been shown to offer some complementary benefits (although see Lopez and Resnik (2006)). More specifically, we decode for the best translation in the intersection of the source-to-target and target-to-source models by minimizing the following objective function:

$$\begin{aligned} \mathcal{C}_{+\text{biling}} := & -\alpha \log \mathrm{P}_{\boldsymbol{\Theta}_{\text{s}\rightarrow\text{t}}}\left(\boldsymbol{y} \mid \boldsymbol{x}\right) \\ & -(1-\alpha) \log \mathrm{P}_{\boldsymbol{\Theta}_{\text{s}\leftarrow\text{t}}}\left(\boldsymbol{x} \mid \boldsymbol{y}\right); \end{aligned} \qquad (9)$$

where α is an interpolation hyper-parameter to be fine-tuned; and $\boldsymbol{\Theta}_{\text{s}\rightarrow\text{t}}$ and $\boldsymbol{\Theta}_{\text{s}\leftarrow\text{t}}$ are the pre-trained

Algorithm 2 The SGD Algorithm for Decoding by Optimisation

```
For all $i$ initialise $\hat{\boldsymbol{r}}_i^0$
for $t = 1, \ldots, \text{MaxIter}$ do                ▷ $Q(.)$ is defined in eqn (6) and $\hat{\boldsymbol{y}}_i = \text{softmax}(\hat{\boldsymbol{r}}_i)$
    For all $i, w$ : calculate $\nabla_{i,w}^{t-1} = \sum_{w' \in V_T} \frac{\partial Q(\hat{\boldsymbol{y}}_1^{t-1}, \ldots, \hat{\boldsymbol{y}}_\ell^{t-1})}{\partial \hat{\boldsymbol{y}}_i(w')} \frac{\partial \hat{\boldsymbol{y}}_i(w')}{\partial \hat{\boldsymbol{r}}_i(w)}$    ▷ using backpropagation
    For all $i, w$ : update $\hat{\boldsymbol{r}}_i^t(w) = \hat{\boldsymbol{r}}_i^{t-1}(w) - \eta \nabla_{i,w}^{t-1}$    ▷ $\eta$ is the step size
return $\arg\min_t Q(\text{softmax}(\hat{\boldsymbol{r}}_1^t), \ldots, \text{softmax}(\hat{\boldsymbol{r}}_\ell^t))$
```

	# tokens	**# types**	**# sents**
BTEC zh→en			
train	422k / 454k	3k / 3k	44,016
dev	10k / 10k	1k / 1k	1,006
test	5k / 5k	1k / 1k	506
TED Talks de→en			
train	4m / 4m	26k / 19k	194,181
dev-test2010	33k / 35k	4k / 3k	1,565
test2014	26k / 27k	4k / 3k	1,305
WMT 2016 de→en			
train	107m / 108m	90k / 78k	4m
dev-test2013&14	154k / 152k	20k / 13k	6003
test2015	54k / 54k	10k / 8k	2169

Table 1: Statistics of the training and evaluation sets; token and types are presented for both source/target languages.

source-to-target and target-to-source models, respectively. Decoding for the best translation under the above objective function leads to an intractable search problem, as the reverse model is global over the target language, meaning there is no obvious means of search with a greedy algorithm or similar.

Discussion. There are two important considerations on how best to initialise the relaxed optimisation in the above settings, and how best to choose the step size. As the relaxed optimisation problem is, in general, non-convex, finding a plausible initialisation is likely to be important for avoiding local optima. Furthermore, a proper step size is a key in the success of the EG-based and SGD-based optimisation algorithms, and there is no obvious method how to best choose its value. We may also adaptively change the step size using (scheduled) annealing or via the line search. We return to this considerations in the experimental evaluation.

5 Experiments

5.1 Setup

Datasets. We conducted our experiments on datasets with different scales, translating between Chinese→English using the BTEC corpus, and German→English using the IWSLT 2015 TED Talks(Cettolo et al., 2014) and WMT 2016[3] corpora. The statistics of the datasets can be found in Table 1.

NMT Models. We implemented our continuous-optimisation based decoding method on top of the Mantidae toolkit[4] (Cohn et al., 2016), and using the *dynet* deep learning library[5] (Neubig et al., 2017). All neural network models were configured with 512 input embedding and hidden layer dimensions, and 256 alignment dimension, with 1 and 2 hidden layers in the source and target, respectively. We used a LSTM recurrent structure (Hochreiter and Schmidhuber, 1997) for both source and target RNN sequences. For the vocabulary, we use word frequency cut-off of 5, and words rarer than this were mapped to a sentinel. For the large-scale WMT dataset, we applied byte-pair encoding (BPE) method (Sennrich et al., 2016) to better handle unknown words.[6] For training our neural models, we use early stopping based on development perplexity, which usually occurs after 5-8 epochs.

Evaluation Metrics. We evaluated in terms of search error, measured using the model score of the inferred solution (either continuous or discrete), as well as measuring the end translation quality with case-insensitive BLEU (Papineni et al., 2002). The continuous cost measures $-\frac{1}{|\hat{\boldsymbol{y}}|} \log P_{\boldsymbol{\Theta}}(\hat{\boldsymbol{y}} \mid \boldsymbol{x})$ under the model $\boldsymbol{\Theta}$; the discrete model score has the same formulation, albeit using the discrete rounded solution $\boldsymbol{y}$ (see §3). Note the cost can be used as a tool for selecting the best inference solution, as well as assessing convergence, as we illustrate below.

[3] http://www.statmt.org/wmt16/translation-task.html

[4] https://github.com/duyvuleo/Mantidae

[5] https://github.com/clab/dynet

[6] With BPE, the out of vocabulary rates on heldout data are $< 1\%$.

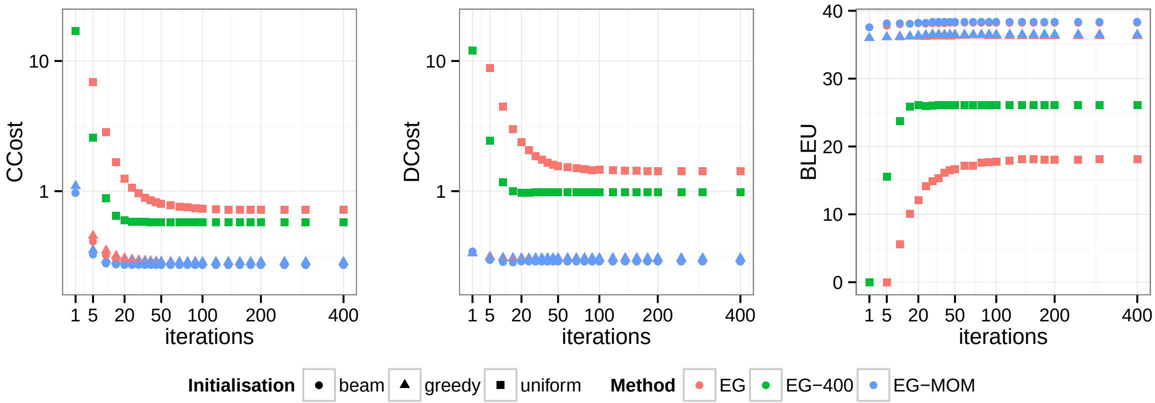


Figure 1: Analysis on effects of initialisation states (uniform vs. greedy vs. beam), step size annealing, momentum mechanism from BTEC zh→en translation. **EG-400**: EG algorithm with step size $\eta = 400$ (otherwise $\eta = 50$); **EG-MOM**: EG algorithm with momentum.

5.2 Results and Analysis

Initialisation and Step Size. As our relaxed optimisation problems are non-convex, local optima are likely to be a problem. We test this empirically, focusing on the effect that initialisation and step size, η, have on the inference quality.

For plausible initialisation states, we evaluate different strategies: *uniform* in which the relaxed variables $\hat{\boldsymbol{y}}$ are initialised to $\frac{1}{|V_T|}$; and *greedy* or *beam* whereby $\hat{\boldsymbol{y}}$ are initialised based on an already good solution produced by a baseline decoder with greedy (gdec) or beam (bdec). Instead of using the Viterbi outputs as a one-hot representation, we initialise to the probability prediction vectors,[7] which serves to limit attraction of the initialisation condition, which is likely to be a local (but not global) optima.

Figure 1 illustrates the effect of initialisation on the EG algorithm, in terms of search error (left and middle) and translation quality (right), as we vary the number of iterations of inference. There is clear evidence of non-convexity: all initialisation methods can be seen to converge using all three measures, however they arrive at highly different solutions. Uniform initialisation is clearly not a viable approach, while greedy and beam initialisation both yield much better results. The best initialisation, beam, outperforms both greedy and beam decoding in terms of BLEU.

Note that the EG algorithm has fairly slow convergence, requiring at least 100 iterations, irrespective of the initialisation. To overcome this, we use momentum (Qian, 1999) to accelerate the convergence by modifying the term $\nabla^t_{i,w}$ in Algorithm 1 with a weighted moving average of past gradients:

$$\nabla^{t-1}_{i,w} = \gamma \nabla^{t-2}_{i,w} + \eta \frac{\partial Q(\hat{\boldsymbol{y}}^{t-1}_1, \ldots, \hat{\boldsymbol{y}}^{t-1}_\ell)}{\partial \hat{\boldsymbol{y}}_i(w)}$$

where we set the momentum term $\gamma = 0.9$. The EG with momentum (**EG-MOM**) converges after fewer iterations (about 35), and results in marginally better BLEU scores. The momentum technique is usually used for SGD involving additive updates; it is interesting to see it also works in EG with multiplicative updates.

The step size, η, is another important hyperparameter for gradient based search. We tune the step size using line search over $[10, 400]$ over the development set. Figure 1 illustrates the effect of changing step size from 50 to 400 (compare **EG** and **EG-400** with **uniform**), which results in a marked difference of about 10 BLEU points, underlining the importance of tuning this value. We found that EG with momentum had less of a reliance on step size, with optimal values in $[10, 50]$; we use this setting hereafter.

Continuous vs Discrete Costs. Another important question is whether the assumption behind continuous relaxation is valid, i.e., if we optimise a continuous cost to solve a discrete problem, do we improve the discrete output? Although the continuous cost diminishes with inference iterations (Figure 1 left), and appears to converge, it is not clear whether this corresponds to a better discrete output (note that the discrete cost and BLEU

[7] Here, EG uses softmax normalization whereas SGD uses the pre-softmax vector.

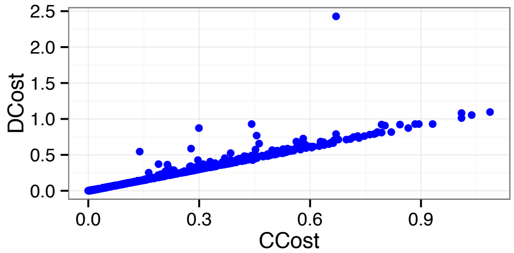


Figure 2: Comparing discrete vs continuous costs from BTEC zh→en translation, using the EG algorithm with momentum, $\eta = 50$. Each point corresponds to a sentence.

	BLEU	AvgLen
bdec$_{\text{left-to-right}}$	26.69	20.73
filtered rerank	26.84	20.66
EGdec w/ beam init	27.34	20.73
full rerank	27.34	21.76
EGdec w/ rerank init	27.78	21.70

Table 2: The BLEU evaluation results with EG algorithm against 100-best reranking on WMT evaluation dataset.

scores do show improvements Figure 1 centre and right.) Figure 2 illustrates the relation between the two cost measures, showing that in most cases the discrete and continuous costs are identical. Linear relaxation fails only for a handful of cases, where the nearest discrete solution is significantly worse than it would appear using the continuous cost.

EG vs SGD. Both the EG and SGD algorithms are iterative methods for solving the relaxed optimisation problem with simplex constraints. We measure empirically their difference in terms of quality of inference and speed of convergence, as illustrated in Figure 3. Observe that SGD requires 150 iterations for convergence, whereas EG requires many fewer (50). This concurs with previous work on learning structured prediction models with EG (Globerson et al., 2007). Further, the EG algorithm consistently produces better results in terms of both model cost and BLEU.

EG vs Reranking. Reranking is an alternative method for integrating global factors into the existing NMT systems. We compare our EG decoding algorithm against the reranking approach with bidirectional factor where the N-best outputs of a left-to-right decoder is re-scored with the forced decoder operating in a right-to-left fashion. The results are shown in Table 2. Our EG algorithm initialised with the reranked output achieves the best BLEU score. We also compare reranking with EG algorithm initialised with the beam decoder, where for direct comparison we filter out sentences with length greater than that of the beam output in the k-best lists. These results show that the EG algorithm is capable of effectively exploiting the search space.

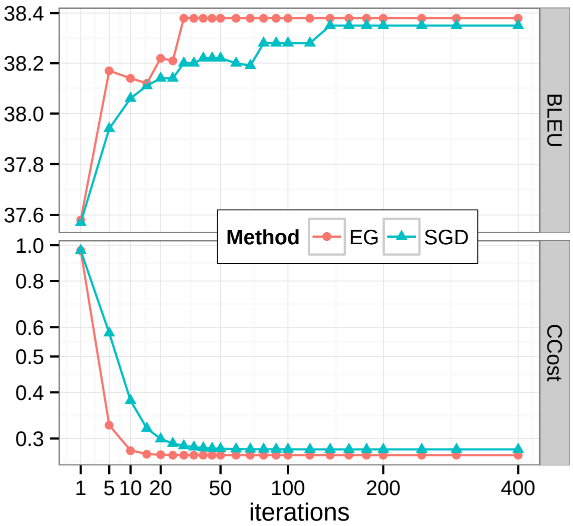


Figure 3: Analysis on convergence and performance comparing SOFTMAX and EG algorithms from BTEC zh→en translation. Both algorithms use momentum and step size 50.

Beyond achieving similar or better translations to re-ranking, note that EG is simpler in implementation, as it does not require $k$best lists, weight tuning and so forth. Instead this is replaced with iterative gradient descent. The run-time of the two methods are comparable, when reranking uses modest k, however EG can be considerably faster when k is large, as is typically done to extract the full benefit from re-ranking. This performance difference is a consequence of GPU acceleration of the dense vector operations in EG inference.

Computational Efficiency. We also quantify the computational efficiency of the proposed decoding approach. Benchmarking on a GPU Titan X for decoding BTEC zh→en, the average time per sentence is 0.02 secs for greedy, 0.07s for beam=5, 0.11s for beam=10, and 3.1s for relaxed EG decoding, which uses an average of 35 EG iterations. The majority of time in the EG algorithm is in the forward and backward passes, taking 30% and 67% of the time, respectively. Our imple-

	BTEC zh → en	TEDTalks de → en	WMT de → en
$\text{gdec}_{\text{left-to-right}}$	35.98	23.16	24.41
$\text{gdec}_{\text{right-to-left}}$	35.86	21.95	23.59
$\text{EGdec}_{\text{greedy init}}$	36.34	23.28	24.63
+bidir.	**36.67**	**23.91**	**25.37**†
+bilingual	**36.88**†	**24.01**†	**25.21**
$\text{bdec}_{\text{left-to-right}}$	38.02	23.95	26.69
$\text{bdec}_{\text{right-to-left}}$	37.38	23.13	26.11
$\text{EGdec}_{\text{beam init}}$	38.38	24.02	26.66
+bidir.	**39.13**†	**24.72**†	**27.34**†
+bilingual	38.25	**24.60**	26.82

Table 3: The BLEU evaluation results across evaluation datasets for EG algorithm variants against the baselines; **bold**: statistically significantly better than the best greedy or beam baseline, †: best performance on dataset.

mentation was not optimised thoroughly, and it is likely that it could be made significantly faster, which we defer to future research.

Main Results. Table 3 shows our experimental results across all datasets, evaluating the EG algorithm and its variants.[8] For the EG algorithm with greedy initialisation (top), we see small but consistent improvements in terms of BLEU. Beam initialisation led to overall higher BLEU scores, and again demonstrating a similar pattern of improvements, albeit of a lower magnitude, over the initialisation values.

Next we evaluate the capability of our inference method with extended NMT models, where approximate algorithms such as greedy or beam search are infeasible. With the **bidirectional** ensemble, we obtained the statistically significant BLEU score improvements compared to the unidirectional models, for either greedy or beam initialisation. This is interesting in the sense that the unidirectional right-to-left model always performs worse than the left-to-right model. However, our method with bidirectional ensemble is capable of combining their strengths in a unified setting. For the **bilingual** ensemble, we see similar effects, with better BLEU score improvements in most cases, albeit of a lower magnitude, over the bidirectional one. This is likely to be due to a disparity with the training condition for the models, which were learned independently of one another.

Overall, decoding in extended NMT models leads to performance improvements compared to baseline methods. This is one of the main findings in this work, and augurs well for its extension to other global model variants.

6 Related Work

Decoding (inference) for neural models is an important task; however, there is limited research in this space perhaps due to the challenging nature of this task, with only a few works exploring some extensions to improve upon them. The most widely-used inference methods include sampling (Cho, 2016), greedy and beam search (Sutskever et al., 2014; Bahdanau et al., 2015, *inter alia*), and reranking (Birch, 2016; Li and Jurafsky, 2016).

Cho (2016) proposed to perturb the neural model by injecting noise(s) in the hidden transition function of the conditional recurrent neural language model during greedy or beam search, and execute multiple parallel decoding runs. This strategy can improves over greedy and beam search; however, it is not clear how, when and where noise should be injected to be beneficial. Recently, Wiseman and Rush (2016) proposed beam search optimisation while *training* neural models, where the model parameters are updated in case the gold standard falls outside of the beam. This exposes the model to its past incorrect predicted labels, hence making the training more robust. This is orthogonal to our approach where we focus on the decoding problem with a pre-trained model.

Reranking has also been proposed as a means of global model combination: Birch (2016) and Li and Jurafsky (2016) re-rank the left-to-right decoded translations based on the scores of a right-to-left model, learning to more diverse translations. Related, Li et al. (2016) learn to adjust the beam diversity with reinforcement learning.

Perhaps most relevant is Snelleman (2016), performed concurrently to this work, who also proposed an inference method for NMT using linear relaxation. Snelleman's method was similar to our SGD approach, however he did not manage to outperform beam search baselines with an encoder-decoder. In contrast we go much further, proposing the EG algorithm, which we show works much more effectively than SGD, and demonstrate how this can be applied to inference in an attentional

[8] Due to the space constraints, we report results for the EG algorithm only. See also translation examples in Figure 4.

BTEC zh→en	
Source	我确定我昨天给旅馆打过电话并且做了预定。
Reference	i am sure that i called the hotel yesterday and made a reservation .
beam dec (l2r)	i 'm sure i called the hotel reservation and i made a reservation .
beam dec (r2l)	i 'm sure i made this hotel reservation and made a reservation .
rerank +bidir.	i 'm sure i called the hotel reservation and i made a reservation .
rerank +biling.	i 'm sure i called the hotel reservation and i made a reservation .
EGdec	i 'm sure i called the hotel yesterday and i made a reservation .
+bidir.	i 'm sure i called the hotel yesterday and i made a reservation .
+biling.	i 'm sure i called the hotel yesterday and i made a reservation .
TED Talks de→en	
Source	wir sind doch alle gute bürger der sozialen medien , bei denen die währung neid ist . stimmt ' s ?
Reference	i mean , we 're all good citizens of social media , are n't we , where the currency is envy ?
beam dec (l2r)	we 're all great UNK of social media , where the currency is envy . right ?
beam dec (r2l)	we 're all good citizens in social media , which is where that is envy . right ?
rerank +bidir.	we 're all good citizens of social media , where the currency is envy . right ?
rerank +biling.	we 're all good citizens of social media , where the currency is envy . right ?
EGdec	we 're all great UNK of social media , where the currency is envy . right ?
+bidir.	we 're all good UNK of social media , where the currency is envy . right ?
+biling.	we 're all good citizens of social media , where the currency is envy . right ?
WMT de→en	
Source	neben dem wm-titel 2007 und dem gewinn der champions league 2014 holte er 2008 (hsg nordhorn) und 2010 (tbv lemgo) den ehf-pokal .
Reference	besides the 2007 world championship he also won the champions league in may and the ehf-cup in 2008 (hsg nordhorn) and 2010 (tbv lemgo) .
beam dec (l2r)*	in addition to the title 2007 in 2007 and the win of the champions league 2014 in 2008 (hsg nordhorn) and 2010 (tbv lemgo) , he won the ehf cup .
beam dec (r2l)	in addition to the world championship title 2007 and winning the champions league in 2014 , he won the ehf-cup in 2008 (hsg nordhorn) and 2010 (tbv lemgo) .
EGdec	in addition to the title 2007 in 2007 and the win of the champions league 2014 in 2008 (hsg nordhorn) and 2010 (tbv lemgo) , he won the ehf cup .
+bidir.	in addition to the title championship in 2007 and the win of the champions league 2014 in 2008 (hsg nordhorn) and 2010 (tbv lemgo) , he won the ehf cup .
+biling.	in addition to the world title title 2007 and the win of the champions league 2014 in 2008 (hsg nordhorn) and 2010 (tbv lemgo) , he won the ehf cup .

Figure 4: Translation examples generated by the models. *: reranking with bidirectional (+bidir.) and bilingual (+biling.) produced the same translation string.

encoder-decoder. Moreover, we demonstrate the utility of related optimisation for inference over global ensembles of models, resulting in consistent improvements in search error and end translation quality.

Recently, relaxation techniques have been applied to deep models for training and inference in text classification (Belanger and McCallum, 2016; Belanger et al., 2017), and fully differentiable training of sequence-to-sequence models with scheduled-sampling (Goyal et al., 2017). Our work has applied the relaxation technique specifically for *decoding* in NMT models.

7 Conclusions

This work presents the first attempt in formulating decoding in NMT as a continuous optimisation problem. The core idea is to drop the integrality (i.e. one-hot vector) constraint from the prediction variables and allow them to have soft assignments within the probability simplex while minimising the loss function produced by the neural model. We have provided two optimisation algorithms – exponentiated gradient (EG) and stochastic gradient descent (SGD) – for optimising the resulting contained optimisation problem, where our findings show the effectiveness of EG compared to SGD. Thanks to our framework, we have been able to decode when intersecting left-to-right and right-to-left as well as source-to-target and target-to-source NMT models. Our results show that our decoding framework is effective and leads to substantial improvements in translations generated from the intersected models, where the typical greedy or beam search algorithms are not applicable.

This work raises several compelling possibilities which we intend to address in future work, such as improving decoding speed, integrating additional constraints such as word coverage and fertility into *decoding*,[9] and applying our method to other intractable structured prediction problems.

[9] These constraints have only been used for training in the previous works (Cohn et al., 2016; Mi et al., 2016).

Acknowledgments

We thank the reviewers for valuable feedbacks and discussions. Cong Duy Vu Hoang is supported by Australian Government Research Training Program Scholarships at the University of Melbourne, Australia. This work was supported by the Australian Research Council through their Discovery and Future Fellowship programmes.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proc. of 3rd International Conference on Learning Representations (ICLR2015)*.

D. Belanger, B. Yang, and A. McCallum. 2017. End-to-End Learning for Structured Prediction Energy Networks. In *Proceedings of the 34th International Conference on International Conference on Machine Learning - Volume 48*, ICML'17.

David Belanger and Andrew McCallum. 2016. Structured prediction energy networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 983–992. JMLR.org.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71.

Rico Sennrich; Barry Haddow; Alexandra Birch. 2016. Edinburgh Neural Machine Translation Systems for WMT 16. In *Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers. Berlin, Germany*.

M. Cettolo, J. Niehues, S. Stuker, L. Bentivogli, and M. Federico. 2014. Report on the 11th IWSLT Evaluation Campaign. In *Proc. of The International Workshop on Spoken Language Translation (IWSLT)*.

K. Cho. 2016. Noisy Parallel Approximate Decoding for Conditional Recurrent Language Model. *ArXiv e-prints*.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating Structural Alignment Biases into an Attentional Neural Translation Model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 876–885, San Diego, California. Association for Computational Linguistics.

Amir Globerson, Terry Y. Koo, Xavier Carreras, and Michael Collins. 2007. Exponentiated Gradient Algorithms for Log-linear Structured Prediction. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 305–312, New York, NY, USA. ACM.

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. 2017. Differentiable scheduled sampling for credit assignment. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, ACL'17, Vancouver, Canada.

A. Graves. 2013. Generating Sequences With Recurrent Neural Networks. *ArXiv e-prints*.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Jyrki Kivinen and Manfred K. Warmuth. 1997. Exponentiated Gradient Versus Gradient Descent for Linear Predictors. *Inf. Comput.*, 132(1):1–63.

J. Li and D. Jurafsky. 2016. Mutual Information and Diverse Decoding Improve Neural Machine Translation. *ArXiv e-prints*.

J. Li, W. Monroe, and D. Jurafsky. 2016. A Simple, Fast Diverse Decoding Algorithm for Neural Generation. *ArXiv e-prints*.

Z. C. Lipton, J. Berkowitz, and C. Elkan. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning. *ArXiv e-prints*.

Adam Lopez and Philip Resnik. 2006. Word-based alignment, phrase-based translation: Whats the link. In *Proceedings of 7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage Embedding Models for Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 955–960, Austin, Texas. Association for Computational Linguistics.

G. Neubig, C. Dyer, Y. Goldberg, A. Matthews, W. Ammar, A. Anastasopoulos, M. Ballesteros, D. Chiang, D. Clothiaux, T. Cohn, K. Duh, M. Faruqui, C. Gan, D. Garrette, Y. Ji, L. Kong, A. Kuncoro, G. Kumar, C. Malaviya, P. Michel, Y. Oda, M. Richardson, N. Saphra, S. Swayamdipta, and P. Yin. 2017. DyNet: The Dynamic Neural Network Toolkit. *ArXiv e-prints*.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings* *of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ning Qian. 1999. On the momentum term in gradient descent learning algorithms . *Neural Networks*, 12(1):145 – 151.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Emanuel Snelleman. 2016. Decoding neural machine translation using gradient descent. Master's thesis, Chalmers University of Technology, Gothenburg, Sweden.

David Sontag. 2010. *Approximate Inference in Graphical Models using LP Relaxations*. Ph.D. thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Taro Watanabe and Eiichiro Sumita. 2002a. Bidirectional Decoding for Statistical Machine Translation. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Taro Watanabe and Eiichiro Sumita. 2002b. Bidirectional decoding for statistical machine translation. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7.

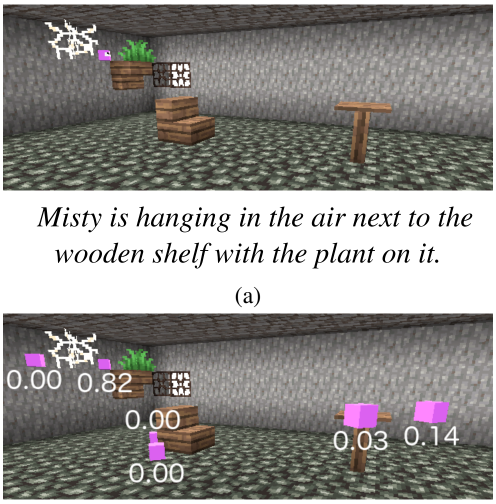
Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-Sequence Learning as Beam-Search Optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics.

Where is Misty? Interpreting Spatial Descriptors by Modeling Regions in Space

Nikita Kitaev and **Dan Klein**
Computer Science Division
University of California, Berkeley
{kitaev,klein}@cs.berkeley.edu

Abstract

We present a model for locating regions in space based on natural language descriptions. Starting with a 3D scene and a sentence, our model is able to associate words in the sentence with regions in the scene, interpret relations such as *on top of* or *next to,* and finally locate the region described in the sentence. All components form a single neural network that is trained end-to-end without prior knowledge of object segmentation. To evaluate our model, we construct and release a new dataset consisting of Minecraft scenes with crowdsourced natural language descriptions. We achieve a 32% relative error reduction compared to a strong neural baseline.



Misty is hanging in the air next to the wooden shelf with the plant on it.

(a)

(b)

Figure 1: An example from our dataset. (a) The Minecraft scene and its natural language description. (b) Given the choice between six possible locations, our model assigns the highest probability to the location consistent with the natural language description.

1 Introduction

In this work, we present a model for grounding spatial descriptors in 3D scenes. Consider interpreting the instructions: *Take the book and put it on the shelf.* One critical element of being able to interpret this sentence is associating the referring expression *the book* with the corresponding object in the world. Another important component of understanding the command above is translating the phrase *on the shelf* to a location in space. We call such phrases spatial descriptors. While spatial descriptors are closely related to referring expressions, they are distinct in that they can refer to locations even when there is nothing there. An intuitive way to model this is to reason over spatial regions as first-class entities, rather than taking an object-centric approach.

Following a long tradition of using game environments for AI, we adopt Minecraft as the setting for our work. Minecraft has previously been used for work on planning and navigation (Oh et al., 2016; Tessler et al., 2016), and we expand on this by using it for grounded language understanding. As a sandbox game, it can be used to construct a wide variety of environments that capture many interesting aspects of the real world. At the same time, it is easy to extract machine-interpretable representations from the game.

We construct a dataset of Minecraft scenes with natural-language annotations, and propose a task that evaluates understanding spatial descriptors. Our task is formulated in terms of locating a pink, cube-shaped character named Misty given a scene, a natural language description, and a set of locations to choose from. An example from our dataset is shown in Figure 1. The Minecraft scene representation does not provide ground-truth information about object identity or segmentation, reflecting the fact that perceptual ambiguity is always

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 157–166
Copenhagen, Denmark, September 7–11, 2017.

present in real-world scenarios. We do, however, assume the availability of 3D depth information (which, for real-world conditions, can be acquired using depth sensors such as RGBD cameras or LiDAR).

We propose and evaluate a neural network that combines convolutional layers operating over 3D regions in space with recurrent layers for processing language. Our model jointly learns to segment objects, associate them with words, and understand spatial relationships – all in an end-to-end manner. We compare with a strong neural baseline and demonstrate a relative error reduction of 32%.

The dataset and model described in this paper are available online.[1]

2 Related Work

Our task includes some of the same elements as referring-expression generation and interpretation. Past work on these tasks includes Golland et al. (2010), Krishnamurthy and Kollar (2013), Socher et al. (2014) and Kazemzadeh et al. (2014). A key difference is that spatial descriptors (as modeled in this paper) refer to locations in space, rather than to objects alone. For example, Krishnamurthy and Kollar (2013) convert natural language to a logical form that is matched against image segments, an approach that is only capable of reasoning about objects already present in the scene (and not skipped over by the segmentation process). Our model's ability to reason over spatial regions also differentiates it from past approaches to tasks beyond referring expressions, such as the work by Tellex et al. (2011) on natural-language commanding of robots. Recent work by Hu et al. (2016) on interpreting referring expressions can capture relationships between objects, relying on the construction of (subject, object, relation) tuples. Their model is limited in that it can only handle one such tuple per utterance. Our model does not have such a restriction, and it additionally expands to a 3D setting.

Our task is also related to work on Visual Question Answering, or VQA (Agrawal et al., 2015). While VQA uses free-form textual answers, our task places targeted emphasis on spatial reasoning by requiring outputs to be locations in the scene. Spatial reasoning remains an important capability for VQA systems, and is one of the elements featured in CLEVR (Johnson et al., 2016), a diagnostic dataset for VQA. Like in our dataset, visual percepts in CLEVR are based on machine-generated scenes. CLEVR also makes use of machine-generated language, while all language in our dataset is written by humans.

Another related task in NLP is spatial role labeling, which includes the identification of spatial descriptors and the assigning of roles to each of their constituent words. This task was studied by Kordjamshidi et al. (2011) and led to the creation of shared tasks such as SpaceEval (Pustejovsky et al., 2015). Our setting differs in that we consider grounded environments instead of studying text in isolation, and evaluate on task performance rather than logical correctness of interpretation.

Spatial descriptors are also present in the task of generating 3D scenes given natural language descriptions. Compared to a recent model by Chang et al. (2017) for scene generation, our model works with lower-level 3D percepts rather than libraries of segmented and tagged objects. We are also able to incorporate learning of vocabulary, perception, and linguistic structure into a single neural network that is trainable end-to-end.

3 Task

At its core, the ability to understand spatial descriptors can be formulated as mapping from a natural-language description to a particular location in space. In Figure 1, we show an instance of our task, which consists of the following components:

- W: a perceptual representation of the world
- x: the natural language description
- $\{y_1, y_2, \ldots, y_n\}$: the candidate set of locations that are under consideration
- $y^\star$: the true location that is being referred to in the scene

Given W and x, a model must select which candidate location y_i best matches the description x.

We will address the particulars of the above representation as we discuss the process for constructing our dataset. Each example $(W, x, \{y_1, \ldots, y_n\}, y^\star)$ in the dataset is made by generating a Minecraft scene (Section 3.1) and selecting a location as the target of description (Section 3.2). We then crowdsource natural language descriptions of the target location in space. To better anchor the language, we populate

[1] https://github.com/nikitakit/voxelworld

the target location with a cube-shaped character we name Misty, and ask workers to describe Misty's location (Section 3.3). We repeat this process for each example in the dataset.

3.1 Scene Generation and Representation

Each of our Minecraft scenes is set in a randomly-generated room. We select a random size for this room, and then populate it with a variety of objects. We include objects that can be placed on the floor (e.g. tables), mounted on the wall (e.g. torches), embedded in the wall (e.g. doors), or hanging from the ceiling (e.g. cobwebs).

We then discard ground-truth knowledge about object segmentation or identity in the process of saving our dataset. This allows our task to evaluate not only models' capacity for understanding language, but also their ability to integrate with perceptual systems. One way of approximating real-world observations would be to take a screenshot of the scene – however, a 2D projection does not provide all of the spatial information that a language user would reasonably have access to. We would like to use a 3D encoding instead, and Minecraft naturally offers a low-level (albeit low-resolution) voxel-based representation that we adopt for this work.

Each Minecraft world W is encoded as a 3D grid of voxels, where a voxel may be empty or contain a particular type of "block," e.g. stone or wood. In general, what humans would interpret as single objects will be made of multiple Minecraft blocks – for example, the table in Figure 1 consists of a "wooden pressure plate" block on top of a "wooden fencepost" block. These same blocks can be used for other purposes as well: the "wooden fencepost" block is also part of fences, lamp-posts, and pillars, while the "wooden pressure plate" block can form shelves, countertops, as well as being placed on the ground to detect when something walks over it. We construct our Minecraft scenes specifically to include examples of such re-use, so that models capable of achieving high performance on this task must demonstrate the capacity to work without ground-truth segmentation or perfect object labeling.

The voxel-grid 3D representation is not specific to the virtual Minecraft setting: it is equally applicable to real-world data where depth information is available. The main difference is that each voxel would need to be associated with a feature vector rather than a block type. One use of such a representation is in Maturana and Scherer (2015)'s work on object classification from data collected with RGBD cameras and LiDAR, which uses a 3D convolutional neural network over a voxel grid. We do not explicitly handle occlusion in this work, but we imagine that real-world extensions can approach it using a combination of multi-viewpoint synthesis, occlusion-aware voxel embeddings, and restricting the set of voxels considered by the model.

3.2 Location Sampling

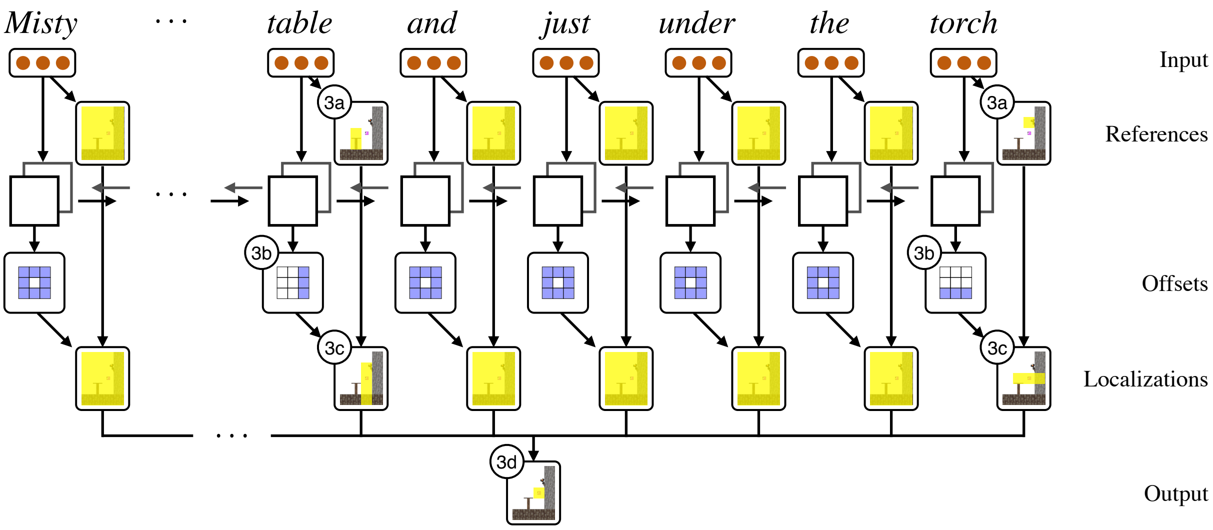
After constructing a scene with representation W, we proceed to sample a location $y^\star$ in the scene. Given our voxel-based scene representation, our location sampling is at voxel granularity. The candidate set we sample from, $\{y_1, \ldots, y_n\}$, consists of empty voxels in the scene. Locations that occur in the middle of a large section of empty space are hard to distinguish visually and to describe precisely, so we require that each candidate y_i be adjacent to at least one object.

3.3 Natural Language Descriptions

For each scene-location pair $(W, y^\star)$ we crowdsource a natural language description x.

The choice of prompt for human annotators is important in eliciting good descriptions. At the location we are asking workers to refer to, we insert a pink-colored cube that we personify and name "Misty." We then ask workers to describe Misty's location such that someone can find her if she were to turn invisible. Having a visually salient target helps anchor human perception, which is why we chose a pink color that contrasts with other visual elements in the scene. We make sure to emphasize the name "Misty" in the instructions, which results in workers almost always referring to Misty by name or with the pronoun *she*. This avoids having to disambiguate a myriad of generic descriptions (*the pink block, the block, the target,* etc.) for what is fundamentally an artificial construct.

To make sure that humans understand the 3D structure of the scene as they describe it, we give them access to a 3D view of the environment and require that they move the camera before submitting a description. This helped increase the quality of our data.



Misty is to the right of the table and just under the torch.

Figure 2: Our model architecture. Note that while the schematic illustrations are shown in 2D, our actual model operates in 3D. Zoomed-in versions of the marked references, offsets, localizations, and output are shown in Figure 3.

4 Model

We next present our model for this task. Our model architecture is shown in Figure 2, with some of the quantities it operates over highlighted in Figure 3. Throughout this section, we will use the example description *Misty is to the right of the table and just under the torch.* Note that while the accompanying scene illustrations are shown in 2D for visual clarity, our actual model operates in 3D and on larger scene sizes.

Our model first associates words with regions in the world. There is no notion of object segmentation in the dataset, so the references it produces are just activations over space given a word. Activations are computed for all words in the sentence, though they will only be meaningful for words such as *table* and *torch* (Figure 3a). Our model next determines the spatial relationships between referenced objects and Misty, using information provided by context words such as *right* and *under.* These relationships are represented as 3D convolutional offset filters (Figure 3b). For each word, its reference and offset filter are convolved to get a localization, i.e. an estimate of Misty's location (Figure 3c). Finally, our model aggregates localizations across all words in the sentence, combining the information provided by the phrases *to the right of the table* and *just under the torch* (Figure 3e).

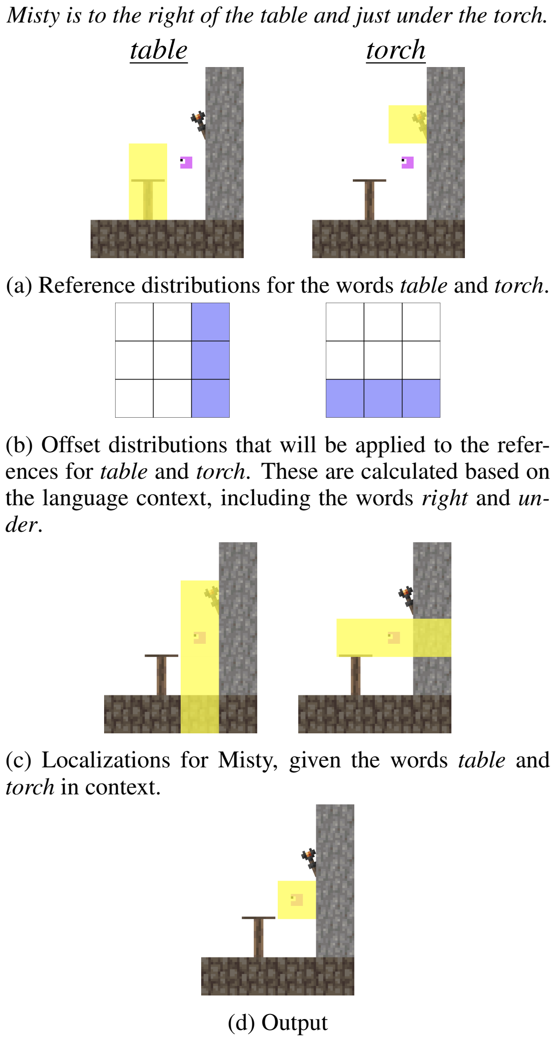
The following sections describe in more detail how references (Section 4.1), offsets (Section 4.2), and localizations (Section 4.3) are computed.

4.1 Input and References

The first component of our model is responsible for associating words with the voxels that they refer to. It assigns a real-valued score $s(x_t, y)$ to each pair consisting of word x_t and voxel coordinate y.

High scores correspond to high compatibility; for any given word, we can visualize the set $s(x_t, \cdot)$ of scores assigned to different voxels by interpreting it as logits that encode a probability distribution over blocks in the scene. In the example, the word *table* would ideally be matched to the uniform reference distribution over blocks that are part of a table, and similarly for the word *torch* (Figure 3a).

The word-voxel scores are computed by combining word and block embeddings. To take advantage of additional unsupervised language and world data, we start with pretrained word embeddings and context-aware location embeddings $f(W, y)$. The function f consists of the first two layers of a convolutional neural network that is pretrained on the task of predicting a voxel's identity given the 5x5x5 neighborhood around it. Since f fails to take into account the actual voxel's identity, we add additional embeddings V that only consider single blocks. The score is then computed as $s(x_t, y) = w_t^\top A f(W, y) + w_t^\top v_y$, where w_t is the word embedding and v_y is the single-block embedding. The parameter matrix


Misty is to the right of the table and just under the torch.
table
torch

(a) Reference distributions for the words *table* and *torch*.

(b) Offset distributions that will be applied to the references for *table* and *torch*. These are calculated based on the language context, including the words *right* and *under*.

(c) Localizations for Misty, given the words *table* and *torch* in context.

(d) Output

(e) Output distribution produced by intersecting the localizations for each word.

Figure 3: A schematic 2D depiction of the representations used throughout our neural network, zoomed in from Figure 2. Our model (a) matches words with objects in the scene, (b) determines offsets from landmark objects to Misty's location, (c) combines these pieces of information to form per-word localizations of Misty, and then (d) uses all localizations to guess Misty's location.

A and the single-block embedding matrix V are trained end-to-end with the rest of the model.

References are computed for all words in the sentence – including function words like *to* or *the*. To signify that a word does not refer to any objects in the world, the next layer of the network expects that we output a uniform distribution over all voxels. Outputting uniform distributions also serves as a good initialization for our model, so we set the elements of A and V to zero at the start of training (our pretrained word embeddings are sufficient to break symmetry).

4.2 Offsets

The per-word references described in Section 4.1 do not themselves indicate Misty's location. Rather, they are used in a spatial descriptor like *to the right of the table*. For every word, our model outputs a distribution over offset vectors that is used to redistribute scores from object locations to possible locations for Misty (Figure 3b). For example, if probability mass is placed on the "one-block-to-the-right" offset vector, this corresponds to predicting that Misty will be one block to the right of the voxels that a word refers to. Offset scores o_t are assigned based on the context the word x_t occurs in, which allows the model to incorporate information from words such as *right* or *under* in its decisions. This is accomplished by running a bidirectional LSTM over the embeddings w_t of the words in the sentence, and using its output to compute offset probabilities:

$$\begin{aligned}[z_0, z_1, \ldots] &= \text{BiLSTM}([w_0, w_1, \ldots]) \\ o'_t &= M z_t \\ o_t(i) &\propto \exp\left(o'_t(i)\right)\end{aligned}$$

Each set of offset scores o_t is reshaped into a 3x3x3 convolutional filter, except that we structurally disallow assigning any probability to the no-offset vector in the center. As a parameter-tying technique, the trainable matrix M is not full-rank; we instead decompose it such that the log-probability of an offset vector factors additively over the components in a cylindrical coordinate system.

4.3 Localizations and Output

For each word, the 3D tensor of word-voxel scores $s(x_t, \cdot)$ is convolved with the offset distribution o_t to produce a distribution of localizations for Misty, $d_t(y)$. A 2D illustration of the result is shown in Figure 3c. Localizations are then summed across all words in the sentence, resulting in a single score for each voxel in the scene (Figure 3e). These scores are interpreted as logits corresponding to a probability distribution over possible locations for Misty:

$$\begin{aligned}d_t(y) &= s(x_t, y) * o_t \\ p(y) &\propto \exp\left\{\sum_t d_t(y)\right\}\end{aligned}$$

Not all words will have localizations that provide information about Misty – for some words the localizations will just be a uniform distribution. We will refer to words that have low-entropy localizations as *landmarks*, with the understanding that being a landmark is actually a soft notion in our model.

Our offset filters o_t are much smaller than our voxel grid, which means that convolving any offset filter with a uniform reference distribution over the voxel grid will also result in a uniform localization distribution (edge effects are immaterial given the small filter size and the fact that Misty is generally not at the immediate edges of the scene). Conversely, given non-uniform references almost any set of offsets will result in a non-uniform localization. The architecture for computing references can output uniform references for function words (like *to* or *the*), but it lacks the linguistic context to determine when words refer to objects but should not be interpreted as landmarks (e.g. when they are part of exposition or a negated expression). We therefore include an additional not-a-landmark class that is softmax-normalized jointly with the offset vector distribution o_t. Probability assigned to this class subtracts from the probability mass for the true offset directions (and therefore from the localizations) – if this class receives a probability of 1, the corresponding localizations will not contribute to the model output.

4.4 Loss and Training

We use a softmax cross-entropy loss for training our model. During training, we find that it helps to not use the candidate set $\{y_1, y_2, \ldots, y_n\}$ and instead calculate a probability $p(y)$ for all blocks in the scene, including solid blocks that cannot possibly contain Misty (perhaps because this penalizes inferring nonsensical spatial relationships).

We run the Adam optimizer (Kingma and Ba, 2014) with step size 0.001 for 100 epochs using batch size 10. We keep an exponential moving average of our trainable parameters, which we save every two epochs. We then select the saved model that has the highest performance on our development set.

We perform several regularization and data augmentation techniques in order to achieve better generalization. Each time we sample a training example, we select a random 19x19x19 crop from the full scene (as long as Misty's location is not cropped out). We also disallow using the context-based block embeddings for the first 20 epochs by holding the parameter matrix A described in Section 4.1 fixed at zero, forcing the model to first learn to associate vocabulary with local features and only later expand to capture the compositional aspects of the environment.

For the natural language descriptions, all tokens are converted to lowercase as part of preprocessing. During training we apply word-level dropout (i.e. replacing words with an UNK token) in the LSTM responsible for computing offsets.

5 Evaluation

5.1 Evaluation Metric

In evaluating this task, we would like to use a metric that can provide meaningful comparison of our model with baseline and human performance. The set of all possible locations for Misty is large enough that it is hard even for a human to guess the correct block on the first try, especially when some descriptions are only precise to within 1 or 2 blocks. The size of this set also varies from scene to scene.

Therefore for our evaluation, we restrict the set $\{y_1, \ldots, y_n\}$ to 6 possible locations: Misty's true location and 5 distractors. This represents a less ambiguous problem that is much easier for humans, while also allowing for the evaluation of future models that may require an expensive computation for each candidate location considered. Our procedure for selecting the distractors is designed to ensure that we test both local and global scene understanding. Each set of six choices is constructed to consist of three clusters of two candidates each. Each cluster location is anchored to a landmark – we sample a landmark block adjacent to Misty and two additional landmark blocks from the entire scene, such that the pairwise distances between landmarks are at least 4 units. We then sample one distractor near Misty's landmark and two distractors near both of the other landmarks.

5.2 Dataset

To make our development and test sets, we construct this six-option variation from a subset of our collected data. For each such example we crowdsource two human solutions using Mechanical Turk. Examples where both humans answered correctly are partitioned into a development and a test set. This filtering procedure serves as our primary method of excluding confusing or uninformative descriptions from the evaluation con-

(a)	*When you come in the door, she's on the floor to the right, just in front of the flower.*
(b)	*Misty is facing to the right of the brown door.*
(c)	*If you were to walk through the door that is on the same wall as the table and plank of floating wood, Misty would be to the left of the door. She is eye level with the plank of wood and floating in front of it.*
(d)	*Misty is in the ground and she is front of door.*
(e)	*Misty is located under a table that is connected to the wall. She is at ground level.*

Table 1: Five natural-language descriptions sampled at random from our dataset.

ditions. We also collect a third human solution to each example in the development and test sets to get an independent estimate of human performance on our task. The final dataset consists of 2321 training examples, 120 dev set examples, and 200 test set examples.

The natural-language descriptions across the full dataset use a vocabulary of 1015 distinct tokens (case-insensitive but including punctuation). The average description length is 19.02 tokens, with a standard deviation of 10.00 tokens. The large spread partially reflects the fact that some people gave short descriptions that referenced a few landmarks, while others gave sequences of instructions on how to find Misty. As a point of comparison, the ReferIt dataset (Kazemzadeh et al., 2014) has a larger vocabulary of 9124 tokens, but a shorter average description length of 3.52 tokens (with a standard deviation of 2.67 tokens).

A random sampling of descriptions from our dataset is shown in Table 1.

5.3 Quantitative Results

Quantitative results are shown in Table 2. Our evaluation metric is constructed such that there is an easily interpretable random baseline. We also evaluate a strong neural baseline that uses an approach we call Seq2Emb. This baseline converts the sentence into a vector using a bidirectional LSTM encoder, and also assigns vector embeddings to each voxel using a two-layer convolutional neural network. The voxel with an embedding that most closely matches the sentence embedding is chosen as the answer.

Our model achieves noticeable gains over the baseline approaches. At the same time, there remains a gap between our model and individual human performance. We see this as an indication that we have constructed a task with appropriate difficulty: it is approachable by building on the current state-of-the-art in machine learning and NLP, while presenting challenges that can motivate continued work on understanding language and how it relates to descriptions of the world.

	Dev Set	Test Set
Random Baseline	16.67	16.67
Seq2Emb Baseline	52.50	44.50
Our Model	67.50	62.50
Human	85.83	87.50

Table 2: Success rates for our dataset split. Our model is able to outperform a strong neural baseline (Seq2Emb).

	% correct
Full model	**67.5**
−contextual block embeddings	65.0
−LSTM (use 3-word convolutions instead)	62.5
−language-dependent spatial operators	61.7

Table 3: Development set results for our full model and three independent ablations.

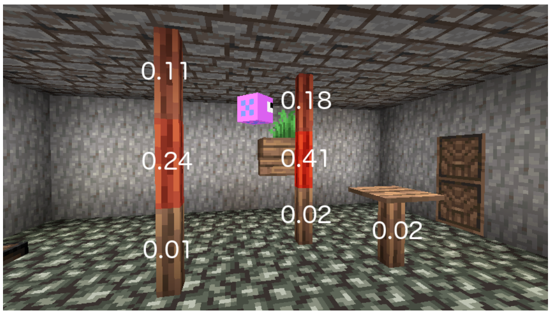
5.4 Ablation Study

We next conduct an ablation study to evaluate the contribution of the individual elements in our model. Our ablation results on the development set are shown in Table 3.

In our first ablation, we remove the compositional block embeddings that make use of multiple blocks. The resulting performance drop of 2.5% reflects the fact that our model uses multi-block information to match words with objects.

We next replace the LSTM in our full model with a 3-word-wide convolutional layer. A single word of left- and right-context provides limited ability to incorporate spatial descriptor words like *left* and *right*, or to distinguish landmarks used to locate Misty from words providing exposition about the scene. This ablation solves 5% fewer examples than our full model, reflecting our LSTM's ability to capture such phenomena.

Finally, we try holding the distribution over offset vectors fixed, by making it a trainable variable rather than a function of the language. This corresponds to enforcing the use of only one spatial



Misty is floating in the middle of the room. She in the upper half of the room, between the two ***poles****.*

Figure 4: Reference distribution representing our model's belief of which blocks the word *poles* refers to. Our model assigns the majority of the probability mass to the poles, while ignoring a table leg that is made of the same block type. Note that the seven numbers overlaid on top account for more than 99% of the total probability mass, and that each of the remaining blocks in the scene has a probability of at most 0.025%.

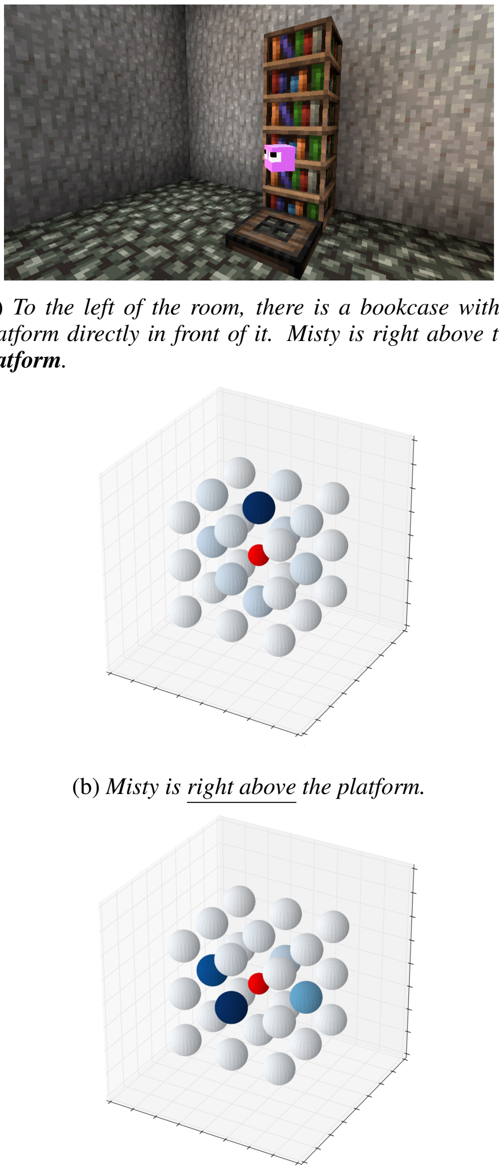
operator that roughly means 'near.' We retain the LSTM for the sole purpose of assigning a score to the not-a-landmark class, meaning that contextual information is still incorporated in the decision of whether to classify a word as a landmark or not. The resulting accuracy is 5.8% lower than our full model, which makes this the worst-performing of our ablations. These results suggest that the ability to infer spatial directions is important to our model's overall performance.

5.5 Qualitative Examination

The modular design of our model allows us to examine the individual behavior of each component in the network, which we explore in this section.

We find that our algorithm is able to learn to associate words with the corresponding voxels in the world. Figure 4 shows the reference distribution associated with the word *poles*, which is constructed by applying a softmax operation to the word-voxel scores for that word. Our algorithm is able to correctly segment out the voxels that are a part of the pole. Moreover, the table on the right side of the scene has a table leg made of the same block type as the pole – and yet, it is is assigned a low probability. This shows that our model is capable of representing compositional objects, and can learn to do so in an end-to-end manner.

We next examine the offset distributions computed by our model. Consider the scene and description shown in Figure 5a. The offset vector distribution at the word *platform*, shown in Figure 5b, shows that the model assigns high proba-

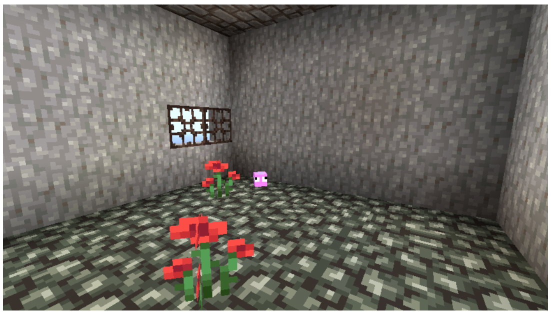


(a) *To the left of the room, there is a bookcase with a platform directly in front of it. Misty is right above the* ***platform****.*

(b) *Misty is right above the platform.*

(c) *Misty is in front of the platform.*

Figure 5: Effects of language context on offset vector distributions. In (a), we show the scene and its description. In (b), we visualize the offset vector distribution at the word *platform*, i.e. the 3D convolutional filter that is applied after finding the platform location. The red dot that indicates the center of the filter will be matched with the platform location. In (c), we have artificially replaced the words *right above* with *in front of*, resulting in a substantial change to this distribution.



Misty is between the wall and the flowers that are close to the corner.

Figure 6: Our algorithm interprets this sentence as *Misty is near the wall and the flowers and close to the corner.* This intersective interpretation is sufficient to correctly guess Misty's location in this scene (as well as others in the dataset).

bility to Misty being above the platform. In Figure 5c, we show the effects of replacing the phrase *right above* with the words *in front of*. This example illustrates our model's capacity for learning spatial directions. We note that the offset distribution given the phrase *in front of* is not as peaked as it is for *right above*, and that distributions for descriptions saying *left* or *right* are even less peaked (and are mostly uniform on the horizontal plane). One explanation for this is the ambiguity between speaker-centric and object-centric reference frames. The reference frame of our convolutional filters is the same as the initial camera frame for our our annotators, but this may not be the true speaker-centric frame because we mandate that annotators move the camera before submitting a description.

We next highlight our model's ability to incorporate multiple landmarks in making its decisions. Consider the scene and description shown in Figure 6. The room has four walls, two flowers, and four corners – no single landmark is sufficient to correctly guess Misty's location. Our model is able to localize the flowers, walls, and corners in this scene and intersect them to locate Misty. Strictly speaking, this approach is not logically equivalent to applying a two-argument *between* operator and recognizing the role of *that* as a relativizer. This is a limitation of our specific model, but the general approach of manipulating spatial region masks need not be constrained in this way. It would be possible to introduce operations into the neural network to model recursive structure in the language. In practice, however, we find that the intersective interpretation suffices for many of the descriptions that occur in our dataset.

6 Conclusion

In this paper, we define the task of interpreting spatial descriptors, construct a new dataset based on Minecraft, and propose a model for this task. We show that convolutional neural networks can be used to reason about regions in space as first-class entities. This approach is trainable end-to-end while also having interpretable values at the intermediate stages of the neural network.

Our architecture handles many of the linguistic phenomena needed to solve this task, including object references and spatial regions. However, there is more work to be done before we can say that the network completely understands the sentences that it reads. Our dataset can be used to investigate future models that expand to handle relativization and other recursive phenomena in language.

Acknowledgments

We thank the many workers on Mechanical Turk who contributed to the creation of our dataset.

This work was made possible by the open source tooling developed around and inspired by Minecraft; in particular we would like to thank the developers of the `voxel.js` project and associated plugins, as well as the developers of `mcedit2`.

Nikita Kitaev is supported by an NSF Graduate Research Fellowship. This research was supported by DARPA through the XAI program.

References

Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, and Devi Parikh. 2015. VQA: Visual Question Answering. *arXiv:1505.00468 [cs]*.

Angel X. Chang, Mihail Eric, Manolis Savva, and Christopher D. Manning. 2017. SceneSeer: 3d Scene Design with Natural Language. *arXiv:1703.00050 [cs]*.

Dave Golland, Percy Liang, and Dan Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *EMNLP*, pages 410–419.

Ronghang Hu, Marcus Rohrbach, Jacob Andreas, Trevor Darrell, and Kate Saenko. 2016. Modeling Relationships in Referential Expressions with Compositional Modular Networks. *arXiv:1611.09978 [cs]*.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross

Girshick. 2016. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. *arXiv:1612.06890 [cs].*

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L. Berg. 2014. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *EMNLP*, pages 787–798.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs].*

Parisa Kordjamshidi, Martijn Van Otterlo, and Marie-Francine Moens. 2011. Spatial role labeling: Towards extraction of spatial relations from natural language. *ACM Transactions on Speech and Language Processing (TSLP)*, 8(3):4.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1:193–206.

Daniel Maturana and Sebastian Scherer. 2015. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE.

Junhyuk Oh, Valliappa Chockalingam, Satinder Singh, and Honglak Lee. 2016. Control of Memory, Active Perception, and Action in Minecraft. *arXiv:1605.09128 [cs].*

James Pustejovsky, Parisa Kordjamshidi, Marie-Francine Moens, Aaron Levine, Seth Dworman, and Zachary Yocum. 2015. SemEval-2015 Task 8: SpaceEval. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 884–894.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Stefanie A. Tellex, Thomas Fleming Kollar, Steven R. Dickerson, Matthew R. Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*.

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J. Mankowitz, and Shie Mannor. 2016. A Deep Hierarchical Approach to Lifelong Learning in Minecraft. *arXiv:1604.07255 [cs].*

Continuous Representation of Location for Geolocation and Lexical Dialectology using Mixture Density Networks

Afshin Rahimi **Timothy Baldwin** **Trevor Cohn**
School of Computing and Information Systems
The University of Melbourne
arahimi@student.unimelb.edu.au
{tbaldwin,t.cohn}@unimelb.edu.au

Abstract

We propose a method for embedding two-dimensional locations in a continuous vector space using a neural network-based model incorporating mixtures of Gaussian distributions, presenting two model variants for text-based geolocation and lexical dialectology. Evaluated over Twitter data, the proposed model outperforms conventional regression-based geolocation and provides a better estimate of uncertainty. We also show the effectiveness of the representation for predicting words from location in lexical dialectology, and evaluate it using the DARE dataset.

1 Introduction

Geolocation is an essential component of applications such as traffic monitoring (Emadi et al., 2017), human mobility pattern analysis (McNeill et al., 2016; Dredze et al., 2016) and disaster response (Ashktorab et al., 2014; Wakamiya et al., 2016), as well as targeted advertising (Anagnostopoulos et al., 2016) and local recommender systems (Ho et al., 2012). Although Twitter provides users with the means to geotag their messages, less than 1% of users opt to turn on geotagging, so third-party service providers tend to use profile data, text content and network information to infer the location of users. Text content is the most widely used source of geolocation information, due to its prevalence across social media services.

Text-based geolocation systems use the geographical bias of language to infer the location of a user or message using models trained on geotagged posts. The models often use a representation of text (e.g. based on a bag-of-words, convolutional or recurrent model) to predict the location either in real-valued latitude/longitude coordinate space or in discretised region-based space, using regression or classification, respectively. Regression models, as a consequence of minimising squared loss for a unimodal distribution, predict inputs with multiple targets to lie between the targets (e.g. a user who mentions content in both NYC and LA is predicted to be in the centre of the U.S.). Classification models, while eliminating this problem by predicting a more granular target, don't provide fine-grained predictions (e.g. specific locations in NYC), and also require heuristic discretisation of locations into regions (e.g. using clustering).

Mixture Density Networks ("MDNs": Bishop (1994)) alleviate these problems by representing location as a mixture of Gaussian distributions. Given a text input, an MDN can generate a mixture model in the form of a probability distribution over all location points. In the example of a user who talks about both NYC and LA, e.g., the model will predict a strong Gaussian component in NYC and another one in LA, and also provide an estimate of uncertainty over all the coordinate space.

Although MDNs are not new, they have not found widespread use in *inverse* regression problems where for a single input, multiple correct outputs are possible. Given the integration of NLP technologies into devices (e.g. phones or robots with natural language interfaces) is growing quickly, there is a potential need for interfacing language with continuous variables as input or target. MDNs can also be used in general text regression problems such as risk assessment (Wang and Hua, 2014), sentiment analysis (Joshi et al., 2010) and loan rate prediction (Bitvai and Cohn, 2015), not only to improve prediction but also to use the mixture model as a representation for the continuous variables. We apply MDNs to geotagged Twitter data in two different settings: (a) predicting location given text; and (b) predicting text given location.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 167–176
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Geotagged text content is not only useful in geolocation, but can also be used in lexical dialectology. Lexical dialectology is (in part) the converse of text-based geolocation (Eisenstein, 2015): instead of predicting location from language, language (e.g. dialect terms) are predicted from a given location. This is a much more challenging task as the lexical items are not known beforehand, and there is no notion of dialect regions in the continuous space of latitude/longitude coordinates. A lexical dialectology model should not only be able to predict dialect terms but also be able to automatically learn dialect regions.

In this work, we use bivariate Gaussian mixtures over geotagged Twitter data in two different settings, and demonstrate their use for geolocation and lexical dialectology. Our contributions are as follows: (1) we propose a continuous representation of location using bivariate Gaussian mixtures; (2) we show that our geolocation model outperforms regression-based models and achieves comparable results with classification models, but with added uncertainty over the continuous output space; (3) we show that our lexical dialectology model is able to predict geographical dialect terms from latitude/longitude input with state-of-the-art accuracy; and (4) we show that the automatically learned Gaussian regions match expert-generated dialect regions of the U.S.[1]

2 Related Work

2.1 Text-based Geolocation

Text-based geolocation models are defined as either a regression or a classification problem. In regression geolocation, the model learns to predict a real-valued latitude/longitude from a text input. This is a very challenging task for data types such as Twitter, as they are often heavily biased toward population centres and urban areas, and far from uniform. As an example, *Norwalk* is the name of a few cities in the U.S among which Norwalk, California (West Coast) and Norwalk, Connecticut (East Coast) are the two most populous cities. Assuming that occurrences of the city's name are almost equal in both city regions within the training set, a trained regression-based geolocation model given *Norwalk* as input, would geolocate it to a point in the middle of the U.S. instead of choosing one of the cities. In the machine learning literature, regression problems where there are multiple real-valued outputs for a given input are called *inverse problems* (Bishop, 1994). Here, standard regression models predict an average point in the middle of all training target points to minimise squared error loss. Bishop (1994) proposes density mixture networks to model such inverse problems, as we discuss in detail in Section 3.

In addition, non-Bayesian interpretations of regression models, which are often used in practice, don't produce any prediction of uncertainty, so other than the predicted point, we have little idea where else the term could have high or low probability. Priedhorsky et al. (2014) propose a Gaussian Mixture Model (GMM) approach instead of squared loss regression, whereby they learn a mixture of bivariate Gaussian distributions for each individual n-gram in the training set. During prediction, they add the Gaussian mixture of each n-gram in the input text, resulting in a new Gaussian mixture which can be used to predict a coordinate with associated uncertainty. To add the mixture components they use a weighted sum, where the weight of each n-gram is assigned by several heuristic features. Learning a GMM for each n-gram is resource-intensive if the size of the training set — and thus the number of n-grams — is large.

Assuming sufficient training samples containing the term *Norwalk* in the two main, a trained classification model would, given this term as input, predict a probability distribution over all regions, and assign higher probabilities to the regions containing the two major cities. The challenge, though, is that the coordinates in the training data must first be partitioned into regions using administrative regions (Cheng et al., 2010; Hecht et al., 2011; Kinsella et al., 2011; Han et al., 2012, 2014), a uniform grid (Serdyukov et al., 2009), or a clustering method such as a k-d tree (Wing and Baldridge, 2011) or K-means (Rahimi et al., to appear). The cluster/region labels can then be used as targets. Once we have a prediction about where a user is more likely to be from, there is no more information about the coordinates inside the predicted region. If a region that contains Wyoming is predicted as the home location of a user, we have no idea which city or county within Wyoming the user might be from, unless we retrain the model using a more fine-grained discretisation or a hierarchical discretisation (Wing and Baldridge, 2014), which is both time-consuming and challenging due to data

[1] Code available at https://github.com/afshinrahimi/geomdn

sparseness.

2.2 Lexical Dialectology

The traditional linguistic approach to lexical dialectology is to find the geographical distributions of known contrast sets such as $\{\textit{you}, \textit{yall}, \textit{yinz}\}$: (Labov et al., 2005; Nerbonne et al., 2008; Gonçalves and Sánchez, 2014; Doyle, 2014; Huang et al., 2015; Nguyen and Eisenstein, to appear). This usually involves surveying a large geographically-uniform sample of people from different locations and analysing where each known alternative is used more frequently. Then, the coordinates are clustered heuristically into dialect regions, based on the lexical choices of users in each region relative to the contrast set. This processing is very costly and time-consuming, and relies critically on knowing the lexical alternatives a priori. For example, it would require a priori knowledge of the fact that people in different regions of the US use *pop* and *soda* to refer to the same type of drink, and a posteriori analysis of the empirical geographical distribution of the different terms. Language, particularly in social media and among younger speakers, is evolving so quickly, in ways that can be measured over large-scale data samples such as Twitter, that we ideally want to be able to infer such contrast sets dynamically. The first step in automatically collecting dialect words is to find terms that are disproportionately distributed in different locations. The two predominant approaches to this problem are model-based (Eisenstein et al., 2010; Ahmed et al., 2013; Eisenstein, 2015) and through the use of statistical metrics (Monroe et al., 2008; Cook et al., 2014). Model-based approaches use a topic model, e.g., to extract region-specific topics, and from this, predict the probability of seeing a word given a geographical region (Eisenstein et al., 2010). However, there are scalability issues, limiting the utility of such models.

In this paper, we propose a neural network architecture that learns a mixture of Gaussian distributions as its activation function, and predicts both locations from word-based inputs (geolocation), and words from location-based inputs (lexical dialectology).

3 Model

3.1 Bivariate Gaussian Distribution

A bivariate Gaussian distribution is a probability distribution over 2d space (in our case, a latitude/longitude coordinate pair). The probability mass function is given by:

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\Sigma) = \frac{1}{(2\pi)}\frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\intercal}\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}$$

where $\boldsymbol{\mu}$ is the 2-dimensional mean vector, the matrix $\Sigma = \begin{pmatrix} {\sigma_1}^2 & \rho_{12}\sigma_1\sigma_2 \\ \rho_{12}\sigma_1\sigma_2 & {\sigma_2}^2 \end{pmatrix}$ is the covariance matrix, and $|\Sigma|$ is its determinant. σ_1 and σ_2 are the standard deviations of the two dimensions, and ρ_{12} is the covariance. $\boldsymbol{x}$ is a latitude/longitude coordinate whose probability we are seeking to predict.

3.2 Mixtures of Gaussians

A mixture of Gaussians is a probabilistic model to represent subpopulations within a global population in the form of a weighted sum of K Gaussian distributions, where a higher weight with a component Gaussian indicates stronger association with that component. The probability mass function of a Gaussian mixture model is given by:

$$\mathcal{P}(\boldsymbol{x}) = \sum_{k=1}^{K}\pi_k\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_k},\Sigma_k)$$

where $\sum_{k=1}^{K}\pi_k = 1$, and the number of components K is a hyper-parameter.

3.3 Mixture Density Network (MDN)

A mixture density network ("MDN": Bishop (1994)) is a latent variable model where the conditional probability of $p(y|x)$ is modelled as a mixture of K Gaussians where the mixing coefficients π and the parameters of Gaussian distributions $\boldsymbol{\mu}$ and Σ are computed as a function of input using a neural network:

$$\mathcal{P}(y|x) = \sum_{k=1}^{K}\pi_k(x)\mathcal{N}\big(y|\mu_k(x),\Sigma_k(x)\big)$$

In the bivariate case of latitude/longitude, the number of parameters of each Gaussian is 6 $(\pi_k(x), \mu_{1k}(x), \mu_{2k}(x), \rho_k(x), \sigma_{1k}(x), \sigma_{2k}(x))$, which are learnt in the output layer of a regular neural network as a function of input x. The output size of the network for K components would be $6 \times K$. The output of an MDN for N samples (e.g. where N is the mini-batch size) is an $N \times 6K$ matrix which is then sliced and reshaped into $(N \times 2 \times K)$, $(N \times 2 \times K)$, $(N \times 1 \times K)$

and $(N \times 1 \times K)$ matrices, providing the model parameters μ, σ, ρ and π. Each parameter type has its own constraints: σ (the standard deviation) should be positive, ρ (the correlation) should be in the interval $[-1, 1]$ and π should be positive and sum to one, as a probability distribution. To force these constraints, the following transformations are often applied to each parameter set:

$$\sigma \sim \mathrm{SoftPlus}(\sigma\prime) = \log(\exp(\sigma\prime) + 1) \in (0, +\infty)$$
$$\pi \sim \mathrm{SoftMax}(\pi\prime)$$
$$\rho \sim \mathrm{SoftSign}(\rho\prime) = \frac{\rho\prime}{1 + |\rho\prime|} \in [-1, 1]$$

As an alternative, it's possible to use transformations like $\exp$ for σ and $\tanh$ for ρ. After applying the transformations to enforce the range constraints, the negative log likelihood loss of each sample x given a 2d coordinate label y is computed as:

$$\mathcal{L}(y|x) = -\log\left\{\sum_{k=1}^{K} \pi_k(x)\mathcal{N}\big(y|\mu_k(x), \Sigma_k(x)\big)\right\}$$

To predict a location, given an unseen input, the output of the network is reshaped into a mixture of Gaussians and μ_k, one of the K components' μ is chosen as the prediction. The selection criteria is either based on the strongest component with highest π, or the component that maximises the overall mixture probability:

$$\max_{\mu_i \in \{\mu_1 \ldots \mu_K\}} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_i|\mu_k, \Sigma_k)$$

For further details on selection criteria, see Bishop (1994).

3.4 Mixture Density Network with Shared Parameters (MDN-SHARED)

In the original MDN model proposed by Bishop (1994), the parameters of the mixture model are separate functions of input, which is appropriate when the inputs and outputs directly relate to each other, but in the case of geolocation or lexical dialectology, the relationship between inputs and outputs is not so obvious. As a result, it might be a difficult task for the model to learn all the parameters of each sample correctly. Instead of using the output to predict all the parameters, we share μ and Σ among all samples as parameters of the output layer, and only use the input to predict π, the mixture probabilities, using a SoftMax layer. We initialise μ by applying K-means clustering to the training coordinates and setting each value of μ to the centroids of the K clusters; we initialise Σ randomly between 0 and 10. We use the original cost function to update the weight matrices, biases and the global shared parameters of the mixture model through backpropagation. Prediction is performed in the same way as for MDN.

3.5 Continuous Representation of Location

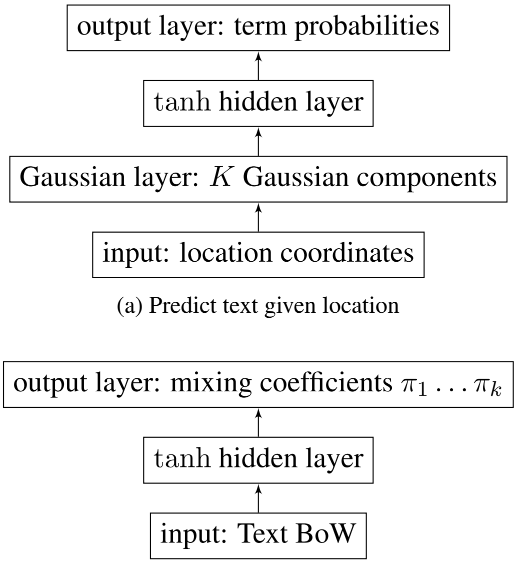
Gaussian mixtures are usually used as the output layer in neural networks (as in MDN) for inverse regression problems. We extend their application by using them as an input representation when the input is a multidimensional continuous variable. In problems such as lexical dialectology, the input is real-valued 2d coordinates, and the goal is to predict dialect words from a given location. Small differences in latitude/longitude may result in big shifts in language use (e.g. in regions such as Switzerland or Gibraltar). One way to model this is to discretise the input space (similar to the discretisation of the output space in classification), with the significant downside that the model is not able to learn/fine-tune regions in a data-driven way. A better solution is to use a K component Gaussian mixture representation of location, where μ and Σ are shared among all samples, and the output of the layer is the probability of input in each of the mixture components. Note that in this representation, there is no need for π parameters as we just need to represent the association of an input location to K regions, which will then be used as input to the next layer of a neural network and used to predict the targets. We use this continuous representation of location to predict dialect words from location input.

4 Experiments

We apply the two described MDN models on two widely-used geotagged Twitter datasets for geolocation, and compare the results with state-of-the-art classification and regression baselines. Also, we use the mixture of Gaussian representation of location to predict dialect terms from coordinates.

4.1 Data

In our experiments, we use two existing Twitter user geolocation datasets: (1) GEOTEXT (Eisenstein et al., 2010), and (2) TWITTER-US (Roller et al., 2012). Each dataset has fixed training, devel-


output layer: term probabilities
tanh hidden layer
Gaussian layer: K Gaussian components
input: location coordinates
output layer: mixing coefficients $\pi_1 \dots \pi_k$
tanh hidden layer
input: Text BoW

(a) Predict text given location

(b) Predict location given text

Figure 1: (a) The lexical dialectology model using a Gaussian representation layer. (b) MDN-SHARED geolocation model where the mixture weights π are predicted for each sample, and μ and Σ are parameters of the output layer, shared between all samples.

opment and test partitions, and a user is represented by the concatenation of their tweets, and labelled with the latitude/longitude of the first collected geotagged tweet.[2] GEOTEXT and TWITTER-US cover the continental US with 9k, 449k users, respectively.[3]

DARE is a dialect-term dataset derived from the Dictionary of American Regional English (Cassidy, 1985) by Rahimi et al. (to appear). DARE consists of dialect regions, terms and the meaning of each term.[4] It represents the aggregation of a number of dialectal surveys over different regions of the U.S., to identify shared dialect regions. Because the dialect regions in DARE maps are not machine readable, populous cities within each dialect region are manually extracted and associated with their dialect region terms. The dataset is made up of around 4.3k dialect terms across 99 U.S. dialect regions.

4.2 Geolocation

We use a 3-layer neural network as shown in Figure 1a where the input is the l_2 normalised bag-of-words model of a given user with stop words, @-mentions and words with document frequency less than 10 removed. The input is fed to a hidden layer with tanh nonlinearity that produces the output of the network (with no nonlinearity applied). The output is the collection of Gaussian mixture parameters (μ, Σ, π) from MDN. For prediction, the μ_k of the mixture component which has the highest probability within the mixture component is selected. In the case of MDN-SHARED, the output is only π, a vector with size K, but the output layer contains extra global parameters μ and Σ $(\sigma_{\text{lat}}, \sigma_{\text{lon}}, \rho)$ which are shared between all the samples. The negative log likelihood objective is optimised using Adam (Kingma and Ba, 2014) and early stopping is used to prevent overfitting. The hidden layer is subject to drop-out and elastic net regularisation (with equal l_1 and l_2 shares). As our baseline, we used a multilayer perceptron regressor with the same input and hidden architecture but with a 2d output with linear activation that predicts the location from text input. The regularisation and drop-out rate, hidden layer size and the number of Gaussian components K (for MDN and MDN-SHARED) are tuned over the development set of each dataset, as shown in Table 1.

We evaluate the predictions of the geolocation models based on three measures (following Cheng et al. (2010) and Eisenstein et al. (2010)):

1. the classification accuracy within a 161km (= 100 mile) radius of the actual location ("Acc@161"); i.e., if the predicted location is within 161km of the actual location, it is considered to be correct
2. the mean error ("Mean") between the predicted location and the actual location of the user, in kilometres
3. the median error ("Median") between the predicted location and the actual location of the user, in kilometres

4.3 Lexical Dialectology

To predict dialect words from location, we use a 4-layer neural network as shown in Figure 1b. The input is a latitude/longitude coordinate, the first hidden layer is a Gaussian mixture with K components which has μ and Σ as its parameters and produces a probability for each component as an

[2] This geolocation representation is naive, but was made by the creators of the original datasets and has been used by others. It has been preserved in this work for comparability with the results of others, despite misgivings about whether this is a faithful representation of the location for a given user.

[3] The datasets can be obtained from https://github.com/utcompling/textgrounder.

[4] http://www.daredictionary.com/

activation function, the second hidden layer with $\tanh$ nonlinearity captures the association between different Gaussians, and the output is a $\mathrm{SoftMax}$ layer which results in a probability distribution over the vocabulary. For a user label, we use an l_1 normalised bag-of-words representation of its text content and use binary tf and idf for term-weighting. The model should learn to predict the probability distribution over the vocabulary and so be capable of predicting dialect words with a higher probability. It also learns regions (parameters of K Gaussians) that represent dialect regions.

We evaluate the lexical dialectology model (`MDN-layer`) using perplexity of the predicted unigram distribution, and compare it with a baseline where the Gaussian mixture layer is replaced with a $\tanh$ hidden layer (`tanh-layer`). Also we retrieve words given points within a region from the DAREDS dataset, and measure recall with respect to relevant dialect terms from DAREDS. To do that, we randomly sample $P = 10000$ latitude/longitude points from the training set and predict the corresponding word distribution. To come up with a ranking over words given region r as query, we use the following measure:

$$
\begin{aligned}
score(w_i|r) = & \frac{1}{N}\sum_{p_j \in r} \log(P(w_i|p_j)) \\
& -\frac{1}{P}\sum_{j=1}^{P} \log(P(w_i|p_j))
\end{aligned}
$$

where N equals the number of points (out of 10000) inside the query dialect region r and P equals the total number of points (here 10000). For example, if we are querying dialect terms from dialect region South (r), N is the number of randomly selected points that fall within the constituent states of South. $score(w_i|r)$ measures the (log) probability ratio of a word w_i inside region r compared to its global score: if a word is local to region r, the ratio will be higher. We use this measure to create a ranking over the vocabulary from which we measure precision and recall at k given gold-standard dialect terms in DAREDS.

5 Results

5.1 Geolocation

The performance of `Regression`, `MDN` and `MDN-SHARED`, along with several state-of-the-art classification models, is shown in Table 2. The `MDN` and `MDN-SHARED` models clearly outperform `Regression`, and achieve competitive or slightly worse results than the classification models but provide uncertainty over the whole output space. The geographical distribution of error for `MDN-SHARED` over the development set of TWITTER-US is shown in Figure 3, indicating larger errors in MidWest and particularly in North Pacific regions (e.g. Oregon).

5.2 Dialectology

The perplexity of the lexical dialectology model using Gaussian mixture representation (`MDN-layer`) is 840 for the 54k vocabulary of TWITTER-US dataset, 1% lower than a similar network architecture with a $\tanh$ hidden layer (`tanh-layer`), which is not a significant improvement. Also we evaluated the model using recall at k and compared it to the `tanh-layer` model which again is competitive with `tanh-layer` but with the advantage of learning dialect regions simultaneously. Because the DARE dialect terms are not used frequently in Twitter, many of the words are not covered in our dataset, despite its size. However, our model is able to retrieve dialect terms that are distinctly associated with regions. The top dialect words for regions New York, Louisiana, Illinois and Pennsylvania are shown in Table 3, and include named entities, dialect words and local hashtags. We also visualised the learned Gaussians of the dialectology model in Figure 2, which as expected show several smaller regions (Gaussians with higher σ) and larger regions in lower populated areas. It is interesting to see that the shape of the learned Gaussians matches natural borders such as coastal regions.

We also visualised the log probability of dialect terms *hella* (an intensifier mainly used in Northern California) and *yall* (means "you all", used in Southern U.S.) resulting from the Gaussian representation model. As shown in Figure 5, the heatmap matches the expected regions.

6 Conclusion

We proposed a continuous representation of location using mixture of Gaussians and applied it to geotagged Twitter data in two different settings: (1) geolocation of social media users, and (2) lexical dialectology. We used `MDN` (Bishop, 1994) in a multilayer neural network as a geolocation model

	GeoText				Twitter-US			
	regul.	dropout	hidden	K	regul.	dropout	hidden	K
Baseline (Regression)	0	0	(100, 50)	—	10^{-5}	0	(100, 50)	—
Proposed method (MDN)	0	0.5	100	100	10^{-5}	0	300	100
Proposed method (MDN-SHARED)	0	0	100	300	0	0	900	900

Table 1: Hyper-parameter settings of the geolocation models tuned over development set of each dataset. K is the number of Gaussian components in MDN and MDN-SHARED. Regression has a tanh hidden layer instead of the Gaussian layer. "—" means the parameter is not applicable to the model.

	GeoText			Twitter-US		
	Acc@161	Mean	Median	Acc@161	Mean	Median
Baseline (Regression)	4	951	733	9	746	557
Proposed method (MDN)	24	983	505	29	696	281
Proposed method (MDN-SHARED)	39	865	412	42	655	216
Classification Methods						
(Rahimi et al., 2015) (LR)	38	880	397	50	686	159
(Wing and Baldridge, 2014) (uniform)	—	—	—	49	703	170
(Wing and Baldridge, 2014) (k-d tree)	—	—	—	48	686	191
(Melo and Martins, 2015)	—	—	—	—	702	208
(Cha et al., 2015)	—	581	425	—	—	—
(Liu and Inkpen, 2015)	—	—	—	—	733	377

Table 2: Geolocation results over GeoText and Twitter-US datasets based on Regression, MDN and MDN-SHARED methods. The results are also compared to state-of-the-art classification methods. "—" signifies that no results were reported for the given metric or dataset.

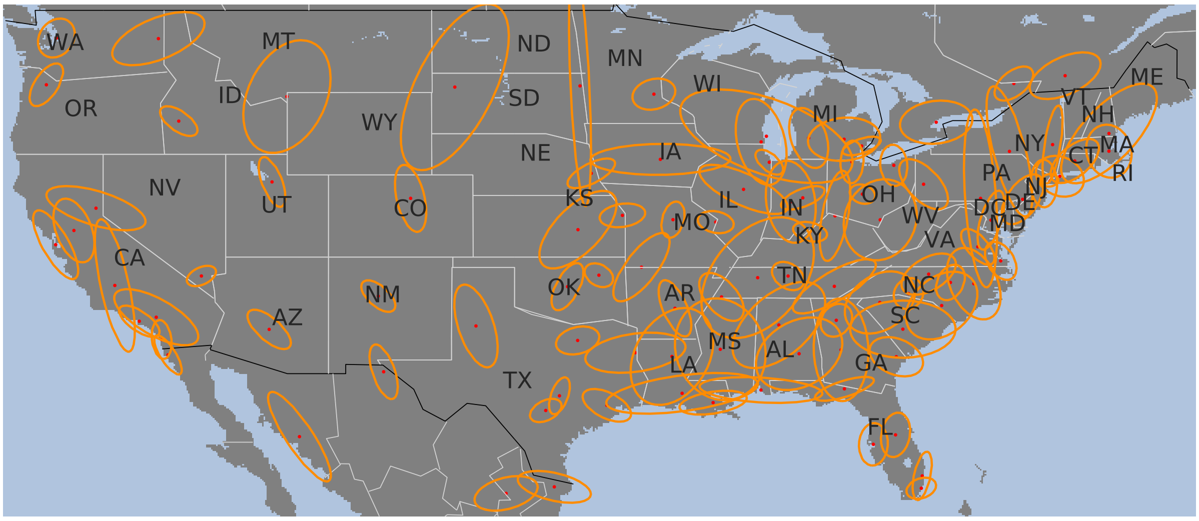


Figure 2: Learned Gaussian representation of input locations over Twitter-US in the lexical dialectology model. The number of Gaussian components, K, is set to 100 . The red points are μ_k and the contours are drawn at $p = 0.01$.

and showed that it outperforms regression models by a large margin. There is also very recent work (Iso et al., 2017) in tweet-level geolocation that shows the effectiveness of MDN.

We modified MDN by sharing the parameters of the Gaussian mixtures in MDN-SHARED and improved upon MDN, achieving competetive results with state-of-the-art classification models. We also applied the Gaussian mixture representation to predict dialect words from location, and showed that it

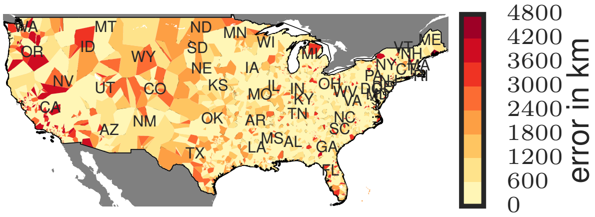

4800
4200
3600
3000
2400
1800
1200
600
0
error in km
WA OR ID MT ND SD MN WI NV UT WY CO NE IA KS MO IL IN OH KY WV VA MI NY PA VT NH ME MA CT RI NJ DE MD CA AZ NM OK AR TN NC SC TX LA MS AL GA FL

Figure 3: The distribution of geolocation error for MDN-SHARED over the development set of TWITTER-US.

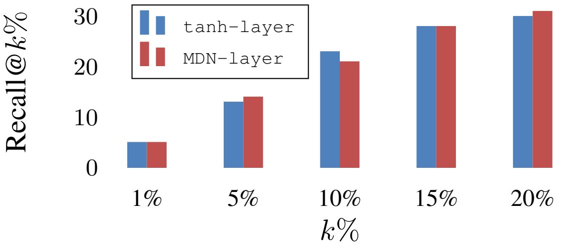
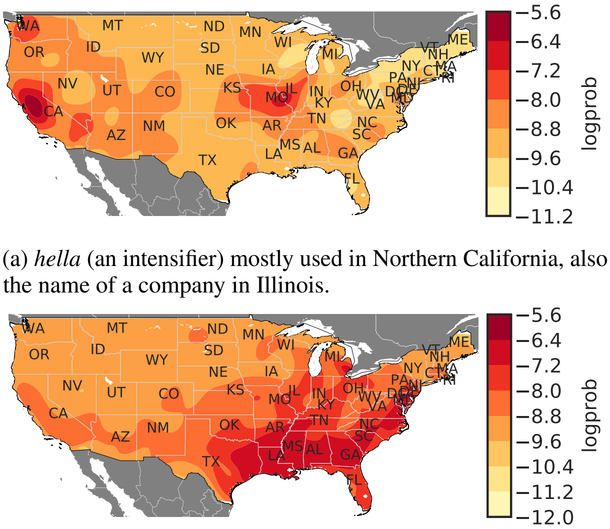


Figure 4: Recall at $k\%$ of 54k vocabulary for retrieving DAREDS dialect words given points within a dialect region.


−5.6
−6.4
−7.2
−8.0
−8.8
−9.6
−10.4
−11.2
logprob
−5.6
−6.4
−7.2
−8.0
−8.8
−9.6
−10.4
−11.2
−12.0
logprob
WA OR ID MT ND SD MN WI NV UT WY CO NE IA KS MO IL IN OH KY WV VA MI NY PA VT NH ME MA CT RI NJ DE MD CA AZ NM OK AR TN NC SC TX LA MS AL GA FL

(a) *hella* (an intensifier) mostly used in Northern California, also the name of a company in Illinois.

(b) *yall* (means you all) mainly used in Southern U.S.

Figure 5: Log probabilities of terms: (a) *hella* and (b) *yall* in continental U.S.

New York	Louisiana	Illinois	Pennsylvania
flatbush	kmsl	metra	cdfu
lirr	jtfo	osco	jawn
reade	wassam	halsted	ard
rivington	kmfsl	kedzie	erked
mta	ndc	lbvs	cthu
nostrand	bookoo	damen	septa
stuyvesant	#icantdeal	niu	prussia
pathmark	#drove	orland	drawlin
bleecker	slangs	cermak	youngbull
bowery	daq	uic	prolli
macdougal	#gramfam	oms	#ttm
broome	gtf	xsport	dickeating
driggs	metairie	cta	ctfu

Table 3: Top terms retrieved from the lexical dialectology model given lat/lon training points within a state ranked by Equation 4.3.

is competitive with simple tanh activation in terms of both perplexity of the predicted unigram model and also recall at k at retrieving DARE dialect words by location input. Furthermore we showed that the learned Gaussian mixtures have intereting properties such as covering high population density regions (e.g. NYC and LA) with a larger number of small Gaussians, and a smaller number of larger Gaussians in low density areas (e.g. the midwest).

Although we applied the mixture of Gaussians to location data, it can be used in other settings where the input or output are from a continuous multivariate distribution. For example it can be applied to predict financial risk (Wang and Hua, 2014) and sentiment (Joshi et al., 2010) given text. We showed that when a global structure exists (e.g. population centres, in the case of geolocation) it is better to share the global parameters of the mixture model to improve generalisation. In this work, we used the bivariate Gaussian distribution in the MDN's mixture leaving the use of other distributions which might better suit the geolocation task for future research.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This work was supported in part by the Australian Research Council.

References

Amr Ahmed, Liangjie Hong, and Alexander J Smola. 2013. Hierarchical geographical modeling of user locations from social media posts. In *Proceedings of the 22nd International Conference on World Wide Web (WWW 2013)*, pages 25–36, Rio de Janeiro, Brazil.

Aris Anagnostopoulos, Fabio Petroni, and Mara Sorella. 2016. Targeted interest-driven advertising in cities using twitter. In *Proceedings of the 10th International Conference on Weblogs and Social Me-*

dia (ICWSM 2016), pages 527–530, Cologne, Germany.

Zahra Ashktorab, Christopher Brown, Manojit Nandi, and Aron Culotta. 2014. Tweedr: Mining Twitter to inform disaster response. In *Proceedings of The 11th International Conference on Information Systems for Crisis Response and Management (ISCRAM 2014)*, pages 354–358, University Park, USA.

Christopher Bishop. 1994. Mixture density networks. Technical report, Aston University.

Zsolt Bitvai and Trevor Cohn. 2015. Predicting peer-to-peer loan rates using bayesian non-linear regression. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, pages 2203–2209, Austin, USA.

Frederic Gomes Cassidy. 1985. *Dictionary of American Regional English*. Belknap Press of Harvard University Press.

Miriam Cha, Youngjune Gwon, and H.T. Kung. 2015. Twitter geolocation and regional classification via sparse coding. In *Proceedings of the 9th International Conference on Weblogs and Social Media (ICWSM 2015)*, pages 582–585, Oxford, UK.

Zhiyuan Cheng, James Caverlee, and Kyumin Lee. 2010. You are where you tweet: a content-based approach to geo-locating Twitter users. In *Proceedings of the 19th ACM International Conference Information and Knowledge Management (CIKM 2010)*, pages 759–768, Toronto, Canada.

Paul Cook, Bo Han, and Timothy Baldwin. 2014. Statistical methods for identifying local dialectal terms from gps-tagged documents. *Dictionaries: Journal of the Dictionary Society of North America*, 35(35):248–271.

Gabriel Doyle. 2014. Mapping dialectal variation by querying social media. In *Proceedings of the 14th Conference of the EACL (EACL 2014)*, pages 98–106, Gothenburg, Sweden.

Mark Dredze, Manuel García-Herranz, Alex Rutherford, and Gideon Mann. 2016. Twitter as a source of global mobility patterns for social good. In *ICML Workshop on #Data4Good: Machine Learning in Social Good Applications*, New York, USA.

Jacob Eisenstein. 2015. Written dialect variation in online social media. In Charles Boberg, John Nerbonne, and Dom Watt, editors, *Handbook of Dialectology*. Wiley.

Jacob Eisenstein, Brendan O'Connor, Noah A Smith, and Eric P Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 1277–1287, Boston, USA.

Noora Al Emadi, Sofiane Abbar, Javier Borge-Holthoefer, Francisco Guzman, and Fabrizio Sebastiani. 2017. QT2S: A system for monitoring road traffic via fine grounding of tweets. In *Proceedings of the 11th International Conference on Weblogs and Social Media (ICWSM 2017)*, pages 456–459, Montreal, Canada.

Bruno Gonçalves and David Sánchez. 2014. Crowdsourcing dialect characterization through Twitter. *PloS One*, 9(11).

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Geolocation prediction in social media data by finding location indicative words. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 1045–1062, Mumbai, India.

Bo Han, Paul Cook, and Timothy Baldwin. 2014. Text-based Twitter user geolocation prediction. *Journal of Artificial Intelligence Research*, 49:451–500.

Brent Hecht, Lichan Hong, Bongwon Suh, and Ed H. Chi. 2011. Tweets from justin bieber's heart: the dynamics of the location field in user profiles. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 237–246, Vancouver, Canada.

Shen-Shyang Ho, Mike Lieberman, Pu Wang, and Hanan Samet. 2012. Mining future spatiotemporal events and their sentiment from online news articles for location-aware recommendation system. In *Proceedings of the First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*, pages 25–32, Redondo Beach, USA.

Yuan Huang, Diansheng Guo, Alice Kasakoff, and Jack Grieve. 2015. Understanding US regional linguistic variation with Twitter data analysis. *Computers, Environment and Urban Systems*, 59:244–255.

Hayate Iso, Shoko Wakamiya, and Eiji Aramaki. 2017. Density estimation for geolocation via convolutional mixture density network. *arXiv preprint arXiv:1705.02750*.

Mahesh Joshi, Dipanjan Das, Kevin Gimpel, and Noah A Smith. 2010. Movie reviews and revenues: An experiment in text regression. In *Proceedings of The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL2010)*, pages 293–296, Los Angeles, USA. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Sheila Kinsella, Vanessa Murdock, and Neil O'Hare. 2011. "I'm eating a sandwich in Glasgow": Modeling locations with tweets. In *Proceedings of the 3rd International Workshop on Search and Mining User-generated Contents*, pages 61–68, Glasgow, UK.

William Labov, Sharon Ash, and Charles Boberg. 2005. *The Atlas of North American English: Phonetics, Phonology and Sound Change*. Walter de Gruyter.

Ji Liu and Diana Inkpen. 2015. Estimating user location in social media with stacked denoising autoencoders. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics — Human Language Technologies (NAACL HLT 2015)*, pages 201–210, Denver, USA.

Graham McNeill, Jonathan Bright, and Scott A Hale. 2016. Estimating local commuting patterns from geolocated Twitter data. *arXiv preprint arXiv:1612.01785*.

Fernando Melo and Bruno Martins. 2015. Geocoding textual documents through the usage of hierarchical classifiers. In *Proceedings of the 9th Workshop on Geographic Information Retrieval (GIR 2015)*, Paris, France.

Burt L. Monroe, Michael P. Colaresi, and Kevin M. Quinn. 2008. Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372–403.

John Nerbonne, Peter Kleiweg, Wilbert Heeringa, and Franz Manni. 2008. Projecting dialect distances to geography: Bootstrap clustering vs. noisy clustering. In *Proceedings of the 31st Annual Meeting of the German Classification Society*, pages 647–654.

Dong Nguyen and Jacob Eisenstein. to appear. A kernel independence test for geographical language variation. *Computational Linguistics*.

Reid Priedhorsky, Aron Culotta, and Sara Y. Del Valle. 2014. Inferring the origin locations of tweets with quantitative confidence. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 1523–1536, Baltimore, USA.

Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. to appear. A neural model for user geolocation and lexical dialectology. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, Vancouver, Canada.

Afshin Rahimi, Duy Vu, Trevor Cohn, and Timothy Baldwin. 2015. Exploiting text and network context for geolocation of social media users. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics — Human Language Technologies (NAACL HLT 2015)*, Denver, USA.

Stephen Roller, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge. 2012. Supervised text-based geolocation using language models on an adaptive grid. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CONLL 2012)*, pages 1500–1510, Jeju, South Korea.

Pavel Serdyukov, Vanessa Murdock, and Roelof Van Zwol. 2009. Placing Flickr photos on a map. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 484–491, Boston, USA.

Shoko Wakamiya, Yukiko Kawai, and Eiji Aramaki. 2016. After the boom no one tweets: Microblog-based influenza detection incorporating indirect information. In *Proceedings of the Sixth International Conference on Emerging Databases: Technologies, Applications, and Theory*, pages 17–25, Jeju, South Korea.

William Yang Wang and Zhenhao Hua. 2014. A semiparametric Gaussian copula regression model for predicting financial risks from earnings calls. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1155–1165, Baltimore, USA.

Benjamin P Wing and Jason Baldridge. 2011. Simple supervised document geolocation with geodesic grids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1 (ACL-HLT 2011)*, pages 955–964, Portland, USA.

Benjamin P Wing and Jason Baldridge. 2014. Hierarchical discriminative classification for text-based geolocation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 336–348, Doha, Qatar.

OBJ2TEXT: Generating Visually Descriptive Language from Object Layouts

Xuwang Yin **Vicente Ordonez**
Department of Computer Science, University of Virginia, Charlottesville, VA.
[xy4cm, vicente]@virginia.edu

Abstract

Generating captions for images is a task that has recently received considerable attention. In this work we focus on caption generation for abstract scenes, or object layouts where the only information provided is a set of objects and their locations. We propose OBJ2TEXT, a sequence-to-sequence model that encodes a set of objects and their locations as an input sequence using an LSTM network, and decodes this representation using an LSTM language model. We show that our model, despite encoding object layouts as a sequence, can represent spatial relationships between objects, and generate descriptions that are globally coherent and semantically relevant. We test our approach in a task of object-layout captioning by using only object annotations as inputs. We additionally show that our model, combined with a state-of-the-art object detector, improves an image captioning model from 0.863 to 0.950 (CIDEr score) in the test benchmark of the standard MS-COCO Captioning task.

1 Introduction

Natural Language generation (NLG) is a long standing goal in natural language processing. There have already been several successes in applications such as financial reporting (Kukich, 1983; Smadja and McKeown, 1990), or weather forecasts (Konstas and Lapata, 2012; Wen et al., 2015), however it is still a challenging task for less structured and open domains. Given recent progress in training robust visual recognition models using convolutional neural networks, the task of generating natural language descriptions for arbitrary images has received considerable attention (Vinyals et al., 2015; Karpathy and Fei-Fei, 2015; Mao et al., 2015). In general, generating visually descriptive language can be useful for various tasks such as human-machine communication, accessibility, image retrieval, and search. However this task is still challenging and it depends on developing both a robust visual recognition model, and a reliable language generation model. In this paper, we instead tackle a task of describing object layouts where the categories for the objects in an input scene and their corresponding locations are known. Object layouts are commonly used for story-boarding, sketching, and computer graphics applications. Additionally, using our object layout captioning model on the outputs of an object detector we are also able to improve image caption-

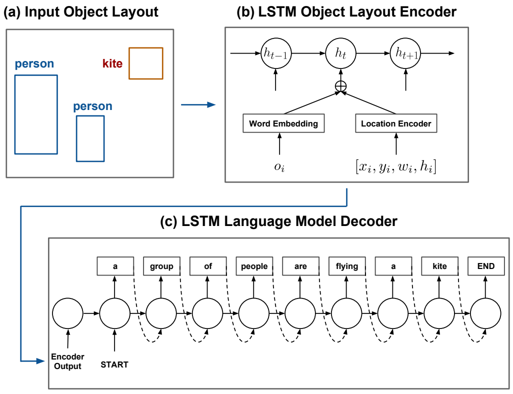


Figure 1: Overview of our proposed model for generating visually descriptive language from object layouts. The input (a) is an object layout that consists of object categories and their corresponding bounding boxes, the encoder (b) uses a two-stream recurrent neural network to encode the input object layout, and the decoder (c) uses a standard LSTM recurrent neural network to generate text.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 177–187
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

ing models. Object layouts contain rich semantic information, however they also abstract away several other visual cues such as color, texture, and appearance, thus introducing a different set of challenges than those found in traditional image captioning.

We propose OBJ2TEXT, a sequence-to-sequence model that encodes object layouts using an LSTM network (Hochreiter and Schmidhuber, 1997), and decodes natural language descriptions using an LSTM-based neural language model[1]. Natural language generation systems usually consist of two steps: content planning, and surface realization. The first step decides on the content to be included in the generated text, and the second step connects the concepts using structural language properties. In our proposed model, OBJ2TEXT, content planning is performed by the encoder, and surface realization is performed by the decoder. Our model is trained in the standard MS-COCO dataset (Lin et al., 2014), which includes both object annotations for the task of object detection, and textual descriptions for the task of image captioning. While most previous research has been devoted to any one of these two tasks, our paper presents, to our knowledge, the first approach for learning mappings between object annotations and textual descriptions. Using several lesioned versions of the proposed model we explored the effect of object counts and locations in the quality and accuracy of the generated natural language descriptions.

Generating visually descriptive language requires beyond syntax, and semantics; an understanding of the physical word. We also take inspiration from recent work by Schmaltz et al. (2016) where the goal was to reconstruct a sentence from a bag-of-words (BOW) representation using a simple surface-level language model based on an encoder-decoder sequence-to-sequence architecture. In contrast to this previous approach, our model is grounded on visual data, and its corresponding spatial information, so it goes beyond word re-ordering. Also relevant to our work is Yao et al. (2016a) which previously explored the task of oracle image captioning by providing a language generation model with a list of manually defined visual concepts known to be present in the image. In addition, our model is able to leverage both quantity and spatial information as additional cues associated with each object/concept, thus allowing it to learn about verbosity, and spatial relations in a supervised fashion.

In summary, our contributions are as follows:

- We demonstrate that despite encoding object layouts as a sequence using an LSTM, our model can still effectively capture spatial information for the captioning task. We perform ablation studies to measure the individual impact of object counts, and locations.
- We show that a model relying only on object annotations as opposed to pixel data, performs competitively in image captioning despite the ambiguity of the setup for this task.
- We show that more accurate and comprehensive descriptions can be generated on the image captioning task by combining our OBJ2TEXT model using the outputs of a state-of-the-art object detector with a standard image captioning approach.

2 Task

We evaluate OBJ2TEXT in the task of object layout captioning, and image captioning. In the first task, the input is an object layout that takes the form of a set of object categories and bounding box pairs $\langle \boldsymbol{o}, \boldsymbol{l} \rangle = \{\langle o_i, l_i \rangle\}$, and the output is natural language. This task resembles the second task of image captioning except that the input is an object layout instead of a standard raster image represented as a pixel array. We experiment in the MS-COCO dataset for both tasks. For the first task, object layouts are derived from ground-truth bounding box annotations, and in the second task object layouts are obtained using the outputs of an object detector over the input image.

3 Related Work

Our work is related to previous works that used clipart scenes for visually-grounded tasks including sentence interpretation (Zitnick and Parikh, 2013; Zitnick et al., 2013), and predicting object dynamics (Fouhey and Zitnick, 2014). The cited advantage of abstract scene representations such as the ones provided by the clipart scenes dataset proposed in (Zitnick and Parikh, 2013) is their ability to separate the complexity of pattern recognition from semantic visual representation. Abstract scene representations also maintain

[1] We build on neuraltalk2 and make our Torch code, and an interactive demo of our model available in the following url: http://vision.cs.virginia.edu/obj2text

common-sense knowledge about the world. The works of Vedantam et al. (2015b); Eysenbach et al. (2016) proposed methods to learn common-sense knowledge from clipart scenes, while the method of Yatskar et al. (2016), similar to our work, leverages object annotations for natural images. Understanding abstract scenes has demonstrated to be a useful capability for both language and vision tasks and our work is another step in this direction.

Our work is also related to other language generation tasks such as image and video captioning (Farhadi et al., 2010; Ordonez et al., 2011; Mason and Charniak, 2014; Ordonez et al., 2015; Xu et al., 2015; Donahue et al., 2015; Mao et al., 2015; Fang et al., 2015). This problem is interesting because it combines two challenging but perhaps complementary tasks: visual recognition, and generating coherent language. Fueled by recent advances in training deep neural networks (Krizhevsky et al., 2012) and the availability of large annotated datasets with images and captions such as the MS-COCO dataset (Lin et al., 2014), recent methods on this task perform end-to-end learning from pixels to text. Most recent approaches use a variation of an encoder-decoder model where a convolutional neural network (CNN) extracts visual features from the input image (encoder), and passes its outputs to a recurrent neural network (RNN) that generates a caption as a sequence of words (decoder) (Karpathy and Fei-Fei, 2015; Vinyals et al., 2015). However, the MS-COCO dataset, containing object annotations, is also a popular benchmark in computer vision for the task of object detection, where the objective is to go from pixels to a collection of object locations. In this paper, we instead frame our problem as going from a collection of object categories and locations (object layouts) to image captions. This requires proposing a novel encoding approach to encode these object layouts instead of pixels, and allows for analyzing the image captioning task from a different perspective. Several other recent works use a similar sequence-to-sequence approach to generate text from source code input (Iyer et al., 2016), or to translate text from one language to another (Bahdanau et al., 2015).

There have also been a few previous works explicitly analyzing the role of spatial and geometric relations between objects for vision and language related tasks. The work of Elliott and Keller (2013) manually defined a dictionary of object-object relations based on geometric cues. The work of Ramisa et al. (2015) is focused on predicting preposition given two entities and their locations in an image. Previous works of Plummer et al. (2015) and Rohrbach et al. (2016) showed that switching from classification-based CNN network to detection-based Fast RCNN network improves performance for phrase localization. The work of Hu et al. (2016) showed that encoding image regions with spatial information is crucial for natural language object retrieval as the task explicitly asks for locations of target objects. Unlike these previous efforts, our model is trained end-to-end for the language generation task, and takes as input a holistic view of the scene layout, potentially learning higher order relations between objects.

4 Model

In this section we describe our base OBJ2TEXT model for encoding object layouts to produce text (section 4.1), as well as two further variations to use our model to generate captions for real images: OBJ2TEXT-YOLO which uses the YOLO object detector (Redmon and Farhadi, 2017) to generate layouts of object locations from real images (section 4.2), and OBJ2TEXT-YOLO + CNN-RNN which further combines the previous model with an encoder-decoder image captioning which uses a convolutional neural network to encode the image (section 4.3).

4.1 OBJ2TEXT

OBJ2TEXT is a sequence-to-sequence model that encodes an input object layout as a sequence, and decodes a textual description by predicting the next word at each time step. Given a training data set comprising N observations $\left\{\langle \boldsymbol{o}^{(n)}, \boldsymbol{l}^{(n)} \rangle\right\}$, where $\langle \boldsymbol{o}^{(n)}, \boldsymbol{l}^{(n)} \rangle$ is a pair of sequences of input category and location vectors, together with a corresponding set of target captions $\left\{\boldsymbol{s}^{(n)}\right\}$, the encoder and decoder are trained jointly by minimizing a loss function over the training set using stochastic gradient descent:

$$W^* = \arg\min_W \sum_{n=1}^{N} \mathcal{L}(\langle \boldsymbol{o}^{(n)}, \boldsymbol{l}^{(n)} \rangle, \boldsymbol{s}^{(n)}), \qquad (1)$$

in which $W = \binom{W_1}{W_2}$ is the group of encoder parameters W_1 and decoder parameters W_2. The loss function is a negative log likelihood function of the generated description given the encoded object layout

$$\mathcal{L}(\langle \boldsymbol{o}^{(n)}, \boldsymbol{l}^{(n)} \rangle, \boldsymbol{s}^{(n)}) = -\log p(\boldsymbol{s}^n | h_L^n, W_2), \quad (2)$$

where h_L^n is computed using the LSTM-based encoder (eqs. 3, and 4) from the object layout inputs $\langle \boldsymbol{o}^{(n)}, \boldsymbol{l}^{(n)} \rangle$, and $p(\boldsymbol{s}^n | h_L^n, W_2)$ is computed using the LSTM-based decoder (eqs. 5, 6 and 7).

At inference time we encode an input layout $\langle \boldsymbol{o}, \boldsymbol{l} \rangle$ into its representation h_L, and sample a sentence word by word based on $p(s_t | h_L, \boldsymbol{s}_{<t})$ as computed by the decoder in time-step t. Finding the optimal sentence $\boldsymbol{s}^* = \arg\max_{\boldsymbol{s}} p(\boldsymbol{s}|h_L)$ requires the evaluation of an exponential number of sentences as in each time-step we have K number of choices for a word vocabulary of size K. As a common practice for an approximate solution, we follow (Vinyals et al., 2015) and use beam search to limit the choices for words at each time-step by only using the ones with the highest probabilities.

Encoder: The encoder at each time-step t takes as input a pair $\langle o_t, l_t \rangle$, where o_t is the object category encoded as a one-hot vector of size V, and $l_t = [B_t^x, B_t^y, B_t^w, B_t^h]$ is the location configuration vector that contains left-most position, top-most position, and the width and height of the bounding box corresponding to object o_t, all normalized in the range [0,1] with respect to input image dimensions. o_t and l_t are mapped to vectors with the same size k and added to form the input x_t to one time-step of the LSTM-based encoder as follows:

$$x_t = W_o o_t + (W_l l_t + b_l), \quad x_t \in \mathbb{R}^k, \quad (3)$$

in which $W_o \in \mathbb{R}^{k \times V}$ is a categorical embedding matrix (the word encoder), and $W_l \in \mathbb{R}^{k \times 4}$ and bias $b_l \in \mathbb{R}^k$ are parameters of a linear transformation unit (the object location encoder).

Setting initial value of cell state vector $c_0^e = 0$ and hidden state vector $h_0^e = 0$, the LSTM-based encoder takes the sequence of input $(x_1, ..., x_{T_1})$ and generates a sequence of hidden state vectors $(h_1^e, ..., h_{T_1}^e)$ using the following step function (we omit cell state variables and internal transition gates for simplicity as we use a standard LSTM cell definition):

$$h_t^e = \text{LSTM}(h_{t-1}^e, x_t; W_1). \quad (4)$$

We use the last hidden state vector $h_L = h_{T_1}^e$ as the encoded representation of the input layout $\langle \boldsymbol{o}_t, \boldsymbol{l}_t \rangle$ to generate the corresponding description $\boldsymbol{s}$.

Decoder: The decoder takes the encoded layout h_L as input and generates a sequence of multinomial distributions over a vocabulary of words using an LSTM neural language model. The joint probability distribution of generated sentence $\boldsymbol{s} = (s_1, ..., s_{T_2})$ is factorized into products of conditional probabilities:

$$p(\boldsymbol{s}|h_L) = \prod_{t=1}^{T_2} p(s_t | h_L, \boldsymbol{s}_{<t}), \quad (5)$$

where each factor is computed using a softmax function over the hidden states of the decoder LSTM as follows:

$$p(s_t | h_L, \boldsymbol{s}_{<t}) = \text{softmax}(W_h h_{t-1}^d + b_h), \quad (6)$$

$$h_t^d = \text{LSTM}(h_{t-1}^d, W_s s_t; W_2), \quad (7)$$

where W_s is the categorical embedding matrix for the one-hot encoded caption sequence of symbols. By setting $h_{-1}^d = 0$ and $c_{-1}^d = 0$ for the initial hidden state and cell state, the layout representation is encoded into the decoder network at the 0 time step as a regular input:

$$h_0^d = \text{LSTM}(h_{-1}^d, h_L; W_2). \quad (8)$$

We use beam search to sample from the LSTM as is routinely performed in previous literature in order to generate text.

4.2 OBJ2TEXT-YOLO

For the task of image captioning we propose OBJ2TEXT-YOLO. This model takes an image as input, extracts an object layout (object categories and locations) with a state-of-the-art object detection model YOLO (Redmon and Farhadi, 2017), and uses OBJ2TEXT as described in section 4.1 to generate a natural language description of the input layout and hence, the input image. The model is trained using the standard back-propagation algorithm, but the error is not back-propagated to the object detection module.

4.3 OBJ2TEXT-YOLO + CNN-RNN

For the image captioning task we experiment with a combined model (see Figure 2) where we take an image as input, and then use two separate computation branches to extract visual feature information and object layout information. These two streams of information are then passed to an LSTM neural language model to generate a description. Visual features are extracted using the

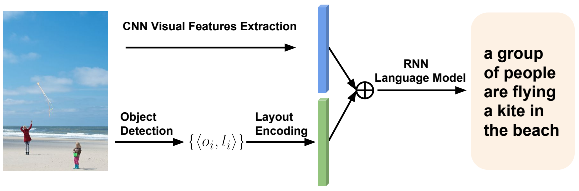

CNN Visual Features Extraction
RNN Language Model
a group of people are flying a kite in the beach
Object Detection
$\{\langle o_i, l_i \rangle\}$
Layout Encoding

Figure 2: Image Captioning by joint learning of visual features and object layout encoding.

VGG-16 (Simonyan and Zisserman, 2015) convolutional neural network pre-trained on the ImageNet classification task (Russakovsky et al., 2015). Object layouts are extracted using the YOLO object detection system and its output object locations are encoded using our proposed OBJ2TEXT encoder. These two streams of information are encoded into vectors of the same size and their sum is input to the language model to generate a textual description. The model is trained using the standard back-propagation algorithm where the error is back-propagated to both branches but not the object detection module. The weights of the image CNN model are fine-tuned only after the layout encoding branch is well trained but no significant overall performance improvements were observed.

5 Experimental Setup

We evaluate the proposed models on the MS-COCO (Lin et al., 2014) dataset which is a popular image captioning benchmark that also contains object extent annotations. In the object layout captioning task the model uses the ground-truth object extents as input object layouts, while in the image captioning task the model takes raw images as input. The qualities of generated descriptions are evaluated using both human evaluations and automatic metrics. We train and validate our models based on the commonly adopted split regime (113,287 training images, 5000 validation and 5000 test images) used in (Karpathy et al., 2016), and also test our model in the MS-COCO official test benchmark.

We implement our models based on the open source image captioning system Neuraltalk2 (Karpathy et al., 2016). Other configurations including data preprocessing and training hyper-parameters also follow Neuraltalk2. We trained our models using a GTX1080 GPU with 8GB of memory for 400k iterations using a batch size of 16 and an Adam optimizer with alpha of 0.8, beta of 0.999 and epsilon of 1e-08. Descriptions of the CNN-RNN approach are generated using the publicly available code and model checkpoint provided by Neuraltalk2 (Karpathy et al., 2016). Captions for online test set evaluations are generated using beam search of size 2, but score histories on split validation set are based on captions generated without beam search (i.e. max sampling at each time-step).

Ablation on Object Locations and Counts:. We setup an experiment where we remove the input locations from the OBJ2TEXT encoder to study the effects on the generated captions, and confirm whether the model is actually using spatial information during surface realization. In this restricted version of our model the LSTM encoder at each time step only takes the object category embedding vector as input. The OBJ2TEXT model additionally encodes different instances of the same object category in different time steps, potentially encoding in some of its hidden states information about how many objects of a particular class are in the image. For example, in the object annotation presented in the input in Figure 1, there are two instances of “person”. We perform an additional experiment where our model does not have access neither to object locations, nor the number of object instances by providing only a set of object categories. Note that in this set of experiments the object layouts are given as inputs, thus we assume full access to ground-truth object annotations, even in the test split. In the experimental results section we use the “-GT” postfix to indicate that input object layouts are obtained from ground-truth object annotations provided by the MS-COCO dataset.

Image Captioning Experiment: In this experiment we assess whether the image captioning model OBJ2TEXT-YOLO that only relies on object categories and locations could give comparable performance with a CNN-RNN model based on Neuraltalk2 (Karpathy et al., 2016) that has full access to visual image features. We also explore how much does a combined OBJ2TEXT-YOLO + CNN-RNN model could improve over a CNN-RNN model by fusing object counts and location information that is not explicitly encoded in a traditional CNN-RNN approach.

Human Evaluation Protocol. We use a two-alternative forced-choice evaluation (2AFC) approach to compare two methods that generate captions. For this, we setup a task on Amazon Mechanical Turk where users are presented with an image and two alternative captions, and they have to choose the caption that best describes the image. Users are not prompted to use any single criteria but rather a holistic assessment of the captions, including their semantics, syntax, and the degree to which they describe the image content. In our experiment we randomly sample 500 captions generated by various models for MS COCO online test set images, and use three users per image to obtain annotations. Note that three users choosing randomly between two options have a chance of 25% to select the same caption for a given image. In our experiments comparing method A vs method B, we report the percentage of times A was picked over B (Choice-all), the percentage of times all users selected the same method, either A or B, (Agreement), and the percentage of times A was picked over B only for these cases where all users agreed (Choice-agreement).

6 Results

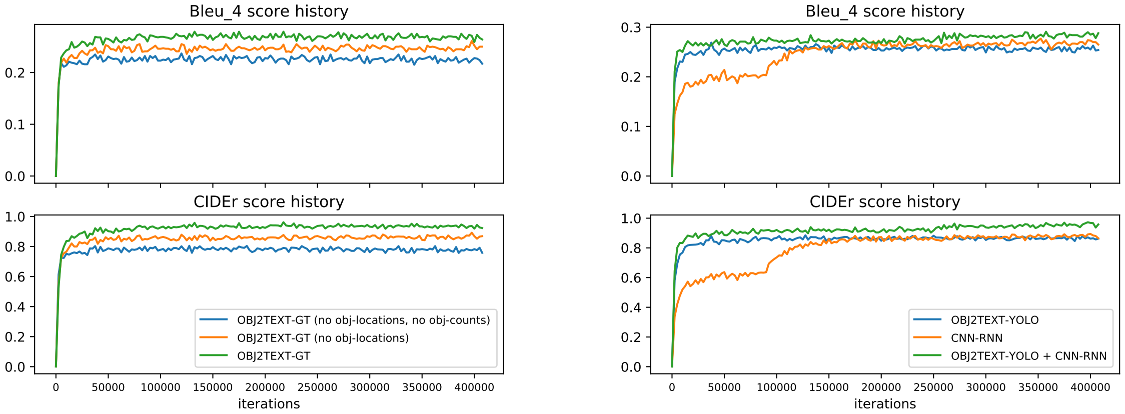
Impact of Object Locations and Counts: Figure 3a shows the CIDEr (Vedantam et al., 2015a), and BLEU-4 (Papineni et al., 2002) score history on our validation set during 400k iterations of training of OBJ2TEXT, as well as a version of our model that does not use object locations, and a version of our model that does not use neither object locations nor object counts. These results show that our model is effectively using both object locations and counts to generate better captions, and absence of any one of these two cues affects performance. Table 1 confirms these results on the test split after a full round of training.

Furthermore, human evaluation results in the first row of Table 2 show that the OBJ2TEXT model with access to object locations is preferred by users, especially in cases where all evaluators agreed on their choice (62% over the baseline that does not have access to locations). In Figure 4 we additionally present qualitative examples showing predictions side-by-side between OBJ2TEXT-GT and OBJ2TEXT-GT (no obj-locations). These results indicate that 1) perhaps not surprisingly, object counts is useful for generating better quality descriptions, and 2) object location information when properly encoded, is an important cue for generating more accurate descriptions. We additionally implemented a nearest neighbor baseline by representing the objects in the input layout using an orderless bag-of-words representation of object counts and the CIDEr score on the test split was only 0.387.

On top of OBJ2TEXT we additionally experimented with the global attention model proposed in (Luong et al., 2015) so that a weighted combination of the encoder hidden states are forwarded to the decoding neural language model, however we did not notice any overall gains in terms of accuracy from this formulation. We observed that this model provided gains only for larger input sequences where it is more likely that the LSTM network forgets its past history (Bahdanau et al., 2015). However in MS-COCO the average number of objects in each image is rather modest, so the last hidden state can capture well the overall nuances of the visual input.

Object Layout Encoding for Image Captioning: Figure 3b shows the CIDEr, and BLEU-4 score history on the validation set during 400k iterations of training of OBJ2TEXT-YOLO, CNN-RNN, and their combination. These results show that OBJ2TEXT-YOLO performs surprisingly close to CNN-RNN, and the model resulting from combining the two, clearly outperforms each method alone. Table 3 shows MS-COCO evaluation results on the test set using their online benchmark service, and confirms results obtained in the validation split, where CNN-RNN seems to have only a slight edge over OBJ2TEXT-YOLO which lacks access to pixel data after the object detection stage. Human evaluation results in Table 2 rows 2, and 3, further confirm these findings. These results show that meaningful descriptions could be generated solely based on object categories and locations information, even without access to color and texture input.

The combined model performs better than the two models, improving the CIDEr score of the basic CNN-RNN model from 0.863 to 0.950, and human evaluation results show that the combined model is preferred over the basic CNN-RNN model for 65.3% of the images for which all evaluators were in agreement about the selected method. These results show that explicitly encoded object counts and location information, which is often overlooked in traditional image captioning approaches, could boost the performance of existing models. Intuitively, object lay-



(a) Score histories of lesioned versions of the proposed model for the task of object layout captioning.

(b) Score histories of image captioning models. Performance boosts of CNN-RNN and combined model around iteration 100K and 250K are due to fine-tuning of the image CNN model.

Figure 3: Score histories of various models on the MS COCO split validation set.

Method	Bleu_4	CIDEr	METEOR	ROUGE-L
OBJ2TEXT-GT (no obj-locations, counts)	0.21	0.759	0.215	0.464
OBJ2TEXT-GT (no obj-locations)	0.233	0.837	0.222	0.482
OBJ2TEXT-GT	**0.253**	**0.922**	**0.238**	**0.507**

Table 1: Performance of lesioned versions of the proposed model on the MS COCO split test set.

out and visual features are complementary: neural network models for visual feature extraction are trained on a classification task where object-level information such as number of instances and locations are ignored in the objective. Object layouts on the other hand, contain categories and their bounding-boxes but don't have access to rich image features such as image background, object attributes and objects with categories not present in the object detection vocabulary.

Figure 5 provides a three-way comparison of captions generated by the three image captioning models, with preferred captions by human evaluators annotated in bold text. Analysis on actual outputs gives us insights into the benefits of combing object layout information and visual features obtained using a CNN. Our OBJ2TEXT-YOLO model makes many mistakes because of lack of image context information since it only has access to object layout, while CNN-RNN makes many mistakes because the visual recognition model is imperfect at predicting the correct content. The combined model is usually able to generate more accurate and comprehensive descriptions.

In this work we only explored encoding spatial information with object labels, but object labels could be readily augmented with rich semantic features that are more detailed descriptions of objects or image regions. For example, the work of You et al. (2016) and Yao et al. (2016b) showed that visual features trained with semantic concepts (text entities mentioned in captions) instead of object labels is useful for image captioning, although they didn't consider encoding semantic concepts with spatial information. In case of object annotations the MS-COCO dataset only provides object labels and bounding-boxes, but there are other datasets such as Flick30K Entities (Plummer et al., 2015), and the Visual Genome dataset (Krishna et al., 2017) that provide richer region-to-phrase correspondence annotations. In addition, the fusion of object counts and spatial information with CNN visual features could in principle benefit other vision and language tasks such as visual question answering. We leave these possible extensions as future work.

7 Conclusion

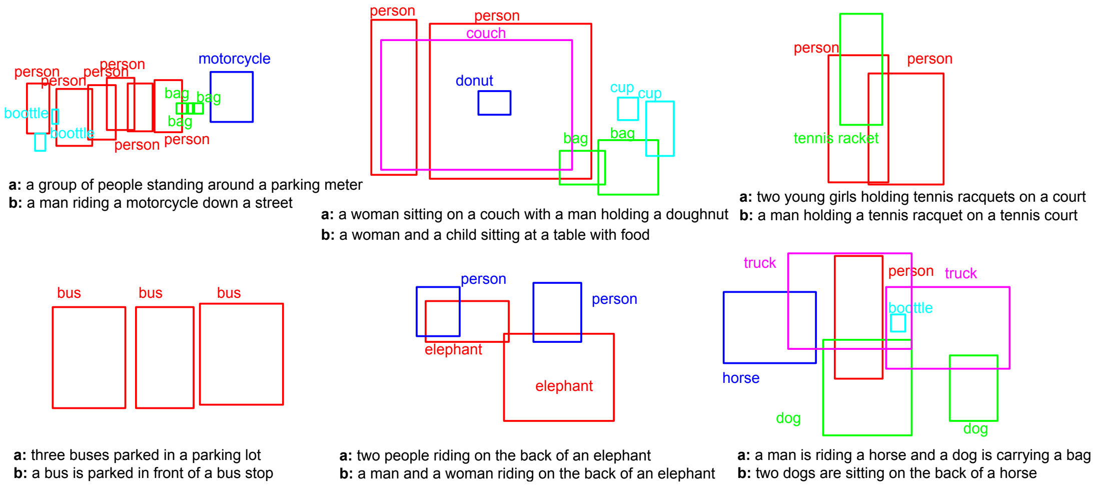
We introduced OBJ2TEXT, a sequence-to-sequence model to generate visual descriptions for object layouts where only categories and locations are specified. Our proposed model

Alternatives	Choice-all	Choice-agreement	Agreement
OBJ2TEXT-GT vs. OBJ2TEXT-GT (no obj-locations)	54.1%	62.1%	40.6%
OBJ2TEXT-YOLO vs. CNN+RNN	45.6%	40.6%	54.7%
OBJ2TEXT-YOLO + CNN-RNN vs. CNN-RNN	58.1%	65.3%	49.5%
OBJ2TEXT-GT vs. HUMAN	23.6%	9.9%	58.8%

Table 2: Human evaluation results using two-alternative forced choice evaluation. Choice-all is percentage the first alternative was chosen. Choice-agreement is percentage the first alternative was chosen only when all annotators agreed. Agreement is percentage where all annotators agreed (random is 25%).

MS COCO Test Set Performance	CIDEr	ROUGE-L	METEOR	B-4	B-3	B-2	B-1
5-Refs							
OBJ2TEXT-YOLO	0.830	0.497	0.228	0.262	0.361	0.500	0.681
CNN-RNN	0.857	0.514	0.237	0.283	0.387	0.529	0.705
OBJ2TEXT-YOLO + CNN-RNN	**0.932**	**0.528**	**0.250**	**0.300**	**0.404**	**0.546**	**0.719**
40-Refs							
OBJ2TEXT-YOLO	0.853	0.636	0.305	0.508	0.624	0.746	0.858
CNN-RNN	0.863	0.654	0.318	0.540	0.656	0.775	0.877
OBJ2TEXT-YOLO + CNN-RNN	**0.950**	**0.671**	**0.334**	**0.569**	**0.686**	**0.802**	**0.896**

Table 3: The 5-Refs and 40-Refs performances of OBJ2TEXT-YOLO, CNN-RNN and the combined approach on the MS COCO online test set. The 5-Refs performance is measured using 5 ground-truth reference captions, while 40-Refs performance is measured using 40 ground-truth reference captions.



a: a group of people standing around a parking meter
b: a man riding a motorcycle down a street

a: a woman sitting on a couch with a man holding a doughnut
b: a woman and a child sitting at a table with food

a: two young girls holding tennis racquets on a court
b: a man holding a tennis racquet on a tennis court

a: three buses parked in a parking lot
b: a bus is parked in front of a bus stop

a: two people riding on the back of an elephant
b: a man and a woman riding on the back of an elephant

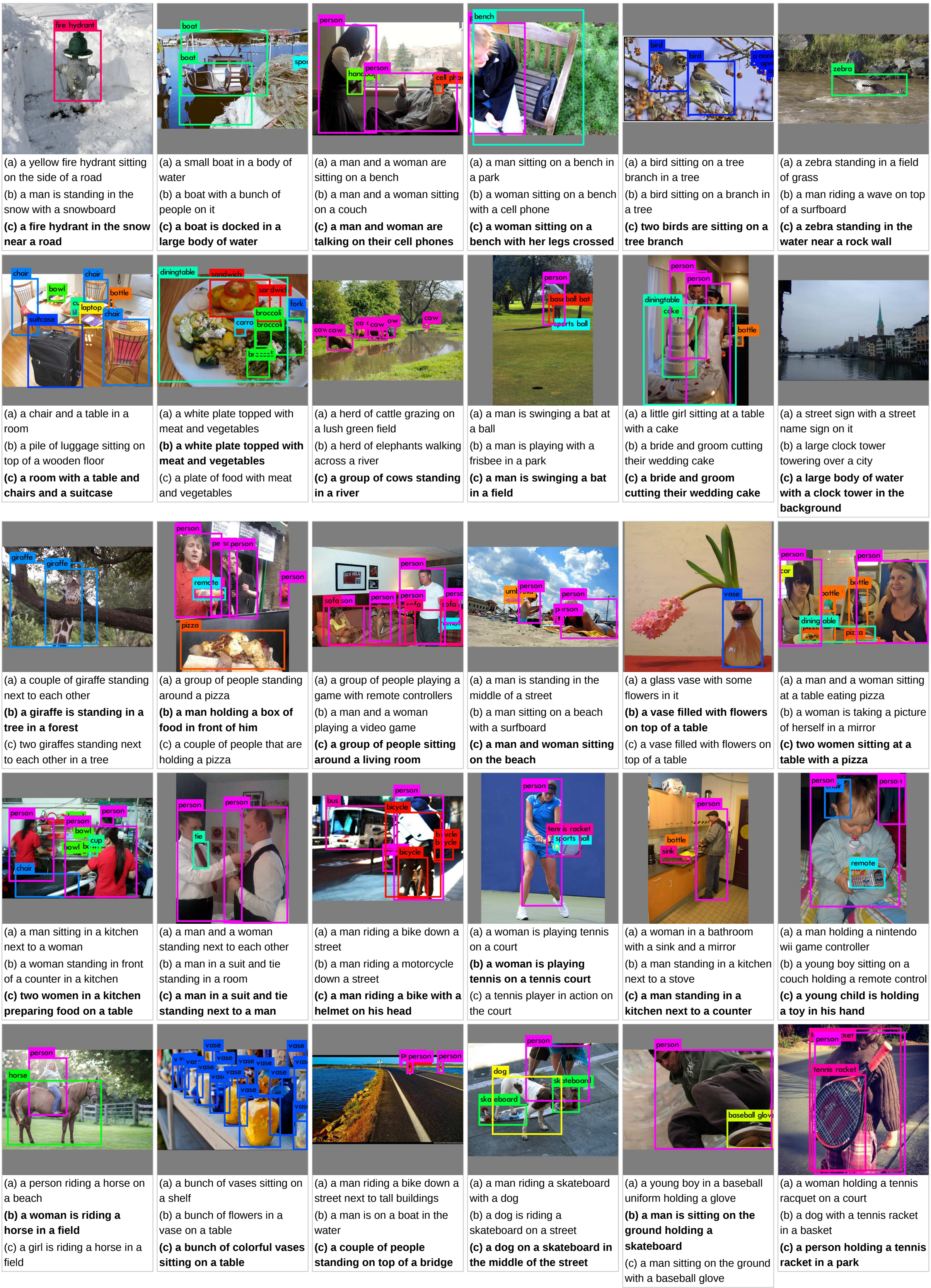
a: a man is riding a horse and a dog is carrying a bag
b: two dogs are sitting on the back of a horse

Figure 4: Qualitative examples comparing generated captions of (**a**) OBJ2TEXT-GT, and (**b**) OBJ2TEXT-GT (no obj-locations).

shows that an orderless visual input representation of concepts is not enough to produce good descriptions, but object extents, locations, and object counts, all contribute to generate more accurate image descriptions. Crucially we show that our encoding mechanism is able to capture useful spatial information using an LSTM network to produce image descriptions, even when the input is provided as a sequence rather than as an explicit 2D representation of objects. Additionally, using our proposed OBJ2TEXT model in combination with an existing image captioning model and a robust object detector we showed improved results in the task of image captioning.

Acknowledgments

This work was supported in part by an NVIDIA Hardware Grant. We are also thankful for the feedback from Mark Yatskar and anonymous reviewers of this paper.



(a) a yellow fire hydrant sitting on the side of a road
(b) a man is standing in the snow with a snowboard
(c) a fire hydrant in the snow near a road

(a) a small boat in a body of water
(b) a boat with a bunch of people on it
(c) a boat is docked in a large body of water

(a) a man and a woman are sitting on a bench
(b) a man and a woman sitting on a couch
(c) a man and woman are talking on their cell phones

(a) a man sitting on a bench in a park
(b) a woman sitting on a bench with a cell phone
(c) a woman sitting on a bench with her legs crossed

(a) a bird sitting on a tree branch in a tree
(b) a bird sitting on a branch in a tree
(c) two birds are sitting on a tree branch

(a) a zebra standing in a field of grass
(b) a man riding a wave on top of a surfboard
(c) a zebra standing in the water near a rock wall

(a) a chair and a table in a room
(b) a pile of luggage sitting on top of a wooden floor
(c) a room with a table and chairs and a suitcase

(a) a white plate topped with meat and vegetables
(b) a white plate topped with meat and vegetables
(c) a plate of food with meat and vegetables

(a) a herd of cattle grazing on a lush green field
(b) a herd of elephants walking across a river
(c) a group of cows standing in a river

(a) a man is swinging a bat at a ball
(b) a man is playing with a frisbee in a park
(c) a man is swinging a bat in a field

(a) a little girl sitting at a table with a cake
(b) a bride and groom cutting their wedding cake
(c) a bride and groom cutting their wedding cake

(a) a street sign with a street name sign on it
(b) a large clock tower towering over a city
(c) a large body of water with a clock tower in the background

(a) a couple of giraffe standing next to each other
(b) a giraffe is standing in a tree in a forest
(c) two giraffes standing next to each other in a tree

(a) a group of people standing around a pizza
(b) a man holding a box of food in front of him
(c) a couple of people that are holding a pizza

(a) a group of people playing a game with remote controllers
(b) a man and a woman playing a video game
(c) a group of people sitting around a living room

(a) a man is standing in the middle of a street
(b) a man sitting on a beach with a surfboard
(c) a man and woman sitting on the beach

(a) a glass vase with some flowers in it
(b) a vase filled with flowers on top of a table
(c) a vase filled with flowers on top of a table

(a) a man and a woman sitting at a table eating pizza
(b) a woman is taking a picture of herself in a mirror
(c) two women sitting at a table with a pizza

(a) a man sitting in a kitchen next to a woman
(b) a woman standing in front of a counter in a kitchen
(c) two women in a kitchen preparing food on a table

(a) a man and a woman standing next to each other
(b) a man in a suit and tie standing in a room
(c) a man in a suit and tie standing next to a man

(a) a man riding a bike down a street
(b) a man riding a motorcycle down a street
(c) a man riding a bike with a helmet on his head

(a) a woman is playing tennis on a court
(b) a woman is playing tennis on a tennis court
(c) a tennis player in action on the court

(a) a woman in a bathroom with a sink and a mirror
(b) a man standing in a kitchen next to a stove
(c) a man standing in a kitchen next to a counter

(a) a man holding a nintendo wii game controller
(b) a young boy sitting on a couch holding a remote control
(c) a young child is holding a toy in his hand

(a) a person riding a horse on a beach
(b) a woman is riding a horse in a field
(c) a girl is riding a horse in a field

(a) a bunch of vases sitting on a shelf
(b) a bunch of flowers in a vase on a table
(c) a bunch of colorful vases sitting on a table

(a) a man riding a bike down a street next to tall buildings
(b) a man is on a boat in the water
(c) a couple of people standing on top of a bridge

(a) a man riding a skateboard with a dog
(b) a dog is riding a skateboard on a street
(c) a dog on a skateboard in the middle of the street

(a) a young boy in a baseball uniform holding a glove
(b) a man is sitting on the ground holding a skateboard
(c) a man sitting on the ground with a baseball glove

(a) a woman holding a tennis racquet on a court
(b) a dog with a tennis racket in a basket
(c) a person holding a tennis racket in a park

Figure 5: Qualitative examples comparing the generated captions of (a) OBJ2TEXT-YOLO, (b) CNN-RNN and (c) OBJ2TEXT-YOLO + CNN-RNN. Images are selected from the 500 human evaluation images and annotated with YOLO object detection results. Captions preferred by human evaluators with agreement are highlighted in bold text.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2625–2634.

Desmond Elliott and Frank Keller. 2013. Image description using visual dependency representations. In *EMNLP*, volume 13, pages 1292–1302.

Benjamin Eysenbach, Carl Vondrick, and Antonio Torralba. 2016. Who is mistaken? *arXiv preprint arXiv:1612.01175*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. 2015. From captions to visual concepts and back. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1473–1482.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *European conference on computer vision*, pages 15–29. Springer.

David F Fouhey and C Lawrence Zitnick. 2014. Predicting object dynamics in scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2019–2026.

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

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. 2016. Natural language object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4555–4564.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083, Berlin, Germany. Association for Computational Linguistics.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137.

Andrej Karpathy et al. 2016. Neuraltalk2. https://github.com/karpathy/neuraltalk2/.

Ioannis Konstas and Mirella Lapata. 2012. Unsupervised concept-to-text generation with hypergraphs. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 752–761.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Neural Information Processing Systems (NIPS)*, pages 1097–1105.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pages 145–150. Association for Computational Linguistics.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-rnn). *ICLR*.

Rebecca Mason and Eugene Charniak. 2014. Nonparametric method for data-driven image captioning. In *ACL (2)*, pages 592–598.

Vicente Ordonez, Xufeng Han, Polina Kuznetsova, Girish Kulkarni, Margaret Mitchell, Kota Yamaguchi, Karl Stratos, Amit Goyal, Jesse Dodge, Alyssa Mensch, III Daume, Hal, Alexander C. Berg, Yejin Choi, and Tamara L. Berg. 2015. Large scale retrieval and generation of image descriptions. *International Journal of Computer Vision*, pages 1–14.

Vicente Ordonez, Girish Kulkarni, and Tamara L Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *Advances in Neural Information Processing Systems*, pages 1143–1151.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of* *the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE international conference on computer vision*, pages 2641–2649.

Arnau Ramisa, JK Wang, Ying Lu, Emmanuel Dellandrea, Francesc Moreno-Noguer, and Robert Gaizauskas. 2015. Combining geometric, textual and visual features for predicting prepositions in image descriptions. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 214–220. Association for Computational Linguistics.

Joseph Redmon and Ali Farhadi. 2017. YOLO9000: better, faster, stronger. In *Computer Vision and Pattern Recognition (CVPR)*.

Anna Rohrbach, Marcus Rohrbach, Ronghang Hu, Trevor Darrell, and Bernt Schiele. 2016. Grounding of textual phrases in images by reconstruction. In *European Conference on Computer Vision*, pages 817–834. Springer.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Allen Schmaltz, Alexander M. Rush, and Stuart Shieber. 2016. Word ordering without syntax. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2319–2324, Austin, Texas.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*.

Frank A Smadja and Kathleen R McKeown. 1990. Automatically extracting and representing collocations for language generation. In *Annual meeting of the Association for Computational Linguistics (ACL)*, pages 252–259.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015a. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575.

Ramakrishna Vedantam, Xiao Lin, Tanmay Batra, C Lawrence Zitnick, and Devi Parikh. 2015b. Learning common sense through visual abstraction. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2542–2550.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1711–1721, Lisbon, Portugal.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Li Yao, Nicolas Ballas, Kyunghyun Cho, John R. Smith, and Yoshua Bengio. 2016a. Oracle performance for visual captioning. In *British Machine Vision Conference (BMVC)*.

Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu, and Tao Mei. 2016b. Boosting image captioning with attributes. *arXiv preprint arXiv:1611.01646*.

Mark Yatskar, Vicente Ordonez, and Ali Farhadi. 2016. Stating the obvious: Extracting visual common sense knowledge. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 193–198, San Diego, California. Association for Computational Linguistics.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4651–4659.

C Lawrence Zitnick and Devi Parikh. 2013. Bringing semantics into focus using visual abstraction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3009–3016.

C Lawrence Zitnick, Devi Parikh, and Lucy Vanderwende. 2013. Learning the visual interpretation of sentences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1681–1688.

End-to-end Neural Coreference Resolution

Kenton Lee[†], Luheng He[†], Mike Lewis[‡], and Luke Zettlemoyer[†*]

[†] Paul G. Allen School of Computer Science & Engineering, Univ. of Washington, Seattle, WA
[*] Allen Institute for Artificial Intelligence, Seattle WA
{kentonl, luheng, lsz}@cs.washington.edu

[‡] Facebook AI Research, Menlo Park, CA
mikelewis@fb.com

Abstract

We introduce the first end-to-end coreference resolution model and show that it significantly outperforms all previous work without using a syntactic parser or hand-engineered mention detector. The key idea is to directly consider all spans in a document as potential mentions and learn distributions over possible antecedents for each. The model computes span embeddings that combine context-dependent boundary representations with a head-finding attention mechanism. It is trained to maximize the marginal likelihood of gold antecedent spans from coreference clusters and is factored to enable aggressive pruning of potential mentions. Experiments demonstrate state-of-the-art performance, with a gain of 1.5 F1 on the OntoNotes benchmark and by 3.1 F1 using a 5-model ensemble, despite the fact that this is the first approach to be successfully trained with no external resources.

1 Introduction

We present the first state-of-the-art coreference resolution model that is learned end-to-end given only gold mention clusters. All recent coreference models, including neural approaches that achieved impressive performance gains (Wiseman et al., 2016; Clark and Manning, 2016b,a), rely on syntactic parsers, both for head-word features and as the input to carefully hand-engineered mention proposal algorithms. We demonstrate for the first time that these resources are not required, and in fact performance can be improved significantly without them, by training an end-to-end neural model that jointly learns which spans are entity mentions and how to best cluster them.

Our model reasons over the space of all spans up to a maximum length and directly optimizes the marginal likelihood of antecedent spans from gold coreference clusters. It includes a *span*-ranking model that decides, for each span, which of the previous spans (if any) is a good antecedent. At the core of our model are vector embeddings representing spans of text in the document, which combine context-dependent boundary representations with a head-finding attention mechanism over the span. The attention component is inspired by parser-derived head-word matching features from previous systems (Durrett and Klein, 2013), but is less susceptible to cascading errors. In our analyses, we show empirically that these learned attention weights correlate strongly with traditional headedness definitions.

Scoring all span pairs in our end-to-end model is impractical, since the complexity would be quartic in the document length. Therefore we factor the model over unary mention scores and pairwise antecedent scores, both of which are simple functions of the learned span embedding. The unary mention scores are used to prune the space of spans and antecedents, to aggressively reduce the number of pairwise computations.

Our final approach outperforms existing models by 1.5 F1 on the OntoNotes benchmark and by 3.1 F1 using a 5-model ensemble. It is not only accurate, but also relatively interpretable. The model factors, for example, directly indicate whether an absent coreference link is due to low mention scores (for either span) or a low score from the mention ranking component. The head-finding attention mechanism also reveals which mention-internal words contribute most to coreference decisions. We leverage this overall interpretability to do detailed quantitative and qualitative analyses, providing insights into the strengths and weaknesses of the approach.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 188–197
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

2 Related Work

Machine learning methods have a long history in coreference resolution (see Ng (2010) for a detailed survey). However, the learning problem is challenging and, until very recently, hand-engineered systems built on top of automatically produced parse trees (Raghunathan et al., 2010) outperformed all learning approaches. Durrett and Klein (2013) showed that highly lexical learning approaches reverse this trend, and more recent neural models (Wiseman et al., 2016; Clark and Manning, 2016b,a) have achieved significant performance gains. However, all of these models still use parsers for head features and include highly engineered mention proposal algorithms.[1] Such pipelined systems suffer from two major drawbacks: (1) parsing mistakes can introduce cascading errors and (2) many of the hand-engineered rules do not generalize to new languages or domains. We present the first non-pipelined results, while providing further performance gains.

More generally, a wide variety of approaches for learning coreference models have been proposed. They can typically be categorized as (1) mention-pair classifiers (Ng and Cardie, 2002; Bengtson and Roth, 2008), (2) entity-level models (Haghighi and Klein, 2010; Clark and Manning, 2015, 2016b; Wiseman et al., 2016), (3) latent-tree models (Fernandes et al., 2012; Björkelund and Kuhn, 2014; Martschat and Strube, 2015), or (4) mention-ranking models (Durrett and Klein, 2013; Wiseman et al., 2015; Clark and Manning, 2016a). Our span-ranking approach is most similar to mention ranking, but we reason over a larger space by jointly detecting mentions and predicting coreference.

3 Task

We formulate the task of end-to-end coreference resolution as a set of decisions for every possible span in the document. The input is a document D containing T words along with metadata such as speaker and genre information.

Let $N = \frac{T(T+1)}{2}$ be the number of possible text spans in D. Denote the start and end indices of a span i in D respectively by START(i) and END(i), for $1 \leq i \leq N$. We assume an ordering of the spans based on START(i); spans with the same start index are ordered by END(i).

The task is to assign to each span i an antecedent y_i. The set of possible assignments for each y_i is $\mathcal{Y}(i) = \{\epsilon, 1, \ldots, i-1\}$, a dummy antecedent ϵ and all preceding spans. True antecedents of span i, i.e. span j such that $1 \leq j \leq i-1$, represent coreference links between i and j. The dummy antecedent ϵ represents two possible scenarios: (1) the span is not an entity mention or (2) the span is an entity mention but it is not coreferent with any previous span. These decisions implicitly define a final clustering, which can be recovered by grouping all spans that are connected by a set of antecedent predictions.

4 Model

We aim to learn a conditional probability distribution $P(y_1, \ldots, y_N \mid D)$ whose most likely configuration produces the correct clustering. We use a product of multinomials for each span:

$$\begin{aligned} P(y_1, \ldots, y_N \mid D) &= \prod_{i=1}^{N} P(y_i \mid D) \\ &= \prod_{i=1}^{N} \frac{\exp(s(i, y_i))}{\sum_{y' \in \mathcal{Y}(i)} \exp(s(i, y'))} \end{aligned}$$

where $s(i, j)$ is a pairwise score for a coreference link between span i and span j in document D. We omit the document D from the notation when the context is unambiguous. There are three factors for this pairwise coreference score: (1) whether span i is a mention, (2) whether span j is a mention, and (3) whether j is an antecedent of i:

$$s(i,j) = \begin{cases} 0 & j = \epsilon \\ s_{\mathrm{m}}(i) + s_{\mathrm{m}}(j) + s_{\mathrm{a}}(i,j) & j \neq \epsilon \end{cases}$$

Here $s_{\mathrm{m}}(i)$ is a unary score for span i being a mention, and $s_{\mathrm{a}}(i,j)$ is pairwise score for span j being an antecedent of span i.

By fixing the score of the dummy antecedent ϵ to 0, the model predicts the best scoring antecedent if any non-dummy scores are positive, and it abstains if they are all negative.

A challenging aspect of this model is that its size is $\mathcal{O}(T^4)$ in the document length. As we will see in Section 5, the above factoring enables aggressive pruning of spans that are unlikely to belong to a coreference cluster according the mention score $s_{\mathrm{m}}(i)$.

[1] For example, Raghunathan et al. (2010) use rules to remove pleonastic mentions of *it* detected by 12 lexicalized regular expressions over English parse trees.

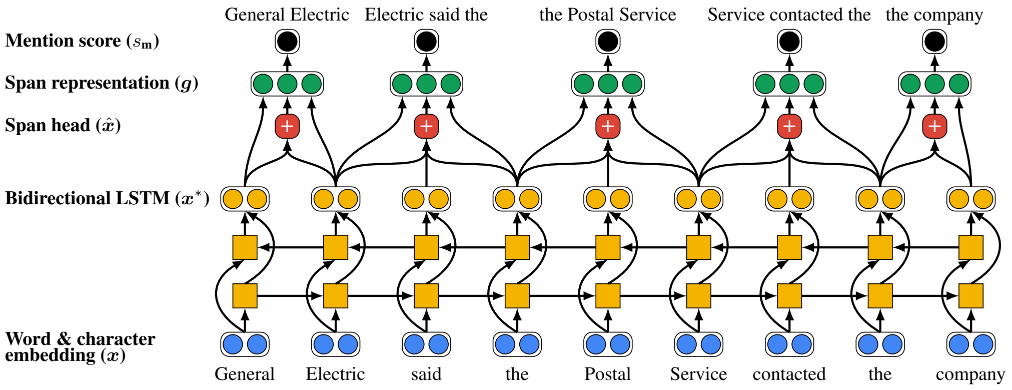


Figure 1: First step of the end-to-end coreference resolution model, which computes embedding representations of spans for scoring potential entity mentions. Low-scoring spans are pruned, so that only a manageable number of spans is considered for coreference decisions. In general, the model considers all possible spans up to a maximum width, but we depict here only a small subset.

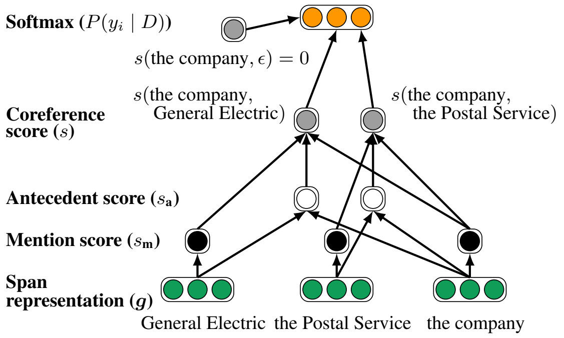

Softmax ($P(y_i \mid D)$)
s(the company, ϵ) = 0
s(the company, General Electric)
s(the company, the Postal Service)
Coreference score (s)
Antecedent score (s_a)
Mention score (s_m)
Span representation (g)
General Electric
the Postal Service
the company

Figure 2: Second step of our model. Antecedent scores are computed from pairs of span representations. The final coreference score of a pair of spans is computed by summing the mention scores of both spans and their pairwise antecedent score.

Scoring Architecture We propose an end-to-end neural architecture that computes the above scores given the document and its metadata.

At the core of the model are vector representations $\boldsymbol{g}_i$ for each possible span i, which we describe in detail in the following section. Given these span representations, the scoring functions above are computed via standard feed-forward neural networks:

$$s_\mathrm{m}(i) = \boldsymbol{w}_\mathrm{m} \cdot \mathrm{FFNN}_\mathrm{m}(\boldsymbol{g}_i)$$
$$s_\mathrm{a}(i,j) = \boldsymbol{w}_\mathrm{a} \cdot \mathrm{FFNN}_\mathrm{a}([\boldsymbol{g}_i, \boldsymbol{g}_j, \boldsymbol{g}_i \circ \boldsymbol{g}_j, \phi(i,j)])$$

where $\cdot$ denotes the dot product, $\circ$ denotes element-wise multiplication, and FFNN denotes a feed-forward neural network that computes a non-linear mapping from input to output vectors.

The antecedent scoring function $s_\mathrm{a}(i,j)$ includes explicit element-wise similarity of each span $\boldsymbol{g}_i \circ \boldsymbol{g}_j$ and a feature vector $\phi(i,j)$ encoding speaker and genre information from the metadata and the distance between the two spans.

Span Representations Two types of information are crucial to accurately predicting coreference links: the context surrounding the mention span and the internal structure within the span. We use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to encode the lexical information of both the inside and outside of each span. We also include an attention mechanism over words in each span to model head words.

We assume vector representations of each word $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$, which are composed of fixed pretrained word embeddings and 1-dimensional convolution neural networks (CNN) over characters (see Section 7.1 for details)

To compute vector representations of each span, we first use bidirectional LSTMs to encode every word in its context:

$$\begin{aligned}
\boldsymbol{f}_{t,\delta} &= \sigma(\mathbf{W}_\mathrm{f}[\boldsymbol{x}_t, \boldsymbol{h}_{t+\delta,\delta}] + \boldsymbol{b}_\mathrm{i}) \\
\boldsymbol{o}_{t,\delta} &= \sigma(\mathbf{W}_\mathrm{o}[\boldsymbol{x}_t, \boldsymbol{h}_{t+\delta,\delta}] + \boldsymbol{b}_\mathrm{o}) \\
\widetilde{\boldsymbol{c}}_{t,\delta} &= \tanh(\mathbf{W}_\mathrm{c}[\boldsymbol{x}_t, \boldsymbol{h}_{t+\delta,\delta}] + \boldsymbol{b}_\mathrm{c}) \\
\boldsymbol{c}_{t,\delta} &= \boldsymbol{f}_{t,\delta} \circ \widetilde{\boldsymbol{c}}_{t,\delta} + (\mathbf{1} - \boldsymbol{f}_{t,\delta}) \circ \boldsymbol{c}_{t+\delta,\delta} \\
\boldsymbol{h}_{t,\delta} &= \boldsymbol{o}_{t,\delta} \circ \tanh(\boldsymbol{c}_{t,\delta}) \\
\boldsymbol{x}^*_t &= [\boldsymbol{h}_{t,1}, \boldsymbol{h}_{t,-1}]
\end{aligned}$$

where $\delta \in \{-1, 1\}$ indicates the directionality of each LSTM, and $\boldsymbol{x}^*_t$ is the concatenated output of the bidirectional LSTM. We use independent LSTMs for every sentence, since cross-sentence context was not helpful in our experiments.

Syntactic heads are typically included as features in previous systems (Durrett and Klein, 2013; Clark and Manning, 2016b,a). Instead of relying on syntactic parses, our model learns a task-specific notion of headedness using an attention mechanism (Bahdanau et al., 2014) over words in each span:

$$\alpha_t = \boldsymbol{w}_\alpha \cdot \text{FFNN}_\alpha(\boldsymbol{x}^*_t)$$

$$a_{i,t} = \frac{\exp(\alpha_t)}{\sum\limits_{k=\text{START}(i)}^{\text{END}(i)} \exp(\alpha_k)}$$

$$\hat{\boldsymbol{x}}_i = \sum_{t=\text{START}(i)}^{\text{END}(i)} a_{i,t} \cdot \boldsymbol{x}_t$$

where $\hat{\boldsymbol{x}}_i$ is a weighted sum of word vectors in span i. The weights $a_{i,t}$ are automatically learned and correlate strongly with traditional definitions of head words as we will see in Section 9.2.

The above span information is concatenated to produce the final representation $\boldsymbol{g}_i$ of span i:

$$\boldsymbol{g}_i = [\boldsymbol{x}^*_{\text{START}(i)}, \boldsymbol{x}^*_{\text{END}(i)}, \hat{\boldsymbol{x}}_i, \phi(i)]$$

This generalizes the recurrent span representations recently proposed for question-answering (Lee et al., 2016), which only include the boundary representations $\boldsymbol{x}^*_{\text{START}(i)}$ and $\boldsymbol{x}^*_{\text{END}(i)}$. We introduce the soft head word vector $\hat{\boldsymbol{x}}_i$ and a feature vector $\phi(i)$ encoding the size of span i.

5 Inference

The size of the full model described above is $O(T^4)$ in the document length T. To maintain computation efficiency, we prune the candidate spans greedily during both training and evaluation.

We only consider spans with up to L words and compute their unary mention scores $s_\text{m}(i)$ (as defined in Section 4). To further reduce the number of spans to consider, we only keep up to λT spans with the highest mention scores and consider only up to K antecedents for each. We also enforce non-crossing bracketing structures with a simple suppression scheme.[2] We accept spans in decreasing order of the mention scores, unless, when considering span i, there exists a previously accepted span j such that $\text{START}(i) < \text{START}(j) \leq \text{END}(i) < \text{END}(j) \lor \text{START}(j) < \text{START}(i) \leq \text{END}(j) < \text{END}(i)$.

Despite these aggressive pruning strategies, we maintain a high recall of gold mentions in our experiments (over 92% when $\lambda = 0.4$).

For the remaining mentions, the joint distribution of antecedents for each document is computed in a forward pass over a single computation graph. The final prediction is the clustering produced by the most likely configuration.

6 Learning

In the training data, only clustering information is observed. Since the antecedents are latent, we optimize the marginal log-likelihood of all correct antecedents implied by the gold clustering:

$$\log \prod_{i=1}^{N} \sum_{\hat{y} \in \mathcal{Y}(i) \cap \text{GOLD}(i)} P(\hat{y})$$

where $\text{GOLD}(i)$ is the set of spans in the gold cluster containing span i. If span i does not belong to a gold cluster or all gold antecedents have been pruned, $\text{GOLD}(i) = \{\epsilon\}$.

By optimizing this objective, the model naturally learns to prune spans accurately. While the initial pruning is completely random, only gold mentions receive positive updates. The model can quickly leverage this learning signal for appropriate credit assignment to the different factors, such as the mention scores s_m used for pruning.

Fixing score of the dummy antecedent to zero removes a spurious degree of freedom in the overall model with respect to mention detection. It also prevents the span pruning from introducing noise. For example, consider the case where span i has a single gold antecedent that was pruned, so $\text{GOLD}(i) = \{\epsilon\}$. The learning objective will only correctly push the scores of non-gold antecedents lower, and it cannot incorrectly push the score of the dummy antecedent higher.

This learning objective can be considered a span-level, cost-insensitive analog of the learning objective proposed by Durrett and Klein (2013). We experimented with these cost-sensitive alternatives, including margin-based variants (Wiseman et al., 2015; Clark and Manning, 2016a), but a simple maximum-likelihood objective proved to be most effective.

[2] The official CoNLL-2012 evaluation only considers predictions without crossing mentions to be valid. Enforcing this consistency is not inherently necessary in our model.

	MUC			B^3			$CEAF_{\phi_4}$			
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Avg. F1
Our model (ensemble)	**81.2**	**73.6**	**77.2**	**72.3**	**61.7**	**66.6**	**65.2**	**60.2**	**62.6**	**68.8**
Our model (single)	78.4	73.4	75.8	68.6	61.8	65.0	62.7	59.0	60.8	67.2
Clark and Manning (2016a)	79.2	70.4	74.6	69.9	58.0	63.4	63.5	55.5	59.2	65.7
Clark and Manning (2016b)	79.9	69.3	74.2	71.0	56.5	63.0	63.8	54.3	58.7	65.3
Wiseman et al. (2016)	77.5	69.8	73.4	66.8	57.0	61.5	62.1	53.9	57.7	64.2
Wiseman et al. (2015)	76.2	69.3	72.6	66.2	55.8	60.5	59.4	54.9	57.1	63.4
Clark and Manning (2015)	76.1	69.4	72.6	65.6	56.0	60.4	59.4	53.0	56.0	63.0
Martschat and Strube (2015)	76.7	68.1	72.2	66.1	54.2	59.6	59.5	52.3	55.7	62.5
Durrett and Klein (2014)	72.6	69.9	71.2	61.2	56.4	58.7	56.2	54.2	55.2	61.7
Björkelund and Kuhn (2014)	74.3	67.5	70.7	62.7	55.0	58.6	59.4	52.3	55.6	61.6
Durrett and Klein (2013)	72.9	65.9	69.2	63.6	52.5	57.5	54.3	54.4	54.3	60.3

Table 1: Results on the test set on the English data from the CoNLL-2012 shared task. The final column (Avg. F1) is the main evaluation metric, computed by averaging the F1 of MUC, B^3, and $CEAF_{\phi_4}$. We improve state-of-the-art performance by 1.5 F1 for the single model and by 3.1 F1.

7 Experiments

We use the English coreference resolution data from the CoNLL-2012 shared task (Pradhan et al., 2012) in our experiments. This dataset contains 2802 training documents, 343 development documents, and 348 test documents. The training documents contain on average 454 words and a maximum of 4009 words.

7.1 Hyperparameters

Word representations The word embeddings are a fixed concatenation of 300-dimensional GloVe embeddings (Pennington et al., 2014) and 50-dimensional embeddings from Turian et al. (2010), both normalized to be unit vectors. Out-of-vocabulary words are represented by a vector of zeros. In the character CNN, characters are represented as learned 8-dimensional embeddings. The convolutions have window sizes of 3, 4, and 5 characters, each consisting of 50 filters.

Hidden dimensions The hidden states in the LSTMs have 200 dimensions. Each feed-forward neural network consists of two hidden layers with 150 dimensions and rectified linear units (Nair and Hinton, 2010).

Feature encoding We encode speaker information as a binary feature indicating whether a pair of spans are from the same speaker. Following Clark and Manning (2016b), the distance features are binned into the following buckets [1, 2, 3, 4, 5-7, 8-15, 16-31, 32-63, 64+]. All features (speaker, genre, span distance, mention width) are represented as learned 20-dimensional embeddings.

Pruning We prune the spans such that the maximum span width $L = 10$, the number of spans per word $\lambda = 0.4$, and the maximum number of antecedents $K = 250$. During training, documents are randomly truncated to up to 50 sentences.

Learning We use ADAM (Kingma and Ba, 2014) for learning with a minibatch size of 1. The LSTM weights are initialized with random orthonormal matrices as described in Saxe et al. (2013). We apply 0.5 dropout to the word embeddings and character CNN outputs. We apply 0.2 dropout to all hidden layers and feature embeddings. Dropout masks are shared across timesteps to preserve long-distance information as described in Gal and Ghahramani (2016). The learning rate is decayed by 0.1% every 100 steps. The model is trained for up to 150 epochs, with early stopping based on the development set.

All code is implemented in TensorFlow (Abadi et al., 2015) and is publicly available. [3]

7.2 Ensembling

We also report ensemble experiments using five models trained with different random initializations. Ensembling is performed for both the span pruning and antecedent decisions.

At test time, we first average the mention scores $s_{\mathrm{m}}(i)$ over each model before pruning the spans.

[3] https://github.com/kentonl/e2e-coref

	Avg. F1	Δ
Our model (ensemble)	69.0	+1.3
Our model (single)	67.7	
– distance and width features	63.9	-3.8
– GloVe embeddings	65.3	-2.4
– speaker and genre metadata	66.3	-1.4
– head-finding attention	66.4	-1.3
– character CNN	66.8	-0.9
– Turian embeddings	66.9	-0.8

Table 2: Comparisons of our single model on the development data. The 5-model ensemble provides a 1.3 F1 improvement. The head-finding attention, features, and all word representations contribute significantly to the full model.

Given the same pruned spans, each model then computes the antecedent scores $s_a(i, j)$ separately, and they are averaged to produce the final scores.

8 Results

We report the precision, recall, and F1 for the standard MUC, B^3, and CEAF$_{\phi_4}$metrics using the official CoNLL-2012 evaluation scripts. The main evaluation is the average F1 of the three metrics.

8.1 Coreference Results

Table 1 compares our model to several previous systems that have driven substantial improvements over the past several years on the OntoNotes benchmark. We outperform previous systems in all metrics. In particular, our single model improves the state-of-the-art average F1 by 1.5, and our 5-model ensemble improves it by 3.1.

The most significant gains come from improvements in recall, which is likely due to our end-to-end setup. During training, pipelined systems typically discard any mentions that the mention detector misses, which for Clark and Manning (2016a) consists of more than 9% of the labeled mentions in the training data. In contrast, we only discard mentions that exceed our maximum mention width of 10, which accounts for less than 2% of the training mentions. The contribution of joint mention scoring is further discussed in Section 8.3

8.2 Ablations

To show the importance of each component in our proposed model, we ablate various parts of the architecture and report the average F1 on the development set of the data (see Figure 2).

	Avg. F1	Δ
Our model (joint mention scoring)	67.7	
w/ rule-based mentions	66.7	-1.0
w/ oracle mentions	85.2	+17.5

Table 3: Comparisons of of various mention proposal methods with our model on the development data. The rule-based mentions are derived from the mention detector from Raghunathan et al. (2010), resulting in a 1 F1 drop in performance. The oracle mentions are from the labeled clusters and improve our model by over 17.5 F1.

Features The distance between spans and the width of spans are crucial signals for coreference resolution, consistent with previous findings from other coreference models. They contribute 3.8 F1 to the final result.

Word representations Since our word embeddings are fixed, having access to a variety of word embeddings allows for a more expressive model without overfitting. We hypothesis that the different learning objectives of the GloVe and Turian embeddings provide orthogonal information (the former is word-order insensitive while the latter is word-order sensitive). Both embeddings contribute to some improvement in development F1.

The character CNN provides morphological information and a way to backoff for out-of-vocabulary words. Since coreference decisions often involve rare named entities, we see a contribution of 0.9 F1 from character-level modeling.

Metadata Speaker and genre indicators many not be available in downstream applications. We show that performance degrades by 1.4 F1 without them, but is still on par with previous state-of-the-art systems that assume access to this metadata.

Head-finding attention Ablations also show a 1.3 F1 degradation in performance without the attention mechanism for finding task-specific heads. As we will see in Section 9.4, the attention mechanism should not be viewed as simply an approximation of syntactic heads. In many cases, it is beneficial to pay attention to multiple words that are useful specifically for coreference but are not traditionally considered to be syntactic heads.

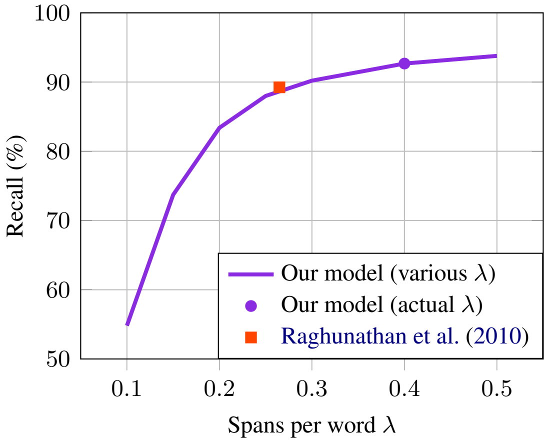


Figure 3: Proportion of gold mentions covered in the development data as we increase the number of spans kept per word. Recall is comparable to the mention detector of previous state-of-the-art systems given the same number of spans. Our model keeps 0.4 spans per word in our experiments, achieving 92.7% recall of gold mentions.

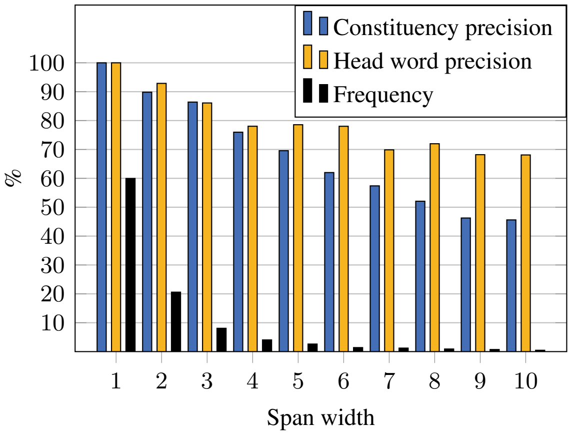


Figure 4: Indirect measure of mention precision using agreement with gold syntax. Constituency precision: % of unpruned spans matching syntactic constituents. Head word precision: % of unpruned constituents whose syntactic head word matches the most attended word. Frequency: % of gold spans with each width.

8.3 Comparing Span Pruning Strategies

To tease apart the contributions of improved mention scoring and improved coreference decisions, we compare the results of our model with alternate span pruning strategies. In these experiments, we use the alternate spans for both training and evaluation. As shown in Table 3, keeping mention candidates detected by the rule-based system over predicted parse trees (Raghunathan et al., 2010) degrades performance by 1 F1. We also provide oracle experiment results, where we keep exactly the mentions that are present in gold coreference clusters. With oracle mentions, we see an improvement of 17.5 F1, suggesting an enormous room for improvement if our model can produce better mention scores and anaphoricity decisions.

9 Analysis

To highlight the strengths and weaknesses of our model, we provide both quantitative and qualitative analyses. In the following discussion, we use predictions from the single model rather than the ensembled model.

9.1 Mention Recall

The training data only provides a weak signal for spans that correspond to entity mentions, since singleton clusters are not explicitly labeled. As a by product of optimizing marginal likelihood, our model automatically learns a useful ranking of spans via the unary mention scores from Section 4.

The top spans, according to the mention scores, cover a large portion of the mentions in gold clusters, as shown in Figure 3. Given a similar number of spans kept, our recall is comparable to the rule-based mention detector (Raghunathan et al., 2010) that produces 0.26 spans per word with a recall of 89.2%. As we increase the number of spans per word (λ in Section 5), we observe higher recall but with diminishing returns. In our experiments, keeping 0.4 spans per word results in 92.7% recall in the development data.

9.2 Mention Precision

While the training data does not offer a direct measure of mention precision, we can use the gold syntactic structures provided in the data as a proxy. Spans with high mention scores should correspond to syntactic constituents.

In Figure 4, we show the precision of top-scoring spans when keeping 0.4 spans per word. For spans with 2–5 words, 75–90% of the predictions are constituents, indicating that the vast majority of the mentions are syntactically plausible. Longer spans, which are all relatively rare, prove more difficult for the model, and precision drops to 46% for spans with 10 words.

1	**(A fire in a Bangladeshi garment factory)** has left at least 37 people dead and 100 hospitalized. Most of the deceased were killed in the crush as workers tried to flee **(the blaze)** in the four-story building.
	A fire in **(a Bangladeshi garment factory)** has left at least 37 people dead and 100 hospitalized. Most of the deceased were killed in the crush as workers tried to flee the blaze in **(the four-story building)**.
2	We are looking for **(a region of central Italy bordering the Adriatic Sea)**. **(The area)** is mostly mountainous and includes Mt. Corno, the highest peak of the Apennines. **(It)** also includes a lot of sheep, good clean-living, healthy sheep, and an Italian entrepreneur has an idea about how to make a little money of them.
3	**(The flight attendants)** have until 6:00 today to ratify labor concessions. **(The pilots')** union and ground crew did so yesterday.
4	**(Prince Charles and his new wife Camilla)** have jumped across the pond and are touring the United States making **(their)** first stop today in New York. It's Charles' first opportunity to showcase his new wife, but few Americans seem to care. Here's Jeanie Mowth. What a difference two decades make. **(Charles and Diana)** visited a JC Penney's on the prince's last official US tour. Twenty years later here's the prince with his new wife.
5	Also such location devices, **(some ships)** have smoke floats **(they)** can toss out so the man overboard will be able to use smoke signals as a way of trying to, let the rescuer locate **(them)**.

Table 4: Examples predictions from the development data. Each row depicts a single coreference cluster predicted by our model. Bold, parenthesized spans indicate mentions in the predicted cluster. The redness of each word indicates the weight of the head-finding attention mechanism ($a_{i,t}$ in Section 4).

9.3 Head Agreement

We also investigate how well the learned head preferences correlate with syntactic heads. For each of the top-scoring spans in the development data that correspond to gold constituents, we compute the word with the highest attention weight.

We plot in Figure 4 the proportion of these words that match syntactic heads. Agreement ranges between 68-93%, which is surprisingly high, since no explicit supervision of syntactic heads is provided. The model simply learns from the clustering data that these head words are useful for making coreference decisions.

9.4 Qualitative Analysis

Our qualitative analysis in Table 4 highlights the strengths and weaknesses of our model. Each row is a visualization of a single coreference cluster predicted by the model. Bolded spans in parentheses belong to the predicted cluster, and the redness of a word indicates its weight from the head-finding attention mechanism ($a_{i,t}$ in Section 4).

Strengths The effectiveness of the attention mechanism for making coreference decisions can be seen in Example 1. The model pays attention to *fire* in the span *A fire in a Bangladeshi garment factory*, allowing it to successfully predict the coreference link with *the blaze*. For a subspan of that mention, *a Bangladeshi garment factory*, the model pays most attention instead to *factory*, allowing it successfully predict the coreference link with *the four-story building*.

The task-specific nature of the attention mechanism is also illustrated in Example 4. The model generally pays attention to coordinators more than the content of the coordination, since coordinators, such as *and*, provide strong cues for plurality.

The model is capable of detecting relatively long and complex noun phrases, such as *a region of central Italy bordering the Adriatic Sea* in Example 2. It also appropriately pays attention to *region*, showing that the attention mechanism provides more than content-word classification. The context encoding provided by the bidirectional LSTMs is critical to making informative head word decisions.

Weaknesses A benefit of using neural models for coreference resolution is their ability to use word embeddings to capture similarity between words, a property that many traditional feature-based models lack. While this can dramatically increase recall, as demonstrated in Example 1, it is also prone to predicting false positive links when the model conflates paraphrasing with relatedness or similarity. In Example 3, the model mistakenly predicts a link between *The flight attendants* and *The pilots'*. The predicted head words *attendants* and *pilots* likely have nearby word embeddings, which is a signal used—and often overused—by the model. The same type of error is made in Example 4, where the model predicts a coreference link between *Prince Charles and his new wife Camilla* and *Charles and Diana*, two non-coreferent mentions that are similar in many ways. These mistakes suggest substantial room for improvement with word or span representations that can cleanly distinguish between equivalence, entailment, and alternation.

Unsurprisingly, our model does little in the uphill battle of making coreference decisions requiring world knowledge. In Example 5, the model incorrectly decides that *them* (in the context of *let the rescuer locate them*) is coreferent with *some ships*, likely due to plurality cues. However, an ideal model that uses common-sense reasoning would instead correctly infer that a rescuer is more likely to look for *the man overboard* rather than the ship from which he fell. This type of reasoning would require either (1) models that integrate external sources of knowledge with more complex inference or (2) a vastly larger corpus of training data to overcome the sparsity of these patterns.

10 Conclusion

We presented a state-of-the-art coreference resolution model that is trained end-to-end for the first time. Our final model ensemble improves performance on the OntoNotes benchmark by over 3 F1 without external preprocessing tools used by previous systems. We showed that our model implicitly learns to generate useful mention candidates from the space of all possible spans. A novel head-finding attention mechanism also learns a task-specific preference for head words, which we empirically showed correlate strongly with traditional head-word definitions.

While our model substantially pushes the state-of-the-art performance, the improvements are potentially complementary to a large body of work on various strategies to improve coreference resolution, including entity-level inference and incorporating world knowledge, which are important avenues for future work.

Acknowledgements

The research was supported in part by DARPA under the DEFT program (FA8750-13-2-0019), the ARO (W911NF-16-1-0121), the NSF (IIS-1252835, IIS-1562364), gifts from Google and Tencent, and an Allen Distinguished Investigator Award. We also thank the UW NLP group for helpful conversations and comments on the work.

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. TensorFlow: Large-scale Machine Learning on Heterogeneous Systems. *Software available from tensorflow.org*.

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

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 294–303. Association for Computational Linguistics.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *ACL*.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *ACL*.

Kevin Clark and Christopher D. Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Kevin Clark and Christopher D. Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. In *Association for Computational Linguistics (ACL)*.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *EMNLP*.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *TACL*, 2:477–490.

Eraldo Rezende Fernandes, Cícero Nogueira Dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 41–48. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1019–1027.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 385–393. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation*, 9(8):1735–1780.

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:405–418.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of ICML*.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 1396–1411. Association for Computational Linguistics.

Vincent Ng and Claire Cardie. 2002. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

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

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40. Association for Computational Linguistics.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multipass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 492–501. Association for Computational Linguistics.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A Simple and General Method for Semi-supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Sam Wiseman, Alexander M Rush, and Stuart M Shieber. 2016. Learning global features for coreference resolution. *arXiv preprint arXiv:1604.03035*.

Sam Wiseman, Alexander M. Rush, Stuart M. Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *ACL*.

Neural Net Models of Open-domain Discourse Coherence

Jiwei Li and Dan Jurafsky
Computer Science Department
Stanford University, Stanford, USA
jiweil,jurafsky@stanford.edu

Abstract

Discourse coherence is strongly associated with text quality, making it important to natural language generation and understanding. Yet existing models of coherence focus on measuring individual aspects of coherence (lexical overlap, rhetorical structure, entity centering) in narrow domains.

In this paper, we describe domain-independent neural models of discourse coherence that are capable of measuring multiple aspects of coherence in existing sentences and can maintain coherence while generating new sentences. We study both discriminative models that learn to distinguish coherent from incoherent discourse, and generative models that produce coherent text, including a novel neural latent-variable Markovian generative model that captures the latent discourse dependencies between sentences in a text.

Our work achieves state-of-the-art performance on multiple coherence evaluations, and marks an initial step in generating coherent texts given discourse contexts.

1 Introduction

Modeling discourse coherence (the way parts of a text are linked into a coherent whole) is essential for summarization (Barzilay and McKeown, 2005), text planning (Hovy, 1988; Marcu, 1997) question-answering (Verberne et al., 2007), and even psychiatric diagnosis (Elvevåg et al., 2007; Bedi et al., 2015).

Various frameworks exist, each tackling aspects of coherence. Lexical cohesion (Halliday and Hasan, 1976; Morris and Hirst, 1991) models chains of words and synonyms. Psychological models of discourse (Foltz et al., 1998; Foltz, 2007; McNamara et al., 2010) use LSA embeddings to generalize lexical cohesion. Relational models like RST (Mann and Thompson, 1988; Lascarides and Asher, 1991) define relations that hierarchically structure texts. The entity grid model (Barzilay and Lapata, 2008) and its extensions[1] capture the referential coherence of entities moving in and out of focus across a text. Each captures only a single aspect of coherence, and all focus on scoring existing sentences, rather than on generating coherent discourse for tasks like abstractive summarization.

Here we introduce two classes of neural models for discourse coherence. Our discriminative models induce coherence by treating human generated texts as coherent examples and texts with random sentence replacements as negative examples, feeding LSTM sentence embeddings of pairs of consecutive sentences to a classifier. These achieve state-of-the-art (96% accuracy) on the standard domain-specific sentence-pair-ordering dataset (Barzilay and Lapata, 2008), but suffer in a larger open-domain setting due to the small semantic space that negative sampling is able to cover.

Our generative models are based on augmenting encoder-decoder models with latent variables to model discourse relationships across sentences, including (1) a model that incorporates an HMM-LDA topic model into the generative model and (2) an end-to-end model that introduces a Markov-structured neural latent variable, inspired by recent work on training latent-variable recurrent nets (Bowman et al., 2015; Serban et al., 2016b). These generative models obtain the best result on a large open-domain setting, including on the difficult task of reconstructing the order of every sentence in a paragraph, and our latent variable generative model significantly improves the coherence of text generated by the model.

Our work marks an initial step in building end-to-end systems to evaluate open-domain discourse coherence, and more importantly, generating coherent texts given discourse contexts.

[1] Adding coreference (Elsner and Charniak, 2008), named entities (Eisner and Charniak, 2011), discourse relations (Lin et al., 2011) and entity graphs (Guinaudeau and Strube, 2013).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 198–209
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

2 The Discriminative Model

The discriminative model treats cliques (sets of sentences surrounding a center sentence) taken from the original articles as coherent positive examples and cliques with random replacements of the center sentence as negative examples. The discriminative model can be viewed as an extended version of Li and Hovy's (2014) model but is practical at large scale[2]. We thus make this section succinct.

Notations Let C denote a sequence of coherent texts taken from original articles generated by humans. C is comprised of a sequence of sentences $C = \{s_{n-L}, ..., s_{n-1}, s_n, s_{n+1}, ..., s_{n+L}\}$ where L denotes the half size of the context window. Suppose each sentence s_n consists of a sequence of words $w_{n1}, ..., w_{nt}, ..., w_{nM}$, where M is the number of tokens in s_n. Each word w is associated with a K dimensional vector h_w and each sentence is associated with a K dimensional vector x_s.

Each C contains $2L + 1$ sentences, and is associated with a $(2L + 1) \times K$ dimensional vector obtained by concatenating the representations of its constituent sentences. The sentence representation is obtained from LSTMs. After word compositions, we use the representation output from the final time step to represent the entire sentence. Another neural network model with a *sigmoid* function on the very top layer is employed to map the concatenation of representations of its constituent sentences to a scalar, indicating the probability of the current clique being a coherent one or an incoherent one.

Weakness Two problems with the discriminative model stand out: First, it relies on negative sampling to generate negative examples. Since the sentence-level semantic space in the open-domain setting is huge, the sampled instances can only cover a tiny proportion of the possible negative candidates, and therefore don't cover the space of possible meanings. As we will show in the experiments section, the discriminative model performs competitively in specific domains, but not in the open domain setting. Secondly and more importantly, discriminative models are only able to tell whether an already-given chunk of text is coherent or not. While they can thus be used in tasks like extractive summarization for sentence re-ordering, they cannot be used for coherent text generation in tasks like dialogue generation or abstractive text summarization.

3 The Generative Model

We therefore introduce three neural generative models of discourse coherence.

3.1 Model 1: the SEQ2SEQ Model and its Variations

In a coherent context, a machine should be able to guess the next utterance given the preceding ones. A straightforward way to do that is to train a SEQ2SEQ model to predict a sentence given its contexts (Sutskever et al., 2014). Generating sentences based on neighboring sentences resembles skip-thought models (Kiros et al., 2015), which build an encoder-decoder model by predicting tokens in neighboring sentences.

As shown in Figure 1a, given two consecutive sentences $[s_i, s_{i+1}]$, one can measure the coherence by the likelihood of generating s_{i+1} given its preceding sentence s_i (denoted by *uni*). This likelihood is scaled by the number of words in s_{i+1} (denoted by N_{i+1}) to avoid favoring short sequences.

$$L(s_i, s_{i+1}) = \frac{1}{N_{i+1}} \log p(s_{i+1}|s_i) \qquad (1)$$

The probability can be directly computed using a pretrained SEQ2SEQ model (Sutskever et al., 2014) or an attention-based model (Bahdanau et al., 2015; Luong et al., 2015).

In a coherent context, a machine should not only be able to guess the next utterance given the preceding ones, but also the preceding one given the following ones. This gives rise to the coherence model (denoted by *bi*) that measures the bidirectional dependency between the two consecutive sentences:

$$\begin{aligned} L(s_i, s_{i+1}) = &\frac{1}{N_i} \log p_B(s_i|s_{i+1}) \\ &+ \log \frac{1}{N_{i+1}} p_F(s_{i+1}|s_i) \end{aligned} \qquad (2)$$

We separately train two models: a forward model $p_F(s_{i+1}|s_i)$ that predicts the next sentence based on the previous one and a backward model $p_B(s_i|s_{i+1})$ that predicts the previous sentence given the next sentence. $p_B(s_i|s_{i+1})$ can be trained in a way similar to $p_F(s_{i+1}|s_i)$ with sources and targets swapped. It is worth noting that p_B and p_F are separate models and do not share parameters.

[2] Li and Hovy's (2014) recursive neural model operates on parse trees, which does not support batched computation and is therefore hard to scale up.

One problem with the described *uni* and *bi* models is that sentences with higher language model probability (e.g., sentences without rare words) also tend to have higher conditional probability given their preceding or succeeding sentences. We are interested in measuring the informational gain from the contexts rather than how fluent the current sentence is. We thus propose eliminating the influence of the language model, which yields the following coherence score:

$$\begin{aligned} L(s_i, s_{i+1}) \\ &= \frac{1}{N_i}[\log p_B(s_i|s_{i+1}) - \log p_L(s_i)] \\ &+ \frac{1}{N_{i+1}}[\log p_B(s_{i+1}|s_i) - \log p_L(s_{i+1})] \end{aligned} \tag{3}$$

where $p_L(s)$ is the language model probability for generating sentence s. We train an LSTM language model, which can be thought of as a SEQ2SEQ model with an empty source. A closer look at Eq. 3 shows that it is of the same form as the mutual information between s_{i+1} and s_i, namely $\log[p(s_{i+1}, s_i)/p(s_{i+1})p(s_i)]$.

Generation The scoring functions in Eqs. 1, 2, and 3 are discriminative, generating coherence scores for an already-given chunk of text. Eqs. 2 and 3 can not be directly used for generation purposes, since they requires the completion of s_{i+1} before the score can be computed. A normal strategy is to generate a big N-best list using Eq. 1 and then rerank the N-best list using Eq. 2 or 3 (Li et al., 2015a). The N-best list can be generated using standard beam search, or other algorithmic variations that promote diversity, coherence, etc. (Shao et al., 2017).

Weakness (1) The SEQ2SEQ model generates words sequentially based on an evolving hidden vector, which is updated by combining the current word representation with previously built hidden vectors. The generation process is thus not exposed to more global features of the discourse like topics. As the hidden vector evolves, the influence from contexts gradually diminishes, with language models quickly dominating. (2) By predicting a sentence conditioning only on its left or right neighbor, the model lacks the ability to handle the longer-term discourse dependencies across the sentences of a text.

To tackle these two issues, we need a model that is able to constantly remind the decoder about the global meaning that it should convey at each word-generation step, a global meaning which can capture the state of the discourse across the sentences of a text. We propose two models of this global meaning, a pipelined approach based on HMM-based topic models (Blei et al., 2003; Gruber et al., 2007), and an end-to-end generative model with variational latent variables.

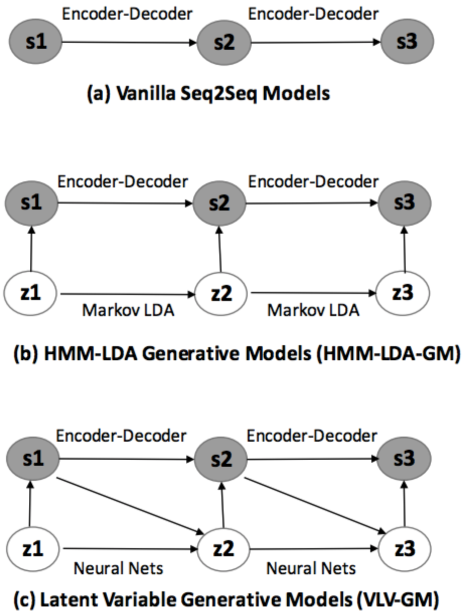


Figure 1: Overview of the proposed generative models for discourse coherence modeling.

3.2 HMM-LDA based Generative Models (HMM-LDA-GM)

In Markov topic models the topic depends on the previous topics in context (Ritter et al., 2010; Paul and Girju, 2010; Wang et al., 2011; Gruber et al., 2007; Paul, 2012). The topic for the current sentence is drawn based on the topic of the preceding sentence (or word) rather than on the global document-level topic distribution in vanilla LDA.

Our first model is a pipelined one (the *HMM-LDA-GM* in Fig. 1b), in which an HMM-LDA model provides the SEQ2SEQ model with global information for token generation, with two components:

(1) **Running HMM-LDA**: we first run a sentence-level HMM-LDA similar to Gruber et al. (2007). Our implementation forces all words in a sentence to be generated from the same topic, and this topic is sampled from a distribution based on the topic from previous sentence. Let t_n denote the distribution of topics for the current sentence, where $t_n \in \mathbb{R}^{1\times T}$. We also associate each LDA

topic with a K dimensional vector, representing the semantics embedded in this topic. The topic-representation matrix is denoted by $V \in \mathbb{R}^{T\times K}$, where T is the pre-specified number of topics in LDA. V is learned in the word predicting process when training encoder-decoder models.

(2) **Training encoder-decoder models**: For the current sentence s_n, given its topic distribution t_n, we first compute the topic representation z_n for s_n using the weighted sum of LDA topic vectors:

$$z_n = t_n \times V \tag{4}$$

z_n can be thought of as a discourse state vector that stores the information the current sentence needs to convey in the discourse, and is used to guide every step of word generation in s_n. We run the encoder-decoder model, which subsequently predicts tokens in s_n given s_{n-1}. This process is the same as the vanilla version of SEQ2SEQ models, the only difference being that z_n is incorporated into each step of decoding for hidden vector updates:

$$p(s_n|z_n, s_{n-1}) = \prod_{t=1}^{M} p(w_t|h_{t-1}, z_n) \tag{5}$$

V is updated along with parameters in the encoder-decoder model.

z_n influences each time step of decoding, and thus addresses the problem that vanilla SEQ2SEQ models gradually lose global information as the hidden representations evolve. z_n is computed based on the topic distribution t_n, which is obtained from the HMM-LDA model, thus modeling the global Markov discourse dependency between sentences of the text.[3] The model can be adapted to the bi-directional setting, in which we separately train two models to handle the forward probability $\log p(t_n|s_{n-1}, ...)$ and the backward one $\log p(t_n|s_{n+1})$. The bi-directional (*bi*) strategy described in Eq. 3 can also be incorporated to remove the influence of language models.

Weakness Topic models (either vanilla or HMM versions) focus on word co-occurrences at the document-level and are thus very lexicon-based. Furthermore, given the diversity of topics in a dataset like Wikipedia but the small number of topic clusters, the LDA model usually produces very coarse-grained topics (politics, sports, history, etc.), assigning very similar topic distributions to consecutive sentences. These topics thus capture topical coherence but are too coarse-grained to capture all the more fine-grained aspects of *discourse* coherence relationships.

3.3 Variational Latent Variable Generative Models (VLV-GM)

We therefore propose instead to train an end-to-end system, in which the meaning transitions between sentences can be naturally learned from the data. Inspired by recent work on generating sentences from a latent space (Serban et al., 2016b; Bowman et al., 2015; Chung et al., 2015), we propose the VSV-GM model in Fig. 1c. Each sentence s_n is again associated with a hidden vector representation $z_n \in \mathbb{R}^K$ which stores the global information that the current sentence needs to talk about, but instead of obtaining z_n from an upstream model like LDA, z_n is learned from the training data. z_n is a stochastic latent variable conditioned on all previous sentences and z_{n-1}:

$$\begin{aligned} &p(z_n|z_{n-1}, s_{n-1}, s_{n-2}, ...) = N(\mu^{\text{true}}_{z_n}, \Sigma^{\text{true}}_{z_n}) \\ &\mu^{\text{true}}_{z_n} = f(z_{n-1}, s_{n-1}, s_{n-2}, ...) \\ &\Sigma^{\text{true}}_{z_n} = g(z_{n-1}, s_{n-1}, s_{n-2}, ...) \end{aligned} \tag{6}$$

where $N(\mu, \Sigma)$ is a multivariate normal distribution with mean $\mu \in \mathbb{R}^K$ and covariance matrix $\Sigma \in \mathbb{R}^{K\times K}$. Σ is a diagonal matrix. As can be seen, the global information z_n for the current sentence depends on the information z_{n-1} for its previous sentence as well as the text of the context sentences. This forms a Markov chain across all sentences. f and g are neural network models that take previous sentences and z_{n-1}, and map them to a real-valued representation using hierarchical LSTMs (Li et al., 2015b)[4].

Each word w_{nt} from s_n is predicted using the concatenation of the representation previously build by the LSTMs (the same vector used in word prediction in vanilla SEQ2SEQ models) and z_n, as shown in Eq.5.

We are interested in the posterior distribution $p(z_n|s_1, s_2, ..., s_{n-1})$, namely, the information that the current sentence needs to convey given the preceding ones. Unfortunately, a highly non-linear mapping from z_n to tokens in s_n results in in-

[3] This pipelined approach is closely related to recent work that incorporates LDA topic information into generation models in an attempt to leverage context information (Ghosh et al., 2016; Xing et al., 2016; Mei et al., 2016)

[4] Sentences are first mapped to vector representations using a LSTM model. Another level of LSTM at the sentence level then composes representations of the multiple sentences to a single vector.

tractable inference of the posterior. A common solution is to use variational inference to learn another distribution, denoted by $q(z_n|s_1, s_2, ..., s_N)$, to approximate the true posterior $p(z_n|s_1, s_2, ..., s_{n-1})$. The model's latent variables are obtained by maximizing the variational lower-bound of observing the dataset:

$$\begin{aligned} & \log p(s_1, .., s_N) \leq \\ & \sum_{t=1}^{N} -D_{KL}(q(z_n|s_n, s_{n-1}, ...)||p(z_n|s_{n-1}, s_{n-2}, ...)) \\ & + E_{q(z_n|s_n, s_{n-1}, ...)} \log p(s_n|z_n, s_{n-1}, s_{n-2}, ...) \end{aligned} \tag{7}$$

This objective to optimize consists of two parts; the first is the KL divergence between the approximate distribution q and the true posterior $p(s_n|z_n, s_{n-1}, s_{n-2}, ...)$, in which we want to approximate the true posterior using q. The second part $E_{q(z_n|s_n, s_{n-1}, ...)} \log p(s_n|z_n, s_{n-1}, s_{n-2}, ...)$, predicts tokens in s_n in the same way as in SEQ2SEQ models with the difference that it considers the global information z_n.

The approximate posterior distribution $q(z_n|s_n, s_{n-1}, ...)$ takes a form similar to $p(z_n|s_{n-1}, s_{n-2}, ...)$:

$$\begin{aligned} & q(z_n|s_n, s_{n-1}, ...) = N(\mu_{z_n}^{\text{approx}}, \Sigma_{z_n}^{\text{approx}}) \\ & \mu_{z_n}^{\text{approx}} = f_q(z_{n-1}, s_n, s_{n-1}, ...) \\ & \Sigma_{z_n}^{\text{approx}} = g_q(z_{n-1}, s_n, s_{n-1}, ...) \end{aligned} \tag{8}$$

f_q and g_q are of similar structures to f and g, using a hierarchical neural network model to map context tokens to vector representations.

Learning and Testing At training time, the approximate posterior $q(z_n|z_{n-1}, s_n, s_{n-1}, ...)$, the true distribution $p(z_n|z_{n-1}, s_{n-1}, s_{n-2}, ...)$, and the generative probability $p(s_n|z_n, s_{n-1}, s_{n-2}, ...)$ are trained jointly by maximizing the variational lower bound with respect to their parameters: a sample z_n is first drawn from the posterior distribution q, namely $N(\mu_{z_n}^{\text{approx}}, \Sigma_{z_n}^{\text{approx}})$. This sample is used to approximate the expectation $E_q \log p(s_n|z_n, s_{n-1}, s_{n-2}, ...)$. Using z_n, we can update the encoder-decoder model using SGD in a way similar to the standard SEQ2SEQ model, the only difference being that the current token to predict not only depends on the LSTM output h_t, but also z_n. Given the sampled z_n, the KL-divergence can be readily computed, and we update the model using standard gradient decent (details shown in the Appendix).

The proposed *VLV-GM* model can be adapted to the bi-directional setting and the *bi* setting similarly to the way LDA-based models are adapted.

The proposed model is closely related to many recent attempts in training variational autoencoders (VAE) (Kingma and Welling, 2013; Rezende et al., 2014), variational or latent-variable recurrent nets (Bowman et al., 2015; Chung et al., 2015; Ji et al., 2016; Bayer and Osendorfer, 2014), hierarchical latent variable encoder-decoder models (Serban et al., 2016b,a).

4 Experimental Results

In this section, we describe experimental results. We first evaluate the proposed models on discriminative tasks such as sentence-pair ordering and full paragraph ordering reconstruction. Then we look at the task of coherent text generation.

Model	Acci	Earthq	Aver
Discriminative Model	**0.930**	**0.992**	**0.956**
SEQ2SEQ (bi)	0.755	0.930	0.842
VLV-GM (bi)	0.770	0.931	0.851
Recursive	0.864	0.976	0.920
Entity Grid Model	0.904	0.872	0.888
HMM	0.822	0.938	0.880
HMM+Entity	0.842	0.911	0.876
HMM+Content	0.742	0.953	0.847
Graph	0.846	0.635	0.740
Foltz et al. (1998)-Glove	0.705	0.682	0.688
Foltz et al. (1998)-LDA	0.660	0.667	0.664

Table 1: Results from different coherence models. Results for the Recursive model is reprinted from Li and Hovy (2014), Entity Grid Model from Louis and Nenkova (2012), HMM, HMM+Entity and HMM+Content from Louis and Nenkova (2012), Graph from Guinaudeau and Strube (2013), and the final two lexical models are recomputed using Glove and LDA to replace the original LSA model of Foltz et al. (1998).

4.1 Sentence Ordering, Domain-specific Data

Dataset We first evaluate the proposed algorithms on the task of predicting the correct ordering of pairs of sentences predicated on the assumption that an article is always more coherent than a random permutation of its sentences (Barzilay and Lapata, 2008). A detailed description of this commonly used dataset and training/testing are found in the Appendix.

We report the performance of the following baselines widely used in the coherence literature.

(1) *Entity Grid Model*: The grid model presented in Barzilay and Lapata (2008). Results are directly taken from Barzilay and Lapata's (2008) paper. We also consider variations of entity grid models, such as Louis and Nenkova (2012) which models the

cluster transition probability and the *Graph Based Approach* which uses a graph to represent the entity transitions needed for local coherence computation (Guinaudeau and Strube, 2013).

(2) Li and Hovy (2014): A recursive neural model computes sentence representations based on parse trees. Negative sampling is used to construct negative incoherent examples. Results are from their papers.

(3) Foltz et al. (1998) computes the semantic relatedness of two text units as the cosine similarity between their LSA vectors. The coherence of a discourse is the average of the cosine of adjacent sentences. We used this intuition, but with more modern embedding models: (1) 300-dimensional Glove word vectors (Pennington et al., 2014), embeddings for a sentence computed by averaging the embeddings of its words (2) Sentence representations obtained from LDA (Blei et al., 2003) with 300 topics, trained on the Wikipedia dataset. Results are reported in Table 2. The extended version of the discriminative model described in this work significantly outperforms the parse-tree based recursive models presented in Li and Hovy (2014) as well as all non-neural baselines. It achieves almost perfect accuracy on the earthquake dataset and 93% on the accident dataset, marking a significant advancement in the benchmark. Generative models (both vanilla SEQ2SEQ and the proposed variational model) do not perform competitively on this dataset. We conjecture that this is due to the small size of the dataset, leading the generative model to overfit.

4.2 Evaluating Ordering on Open-domain

Since the dataset presented in Barzilay and Lapata (2008) is quite domain-specific, we propose testing coherence with a much larger, open-domain dataset: Wikipedia. We created a test set by randomly selecting 984 paragraphs from Wikipedia dump 2014, each paragraph consisting of at least 16 sentences. The training set is 30 million sentences not overlapping with the test set.

4.2.1 Binary Permutation Classification

We adopt the same strategy as in Barzilay and Lapata (2008), in which we generate pairs of sentence permutations from the original Wikipedia paragraphs. We follow the protocols described in the subsection and each pair whose original paragraph's score is higher than its permutation is treated as being correctly classified, else incorrectly classified. Models are evaluated using accuracy. We implement the Entity Grid Model (Barzilay and Lapata, 2008) using the Wikipedia training set as a baseline, the detail of which is presented in the Appendix. Other baselines consist of the Glove and LDA updates of the lexical coherence baselines (Foltz et al., 1998).

Model	Accuracy
VLV-GM (MMI)	**0.873**
VLV-GM (bi)	0.860
VLV-GM (uni)	0.839
LDA-HMM-GM (MMI)	0.847
LDA-HMM-GM (bi)	0.837
LDA-HMM-GM (uni)	0.814
SEQ2SEQ (MMI)	0.840
SEQ2SEQ (bi)	0.821
SEQ2SEQ (uni)	0.803
Discriminative Model	0.715
Entity Grid Model	0.686
Foltz et al. (1998)-Glove	0.597
Foltz et al. (1998)-LDA	0.575

Table 2: Performance on the open-domain binary classification dataset of 984 Wikipedia paragraphs.

Results Table 2 presents results on the binary classification task. Contrary to the findings on the domain specific dataset in the previous subsection, the discriminative model does not yield compelling results, performing only slightly better than the entity grid model. We believe the poor performance is due to the sentence-level negative sampling used by the discriminative model. Due to the huge semantic space in the open-domain setting, the sampled instances can only cover a tiny proportion of the possible negative candidates, and therefore don't cover the space of possible meanings. By contrast the dataset in Barzilay and Lapata (2008) is very domain-specific, and the semantic space is thus relatively small. By treating all other sentences in the document as negative, the discriminative strategy's negative samples form a much larger proportion of the semantic space, leading to good performance.

Generative models perform significantly better than all other baselines. Compared with the dataset in Barzilay and Lapata (2008), overfitting is not an issue here due to the great amount of training data. In line with our expectation, the *MMI* model outperforms the *bidirectional* model, which in turn outperforms the *unidirectional* model across all three generative model settings. We thus only report *MMI* results for experiments below. The *VLV-GM* model outperforms that the *LDA-HMM-GM* model, which is slightly better than the vanila SEQ2SEQ models.

Model	Accuracy
VLV-GM (MMI)	**0.256**
LDA-HMM-GM (MMI)	0.237
SEQ2SEQ (MMI)	0.226
Entity Grid Model	0.143
Foltz et al. (1998) (Glove)	0.084

Table 3: Performances of the proposed models on the open-domain paragraph reconstruction dataset.

Model	adver-1	adver-2	adver-3
VLV-GM (MMI)	**0.174**	**0.120**	**0.054**
LDA-HMM-GM (MMI)	0.130	0.104	0.043
SEQ2SEQ (MMI)	0.120	0.090	0.039
SEQ2SEQ (bi)	0.108	0.078	0.030
SEQ2SEQ (uni)	0.101	0.068	0.024

Table 4: Adversarial Success for different models.

4.2.2 Paragraph Reconstruction

The accuracy of our models on the binary task of detecting the original sentence ordering is very high, on both the prior small task and our large open-domain version. We therefore believe it is time for the community to move to a more difficult task for measuring coherence.

We suggest the task of *reconstructing an original paragraph from a bag of constituent sentences*, which has been previously used in coherence evaluation (Lapata, 2003). More formally, given a set of permuted sentences $s_1, s_2, ..., s_N$ (N the number of sentences in the original document), our goal is return the original (presumably most coherent) ordering of s.

Because the *discriminative model* calculates the coherence of a sentence given the known previous and following sentences, it cannot be applied to this task since we don't know the surrounding context. Hence, we only use the *generative model*. The first sentence of a paragraph is given: for each step, we compute the coherence score of placing each remaining candidate sentence to the right of the partially constructed document. We use beam search with beam size 10. We use the Entity Grid model as a baseline for both the settings.

Evaluating the absolute positions of sentences would be too harsh, penalizing orderings that maintain relative position between sentences through which local coherence can be manifested. We therefore use Kendall's τ (Lapata, 2003, 2006), a metric of rank correlation for evaluation. See the Appendix for details of Kendall's τ computation. We observe a pattern similar to the results on the binary classification task, where the *VLV-GM* model performs the best.

4.3 Adversarial evaluation on Text Generation Quality

Both the tasks above are discriminative ones. We also want to evaluate different models' ability to generate coherent text chunks. The experiment is set up as follow: each encoder-decoder model is first given a set of context sentences (3 sentences). The model then generates a succeeding sentence using beam-search given the contexts. For the *unidirectional* setting, we directly take the most probable sequence and for the *bi-directional* and *MMI*, we rerank the N-best list using the backward probability and language model probability.

We conduct experiments on multi-sentence generation, in which we repeat the generative process described above for N times, where N=1,2,3. At the end of each turn, the context is updated by adding in the newly generated sequence, and this sequence is used as the source input to the encoder-decoder model for next sequence generation. For example, when N is set to 2, given the three context sentences *context-a*, *context-b* and *context-c*, we first generate *sen-d* given the three context sentences and then generate *sen-e* given the sen-d, *context-a*, *context-b* and *context-c*.

For evaluation, standard word overlap metrics such as BLEU or ROUGE are not suited for our task, and we use adversarial evaluation Bowman et al. (2015); Anjuli and Vinyals (2016). In adversarial evaluation, we train a binary discriminant function to classify a sequence as machine generated or human generated, in an attempt to evaluate the model's sentence generation capability. The evaluator takes as input the concatenation of the contexts and the generated sentences (i.e., *context-a*, *context-b* and *context-c*, *sen-d* , *sen-e* in the example described above),[5] and outputs a scalar, indicating the probability of the current text chunk being human-generated. Training/dev/test sets are held-out sets from the one on which generative models are trained. They respectively contain 128,000/12,800/12,800 instances. Since discriminative models cannot generate sentences, and thus cannot be used for adversarial evaluation, they are skipped in this section.

We report Adversarial Success (*AdverSuc* for short), which is the fraction of instances in which a model is capable of fooling the evaluator. *Adver-*

[5] The model uses a hierarchical neural structure that first maps each sentence to a vector representation, with another level of LSTM on top of the constituent sentences, producing a single vector to represent the entire chunk of texts.

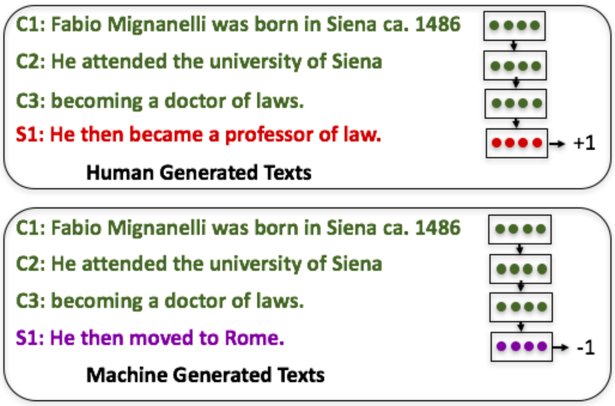

C1: Fabio Mignanelli was born in Siena ca. 1486
C2: He attended the university of Siena
C3: becoming a doctor of laws.
S1: He then became a professor of law.
+1
Human Generated Texts
C1: Fabio Mignanelli was born in Siena ca. 1486
C2: He attended the university of Siena
C3: becoming a doctor of laws.
S1: He then moved to Rome.
-1
Machine Generated Texts

Figure 2: An overview of training the adversarial evaluator using a hierarchical neural model. Green denotes input contexts. Red denotes a sentence from human-generated texts, treated as a positive example. Purple denotes a sentence from machine-decoded texts, treated as a negative example.

Suc is the difference between 1 and the accuracy achieved by the evaluator. Higher values of *AdverSuc* for a dialogue generation model are better. *AdverSuc-N* denotes the adversarial accuracy value on machine-generated texts with N turns.

Table 4 show *AdverSuc* numbers for different models. As can be seen, the latent variable model VLV-GM is able to generate chunk of texts that are most indistinguishable from coherent texts from humans. This is due to its ability to handle the dependency between neighboring sentences. Performance declines as the number of turns increases due to the accumulation of errors and current models' inability to model long-term sentence-level dependency. All models perform poorly on the *adver-3* evaluation metric, with the best adversarial success value being 0.081 (the trained evaluator is able to distinguish between human-generated and machine generated dialogues with greater than 90 percent accuracy for all models).

4.4 Qualitative Analysis

With the aim of guiding future investigations, we also briefly explore our model qualitatively, examining the coherence scores assigned to some artificial miniature discourses that exhibit various kinds of coherence.

Case 1: Lexical Coherence

Pinochet was arrested. His arrest was unexpected. **1.79**
Pinochet was arrested. His death was unexpected. **0.84**
Mary ate some apples. She likes apples. **2.03**
Mary ate some apples. She likes pears. **0.27**
Mary ate some apples. She likes Paris. **-1.35**

The examples suggest that the model handles lexical coherence, correctly favoring the 1st over the 2nd, and the 3rd over the 4th examples. Note that the coherence score for the final example is negative, which means conditioning on the first sentence actually decreases the likelihood of generating the second one.

Case 2: Temporal Order

Washington was unanimously elected president in the first two national elections. He oversaw the creation of a strong, well-financed national government. **1.48**
Washington oversaw the creation of a strong, well-financed national government. He was unanimously elected president in the first two national elections. **0.72**

Case 3: Causal Relationship

Bret enjoys video games; therefore, he sometimes is late to appointments. **0.69**
Bret sometimes is late to appointments; therefore, he enjoys video games. **-0.07**

Cases 2 and 3 suggest the model may, at least in these simple cases, be capable of addressing the much more complex task of dealing with temporal and causal relationships. Presumably this is because the model is exposed in training to the general preference of natural text for temporal order, and even for the more subtle causal links.

Case 4: Centering/Referential Coherence

Mary ate some apples. She likes apples. **3.06**
She ate some apples. Mary likes apples. **2.41**

The model seems to deal with simple cases of referential coherence.

Example3: **2.40**
John went to his favorite music store to buy a piano.
He had frequented the store for many years.
He was excited that he could finally buy a piano.
He arrived just as the store was closing for the day.
Example4: **1.62**
John went to his favorite music store to buy a piano.
It was a store John had frequented for many years
He was excited that he could finally buy a piano..
It was closing just as John arrived.

In these final examples from Miltsakaki and Kukich (2004), the model successfully captures the fact that the second text is less coherent due to *rough shifts*. This suggests that the discourse embedding space may be able to capture a representation of entity focus.

Of course all of these these qualitative evaluations are only suggestive, and a deeper understanding of what the discourse embedding space is capturing will likely require more sophisticated visualizations.

5 Conclusion

We investigate the problem of open-domain discourse coherence, training discriminative models that treating natural texts as coherent and permutations as non-coherent, and Markov generative models that can predict sentences given their neighbors.

Our work shows that the traditional evaluation metric (ordering pairs of sentences in small domains) is completely solvable by our discriminative models, and we therefore suggest the community move to the harder task of open-domain full-paragraph sentence ordering.

The proposed models also offer an initial step in generating coherent texts given contexts, which has the potential to benefit a wide range of generation tasks in NLP. Our latent variable neural models, by offering a new way to learn latent discourse-level features of a text, also suggest new directions in discourse representation that may bring benefits to any discourse-aware NLP task.

Acknowledgements The authors thank Will Monroe, Sida Wang, Kelvin Guu and the other members of the Stanford NLP Group for helpful discussions and comments. Jiwei Li is supported by a Facebook Fellowship, which we gratefully acknowledge. This work is also partially supported by the NSF under award IIS-1514268, and the DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF- 15-1-0462. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, the NSF, or Facebook.

References

Ernst Althaus, Nikiforos Karamanis, and Alexander Koller. 2004. Computing locally coherent discourses. In *Proceedings of ACL 2004*.

Kannan Anjuli and Oriol Vinyals. 2016. Adversarial evaluation of dialogue models. *NIPS 2016 Workshop on Adversarial Training* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of the International Conference on Learning Representations (ICLR)*.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics* 34(1):1–34.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL*. pages 113–120.

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics* 31(3):297–328.

Justin Bayer and Christian Osendorfer. 2014. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610* .

Gillinder Bedi, Facundo Carrillo, Guillermo A Cecchi, Diego Fernández Slezak, Mariano Sigman, Natália B Mota, Sidarta Ribeiro, Daniel C Javitt, Mauro Copelli, and Cheryl M Corcoran. 2015. Automated analysis of free speech predicts psychosis onset in high-risk youths. *npj Schizophrenia* 1.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* .

Jill Burstein, Joel Tetreault, and Slava Andreyev. 2010. Using entity-based features to model coherence in student essays. In *Human language technologies: The 2010 annual conference of the North American chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 681–684.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*. pages 2980–2988.

Carl Doersch. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* .

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research* 12:2121–2159.

Micha Eisner and Eugene Charniak. 2011. Extending the entity grid with entity-specific features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, pages 125–129.

Micha Elsner, Joseph L Austerweil, and Eugene Charniak. 2007. A unified local and global model for discourse coherence. In *HLT-NAACL*. pages 436–443.

Micha Elsner and Eugene Charniak. 2008. Coreference-inspired coherence modeling. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human*

Language Technologies: Short Papers. Association for Computational Linguistics, pages 41–44.

Brita Elvevåg, Peter W Foltz, Daniel R Weinberger, and Terry E Goldberg. 2007. Quantifying incoherence in speech: An automated methodology and novel application to schizophrenia. *Schizophrenia research* 93(1):304–316.

Vanessa Wei Feng and Graeme Hirst. 2012. Extending the entity-based coherence model with multiple ranks. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 315–324.

Peter W Foltz. 2007. Discourse coherence and lsa. *Handbook of latent semantic analysis* pages 167–184.

Peter W Foltz, Walter Kintsch, and Thomas K Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse processes* 25(2-3):285–307.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual lstm (clstm) models for large scale nlp tasks. *arXiv preprint arXiv:1602.06291* .

Amit Gruber, Yair Weiss, and Michal Rosen-Zvi. 2007. Hidden topic markov models. In *AISTATS*. volume 2, pages 163–170.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. pages 93–103.

M. A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman.

Eduard H Hovy. 1988. Planning coherent multisentential text. In *Proceedings of the 26th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 163–169.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse relation language models. *arXiv preprint arXiv:1603.01913* .

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* .

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*. pages 3276–3284.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 545–552.

Mirella Lapata. 2006. Automatic evaluation of information ordering: Kendall's tau. *Computational Linguistics* 32(4):471–484.

Alex Lascarides and Nicholas Asher. 1991. Discourse relations and defeasible knowledge. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 55–62.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015a. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* .

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015b. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057* .

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 997–1006.

Annie Louis and Ani Nenkova. 2012. A coherence model based on syntactic patterns. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 1157–1168.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *EMNLP* .

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text* 8(3):243–281.

Daniel Marcu. 1997. From local to global coherence: A bottom-up approach to text planning. In *AAAI/IAAI*. Citeseer, pages 629–635.

Danielle S. McNamara, Max M. Louwerse, Philip M. McCarthy, and Arthur C. Graesser. 2010. Coh-metrix: Capturing linguistic features of cohesion. *Discourse Processes* 47(4):292–330.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. Coherent dialogue with attention-based language models. *arXiv preprint arXiv:1611.06997* .

Eleni Miltsakaki and Karen Kukich. 2004. Evaluation of text coherence for electronic essay scoring systems. *Natural Language Engineering* 10(01):25–55.

J. Morris and G. Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics* 17(1):21–48.

Michael Paul and Roxana Girju. 2010. A two-dimensional topic-aspect model for discovering multi-faceted topics. *Urbana* 51(61801):36.

Michael J Paul. 2012. Mixed membership markov models for unsupervised conversation modeling. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 94–104.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082* .

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 172–180.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron Courville. 2016a. Multiresolution recurrent neural networks: An application to dialogue response generation. *arXiv preprint arXiv:1606.00776* .

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016b. A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv preprint arXiv:1605.06069* .

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating long and diverse responses with neural conversation models. *arXiv preprint arXiv:1701.03185* .

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

Stephen Tratz and Eduard Hovy. 2011. A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1257–1268.

Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2007. Evaluating discourse-based answer extraction for why-question answering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 735–736.

Hongning Wang, Duo Zhang, and ChengXiang Zhai. 2011. Structural topic model for latent topical structure analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 1526–1535.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2016. Topic augmented neural response generation with a joint attention mechanism. *arXiv preprint arXiv:1606.08340* .

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

6 Supplemental Material

Details for the domain specific dataset (Barzilay and Lapata, 2008) The corpus consists of 200 articles each from two domains: NTSB airplane accident reports (V=4758, 10.6 sentences/document) and AP earthquake reports (V=3287, 11.5 sentences/document), split into training and testing. For each document, pairs of permutations are generated[6]. Each pair contains the original document order and a random permutation of the sentences from the same document.

Training/Testing details for models on the domain specific dataset We use reduced versions of both generative and discriminative models to allow fair comparison with baselines. For the discriminative model, we generate noise negative examples from random replacements in the training set, with the only difference that random replacements only come from the same document. We use 300 dimensional embeddings borrowed from GLOVE (Pennington et al., 2014) to initialize word embeddings. Word embeddings are kept fixed during training and we update LSTM parameters using AdaGrad (Duchi et al., 2011). For the generative model, due to the small size of the dataset, we train a one layer SEQ2SEQ model with word dimensionality and number of hidden neurons set to 100. The model is trained using SGD with AdaGrad (Zeiler, 2012).

The task requires a coherence score for the whole document, which is comprised of multiple cliques. We adopt the strategy described in Li and Hovy (2014) by breaking the document into a series of cliques which is comprised of a sequence of

[6] Permutations downloaded from `people.csail.mit.edu/regina/coherence/CLsubmission/`.

consecutive sentences. The document-level coherence score is attained by averaging its constituent cliques. We say a document is more coherent if it achieves a higher average score within its constituent cliques.

Implementation of Entity Grid Model For each noun in a sentence, we extract its syntactic role (subject, object or other). We use a wikipedia dump parsed using the Fanse Parser (Tratz and Hovy, 2011). Subjects and objects are extracted based on *nsubj* and *dobj* relations in the dependency trees. (Barzilay and Lapata, 2008) define two versions of the Entity Grid Model, one using full coreference and a simpler method using only exact-string coreference; Due to the difficulty of running full coreference resolution tens of millions of Wikipedia sentences, we follow other researchers in using Barzilay and Lapata's simpler method (Feng and Hirst, 2012; Burstein et al., 2010; Barzilay and Lapata, 2008).[7]

Kendall's τ Kendall's τ is computed based on the number of inversions in the rankings as follows:

$$\tau = 1 - \frac{2\text{\# of inversions}}{N \times (N-1)} \tag{9}$$

where N denotes the number of sentences in the original document and inversions denote the number of interchanges of consecutive elements needed to reconstruct the original document. Kendall's τ can be efficiently computed by counting the number of intersections of lines when aligning the original document and the generated document. We refer the readers to Lapata (2003) for more details.

Derivation for Variation Inference For simplicity, we use μ_{post} and Σ_{approx} to denote $\mu^{\text{approx}}(z_n)$ and $\Sigma^{\text{approx}}(z_n)$, μ_{true} and Σ_{true} to denote $\mu^{\text{true}}(z_n)$ and $\Sigma^{\text{true}}(z_n)$. The KL-divergence between the approximate distribution $q(z_n|z_{n-1}, s_n, s_{n-1}, ...)$ and the true distribution $p(z_n|z_{n-1}, s_{n-1}, s_{n-2}, ...)$ in the variational inference is given by:

$$\begin{aligned} & D_{KL}(q(z_n|z_{n-1}, s_n, s_{n-1}, ...)||p(z_n|z_{n-1}, s_{n-1}, s_{n-2}, ...) \\ =& \frac{1}{2}(\text{tr}(\Sigma_{true}^{-1}\Sigma_{approx}) - k + \log\frac{\det\Sigma_{true}}{\det\Sigma_{approx}} \\ +& (\mu_{true} - \mu_{approx})^{-1}\Sigma_{true}^{-1}(\mu_{true} - \mu_{approx})) \end{aligned} \tag{10}$$

where k denotes the dimensionality of the vector. Since z_n has already been sampled and thus known, μ_{approx}, Σ_{approx}, μ_{true}, Σ_{true} and consequently Eq10 can be readily computed. The gradient with respect to μ_{approx}, Σ_{approx}, μ_{true}, Σ_{true} can be respectively computed, and the error is then backpropagated to the hierarchical neural models that are used to compute them. We refer the readers to Doersch (2016) for more details about how a general VAE model can be trained.

Our generate models offer a powerful way to represent the latent discourse structure in a complex embedding space, but one that is hard to visualize. To help understand what the model is doing, we examine some relevant examples, annotated with the (log-likelihood) coherence score from the *MMI* generative model, with the goal of seeing (qualitatively) the kinds of coherence the model seems to be representing. (The MMI can be viewed as the informational gain from conditioning the generation of the current sentence on its neighbors.)

[7] Our implementation of the Entity Grid Model is built upon public available code at https://github.com/karins/CoherenceFramework.

Affinity-Preserving Random Walk for Multi-Document Summarization

Kexiang Wang, Tianyu Liu, Zhifang Sui and **Baobao Chang**
Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
Collaborative Innovation Center for Language Ability, Xuzhou 221009 China
{wkx,tianyu0421,szf,chbb}@pku.edu.cn

Abstract

Multi-document summarization provides users with a short text that summarizes the information in a set of related documents. This paper introduces affinity-preserving random walk to the summarization task, which preserves the affinity relations of sentences by an absorbing random walk model. Meanwhile, we put forward adjustable affinity-preserving random walk to enforce the diversity constraint of summarization in the random walk process. The ROUGE evaluations on DUC 2003 topic-focused summarization task and DUC 2004 generic summarization task show the good performance of our method, which has the best ROUGE-2 recall among the graph-based ranking methods.

1 Introduction

Multi-document summarization provides users with summary that reflects the main information in a set of given documents. The documents are often related and talk about more than one topics. Generic multi-document summarization and topic-focused multi-document summarization are two typical kinds of summarization. The former is a summarization delivering the main information of the documents with no bias while the latter is a one delivering the main information biased to a given topic description (a few sentences or even phrases). Most existing summarization systems are designed for these two kinds of summarization.

There are two goals for generic multi-document summarization. The first one is **saliency**. Summary sentences should be central sentences that capture the majority of information in a document cluster. The sentences with little information about the document cluster should not be included in the summary. The second one is **diversity**. The information overlap between summary sentences should be as minimal as possible due to the length limit of summary. In other words, the information coverage of summary is a determinant, which requires that the summary sentences should cover diverse aspects of information. Besides the two goals, there is another goal for the topic-focused summarization and that is **relevancy**. It requires that the summary sentences be relevant to the topic description. A series of conferences and workshops on automatic text summarization (e.g. NTCIR, DUC), special topic sessions in ACL, EMNLP and SIGIR have advanced the techniques to achieve these goals and many approaches have been proposed so far.

In this paper, we focus on the extractive summarization methods, which extract the summary sentences from the input document cluster. We propose affinity-preserving random walk for multi-document summarization. The method is a graph-based ranking method, which takes into account the global information collectively computed from the entire sentence affinity graph. Different from the previous graph-based ranking methods, our method adopts "global normalization" to transform sentence affinity matrix into sentence transition matrix and formulates the sentence ranking process in an absorbing random walk model. Meanwhile, the adjustable affinity-preserving random walk is proposed to facilitate the diversity of summary by adjusting the sentence transition matrix after each iteration of random walk. Experimental results on DUC generic and topic-focused multi-document summarization tasks show the competitive performance of our method. To our best knowledge, our system has the best ROUGE-2 recall on both tasks among all existing graph-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 210–220
Copenhagen, Denmark, September 7–11, 2017.

based ranking methods, which defeats most other methods.

We summarize our contributions as follows. (1) We preserve the original affinity relations between sentences in a novel affinity-preserving random walk view for multi-document summarization. The preservation of affinity leads to a more salient summary. And it is applicable to both generic and topic-focused summarization. (2) We propose adjustable affinity-preserving random walk to enforce the diversity constraint of summarization in the random walk process. (3) Experiments on DUC 2003 and DUC 2004 tasks demonstrate the competitive performance of our method.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 describes traditional random walk model for summarization. Section 4 proposes affinity-preserving random walk for the saliency goal of summarization and this section also proposes adjustable affinity-preserving random walk to produce both salient and diverse summary. Section 5 gives our evaluation results and the conclusion is made in Section 6.

2 Related Work

Our method belongs to the graph-based ranking methods to select sentences in the documents to produce the summary. Erkan and Radev (2004) proposed LexPageRank to compute the sentence saliency based on the concept of eigenvector centrality. It constructs the sentence affinity graph and computes the sentence saliency based on an algorithm similar to PageRank (Page et al., 1999). Like PageRank, the affinity matrix is converted into the row-stochastic matrix, which is used as the transition matrix of random walk on the weighted graph. Wan (2007) proposed manifold ranking for topic-focused multi-document summarization. It makes full use of both the relationships among all sentences in the documents and the relationships between the given topic description and the sentences. Manifold ranking conducts a different normalization on the sentence affinity matrix to guarantee the algorithm's convergence. GRASSHOPPER (Zhu et al., 2007) ranks sentences with an emphasis on the diversity constraint of summarization. It turns already ranked sentences into absorbing states, which effectively prevents redundant sentences from receiving a high rank. The algorithm is based on an absorbing random walk and produces only one summary sentence after one particular random walk becomes stationary. And the normalization from sentence affinity matrix to sentence transition matrix is the same as PageRank. DivRank (Mei et al., 2010) is a method to balance the saliency and diversity of the top ranked sentences in a reinforced random walk model. Also, the normalization in DivRank from affinity matrix to transition matrix is the same as PageRank. Another notable diversified graph-based ranking method GCD (Dubey et al., 2011) relies on large amounts of training data to learn edge conductances.

Our method formulates the multi-document summarization as an affinity-preserving random walk and uses the "global normalization" to transform sentence affinity matrix into sentence transition matrix, which is different from all those proposed methods. And the adjustable transition matrix in our method balances the saliency and diversity goals of summarization. Like GRASSHOPPER, our method relies on the absorbing random walk model. The difference is that our method does not turn the sentence vertex into absorbing state but add an absorbing vertex to the original sentence affinity graph. And all summary sentences are extracted after the random walk reaches a stationary state in our method. Like DivRank, the sentence transition matrix is adjustable in our method to enforce the diversity constraint of summarization. However, our method differs from DivRank in the mechanism to adjust the transition matrix.

3 Traditional Random Walk for Summarization

Suppose $G = (S, E)$ is a graph with vertex set S and edge set $E \subset S^2$. Suppose there is a conductance $c(s_i, s_j) > 0$ associated with each edge $(s_i, s_j) \in E$ and $c(s_i, s_j) = 0$ associated with the set $S^2 - E$ (the conductance of nonexistent edge is zero). Let

$$C(s_i) = \sum_{s_j \in S} c(s_i, s_j), \ s_i \in S \tag{1}$$

so that $C(s_i)$ is the total conductance of the edges coming from s_i. And the traditional random walk on graph is defined as

Definition 3.1. *The discrete-time Markov chain $X = (X_0, X_1, X_2, ...)$ with state space and tran-*

sition probability matrix **P** *given by*

$$P(s_i, s_j) = \frac{c(s_i, s_j)}{C(s_i)},\ (s_i, s_j) \in S^2 \qquad (2)$$

is called a random walk on the graph G.

This chain governs a particle moving along the vertices of G. If the particle in the state X_m is at vertex $s_i \in S$, it will be at a neighbor of s_i in the next state X_{m+1}, which is chosen randomly in proportion to the conductance. We can prove that $\sum_{s_j \in S} P(s_i, s_j) = 1$ for any $s_i \in S$ ($C(s_i) \neq 0$) so $\mathbf{P}$ is a row-stochastic matrix by $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$, where $\mathbf{D}$ is a diagonal matrix with entries $\mathbf{D}_{ii} = C(s_i)$ and $\mathbf{W}$ is the adjacency matrix of G where $\mathbf{W}_{ij} = c(s_i, s_j)$.

For the summarization task, G is the sentence affinity graph. The vertex set $S = \{s_1, s_2, ..., s_n\}$ contains every sentence in the document cluster and the edge set E contains the pairwise affinity between sentences. We use the $tf{*}isf$ formula to calculate the weight associated with each term occurring in the sentence, where tf is the term frequency in the sentence and isf is the inverse sentence frequency of the term among all sentences. isf is often calculated as $1{+}log(n/n_t)$, where n is the total number of sentences and n_t is the number of sentences containing the term t. $\mathbf{W}_{ij}$ is computed using the standard cosine measure (Baeza-Yates et al., 1999).

$$\mathbf{W}_{ij} = sim_{cosine}(\mathbf{v}_i, \mathbf{v}_j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\|_2 \times \|\mathbf{v}_j\|_2} \qquad (3)$$

where $\mathbf{v}_i$ and $\mathbf{v}_j$ are the corresponding term vectors of s_i and s_j. Two vertices are connected if their affinity is larger than 0 and $\mathbf{W}_{ii}$ is set as 0 to avoid self transition. We get an undirected graph G as well as a symmetric sentence affinity matrix $\mathbf{W}$ in this way. Then we transform $\mathbf{W}$ into $\mathbf{P}$ by $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ and use the stationary distribution of random walk as sentence ranking scores. The traditional random walk model is a simple practice of PageRank algorithm for multi-document summarization.

4 Affinity-Preserving Random Walk for Summarization

4.1 Prior of Multi-Document Summarization

In the above traditional random walk on graph, the normalization from affinity matrix $\mathbf{W}$ to transition matrix $\mathbf{P}$ is to make $\mathbf{P}$ a row-stochastic matrix. This can be interpreted as a democratic normalization because the surfer of a traditional random walk visits neighbors of a vertex with probability 1. The surfer has to choose a neighbor to visit next although it is a random choice w.r.t. the conductance distribution of the vertex. However this democratic normalization is not suitable for multi-document summarization due to the fact that most sentences are not salient and should not be normalized democratically as the few salient ones. The prior here is that the number ratio of good candidate sentences over bad candidate sentences is very low due to the summary length limit. Good candidate sentences are the sentences highly overlapping with sentences in the reference summary written by humans. And the remaining sentences are bad candidate sentences. The democratic normalization of $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$ will extend the adverse effect of bad candidate sentences and suppress the effect of good candidate sentences, because the total conductance of the bad candidate sentence is usually smaller than that of the good candidate sentence. In this case, the random surfer has to choose a neighbor to visit even when she is currently at a bad candidate sentence, which will direct her to visit other bad candidate sentences neighboring to the current sentence. The invariant behavior of the surfer at all vertices in the graph is not consistent with the prior which makes a distinction between good and bad candidate sentences. It may pervert the random walk process to get an ideal distribution in which only few sentences are assigned with a high ranking score.

It is worth noting here that manifold ranking (Wan et al., 2007) for summarization uses a different normalization: $\mathbf{P} = \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$. It is a symmetric normalization on both endpoints of an edge, which makes $\mathbf{P}$ a suitable choice in the manifold ranking process to smooth the scores of neighboring vertices. The symmetric normalization can be deduced from the objective function of manifold ranking (Zhou et al., 2003) and does not make a distinction between the good and bad candidate sentences. It is also not consistent with the prior. We can conclude that existing graph-based ranking methods can not well characterize the prior of multi-document summarization.

4.2 Affinity-Preserving Random Walk

We need a new normalization method that distinguishes between good and bad candidate sen-

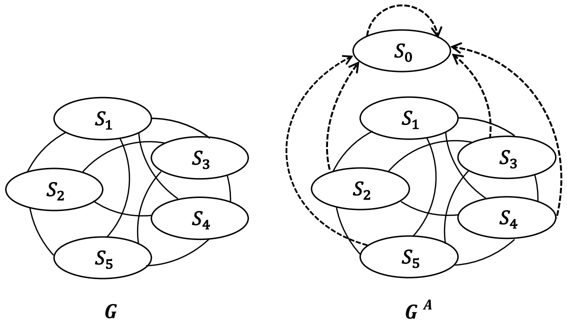


Figure 4.1: Sentence graphs for summarization. G: sentence affinity graph constructed from the document cluster. G^A: sentence augmented graph with an absorbing vertex s_0. C_{max} equals to $C(s_1)$ indicating that sentence s_1 has the maximum conductance, so there is no edge (s_1, s_0).

tences to satisfy the prior of multi-document summarization. Affinity-preserving random walk has an intrinsic mechanism that preserves the original affinity relations between sentences in the documents, which is proposed and defined as follows.

Definition 4.1. *For the graph G, the vertex set S has $(n+1)$ vertices: $s_0, s_1, s_2, ..., s_n$. The maximum conductance $C_{max} = max_{i=1,2,...n} C(s_i)$. Of the $(n+1)$ vertices, s_0 is an absorbing vertex with $c(s_i, s_0) \geqslant 0, c(s_0, s_i) = 0$, and $c(s_0, s_0) = C_{max}$ for $i = 1, 2, ..., n$. The remaining vertices are the normal vertices with $c(s_i, s_j) \geqslant 0$ for $i, j = 1, 2, ..., n$. The discrete-time Markov chain $X = (X_0, X_1, X_2, ...)$ with state space and transition probability matrix **P** given by*

$$\begin{aligned} &P(s_i, s_j) = \frac{c(s_i, s_j)}{C_{max}} \\ &P(s_i, s_0) = 1 - \frac{C(s_i)}{C_{max}}, \; P(s_0, s_i) = 0 \\ &P(s_0, s_0) = 1 \\ &\textit{for } i, j = 1, 2, ..., n \end{aligned} \tag{4}$$

is called an affinity-preserving random walk on the graph G.

For our summarization task, we construct a sentence augmented graph G^A (as shown in Figure 4.1) by adding an absorbing vertex s_0 to the sentence affinity graph G. The unabsorbed vertices s_i $(i = 1, 2, ..., n)$ represent sentences in the documents. The affinity-preserving random walk process as defined above is implemented on G^A to rank sentences in the documents. In the affinity-preserving random walk, once the surfer reaches the absorbing vertex, she cannot walk out of there. Because $P(s_i, s_0)$ is small for the vertex s_i with a large conductance, it is less likely for the surfer at s_i to walk into s_0. As for the vertex with a small conductance, the surfer has a tendency to be absorbed by s_0. The absorbing vertex here plays a role of soaking unreliable ranking scores from large numbers of bad candidate sentences and highlighting the few good candidate sentences. The affinity matrix **W** is normalized by its first norm (equivalent to C_{max}) in the affinity-preserving random walk, which results in a kind of "*soft*" stochastic matrix for n unabsorbed vertices. "*soft*" here means that the sum of row elements in the matrix can be less than 1. By contrast, **P** in the traditional random walk is a "*hard*" stochastic matrix in which every sum of row elements has to be 1. Meanwhile, **P** in this absorbing Markov chain (Seneta, 2006) preserves the original affinity relations in **W** as all sentences are globally normalized by the same factor (i.e. C_{max}). We call this approach an "affinity-preserving random walk" as it is used in (Cho et al., 2010), which deals with a graph matching problem that aims at assigning 1-vs-1 correspondences between two graphs. The similar idea is also applied in the case of ontology matching (Xiang et al., 2015). Transition matrix **P** including the absorbing vertex is formulated in (Cho et al., 2010) as follows

$$\mathbf{P} = \begin{pmatrix} 1 & \mathbf{0}^T \\ \mathbf{e} - \mathbf{c}/\|\mathbf{W}\|_1 & \mathbf{W}/\|\mathbf{W}\|_1 \end{pmatrix} \tag{5}$$

where $\mathbf{0}^T$ is a $1 \times n$ vector with all elements 0, **e** is an $n \times 1$ vector with all elements 1, $\mathbf{c} = [C(s_1), C(s_2), ..., C(s_n)]^T$ is a vector containing the conductances of n sentences and $\mathbf{W}/\|\mathbf{W}\|_1$ is the $n \times n$ soft substochastic matrix. However, the stationary distribution of such a random walk on graph with one absorbing vertex is always $\begin{pmatrix} 1 & \mathbf{0}^T \end{pmatrix}$, which is not a good characterization of the sentence ranking distribution. We turn to the quasi-stationary distribution $\bar{\mathbf{x}}$ (Cho et al., 2010; Darroch and Seneta, 1965) of absorbing random walk for ranking sentences. $\bar{\mathbf{x}}^{(K)}$ is defined as

$$\begin{aligned} \bar{\mathbf{x}}_i^{(K)} &= Prob(X^{(K)} = s_i | X^{(K)} \neq s_0) \\ &= \frac{\mathbf{x}_i^{(K)}}{\sum_j \mathbf{x}_j^{(K)}} \end{aligned} \tag{6}$$

where $X^{(K)}$ denotes the position of random surfer at time K. It can be proven that $\bar{\mathbf{x}}^{(K)}$ has its stationary distribution $\bar{\mathbf{x}}$ if $\mathbf{W}$ is irreducible (Darroch and Seneta, 1965). We remove the sentences that have the total conductance 0 (i.e. the isolated sentences) when constructing the sentence affinity graph G. In this way, G will be strongly connected and has an irreducible adjacency matrix $\mathbf{W}$.

We introduce the teleport vector $\mathbf{y}$ as used in personalized PageRank (Page et al., 1999; Haveliwala, 2002; Jeh and Widom, 2003) for the summarization task. In the generic summarization case, we define the vector $\mathbf{y}$ in a way that reflects the position of each sentence in a document. If the sentence s_{i+1} is right after the sentence s_i in the same document, then

$$\frac{\mathbf{y}_{i+1}}{\mathbf{y}_i} = \lambda^{-1}, \lambda > 1 \qquad (7)$$

where λ is the decay factor. In the topic-focused summarization case, we incorporate the topic description as a vertex in the random walk process, which is a standard way of achieving the relevancy goal of this kind of summarization. Vector $\mathbf{y}$ is defined to be $[y_1, y_2, ..., y_n, y_{n+1}]^T$ in which $y_i = 0 (1 \leqslant i \leqslant n)$ and $y_{n+1} = 1$ when the first n elements represent sentences in the documents and the last one represents the topic description. We normalize $\mathbf{y}$ by its first norm to get a prior sentence ranking for multi-document summarization. Based on $\mathbf{W}$ and $\mathbf{y}$, sentence ranking scores in affinity-preserving random walk can be formulated in a recursive form as follows

$$\mathbf{x} = \frac{\mu \mathbf{W}^T / \|\mathbf{W}\|_1 \mathbf{x} + (1-\mu)\mathbf{y}}{\|\mu \mathbf{W}^T / \|\mathbf{W}\|_1 \mathbf{x} + (1-\mu)\mathbf{y}\|_1} \qquad (8)$$

where $\mathbf{x} = [Score(s_i)]_{n \times 1}$ is the vector of sentence ranking scores. μ is the damping factor that trades off between two actions: the transition according to $\mathbf{W}^T/\|\mathbf{W}\|_1$ and the teleport specified by $\mathbf{y}$. Transpose operation in Eq.(8) can be removed because of symmetry of $\mathbf{W}$. The final transition matrix of affinity-preserving random walk is given by $\mathbf{A} = \mu\mathbf{W}/\|\mathbf{W}\|_1 + (1-\mu)\mathbf{y} \cdot \mathbf{e}^T$ and $\mathbf{x}$ should be normalized by its first norm after each iteration of random walk. Like PageRank, the quasi-stationary distribution is obtained by the normalized principal eigenvector of $\mathbf{A}$.

For implementation, the initial ranking scores of all sentences are set to $1/n$ and the iterative process in Eq.(8) is adopted to compute new ranking scores of sentences. Usually convergence of the iterative algorithm is achieved when the difference between scores computed at two successive iterations falls below a given threshold.

4.3 Adjustable Affinity-Preserving Random Walk for Summarization

Affinity-preserving random walk preserves the affinity relations between sentences and gives high ranking scores to the salient sentences. However, the diversity constraint of summarization has not been taken into account. The surfer of affinity-preserving random walk has no knowledge about what a diverse summary should be. If we just take redundancy removing as the post-processing separate step to improve diversity, sentences that highly overlap with other summary sentences may be chosen and sentences that include information about different topics may be submerged. This phenomenon can be explained by the theorem as follows.

Theorem 4.1. *Suppose $\bar{\boldsymbol{x}}$ is the quasi-stationary distribution of affinity-preserving random walk as defined in Sec.4.2 and $\boldsymbol{x}$ is the solution of a continuous quadratic optimization problem $argmax(\boldsymbol{x}^T \boldsymbol{A}\, \boldsymbol{x})$ s.t. $\boldsymbol{x} \in [0,1]^n$, $\|\boldsymbol{x}\|_2 = 1$ and $\boldsymbol{A}$ has definition in Sec.4.2. The following equation holds*

$$\bar{\boldsymbol{x}} = \boldsymbol{x}/\|\boldsymbol{x}\|_1 \qquad (9)$$

when $\mu = 1$.

Proof. In mathematics, for a given symmetric real matrix $\mathbf{A}$ (when $\mu = 1$) and nonzero real vector $\mathbf{x}$, the Rayleigh quotient $R(\mathbf{A}, \mathbf{x})$ is defined as

$$R(\mathbf{A}, \mathbf{x}) = \frac{\mathbf{x}^T \mathbf{A} \mathbf{x}}{\mathbf{x}^T \mathbf{x}}$$

and it reaches its maximum value when $\mathbf{x}$ is the principal eigenvector of $\mathbf{A}$. If $\|\mathbf{x}\|_2 = 1$, $R(\mathbf{A}, \mathbf{x})$ is equivalent to $\mathbf{x}^T \mathbf{A}\, \mathbf{x}$. So the solution $\mathbf{x}$ is the principal eigenvector of $\mathbf{A}$. From Section 4.2, $\bar{\mathbf{x}}$ is the normalized principal eigenvector of $\mathbf{A}$. $\bar{\mathbf{x}}$ and $\mathbf{x}$ have the relation in Eq.(9). The conclusion is made. □

From Theorem 4.1, affinity-preserving random walk tends to produce a stationary distribution in which the total sum of affinity between sentences (i.e. $\mathbf{x}^T \mathbf{A} \mathbf{x}$) is large if there is a subtle teleporting effect. It will lead to a summary consisting of many sentences overlapping with each other, which clearly violates the diversity constraint of summarization. Good candidate sentences may not have high affinity with other sentences and are likely to be submerged by affinity-preserving random walk. Conversely, some bad candidate sentences could have high affinity with others and will be highlighted by the random walk process. An extreme example is that a cluster of sentences will all get high ranking scores if they are very similar to each other. However, only one sentence in this cluster should be included in the summary and the others should be suppressed. The random surfer is caught in a trap of larger sentence cluster, which operates against the exploration of good candidates in smaller cluster.

We introduce adjustable affinity-preserving random walk to enforce the diversity constraint of summarization. In the original affinity-preserving random walk, sentence transition matrix **A** is fixed and set as $\mu\mathbf{W}/\|\mathbf{W}\|_1+(1-\mu)\mathbf{y}\cdot\mathbf{e}^T$. Edge (s_i, s_0) (if $C(s_i) \neq C_{max}$) always exists and has an invariant conductance $c(s_i, s_0)$, which means that the surfer at s_i walks into s_0 in the same manner for the entire random walk process. The random surfer makes her decision only based on invariant **A** to select salient sentences and form the summary. To equip the surfer with knowledge about what a diverse summary should be, we propose to adjust sentence transition matrix **A** in each iteration of random walk. The key point is that good candidate sentences should be normalized *locally* while bad ones should be normalized *globally* in the transformation from affinity matrix **W** to transition matrix **A**.

A "virtual" summary V is produced based on **x** in each iteration of affinity-preserving random walk. "Virtual" here means that V is a summary based on transient distribution **x**, which differs from the final summary based on quasi-stationary distribution $\bar{\mathbf{x}}$. The method of diversity penalty imposition (Wan et al., 2007) is used to produce V, which is denoted by the *producingSummary* function in Algorithm 4.1. It is a simple greedy algorithm to select sentences that are both salient and diverse, which often plays a role of greedy post-processing step to produce the final summary. Rather, we use it to produce virtual summary V that satisfies both the saliency and diversity constraints based on a specific iteration. V is an indicator for the diversity constraint of summarization.

The sentence transition matrix **A** in the iteration $(K+1)$ is then constructed with help of the virtual summary V in the iteration K. Here, different normalization methods are used to transform **W** into **A**. If V includes the sentence s_i, elements in the corresponding *i*-th row of **W** will be normalized by the sum of the row (i.e. $C(s_i)$). Otherwise, elements will be normalized by the maximum sum of row elements in **W** (i.e. C_{max}). In this way, "local normalization" is adopted for the sentences in V while "global normalization" is adopted for the sentences not in V. We differentiate the normalization methods to lead the surfer of affinity-preserving random walk to explore more in the neighborhood of the sentences in V rather than end in the absorbing vertex s_0. As a result, the sentences that satisfy the saliency and diversity constraints will be highlighted even though they are in a small sentence cluster. We characterize different normalizations in the diagonal matrix **D**. $\mathbf{D}_{ii}$ is $C(s_i)$ if $s_i \in V$ and $\mathbf{D}_{ii}$ is C_{max} if $s_i \notin V$,which is denoted by the *adjustingNormalization* function in Algorithm 4.1. **D** in the current iteration is here dependent on **x** in the previous iteration. We will have different sentence augmented graphs G^A in each iteration. Figure 4.2 shows an example of $G^A(K)$ and $G^A(K+1)$ in the respective iterations K and $(K+1)$. The probability distribution in the adjustable affinity-preserving random walk is updated as follows.

Algorithm 4.1: Adjustable Affinity-Preserving Random Walk for Multi-Document Summarization

```
Input: The sentence affinity matrix, W; The
       starting and maximum number of
       iteration, $B$ and $M$; The teleport
       vector, y; The damping factor, $\mu$;
Output: The multi-document summary, $V$;
$\mathbf{A} \leftarrow \mu\mathbf{W}/C_{max} + (1-\mu)\mathbf{y}\cdot\mathbf{e}^T$
Initialize the starting distribution x as uniform
for $i \leftarrow 1, 2, ..., M$ do
    if $i > B$ then
        $V \leftarrow$ producingSummary(x)
        $\mathbf{D} \leftarrow$ adjustingNormalization($V$)
        $\mathbf{A} \leftarrow \mu(\mathbf{D}^{-1}\mathbf{W})^T + (1-\mu)\mathbf{y}\cdot\mathbf{e}^T$
    $\bar{\mathbf{x}} \leftarrow \mathbf{A}\mathbf{x}$
    $\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}}/\|\bar{\mathbf{x}}\|_1$
    if $\|\bar{\mathbf{x}} - \mathbf{x}\|_1 < \varepsilon$ then
        break
    $\mathbf{x} \leftarrow \bar{\mathbf{x}}$
$V \leftarrow$ producingSummary(x)
Return $V$
```

$$\mathbf{x} = \frac{\mu(\mathbf{D}^{-1}\mathbf{W})^T\mathbf{x} + (1-\mu)\mathbf{y}}{\|\mu(\mathbf{D}^{-1}\mathbf{W})^T\mathbf{x} + (1-\mu)\mathbf{y}\|_1} \tag{10}$$

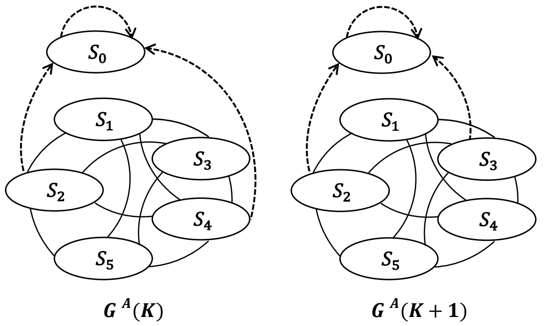


Figure 4.2: Sentence augmented graphs for summarization in two successive iterations. $G^A(K)$: augmented graph in the iteration K. Virtual summary $V = \{s_1, s_3, s_5\}$, which is constructed from $\mathbf{x}$ in the iteration $(K-1)$ by *producingSummary*. $\mathbf{D} = diag([C(s_1), C_{max}, C(s_3), C_{max}, C(s_5)])$. $G^A(K+1)$: augmented graph in the iteration $(K+1)$. $V = \{s_1, s_4, s_5\}$, which is constructed from $\mathbf{x}$ in the iteration K by *producingSummary*. $\mathbf{D} = diag([C(s_1), C_{max}, C_{max}, C(s_4), C(s_5)])$. In both cases, C_{max} equals to $C(s_1)$ indicating that sentence s_1 has the maximum conductance.

This is an adjustable Markov chain for which the transition matrix $\mathbf{A}$ is $\mu(\mathbf{D}^{-1}\mathbf{W})^T + (1-\mu)\mathbf{y} \cdot \mathbf{e}^T$. In this setting, $\mathbf{A}$ is dependent on the transient distribution $\mathbf{x}$ in the previous iteration, which differs from the invariant transition matrix in Eq.(8). As the diversity constraint is embedded in $\mathbf{A}$, subsequent random walks move to the solution that induces a better summary. The algorithm of the adjustable affinity-preserving random walk for multi-document summarization is demonstrated in Algorithm 4.1.

The parameter B in Algorithm 4.1 is used to produce a transient distribution which is reliable enough to adjust the transition matrix. To get the final multi-document summary, we use the same *producingSummary* function.

5 Experiments

5.1 Data Sets

Generic and topic-focused multi-document summarization have been the main tasks in DUC[1]. Task 2 of DUC 2004 is a generic summarization task and task 3 of DUC 2003 is a topic-focused summarization task. Both tasks are used for performance evaluation of our method. In the experiments, task 2 of DUC 2003 is used for the parameter tuning of our method. We preprocess the document data sets by removing stopwords from each sentence and stemming the remaining words using the Porter's stemmer[2]. Also, the sentences containing the said clause (if a said, says, told, tells word and quotation marks appear simultaneously) are filtered out.

For evaluation, four reference summaries generated by human judges for each document cluster are provided by DUC as the ground truth. A brief summary over the evaluation datasets is shown in Table 5.1. According to (Hong et al., 2014), we adjust the length limit of summary in DUC 2004 from 665 bytes to 100 words as it provides the same setting for system evaluations.

	DUC 2003	DUC 2003	DUC 2004
Task	Task 2	Task 3	Task 2
Type	Generic	Topic-focused	Generic
Cluster numbers	30	30	50
Data source	TDT	TREC	TDT
Summary length	100 words	100 words	100 words

Table 5.1: Summary of data sets used in our experiments.

5.2 Evaluation Metric

We use the ROUGE-1.5.5 (Lin and Hovy, 2003) toolkit for evaluation, which has been officially adopted by DUC for automatic summarization evaluation. The toolkit measures summary quality by counting overlapping units such as the n-gram, word sequences and word pairs between the candidate summary and the reference summary. ROUGE-N is an n-gram based measure and the ROUGE-N recall is computed as follows

$$\textit{ROUGE-N}_R = \frac{\sum\limits_{S \in \{\textit{RefSum}\}} \sum\limits_{\textit{n-gram} \in S} \textit{Count}_{\textit{match}}(\textit{n-gram})}{\sum\limits_{S \in \{\textit{RefSum}\}} \sum\limits_{\textit{n-gram} \in S} \textit{Count}(\textit{n-gram})} \quad (11)$$

where n stands for the length of the n-gram, and *Count*$_{match}$*(n-gram)* is the maximum number of n-grams co-occurring in the candidate summary and the set of reference summaries. *Count(n-gram)* is the number of n-grams in the reference summaries.

We conduct our ROUGE experiments following the recommended standard in (Owczarzak et al.,

[1] http://www-nlpir.nist.gov/projects/duc/intro.html

[2] https://tartarus.org/martin/PorterStemmer/

2012; Hong et al., 2014)[3]. We compute ROUGE-2 recall with stemming and stopwords not removed, which provides the best agreement with manual evaluations. We also compute ROUGE-1 recall which has the highest recall of ability to identify the better summary in a pair, and ROUGE-4 recall which has the highest precision of ability to identify the better summary in a pair (Owczarzak et al., 2012).

5.3 Experimental Results

In the experiments, the parameters of our method are set as follows: the decay factor λ is 2, the maximum number of iteration M is 100, the number of starting iteration B is 30, the damping factor μ is 0.85 and the minimum error ε is 1E-30.

System	R-1	R-2	R-4
Cont. LexPageRank*	35.95	7.47	0.82
FreqSum	35.30	8.11	1.00
CLASSY 04	37.62	8.96	1.51
CLASSY 11	37.22	9.20	1.48
GRASSHOPPER*	37.20	9.26	1.50
DivRank*	37.60	9.30	1.52
GCD*	38.68	9.31	1.45
Submodular	39.18	9.35	1.39
APRW*	38.10	9.39	1.35
DPP	39.79	9.62	1.57
ICSISumm	38.41	9.78	1.73
AAPRW*	38.92	10.06	1.61
WFS-NMF	39.24	10.94	1.65

Table 5.2: System comparisons on task 2 of DUC 2004 (%). *: Graph-based ranking methods.

Table 5.2 shows the performance of our method and other eleven well-known systems on task 2 of DUC 2004 according to ROUGE-1,2,4 recall, sorted by ROUGE-2 recall in the ascending order. Some of the results are from (Hong et al., 2014). *Cont. LexPageRank* (Erkan and Radev, 2004) is a graph-based ranking method and a representative of traditional random walk approach. Here we employ the continuous version of LexPageRank. *FreqSum* (Nenkova et al., 2006) is a simple approach to approximate the importance of words with their probability in the input and then select sentences with high average word probability. *CLASSY 04* (Conroy et al., 2004) was the participant of the official DUC 2004 evaluation with the best evaluation score. It employs a Hidden Markov Model using topic signature feature and requires a linguistic preprocessing component. *CLASSY 11* (Conroy et al., 2011) is the successor of *CLASSY 04* and selects the non-redundant sentences using the non-negative matrix factorization algorithm. In the *Submodular* system (Lin and Bilmes, 2011), multi-document summarization is formulated as a submodular set function maximization problem. *DPP* (Lin and Bilmes, 2011) combines a sentence saliency model with a global diversity model encouraging non-overlapping information. *ICSISumm* (Gillick and Favre, 2009) aims at finding the globally optimal summary by formulating the summarization task in Integer Linear Programming. *WFS-NMF* (Wang et al., 2010) extends the non-negative matrix factorization algorithm and provides a good framework for weighting different terms and documents. *GRASSHOPPER*, *DivRank* and *GCD* are the three graph-based ranking models mentioned in Section 2. *APRW* and *AAPRW* are our methods. *APRW* is the method of affinity-preserving random walk described in Section 4.2 and *AAPRW* is the method of adjustable affinity-preserving random walk described in Section 4.3.

System	R-1	R-2	R-4
S17	31.81	4.98	0.47
S13	31.99	5.83	0.73
S16	35.00	7.31	1.04
Manifold Ranking	37.33	7.68	1.26
APRW	35.72	7.72	1.34
AAPRW	36.36	8.21	1.40

Table 5.3: System comparisons on task 3 of DUC 2003 (%).

Table 5.3 shows the evaluation results on task 3 of DUC 2003 according to ROUGE-1,2,4 recall, sorted also by ROUGE-2 recall in the ascending order. *S13*, *S16* and *S17* are the system IDs of the top performing systems in the official DUC 2003 evaluation, whose details are described in DUC publications (Zhou and Hovy, 2003; Chali et al., 2003). *Manifold Ranking* is the method proposed in (Wan et al., 2007) to make use of both the relationships among all sentences in the documents and the relationships between the given topic de-

[3] ROUGE-1.5.5 with the parameters: -n 4 -m -a -l 100 -x -c 95 -r 1000 -f A -p 0.5 -t 0

scription and the sentences. *APRW* and *AAPRW* are our methods.

From Tables 5.2 and 5.3, our method has the best ROUGE-2 score among all graph-based ranking methods for generic multi-document summarization, and it also has the best ROUGE-2 score for topic-focused multi-document summarization. *AAPRW* has the ROUGE-2 score 10.06% on DUC 2004 task 2, which is 0.28% higher than the best system *ICSISumm* reported by (Hong et al., 2014) and 1.1% higher than the official best system *CLASSY 04*. *WFS-NMF* has the overall best score on DUC 2004 task 2 due to the sentence feature selection and the weights on the document side, which is reported by (Wang et al., 2010; Alguliev et al., 2013). *AAPRW* has the ROUGE-2 score 8.21% on DUC 2003 task 3, which is 0.53% higher than *Manifold Ranking* and 0.9% higher than the official best system *S16*. In DUC 2004 *AAPRW* has 0.67% more ROUGE-2 score than *APRW* and the gap is 0.49% in DUC 2003, which proves the effectiveness of the adjustable transition matrix in the random walk process. It is worth mentioning that our method has the best ROUGE-4 score on the DUC 2003 topic-focused summarization task.

We conducted the two-sided Wilcoxon signed-rank tests between each pair of *AAPRW* and other methods. For the generic summarization in DUC 2004, our method provides a significant improvement over the official best system *CLASSY 04* on ROUGE-2 (with p-value lower than 0.05). For the query-focused summarization in DUC 2003, our method also provides a significant improvement over *S17*, *S13* and *S16* on ROUGE-2.

In order to further investigate the influences of the parameter in our proposed method, the damping factor μ is varied from 0 to 1. Figures 5.1 and 5.2 show the ROUGE-1 and ROUGE-2 recall curves of our method on the two data sets, respectively. We can see from the figures that the damping factor has an effect on the performance of multi-document summarization.

6 Conclusion and Future Work

In this paper we propose the adjustable affinity-preserving random walk for generic and topic-focused multi-document summarization, which deals with the saliency and diversity goals in a unified framework. Experiments demonstrate the effectiveness of our method.

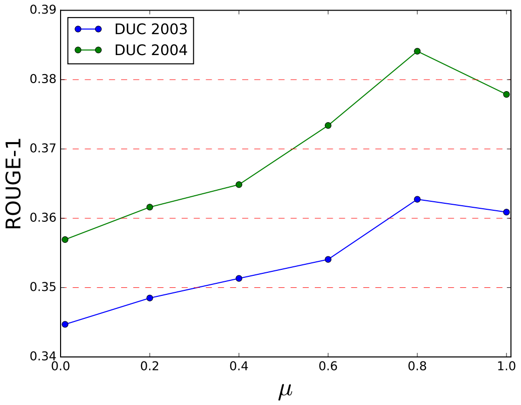


Figure 5.1: ROUGE-1 recall scores vs. μ of our method.

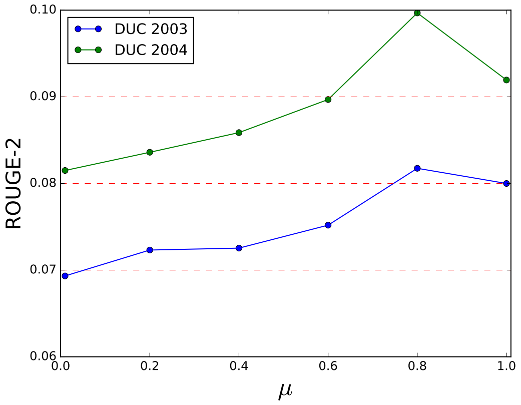


Figure 5.2: ROUGE-2 recall scores vs. μ of our method.

In the future work, we will focus on the self transition of adjustable affinity preserving random walk, which could be used to remove the redundancy between summary sentences.

Acknowledgements

We would like to thank our three anonymous reviewers for their helpful advice on various aspects of this work. This research is supported by National Key Basic Research Program of China (No.2014CB340504) and National Natural Science Foundation of China (No.61375074,61273318). The contact authors for this paper are Zhifang Sui and Baobao Chang.

References

Rasim M Alguliev, Ramiz M Aliguliyev, and Nijat R Isazade. 2013. Multiple documents summarization based on evolutionary optimization algorithm. *Expert Systems with Applications*, 40(5):1675–1689.

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern information retrieval*, volume 463. ACM press New York.

Yllias Chali, Maheedhar Kolla, Nanak Singh, and Zhenshuan Zhang. 2003. The university of lethbridge text summarizer at duc 2003. In *the Proceedings of the HLT/NAACL workshop on Automatic Summarization/Document Understanding Conference (DUC 2003)*.

Minsu Cho, Jungmin Lee, and Kyoung Lee. 2010. Reweighted random walks for graph matching. *Computer Vision–ECCV 2010*, pages 492–505.

John M Conroy, Judith D Schlesinger, Jade Goldstein, and Dianne P Oleary. 2004. Left-brain/right-brain multi-document summarization. In *Proceedings of the Document Understanding Conference (DUC 2004)*.

John M Conroy, Judith D Schlesinger, Jeff Kubina, Peter A Rankel, and Dianne P O'Leary. 2011. Classy 2011 at tac: Guided and multi-lingual summaries and evaluation metrics. *TAC*, 11:1–8.

John N Darroch and Eugene Seneta. 1965. On quasi-stationary distributions in absorbing discrete-time finite markov chains. *Journal of Applied Probability*, 2(1):88–100.

Avinava Dubey, Soumen Chakrabarti, and Chiranjib Bhattacharyya. 2011. Diversity in ranking via resistive graph centers. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 78–86. ACM.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 10–18. Association for Computational Linguistics.

Taher H Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526. ACM.

Kai Hong, John M Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. In *LREC*, pages 1608–1616.

Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pages 271–279. ACM.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics.

Qiaozhu Mei, Jian Guo, and Dragomir Radev. 2010. Divrank: the interplay of prestige and diversity in information networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1009–1018. Acm.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 573–580. ACM.

Karolina Owczarzak, John M Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9. Association for Computational Linguistics.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Eugene Seneta. 2006. *Non-negative matrices and Markov chains*. Springer Science & Business Media.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Manifold-ranking based topic-focused multi-document summarization. In *IJCAI*, volume 7, pages 2903–2908.

Dingding Wang, Tao Li, and Chris Ding. 2010. Weighted feature subset non-negative matrix factorization and its applications to document understanding. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 541–550. IEEE.

Chuncheng Xiang, Baobao Chang, and Zhifang Sui. 2015. An ontology matching approach based on affinity-preserving random walks. In *IJCAI*, pages 1471–1478.

Dengyong Zhou, Jason Weston, Arthur Gretton, Olivier Bousquet, and Bernhard Schölkopf. 2003. Ranking on data manifolds. In *NIPS*, volume 3.

Liang Zhou and Eduard Hovy. 2003. Headline summarization at isi. In *Document Understanding Conference (DUC-2003), Edmonton, Alberta, Canada*.

Xiaojin Zhu, Andrew B Goldberg, Jurgen Van Gael, and David Andrzejewski. 2007. Improving diversity in ranking using absorbing random walks. In *HLT-NAACL*, pages 97–104.

A Mention-Ranking Model for Abstract Anaphora Resolution

Ana Marasović, Leo Born,[†] Juri Opitz,[†] and Anette Frank[†]
Research Training Group AIPHES
Department of Computational Linguistics
Heidelberg University
69120 Heidelberg, Germany
{marasovic,born,opitz,frank}@cl.uni-heidelberg.de

Abstract

Resolving abstract anaphora is an important, but difficult task for text understanding. Yet, with recent advances in representation learning this task becomes a more tangible aim. A central property of *abstract anaphora* is that it establishes a relation between the *anaphor* embedded in the *anaphoric sentence* and its (typically non-nominal) *antecedent*. We propose a mention-ranking model that learns how abstract anaphors relate to their antecedents with an LSTM-Siamese Net. We overcome the lack of training data by generating artificial anaphoric sentence–antecedent pairs. Our model outperforms state-of-the-art results on *shell noun resolution*. We also report first benchmark results on an abstract anaphora subset of the ARRAU corpus. This corpus presents a greater challenge due to a mixture of nominal and pronominal anaphors and a greater range of confounders. We found model variants that outperform the baselines for nominal anaphors, without training on individual anaphor data, but still lag behind for pronominal anaphors. Our model selects syntactically plausible candidates and – if disregarding syntax – discriminates candidates using deeper features.

1 Introduction

Current research in anaphora (or coreference) resolution is focused on resolving noun phrases referring to concrete objects or entities in the real world, which is arguably the most frequently occurring type. Distinct from these are diverse types of *abstract* anaphora (AA) (Asher, 1993) where reference is made to propositions, facts, events or properties. An example is given in (1) below.[1]

While recent approaches address the resolution of selected abstract *shell nouns* (Kolhatkar and Hirst, 2014), we aim to resolve a wide range of abstract anaphors, such as the NP *this trend* in (1), as well as pronominal anaphors (*this*, *that*, or *it*).

Henceforth, we refer to a sentence that contains an abstract anaphor as *the anaphoric sentence (AnaphS)*, and to a constituent that the anaphor refers to as *the antecedent (Antec)* (cf. (1)).

(1) Ever-more powerful desktop computers, designed with one or more microprocessors as their ”brains”, are expected to increasingly take on functions carried out by more expensive minicomputers and mainframes. ”[$_{Antec}$ ***The guys that make traditional hardware are really being obsoleted by microprocessor-based machines***]”, said Mr. Benton. [$_{AnaphS}$ As a result of ***this trend***$_{AA}$, longtime powerhouses HP, IBM and Digital Equipment Corp. are scrambling to counterattack with microprocessor-based systems of their own.]

A major obstacle for solving this task is the lack of sufficient amounts of annotated training data. We propose a method to generate large amounts of training instances covering a wide range of abstract anaphor types. This enables us to use neural methods which have shown great success in related tasks: coreference resolution (Clark and Manning, 2016a), textual entailment (Bowman et al., 2016), learning textual similarity (Mueller and Thyagarajan, 2016), and discourse relation sense classification (Rutherford et al., 2017).

Our model is inspired by the mention-ranking model for coreference resolution (Wiseman et al., 2015; Clark and Manning, 2015, 2016a,b) and combines it with a Siamese Net (Mueller and Thyagarajan, 2016), (Neculoiu et al., 2016) for

[†] Leo Born, Juri Opitz and Anette Frank contributed equally to this work.

[1] Example drawn from ARRAU (Uryupina et al., 2016).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 221–232
Copenhagen, Denmark, September 7–11, 2017.

learning similarity between sentences. Given an *anaphoric sentence* (AntecS in (1)) and a candidate antecedent (any constituent in a given context, e.g. *being obsoleted by microprocessor-based machines* in (1)), the LSTM-Siamese Net learns representations for the candidate and the anaphoric sentence in a shared space. These representations are combined into a joint representation used to calculate a score that characterizes the relation between them. The learned score is used to select the highest-scoring antecedent candidate for the given anaphoric sentence and hence its anaphor. We consider one anaphor at a time and provide the embedding of the *context of the anaphor* and the embedding of the *head of the anaphoric phrase* to the input to characterize each individual anaphor – similar to the encoding proposed by Zhou and Xu (2015) for individuating multiply occurring predicates in SRL. With deeper inspection we show that the model learns a relation between the anaphor in the anaphoric sentence and its antecedent. Fig. 1 displays our architecture.

In contrast to other work, our method for generating training data is not confined to specific types of anaphora such as shell nouns (Kolhatkar and Hirst, 2014) or anaphoric connectives (Stede and Grishina, 2016). It produces large amounts of instances and is easily adaptable to other languages. This enables us to build a robust, knowledge-lean model for abstract anaphora resolution that easily extends to multiple languages.

We evaluate our model on the shell noun resolution dataset of Kolhatkar et al. (2013b) and show that it outperforms their state-of-the-art results. Moreover, we report results of the model (trained on our newly constructed dataset) on unrestricted abstract anaphora instances from the ARRAU corpus (Poesio and Artstein, 2008; Uryupina et al., 2016). To our knowledge this provides the first state-of-the-art benchmark on this data subset.

Our TensorFlow[2] implementation of the model and scripts for data extraction are available at: `https://github.com/amarasovic/neural-abstract-anaphora`.

2 Related and prior work

Abstract anaphora has been extensively studied in linguistics and shown to exhibit specific properties in terms of semantic antecedent types, their degrees of abstractness, and general discourse properties (Asher, 1993; Webber, 1991). In contrast to nominal anaphora, abstract anaphora is difficult to resolve, given that agreement and lexical match features are not applicable. Annotation of abstract anaphora is also difficult for humans (Dipper and Zinsmeister, 2012), and thus, only few smaller-scale corpora have been constructed. We evaluate our models on a subset of the ARRAU corpus (Uryupina et al., 2016) that contains abstract anaphors and the shell noun corpus used in Kolhatkar et al. (2013b).[3] We are not aware of other freely available abstract anaphora datasets.

Little work exists for the **automatic resolution of abstract anaphora**. Early work (Eckert and Strube, 2000; Strube and Müller, 2003; Byron, 2004; Müller, 2008) has focused on spoken language, which exhibits specific properties. Recently, **event coreference** has been addressed using feature-based classifiers (Jauhar et al., 2015; Lu and Ng, 2016). Event coreference is restricted to a subclass of *events*, and usually focuses on coreference between verb (phrase) and noun (phrase) mentions of similar abstractness levels (e.g. *purchase – acquire*) with no special focus on (pro)nominal anaphora. Abstract anaphora typically involves a full-fledged clausal antecedent that is referred to by a highly abstract (pro)nominal anaphor, as in (1).

Rajagopal et al. (2016) proposed a model for resolution of events in biomedical text that refer to a single or multiple clauses. However, instead of selecting the correct antecedent clause(s) (our task) for a given event, their model is restricted to classifying the event into six abstract categories: this these *changes, responses, analysis, context, finding, observation*, based on its surrounding context. While related, their task is not comparable to the full-fledged abstract anaphora resolution task, since the events to be classified are known to be coreferent and chosen from a set of restricted abstract types.

More related to our work is Anand and Hardt (2016) who present an antecedent ranking account for **sluicing** using classical machine learning based on a small training dataset. They employ features modeling distance, containment, discourse structure, and – less effectively – content and lexical correlates.[4]

Closest to our work is Kolhatkar et al. (2013b)

[2] Abadi et al. (2015)

[3] We thank the authors for making their data available.

[4] Their data set was not publicized.

(KZH13) and Kolhatkar and Hirst (2014) (KH14) on **shell noun resolution**, using classical machine learning techniques. Shell nouns are abstract nouns, such as *fact*, *possibility*, or *issue*, which can only be interpreted jointly with their *shell* content (their embedded clause as in (2) or antecedent as in (3)). KZH13 refer to shell nouns whose antecedent occurs in the prior discourse as *anaphoric shell nouns* (ASNs) (cf. (3)), and *cataphoric shell nouns* (CSNs) otherwise (cf. (2)).[5]

(2) Congress has focused almost solely on ***the fact*** that [*special education is expensive - and that it takes away money from regular education.*]

(3) Environmental Defense [...] notes that [$_{Antec}$ *Mowing the lawn with a gas mower produces as much pollution [...] as driving a car 172 miles.*] [$_{AnaphS}$ ***This fact*** may [...] explain the recent surge in the sales of [...] old-fashioned push mowers [...]].

KZH13 presented an approach for resolving six typical shell nouns following the observation that CSNs are easy to resolve based on their syntactic structure alone, and the assumption that ASNs share linguistic properties with their embedded (CSN) counterparts. They manually developed rules to identify the embedded clause (i.e. cataphoric antecedent) of CSNs and trained SVM^{rank} (Joachims, 2002) on such instances. The trained SVM^{rank} model is then used to resolve ASNs. KH14 generalized their method to be able to create training data for any given shell noun, however, their method heavily exploits the specific properties of shell nouns and does not apply to other types of abstract anaphora.

Stede and Grishina (2016) study a related phenomenon for German. They examine inherently anaphoric connectives (such as *demzufolge – according to which*) that could be used to access their abstract antecedent in the immediate context. Yet, such connectives are restricted in type, and the study shows that such connectives are often ambiguous with nominal anaphors and require sense disambiguation. We conclude that they cannot be easily used to acquire antecedents automatically.

In our work, we explore a different direction: we construct artificial training data using a general pattern that identifies embedded sentence constituents, which allows us to extract relatively secure training data for abstract anaphora that captures a wide range of anaphora-antecedent relations, and apply this data to train a model for the resolution of unconstrained abstract anaphora.

Recent work in entity coreference resolution has proposed powerful neural network-based models that we will adapt to the task of abstract anaphora resolution. Most relevant for our task is the **mention-ranking neural coreference model** proposed in Clark and Manning (2015), and their improved model in Clark and Manning (2016a), which integrates a loss function (Wiseman et al., 2015) which learns distinct feature representations for anaphoricity detection and antecedent ranking.

Siamese Nets distinguish between similar and dissimilar pairs of samples by optimizing a loss over the metric induced by the representations. It is widely used in vision (Chopra et al., 2005), and in NLP for semantic similarity, entailment, query normalization and QA (Mueller and Thyagarajan, 2016; Neculoiu et al., 2016; Das et al., 2016).

3 Mention-Ranking Model

Given an anaphoric sentence s with a marked anaphor (mention) and a candidate antecedent c, the mention-ranking (MR) model assigns the pair (c, s) a score, using representations produced by an LSTM-Siamese Net. The highest-scoring candidate is assigned to the marked anaphor in the anaphoric sentence. Fig. 1 displays the model.

We learn representations of an anaphoric sentence s and a candidate antecedent c using a bi-directional Long Short-Term Memory (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005). One bi-LSTM is applied to the anaphoric sentence s and a candidate antecedent c, hence the term siamese. Each word is represented with a vector $\boldsymbol{w_i}$ constructed by concatenating embeddings of the word, of the context of the anaphor (average of embeddings of the anaphoric phrase, the previous and the next word), of the head of the anaphoric phrase[6], and, finally, an embedding of the constituent tag of the candidate, or the S constituent tag if the word is in the anaphoric sentence. For each sequence s or c, the word vectors $\boldsymbol{w_i}$ are sequentially fed into the bi-LSTM, which produces outputs from the forward pass, $\overrightarrow{\boldsymbol{h_i}}$, and outputs $\overleftarrow{\boldsymbol{h_i}}$ from the backward pass. The final output of the i-th word is defined as $\boldsymbol{h_i} = [\overleftarrow{\boldsymbol{h_i}}; \overrightarrow{\boldsymbol{h_i}}]$. To get a representation of the full sequence, $\boldsymbol{h_s}$ or $\boldsymbol{h_c}$, all outputs are averaged, except for those that correspond to padding tokens.

[5] We follow this terminology for their approach and data representation.

[6] Henceforth we refer to it as embedding of the anaphor.

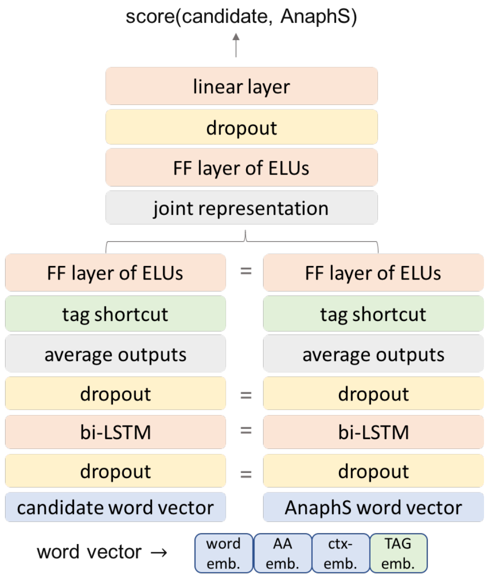


Figure 1: Mention-ranking architecture for abstract anaphora resolution (MR-LSTM).

To prevent forgetting the constituent tag of the sequence, we concatenate the corresponding tag embedding with $\boldsymbol{h_s}$ or $\boldsymbol{h_c}$ (we call this a *shortcut* for the tag information). The resulting vector is fed into a feed-forward layer of exponential linear units (ELUs) (Clevert et al., 2016) to produce the final representation $\tilde{\boldsymbol{h}}_{\boldsymbol{s}}$ or $\tilde{\boldsymbol{h}}_{\boldsymbol{c}}$ of the sequence.

From $\tilde{\boldsymbol{h}}_{\boldsymbol{c}}$ and $\tilde{\boldsymbol{h}}_{\boldsymbol{s}}$ we compute a vector $\boldsymbol{h_{c,s}} = [|\tilde{\boldsymbol{h}}_{\boldsymbol{c}} - \tilde{\boldsymbol{h}}_{\boldsymbol{s}}|; \tilde{\boldsymbol{h}}_{\boldsymbol{c}} \odot \tilde{\boldsymbol{h}}_{\boldsymbol{s}}]$ (Tai et al., 2015), where $|–|$ denotes the absolute values of the element-wise subtraction, and $\odot$ the element-wise multiplication. Then $\boldsymbol{h_{c,s}}$ is fed into a feed-forward layer of ELUs to obtain the final joint representation, $\tilde{\boldsymbol{h}}_{\boldsymbol{c,s}}$, of the pair (c, s). Finally, we compute the score for the pair (c, s) that represents relatedness between them, by applying a single fully connected linear layer to the joint representation:

$$score(c, s) = W\tilde{\boldsymbol{h}}_{\boldsymbol{c,s}} + b \in \mathbb{R}, \qquad (1)$$

where W is a $1 \times d$ weight matrix, and d the dimension of the vector $\tilde{\boldsymbol{h}}_{\boldsymbol{c,s}}$.

We train the described mention-ranking model with the max-margin training objective from Wiseman et al. (2015), used for the antecedent ranking subtask. Suppose that the training set $\mathcal{D} = \{(a_i, s_i, \mathcal{T}(a_i), \mathcal{N}(a_i)\}_{i=1}^{n}$, where a_i is the i-th abstract anaphor, s_i the corresponding anaphoric sentence, $\mathcal{T}(a_i)$ the set of antecedents of a_i and $\mathcal{N}(a_i)$ the set of candidates that are not antecedents (negative candidates). Let $\tilde{t}_i = \arg\max_{t\in\mathcal{T}(a_i)} score(t_i, s_i)$ be the highest scoring antecedent of a_i. Then the loss is given by

$$\sum_{i=1}^{n} \max(0, \max_{c\in\mathcal{N}(a_i)} \{1+score(c, s_i)-score(\tilde{t}_i, s_i)\}).$$

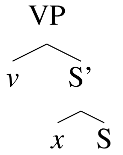


Figure 2: A general pattern for artificially creating anaphoric sentence–antecedent pairs.

4 Training data construction

We create large-scale training data for abstract anaphora resolution by exploiting a common construction, consisting of a verb with an embedded sentence (complement or adverbial) (cf. Fig. 2). We detect this pattern in a parsed corpus, 'cut off' the S′ constituent and replace it with a suitable anaphor to create the anaphoric sentence (AnaphS), while S yields the antecedent (Antec). This method covers a wide range of anaphora-antecedent constellations, due to diverse semantic or discourse relations that hold between the clause hosting the verb and the embedded sentence.

First, the pattern applies to verbs that embed sentential arguments. In (4), the verb *doubt* establishes a specific semantic relation between the embedding sentence and its sentential complement.

(4) He doubts [$_{S'}$ [$_S$ a Bismarckian super state will emerge that would dominate Europe], but warns of "a risk of profound change in the [..] European Community from a Germany that is too strong, even if democratic"].

From this we extract the artificial antecedent *A Bismarckian super state will emerge that would dominate Europe*, and its corresponding anaphoric sentence *He doubts* ***this****, but warns of "a risk of profound change ... even if democratic"*, which we construct by randomly choosing one of a predefined set of appropriate anaphors (here: *this, that, it*), cf. Table 1. The second row in Table 1 is used when the head of S′ is filled by an overt complementizer (*doubts that*), as opposed to (4). The remaining rows in Table 1 apply to adverbial clauses of different types.

Adverbial clauses encode specific discourse relations with their embedding sentences, often indicated by their conjunctions. In (5), for example, the causal conjunction **as** relates a cause (embedded sentence) and its effect (embedding sentence):

type	head of S′	possible anaphoric phrase
empty	∅	this, that
general	that, this	that, this
causal	because, as	therefore, because of this/that,
temporal	while, since, etc.	during this/that
conditional	if, whether	if this/that is true

Table 1: S′-heads and the anaphoric types and phrases they induce (most frequent interpretation).

(5) There is speculation that property casualty firms will sell even more munis [$_{S'}$ **as** [$_S$ they scramble to raise cash to pay claims related to Hurricane Hugo [..]]].

We randomly replace causal conjunctions *because, as* with appropriately adjusted anaphors, e.g. *because of that*, *due to this* or *therefore* that make the causal relation explicit in the anaphor.[7]

Compared to the shell noun corpus of KZH13, who made use of a carefully constructed set of extraction patterns, a downside of our method is that our artificially created antecedents are uniformly of type S. However, the majority of abstract anaphora antecedents found in the existing datasets are of type S. Also, our models are intended to induce semantic representations, and so we expect syntactic form to be less critical, compared to a feature-based model.[8] Finally, the general extraction pattern in Fig. 2, covers a much wider range of anaphoric types.

Using this method we generated a dataset of artificial anaphoric sentence–antecedent pairs from the WSJ part of the PTB Corpus (Marcus et al., 1993), automatically parsed using the Stanford Parser (Klein and Manning, 2003).

5 Experimental setup

5.1 Datasets

We evaluate our model on two types of anaphora: (a) *shell noun anaphora* and (b) (pro)nominal abstract anaphors extracted from ARRAU.

a. Shell noun resolution dataset. For comparability we train and evaluate our model for *shell noun resolution*, using the original training (CSN) and test (ASN) corpus of Kolhatkar et al. (2013a,b).[9] We follow the data preparation and evaluation protocol of Kolhatkar et al. (2013b) (**KZH13**).

The **CSN corpus** was constructed from the NYT corpus using manually developed patterns to identify the antecedent of cataphoric shell nouns (CSNs). In KZH13, all syntactic constituents of the sentence that contains both the CSN and its antecedent were considered as candidates for training a ranking model. Candidates that differ from the antecedent in only one word or one word and punctuation were as well considered as antecedents[10]. To all other candidates we refer to as *negative* candidates. For every shell noun, KZH13 used the corresponding part of the CSN data to train SVM^{rank}.

The **ASN corpus** serves as the test corpus. It was also constructed from the NYT corpus, by selecting anaphoric instances with the pattern "*this* ⟨*shell noun*⟩" for all covered shell nouns. For validation, Kolhatkar et al. (2013a) crowdsourced annotations for the sentence which contains the antecedent, which KZH13 refer to as a *broad region*. Candidates for the antecedent were obtained by using all syntactic constituents of the broad region as candidates and ranking them using the SVM^{rank} model trained on the CSN corpus. The top 10 ranked candidates were presented to the crowd workers and they chose the best answer that represents the ASN antecedent. The workers were encouraged to select *None* when they did not agree with any of the displayed answers and could provide information about how satisfied they were with the displayed candidates. We consider this dataset as gold, as do KZH13, although it may be biased towards the offered candidates.[11]

b. Abstract anaphora resolution data set. We use the automatically constructed data from the WSJ corpus (Section 4) for training.[12] Our test data for unrestricted abstract anaphora resolution is obtained from the **ARRAU corpus** (Uryupina et al., 2016). We extracted all abstract anaphoric instances from the WSJ part of ARRAU that are marked with the category *abstract* or *plan*,[13] and call the subcorpus **ARRAU-AA**.

[7] In case of ambiguous conjunctions (e.g. *as* interpreted as causal or temporal), we generally choose the most frequent interpretation.

[8] This also alleviates problems with languages like German, where (non-)embedded sentences differ in surface position of the finite verb. We can either adapt the order or ignore it, when producing anaphoric sentence – antecedent pairs.

[9] We thank the authors for providing the available data.

[10] We obtained this information from the authors directly.

[11] The authors provided us with the workers' annotations of the broad region, antecedents chosen by the workers and links to the NYT corpus. The extraction of the anaphoric sentence and the candidates had to be redone.

[12] We excluded any documents that are part of ARRAU.

[13] ARRAU distinguishes *abstract* anaphors and (mostly) pronominal anaphors referring to an action or plan, as *plan*.

		shell noun		abstract anaphora	
		CSN train	**ASN** test	**artifical** train	**ARRAU-AA** test
# shell nouns / anaphors		114492	2303	8527	600
median # of tokens	Antec	12.75	13.87	11	20.5
	AnaphS	11.5	24	19	28
median #	Antec	2	4.5	2	1
	negatives	44.5	39	15	48
#	nominal	114492	2303	0	397
	pronominal	0	0	8527	203

Table 2: Data statistics. For the ASN and CSN we report statistics over all shell nouns, but classifiers are trained independently.

Candidates extraction. Following KZH13, for every anaphor we create a list of candidates by extracting all syntactic constituents from sentences which contain antecedents. Candidates that differ from antecedents in only one word, or one word and punctuation, were as well considered as antecedents. Constituents that are not antecedents are considered as negative candidates.

Data statistics. Table 2 gives statistics of the datasets: the number of anaphors (row 1), the median length (in tokens) of antecedents (row 2), the median length (in tokens) for all anaphoric sentences (row 3), the median of the number of antecedents and candidates that are not antecedents (negatives) (rows 4–5), the number of pronominal and nominal anaphors (rows 6–7). Both training sets, artificial and CSN, have only one possible antecedent for which we accept two minimal variants differing in only one word or one word and punctuation. On the contrary, both test sets by design allow annotation of more than one antecedent that differ in more than one word. Every anaphor in the artificial training dataset is pronominal, whereas anaphors in CSN and ASN are nominal only. ARRAU-AA has a mixture of nominal and pronominal anaphors.

Data pre-processing. Other details can be found in Supplementary Materials.

5.2 Baselines and evaluation metrics

Following KZH13, we report *success@n* (*s@n*), which measures whether the antecedent, or a candidate that differs in one word[14], is in the first n ranked candidates, for $n \in \{1, 2, 3, 4\}$. Additionally, we report the preceding sentence baseline (PS$_{BL}$) that chooses the previous sentence for the antecedent and *TAGbaseline* (TAG$_{BL}$) that randomly chooses a candidate with the constituent tag label in {S, VP, ROOT, SBAR}. For TAG$_{BL}$ we report the average of 10 runs with 10 fixed seeds. PS$_{BL}$ always performs worse than the KZH13 model on the ASN, so we report it only for ARRAU-AA.

5.3 Training details for our models

Hyperparameters tuning. We recorded performance with manually chosen HPs and then tuned HPs with Tree-structured Parzen Estimators (TPE) (Bergstra et al., 2011)[15]. TPE chooses HPs for the next (out of 10) trails on the basis of the $s@1$ score on the devset. As devsets we employ the ARRAU-AA corpus for shell noun resolution and the ASN corpus for unrestricted abstract anaphora resolution. For each trial we record performance on the test set. We report the best test $s@1$ score in 10 trials if it is better than the scores from default HPs. The default HPs and prior distributions for HPs used by TPE are given below. The (exact) HPs we used can be found in Supplementary Materials.

Input representation. To construct word vectors $\boldsymbol{w_i}$ as defined in Section 3, we used 100-dim. GloVe word embeddings pre-trained on the Gigaword and Wikipedia (Pennington et al., 2014), and did not fine-tune them. Vocabulary was built from the words in the training data with frequency in $\{3, \mathcal{U}(1, 10)\}$, and OOV words were replaced with an *UNK* token. Embeddings for tags are initialized with values drawn from the uniform distribution $\mathcal{U}\big(-\frac{1}{\sqrt{d+t}}, \frac{1}{\sqrt{d+t}}\big)$, where t is the number of tags[16] and $d \in \{50, \textit{qlog-}\mathcal{U}(30, 100)\}$ the size of the tag embeddings.[17] We experimented with removing embeddings for tag, anaphor and context.

Weights initialization. The size of the LSTMs hidden states was set to $\{100, \textit{qlog-}\mathcal{U}(30, 150)\}$. We initialized the weight matrices of the LSTMs with random orthogonal matrices (Henaff et al., 2016), all other weight matrices with the initialization proposed in He et al. (2015). The first feed-forward layer size is set to a value in $\{400, \textit{qlog-}\mathcal{U}(200, 800)\}$, the second to a value in $\{1024, \textit{qlog-}\mathcal{U}(400, 2000)\}$. Forget biases in the LSTM were initialized with 1s (Józefowicz et al., 2015), all other biases with 0s.

[14] We obtained this information in personal communication with one of the authors.

[15] https://github.com/hyperopt/hyperopt.

[16] We used a list of tags obtained from the Stanford Parser.

[17] *qlog-*$\mathcal{U}$ is the so-called qlog-uniform distribution.

		s @ 1	s @ 2	s @ 3	s @ 4
fact (train: 43809, test: 472)	MR-LSTM	**83.47**	**85.38**	**86.44**	**87.08**
	KZH13	70.00	86.00	92.00	95.00
	TAG$_{BL}$	46.99	-	-	-
reason (train: 4529, test: 442)	MR-LSTM	71.27	77.38	80.09	80.54
	+ tuning	**87.78**	**91.63**	**93.44**	**93.89**
	KZH13	72.00	86.90	90.00	94.00
	TAG$_{BL}$	42.40	-	-	-
issue (train: 2664, test: 303)	MR-LSTM	**88.12**	**91.09**	**93.07**	**93.40**
	KZH13	47.00	61.00	72.00	81.00
	TAG$_{BL}$	44.92	-	-	-
decision (train: 42289, test: 389)	MR-LSTM	**76.09**	**85.86**	**91.00**	**93.06**
	KZH13	35.00	53.00	67.00	76.00
	TAG$_{BL}$	45.55	-	-	-
question (train: 9327, test: 440)	MR-LSTM	**89.77**	**94.09**	**95.00**	**95.68**
	KZH13	70.00	83.00	88.00	91.00
	TAG$_{BL}$	42.02	-	-	-
possibility (train: 11874, test: 277)	MR-LSTM	**93.14**	**94.58**	**95.31**	**95.67**
	KZH13	56.00	76.00	87.00	92.00
	TAG$_{BL}$	48.66	-	-	-

Table 3: Shell noun resolution results.

Optimization. We trained our model in mini-batches using Adam (Kingma and Ba, 2015) with the learning rate of 10^{-4} and maximal batch size 64. We clip gradients by global norm (Pascanu et al., 2013), with a clipping value in $\{1.0, \mathcal{U}(1, 100)\}$. We train for 10 epochs and choose the model that performs best on the devset.

Regularization. We used the l_2-regularization with $\lambda \in \{10^{-5}, log\text{-}\mathcal{U}(10^{-7}, 10^{-2})\}$. Dropout (Srivastava et al., 2014) with a keep probability $k_p \in \{0.8, \mathcal{U}(0.5, 1.0)\}$ was applied to the outputs of the LSTMs, both feed-forward layers and optionally to the input with $k_p \in \mathcal{U}(0.8, 1.0)$.

6 Results and analysis

6.1 Results on shell noun resolution dataset

Table 3 provides the results of the mention-ranking model (MR-LSTM) on the ASN corpus using default HPs. Column 2 states which model produced the results: KZH13 refers to the best reported results in Kolhatkar et al. (2013b) and TAG$_{BL}$ is the baseline described in Section 5.2.

In terms of $s@1$ score, MR-LSTM outperforms both KZH13's results and TAG$_{BL}$ without even necessitating HP tuning. For the outlier *reason* we tuned HPs (on ARRAU-AA) for different variants of the architecture: the full architecture, without embedding of the context of the anaphor (ctx), of the anaphor (aa), of both constituent tag embedding and shortcut (tag,cut), dropping only the shortcut (cut), using only word embeddings as input (ctx,aa,tag,cut), without the first (ffl1) and second (ffl2) layer. From Table 4 we observe: (1) with HPs tuned on ARRAU-AA, we obtain results well beyond KZH13, (2) all ablated model variants perform worse than the full model, (3) a large performance drop when omitting syntactic information (tag,cut) suggests that the model makes good use of it. However, this could also be due to a bias in the tag distribution, given that all candidates stem from the single sentence that contains antecedents. The median occurrence of the S tag among both antecedents and negative candidates is 1, thus the model could achieve 50.00 $s@1$ by picking S-type constituents, just as TAG$_{BL}$ achieves 42.02 for *reason* and 48.66 for *possibility*.

						reason			
ctx	aa	tag	cut	ffl1	ffl2	s@1	s@2	s@ 3	s@ 4
✓	✓	✓	✓	✓	✓	**87.78**	**91.63**	**93.44**	**93.89**
✗	✓	✓	✓	✓	✓	85.97	87.56	89.14	89.82
✓	✗	✓	✓	✓	✓	86.65	88.91	91.18	91.40
✓	✓	✗	✗	✓	✓	68.10	80.32	85.29	89.37
✓	✓	✓	✗	✓	✓	85.52	88.24	89.59	90.05
✗	✗	✗	✗	✓	✓	66.97	80.54	85.75	88.24
✓	✓	✓	✓	✗	✓	87.56	91.63	92.76	94.12
✓	✓	✓	✓	✓	✗	85.97	88.69	89.14	90.05

Table 4: Architecture ablation for *reason*.

Tuning of HPs gives us insight into how different model variants cope with the task. For example, without tuning the model with and without syntactic information achieves 71.27 and 19.68 (not shown in table) $s@1$ score, respectively, and with tuning: 87.78 and 68.10. Performance of 68.10 $s@1$ score indicates that the model is able to learn without syntactic guidance, contrary to the 19.68 $s@1$ score before tuning.

6.2 Results on the ARRAU corpus

Table 5 shows the performance of different variants of the MR-LSTM with HPs tuned on the ASN corpus (always better than the default HPs), when evaluated on 3 different subparts of the ARRAU-AA: all 600 abstract anaphors, 397 nominal and 203 pronominal ones. HPs were tuned on the ASN corpus for every variant separately, without shuffling of the training data. For the best performing variant, without syntactic information (tag,cut), we report the results with HPs that yielded the best $s@1$ test score for all anaphors (row 4), when training with those HPs on shuffled training data (row 5), and with HPs that yielded the best $s@1$

						all (600)				nominal (397)				pronominal (203)			
ctx	aa	tag	cut	ffl1	ffl2	s@1	s@2	s@ 3	s@ 4	s@1	s@2	s@ 3	s@ 4	s@1	s@2	s@ 3	s@ 4
✓	✓	✓	✓	✓	✓	24.17	43.67	54.50	63.00	29.47	50.63	62.47	72.04	13.79	30.05	38.92	45.32
✗	✓	✓	✓	✓	✓	29.67	52.50	66.00	**75.00**	33.50	58.19	72.04	**80.86**	22.17	41.38	**54.19**	**63.55**
✓	✗	✓	✓	✓	✓	22.83	39.00	52.00	61.33	22.42	41.31	54.66	64.48	23.65	34.48	46.80	55.17
✓	✓	✗	✗	✓	✓	38.33	54.83	63.17	69.33	46.60	**64.48**	72.54	79.09	22.17	35.96	44.83	50.25
✓	✓	✗	✗	✓	✓	**43.83**	**56.33**	**66.33**	73.00	**51.89**	**64.48**	**73.55**	79.85	28.08	40.39	52.22	59.61
✓	✓	✗	✗	✓	✓	38.17	52.50	61.33	68.67	43.07	57.43	65.49	72.04	28.57	**42.86**	53.20	62.07
✓	✓	✓	✗	✓	✓	30.17	48.00	57.83	67.33	30.73	50.88	61.21	71.54	**29.06**	42.36	51.23	59.11
✗	✗	✗	✗	✓	✓	26.33	40.50	50.67	58.67	28.46	41.81	52.14	59.70	22.17	37.93	47.78	56.65
✓	✓	✓	✓	✗	✓	21.33	41.17	53.17	60.33	23.43	47.36	60.45	69.52	17.24	29.06	38.92	42.36
✓	✓	✓	✓	✓	✗	12.00	24.67	33.50	41.50	13.35	27.20	37.28	45.84	9.36	19.70	26.11	33.00
PS$_{BL}$						27.67	-	-	-	30.48	-	-	-	22.17	-	-	-
TAG$_{BL}$						38.43	-	-	-	40.10	-	-	-	**35.17**	-	-	-

Table 5: Results table for the ARRAU-AA test set. Refer to text for explanation of duplicated rows.

score for pronominal anaphors (row 6).

The MR-LSTM is more successful in resolving nominal than pronominal anaphors, although the training data provides only pronominal ones. This indicates that resolving pronominal abstract anaphora is harder compared to nominal abstract anaphora, such as shell nouns. Moreover, for shell noun resolution in KZH13's dataset, the MR-LSTM achieved s@1 scores in the range 76.09–93.14, while the best variant of the model achieves 51.89 s@1 score for nominal anaphors in ARRAU-AA. Although lower performance is expected, since we do not have specific training data for individual nominals in ARRAU-AA, we suspect that the reason for better performance for shell noun resolution in KZH13 is due to a larger number of positive candidates in ASN (cf. Table 2, rows: antecedents/negatives).

We also note that HPs that yield good performance for resolving nominal anaphors are not necessarily good for pronominal ones (cf. rows 4–6 in Table 5). Since the TPE tuner was tuned on the nominal-only ASN data, this suggest that it would be better to tune HPs for pronominal anaphors on a different dataset or stripping the nouns in ASN.

Contrary to shell noun resolution, omitting syntactic information boosts performance in ARRAU-AA. We conclude that when the model is provided with syntactic information, it learns to pick S-type candidates, but does not continue to learn deeper features to further distinguish them or needs more data to do so. Thus, the model is not able to point to exactly one antecedent, resulting in a lower s@1 score, but does well in picking a few good candidates, which yields good s@2-4 scores. This is what we can observe from row 2 vs. row 6 in Table 5: the MR-LSTM without context embedding (ctx) achieves a comparable s@2 score with the variant that omits syntactic information, but better s@3-4 scores. Further, median occurrence of tags not in {S, VP, ROOT, SBAR} among top-4 ranked candidates is 0 for the full architecture, and 1 when syntactic information is omitted. The need for discriminating capacity of the model is more emphasized in ARRAU-AA, given that the median occurrence of S-type candidates among negatives is 2 for nominal and even 3 for pronominal anaphors, whereas it is 1 for ASN. This is in line with the lower TAG$_{BL}$ in ARRAU-AA.

Finally, not all parts of the architecture contribute to system performance, contrary to what is observed for *reason*. For nominal anaphors, the anaphor (aa) and feed-forward layers (ffl1, ffl2) are beneficial, for pronominals only the second ffl.

6.3 Exploring the model

We finally analyze deeper aspects of the model: (1) whether a learned representation between the anaphoric *sentence* and an antecedent establishes a relation between *a specific anaphor we want to resolve* and the antecedent and (2) whether the max-margin objective enforces a separation of the joint representations in the shared space.

(1) We claim that by providing embeddings of both the anaphor and the sentence containing the anaphor we ensure that the learned relation between antecedent and anaphoric sentence is dependent on the anaphor under consideration. Fig. 3 illustrates the heatmap for an anaphoric sentence with two anaphors. The i-th column of the heatmap corresponds to absolute differences between the output of the bi-LSTM for the i-th word in the anaphoric sentence when the first vs. second anaphor is resolved. Stronger color indi-

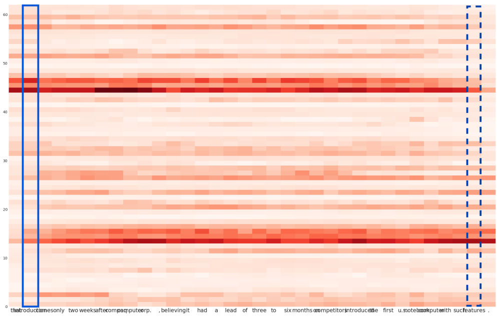


Figure 3: Visualizing the differences between outputs of the bi-LSTM over time for an anaphoric sentence containing two anaphors.

cates larger difference, the blue rectangle represents the column for the head of the first anaphor, the dashed blue rectangle the column for the head of the second anaphor. Clearly, the representations differ when the first vs. second anaphor is being resolved and consequently, joint representations with an antecedent will differ too.

(2) It is known that the max-margin objective separates the best-scoring positive candidate from the best-scoring negative candidate. To investigate what the objective accomplishes in the MR-LSTM model, we analyze the joint representations of candidates and the anaphoric sentence (i.e., outputs of ffl2) after training. For a randomly chosen instance from ARRAU-AA, we plotted outputs of ffl2 with the tSNE algorithm (v.d. Maaten and Hinton, 2008). Fig. 4 illustrates that the joint representation of the first ranked candidate and the anaphoric sentence is clearly separated from other joint representations. This shows that the max-margin objective separates the best scoring positive candidate from the best scoring negative candidate by separating their respective joint representations with the anaphoric sentence.

7 Conclusions

We presented a neural mention-ranking model for the resolution of unconstrained abstract anaphora, and applied it to two datasets with different types of abstract anaphora: the shell noun dataset and a subpart of ARRAU with (pro)nominal abstract anaphora of any type. To our knowledge this work is the first to address the unrestricted abstract anaphora resolution task with a neural network. Our model also outperforms state-of-the-art results on the shell noun dataset.

In this work we explored the use of purely artificially created training data and how far it can bring us. In future work, we plan to investigate mixtures of (more) artificial and natural data from different sources (e.g. ASN, CSN).

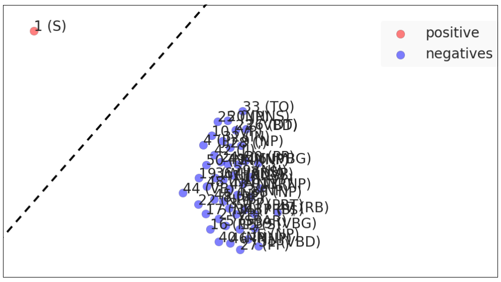


Figure 4: tSNE projection of outputs of ffl2. Labels are the predicted ranks and the constituent tag.

On the more challenging ARRAU-AA, we found model variants that surpass the baselines for the entire and the nominal part of ARRAU-AA, although we do not train models on individual (nominal) anaphor training data like the related work for shell noun resolution. However, our model still lags behind for pronominal anaphors. Our results suggest that models for nominal and pronominal anaphors should be learned independently, starting with tuning of HPs on a more suitable devset for pronominal anaphors.

We show that the model can exploit syntactic information to select plausible candidates, but that when it does so, it does not learn how to distinguish candidates of equal syntactic type. By contrast, if the model is not provided with syntactic information, it learns deeper features that enable it to pick the correct antecedent without narrowing down the choice of candidates. Thus, in order to improve performance, the model should be enforced to first select reasonable candidates and then continue to learn features to distinguish them, using a larger training set that is easy to provide.

In future work we will design such a model, and offer it candidates chosen not only from sentences containing the antecedent, but the larger context.

Acknowledgments

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1. We would like to thank anonymous reviewers for useful comments and especially thank Todor Mihaylov for the model implementations advices and everyone in the Computational Linguistics Group for helpful discussion.

References

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

Pranav Anand and Daniel Hardt. 2016. Antecedent selection for sluicing: Structure and content. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1243, Austin, Texas.

Nicholas Asher. 1993. *Reference to Abstract Objects in Discourse*. Kluwer Academic Publishers.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Proceedings of the 35th Annual Conference on Neural Information Processing Systems (NIPS)*, Granada, Spain.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1466–1477, Berlin, Germany.

Donna K. Byron. 2004. *Resolving pronominal reference to abstract entities*. Ph.D. thesis, University of Rochester, Rochester, New York.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL)*, Beijing, China.

Kevin Clark and Christopher D. Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Austin, Texas.

Kevin Clark and Christopher D. Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berling, Germany.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (elus). In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico.

Arpita Das, Harish Yenala, Manoj Kumar Chinnakotla, and Manish Shrivastava. 2016. Together we stand: Siamese networks for similar question retrieval. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berling, Germany.

Stefanie Dipper and Heike Zinsmeister. 2012. Annotating Abstract Anaphora. *Language Resources and Evaluation*, 46(1):37–52.

Miriam Eckert and Michael Strube. 2000. Dialogue acts, synchronising units and anaphora resolution.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise Phoneme Classification With Bidirectional LSTM And Other Neural Network Architectures. *Neural Networks*, 18:602–610.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*.

Mikael Henaff, Arthur Szlam, and Yann LeCun. 2016. Recurrent orthogonal networks and long-memory tasks. In *Proceedings of the the 33rd International Conference on Machine Learning (ICML)*, New York City, USA.

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

Sujay Kumar Jauhar, Raul Guerra, Edgar Gonzàlez Pellicer, and Marta Recasens. 2015. Resolving discourse-deictic pronouns: A two-stage approach to do it. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 299–308, Denver, Colorado.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142.

Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, USA.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Varada Kolhatkar and Graeme Hirst. 2014. Resolving shell nouns. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 499–510, Doha, Qatar.

Varada Kolhatkar, Heike Zinsmeister, and Graeme Hirst. 2013a. Annotating anaphoric shell nouns with their antecedents. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 112–121, Sofia, Bulgaria.

Varada Kolhatkar, Heike Zinsmeister, and Graeme Hirst. 2013b. Interpreting anaphoric shell nouns using antecedents of cataphoric shell nouns as training data. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 300–310, Seattle, Washington, USA.

Jing Lu and Vincent Ng. 2016. Event Coreference Resolution with Multi-Pass Sieves. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*, pages 3996–4003, Portoroz.

Laurens v.d. Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the 13th Conference on Artificial Intelligence (AAAI)*, pages 2786–2792, Phoenix, Arizona.

Christoph Müller. 2008. *Fully Automatic Resolution of* It, This *and* That *in Unrestricted Multi-Party Dialog*. Ph.D. thesis, Universität Tübingen, Tübingen.

Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning Text Similarity with Siamese Recurrent Networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157, Berlin, Germany.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, Atlanta, USA.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Massimo Poesio and Ron Artstein. 2008. Anaphoric Annotation in the ARRAU Corpus. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Dheeraj Rajagopal, Eduard Hovy, and Teruko Mitamura. 2016. Unsupervised event coreference for abstract words. In *Proceedings of EMNLP 2016 Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*, pages 22–26, Austin, Texas.

Attapol T. Rutherford, Vera Demberg, and Nianwen Xue. 2017. A Systematic Study of Neural Discourse Models for Implicit Discourse Relation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Manfred Stede and Yulia Grishina. 2016. Anaphoricity in Connectives: A Case Study on German. In *Proceedings of the Coreference Resolution Beyond OntoNotes (CORBON) Workshop*, San Diego, California.

Michael Strube and Christoph Müller. 2003. A machine learning approach to pronoun resolution in spoken dialogue. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 168–175, Sapporo, Japan.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL)*, Beijing, China.

Olga Uryupina, Ron Artstein, Antonella Bristot, Federica Cavicchio, Kepa J Rodriguez, and Massimo Poesio. 2016. ARRAU: Linguistically-Motivated Annotation of Anaphoric Descriptions. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*, pages 2058–2062, Portoroz.

Bonnie Lynn Webber. 1991. Structure and ostension in the interpretation of discourse deixis. *Language and Cognitive processes*, 6(2):107–135.

Sam Joshua Wiseman, Alexander Matthew Rush, Stuart Merrill Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL)*, Beijing, China.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1127–1137, Beijing, China.

Hierarchical Embeddings for Hypernymy Detection and Directionality

Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, Ngoc Thang Vu
Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
Pfaffenwaldring 5B, 70569 Stuttgart, Germany
{nguyenkh,koepermn,schulte,thangvu}@ims.uni-stuttgart.de

Abstract

We present a novel neural model *HyperVec* to learn hierarchical embeddings for hypernymy detection and directionality. While previous embeddings have shown limitations on prototypical hypernyms, *HyperVec* represents an unsupervised measure where embeddings are learned in a specific order and capture the hypernym–hyponym distributional hierarchy. Moreover, our model is able to generalize over unseen hypernymy pairs, when using only small sets of training data, and by mapping to other languages. Results on benchmark datasets show that *HyperVec* outperforms both state-of-the-art unsupervised measures and embedding models on hypernymy detection and directionality, and on predicting graded lexical entailment.

1 Introduction

Hypernymy represents a major semantic relation and a key organization principle of semantic memory (Miller and Fellbaum, 1991; Murphy, 2002). It is an asymmetric relation between two terms, a hypernym (superordinate) and a hyponym (subordiate), as in *animal–bird* and *flower–rose*, where the hyponym necessarily implies the hypernym, but not vice versa. From a computational point of view, automatic hypernymy detection is useful for NLP tasks such as taxonomy creation (Snow et al., 2006; Navigli et al., 2011), recognizing textual entailment (Dagan et al., 2013), and text generation (Biran and McKeown, 2013), among many others.

Two families of approaches to identify and discriminate hypernyms are predominent in NLP, both of them relying on word vector representations. ***Distributional count approaches*** make use of either directionally unsupervised measures or of supervised classification methods. Unsupervised measures exploit the *distributional inclusion hypothesis* (Geffet and Dagan, 2005; Zhitomirsky-Geffet and Dagan, 2009), or the *distributional informativeness hypothesis* (Santus et al., 2014; Rimell, 2014). These measures assign scores to semantic relation pairs, and hypernymy scores are expected to be higher than those of other relation pairs. Typically, Average Precision (AP) (Kotlerman et al., 2010) is applied to rank and distinguish between the predicted relations. Supervised classification methods represent each pair of words as a single vector, by using the concatenation or the element-wise difference of their vectors (Baroni et al., 2012; Roller et al., 2014; Weeds et al., 2014). The resulting vector is fed into a Support Vector Machine (SVM) or into Logistic Regression (LR), to predict hypernymy. Across approaches, Shwartz et al. (2017) demonstrated that there is no single unsupervised measure which consistently deals well with discriminating hypernymy from other semantic relations. Furthermore, Levy et al. (2015) showed that supervised methods memorize *prototypical hypernyms* instead of *learning* a relation between two words.

Approaches of hypernymy-specific embeddings utilize neural models to learn vector representations for hypernymy. Yu et al. (2015) proposed a supervised method to learn term embeddings for hypernymy identification, based on pre-extracted hypernymy pairs. Recently, Tuan et al. (2016) proposed a dynamic weighting neural model to learn term embeddings in which the model encodes not only the information of hypernyms vs. hyponyms, but also their contextual information. The performance of this family of models is typically evaluated by using an SVM to discriminate hypernymy from other relations.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 233–243
Copenhagen, Denmark, September 7–11, 2017.

In this paper, we propose a novel neural model *HyperVec* to learn hierarchical embeddings that **(i)** discriminate hypernymy from other relations (**detection task**), and **(ii)** distinguish between the hypernym and the hyponym in a given hypernymy relation pair (**directionality task**). Our model learns to strengthen the distributional similarity of hypernym pairs in comparison to other relation pairs, by moving hyponym and hypernym vectors close to each other. In addition, we generate a distributional hierarchy between hyponyms and hypernyms. Relying on these two new aspects of hypernymy distributions, the similarity of hypernym pairs receives higher scores than the similarity of other relation pairs; and the distributional hierarchy of hyponyms and hypernyms indicates the directionality of hypernymy.

Our model is inspired by the *distributional inclusion hypothesis*, that prominent context words of hyponyms are expected to appear in a subset of the hypernym contexts. We assume that each context word which appears with both a hyponym and its hypernym can be used as an indicator to determine which of the two words is semantically more general: Common context word vectors which represent distinctive characteristics of a hyponym are expected to be closer to the hyponym vector than to its hypernym vector. For example, the context word *flap* is more characteristic for a *bird* than for its hypernym *animal*; hence, the vector of *flap* should be closer to the vector of *bird* than to the vector of *animal*.

We evaluate our *HyperVec* model on both unsupervised and supervised hypernymy detection and directionality tasks. In addition, we apply the model to the task of graded lexical entailment (Vulić et al., 2016), and we assess the capability of *HyperVec* on generalizing hypernymy by mapping to German and Italian. Results on benchmark datasets of hypernymy show that the hierarchical embeddings outperform state-of-the-art measures and previous embedding models. Furthermore, the implementation of our models is made publicly available.[1]

2 Related Work

Unsupervised hypernymy measures: A variety of directional measures for unsupervised hypernymy detection (Weeds and Weir, 2003; Weeds et al., 2004; Clarke, 2009; Kotlerman et al., 2010; Lenci and Benotto, 2012) all rely on some variation of the *distributional inclusion hypothesis*: If u is a semantically narrower term than v, then a significant number of salient distributional features of u is expected to be included in the feature vector of v as well. In addition, Santus et al. (2014) proposed the *distributional informativeness hypothesis*, that hypernyms tend to be less informative than hyponyms, and that they occur in more general contexts than their hyponyms. All of these approaches represent words as vectors in distributional semantic models (Turney and Pantel, 2010), relying on the *distributional hypothesis* (Harris, 1954; Firth, 1957). For evaluation, these directional models use the AP measure to assess the proportion of hypernyms at the top of a score-sorted list. In a different vein, Kiela et al. (2015) introduced three unsupervised methods drawn from visual properties of images to determine a concept's generality in hypernymy tasks.

Supervised hypernymy methods: The studies in this area are based on word embeddings which represent words as low-dimensional and real-valued vectors (Mikolov et al., 2013b; Pennington et al., 2014). Each hypernymy pair is encoded by some combination of the two word vectors, such as concatenation (Baroni et al., 2012) or difference (Roller et al., 2014; Weeds et al., 2014). Hypernymy is distinguished from other relations by using a classification approach, such as SVM or LR. Because word embeddings are trained for similar and symmetric vectors, it is however unclear whether the supervised methods do actually learn the asymmetry in hypernymy (Levy et al., 2015).

Hypernymy-specific embeddings: These approaches are closest to our work. Yu et al. (2015) proposed a dynamic distance-margin model to learn term embeddings that capture properties of hypernymy. The neural model is trained on the taxonomic relation data which is pre-extracted. The resulting term embeddings are fed to an SVM classifier to predict hypernymy. However, this model only learns term pairs without considering their contexts, leading to a lack of generalization for term embeddings. Tuan et al. (2016) introduced a dynamic weighting neural network to learn term embeddings that encode information about hypernymy and also about their contexts, considering all words between a hypernym and its

[1] www.ims.uni-stuttgart.de/data/hypervec

hyponym in a sentence. The proposed model is trained on a set of hypernym relations extracted from WordNet (Miller, 1995). The embeddings are applied as features to detect hypernymy, using an SVM classifier. Tuan et al. (2016) handles the drawback of the approach by Yu et al. (2015), considering the contextual information between two terms; however the method still is not able to determine the directionality of a hypernym pair. Vendrov et al. (2016) proposed a method to encode order into learned distributed representations, to explicitly model partial order structure of the visual-semantic hierarchy or the hierarchy of hypernymy in WordNet. The resulting vectors are used to predict the transitive hypernym relations in WordNet.

3 Hierarchical Embeddings

In this section, we present our model of hierarchical embeddings *HyperVec*. Section 3.1 describes how we learn the embeddings for hypernymy, and Section 3.2 introduces the unsupervised measure *HyperScore* that is applied to the hypernymy tasks.

3.1 Learning Hierarchical Embeddings

Our approach makes use of a set of hypernyms which could be obtained from either exploiting the transitivity of the hypernymy relation (Fallucchi and Zanzotto, 2011) or lexical databases, to learn hierarchical embeddings. We rely on WordNet, a large lexical database of English (Fellbaum, 1998), and extract all hypernym–hyponym pairs for nouns and for verbs, including both direct and indirect hypernymy, e.g., *animal–bird, bird–robin, animal–robin*. Before training our model, we exclude all hypernym pairs which appear in any datasets used for evaluation.

In the following, Section 3.1.1 first describes the Skip-gram model which is integrated into our model for optimization. Section 3.1.2 then describes the objective functions to train the hierarchical embeddings for hypernymy.

3.1.1 Skip-gram Model

The Skip-gram model is a word embeddings method suggested by Mikolov et al. (2013b). Levy and Goldberg (2014) introduced a variant of the Skip-gram model with negative sampling (SGNS), in which the objective function is defined as follows:

$$\mathrm{J}_{SGNS} = \sum_{w \in V_W} \sum_{c \in V_C} \mathrm{J}_{(w,c)} \tag{1}$$

$$\begin{aligned} \mathrm{J}_{(w,c)} &= \#(w,c) \log \sigma(\vec{w}, \vec{c}) \\ &+ k \cdot \mathbb{E}_{c_N \sim P_D}[\log \sigma(-\vec{w}, \vec{c}_N)] \end{aligned} \tag{2}$$

where the skip-gram with negative sampling is trained on a corpus of words $w \in V_W$ and their contexts $c \in V_C$, with V_W and V_C the word and context vocabularies, respectively. The collection of observed words and context pairs is denoted as D; the term $\#(w,c)$ refers to the number of times the pair (w,c) appeared in D; the term $\sigma(x)$ is the sigmoid function; the term k is the number of negative samples and the term c_N is the sampled context, drawn according to the empirical unigram distribution P.

3.1.2 Hierarchical Hypernymy Model

Vector representations for detecting hypernymy are usually encoded by standard first-order distributional co-occurrences. In this way, they are insufficient to differentiate hypernymy from other paradigmatic relations such as synonymy, meronymy, antonymy, etc. Incorporating directional measures of hypernymy to detect hypernymy by exploiting the common contexts of hypernym and hyponym improves this relation distinction, but still suffers from distinguishing between hypernymy and meronymy.

Our novel approach presents two solutions to deal with these challenges. First of all, the embeddings are learned in a specific order, such that the similarity score for hypernymy is higher than the similarity score for other relations. For example, the hypernym pair *animal–frog* will be assigned a higher cosine score than the co-hyponymy pair *eagle–frog*. Secondly, the embeddings are learned to capture the distributional hierarchy between hyponym and hypernym, as an indicator to differentiate between hypernym and hyponym. For example, given a hyponym–hypernym pair (p, q), we can exploit the Euclidean norms of $\vec{q}$ and $\vec{p}$ to differentiate between the two words, such that the Euclidean norm of the hypernym $\vec{q}$ is larger than the Euclidean norm of the hyponym $\vec{p}$.

Inspired by the distributional lexical contrast model in Nguyen et al. (2016) for distinguishing antonymy from synonymy, this paper proposes two objective functions to learn hierarchical embeddings for hypernymy. Before moving to the details of the two objective functions, we first define the terms as follows: $\mathbb{W}(c)$ refers to the set of words co-occurring with the context c in a certain window-size; $\mathbb{H}(w)$ denotes the set of hypernyms for the word w; the two terms $\mathbb{H}^{+}(w,c)$ and $\mathbb{H}^{-}(w,c)$ are drawn from $\mathbb{H}(w)$, and are defined as follows:

$$\mathbb{H}^{+}(w,c) = \{u \in \mathbb{W}(c) \cap \mathbb{H}(w) : cos(\vec{w},\vec{c}) - cos(\vec{u},\vec{c}) \geq \theta\}$$
$$\mathbb{H}^{-}(w,c) = \{v \in \mathbb{W}(c) \cap \mathbb{H}(w) : cos(\vec{w},\vec{c}) - cos(\vec{v},\vec{c}) < \theta\}$$

where $cos(\vec{x},\vec{y})$ stands for the cosine similarity of the two vectors $\vec{x}$ and $\vec{y}$; θ is the margin. The set $\mathbb{H}^{+}(w,c)$ contains all hypernyms of the word w that share the context c and satisfy the constraint that the cosine similarity of pair (w,c) is higher than the cosine similarity of pair (u,c) within a max-margin framework θ. Similarly, the set $\mathbb{H}^{-}(w,c)$ represents all hypernyms of the word w with respect to the common context c in which the cosine similarity difference between the pair (w,c) and the pair (v,c) is within a min-margin framework θ. The two objective functions are defined as follows:

$$\mathrm{L}_{(w,c)} = \frac{1}{\#(w,u)} \sum\nolimits_{u \in \mathbb{H}^{+}(w,c)} \partial(\vec{w},\vec{u}) \quad (3)$$
$$\mathrm{L}_{(v,w,c)} = \sum\nolimits_{v \in \mathbb{H}^{-}(w,c)} \partial(\vec{v},\vec{w}) \quad (4)$$

where the term $\partial(\vec{x},\vec{y})$ stands for the cosine derivative of $(\vec{x},\vec{y})$; and ∂ then is optimized by the negative sampling procedure.

The objective function in Equation 3 minimizes the distributional difference between the hyponym w and the hypernym u by exploiting the common context c. More specifically, if the common context c is the distinctive characteristic of the hyponym w (i.e. the common context c is closer to the hyponym w than to the hypernym u), the objective function $\mathrm{L}_{(w,c)}$ tries to decrease the distributional generality of hypernym u by moving w closer to u. For example, given a hypernym-hyponym pair *animal–bird*, the context *flap* is a distinctive characteristic of *bird*, because almost every *bird* can flap, but not every *animal* can flap. Therefore, the context *flap* is closer to the hyponym *bird* than to the hypernym *animal*. The model then tries to move *bird* closer to *animal* in order to enforce the similarity between *bird* and *animal*, and to decrease the distributional generality of *animal*.

In contrast to Equation 3, the objective function in Equation 4 minimizes the distributional difference between the hyponym w and the hypernym v by exploiting the common context c, which is a distinctive characteristic of the hypernym v. In this case, the objective function $\mathrm{L}_{(v,w,c)}$ tries to reduce the distributional generality of hyponym w by moving v closer to w. For example, the context word *rights*, a distinctive characteristic of the hypernym *animal*, should be closer to *animal* than to *bird*. Hence, the model tries to move the hypernym *animal* closer to the hyponym *bird*. Given that hypernymy is an asymmetric and also a hierarchical relation, where each hypernym may contain several hyponyms, our objective functions updates simultaneously both the hypernym and all of its hyponyms; therefore, our objective functions are able to capture the hierarchical relations between the hypernym and its hyponyms. Moreover, in our model, the margin framework θ plays a role in learning the hierarchy of hypernymy, and in preventing the model from minimizing the distance of synonymy or antonymy, because synonymy and antonymy share many contexts.

In the final step, the objective function which is used to learn the hierarchical embeddings for hypernymy combines Equations 1, 2, 3, and 4 by the objective function in Equations 5 and 6:

$$\mathrm{J}_{(w,v,c)} = \mathrm{J}_{(w,c)} + \mathrm{L}_{(w,c)} + \mathrm{L}_{(v,w,c)} \quad (5)$$
$$\mathrm{J} = \sum_{w \in V_W} \sum_{c \in V_C} \mathrm{J}_{(w,v,c)} \quad (6)$$

3.2 Unsupervised Hypernymy Measure

HyperVec is expected to show the two following properties: (i) the hyponym and the hypernym are close to each other, and (ii) there exists a distributional hierarchy between hypernyms and their hyponyms. Given a hypernymy pair (u,v) in which u is the hyponym and v is the hypernym, we propose a measure to detect hypernymy and to determine the directionality of hypernymy by using the hierarchical embeddings as follows:

$$HyperScore(u,v) = cos(\vec{u},\vec{v}) * \frac{\|\vec{v}\|}{\|\vec{u}\|} \quad (7)$$

where $cos(\vec{u},\vec{v})$ is the cosine similarity between $\vec{u}$ and $\vec{v}$, and $\|\cdot\|$ is the magnitude of the vector (or the Euclidean norm). The cosine similarity is applied to distinguish hypernymy from other relations, due to the first property of the hierarchical embeddings, while the second property is used to decide about the directionality of hypernymy, assuming that the magnitude of the hypernym is larger than the magnitude of the hyponym. Note that the proposed hypernymy measure is unsupervised when the resource is only used to learn hierarchical embeddings.

4 Experiments

In this section, we first describe the experimental settings in our experiments (Section 4.1). We then evaluate the performance of *HyperVec* on three different tasks: i) unsupervised hypernymy detection and directionality (Section 4.2), where we assess *HyperVec* on ranking and classifying hypernymy; ii) supervised hypernymy detection (Section 4.3), where we apply supervised classification to detect hypernymy; iii) graded lexical entailment (Section 4.4), where we predict the strength of hypernymy pairs.

4.1 Experimental Settings

We use the ENCOW14A corpus (Schäfer and Bildhauer, 2012; Schäfer, 2015) with approx. 14.5 billion tokens for training the hierarchical embeddings and the default SGNS model. We train our model with 100 dimensions, a window size of 5, 15 negative samples, and 0.025 as the learning rate. The threshold θ is set to 0.05. The hypernymy resource for nouns comprises $105,020$ hyponyms, $24,925$ hypernyms, and $1,878,484$ hyponym–hypernym pairs. The hypernymy resource for verbs consists of $11,328$ hyponyms, $4,848$ hypernyms, and $130,350$ hyponym–hypernym pairs.

4.2 Unsupervised Hypernymy Detection and Directionality

In this section, we assess our model on two experimental setups: i) a ranking retrieval setup that expects hypernymy pairs to have a higher similarity score than instances from other semantic relations; ii) a classification setup that requires both hypernymy detection and directionality.

4.2.1 Ranking Retrieval

Shwartz et al. (2017) conducted an extensive evaluation of a large number of unsupervised distributional measures for hypernymy ranking retrieval proposed in previous work (Weeds and Weir, 2003; Santus et al., 2014; Clarke, 2009; Kotlerman et al., 2010; Lenci and Benotto, 2012; Santus et al., 2016). The evaluation was performed on four semantic relation datasets: **BLESS** (Baroni and Lenci, 2011), **WEEDS** (Weeds et al., 2004), **EVALUTION** (Santus et al., 2015), and **LENCI&BENOTTO** (Benotto, 2015). Table 1 describes the detail of these datasets in terms of the semantic relations and the number of instances. The Average Precision (AP) ranking measure is used to evaluate the performance of the measures.

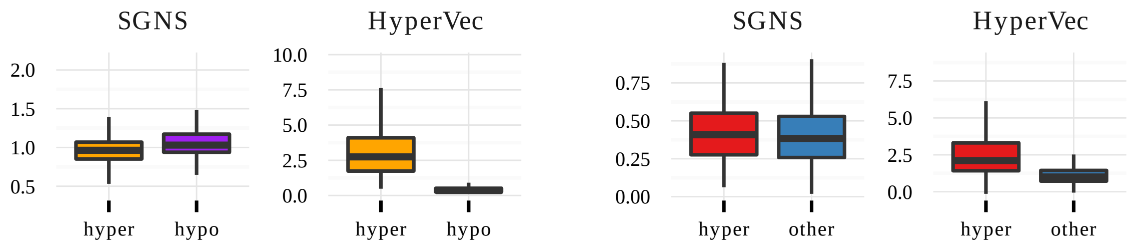
In comparison to the state-of-the-art unsupervised measures compared by Shwartz et al. (2017) (henceforth, baseline models), we apply our unsupervised measure *HyperScore* (Equation 7) to rank hypernymy against other relations. Table 2

Dataset	Relation	#Instance	Total
BLESS	hypernymy	1,337	26,554
	meronymy	2,943	
	coordination	3,565	
	event	3,824	
	attribute	2,731	
	random-n	6,702	
	random-j	2,187	
	random-v	3,265	
EVALution	hypernymy	3,637	13,465
	meronymy	1,819	
	attribute	2,965	
	synonymy	1,888	
	antonymy	3,156	
Lenci&Benotto	hypernymy	1,933	5,010
	synonymy	1,311	
	antonymy	1,766	
Weeds	hypernymy	1,469	2,928
	coordination	1,459	

Table 1: Details of the semantic relations and the number of instances in each dataset.

Dataset	Hypernymy vs.	Baseline	HyperScore
EVALution	other relations	0.353	**0.538**
	meronymy	0.675	**0.811**
	attribute	0.651	**0.800**
	antonymy	0.55	**0.743**
	synonymy	0.657	**0.793**
BLESS	other relations	0.051	**0.454**
	meronymy	0.76	**0.913**
	coordination	0.537	**0.888**
	attribute	0.74	**0.918**
	event	**0.779**	0.620
Lenci&Benotto	other relations	0.382	**0.574**
	antonymy	0.624	**0.696**
	synonymy	0.725	**0.751**
Weeds	coordination	0.441	**0.850**

Table 2: AP results of *HyperScore* in comparison to state-of-the-art measures.



(a) Directionality task: hypernym vs. hyponym.

(b) Hypernymy detection: hypernymy vs. other relations.

Figure 1: Comparing *SGNS* and *HyperVec* on binary classification tasks. The y-axis shows the magnitude values of the vectors.

presents the results of using *HyperScore* vs. the best baseline models, across datasets. When detecting hypernymy among all other relations (which is the most challenging task), *HyperScore* significantly outperforms all baseline variants on all datasets. The strongest difference is reached on the BLESS dataset, where *HyperScore* achieves an improvement of 40% AP score over the best baseline model. When ranking hypernymy in comparison to a single other relation, *HyperScore* also improves over the baseline models, except for the *event* relation in the BLESS dataset. We assume that this is due to the different parts-of-speech (adjective and noun) involved in the relation, where *HyperVec* fails to establish a hierarchy.

4.2.2 Classification

In this setup, we rely on three datasets of semantic relations, which were all used in various state-of-the-art approaches before, and brought together for hypernymy evaluation by Kiela et al. (2015). **(i)** A subset of **BLESS** contains 1,337 hyponym-hypernym pairs. The task is to predict the directionality of hypernymy within a binary classification. Our approach requires no threshold; we only need to compare the magnitudes of the two words and to assign the hypernym label to the word with the larger magnitude. Figure 1a indicates that the magnitude values of the *SGNS* model cannot distinguish between a hyponym and a hypernym, while the hierarchical embeddings provide a larger magnitude for the hypernym. **(ii)** Following Weeds et al. (2014), we conduct a binary classification with a subset of 1,168 BLESS word pairs. In this dataset (**WBLESS**), one class is represented by hyponym–hypernym pairs, and the other class is a combination of reversed hypernym–hyponym pairs, plus additional holonym-meronym pairs, co-hyponyms and randomly matched nouns. For this classification we make use of our *HyperScore* measure that ranks hypernymy pairs higher than other relation pairs. A threshold decides about the splitting point between the two classes: *hyper* vs. *other*. Instead of using a manually defined threshold as done by Kiela et al. (2015), we decided to run 1 000 iterations which randomly sampled only 2% of the available pairs for learning a threshold, using the remaining 98% for test purposes. We present average accuracy results across all iterations. Figure 1b compares the default cosine similarities between the relation pairs (as applied by *SGNS*) and *HyperScore* (as applied by *HyperVec*) on this task. Using *HyperScore*, the class “hyper” can clearly be distinguished from the class “other”. **(iii) BIBLESS** represents the most challenging dataset; the relation pairs from WBLESS are split into three classes instead of two: hypernymy pairs, reversed hypernymy pairs, and other relation pairs. In this case, we perform a three-way classification. We apply the same technique as used for the WBLESS classification, but in cases where we classify *hyper* we additionally classify the hypernymy direction, to decide between hyponym–hypernym pairs and reversed hypernym–hyponym pairs.

	BLESS	WBLESS	BIBLESS
Kiela et al. (2015)	0.88	0.75	0.57
Santus et al. (2014)	0.87	—	—
Weeds et al. (2014)	—	0.75	—
SGNS	0.44	0.48	0.34
HyperVec	**0.92**	**0.87**	**0.81**

Table 3: Accuracy for hypernymy directionality.

Table 3 compares our results against related work. *HyperVec* outperforms all other methods on all three tasks. In addition we see again that an unmodified *SGNS* model cannot solve any of the three tasks.

4.3 Supervised Hypernymy Detection

For supervised hypernymy detection, we make use of the two datasets: the full **BLESS** dataset, and **ENTAILMENT** (Baroni et al., 2012), containing 2,770 relation pairs in total, including 1,385 hypernym pairs and 1,385 other relations pairs. We follow the same procedure as Yu et al. (2015) and Tuan et al. (2016) to assess *HyperVec* on the two datasets. Regarding BLESS, we extract pairs for four types of relations: hypernymy, meronymy, co-hyponymy (or *coordination*), and add the random relation for nouns. For the evaluation, we randomly select one concept and its relatum for testing, and train the supervised model on the 199 remaining concepts and its relatum. We then report the average accuracy across all concepts. For the ENTAILMENT dataset, we randomly select one hypernym pair for testing and train on all remaining hypernym pairs. Again, we report the average accuracy across all hypernyms.

We apply an SVM classifier to detect hypernymy based on *HyperVec*. Given a hyponym–hypernym pair (u, v), we concatenate four components to construct the vector for a pair (u, v) as follows: the vector difference between hypernym and hyponym ($\vec{v} - \vec{u}$); the cosine similarity between the hypernym and hyponym vectors ($cos(\vec{u}, \vec{v})$); the magnitude of the hyponym ($\|\vec{u}\|$); and the magnitude of the hypernym ($\|\vec{v}\|$). The resulting vector is fed into the SVM classifier to detect hypernymy. Similar to the two previous works, we train the SVM classifier with the RBF kernel, $\lambda = 0.03125$, and the penalty $C = 8.0$.

Table 4 shows the performance of *HyperVec* and the two baseline models reported by Tuan et al. (2016). *HyperVec* slightly outperforms the method of Tuan et al. (2016) on the BLESS dataset, and is equivalent to the performance of their method on the ENTAILMENT dataset. In comparison to the method of Yu et al. (2015), *HyperVec* achieves significant improvements.

Models	BLESS	ENTAILMENT
Yu et al. (2015)	0.90	0.87
Tuan et al. (2016)	0.93	0.91
HyperVec	**0.94**	0.91

Table 4: Classification results for BLESS and ENTAILMENT in terms of accuracy.

4.4 Graded Lexical Entailment

In this experiment, we apply *HyperVec* to the dataset of graded lexical entailment, *HyperLex*, as introduced by Vulić et al. (2016). The *HyperLex* dataset provides soft lexical entailment on a continuous scale, rather than simplifying into a binary decision. *HyperLex* contains 2,616 word pairs across seven semantic relations and two word classes (nouns and verbs). Each word pair is rated by a score that indicates the strength of the semantic relation between the two words. For example, the score of the hypernym pair *duck–animal* is 5.9 out of 6.0, while the score of the reversed pair *animal–duck* is only 1.0.

We compared *HyperScore* against the most prominent state-of-the-art hypernymy and lexical entailment models from previous work:

- Directional entailment measures (DEM) (Weeds and Weir, 2003; Weeds et al., 2004; Clarke, 2009; Kotlerman et al., 2010; Lenci and Benotto, 2012)
- Generality measures (SQLS) (Santus et al., 2014)
- Visual generality measures (VIS) (Kiela et al., 2015)
- Consideration of concept frequency ratio (FR) (Vulić et al., 2016)
- WordNet-based similarity measures (WN) (Wu and Palmer, 1994; Pedersen et al., 2004)
- Order embeddings (OrderEmb) (Vendrov et al., 2016)
- Skip-gram embeddings (SGNS) (Mikolov et al., 2013b; Levy and Goldberg, 2014)
- Embeddings fine-tuned to a paraphrase database with linguistic constraints (PARAGRAM) (Mrkšić et al., 2016)
- Gaussian embeddings (Word2Gauss) (Vilnis and McCallum, 2015)

The performance of the models is assessed through Spearman's rank-order correlation coefficient ρ (Siegel and Castellan, 1988), comparing the ranks of the models' scores and the human judgments for the given word pairs.

Measures		Embeddings	
Model	ρ	**Model**	ρ
FR	0.279	SGNS	0.205
DEM	0.180	PARAGRAM	0.320
SLQS	0.228	OrderEmb	0.191
WN	0.234	Word2Gauss	0.206
VIS	0.209	*HyperScore*	**0.540**

Table 5: Results (ρ) of *HyperScore* and state-of-the-art measures and word embedding models on graded lexical entailment.

Table 5 shows that *HyperScore* significantly outperforms both state-of-the-art measures and word embedding models. *HyperScore* outperforms even the previously best word embedding model PARAGRAM by .22, and the previously best measures FR by .27. The reason that *HyperVec* outperforms all other models is that the hierarchy between hypernym and hypornym within *HyperVec* differentiates hyponym–hypernym pairs from hypernym–hyponym pairs. For example, the *HyperScore* for the pairs *duck–animal* and *animal–duck* are 3.02 and 0.30, respectively. Thus, the magnitude proportion of the hypernym–hyponym pair *duck–animal* is larger than that for the pair *animal–duck*.

5 Generalizing Hypernymy

Having demonstrated the general abilities of *HyperVec*, this final section explores its potential for generalization in two different ways, (i) by relying on a small seed set only, rather than using a large set of training data; and (ii) by projecting *HyperVec* to other languages.

Hypernymy Seed Generalization: We utilize only a small hypernym set from the hypernymy resource to train *HyperVec*, relying on 200 concepts from the BLESS dataset. The motivation behind using these concepts is threefold: i) these concepts are distinct and unambiguous noun concepts; ii) the concepts were equally divided between living and non-living entities; iii) concepts have been grouped into 17 broader classes. Based on the seed set, we collected the hyponyms of each concept from WordNet, and then re-trained *HyperVec*. On the hypernymy ranking retrieval task (Section 4.2.1), *HyperScore* outperforms the baselines across all datasets (cf. Table 1) with AP values of 0.39, 0.448, and 0.585 for EVALution, LenciBenotto, and Weeds, respectively. For the graded lexical entailment task (Section 4.4), *HyperScore* obtains a correlation of $\rho = 0.30$, outperforming all models except for PARAGRAM with $\rho = 0.32$. Overall, the results show that *HyperVec* is indeed able to generalize hypernymy from small seeds of training data.

Generalizing Hypernymy across Languages: We assume that hypernymy detection can be improved across languages by projecting representations from any arbitrary language into our modified English *HyperVec* space. We conduct experiments for German and Italian, where the language-specific representations are obtained using the same hyper-parameter settings as for our English *SGNS* model (cf. Section 4.1). As corpus resource we relied on Wikipedia dumps[2]. Note that we do not use any additional resource, such as the German or Italian WordNet, to tune the embeddings for hypernymy detection. Based on the representations, a mapping function between a source language (German, Italian) and our English *HyperVec* space is learned, by relying on the least-squares error method from previous work using cross-lingual data (Mikolov et al., 2013a) and different modalities (Lazaridou et al., 2015).

To learn a mapping function between two languages, a one-to-one correspondence (word translations) between two sets of vectors is required. We obtained these translations by using the parallel Europarl[3] V7 corpus for German–English and Italian–English. Word alignment counts were extracted using *fast_align* (Dyer et al., 2013). We then assigned each source word to the English word with the maximum number of alignments in the parallel corpus. We could match 25,547 pairs for DE→EN and 47,475 pairs for IT→EN.

Taking the aligned subset of both spaces, we assume that X is the matrix obtained by concatenating all source vectors, and likewise Y is the matrix obtained by concatenating all corresponding English elements. Applying the $\ell 2$-regularized least-squares error objective can be described using the following equation:

$$\hat{\mathbf{W}} = \underset{\mathbf{W} \in \mathbb{R}^{d1 \times d2}}{\operatorname{argmin}} \|\mathbf{XW} - \mathbf{Y}\| + \lambda \|\mathbf{W}\| \qquad (8)$$

Although we learn the mapping only on a subset of aligned words, it allows us to project every word in

[2] The Wikipedia dump for German and Italian were both downloaded in January 2017.

[3] http://www.statmt.org/europarl/

a source vocabulary to its English *HyperVec* position by using **W**.

Finally we compare the original representations and the mapped representation on the hypernymy ranking retrieval task (similar to Section 4.2.1). As gold resources we relied on German and Italian nouns pairs. For German we used the 282 German pairs collected via Amazon Mechanical Turk by Scheible and Schulte im Walde (2014). The 1,350 Italian pairs were collected via Crowdflower by Sucameli (2015) in the same way. Both collections contain hypernymy, antonymy and synonymy pairs. As before, we evaluate the ranking by AP, and we compare the cosine of the unmodified default representations against the *HyperScore* of the projected representations.

German	Hyp/All	Hyp/Syn	Hyp/Ant
DE-*SGNS*	0.28	0.48	0.40
DE→EN*HyperVec*	**0.37**	**0.65**	**0.47**
Italian			
IT-*SGNS*	0.38	0.50	0.60
IT→EN*HyperVec*	**0.44**	**0.57**	**0.65**

Table 6: AP results across languages, comparing *SGNS* and the projected representations.

The results are shown in Table 6. We clearly see that for both languages the default *SGNS* embeddings do not provide higher similarity scores for hypernymy pairs (except for Italian Hyp/Ant), but both languages provide higher scores when we map the embeddings into the English *HyperVec* space.

6 Conclusion

This paper proposed a novel neural model *HyperVec* to learn hierarchical embeddings for hypernymy. *HyperVec* has been shown to strengthen hypernymy similarity, and to capture the distributional hierarchy of hypernymy. Together with a newly proposed unsupervised measure *HyperScore* our experiments demonstrated (i) significant improvements against state-of-the-art measures, and (ii) the capability to generalize hypernymy and learn the relation instead of memorizing *prototypical hypernyms*.

Acknowledgments

The research was supported by the Ministry of Education and Training of the Socialist Republic of Vietnam (Scholarship 977/QD-BGDDT; Kim-Anh Nguyen), the DFG Collaborative Research Centre SFB 732 (Kim-Anh Nguyen, Maximilian Köper, Ngoc Thang Vu), and the DFG Heisenberg Fellowship SCHU-2580/1 (Sabine Schulte im Walde). We would like to thank three anonymous reviewers for their comments and suggestions.

References

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 23–32, Avignon, France.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics (GEMS)*, pages 1–10, Edinburgh, Scotland.

Giulia Benotto. 2015. *Distributional models for semantic relations: A study on hyponymy and antonymy*. Ph.D. thesis, University of Pisa.

Or Biran and Kathleen McKeown. 2013. Classifying taxonomic relations between pairs of wikipedia articles. In *Proceddings of Sixth International Joint Conference on Natural Language Processing (IJCNLP)*, pages 788–794, Nagoya, Japan.

Daoud Clarke. 2009. Context-theoretic semantics for natural language: An overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics (GEMS)*, pages 112–119, Athens, Greece.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A Simple, Fast, and Effective Reparameterization of IBM Model 2. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 644–648, Atlanta, USA.

Francesca Fallucchi and Fabio Massimo Zanzotto. 2011. Inductive probabilistic taxonomy learning using singular value decomposition. *Natural Language Engineering*, 17(1):71–94.

Christiane Fellbaum, editor. 1998. *WordNet – An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA.

John R. Firth. 1957. *Papers in Linguistics 1934-51*. Longmans, London, UK.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 107–114, Michigan, US.

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

Douwe Kiela, Laura Rimell, Ivan Vulić, and Stephen Clark. 2015. Exploiting image generality for lexical entailment detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL)*, pages 119–124, Beijing, China.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359–389.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and Pollution: Delving into Cross-Space Mapping for Zero-Shot Learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 270–280, Beijing, China.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval)*, pages 75–79, Montréal, Canada.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceddings of the 27th International Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 2177–2185, Montréal, Canada.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 970–976, Denver, Colorado.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a. Exploiting Similarities among Languages for Machine Translation. *CoRR*, abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119, Lake Tahoe, Nevada, US.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

George A. Miller and Christiane Fellbaum. 1991. Semantic networks of english. *Cognition*, 41:197–229.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, M. Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 142–148, San Diego, California.

Gregory Murphy. 2002. *The Big Book of Concepts*. MIT Press, Cambridge, MA, USA.

Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1872–1877, Barcelona, Catalonia, Spain.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 454–459, Berlin, Germany.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet: : Similarity - measuring the relatedness of concepts. In *Proceedings of the 19th National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence (AAAI)*, pages 1024–1025, California, USA.

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

Laura Rimell. 2014. Distributional lexical entailment by topic coherence. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 511–519, Gothenburg, Sweden.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, pages 1025–1036, Dublin, Ireland.

Enrico Santus, Alessandro Lenci, Tin-Shing Chiu, Qin Lu, and Chu-Ren Huang. 2016. Unsupervised measure of word similarity: How to outperform co-occurrence and vector cosine in vsms. In *Proceedings of the Thirtieth Conference on Artificial Intelligence AAAI)*, pages 4260–4261, Arizona, USA.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte Im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 38–42, Gothenburg, Sweden.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. Evalution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of the 4th Workshop on Linked Data in Linguistics: Resources and Applications*, Beijing, China.

Roland Schäfer. 2015. Processing and querying large web corpora with the COW14 architecture. In *Proceedings of the 3rd Workshop on Challenges in the Management of Large Corpora*, pages 28–34, Lancaster, UK.

Roland Schäfer and Felix Bildhauer. 2012. Building large corpora from the web using a new efficient tool chain. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 486–493, Istanbul, Turkey.

Silke Scheible and Sabine Schulte im Walde. 2014. A Database of Paradigmatic Semantic Relation Pairs for German Nouns, Verbs, and Adjectives. In *Proceedings of Workshop on Lexical and Grammatical Resources for Language Processing*, pages 111–119, Dublin, Ireland.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. 2017. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Valencia, Spain.

Sidney Siegel and N. John Castellan. 1988. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, Boston, MA.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 801–808, Sydney, Australia.

Irene Sucameli. 2015. Analisi computazionale delle relazioni semantiche: Uno studio della lingua italiana. B.s. thesis, University of Pisa.

Luu Anh Tuan, Yi Tay, Siu Cheung Hui, and See Kiong Ng. 2016. Learning term embeddings for taxonomic relation identification using dynamic weighting neural network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 403–413, Austin, Texas.

Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico.

Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, California, USA.

Ivan Vulić, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. 2016. Hyperlex: A large-scale evaluation of graded lexical entailment. *arXiv*.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David J. Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, pages 2249–2259, Dublin, Ireland.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 81–88, Stroudsburg, PA, USA.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 1015–1021, Geneva, Switzerland.

Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics (ACL)*, pages 133–138, Las Cruces, New Mexico.

Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. 2015. Learning term embeddings for hypernymy identification. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI)*, pages 1390–1397, Buenos Aires, Argentina.

Maayan Zhitomirsky-Geffet and Ido Dagan. 2009. Bootstrapping distributional feature vector quality. *Computational Linguistics*, 35(3):435–461.

Ngram2vec: Learning Improved Word Representations from Ngram Co-occurrence Statistics

Zhe Zhao[1,2]
helloworld@ruc.edu.cn

Tao Liu[1,2]
tliu@ruc.edu.cn

Shen Li[3,4]
shen@mail.bnu.edu.cn

Bofang Li[1,2]
libofang@ruc.edu.cn

Xiaoyong Du[1,2]
duyong@ruc.edu.cn

[1] School of Information, Renmin University of China
[2] Key Laboratory of Data Engineering and Knowledge Engineering, MOE
[3] Institute of Chinese Information Processing, Beijing Normal University
[4] UltraPower-BNU Joint Laboratory for Artificial Intelligence, Beijing Normal University

Abstract

The existing word representation methods mostly limit their information source to word co-occurrence statistics. In this paper, we introduce ngrams into four representation methods: SGNS, GloVe, PPMI matrix, and its SVD factorization. Comprehensive experiments are conducted on word analogy and similarity tasks. The results show that improved word representations are learned from ngram co-occurrence statistics. We also demonstrate that the trained ngram representations are useful in many aspects such as finding antonyms and collocations. Besides, a novel approach of building co-occurrence matrix is proposed to alleviate the hardware burdens brought by ngrams.

1 Introduction

Recently, deep learning approaches have achieved state-of-the-art results on a range of NLP tasks. One of the most fundamental work in this field is word embedding, where low-dimensional word representations are learned from unlabeled corpora through neural models. The trained word embeddings reflect semantic and syntactic information of words. They are not only useful in revealing lexical semantics, but also used as inputs of various downstream tasks for better performance (Kim, 2014; Collobert et al., 2011; Pennington et al., 2014).

Most of the word embedding models are trained upon <*word, context*> pairs in the local window. Among them, word2vec gains its popularity by its amazing effectiveness and efficiency (Mikolov et al., 2013b,a). It achieves state-of-the-art results on a range of linguistic tasks with only a fraction of time compared with previous techniques. A challenger of word2vec is GloVe (Pennington et al., 2014). Instead of training on <*word, context*> pairs, GloVe directly utilizes word co-occurrence matrix. They claim that the change brings the improvement over word2vec on both accuracy and speed. Levy and Goldberg (2014b) further reveal that the attractive properties observed in word embeddings are not restricted to neural models such as word2vec and GloVe. They use traditional count-based method (PPMI matrix with hyper-parameter tuning) to represent words, and achieve comparable results with the above neural embedding models.

The above models limit their information source to word co-occurrence statistics (Levy et al., 2015). To learn improved word representations, we extend the information source from co-occurrence of ***'word-word'*** type to co-occurrence of ***'ngram-ngram'*** type. The idea of using ngrams is well supported by language modeling, one of the oldest problems studied in statistical NLP. In language models, co-occurrence of words and ngrams is used to predict the next word (Kneser and Ney, 1995; Katz, 1987). Actually, the idea of word embedding models roots in language models. They are closely related but are used for different purposes. Word embedding models aim at learning useful word representations instead of word prediction. Since ngram is a vital part in language modeling, we are inspired to integrate ngram statistical information into the recent word representation methods for better performance.

The idea of using ngrams is intuitive. However, there is still rare work using ngrams in recent representation methods. In this paper, we introduce

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 244–253
Copenhagen, Denmark, September 7–11, 2017.

ngrams into SGNS, GloVe, PPMI, and its SVD factorization. To evaluate the ngram-based models, comprehensive experiments are conducted on word analogy and similarity tasks. Experimental results demonstrate that the improved word representations are learned from ngram co-occurrence statistics. Besides that, we qualitatively evaluate the trained ngram representations. We show that they are able to reflect ngrams' meanings and syntactic patterns (e.g. 'be + past participle' pattern). The high-quality ngram representations are useful in many ways. For example, ngrams in negative form (e.g. 'not interesting') can be used for finding antonyms (e.g. 'boring').

Finally, a novel method is proposed to build ngram co-occurrence matrix. Our method reduces the disk I/O as much as possible, largely alleviating the costs brought by ngrams. We unify different representation methods in a pipeline. The source code is organized as ***ngram2vec*** toolkit and released at `https://github.com/zhezhaoa/ngram2vec`.

2 Related Work

SGNS, GloVe, PPMI, and its SVD factorization are used as baselines. The information used by them does not go beyond word co-occurrence statistics. However, their approaches to using the information are different. We review these methods in the following 3 sections. In section 2.4, we revisit the use of ngrams in the deep learning context.

2.1 SGNS

Skip-gram with negative sampling (SGNS) is a model in word2vec toolkit (Mikolov et al., 2013b,a). Its training procedure follows the majority of neural embedding models (Bengio et al., 2003): (1) *Scan the corpus and use <word, context> pairs in the local window as training samples.* (2) *Train the models to make words useful for predicting contexts (or in reverse).* The details of SGNS is discussed in Section 3.1. Compared to previous neural embedding models, SGNS speeds up the training process, reducing the training time from days or weeks to hours. Also, the trained embeddings possess attractive properties. They are able to reflect relations between two words accurately, which is evaluated by a fancy task called word analogy.

Due to the above advantages, many models are proposed on the basis of SGNS. For example, Faruqui et al. (2015) introduce knowledge in lexical resources into the models in word2vec. Zhao et al. (2016) extend the contexts from the local window to the entire documents. Li et al. (2015) use supervised information to guide the training. Dependency parse-tree is used for defining context in (Levy and Goldberg, 2014a). LSTM is used for modeling context in (Melamud et al., 2016) Sub-word information is considered in (Sun et al., 2016; Soricut and Och, 2015).

2.2 GloVe

Different from typical neural embedding models which are trained on <*word, context*> pairs, GloVe learns word representation on the basis of co-occurrence matrix (Pennington et al., 2014). GloVe breaks traditional 'words predict contexts' paradigm. Its objective is to reconstruct non-zero values in the matrix. The direct use of matrix is reported to bring improved results and higher speed. However, there is still dispute about the advantages of GloVe over word2vec (Levy et al., 2015; Schnabel et al., 2015). GloVe and other embedding models are essentially based on word co-occurrence statistics of the corpus. The <*word, context*> pairs and co-occurrence matrix can be converted to each other. Suzuki and Nagata (2015) try to unify GloVe and SGNS in one framework.

2.3 PPMI & SVD

When we are satisfied with the huge promotions achieved by embedding models on linguistic tasks, a natural question is raised: where the superiorities come from. One conjecture is that it's due to the neural networks. However, Levy and Goldberg (2014c) reveal that SGNS is just factoring PMI matrix implicitly. Also, Levy and Goldberg (2014b) show that positive PMI (PPMI) matrix still rivals the newly proposed embedding models on a range of linguistic tasks. Properties like word analogy are not restricted to neural models. To obtain dense word representations from PPMI matrix, we factorize PPMI matrix with SVD, a classic dimensionality reduction method for learning low-dimensional vectors from sparse matrix (Deerwester et al., 1990).

2.4 Ngram in Deep Learning

In the deep learning literature, ngram has shown to be useful in generating text representations. Recently, convolutional neural networks (CNNs) are reported to perform well on a range of NLP tasks (Blunsom et al., 2014; Hu et al., 2014; Severyn and Moschitti, 2015). CNNs are essentially using ngram information to represent texts. They use 1-D convolutional layers to extract ngram features and the distinct features are selected by max-pooling layers. In (Li et al., 2016), ngram embedding is introduced into Paragraph Vector model, where text embedding is trained to be useful to predict ngrams in the text. In the word embedding literature, a related work is done by Melamud et al. (2014), where word embedding models are used as baselines. They propose to use ngram language models to model the context, showing the effectiveness of ngrams on similarity tasks. Another work that is related to ngram is from Mikolov et al. (2013b), where phrases are embedded into vectors. It should be noted that phrases are different from ngrams. Phrases have clear semantics and the number of phrases is much less than the number of ngrams. Using phrase embedding has little impact on word embedding's quality.

3 Model

In this section, we introduce ngrams into SGNS, GloVe, PPMI, and SVD. Section 3.1 reviews the SGNS. Section 3.2 and 3.3 show the details of introducing ngrams into SGNS. In section 3.4, we show the way of using ngrams in GloVe, PPMI, and SVD, and propose a novel way of building ngram co-occurrence matrix.

3.1 Word Predicts Word: the Revisit of SGNS

First we establish some notations. The raw input is a corpus $T = \{w_1, w_2, ..., w_{|T|}\}$. Let W and C denote word and context vocabularies. θ is the parameters to be optimized. SGNS's parameters involve two parts: word embedding matrix and context embedding matrix. With embedding $\vec{w} \in R^d$, the total number of parameters is $(|W|+|C|)*d$.

The SGNS's objective is to maximize the conditional probabilities of contexts given center words:

$$\sum_{t=1}^{|T|}\left[\sum_{c\in C(w_t)} log\, p(c|w_t;\theta)\right] \qquad (1)$$

where $C(w_t) = \{w_i, t - win \le i \le t + win\, and\, i \ne t\}$ and *win* denotes the window size. As illustrated in figure 1, the center word 'written' predicts its surrounding words 'Potter', 'is', 'by', and 'J.K.'. In this paper, negative sampling (Mikolov et al., 2013b) is used to approximate the conditional probability:

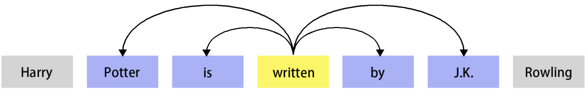


Figure 1: Illustration of 'word predicts word'.

$$p(c|w) = \sigma(\vec{w}^T\vec{c})\prod_{j=1}^{k} E_{c_j\sim P_n(C)}\sigma(-\vec{w}^T\vec{c_j}) \qquad (2)$$

where σ is sigmoid function. k samples (from c_1 to c_k) are drawn from context distribution raised to the power of *n*.

3.2 Word Predicts Ngram

In this section, we introduce ngrams into context vocabulary. We treat each ngram as a normal word and give it a unique embedding. During the training, the center word should not only predict its surrounding words, but also predict its surrounding ngrams. As shown in figure 2, center word 'written' predicts the bigrams in the local window such as 'by J.K.'. The objective of 'word predicts ngram' is similar with the original SGNS. The only difference is the definition of the *C(w)*. In ngram case, *C(w)* is formally defined as follows:

$$C(w_t) = \bigcup_{n=1}^{N}\{w_{i:i+n}|w_{i:i+n}\ is\ not\ w_t\ AND \\ t - win \le i \le t + win - n + 1\} \qquad (3)$$

where $w_{i:i+n}$ denotes the ngram $w_i w_{i+1}...w_{i+n-1}$ and N is the order of context ngram. Two points need to be noticed from the above definition. The first is how to determine the distance between center word and context ngram. In this paper, we use the distance between the word and the ngram's far-end word. As show in figure 2, the distance between 'written' and 'Harry Potter' is 3. As a result, 'Harry Potter' is not included in the center word's context. This distance definition ensures that the ngram models don't use the information beyond the pre-specified window, which guarantees fair comparisons with baselines. Another point is whether the overlap of word and ngram is allowed or not. In the overlap situation, ngrams are used as context even they contain the center word. As the example in figure 2 shows,

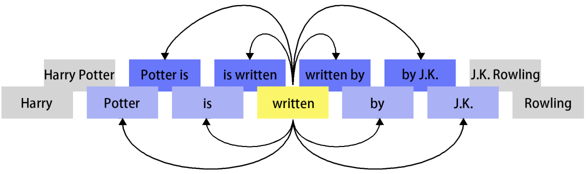


Figure 2: Illustration of ‘word predicts ngram’.

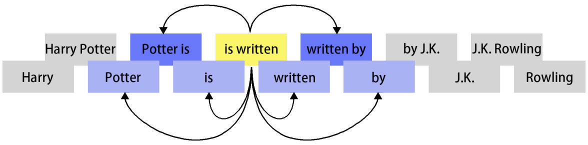


Figure 3: Illustration of ‘ngram predicts ngram’.

ngram ‘is written’ and ‘written by’ are predicted by the center word ‘written’. In the non-overlap case, these ngrams are excluded. The properties of word embeddings are different when overlap is allowed or not, which will be discussed in experiments section.

3.3 Ngram Predicts Ngram

We further extend the model to introduce ngrams into center word vocabulary. During the training, center ngrams (including words) predict their surrounding ngrams. As shown in figure 3, center bigram ‘is written’ predicts its surrounding words and bigrams. The objective of ‘ngram predicts ngram’ is as follows:

$$\sum_{t=1}^{|T|} \sum_{n_w=1}^{N_w} \left[\sum_{c \in C(w_{t:t+n_w})} log\, p(c|w_{t:t+n_w}; \theta) \right] \quad (4)$$

where N_w is the order of center ngram. The definition of $C(w_{t:t+n_w})$ is as follows:

$$\bigcup_{n_c=1}^{N_c} \{w_{i:i+n_c} | w_{i:i+n_c}\ is\ not\ w_{t:t+n_w}\ AND \\ t - win + n_w - 1 \leq i \leq t + win - n_c + 1\} \quad (5)$$

where N_c is the order of context ngram. To this end, the word embeddings are not only affected by the ngrams in the context, but also indirectly affected by co-occurrence statistics of ‘ngram-ngram’ type in the corpus.

SGNS is proven to be equivalent with factorizing pointwise mutual information (PMI) matrix (Levy and Goldberg, 2014c). Following their work, we can easily show that models in section 3.2 and 3.3 are implicitly factoring PMI matrix of ‘word-ngram’ and ‘ngram-ngram’ type. In the next section, we will discuss the content of introducing ngrams into positive PMI (PPMI) matrix.

3.4 Co-occurrence Matrix Construction

Introducing ngrams into GloVe, PPMI, and SVD is straightforward: the only change is to replace word co-occurrence matrices with ngram ones. In the above three sections, we have discussed the way of taking out *<word(ngram), word(ngram)>* pairs from a corpus. Afterwards, we build the co-occurrence matrix upon these pairs. The rest steps are identical with the original baseline models.

Win	Type	#Pairs
2	uni_uni	0.36B
	uni_bi	1.14B
	uni_tri	1.40B
	bi_bi	2.78B
	bi_tri	3.65B
5	uni_uni	0.91B
	uni_bi	2.79B
	uni_tri	3.81B
	bi_bi	7.97B

Table 1: The number of pairs at different settings. The type column lists the order of ngrams considered in center word/context vocabularies. For example, uni_bi denotes that center word vocabulary contains unigrams (words) and context vocabulary contains both unigrams and bigrams. The setting of other hyper-parameters is discussed in Section 4.2.

However, building the co-occurrence matrix is not an easy task as it apparently looks like. The introduction of ngrams brings huge burdens on the hardware. The matrix construction cost is closely related to the number of pairs (**#Pairs**). Table 1 shows the statistics of pairs extracted from corpus wiki2010 [1]. We can observe that **#Pairs** is huge when ngrams are considered.

To speed up the process of building ngram co-occurrence matrix, we take advantages of ‘mixture’ strategy (Pennington et al., 2014) and ‘stripes’ strategy (Dyer et al., 2008; Lin, 2008). The two strategies optimize the process in different aspects. Computational cost is reduced significantly when they are used together.

[1] http://nlp.stanford.edu/data/WestburyLab.wikicorp.201004.txt.bz2

When words (or ngrams) are sorted in descending order by frequency, the co-occurrence matrix's top-left corner is dense while the rest part is sparse. Based on this observation, the 'mixture' of two data structures are used for storing matrix. Elements in the top-left corner are stored in a 2D array, which stays in memory. The rest of the elements are stored in the form of <*ngram, H*>, where *H*<*context, count*> is an associative array recording the number of times the *ngram* and *context* co-occurs ('stripes' strategy). Compared with storing <ngram, context> pairs explicitly, the 'stripes' strategy provides more opportunities to aggregate pairs outside of the top-left corner.

Algorithm 1 shows the way of using the 'mixture' and 'stripes' strategies together. In the first stage, pairs are stored in different data structures according to *topLeft* function. Intermediate results are written to temporary files when memory is full. In the second stage, we merge these sorted temporary files to generate co-occurrence matrix. The *getSmallest* function takes out the pair <*ngram, H*> with the smallest *key* from temporary files. In practice, algorithm 1 is efficient. Instead of using computer clusters (Lin, 2008), we can build the matrix of 'bi_bi' type even in a laptop. It only requires 12GB to store temporary files (win=2, subsampling=0, memory size=4GB), which is much smaller than the implementations in (Pennington et al., 2014; Levy et al., 2015) . More detailed analysis about these strategies can be found in the ***ngram2vec*** toolkit.

Algorithm 1: An algorithm for building n-gram co-occurrence matrix

```
Input  : Pairs $P$, Sorted vocabulary $V$
Output: Sorted and aggregated pairs
The 2D array $A[\,][\,]$;
The dictionary $D < ngram, H >$;
The temporary files array $tfs[\,]$; $fid$=1;
for pair $p < n, c >$ in $P$ do
    if $topLeft(n, c) == 1$ then
        $A[getId(n)][getId(c)]$ += 1;
    else
        $D\{n\}\{c\}$ += 1;
        if Memory is full or $P$ is empty then
            Sort $D$ by key (ngram);
            Write $D$ to $tfs[fid]$;
            $fid$ += 1;
        end
    end
end
Write $A$ to $tfs[0]$ in the form of $< ngram, H >$;
$old = getSmallest(tfs)$ ;
while !(All files in $tfs$ are empty) do
    $new = getSmallest(tfs)$ ;
    if $old.ngram == new.ngram$ then
        $old =$
        $< old.ngram, merge(old.H, new.H) >$;
    else
        Write $old$ to disk;
        $old = new$
    end
end
```

4 Experiments

4.1 Datasets

The tasks used in this paper is the same with the work of Levy et al. (2015), including six similarity and two analogy datasets. In similarity task, a scalar (e.g. a score from 0 to 10) is used to measure the relation between the two words. For example, in a similarity dataset, the 'train, car' pair is given the score of 6.31. A problem of similarity task is that scalar only reflects the strength of the relation, while the type of relation is totally ignored (Schnabel et al., 2015).

Due to the deficiency of similarity task, analogy task is widely used as benchmark recently for evaluation of word embedding models. To answer analogy questions, relations between the two words are reflected by a vector, which is usually obtained by the difference between word embeddings. Different from a scalar, the vector provides more accurate descriptions of relations. For example, capital-country relation is encoded in *vec(Athens)-vec(Greece)*, *vec(Tokyo)-vec(Japan)* and so on. More concretely, the questions in the analogy task are in the form of 'a is to b as c is to d'. 'd' is an unknown word in the test phase. To correctly answer the questions, the models should embed the two relations, *vec(a)-vec(b)* and *vec(c)-vec(d)*, into similar positions in the space. Following the work of Levy and Goldberg (2014b), both additive (add) and multiplicative (mul) functions are used for finding word 'd'. The latter one is more suitable for sparse representation in practice.

4.2 Pipeline and Hyper-parameter Setting

We implement SGNS, GloVe, PPMI, and SVD in a pipeline, allowing the reuse of code and intermediate results. Figure 4 illustrates the overview of the pipeline. Firstly, <*word(ngram), word(ngram)*> pairs are extracted from the corpus as the input of SGNS. Afterwards, we build the co-occurrence matrix upon the pairs. GloVe and PPMI learn word representations on the basis of co-occurrence

Win	Type		Google Tot. / Sem. / Syn.		MSR	
			Add	Mul	Add	Mul
2	uni_uni		.579 / .543 / .608	.597 / .561 / .627	.513	.533
	overlap	uni_bi	.587 / .651 / .533	.626 / .681 / .580	.473	.508
		uni_tri	.505 / .615 / .414	.553 / .657 / .466	.358	.396
		bi_bi	**.664** / **.739** / .602	**.680** / **.739** / .631	.547	.575
		bi_tri	.572 / .695 / .470	.601 / .713 / .508	.416	.447
	non-overlap	uni_bi	.610 / .558 / .653	.633 / .581 / .676	.568	.595
		bi_bi	.644 / .607 / **.674**	.659 / .613 / **.696**	**.590**	**.616**
5	uni_uni		.653 / .669 / .639	.668 / .678 / .660	.511	.535
	overlap	uni_bi	.696 / .745 / .655	.714 / .752 / .683	.518	.542
		uni_tri	.679 / .738 / .630	.699 / .750 / .657	.542	.549
		bi_bi	.704 / **.764** / .654	.718 / **.764** / .681	.537	.560
	non-overlap	uni_bi	.696 / .722 / .675	.716 / .731 / .703	.549	.579
		uni_tri	.687 / .711 / .668	.705 / .717 / .696	.542	.574
		bi_bi	**.712** / .745 / **.684**	**.725** / .742 / **.710**	**.569**	**.607**

Table 2: Performance of (ngram) SGNS on analogy datasets.

Win	Type	Sim.	Rel.	Bruni	Radinsky	Luong	Hill
2	uni_uni	.745	.586	.713	.635	.387	.419
	uni_bi	.739	**.600**	.698	.627	.395	**.429**
	uni_tri	.700	.535	.658	.591	.380	.415
	bi_bi	**.757**	.574	**.724**	**.644**	**.408**	.407
	bi_tri	.724	.564	.669	.605	.403	.412
5	uni_uni	.789	.648	.756	.652	.407	.401
	uni_bi	.794	.681	.752	.653	.437	.431
	uni_tri	.783	.673	.743	.652	.432	**.436**
	bi_bi	**.816**	**.703**	**.760**	**.671**	**.446**	.421

Table 3: Performance of (ngram) SGNS on similarity datasets.

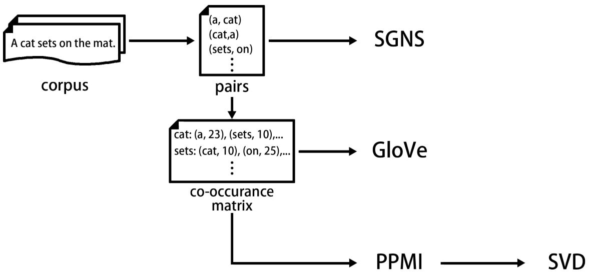


Figure 4: The pipeline.

matrix. SVD factorizes the PPMI matrix to obtain low-dimensional representation.

Most hyper-parameters come from 'corpus to pairs' part and four representation models. 'corpus to pairs' part determines the source of information for the subsequent models and its hyper-parameter setting is as follows: low-frequency words (n-grams) are removed with a threshold of 10. High-frequency words (ngrams) are removed with sub-sampling at the degree of 1e-5 [2]. Window size is set to 2 and 5. Clean strategy (Levy et al., 2015) is used to ensure no information beyond pre-specified window is included. Overlap setting is used in default. For hyper-parameters of four representation models, we use the embeddings of 300 dimensions in dense representations. SGNS is trained by 3 iterations. The rest strictly follow the baseline models [3]. We consider unigrams (words), bigrams, and trigrams in this work. The implementation of higher-order models and their results will be released with ***ngram2vec*** toolkit.

4.3 Ngrams on SGNS

SGNS is a popular word embedding model. Even compared with its challengers such as GloVe, S-GNS is reported to have more robust performance with faster training speed (Levy et al., 2015). Table 2 lists the results on analogy datasets. We can observe that the introduction of bigrams provides significant improvements at different hyper-parameter settings. The SGNS of 'bi_bi' type provides the highest results. It is very effective on capturing semantic information (Google semantic). Around 10 percent improvements are wit-

[2] Sub-sampling is not used in GloVe, which follows its o-riginal setting.

[3] http://bitbucket.org/omerlevy/hyperwords for SGNS, PPMI and SVD; http://nlp.stanford.edu/projects/glove/ for GloVe.

Win	Type	Google		MSR		Sim.	Rel.	Bruni	Radinsky	Luong	Hill
		Add	Mul	Add	Mul						
2	uni_uni	.403 / .441 / .372	.614 / .725 / .522	.235	.419	.709	.593	.705	.603	.293	.385
	uni_bi	.403 / .550 / .281	.733 / .854 / .632	.241	.572	.731	.581	.721	.627	.341	.394
5	uni_uni	.423 / .505 / .355	.580 / .740 / .447	.198	.339	.721	.619	.712	.619	.252	.341
	uni_bi	.453 / .590 / .338	.730 / .841 / .637	.281	.579	.707	.547	.696	.619	.296	.378

Table 4: Performance of (ngram) PPMI on analogy and similarity datasets.

Win	Type	Google		MSR		Sim.	Rel.	Bruni	Radinsky	Luong	Hill
		Add	Mul	Add	Mul						
2	uni_uni	.535 / .599 / .482	.540 / .610 / .481	.444	.445	.681	.529	.698	.608	.381	.351
	uni_bi	.543 / .601 / .493	.549 / .612 / .496	.464	.472	.686	.545	.695	.631	.389	.352
5	uni_uni	.625 / .689 / .572	.626 / .696 / .568	.476	.490	.747	.600	.735	.657	.389	.347
	uni_bi	.631 / .699 / .575	.633 / .703 / .574	.477	.504	.752	.610	.737	.631	.395	.342

Table 5: Performance of (ngram) GloVe on analogy and similarity datasets.

Win	Type	Google		MSR		Sim.	Rel.	Bruni	Radinsky	Luong	Hill
		Add	Mul	Add	Mul						
2	uni_uni	.419 / .388 / .446	.439 / .394 / .477	.321	.353	.714	.593	.712	.625	.410	.344
	uni_bi	.387 / .322 / .440	.410 / .327 / .479	.372	.402	.739	.546	.688	.636	.427	.347
5	uni_uni	.433 / .426 / .439	.460 / .463 / .458	.290	.321	.752	.633	.731	.623	.411	.326
	uni_bi	.410 / .340 / .468	.446 / .365 / .513	.374	.416	.751	.559	.698	.639	.426	.363

Table 6: Performance of (ngram) SVD on analogy and similarity datasets.

nessed on semantic questions compared with uni_uni baseline. For syntactic questions (Google syntactic and MSR datasets), around 5 percent improvements are obtained on average.

The effect of overlap is large on analogy datasets. Semantic questions prefer the overlap setting. Around 10 and 3 percent improvements are witnessed compared with non-overlap setting at the window size of 2 and 5. While in syntactic case, non-overlap setting performs better by a margin of around 5 percent.

The introduction of trigrams deteriorates the models' performance on analogy datasets (especially at the window size of 2). It is probably because that trigram is sparse on wiki2010, a relatively small corpus with 1 billion tokens. We conjecture that high order ngrams are more suitable for large corpora and will report the results in our future work. It should be noticed that trigram is not included in vocabulary in non-overlap case at the window size of 2. The shortest distance between a word and a trigram is 3, which exceeds the window size.

Table 3 illustrates the SGNS's performance on similarity task. The conclusion is similar with the case in analogy datasets. The use of bigrams is effective while the introduction of trigrams deteriorates the performance in most cases. In general, the bigrams bring significant improvements over SGNS on a range of linguistic tasks. It is generally known that ngram is a vital part in traditional language modeling problem. Results in table 2 and 3 confirm the effectiveness of ngrams again on SGNS, a more advanced word embedding model.

4.4 Ngrams on PPMI, GloVe, SVD

In this section, we only report the results of models of 'uni_uni' and 'uni_bi' types. Using higher order co-occurrence statistics brings immense costs (especially at the window size of 5). Levy and Goldberg (2014b) demonstrate that traditional count-based models can still achieve competitive results on many linguistic tasks, challenging the dominance of neural embedding models. Table 4 lists the results of PPMI matrix on analogy and similarity datasets. PPMI prefers Multiplicative (Mul) evalution. To this end, we focus on analyzing the results on Mul columns. When bigrams are used, significant improvements are witnessed on analogy task. On Google dataset, bigrams bring over 10 percent increase on the total accuracies. At the window size of 2, the accuracy in semantic questions even reaches 0.854, which is the state-of-the-art result to the best of our knowledge. On MSR dataset, around 20 percent improvements are achieved. The use of bigrams does not always bring improvements on similarity datasets. PPMI matrix of 'uni_bi' type improves the results on 5 datasets at the window size of 2. At the window size of 5, using bigrams only improves the results on 2 datasets.

Table 5 and 6 list GloVe and SVD's results. For GloVe, consistent (but minor) improvements are achieved on analogy task with the introduction of bigrams. On similarity datasets, improvements are witnessed on most cases. For SVD, bigrams sometimes lead to worse results in both analogy and similarity tasks. In general, significant improvements are not witnessed on GloVe and SVD. Our preliminary conjecture is that the default hyper-parameter setting should be blamed. We strictly follow the hyper-parameters used in baseline models, making no adjustments to cater to the introduction of ngrams. Besides that, some common techniques such as dynamic window, decreasing weighting function, dirty sub-sampling are discarded. The relationships between ngrams and various hyper-parameters require further exploration. Though trivial, it may lead to much better results and give researchers better understanding of different representation methods. That will be the focus of our future work.

4.5 Qualitative Evaluations of Ngram Embedding

In this section, we analyze the properties of ngram embeddings trained by SGNS of 'bi_bi' type. Ideally, the trained ngram embeddings should reflect ngrams' semantic meanings. For example, *vec(wasn't able)* should be close to *vec(unable)*. *vec(is written)* should be close to *vec(write)* and *vec(book)*. Also, the trained ngram embeddings should preserve ngrams' syntactic patterns. For example, 'was written' is in the form of 'be + past participle' and the nearest neighbors should possess similar patterns, such as 'is written' and 'was transcribed'.

Table 7 lists the target ngrams and their top nearest neighbours. We divide the target ngrams into six groups according to their patterns. We can observe that the returned words and ngrams are very intuitive. As might be expected, synonyms of the target ngrams are returned in top positions (e.g. 'give off' and 'emit'; 'heavy rain' and 'downpours'). From the results of the first group, it can be observed that bigram in negative form 'not X' is useful for finding the antonym of word 'X'. Besides that, the trained ngram embeddings also preserve some common sense. For example, the returned result of 'highest mountain' is a list of mountain names (with a few exceptions such as 'unclimbed'). In terms of syntactic patterns, we can observe that in most cases, the returned ngrams are in the similar form with target ngrams. In general, the trained embeddings basically reflect semantic meanings and syntactic patterns of ngrams.

With high-quality ngram embeddings, we have the opportunity to do more interesting things in our future work. For example, we will construct a antonym dataset to evaluate ngram embeddings systematically. Besides that, we will find more scenarios for using ngram embeddings. In our view, ngram embeddings have potential to be used in many NLP tasks. For example, Johnson and Zhang (2015) use one-hot ngram representation as the input of CNN. Li et al. (2016) use ngram embeddings to represent texts. Intuitively, initializing these models with pre-trained ngram embeddings may further improve the accuracies.

5 Conclusion

We introduce ngrams into four representation methods. The experimental results demonstrate ngrams' effectiveness for learning improved word representations. In addition, we find that the trained ngram embeddings are able to reflect their semantic meanings and syntactic patterns. To alleviate the costs brought by ngrams, we propose a novel way of building co-occurrence matrix, enabling the ngram-based models to run on cheap hardware.

Acknowledgments

This work is supported by National Natural Science Foundation of China (Grant No. 61472428 and No. 71531012), the Fundamental Research Funds for the Central Universities, the Research Funds of Renmin University of China No. 14XNLQ06. This work is partially supported by ECNU-RUC-InfoSys Joint Data Science Lab and a gift from Tencent.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL 2014*.

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

Pattern	Target	Word	Bigram
Negative Form	**wasn't able**	**unable(.745), couldn't(.723), didn't(.680)**	**was unable(.832), didn't manage(.799)**
	don't need	**don't(.773), dont(.751), needn't(.715)**	**dont need(.790), don't have(.785), dont want(.769)**
	not enough	**enough(.708), insufficient(.701), sufficient(.629)**	**not sufficient(.750), wasn't enough(.729)**
Adj. Modifier	**heavy rain**	**torrential(.844), downpours(.780), rain(.766)**	**torrential rain(.829), heavy rainfall(.799)**
	strong supporter	**supporter(.828), proponent(.733), admirer(.602)**	**staunch supporter(.870), vocal supporter(.810)**
	high quality	**high-quality(.867), quality(.744)**	**good quality(.813), top quality(.751)**
Passive Voice	**was written**	**written(.793), penned(.675), co-written(.629)**	**were written(.785), is written(.744), written by(.739)**
	was sent	**sent(.844), dispatched(.661), went(.630)**	**then sent(.779), later sent(.776), was dispatched(.774)**
	was pulled	**pulled(.730), yanked(.629), limped(.593)**	**were pulled(.706), pulled from(.691), was ripped(.682)**
Perfect Tense	**has achieved**	**achieved(.683), achieves(.680), achieving(.625)**	**has attained(.775), has enjoyed(.741), has gained(.733)**
	has impacted	**interconnectedness(.679), pervade(.676)**	**have impacted(.838), is affecting(.773), have shaped(.772)**
	has published	**authored(.722), publishes(.705), coauthored(.791)**	**has authored(.852), has edited(.795), has written(.791)**
Phrasal Verb	**give off**	**exude(.796), fluoresce(.789), emit(.754)**	**gave off(.837), giving off(.820), and emit(.816)**
	make up	**comprise(.726), constitute(.616), make(.541)**	**makes up(.705), making up(.702), comprise the(.672)**
	picked up	**picked(.870), snagged(.544), scooped(.538)**	**later picked(.712), and picked(.682), then picked(.681)**
Common Sense	**highest mountain**	**muztagh(.669), prokletije(.664), cadair(.658)**	**highest peak(.873), tallest mountain(.857)**
	avian influenza	**h5n1(.870), zoonotic(.812), adenovirus(.806)**	**avian flu(.885), the h5n1(.870), flu virus(.868)**
	computer vision	**human-computer(.789), holography(.767)**	**image processing(.850), object recognition(.818)**

Table 7: Target bigrams and their nearest neighbours associated with similarity scores.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.

Christopher Dyer, Aaron Cordova, Alex Mont, and Jimmy Lin. 2008. Fast, easy, and cheap: Construction of statistical machine translation models with mapreduce. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 199–207.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of NIPS 2014*, pages 2042–2050.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of NAACL 2015*, pages 103–112.

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 35(3):400–401.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP 2014*, pages 1746–1751.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of ICASSP 1995*, pages 181–184.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *ACL (2)*, pages 302–308.

Omer Levy and Yoav Goldberg. 2014b. Linguistic regularities in sparse and explicit word representations. In *Proceedings of CoNLL 2014*, pages 171–180.

Omer Levy and Yoav Goldberg. 2014c. Neural word embedding as implicit matrix factorization. In *Proceedings of NIPS 2014*, pages 2177–2185.

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

Bofang Li, Zhe Zhao, Tao Liu, Puwei Wang, and Xiaoyong Du. 2016. Weighted neural bag-of-n-grams model: New baselines for text classification. In *Proceedings of COLING 2016*, pages 1591–1600.

Yitan Li, Linli Xu, Fei Tian, Liang Jiang, Xiaowei Zhong, and Enhong Chen. 2015. Word embedding revisited: A new representation learning and explicit matrix factorization perspective. In *Proceedings of IJCAI 2015*, pages 3650–3656.

Jimmy J. Lin. 2008. Scalable language processing algorithms for the masses: A case study in computing word co-occurrence matrices with mapreduce. In *Proceedings of EMNLP 2008*, pages 419–428.

Oren Melamud, Ido Dagan, Jacob Goldberger, Idan Szpektor, and Deniz Yuret. 2014. Probabilistic modeling of joint-context in distributional similarity. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, CoNLL 2014, Baltimore, Maryland, USA, June 26-27, 2014*, pages 181–190.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 51–61.

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

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

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

Tobias Schnabel, Igor Labutov, David M. Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of EMNLP 2015*, pages 298–307.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of SIGIR 2015*, pages 373–382.

Radu Soricut and Franz Josef Och. 2015. Unsupervised morphology induction using word embeddings. In *Proceedings of NAACL 2015*, pages 1627–1637.

Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2016. Inside out: Two jointly predictive models for word representations and phrase representations. In *Proceedings of AAAI 2016*, pages 2821–2827.

Jun Suzuki and Masaaki Nagata. 2015. A unified learning framework of skip-grams and global vectors. In *Proceedings of ACL 2015, Volume 2: Short Papers*, pages 186–191.

Zhe Zhao, Tao Liu, Bofang Li, and Xiaoyong Du. 2016. Cluster-driven model for improved word and text embedding. In *Proceedings of ECAI 2016*, pages 99–106.

`Dict2vec` : Learning Word Embeddings using Lexical Dictionaries

Julien Tissier and **Christophe Gravier** and **Amaury Habrard**
Univ. Lyon, UJM Saint-Etienne
CNRS, Lab Hubert Curien UMR 5516
F-42023, Saint-Etienne, France
{firstname.lastname}@univ-st-etienne.fr

Abstract

Learning word embeddings on large unlabeled corpus has been shown to be successful in improving many natural language tasks. The most efficient and popular approaches learn or retrofit such representations using additional external data. Resulting embeddings are generally better than their corpus-only counterparts, although such resources cover a fraction of words in the vocabulary. In this paper, we propose a new approach, `Dict2vec`, based on one of the largest yet refined datasource for describing words – natural language dictionaries. `Dict2vec` builds new word pairs from dictionary entries so that semantically-related words are moved closer, and negative sampling filters out pairs whose words are unrelated in dictionaries. We evaluate the word representations obtained using `Dict2vec` on eleven datasets for the word similarity task and on four datasets for a text classification task.

1 Introduction

Learning word embeddings usually relies on the distributional hypothesis – words appearing in similar contexts must have similar meanings, and thus close representations. Finding such representations for words and sentences has been one hot topic over the last few years in Natural Language Processing (NLP) (Mikolov et al., 2013; Pennington et al., 2014) and has led to many improvements in core NLP tasks such as Word Sense Disambiguation (Iacobacci et al., 2016), Machine Translation (Devlin et al., 2014), Machine Comprehension (Hewlett et al., 2016), and Semantic Role Labeling (Zhou and Xu, 2015; Collobert et al., 2011) – to name a few.

These methods suffer from a classic drawback of unsupervised learning: the lack of supervision between a word and those appearing in the associated contexts. Indeed, it is likely that some terms of the context are not related to the considered word. On the other hand, the fact that two words do not appear together – or more likely, not often enough together – in any context of the training corpora is not a guarantee that these words are not semantically related. Recent approaches have proposed to tackle this issue using an attentive model for context selection (Ling et al., 2015), or by using external sources – like knowledge graphs – in order to improve the embeddings (Wang et al., 2014). Similarities derived from such resources are part of the objective function during the learning phase (Yu and Dredze, 2014; Kiela et al., 2015) or used in a retrofitting scheme (Faruqui et al., 2015). These approaches tend to specialize the embeddings to the resource used and its associated similarity measures – while the construction and maintenance of these resources are a set of complex, time-consuming, and error-prone tasks.

In this paper, we propose a novel word embedding learning strategy, called `Dict2vec`, that leverages existing online natural language dictionaries. We assume that dictionary entries (a definition of a word) contain latent word similarity and relatedness information that can improve language representations. Such entries provide, in essence, an additional context that conveys general semantic coverage for most words. `Dict2vec` adds new co-occurrences information based on the terms occurring in the definitions of a word. This information introduces weak supervision that can be used to improve the embeddings. We can indeed distinguish word pairs for which each word appears in the definition of the other (strong pairs) and pairs where only one appears in the definition of the other (weak pairs) – each

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 254–263
Copenhagen, Denmark, September 7–11, 2017.

having their own weight as two hyperparameters. Not only this information is useful at learning time to control words vectors to be close for such word pairs, but also it becomes possible to devise a controlled negative sampling. Controlled negative sampling as introduced in `Dict2vec` consists in filtering out random negative examples in conventional negative sampling that forms a (strong or weak) pair with the target word – they are obviously non-negative examples. Processing online dictionaries in `Dict2vec` does not require a human-in-the-loop – it is fully automated. The neural network architecture from `Dict2vec` (Section 3) extends `Word2vec` (Mikolov et al., 2013) approach which uses a Skip-gram model with negative sampling.

Our main results are as follows :

- `Dict2vec` exhibits a statistically significant improvement around 12.5% against state-of-the-art solutions on eleven most common evaluation datasets for the word similarity task when embeddings are learned using the full Wikipedia dump.
- This edge is even more significant for small training datasets (50 millions first tokens of Wikipedia) than using the full dataset, as the average improvement reaches 30%.
- Since `Dict2vec` does significantly better than competitors for small dimensions (in the $[20;100]$ range) for small corpus, it can yield smaller yet efficient embeddings – even when trained on smaller corpus – which is one of the utmost practical interest for the working natural language processing practitioners.
- We also show that the embeddings learned by `Dict2vec` perform similarly to other baselines on an extrinsic text classification task.

Dict2vec software is an extension and an optimization from the original Word2vec framework leading to a more efficient learning. Source code to fetch dictionaries, train Dict2vec models and evaluate word embeddings are publicly availabe[1] and can be used by the community as a seed for future works.

The paper is organized as follows. Section 2 presents related works, along with a special focus on Word2vec, which we later derive in our approach presented in Section 3. Our experimental setup and evaluation settings are introduced in Section 4 and we discuss the results in Section 5. Section 6 concludes the paper.

2 Learning Word Embeddings

2.1 The Neural Network Approach

In the original model from Collobert and Weston (2008), a window approach was used to feed a neural network and learn word embeddings. Since there are long-range relations between words, the window-based approach was later extended to a sentence-based approach (Collobert et al., 2011) leading to capture more semantic similarities into word vectors. Recurrent neural networks are another way to exploit the context of a word by considering the sequence of words preceding it (Mikolov et al., 2010; Sutskever et al., 2011). Each neuron receives the current window as an input, but also its own output from the previous step.

Mikolov et al. (2013) introduced the Skip-gram architecture built on a single hidden layer neural network to learn efficiently a vector representation for each word w of a vocabulary V from a large corpora of size C. Skip-gram iterates over all (target, context) pairs (w_t,w_c) from every window of the corpus and tries to predict w_c knowing w_t. The objective function is therefore to maximize the log-likelihood :

$$\sum_{t=1}^{C} \sum_{k=-n}^{n} \log p(w_{t+k}|w_t) \tag{1}$$

where n represents the size of the window (composed of n words around the central word w_t) and the probability can be expressed as :

$$p(w_{t+k}|w_t) = \frac{e^{v_{t+k} \cdot v_t}}{\sum_{w \in V} e^{v \cdot v_t}} \tag{2}$$

with v_{t+k} (resp. v_t) the vector associated to w_{t+k} (resp. w_t).

This model relies on the principle "You shall know a word by the company it keeps" – Firth (1957). Thus, words that are frequent within the context of the target word will tend to have close representations, as the model will update their vectors so that they will be closer. Two main drawbacks can be said about this approach. First, words within the same window are not always related. Consider the sentence "Turing is widely considered to be the father of theoretical

[1] https://github.com/tca19/dict2vec

computer science and artificial intelligence."[2], the words (Turing,widely) and (father,theoretical) will be moved closer while they are not semantically related. Second, strong semantic relations between words (like synonymy or meronymy) happens rarely within the same window, so these relations will not be well embedded into vectors.

fastText introduced in Bojanowski et al. (2016) uses internal additional information from the corpus to solve the latter drawback. They train a Skip-gram architecture to predict a word w_c given the central word w_t and all the n-grams $\mathcal{G}_{w_t}$ (subwords of 3 up to 6 letters) of w_t. The objective function becomes :

$$\sum_{t=1}^{C} \sum_{k=-n}^{n} \sum_{w \in \mathcal{G}_{w_t}} \log p(w_{t+k}|w) \qquad (3)$$

Along learning one vector per word, fastText also learns one vector per n-gram. fastText is able to extract more semantic relations between words that share common n-gram(s) (like fish and fishing) which can also help to provide good embeddings for rare words since we can obtain a vector by summing vectors of its n-grams.

In what follows, we report related works that leverage external resources in order to address the two raised issues about the window approach.

2.2 Using External Resources

Even with larger and larger text data available on the Web, extracting and encoding every linguistic relations into word embeddings directly from corpora is a difficult task. One way to add more relations into embeddings is to use external data. Lexical databases like WordNet or sets of synonyms like MyThes thesaurus can be used during learning or in a post-processing step to specialize word embeddings. For example, Yu and Dredze (2014) include prior knowledge about synonyms from WordNet and the Paraphrase Database in a joint model built upon Word2vec. Faruqui et al. (2015) introduce a graph-based retrofitting method where they post-process learned vectors with respect to semantic relationships extracted from additional lexical resources. Kiela et al. (2015) propose to specialize the embeddings either on similarity or relatedness relations in a Skip-gram joint learning approach by adding new contexts from external thesaurus or from a norm association base in the function to optimize. Bian et al. (2014) combine several sources (syllables, POS tags, antonyms/synonyms, Freebase relations) and incorporate them into a CBOW model. These approaches have generally the objective to improve tasks such as document classification, synonym detection or word similarity. They rely on additional resources whose construction is a time-consuming and error-prone task and tend generally to specialize the embeddings to the external corpus used. Moreover, lexical databases contain less information than dictionaries (117k entries in WordNet, 200k in a dictionary) and less accurate content (some different words in WordNet belong to the same synset thus have the same definition).

Another type of external resources are knowledge bases, containing triplets. Each triplet links two entities with a relation, for example Paris – is capital of – France. Several methods (Weston et al., 2013; Wang et al., 2014; Xu et al., 2014) have been proposed to use the information from knowledge base to improve semantic relations in word embeddings, and extract more easily relational facts from text. These approaches are focused on knowledge base dependent task.

3 `Dict2vec`

The definition of a word is a group of words or sentences explaining its meaning. A dictionary is a set of tuples (word, definition) for several words. For example, one may find in a dictionary :

> **car**: A road vehicle, typically with four wheels, powered by an internal combustion engine and able to carry a small number of people.[3]

The presence of words like "vehicle", "road" or "engine" in the definition of "car" illustrates the relevance of using word definitions for obtaining weak supervision allowing us to get semantically related pairs of words.

`Dict2vec` models this information by building strong and weak pairs of words (§3.1), in order to provide both a novel positive sampling objective (§3.2) and a novel controlled negative sampling objective (§3.3). These objectives participate to the global objective function of `Dict2vec` (§3.4).

[2] https://en.wikipedia.org/wiki/Alan_Turing

[3] Definition from Oxford dictionary.

3.1 Strong pairs, weak pairs

In a definition, each word does not have the same semantic relevance. In the definition of "car", the words "internal" or "number" are less relevant than "vehicle". We introduce the concept of strong and weak pairs in order to capture this relevance. If the word w_a is in the definition of the word w_b and w_b is in the definition of w_a, they form a strong pair, as well as the K closest words to w_a (resp. w_b) form a strong pair with w_b (resp. w_a). If the word w_a is in the definition of w_b but w_b is not in the definition of w_a, they form a weak pair.

The word "vehicle" is in the definition of "car" and "car" is in the definition of "vehicle". Hence, (car–vehicle) is a strong pair. The word "road" is in the definition of "car", but "car" is not in the definition of "road". Therefore, (car–road) is a weak pair.

Some weak pairs can be promoted as strong pairs if the two words are among the K closest neighbours of each other. We chose the K closest words according to the cosine distance from a pretrained word embedding and find that using $K = 5$ is a good trade-off between semantic and syntactic extracted information.

3.2 Positive sampling

We introduce the concept of positive sampling based on strong and weak pairs. We move closer vectors of words forming either a strong or a weak pair in addition to moving vectors of words co-occurring within the same window.

Let $\mathcal{S}(w)$ be the set of all words forming a strong pair with the word w and $\mathcal{W}(w)$ be the set of all words forming a weak pair with w. For each target w_t from the corpus, we build $\mathcal{V}_s(w_t)$ a random set of n_s words drawn with replacement from $\mathcal{S}(w_t)$ and $\mathcal{V}_w(w_t)$ a random set of n_w words drawn with replacement from $\mathcal{W}(w_t)$. We compute the cost of positive sampling J_{pos} for each target as follows:

$$\begin{aligned} J_{pos}(w_t) = \ & \beta_s \sum_{w_i \in \mathcal{V}_s(w_t)} \ell(v_t \cdot v_i) \\ & + \beta_w \sum_{w_j \in \mathcal{V}_w(w_t)} \ell(v_t \cdot v_j) \end{aligned} \tag{4}$$

where ℓ is the logistic loss function defined by $\ell : x \mapsto \log(1 + e^{-x})$ and v_t (resp. v_i and v_j) is the vector associated to w_t (resp. w_i and w_j).

The objective is to minimize this cost for all targets, thus moving closer words forming a strong or a weak pair.

The coefficients β_s and β_w, as well as the number of drawn pairs n_s and n_w, tune the importance of strong and weak pairs during the learning phase. We discuss the choice of these hyperparameters in Section 5. When $\beta_s = 0$ and $\beta_w = 0$, our model is the Skip-gram model of Mikolov et al. (2013).

3.3 Controlled negative sampling

Negative sampling consists in considering two random words from the vocabulary $\mathcal{V}$ to be unrelated. For each word w_t from the vocabulary, we generate a set $\mathcal{F}(w_t)$ of k randomly selected words from the vocabulary :

$$\mathcal{F}(w_t) = \{w_i\}^k, w_i \in \mathcal{V} \setminus \{w_t\} \tag{5}$$

The model aims at separating the vectors of words from $\mathcal{F}(w_t)$ and the vector of w_t. More formally, this is equivalent to minimize the cost J_{neg} for each target word w_t as follows:

$$J_{neg}(w_t) = \sum_{w_i \in \mathcal{F}(w_t)} \ell(-v_t \cdot v_i) \tag{6}$$

where the notation ℓ, v_t and v_i are the same as described in previous subsection.

However, there is a non-zero probability that w_i and w_t are related. Therefore, the model will move their vectors further instead of moving them closer. With strong/weak word pairs in Dict2vec, it becomes possible to better ensure that this is less likely to occur: we prevent a negative example to be a word that forms a weak or strong pair with with w_t. The negative sampling objective from Equation 6 becomes :

$$J_{neg}(w_t) = \sum_{\substack{w_i \in \mathcal{F}(w_t) \\ w_i \notin \mathcal{S}(w_t) \\ w_i \notin \mathcal{W}(w_t)}} \ell(-v_t \cdot v_i) \tag{7}$$

In our experiments, we noticed this method discards around 2% of generated negative pairs. The influence on evaluation depends on the nature of the corpus and is discussed at Section 5.4.

3.4 Global objective function

Our objective function is derived from the noise-contrastive estimation which is a more efficient objective function than the log-likelihood in Equation 1 according to Mikolov et al. (2013). We add the positive sampling and the controlled negative sampling described before and compute the cost for each (target,context) pair (w_t, w_c) from the corpus as follows:

$$J(w_t, w_c) = \ell(v_t \cdot v_c) + J_{pos}(w_t) + J_{neg}(w_t) \tag{8}$$

The global objective is obtained by summing every pair's cost over the entire corpus :

$$J = \sum_{t=1}^{C} \sum_{c=-n}^{n} J(w_t, w_{t+c}) \tag{9}$$

4 Experimental setup

4.1 Fetching online definitions

We extract all unique words with more than 5 occurrences from a full Wikipedia dump, representing around 2.2M words. Since there is no dictionary that contains a definition for all existing words (the word w might be in the dictionary D_i but not in D_j), we combine several dictionaries to get a definition for almost all of these words (some words are too rare to have a definition anyway). We use the English version of Cambridge, Oxford, Collins and dictionary.com. For each word, we download the 4 different webpages, and use regex to extract the definitions from the HTML template specific to each website, making the process fully accurate. Our approach does not focus on polysemy, so we concatenate all definitions for each word. Then we concatenate results from all dictionaries, remove stop words and punctuation and lowercase all words. For our illustrative example in Section 3, we obtain :

> **car**: road vehicle engine wheels seats small [...] platform lift.

Among the 2.2M unique words, only 200K does have a definition. We generate strong and weak pairs from the downloaded definitions according to the rule described in subsection 3.1 leading to 417K strong pairs (when the parameter K from 3.1 is set to 5) and 3.9M weak pairs.

4.2 Training settings

We train our model with the generated pairs from subsection 4.1 and the November 2016 English dump from Wikipedia [4]. After removing all XML tags and converting all words to lowercase (with the help of Mahoney's script[5]), we separate the corpus into 3 files containing respectively the first 50M tokens, the first 200M tokens, and the full dump. Our model uses additional knowledge during training. For a fair comparison against other frameworks, we also incorporate this information into the training data and create two versions for each file : one containing only data from Wikipedia (corpus A) and one with data from Wikipedia concatenated with the definitions extracted (corpus B).

We use the same hyperparameters we usually find in the literature for all models. We use 5 negatives samples, 5 epochs, a window size of 5, a vector size of 100 (resp. 200 and 300) for the 50M file (resp. 200M and full dump) and we remove the words with less than 5 occurrences. We follow the same evaluation protocol as Word2vec and fastText to provide the fairest comparison against competitors, so every other hyperparameters (K, β_s, β_w, n_s, n_w) are tuned using a grid search to maximize the weighted average score. For n_s and n_w, we go from 0 to 10 with a step of 1 and find the optimal values to be $n_s = 4$ and $n_w = 5$. For β_s and β_w we go from 0 to 2 with a step of 0.05 and find $\beta_s = 0.8$ and $\beta_w = 0.45$ to be the best values for our model. Table 1 reports training times for the three models (all experiments were run on a E3-1246 v3 processor).

	50M	200M	Full
Word2vec	15m30	86m	2600m
fastText	8m44	66m	1870m
Dict2vec	4m09	26m	642m

Table 1: Training time (in min) of Word2vec, fastText and Dict2vec models for several corpus.

4.3 Word similarity evaluation

We follow the standard method for word similarity evaluation by computing the Spearman's rank correlation coefficient (Spearman, 1904) between human similarity evaluation of pairs of words, and the cosine similarity of the corresponding word vectors. A score close to 1 indicates an embedding close to the human judgement.

We use MC-30 (Miller and Charles, 1991), MEN (Bruni et al., 2014), MTurk-287 (Radinsky et al., 2011), MTurk-771 (Halawi et al., 2012),

[4] https://dumps.wikimedia.org/enwiki/20161101/

[5] http://mattmahoney.net/dc/textdata#appendixa

RG-65 (Rubenstein and Goodenough, 1965), RW (Luong et al., 2013), SimVerb-3500 (Gerz et al., 2016), WordSim-353 (Finkelstein et al., 2001) and YP-130 (Yang and Powers, 2006) classic datasets. We follow the same protocol used by Word2vec and fastText by discarding pairs which contain a word that is not in our embedding. Since all models are trained with the same corpora, the embeddings have the same words, therefore all competitors share the same OOV rates.

We run each experiment 3 times and report in Table 2 the average score to minimize the effect of the neural network random initialization. We compute the average by weighting each score by the number of pairs evaluated in its dataset in the same way as Iacobacci et al. (2016). We multiply each score by $1,000$ to improve readability.

4.4 Text classification evaluation

Our text classification task follows the same setup as the one for fastText in Joulin et al. (2016). We train a neural network composed of a single hidden layer where the input layer corresponds to the bag of words of a document and the output layer is the probability to belong to each label. The weights between the input and the hidden layer are initialized with the generated embeddings and are fixed during training, so that the evaluation score solely depends on the embedding. We update the weights of the neural network classifier with gradient descent. We use the datasets AG-News [6], DBpedia (Auer et al., 2007) and Yelp reviews (polarity and full)[7]. We split each datasets into a training and a test file. We use the same training and test files for all models and report the classification accuracy obtained on the test file.

4.5 Baselines

We train Word2vec[8] and fastText[9] on the same 3 files and their 2 respective versions (A and B) described in 4.2 and use the same hyperparameters also described in 4.2 for all models. We train Word2vec with the Skip-gram model since our method is based on the Skip-gram model. We also train GloVe with their respective hyperparameters described in Pennington et al. (2014), but the results are lower than all other baselines (weighted average on word similarity task is 350 on the 50M file, 389 on the 200M file and 454 on the full dump) so we do no report GloVe's results.

We also retrofit the learned embeddings on corpus A with the Faruqui's method to compare another method using additional resources. The retrofitting introduces external knowledge from the WordNet semantic lexicon (Miller, 1995). We use the **Faruqui's Retrofitting**[10] with the WN_{all} semantic lexicon from WordNet and 10 iterations as advised in the paper of Faruqui et al. (2015). Furthermore, we compare the performance of our method when using WordNet additional resources instead of dictionaries.

5 Results and model analysis

5.1 Semantic similarity

Table 2 (top) reports the Spearman's rank correlation scores obtained with the method described in subsection 4.3. We observe that our model outperforms state-of-the-art approaches for most of the datasets on the 50M and 200M tokens files, and almost all datasets on the full dump (this is significant according to a two-sided Wilcoxon signed-rank test with $\alpha = 0.05$). With the weighted average score, our model improves fastText's performance on raw corpus (column A) by 28.3% on the 50M file, by 17.7% on the 200M and by 12.8% on the full dump. Even when we train fastText with the same additional knowledge as ours (column B), our model improves performance by 2.9% on the 50M file, by 5.1% in the 200M and by 11.9% on the full dump.

We notice the column B (corpus composed of Wikipedia and definitions) has better results than the column A for the 50M ($+24\%$ on average) and the 200M file ($+12\%$ on average). This demonstrates the strong semantic relations one can find in definitions, and that simply incorporating definitions in small training file can boost the performance of the embeddings. Moreover, when the training file is large (full dump), our supervised method with pairs is more efficient, as the boost brought by the concatenation of definitions is insignificant ($+1.5\%$ on average).

We also note that the number of strong and weak pairs drawn must be set according to the size of the training file. For the 50M and 200M tokens files, we train our model with hyperparameters $n_s = 4$ and $n_w = 5$. For the full dump (20

[6] https://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

[7] https://www.yelp.com/dataset_challenge

[8] https://github.com/dav/word2vec

[9] https://github.com/facebookresearch/fastText

[10] https://github.com/mfaruqui/retrofitting

	50M							200M							Full						
		w2v		FT		our			w2v		FT		our			w2v		FT		our	
	oov	A	B	A	B	A	B	oov	A	B	A	B	A	B	oov	A	B	A	B	A	B
MC-30	0%	697	847	722	823	840	**859**	0%	742	830	795	814	**854**	827	0%	809	826	831	815	**860**	847
MEN-TR-3k	0%	692	753	697	**767**	733	762	0%	734	758	754	**772**	752	768	0%	733	728	752	751	**756**	755
MTurk-287	0%	657	**688**	657	685	665	682	0%	642	**671**	671	661	667	666	0%	660	656	**672**	671	661	660
MTurk-771	0%	596	677	597	692	685	**713**	0%	628	669	632	675	682	**704**	0%	623	620	631	638	**696**	694
RG-65	0%	714	865	671	842	824	**875**	0%	771	842	755	829	857	**877**	0%	787	802	817	820	**875**	867
RW	36%	375	420	442	**512**	475	489	16%	377	408	475	**507**	467	478	2%	407	427	464	468	**482**	476
SimVerb	3%	165	371	179	374	363	**432**	0%	183	306	206	329	377	**424**	0%	186	214	222	233	**384**	379
WS353-ALL	0%	660	739	657	739	738	**753**	0%	694	734	701	735	**762**	758	0%	705	721	729	723	756	**758**
WS353-REL	0%	619	**700**	623	696	679	688	0%	665	706	644	685	**710**	699	0%	664	681	687	686	702	**703**
WS353-SIM	0%	714	**797**	714	790	774	784	0%	743	**792**	758	792	784	787	0%	757	767	775	779	781	**781**
YP-130	3%	458	679	415	674	666	**696**	0%	449	592	509	639	616	**665**	0%	502	475	533	553	**646**	607
W.Average		453	562	467	582	564	**599**		471	533	503	563	569	**592**		476	488	508	512	**573**	570
AG-News		**874**	871	868	871	871	866		**886**	882	880	881	880	880		885	885	**887**	887	881	884
DBPedia		936	942	942	944	**944**	944		952	956	957	958	**960**	959		966	966	967	967	968	**969**
Yelp Pol.		808	835	821	**842**	832	834		837	855	852	859	856	**859**		865	867	872	874	**876**	875
Yelp Full		451	469	460	**473**	471	472		477	491	488	495	499	**501**		506	506	512	514	516	**518**

Table 2: Spearman's rank correlation coefficients between vectors' cosine similarity and human judgement for several datasets (top) and accuracies on text classification task (bottom). We train and evaluate each model 3 times and report the average score for each dataset, as well as the weighted average for all word similarity datasets.

		50M			200M			Full		
		w2v$_R$	FT$_R$	our$_R$	w2v$_R$	FT$_R$	our$_R$	w2v$_R$	FT$_R$	our$_R$
MC-30	vs self	+13.9%	+9.2%	+1.3%	+5.8%	+4.8%	+3.0%	+5.2%	+2.9%	+1.2%
	vs our	**-7.3%**	**-4.4%**	–	**-3.6%**	**-2.4%**	–	**-1.0%**	**-0.6%**	–
MEN-TR-3k	vs self	+0.9%	-0.7%	-0.1%	+0.7%	-1.9%	+0.4%	+1.4%	-2.8%	+1.6%
	vs our	**-4.2%**	**-7.4%**	–	**-1.3%**	**-1.6%**	–	**-1.7%**	**-3.3%**	–
MTurk-287	vs self	+1.4%	+0.2%	+3.5%	-2.9%	-3.3%	+3.0%	-0.9%	-5.7%	+1.1%
	vs our	**-1.2%**	**-4.0%**	–	**-4.3%**	**-2.7%**	–	**-1.1%**	**-4.1%**	–
MTurk-771	vs self	+8.2%	+4.9%	+1.6%	+6.3%	+4.3%	+1.5%	+4.5%	+1.6%	+0.6%
	vs our	**-7.3%**	**-8.8%**	–	**-3.8%**	**-3.4%**	–	**-6.5%**	**-7.9%**	–
RG-65	vs self	+10.9%	+17.1%	+4.0%	+6.6%	+8.5%	+3.0%	+7.0%	+5.0%	+2.4%
	vs our	**-2.1%**	**-4.9%**	–	**-2.2%**	**-4.4%**	–	**-3.8%**	**-1.9%**	–
RW	vs self	-20.3%	-24.4%	-14.3%	-24.4%	-25.9%	-20.3%	-18.7%	-25.4%	-19.1%
	vs our	**-37.4%**	**-26.9%**	–	**-37.7%**	**-24.6%**	–	**-31.3%**	**-28.2%**	–
Simverb	vs self	+46.0%	+34.0%	+19.8%	+49.7%	+39.8%	+19.9%	+44.6%	+38.7%	+16.7%
	vs our	**-34.4%**	**-28.4%**	–	**-30.5%**	**-23.6%**	–	**-29.9%**	**-19.8%**	–
WS353-ALL	vs self	-4.2%	-10.8%	-1.1%	-3.2%	-8.0%	-1.3%	-4.4%	-10.7%	-2.0%
	vs our	**-13.8%**	**-19.2%**	–	**-11.9%**	**-15.4%**	–	**-10.8%**	**-13.9%**	–
WS353-REL	vs self	-16.1%	-22.7%	-4.9%	-10.4%	-17.9%	-4.5%	-10.7%	-19.7%	-6.0%
	vs our	**-20.9%**	**-27.2%**	–	**-17.7%**	**-25.5%**	–	**-15.5%**	**-21.4%**	–
WS353-SIM	vs self	+4.3%	+0.8%	+3.2%	+2.6%	+0.0%	+3.2%	+0.0%	-3.6%	+2.4%
	vs our	**-3.9%**	**-6.7%**	–	**-2.9%**	**-3.3%**	–	**-3.0%**	**-4.4%**	–
YP-130	vs self	+17.8%	+3.2%	+3.3%	+13.0%	+6.9%	+8.0%	+11.1%	+8.3%	+5.0%
	vs our	**-23.6%**	**-28.2%**	–	**-16.6%**	**-11.7%**	–	**-13.6%**	**-10.7%**	–

Table 3: Percentage changes of word similarity scores for several datasets after the Faruqui's retrofitting method is applied. We compare each model to their own non-retrofitted version (vs self) and our non-retrofitted version (vs our). A positive percentage indicates the level of improvement of the retrofitting approach, while a negative percentage shows that the compared method is better without retrofitting. As an illustration: the +13.9% at the top left means that retrofitting Word2vec's vectors improves the initial vectors output by 13.9%, while the -7.3% below indicates that our approach without retrofitting is better than the retrofitted Word2vec's vectors.

times larger than the 200M tokens file), the number of windows in the corpus is largely increased, so is the number of (target,context) pairs. Therefore, we need to adjust the influence of strong and weak pairs and decrease n_s and n_w. We set $n_s = 2$, $n_w = 3$ to train on the full dump.

The Faruqui's retrofitting method improves the word similarity scores on all frameworks for all datasets, except on RW and WS353 (Table 3). But even when Word2vec and fastText are retrofitted, their scores are still worse than our non-retrofitted model (every percentage on the *vs our* line are negative). We also notice that our model is compatible with a retrofitting improvement method as our scores are also increased with Faruqui's method.

We also observe that, although our model is superior on each corpus size, our model trained on the 50M tokens file outperforms the other models trained on the full dump (an improvement of 17% compared to the results of fastText, our best competitor, trained on the full dump). This means considering strong and weak pairs is more efficient than increasing the corpus size and that using dictionaries is a good way to improve the quality of the embeddings when the training file is small.

The models based on knowledge bases cited in §2.2 do not provide word similarity scores on all the datasets we used. However, for the reported scores, Dict2vec outperforms these models : Kiela et al. (2015) achieves a correlation of 0.72 on the MEN dataset (vs. 0.756); Xu et al. (2014) achieves 0.683 on the WS353-ALL dataset (vs. 0.758).

5.2 Text classification accuracy

Table 2 (bottom) reports the classification accuracy for the considered datasets. Our model achieves the same performances as Word2vec and fastText on the 50M file and slightly improves results on the 200M file and the full dump. Using supervision with pairs during training does not make our model specific to the word similarity task which shows that our embeddings can also be used in downstream extrinsic tasks.

Note that for this experiment, the embeddings were fixed and not updated during learning (we only learned the classifier parameters) since our objective was rather to evaluate the capability of the embeddings to be used for another task rather than obtaining the best possible models. It is anyway possible to obtain better results by updating the embeddings and the classifier parameters with respect to the supervised information to adapt the embeddings to the classification task at hand as done in Joulin et al. (2016).

5.3 Dictionaries vs. WordNet

		Raw	R_{WN}	R_{dict}
50M	w2v	453	474	**479**
	FT	467	476	**489**
	our	569	582	**582**
200M	w2v	471	488	**494**
	FT	503	504	**524**
	our	569	581	**587**
full	w2v	488	507	**512**
	FT	508	503	**529**
	our	571	583	**592**

Table 4: Weighted average Spearman correlation score of raw vectors and after retrofitting with WordNet pairs (R_{WN}) and dictionary pairs (R_{dict}).

Table 4 reports the Spearman's rank correlation score for vectors obtained after training (Raw column) and the scores after we retrofit those vectors with pairs from WordNet (R_{WN}) and extracted pairs from dictionaries (R_{dict}). Retrofitting with dictionaries outperforms retrofitting with WordNet lexicons, meaning that data from dictionaries are better suited to improve embeddings toward semantic similarities when retrofitting.

	50M	200M	full
No pairs	453	471	488
With WordNet pairs	564	566	559
With dictionary pairs	**569**	**569**	**571**

Table 5: Weighted average Spearman correlation score of Dict2vec vectors when trained without pairs and with WordNet or dictionary pairs.

We also trained Dict2vec with pairs from WordNet as well as no additional pairs during training (in this case, this is the Skip-gram model from Word2vec). Results are reported in Table 5. Training with WordNet pairs increases the scores, showing that the supervision brought by positive sampling is beneficial to the model, but lags behind the training using dictionary pairs demonstrating once again that dictionaries contain more semantic information than WordNet.

5.4 Positive and negative sampling

For the positive sampling, an empirical grid search shows that a $\frac{1}{2}$ ratio between β_s and β_w is a good rule-of-thumb for tuning these hyperparameters. We also notice that when these coefficients are too low ($\beta_s \leq 0.5$ and $\beta_w \leq 0.2$), results get worse because the model does not take into account the information from the strong and weak pairs. On the other side, when they are too high ($\beta_s \geq 1.2$ and $\beta_w \geq 0.6$), the model discards too much the information from the context in favor of the information from the pairs. This behaviour is similar when the number of strong and weak pairs is too low or too high ($n_s, n_w \leq 2$ or $n_s, n_w \geq 5$).

For the negative sampling, we notice that the control brought by the pairs increases the average weighted score by 0.7% compared to the uncontrolled version. We also observe that increasing the number of negative samples does not significantly improve the results except for the RW dataset where using 25 negative samples can boost performances by 10%. Indeed, this dataset is mostly composed of rare words so the embeddings must learn to differentiate unrelated words rather than moving closer related ones.

5.5 Vector size

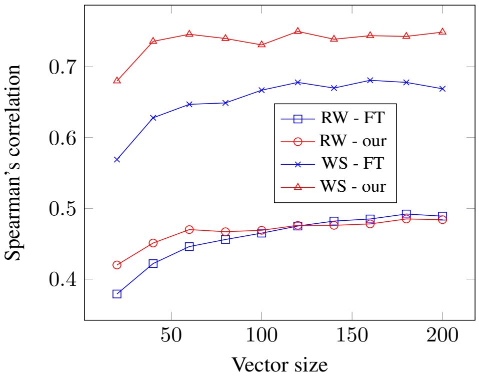


Figure 1: Spearman's rank correlation coefficient for RW-STANFORD (RW) and WS-353-ALL (WS) on the fastText model (FT) and our, with different vector size. Training is done on the corpus A of 50M tokens.

In Fig. 1, we observe that our model is still able to outperform state-of-the-art approaches when we reduce the dimension of the embeddings to 20 or 40. We also notice that increasing the vector size does increase the performance, but only until a dimension around 100, which is the common dimension used when training on the 50M tokens file for related approaches reported here.

6 Conclusion

In this paper, we presented `Dict2vec`, a new approach for learning word embeddings using lexical dictionaries. It is based on a Skip-gram model where the objective function is extended by leveraging word pairs extracted from the definitions weighted differently with respect to the strength of the pairs. Our approach shows better results than state-of-the-art word embeddings methods for the word similarity task, including methods based on a retrofitting from external sources. We also provide the full source code to reproduce the experiments.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. *The semantic web* pages 722–735.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pages 132–148.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *J. Artif. Intell. Res.(JAIR)* 49(1-47).

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM, pages 160–167.

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

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *ACL (1)*. Citeseer, pages 1370–1380.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, pages 1606–1615.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*. ACM, pages 406–414.

John Rupert Firth. 1957. *Papers in Linguistics 1934-1951: Repr*. Oxford University Press.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. *arXiv preprint arXiv:1608.00869* .

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 1406–1414.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. Wikireading: A novel large-scale language understanding task over wikipedia. *arXiv preprint arXiv:1608.03542* .

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. volume 1, pages 897–907.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* .

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

Wang Ling, Lin Chu-Cheng, Yulia Tsvetkov, and Silvio Amir. 2015. Not all contexts are created equal: Better word representations with variable attention. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1367–1372.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*. pages 104–113.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*. volume 2, page 3.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes* 6(1):1–28.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–43.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*. ACM, pages 337–346.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM* 8(10):627–633.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American journal of psychology* 15(1):72–101.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. pages 1017–1024.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *EMNLP*. Citeseer, pages 1591–1601.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. 2013. Connecting language and knowledge bases with embedding models for relation extraction. *arXiv preprint arXiv:1307.7973* .

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, pages 1219–1228.

Dongqiang Yang and David MW Powers. 2006. *Verb similarity on the taxonomy of WordNet*. Masaryk University.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *ACL (2)*. pages 545–550.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *ACL (1)*. pages 1127–1137.

Learning Chinese Word Representations From Glyphs Of Characters

Tzu-Ray Su and **Hung-Yi Lee**
Dept. of Electrical Engineering, National Taiwan University
No. 1, Sec. 4, Roosevelt Road, Taipei, Taiwan
{b01901007,hungyilee}@ntu.edu.tw

Abstract

In this paper, we propose new methods to learn Chinese word representations. Chinese characters are composed of graphical components, which carry rich semantics. It is common for a Chinese learner to comprehend the meaning of a word from these graphical components. As a result, we propose models that enhance word representations by character glyphs. The character glyph features are directly learned from the bitmaps of characters by convolutional auto-encoder(convAE), and the glyph features improve Chinese word representations which are already enhanced by character embeddings. Another contribution in this paper is that we created several evaluation datasets in traditional Chinese and made them public.

1 Introduction

No matter which target language it is, high quality word representations (also known as word "embeddings") are keys to many natural language processing tasks, for example, sentence classification (Kim, 2014), question answering (Zhou et al., 2015), machine translation (Sutskever et al., 2014), etc. Besides, word-level representations are building blocks in producing phrase-level (Cho et al., 2014) and sentence-level (Kiros et al., 2015) representations.

In this paper, we focus on learning Chinese word representations. A Chinese word is composed of characters which contain rich semantics. The meaning of a Chinese word is often related to the meaning of its compositional characters. Therefore, Chinese word embedding can be enhanced by its compositional character embeddings (Chen et al., 2015; Xu et al., 2016). Furthermore, a Chinese character is composed of several graphical components. Characters with the same component share similar semantic or pronunciation. When a Chinese user encounters a previously unseen character, it is instinctive to guess the meaning (and pronunciation) from its graphical components, so understanding the graphical components and associating them with semantics help people learning Chinese. Radicals[1] are the graphical components used to index Chinese characters in a dictionary. By identifying the radical of a character, one obtains a rough meaning of that character, so it is used in learning Chinese word embedding (Yin et al., 2016) and character embedding (Sun et al., 2014; Li et al., 2015). However, other components in addition to radicals may contain potentially useful information in word representation learning.

Our research begins with a question: *Can machines learn Chinese word representations from glyphs of characters?* By exploiting the glyphs of characters as images in word representation learning, all the graphical components in a character are considered, not limited to radicals. In our proposed methods, we render character glyphs to fixed-size grayscale images which are referred to as "character bitmaps", as illustrated in Fig.1. A similar idea was also used in (Liu et al., 2017) to help classifying wikipedia article titles into 12 categories. We use a convAE to extract character features from the bitmap to represent the glyphs. It is also possible to represent the glyph of a character by the graphical components in it. We do not choose this way because there is no unique way to decompose a character, and directly learning representation from bitmaps is more straightforward. Then we use the models parallel to Skip-gram (Mikolov et al., 2013a) or GloVe (Penning-

[1] https://en.wikipedia.org/wiki/Radical_(Chinese_characters)

ton et al., 2014) to learn word representations from the character glyph features. Although we only consider traditional Chinese characters in this paper, and the examples given below are based on the traditional characters, the same ideas and methods can be applied on the simplified characters.

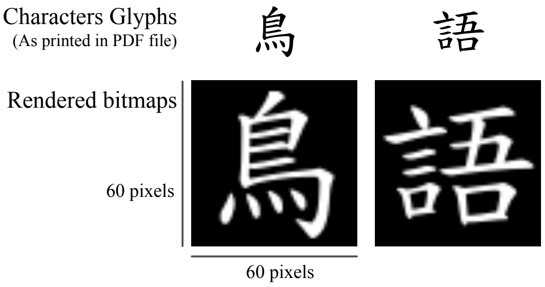


Figure 1: A Chinese character is represented as a fixed-size gray-scale image which is referred to as "character bitmap" in this paper.

2 Background Knowledge and Related Works

To give a clear illustration of our own work, we briefly introduce the representative methods of word representation learning in Section 2.1. In Section 2.2, we will introduce some of the linguistic properties of Chinese, and then introduce the methods that utilize these properties to improve word representations.

2.1 Word Representation Learning

Mainstream research of word representation is built upon the distributional hypothesis, that is, words with similar contexts share similar meanings. Usually a large-scale corpus is used, and word representations are produced from the co-occurrence information of a word and its context. Existing methods of producing word representations could be separated into two families (Levy et al., 2015): count-based family (Turney and Pantel, 2010; Bullinaria and Levy, 2007), and prediction-based family. Word representations can be obtained by training a neural-network-based models (Bengio et al., 2003; Collobert et al., 2011). The representative methods are briefly introduced below.

2.1.1 CBOW and Skipgram

Both continuous bag-of-words (CBOW) model and Skipgram model train with words and contexts in a sliding local context window (Mikolov et al., 2013a). Both of them assign each word w_i with an embedding $\vec{w}_i$. CBOW predicts the word given its context embeddings, while Skipgram predicts contexts given the word embedding. Predicting the occurrence of word/context in CBOW and Skipgram models could be viewed as learning a multi-class classification neural network (the number of classes is the size of vocabulary). In (Mikolov et al., 2013b), the authors introduced several techniques to improve the performance. Negative sampling is introduced to speed up learning, and subsampling frequent words is introduced to randomly discard training examples with frequent words (such as "the", "a", "of"), and has an effect similar to the removal of stop words.

2.1.2 GloVe

Instead of using local context windows, (Pennington et al., 2014) proposed GloVe model. Training GloVe word representations begins with creating a co-occurrence matrix X from a corpus, where each matrix entry X_{ij} represents the counts that word w_j appears in the context of word w_i. In (Pennington et al., 2014), the authors used a harmonic weighting function for co-occurrence count, that is, word-context pairs with distance d contributes $\frac{1}{d}$ to the global co-occurrence count.

Let $\vec{w}_i$ be the word representation of word w_i, and $\tilde{\vec{w}}_j$ be the word representation of word w_j as context, GloVe model minimizes the loss:

$$\sum_{i,j\in \substack{non-zero\\ entries\ of\ X}} f(X_{ij})(\vec{w}_i^T\tilde{\vec{w}}_j+b_i+\tilde{b}_j-log(X_{ij})),$$

where b_i is the bias for word w_i, and $\tilde{b}_j$ is the bias for context w_j. A weighting function $f(X_{ij})$ is introduced because the authors consider rare co-occurrence word-context pairs carry less information than frequent ones, and their contributions to the total loss should be decreased. The weighting function $f(X_{ij})$ is defined as below. It depends on the co-occurrence count, and the authors set parameters $x_{max} = 100$, $\alpha = 0.75$.

$$f(X_{ij}) = \begin{cases} (\frac{X_{ij}}{x_{max}})^{\alpha} & \text{if } X_{ij} < x_{max} \\ 1 & \text{otherwise} \end{cases}$$

In the GloVe model, each word has 2 representations $\vec{w}$ and $\tilde{\vec{w}}$. The authors suggest using $\vec{w} + \tilde{\vec{w}}$ as the word representation, and reported improvements over using $\vec{w}$ only.

2.2 Improving Chinese Word Representation Learning

2.2.1 The Chinese Language

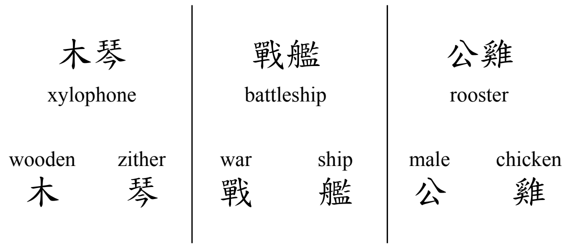


Figure 2: Examples of compositional Chinese words. Still, the reader should keep in mind that NOT all Chinese words are compositional (related to the meanings of its compositional characters).

A Chinese word is composed of a sequence of characters. The meanings of some Chinese words are related to the composition of the meanings of their characters. For example, "戰艦" (battleship), is composed of two characters, "戰" (war) and "艦" (ship). More examples are given in Fig. 2. To improve Chinese word representations with sub-word information, character-enhanced word embedding (CWE) (Chen et al., 2015) in Section 2.2.2 is proposed.

(A) Radicals:		虫	木	氵(水)
(B) Semantics:		anthropods, reptiles	plants, wooden materials	water, liquid
(C) Characters:	(C-1)	蝶 (butterfly)	棉 (cotton)	海 (sea)
	(C-2)	蜂 (bee)	楓 (maple)	溪 (river)
	(C-3)	蚊 (mosquito)	棍 (stick)	湯 (soup)
	(C-4)	蛇 (snake)	梅 (plum)	淚 (tear)
	(C-5)	蟹 (crab)	果 (fruit)	泉 (spring)

Figure 3: Some examples of radicals and the characters containing them. In rows (C-1) to (C-4), the radicals are at the left hand side of the character, while in row (C-5), the radicals are at the bottom, and may have different of shapes.

A Chinese character is composed of several graphical components. Characters with the same component share similar semantic or phonetic properties. In a Chinese dictionary characters with similar coarse semantics are grouped into categories for the ease of searching. The common graphical component which relates to the common semantic is chosen to index the category, known as a radical. Examples are given in Fig. 3. There are three radicals in row (A), and their semantic meanings are in row (B). In each column, there are five characters containing each radical. It is easy to find that the characters having the same radical have meanings related to the radical in some aspect. A radical can be put in different positions in a character. For example, in rows (C-1) to (C-4), the radicals are at the left hand side of a character, but in row (C-5), the radicals are at the bottom. The shape of a radical can be different in different positions. For example, the third radical which represents "water" or "liquid" has different forms when it is at the left hand side or the bottom of a character. Because radicals serve as a strong semantic indicator of a character, multi-granularity embedding (MGE) (Yin et al., 2016) in Section 2.2.3 incorporates radical embeddings in learning word representation.

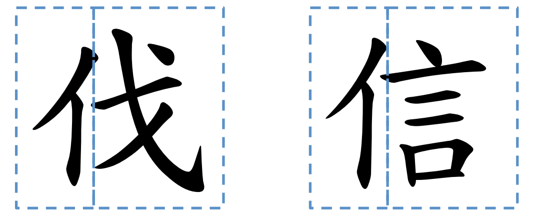


Figure 4: Both characters in the figure have the same radical "亻" (means humans) at the left hand side, but their meanings are the composition of the graphical components at the right hand side and their radical.

Usually the components other than radicals determine the pronunciation of the characters, but in some cases they also influence the meaning of a character. Two examples are given in Fig. 4[2]. Both characters in Fig. 4 have the same radical "亻" (means humans) at the left hand side, but the graphical components at the right hand side also have semantic meanings related to the characters. Considering the left character "伐" (means attack). Its right component "戈" means "weapon", and the meaning of the character "伐" is the composition of the meaning of its two components (a human with a weapon). None of the previous word embedding approach considers all the components of Chinese characters in our best knowledge.

[2] The two example characters here have the same glyphs in the traditional and simplified Chinese characters.

2.2.2 Character-enhanced Word Embedding (CWE)

The main idea of CWE is that word embedding is enhanced by its compositional character embeddings. CWE predicts the word from both word and character embeddings of contexts, as illustrated in Fig. 5 (a). For word w_i, the CWE word embedding $\vec{w}_i^{cwe}$ has the following form:

$$\vec{w}_i^{cwe} = \vec{w}_i + \frac{1}{|C(i)|} \sum_{c_j \in C(i)} \vec{c}_j$$

where $\vec{w}_i$ is the word embedding, $\vec{c}_j$ is the embedding of the j-th character in w_i, and $C(i)$ is the set of compositional characters of word w_i. Mean value of CWE word embeddings of contexts are then used to predict the word w_i.

Sometimes one character has several different meanings, this is known as the ambiguity problem. To deal with this, each character is assigned with a bag of embeddings. During training, one of the embeddings is picked to form the modified word embedding. The authors proposed three methods to decide which embedding is picked: position-based, cluster-based, and non-parametric cluster-based character embeddings.

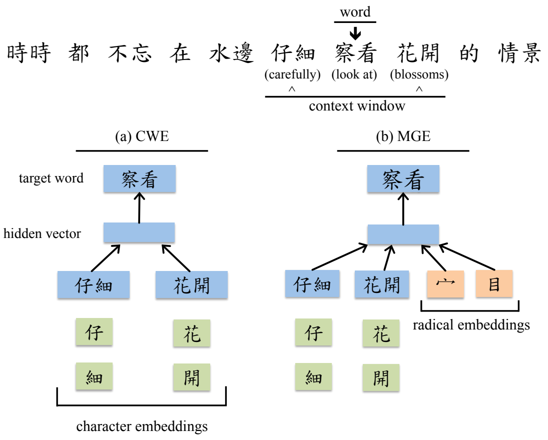


Figure 5: Model comparison of Character-enhanced Word Embedding (CWE) and Multi-granularity Embedding (MGE).

2.2.3 Multi-granularity Embedding (MGE)

Based on CBOW and CWE, (Yin et al., 2016) proposed MGE, which predicts target word with its radical embeddings and modified word embeddings of context in CWE, as shown in Fig.5 (b).

There is no ambiguity of radicals, so each radical is assigned with one embedding $\vec{r}$. We denote $\vec{r}_k$ as the radical embedding of character c_k. MGE predicts the target word w_i with the following hidden vector:

$$\vec{h}_i = \frac{1}{|C(i)|} \sum_{c_k \in C(i)} \vec{r}_k + \frac{1}{|W(i)|} \sum_{w_j \in W(i)} \vec{w}_j^{cwe}$$

, where W(i) is the set of contexts words of w_i, $\vec{w}_j^{cwe}$ is the CWE word embedding of w_j. MGE picks character embeddings with the position-based method in CWE, and picks radical embeddings according to a character-radical index built from a dictionary during training. When non-compositional word is encountered, only the word embedding is used to form $\vec{h}_i$.

3 Model

We first extract glyph features from bitmaps with the convAE in Section 3.1. The glyph features are used to enhance the existing word representation learning models in Section 3.2. In Section 3.3, we try to learn word representations directly from the glyph features.

3.1 Character Bitmap Feature Extraction

A convAE (Masci et al., 2011) is used to reduce the dimensions of rendered character bitmaps and capture high-level features. The architecture of the convAE is shown in Fig. 6. The convAE is composed of 5 convolutional layers in both encoder and decoder. The stride larger than one is used instead of pooling layers. Convolutional and deconvolutional layers on the same level share the same kernel. The input image is a 60×60 8-bit grayscale bitmap, and the encoder extracts 512-dimensional feature. The feature of character c_k from the encoder is refer to as character glyph feature $\vec{g}_k$ in the paper.

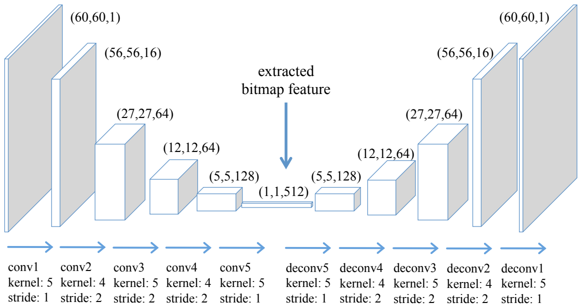


Figure 6: The architecture of convAE.

3.2 Glyph-Enhanced Word Embedding (GWE)

3.2.1 Enhanced by Context Word Glyphs

We modify CWE model based on CBOW in Section 2.2.2 to incorporate context character glyph features (ctxG). This modified word embedding $\vec{w}_i^{ctxG}$ of word w_i has the form:

$$\vec{w}_i^{ctxG} = \vec{w}_i + \frac{1}{|C(i)|} \sum_{c_j \in C(i)} (\vec{c}_j + \vec{g}_j),$$

where $C(i)$ is the compositional characters of w_i and $\vec{g}_j$ is the glyph feature of c_j. The model predicts target word w_i from ctxG word embeddings of contexts, as shown in Fig.7. The parameters in the convAE are pre-trained, thus not jointly learned with embeddings $\vec{w}$ and $\vec{c}$, so character glyph features $\vec{g}$ are fixed during training.

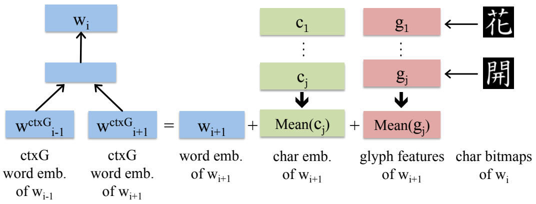


Figure 7: Illustration of exploiting context word glyphs. Mean value of character glyph features in the context is added to the hidden vector that predicts target word.

3.2.2 Enhanced by Target Word Glyphs

Here we propose another variant. In this model, the model structure is the same as in Fig.7. The difference lies in the hidden vector used to predict the target word. Instead of adding mean value of character glyph features of the contexts, it adds mean value of glyph feature of the target word (tarG), as shown in Fig.8. As in Section 3.2.1, convAE is not jointly learned.

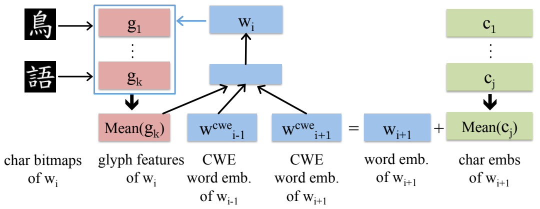


Figure 8: Illustration of exploiting target word glyphs. Mean value of character glyph features of target words help predicting target word itself.

3.3 Directly Learn From Character Glyph Features

3.3.1 RNN-Skipgram

We learn word representation $\vec{w}_i$ directly from the sequence of character glyph features $\{\vec{g}_k, c_k \in C(i)\}$ of word w_i, with the objective of Skipgram. As in Fig.9, a 2-layer Gated Recurrent Units (GRU) (Cho et al., 2014) network followed by 2 fully connected ELU (Clevert et al., 2015) layers produces word representation $\vec{w}_i$ from input sequence $\{\vec{g}_k\}$ of word w_i. $\vec{w}_i$ is then used to predict the contexts of w_i. In the training we use negative sampling and subsampling on frequent words from (Mikolov et al., 2013b).

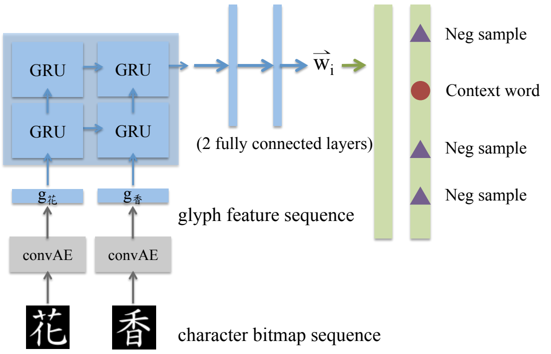


Figure 9: Model architecture of RNN-Skipgram model. Produced word representation $\vec{w}_i$ is used to predict the context of word w_i.

3.3.2 RNN-GloVe

We modify GloVe model to directly learn from character glyph features as in Fig.10. We feed character glyph feature sequence $\{\vec{g}_k, c_k \in C(i)\}$, $\{\vec{g}_{k'}, c_{k'} \in C(j)\}$ of word w_i and context w_j to a shared GRU network. Outputs of GRU are then fed to two different fully connected ELU layers to produce word representations $\vec{w}_i$ and $\vec{\tilde{w}}_j$. The inner product of $\vec{w}_i$ and $\vec{\tilde{w}}_j$ is the prediction of log co-occurrence $log(X_{ij})$. We apply the same loss function with weights in GloVe. We follow (Pennington et al., 2014) and use $\vec{w}_i + \vec{\tilde{w}}_i$ for evaluations of word representation.

4 Experimental Setup

4.1 Preprocessing

We learned word representations with traditional Chinese texts from Central News Agency daily newspapers from 1991 to 2002 (Chinese Giga-

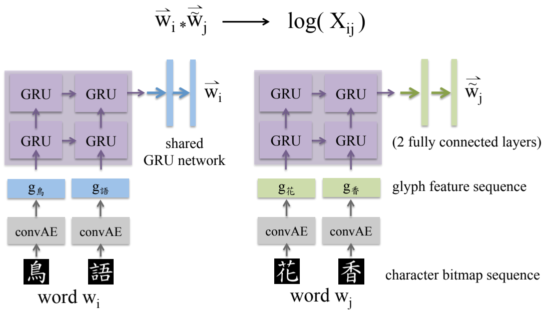


Figure 10: Model architecture of RNN-GloVe. A shared GRU network and 2 different sets of fully connected ELU layers produce $\vec{w}_i$ and $\vec{\tilde{w}}_j$. Inner product of $\vec{w}_i$ and $\vec{\tilde{w}}_j$ is the prediction of log co-occurrence $log(X_{ij})$.

word, LDC2003T09). All foreign words, numerical words, and punctuations were removed. Word segmentation was performed using open source python package `jieba`[3]. In all 316,960,386 segmented words, we extracted 8780 unique characters, and used a true type font (BiauKai) to render each character glyph to a 60×60 8-bit grayscale bitmap. Furthermore, We removed words whose frequency $<=$ 25, leaving 158,565 unique words as the vocabulary set.

4.2 Extracting Visual Features of Character Bitmap

Inspired by (Zeiler et al., 2011), layer-wise training was applied to our convAE. From lower level to higher, the kernel of each layer is trained individually, with other kernels frozen for 100 epochs. Loss function is the Euclidean distance between input and reconstructed bitmap, and we added $l1$ regularization to the activations of convolution layers. We chose Adagrad as the optimizing algorithm, and set batch size $=$ 20 and learning rate $= 0.001$.

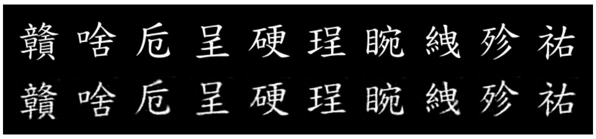


Figure 11: The input bitmaps of convAE and their reconstructions. The input bitmaps are in the upper row, while the reconstructions are in the lower row.

The comparison between the input bitmaps and their reconstructions is shown in Fig 11. The input bitmaps are in the upper row, while the reconstructions are in the lower row. We further visualized the extracted character glyph features with t-SNE (Maaten and Hinton, 2008). Part of the visualization result is shown in Fig. 12. From Fig. 12, we found that the characters with the same components are clustered. The result shows that the features extracted by the convAE are capable of expressing the graphical information in the bitmaps.

4.3 Training Details of Word Representations

We used CWE code[4] to implement both CBOW and Skipgram, along with the CWE. The number of multi-embedding was set to 3. We modified the CWE code to produce GWE representations. For CBOW, Skipgram, CWE, GWE and RNN-Skipgram, we used the following hyperparameters. Context window was set to 5 to both sides of a word. We used 10 negative samples, and threshold t of subsampling was set to 10^{-5}.

Since Yin at al. did not publish their code, we followed their paper and reproduced the MGE model. We created the mapping between characters and radicals from the Unihan database[5]. Each character corresponds to one of the 214 radicals in this dataset, and the same hyperparameters were used in training as above. Note that we did not separate non-compositional words during training as the original CWE and MGE did.

We used the GloVe code[6] to train the baseline GloVe vectors. In construction of co-occurrence matrix for GloVe and RNN-GloVe, we followed the parameter settings of $x_{max} = 100$ and $\alpha = 0.75$ in (Pennington et al., 2014). Context window was 5 words to the both sides of a word, and harmonic weighting was used on co-occurrence counts. For the RNN-GloVe model, we removed entries whose value < 0.5 to speed up training.

RNN-Skipgram and RNN-GloVe generated 200-dimensional word embeddings, while other models generated 512-dimensional word embeddings.

To encourage further research, we published our convAE and embedding models on github[7]. Evaluation datasets were also uploaded, whose details will be explained in Section 5.

[3] `https://github.com/fxsjy/jieba`

[4] `https://github.com/Leonard-Xu/CWE`

[5] `http://unicode.org/charts/unihan.html`

[6] `https://github.com/stanfordnlp/GloVe`

[7] `https://github.com/ray1007/GWE`

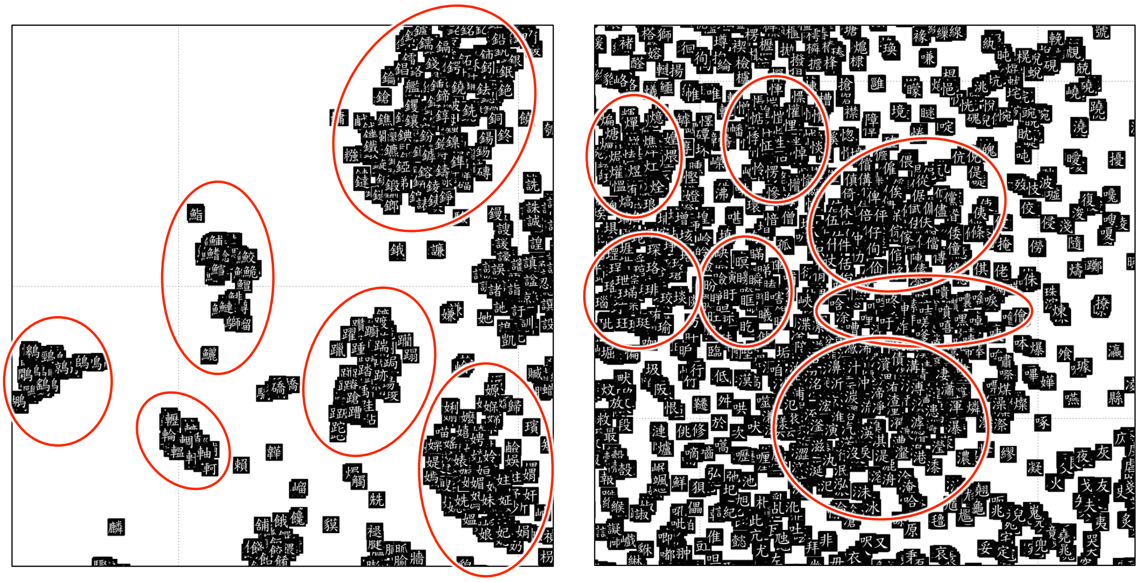


Figure 12: Parts of t-SNE visualization of character glyph features. Most of the characters in the ovals share the same components.

5 Evaluation

5.1 Word Similarity

A word similarity test contains multiple word pairs and their human annotated similarity scores. Word representations are considered good if the calculated similarity and human annotated scores have a high rank correlation. We computed the Spearman's correlation between human annotated scores and cosine similarity of word representations.

Since there is little resource for traditional Chinese, we translated WordSim-240 and WordSim-296 datasets provided by (Chen et al., 2015). Note that this translation is non-trivial. Some frequent words are considered out-of-vocabulary (OOV) due to the different usage between the simplified and traditional. For example, “butter” is translated to “黃油” in simplified, but “奶油” in traditional. Besides, we manually translated SimLex-999 (Hill et al., 2016) to traditional Chinese, and used it as the third testing dataset. We also made these datasets public along with our code.

When calculating similarities, word pairs containing OOVs were removed. In Table 1, there are only 237, 284 and 979 word pairs left in WordSim-240, WordSim-296 and SimLex-999, respectively. The results are presented in Table 1. The results of ordinary CBOW and Skipgram are shown in the table. CBOW/Skipgram+CWE represents CWE trained as CBOW or Skipgram. For CWE, we only show the results of position-based character embeddings here because the results of cluster-based character embeddings are worse in the experiments. We found that CWE only consistently improved the performance on SimLex-999 for both CBOW and Skipgram probably because SimLex-999 contains more words that could be understood from their compositional characters. On SimLex-999, we observed that CWE was better with CBOW than Skipgram. We think the reason is that CBOW+CWE predicts the target word with the mean value of all character embeddings in the context, thus has a less noisy feature; however Skipgram+CWE uses character embeddings of an individual word. This noisy feature could cause

Model	WS-240	WS-296	SL-999
CBOW	**0.5203**	0.5550	0.3330
+CWE	0.4914	**0.5553**	0.3471
+CWE+MGE	0.3767	0.2962	0.2762
+CWE+ctxG	0.4982	0.5549	**0.3538**
+CWE+tarG	0.5038	0.5503	0.3493
Skipgram	**0.5922**	0.5876	0.3663
+CWE	0.5916	**0.5936**	0.3668
+CWE+ctxG	0.5886	0.5856	**0.3686**
RNN-Skipgram	0.3414	0.3698	0.2464
RNN-Glove	0.2963	0.1563	0.1010

Table 1: Spearman's correlation between human annotated scores and cosine similarity of word representations on three datasets: WordSim-240, WordSim-296 and SimLex-999. The higher the values, the better the results.

negative effects on predicting the target word. The GWEs were learned based on CWE in two ways. “ctxG” represents using glyph features of context words, while “tarG” represents using glyph features of target words. The glyph features improved CWE on WordSim-240 and SimLex-999, but not WordSim-296.

As for MGE results, we were not able to reproduce the performance in (Yin et al., 2016). We list possible reasons as below: we did not separate non-compositional word during training (character and radical embeddings are not used for these words), and the we created character-radical index from different data source. We conjecture that the first to be the most crucial factor in reproducing MGE.

The results of RNN-Skipgram and RNN-GloVe are also in Table 1. Their results are not comparable with CBOW and Skipgram. From the results, we conclude that it is not easy to produce word representations directly from glyphs. We think the reason is that RNN representations are dependent on each other. Updating model parameters for word w_i would also change the word representation of word w_j. As a result it is much more difficult to train such models.

We further inspect the impact of glyph features by doing significance test[8] between proposed methods and existing ones. The p-values of the tests are given in Table 2. We found only “tarG” method has a p-value less than 0.05 over CWE.

	+CWE+ctxG	+CWE+tarG
CBOW	0.085	0.215
CBOW+CWE	0.190	**0.008**

Table 2: p-values of significance tests between proposed methods and existing ones.

5.2 Word Analogy

An analogy problem has the following form: *“king”:“queen” = “man”:“?”*, and *“woman”* is answer to *“?”*. By answering the question correctly, the model is considered capable of expressing semantic relationships. Furthermore, the analogy relation could be expressed by vector arithmetic of word representations as shown in (Mikolov et al., 2013b). For the above problem, we find word w_i such that $w_i = \arg\max_w \ cos(\vec{w}, \vec{w}_{queen} - \vec{w}_{king} + \vec{w}_{man})$.

Method	Capital	City	Family	J&P
CBOW	**0.8006**	**0.7200**	0.4228	0.3100
+CWE	0.7858	0.5829	0.4743	0.2667
+MGE	0.0384	0.0114	0.1287	0.0433
+CWE+ctxG	0.7888	0.5771	0.4963	0.2917
+CWE+tarG	0.7858	0.5829	**0.5184**	0.2817
Skipgram	**0.7962**	**0.8971**	0.4779	0.2317
+CWE	0.7932	0.8686	0.5404	0.2000
+CWE+ctxG	0.7932	0.8686	**0.5662**	0.2000
RNN-Skipgram	0.0000	0.0057	0.0368	-
RNN-Glove	0.0281	0.0057	0.0184	-

Table 3: Accuracy of analogy problems for capitals of countries, (China) states/provinces of cities, family relations, and our proposed *job&place* (J&P) dataset. The higher the values, the better the results.

As in the previous subsection, we translated the word analogy dataset in (Chen et al., 2015) to traditional. The dataset contains 3 groups of analogy problems: capitals of countries, (China) states/provinces of cities, and family relations. Considering that most capital and city names do not relate to the meaning of their compositional characters, and that we did not separate non-compositional word in our experiments, we proposed a new analogy dataset composed of jobs and places (*job&place*). Nonetheless, there might be multiple corresponding places for a single job. For instance, A *“doctor”* could be in a *“hospital”* or *“clinic”*. In this *job&place* dataset, we provide a set of places for each job. The model is considered to answer correctly as long as the predicted word is in this set.

We take the mean of all word representations of places ($mean(\vec{w}_{places_1})$) for the first job (job_1), and find the place for another job (job_2) by calculating w_i such that $w_i = \arg\max_w \ cos(\vec{w}, mean(\vec{w}_{places_1}) - \vec{w}_{job_1} + \vec{w}_{job_2})$.

The results are shown in Table 3. we observed CWE only improved accuracy only for the family group. The results are not surprising. The words of family relations are compositional in Chinese, however capital and city names are usually not. We observed that GWE further improved CWE for words in the family group. From Table 3, we found that glyph features are helpful when the characters can enhance word representations. This is very reasonable because glyph features are fruitful representations of characters. If character information does not play a role in learning word representations, character glyphs may not be useful. The same phenomenon is observed in Table 1.

[8] We followed the method described in `https://stats.stackexchange.com/questions/17696/`

In our *job&place*, we still observed that GWE improving CWE, however both CWE and GWE were slightly worse than CBOW. We also observed that Skipgram-based methods became worse than CBOW-based methods, while in all previous evaluation Skipgram-based methods are consistently better.

The results of RNN-Skipgram and RNN-GloVe are still poor. We observe that the word representations learned from RNN can no longer be expressed by vector arithmetic. The reason is still under investigation.

5.3 Case Study

To further probe the effect of glyph features, we show the following word pairs in SimLex-999 whose calculated cosine similarities are higher based on GWE models than CWE. The pairs may not look alike, but their components share related semantics. For example, in “伶俐” (clever), the component “利”(sharp) is compositional to the meaning of “俐”(acute), describing someone with a sharp mind. Other examples show the ability to associate semantics with radicals.

Models	詞 (word) & 字典 (dictionary)	椅子 (chair) & 板凳 (bench)
CBOW	0.2342	0.3469
+CWE	0.2918	0.3640
+CWE+ctx_Glyph	**0.3361**	**0.3903**
+CWE+tar_Glyph	0.2857	0.3746

Models	鳥 (bird) & 火雞 (turkey)	聰明 (smart) & 伶俐 (clever)
CBOW	0.2640	0.2634
+CWE	0.3064	0.2409
+CWE+ctxG	0.3190	0.2710
+CWE+ctxG	**0.3422**	**0.2976**

Table 4: Case study on word pairs in SimLex-999.

We also provide several counter-examples. Below are some word pairs which are not similar, however GWE methods produces higher similarity than CBOW or CWE. Take “山峰” (mountain) and “蜂蜜” (honey) as example. Since they share no common characters, the only thing in common is the component “夆”, and we assume this to be the reason for the higher similarity. Also note that in the pair “無趣” (boring) and “好笑” (funny), the CWE similarity is also higher. We conclude that the character “無” (none) is not strong enough, so the character “趣” (fun) overrides the word “無趣” (boring), thus a higher score was mistakenly assigned.

Models	山峰 (mountain) & 蜂蜜 (honey)	書桌 (desk) & 水果 (fruit)
CBOW	0.0581	0.0495
+CWE	0.0842	0.0719
+CWE+ctxG	0.0736	**0.0942**
+CWE+tarG	**0.1093**	0.0733

Models	無趣 (boring) & 好笑 (funny)	胃 (stomach) & 腰 (waist)
CBOW	0.3645	0.2388
+CWE	0.5351	0.2073
+CWE+ctxG	0.5209	0.2500
+CWE+ctxG	**0.5426**	**0.2643**

Table 5: Counter examples to which GWE methods give higher similarity scores than CBOW or CWE.

6 Conclusions

This work is a pioneer in enhancing Chinese word representations with character glyphs. The character glyph features are directly learned from the bitmaps of characters by convAE. We then proposed 2 methods in learning Chinese word representations: the first is to use character glyph features as enhancement; the other is to directly learn word representation from sequences of glyph features. In experiments, we found the latter totally infeasible. Training word representations with RNN without word and character information is challenging. Nonetheless, the glyph features improved the character-enhanced Chinese word representations, especially on the word analogy task related to family.

The results of exploiting character glyph features in word representation learning was ordinary. Perhaps the co-occurrence information in the corpus plays a bigger role than glyph features. Nonetheless, the idea to treat each Chinese character as image is innovative. As more character-level models(Zheng et al., 2013; Kim, 2014; Zhang et al., 2015) are proposed in the NLP field, we believe glyph features could serve as an enhancement, and we will further examine the effect of glyph features on other tasks, such as word segmentation, POS tagging, dependency parsing, or downstream tasks such as text classification, or document retrieval.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

John A Bullinaria and Joseph P Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods*, 39(3):510–526.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huan-Bo Luan. 2015. Joint learning of character and word embeddings. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1236–1242.

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

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

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

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

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

Yanran Li, Wenjie Li, Fei Sun, and Sujian Li. 2015. Component-enhanced chinese character embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 829–834, Lisbon, Portugal. Association for Computational Linguistics.

Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. 2017. Learning character-level compositionality with visual features. *CoRR*, abs/1704.04859.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. 2011. Stacked convolutional autoencoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *In Proceedings of the International Conference on Learning Representations (ICLR)*.

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

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Yaming Sun, Lei Lin, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2014. Radical-enhanced chinese character embedding. In *International Conference on Neural Information Processing*, pages 279–286. Springer.

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

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Jian Xu, Jiawei Liu, Liangang Zhang, Zhengyu Li, and Huanhuan Chen. 2016. Improve chinese word embeddings by exploiting internal structure. In *Proceedings of NAACL-HLT*, pages 1041–1050.

Rongchao Yin, Quan Wang, Peng Li, Rui Li, and Bin Wang. 2016. Multi-granularity chinese word embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 981–986.

Matthew D Zeiler, Graham W Taylor, and Rob Fergus. 2011. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2018–2025. IEEE.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *EMNLP*, pages 647–657.

Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. 2015. Learning continuous word embedding with metadata for question retrieval in community question answering. In *ACL (1)*, pages 250–259.

Learning Paraphrastic Sentence Embeddings from Back-Translated Bitext

John Wieting[1] **Jonathan Mallinson**[2] **Kevin Gimpel**[1]
[1]Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA
[2]School of Informatics, University of Edinburgh, Edinburgh, UK
{jwieting,kgimpel}@ttic.edu, j.mallinson@ed.ac.uk

Abstract

We consider the problem of learning general-purpose, paraphrastic sentence embeddings in the setting of Wieting et al. (2016b). We use neural machine translation to generate sentential paraphrases via back-translation of bilingual sentence pairs. We evaluate the paraphrase pairs by their ability to serve as training data for learning paraphrastic sentence embeddings. We find that the data quality is stronger than prior work based on bitext and on par with manually-written English paraphrase pairs, with the advantage that our approach can scale up to generate large training sets for many languages and domains. We experiment with several language pairs and data sources, and develop a variety of data filtering techniques. In the process, we explore how neural machine translation output differs from human-written sentences, finding clear differences in length, the amount of repetition, and the use of rare words.[1]

1 Introduction

Pretrained word embeddings have received a great deal of attention from the research community, but there is much less work on developing pretrained embeddings for *sentences*. Here we target sentence embeddings that are "paraphrastic" in the sense that two sentences with similar meanings are close in the embedding space. Wieting et al. (2016b) developed paraphrastic sentence embeddings that are useful for semantic textual similarity tasks and can also be used as initialization for supervised semantic tasks.

R:	We understand that has already commenced, but there is a long way to go.
T:	This situation has already commenced, but much still needs to be done.
R:	The restaurant is closed on Sundays. No breakfast is available on Sunday mornings.
T:	The restaurant stays closed Sundays so no breakfast is served these days.
R:	Improved central bank policy is another huge factor.
T:	Another crucial factor is the improved policy of the central banks.

Table 1: Illustrative examples of references (R) paired with back-translations (T).

To learn their sentence embeddings, Wieting et al. used the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013). PPDB contains a large set of paraphrastic textual fragments extracted automatically from bilingual text ("bitext"), which is readily available for languages and domains. Versions of PPDB have been released for several languages (Ganitkevitch and Callison-Burch, 2014).

However, more recent work has shown that the fragmental nature of PPDB's pairs can be problematic, especially for recurrent networks (Wieting and Gimpel, 2017). Better performance can be achieved with a smaller set of sentence pairs derived from aligning Simple English and standard English Wikipedia (Coster and Kauchak, 2011). While effective, this type of data is inherently limited in size and scope, and not available for languages other than English.

PPDB is appealing in that it only requires bitext. We would like to retain this property but develop a data resource with sentence pairs rather than phrase pairs. We turn to neural machine translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2014; Sennrich et al., 2016a), which has matured recently to yield strong performance especially in terms of producing grammatical outputs.

In this paper, we build NMT systems for three

[1] Generated paraphrases and code are available at http://ttic.uchicago.edu/~wieting.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 274–285
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

language pairs, then use them to back-translate the non-English side of the training bitext. The resulting data consists of sentence pairs containing an English reference and the output of an X-to-English NMT system. Table 1 shows examples. We use this data for training paraphrastic sentence embeddings, yielding results that are much stronger than when using PPDB and competitive with the Simple English Wikipedia data.

Since bitext is abundant and available for many language pairs and domains,[2] we also develop several methods of filtering the data, including based on sentence length, quality measures, and measures of difference between the reference and its back-translation. We find length to be an effective filtering method, showing that very short length ranges—where the translation is 1 to 10 words—are best for learning.

In studying quality measures for filtering, we train a classifier to predict if a sentence is a reference or a back-translation, then score sentences by the classifier score. This investigation allows us to examine the kinds of phenomena that best distinguish NMT output from references in this controlled setting of translating the bitext training data. NMT output has more repetitions of both words and longer n-grams, and uses fewer rare words than the references.

We release our generated sentence pairs to the research community with the hope that the data can inspire others to develop additional filtering methods, to experiment with richer architectures for sentence embeddings, and to further analyze the differences between neural machine translations and references.

2 Related Work

We describe related work in learning general-purpose sentence embeddings, work in automatically generating or discovering paraphrases, and finally prior work in leveraging neural machine translation for embedding learning.

Paraphrastic sentence embeddings. Our learning and evaluation setting is the same as that considered by Wieting et al. (2016b) and Wieting et al. (2016a), in which the goal is to learn paraphrastic sentence embeddings that can be used for downstream tasks. They trained models on PPDB and evaluated them using a suite of semantic textual similarity (STS) tasks and supervised semantic tasks. Others have begun to consider this setting as well (Arora et al., 2017).

Other work in learning general purpose sentence embeddings has used autoencoders (Socher et al., 2011; Hill et al., 2016), encoder-decoder architectures (Kiros et al., 2015), or other learning frameworks (Le and Mikolov, 2014; Pham et al., 2015). Wieting et al. (2016b) and Hill et al. (2016) provide many empirical comparisons to this prior work. For conciseness, we compare only to the strongest configurations from their results.

Paraphrase generation and discovery. There is a rich history of research in generating or finding naturally-occurring sentential paraphrases (Barzilay and McKeown, 2001; Dolan et al., 2004; Dolan and Brockett, 2005; Quirk et al., 2004; Zhao et al., 2010; Coster and Kauchak, 2011; Xu et al., 2014, 2015).

The most relevant work uses bilingual corpora, e.g., Zhao et al. (2008) and Bannard and Callison-Burch (2005), the latter leading to PPDB. Our goals are highly similar to those of the PPDB project, which has also been produced for many languages (Ganitkevitch and Callison-Burch, 2014) since it only relies on the availability of bilingual text.

Prior work has shown that PPDB can be used for learning embeddings for words and phrases (Faruqui et al., 2015; Wieting et al., 2015). However, when learning sentence embeddings, Wieting and Gimpel (2017) showed that PPDB is not as effective as sentential paraphrases, especially for recurrent networks. These results are intuitive because the phrases in PPDB are short and often cut across constituent boundaries. For sentential paraphrases, Wieting and Gimpel (2017) used a dataset developed for text simplification by Coster and Kauchak (2011). It was created by aligning sentences from Simple English and standard English Wikipedia. We compare our data to both PPDB and this Wikipedia dataset.

Neural machine translation for paraphrastic embedding learning. Sutskever et al. (2014) trained NMT systems and visualized part of the space of the source language encoder for their English→French system. Hill et al. (2016) evaluated the encoders of English-to-X NMT systems as sentence representations, finding them to

[2] For example, CzEng 1.6 (Bojar et al., 2016) contains a billion words across its 8 domains.

perform poorly compared to several other methods based on unlabeled data. Mallinson et al. (2017) adapted trained NMT models to produce sentence similarity scores in semantic evaluations. They used pairs of NMT systems, one to translate an English sentence into multiple foreign translations and the other to then translate back to English. Other work has used neural MT architectures and training settings to obtain better word embeddings (Hill et al., 2014a,b).

Our approach differs in that we only use the NMT system to generate training data for training sentence embeddings, rather than use it as the source of the model. This permits us to decouple decisions made in designing the NMT architecture from decisions about which models we will use for learning sentence embeddings. Thus we can benefit from orthogonal work in designing neural architectures to embed sentences.

3 Neural Machine Translation

We now describe the NMT systems we use for generating data for learning sentence embeddings. In our experiments, we use three encoder-decoder NMT models: Czech→English, French→English, and German→English.

We used Groundhog[3] as the implementation of the NMT systems for all experiments. We generally followed the settings and training procedure from previous work (Bahdanau et al., 2014; Sennrich et al., 2016a). As such, all networks have a hidden layer size of 1000 and an embedding layer size of 620. During training, we used Adadelta (Zeiler, 2012), a minibatch size of 80, and the training set was reshuffled between epochs. We trained a network for approximately 7 days on a single GPU (TITAN X), then the embedding layer was fixed and training continued, as suggested by Jean et al. (2015), for 12 hours. Additionally, the softmax was calculated over a filtered list of candidate translations. Following Jean et al. (2015), during decoding, we restrict the softmax layers' output vocabulary to include: the 10000 most common words, the top 25 unigram translations, and the gold translations' unigrams.

All systems were trained on the available training data from the WMT15 shared translation task (15.7 million, 39.2 million, and 4.2 million sentence pairs for CS→EN, FR→EN, and DE→EN, respectively). The training data included: Europarl v7 (Koehn, 2005), the Common Crawl corpus, the UN corpus (Eisele and Chen, 2010), News Commentary v10, the 10^9 French-English corpus, and CzEng 1.0 (Bojar et al., 2016). A breakdown of the sizes of these corpora can be found in Table 3. The data was pre-processed using standard pre-processing scripts found in Moses (Koehn et al., 2007). Rare words were split into sub-word units, following Sennrich et al. (2016b). BLEU scores on the WMT2015 test set for each NMT system can be seen in Table 3.

	Czech	French	German
Europarl	650,000	2,000,000	2,000,000
Common Crawl	160,000	3,000,000	2,000,000
News Commentary	150,000	200,000	200,000
UN	-	12,000,000	-
10^9 French-English	-	22,000,000	-
CzEng	14,700,000	-	-

Table 2: Dataset sizes (numbers of sentence pairs) for data domains used for training NMT systems.

Language	% BLEU
Czech→English	19.7
French→English	20.1
German→English	28.2

Table 3: BLEU scores on the WMT2015 test set.

To produce paraphrases we use "back-translation", i.e., we use our X→English NMT systems to translate the non-English sentence in each training sentence pair into English. We directly use the bitext on which the models were trained. This could potentially lead to pairs in which the reference and translation match exactly, if the model has learned to memorize the reference translations seen during training. However, in practice, since we have so much bitext to draw from, we can easily find data in which they do not match exactly.

Thus our generated data consists of pairs of English references from the bitext along with the NMT-produced English back-translations. We use beam search with a width of 50 to generate multiple translations for each non-English sentence, each of which is a candidate paraphrase for the English reference.

Example outputs of this process are in Table 1, showing some rich paraphrase phenomena in the data. These examples show non-trivial phrase substitutions ("there is a long way to go" and "much still needs to be done"), sentences being merged and simplified, and sentences being rearranged.

[3] Available at https://github.com/sebastien-j/LV_groundhog.

For examples of erroneous paraphrases that can be generated by this process, see Table 11.

4 Models and Training

Our goal is to compare our paraphrase dataset to other datasets by using each to train sentence embeddings, keeping the models and learning procedure fixed. So we select models and a loss function from prior work (Wieting et al., 2016b; Wieting and Gimpel, 2017).

4.1 Models

We wish to embed a word sequence s into a fixed-length vector. We denote the tth word in s as s_t, and we denote its word embedding by x_t. We focus on two models in this paper. The first model, which we call AVG, simply averages the embeddings x_t of all words in s. The only parameters learned in this model are those in the word embeddings themselves, which are stored in the word embedding matrix W_w. This model was found by Wieting et al. (2016b) to perform very strongly for semantic similarity tasks.

The second model, the GATED RECURRENT AVERAGING NETWORK (GRAN) (Wieting and Gimpel, 2017), combines the benefits of AVG and long short-term memory (LSTM) recurrent neural networks (Hochreiter and Schmidhuber, 1997). It first uses an LSTM to generate a hidden vector, h_t, for each word s_t in s. Then h_t is used to compute a gate that is elementwise-multiplied with x_t, resulting in a new hidden vector a_t for each step t:

$$a_t = x_t \odot \sigma(W_x x_t + W_h h_t + b) \qquad (1)$$

where W_x and W_h are parameter matrices, b is a parameter vector, and σ is the elementwise logistic sigmoid function. After all a_t have been generated for a sentence, they are averaged to produce the embedding for that sentence. The GRAN reduces to AVG if the output of the gate is always 1. This model includes as learnable parameters those of the LSTM, the word embeddings, and the additional parameters in Eq. (1). We use W_c to denote the "compositional" parameters, i.e., all parameters other than the word embeddings.

Our motivation for choosing these two models is that they both work well in this transfer learning setting (Wieting et al., 2016b) and they are architecturally similar with one crucial difference: only the GRAN takes into account word order. This difference plays an important role in the effectiveness of the different filtering methods as explored in Section 5.

4.2 Training

We follow the training procedure of Wieting et al. (2015) and Wieting et al. (2016b). The training data is a set S of paraphrastic pairs $\langle s_1, s_2\rangle$ and we optimize a margin-based loss:

$$\min_{W_c, W_w} \frac{1}{|S|}\Bigg(\sum_{\langle s_1,s_2\rangle \in S} \max(0, \delta - \cos(g(s_1), g(s_2)) \\ + \cos(g(s_1), g(t_1))) + \max(0, \delta - \cos(g(s_1), g(s_2)) \\ + \cos(g(s_2), g(t_2)))\Bigg) + \lambda_c \|W_c\|^2 + \lambda_w \|W_{w_{initial}} - W_w\|^2$$

where g is the model (AVG or GRAN), δ is the margin, λ_c and λ_w are regularization parameters, $W_{w_{initial}}$ is the initial word embedding matrix, and t_1 and t_2 are "negative examples" taken from a mini-batch during optimization. The intuition is that we want the two texts to be more similar to each other $(\cos(g(s_1), g(s_2)))$ than either is to their respective negative examples t_1 and t_2, by a margin of at least δ. To select t_1 and t_2, we choose the most similar sentence in some set (other than those in the given pair). For simplicity we use the mini-batch for this set, i.e., we choose t_1 for a given $\langle s_1, s_2\rangle$ as follows:

$$t_1 = \operatorname*{argmax}_{t:\langle t,\cdot\rangle \in S_b \setminus \{\langle s_1,s_2\rangle\}} \cos(g(s_1), g(t))$$

where $S_b \subseteq S$ is the current mini-batch. That is, we want to choose a negative example t_i that is similar to s_i according to the current model. The downside is that we may occasionally choose a phrase t_i that is actually a true paraphrase of s_i.

5 Experiments

We now investigate how best to use our generated paraphrase data for training universal paraphrastic sentence embeddings. We consider 10 data sources: Common Crawl (CC), Europarl (EP), and News Commentary (News) from all 3 language pairs, as well as the 10^9 French-English data (Giga). We extract 150,000 reference/back-translation pairs from each data source. We use 100,000 of these to mine for training data for our sentence embedding models, and the remaining 50,000 are used as train/validation/test data for the reference classification and language models described below.

5.1 Evaluation

We evaluate the quality of a paraphrase dataset by using the experimental setting of Wieting et al. (2016b). We use the paraphrases as training data to create paraphrastic sentence embeddings, using the cosine of the embeddings as the measure of semantic relatedness, then evaluate the embeddings on the SemEval semantic textual similarity (STS) tasks from 2012 to 2015 (Agirre et al., 2012, 2013, 2014, 2015), the SemEval 2015 Twitter task (Xu et al., 2015), and the SemEval 2014 SICK Semantic Relatedness task (Marelli et al., 2014).

Given two sentences, the aim of the STS tasks is to predict their similarity on a 0-5 scale, where 0 indicates the sentences are on different topics and 5 indicates that they are completely equivalent. As our test set, we report the average Pearson's r over these 22 sentence similarity tasks.[4] As development data, we use the 2016 STS tasks (Agirre et al., 2016), where the tuning criterion is the average Pearson's r over its 5 datasets.

5.2 Experimental Setup

For fair comparison among different datasets and dataset filtering methods described below, we use only 24,000 training examples for nearly all experiments. Different filtering methods produce different amounts of training data, and using 24,000 examples allows us to keep the amount of training data constant across filtering methods. It also allows us to complete these several thousand experiments in a reasonable amount of time. In Section 5.8 below, we discuss experiments that scale up to larger amounts of training data.

We use PARAGRAM-SL999 embeddings (Wieting et al., 2015) to initialize the word embedding matrix (W_w) for both models. For all experiments, we fix the mini-batch size to 100, λ_w to 0, λ_c to 0, and the margin δ to 0.4. We train AVG for 20 epochs, and the GRAN for 3, since it converges much faster. For optimization we use Adam (Kingma and Ba, 2014) with a learning rate of 0.001.

Lang.	Data	GRAN	AVG
SimpWiki		67.2	65.8
PPDB		64.5	65.8
CS	CC	65.5	65.4
	EP	66.5	65.1
	News	67.2	65.1
FR	CC	67.3	66.1
	EP	**67.8**	65.7
	Giga	67.4	65.9
	News	67.0	65.2
DE	CC	66.5	**66.2**
	EP	67.2	65.6
	News	66.5	64.7

Table 4: Test results (average Pearson's $r \times 100$ over 22 STS datasets) using a random selection of 24,000 examples from each data source.

We compare to two data resources used in previous work to learn paraphrastic sentence embeddings. The first is phrase pairs from PPDB, used by Wieting et al. (2016b) and Wieting et al. (2016a). PPDB comes in different sizes (S, M, L, XL, XXL, and XXXL), where each larger size subsumes all smaller ones. The pairs in PPDB are sorted by a confidence measure and so the smaller sets contain higher precision paraphrases. We use PPDB XL in this paper, which consists of fairly high precision paraphrases. The other data source is the aligned Simple English / standard English Wikipedia data developed by Coster and Kauchak (2011) and used for learning paraphrastic sentence embeddings by Wieting and Gimpel (2017). We refer to this data source as "SimpWiki". We refer to our back-translated data as "NMT".

5.3 Dataset Comparison

We first compare datasets, randomly sampling 24,000 sentence pairs from each of PPDB, SimpWiki, and each of our NMT datasets. The only hyperparameter to tune for this experiment is the stopping epoch, which we tune based on our development set. The results are shown in Table 4.

We find that the NMT datasets are all effective as training data, outperforming PPDB in all cases when using the GRAN. There are exceptions when using AVG, for which PPDB is quite strong. This is sensible because AVG is not sensitive to word order, so the fragments in PPDB do not cause problems. However, when using the GRAN, which is sensitive to word order, the NMT data is consistently better than PPDB. It often exceeds the performance of training on the SimpWiki data, which consists entirely of human-written sentences.

5.4 Filtering Methods

Above we showed that the NMT data is better than PPDB when using a GRAN and often as good as SimpWiki. Since we have access to so much more NMT data than SimpWiki (which is limited

[4] Statistical significance testing is nontrivial due to averaging Pearson's r so we leave it to future work.

to fewer than 200k sentence pairs), we next experiment with several approaches for filtering the NMT data. We first consider filtering based on length, described in Section 5.5. We then consider filtering based on several quality measures designed to find more natural and higher-quality translations, described in Section 5.6. Finally, we consider several measures of diversity. By diversity we mean here a measure of the lexical and syntactic difference between the reference and its paraphrase. We describe these experiments in Section 5.7. We note that these filtering methods are not all mutually exclusive and could be combined, though in this paper we experiment with each individually and leave combination to future work.

5.5 Length Filtering

We first consider filtering candidate sentence pairs by length, i.e., the number of tokens in the translation. The tunable parameters are the upper and lower bounds of the translation lengths.

We experiment with a partition of length ranges, showing the results in Table 5. These results are averages across all language pairs and data sources of training data for each length range shown. We find it best to select NMT data where the translations have between 0 and 10 tokens, with performance dropping as sentence length increases. This is true for both the GRAN and Avg models. We do the same filtering for the SimpWiki data, though the trend is not nearly as strong. Therefore this is unlikely due to the nature of the evaluation data, and may be due to machine translation quality dropping as sentence length increases. This trend appears even though the datasets with higher ranges have more tokens of training data, since only the number of training sentence pairs is kept constant across configurations.

We then tune the length range using our development data, considering the following length ranges: [0,10], [0,15], [0,20], [0,30], [0,100], [10,20], [10,30], [10,100], [15,25], [15,30], [15,100], [20,30], [20,100], [30,100]. We tune over ranges as well as language, data source, and stopping epoch, each time training on 24,000 sentence pairs. We report the average test results over all languages and datasets in Table 6. We compare to a baseline that draws a random set of data, showing that length-based filtering leads to gains of nearly half a point on average across our test sets.

		Length Range			
Data	Model	0-10	10-20	20-30	30-100
SimpWiki	GRAN	67.4	67.7	67.1	67.3
	Avg	65.9	65.7	65.6	65.9
NMT	GRAN	66.6	66.5	66.0	64.8
	Avg	65.7	65.6	65.3	65.0

Table 5: Test correlations for our models when trained on sentences with particular length ranges (averaged over languages and data sources for the NMT rows). Results are on STS datasets (Pearson's $r \times 100$).

	NMT		SimpWiki	
Filtering Method	GRAN	Avg	GRAN	Avg
None (Random)	66.9	65.5	67.2	65.8
Length	67.3	66.0	67.4	66.2
Tuned Len. Range	[0,10]	[0,10]	[0,10]	[0,15]

Table 6: Length filtering test results after tuning length ranges on development data (averaged over languages and data sources for the NMT rows). Results are on STS datasets (Pearson's $r \times 100$).

The tuned length ranges are short for both NMT and SimpWiki. The distribution of lengths in the NMT and SimpWiki data is fairly similar. The 10 NMT datasets all have mean translation lengths between 22 and 28 tokens. The data has fairly large standard deviations (11-25 tokens) indicating that there are some very long translations in the data. SimpWiki has a mean length of 24.2 and a standard deviation of 13.1.

5.6 Quality Filtering

We also consider filtering based on several measures of the "quality" of the back-translation:

- **Translation Cost**: We use the cost (negative log likelihood) of the translation from the NMT system, divided by the number of tokens in the translation.
- **Language Model**: We train a separate language model for each language/data pair on 40,000 references that are separate from the 100,000 used for mining data. Due to the small data size, we train a 3-gram language model and use the KenLM toolkit (Heafield, 2011).
- **Reference/Translation Classification**: We train binary classifiers to predict whether a given sentence is a reference or translation (described in Section 5.6.1). We use the probability of being a reference as the score for filtering.

For translation cost, we tune the upper bound of the cost over the range $[0.2, 1]$ using increments

Filtering Method	GRAN	AVG
None (Random)	66.9	65.5
Translation Cost	66.6	65.4
Language Model	66.7	65.5
Reference Classification	67.0	65.5

Table 7: Quality filtering test results after tuning quality hyperparameters on development data (averaged over languages and data sources for the NMT rows). Results are on STS datasets (Pearson's $r \times 100$).

of 0.1. For the language model, we tune an upper bound on the perplexity of the translations among the set $\{25, 50, 75, 100, 150, 200, \infty\}$. For the classifier, we tune the minimum probability of being a reference over the range $[0, 0.9]$ using increments of 0.1.

Table 7 shows average test results over all languages and datasets after tuning hyperparameters on our development data for each. The translation cost and language model are not helpful for filtering, as random selection outperforms them. Both methods are outperformed by the reference classifier, which slightly outperforms random selection when using the stronger GRAN model. We now discuss further how we trained the reference classifier and the data characteristics that it reveals. We did not experiment with quality filtering for SimpWiki since it is human-written text.

5.6.1 Reference/Translation Classification

We experiment with predicting whether a given sentence is a reference or a back-translation, hypothesizing that generated sentences with high probabilities of being references are of higher quality. We train two kinds of binary classifiers, one using an LSTM and the other using word averaging, followed by a softmax layer. We select 40,000 reference/translation pairs for training and 5,000 for each of validation and testing. A single example is a sentence with label 1 if it is a reference translation and 0 if it is a translation.

In training, we consider the entire k-best list as examples of translations, selecting one translation to be the 0-labeled example. We either do this randomly or we score each sentence in the k-best list using our model and select the one with the highest probability of being a reference as the 0-labeled example. We tune this choice as well as an L_2 regularizer on the word embeddings (tuned over $\{10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 0\}$). We use PARAGRAM-SL999 embeddings (Wieting et al., 2015) to initialize the word embeddings for both models. Models were trained by minimizing cross entropy for 10 epochs using Adam with learning rate 0.001. We performed this procedure separately for each of the 10 language/data pairs.

Model	Lang.	Data	Test Acc.	+ Acc.	- Acc.
LSTM	CS	CC	72.2	72.2	72.3
		EP	72.3	64.3	80.3
		News	79.7	73.2	86.3
	FR	CC	80.7	82.1	79.3
		EP	79.3	75.2	83.4
		Giga	**93.1**	**92.3**	93.8
		News	84.2	81.2	87.3
	DE	CC	79.3	71.7	86.9
		EP	85.1	78.0	92.2
		News	89.8	82.3	**97.4**
AVG	CS	CC	71.2	68.9	73.5
		EP	69.1	63.0	75.1
		News	77.6	71.7	83.6
	FR	CC	78.8	80.4	77.2
		EP	78.9	75.5	82.3
		Giga	**92.5**	**91.5**	93.4
		News	82.8	81.1	84.5
	DE	CC	77.3	70.4	84.1
		EP	82.7	73.4	91.9
		News	87.6	80.0	**95.3**

Table 8: Results of reference/translation classification (accuracy×100). The highest score in each column is in boldface. Final two columns show accuracies of positive (reference) and negative classes, respectively.

The results are shown in Table 8. While performance varies greatly across data sources, the LSTM always outperforms the word averaging model. For our translation-reference classification, we note that our results can be further improved. We also trained models on 90,000 examples, essentially doubling the amount of data, and the results improved by about 2% absolute on each dataset on both the validation and testing data.

Analyzing Reference Classification. We inspected the output of our reference classifier and noted a few qualitative trends which we then verified empirically. First, neural MT systems tend to use a smaller vocabulary and exhibit more restricted use of phrases. They correspondingly tend to show more repetition in terms of both words and longer n-grams. This hypothesis can be verified empirically in several ways. We do so by calculating the entropy of the unigrams and trigrams for both the references and the translations from our 150,000 reference-translation pairs.[5] We also calculate the repetition percentage of unigrams and

[5] We randomly selected translations from the beam search.

Lang.	Data	Ent. (uni)	Ent. (tri)	Rep. (uni)	Rep. (tri)
CS	CC	0.50	1.13	-7.57%	-5.58%
	EP	0.14	0.31	-0.88%	-0.11%
	News	0.16	0.31	-0.96%	-0.16%
FR	CC	0.97	1.40	-8.50%	-7.53%
	EP	0.51	0.69	-1.85%	-0.58%
	Giga	0.97	1.21	-5.30%	-7.74%
	News	0.67	0.75	-2.98%	-0.85%
DE	CC	0.29	0.57	-1.09%	-0.73%
	EP	0.32	0.53	-0.14%	-0.11%
	News	0.40	0.37	-1.02%	-0.24%
All		0.46	0.74	-2.80%	-2.26%

Table 9: Differences in entropy and repetition of unigrams/trigrams in references and translations. Negative values indicate translations have a higher value, so references show consistently higher entropies and lower repetition rates.

trigrams in both the references and translations. This is defined as the percentage of words that are repetitions (i.e., have already appeared in the sentence). For unigrams, we only consider words consisting of at least 3 characters.

The results are shown in Table 9, in which we subtract the translation value from the reference value for each measure. The translated text has lower n-gram entropies and higher rates of repetition. This appears for all datasets, but is strongest for common crawl and French-English 10[9].

We also noticed that translations are less likely to use rare words, instead willing to use a larger sequence of short words to convey the same meaning. We found that translations were sometimes more vague and, unsurprisingly, were more likely to be ungrammatical.

We check whether our classifier is learning these patterns by computing the reference probabilities $P(\mathrm{R})$ of 100,000 randomly sampled translation-reference pairs from each dataset (the same used to train models). We then compute the correlation between our classification score and different metrics: the repetition rate of the sentence, the average inverse-document frequency (IDF) of the sentence,[6] and the translation length.

The results are shown in Table 10. Negative correlations with repetitions indicates that fewer repetitions lead to higher $P(\mathrm{R})$. A positive correlation with average IDF indicates that $P(\mathrm{R})$ rewards the use of rare words. Interestingly, negative correlation with length suggests that the classifier prefers more concise sentences.[7] We show examples of these phenomena in Table 11. The first two examples show the tendency of NMT to repeat words and phrases. The second two show how they tend to use sequences of common words ("put at risk") rather than rare words ("endangering").

Metric	Spearman's ρ
Unigram repetition rate	-35.1
Trigram repetition rate	-18.4
Average IDF	27.8
Length	-34.0

Table 10: Spearman's ρ between our reference classifier probability and various measures.

	Sentence	$P(\mathrm{R})$
R:	Room was comfortable and the staff at the front desk were very helpful.	1.0
T:	The staff were very nice and the room was very nice and the staff were very nice.	<0.01
R:	The enchantment of your wedding day, captured in images by Flore-Ael Surun.	0.98
T:	The wedding of the wedding, put into images by Flore-Ael A.	<0.01
R:	Mexico and Sweden are longstanding supporters of the CTBT.	1.0
T:	Mexico and Sweden have been supporters of CTBT for a long time now.	0.06
R:	We thought Mr Haider ' s Austria was endangering our freedom.	1.0
T:	We thought that our freedom was put at risk by Austria by Mr Haider.	0.09

Table 11: Illustrative examples of references (R) and back-translations (T), along with probabilities from the reference classifier. See text for details.

5.7 Diversity Filtering

We consider several filtering criteria based on measures that encourage particular amounts of disparity between the reference and its back-translation:

- **n-gram Overlap**: Our n-gram overlap measures are calculated by counting n-grams of a given order in both the reference and translation, then dividing the number of shared n-grams by the total number of n-grams in the reference or translation, whichever has fewer. We use three n-gram overlap scores ($n \in \{1, 2, 3\}$).
- **BLEU Score**: We use a smoothed sentence-level BLEU variant from Nakov et al. (2012) that uses smoothing for all n-gram lengths and also smooths the brevity penalty.

[6] Wikipedia was used to calculate the frequencies of the tokens. All tokens were lowercased.

[7] This is noteworthy because the average sentence length of translations and references is not significantly different.

	NMT		SimpWiki	
Filtering Method	GRAN	AVG	GRAN	AVG
Random	66.9	65.5	67.2	65.8
Unigram Overlap	66.6	66.1	67.8	67.4
Bigram Overlap	67.0	65.5	68.0	67.2
Trigram Overlap	66.9	65.4	67.8	66.6
BLEU Score	67.1	65.3	67.5	66.5

Table 12: Diversity filtering test results after tuning filtering hyperparameters on development data (averaged over languages and data sources for the NMT rows). Results are on STS datasets (Pearson's $r \times 100$).

Data	GRAN	AVG
PPDB	64.6	66.3
SimpWiki (100k/168k)	67.4	**67.7**
CC-CS (24k)	66.8	-
CC-CS (100k)	**68.5**	-
CC-DE (24k)	-	66.6
CC-DE (168k)	-	67.6

Table 13: Test results with more training data. More data helps both AVG and GRAN to match or surpass training on SimpWiki. Both comfortably surpass PPDB. The number of training examples used is in parentheses.

For both methods, the tunable hyperparameters are the upper and lower bounds for the above scores. We tune over the cross product of lower bounds $\{0, 0.1, 0.2, 0.3\}$ and upper bounds $\{0.6, 0.7, 0.8, 0.9, 1.0\}$. Our intuition is that the best data will have some amount of n-gram overlap, but not too much. Too much n-gram overlap will lead to pairs that are not useful for learning.

The results are shown in Table 12, for both models and for both NMT and SimpWiki. We find that the diversity filtering methods lead to consistent improvements when training on SimpWiki. We believe this is because many of the sentence pairs in SimpWiki are near-duplicates and these filtering methods favor data with more differences.

Diversity filtering can also help when selecting NMT data, though the differences are smaller. We do note that unigram overlap is the strongest filtering strategy for AVG. When looking at the threshold tuning, the best lower bounds are often 0 or 0.1 and the best upper bounds are typically 0.6-0.7, indicating that sentence pairs with a high degree of word overlap are not useful for training. We also find that the GRAN benefits more from filtering based on higher-order n-gram overlap than AVG.

5.8 Scaling Up

Unlike the SimpWiki data, which is naturally limited and only available for English, we can scale our approach. Since we use data on which the NMT systems were trained and perform back-translation, we can easily produce large training sets of paraphrastic sentence pairs for many languages and data domains, limited only by the availability of bitext.

To test this, we took the tuned filtering methods and language/data pairs (according to our development dataset only), and trained them on more data. These were CC-CS for GRAN and CC-DE for AVG. We also trained each model on the same number of sentence pairs from SimpWiki.[8] We also compare to PPDB XL, and since PPDB has fewer tokens per example, we use enough PPDB data so that it has at least as many tokens as the SimpWiki data used in the experiment.[9]

Table 13 shows clear improvements when using more training data, providing evidence that our approach can scale to larger datasets. The NMT data surpasses SimpWiki for the GRAN, while the SimpWiki and NMT data perform similarly for AVG. PPDB is outperformed by both data sources for both models. Even when we train on all 52M tokens in PPDB XXL, AVG only reaches 66.5.

6 Conclusion

We showed how back-translation can be used to generate effective training data for paraphrastic sentence embeddings. We explored filtering strategies that improve the generated data; in doing so, we identified characteristics that distinguish NMT output from references. Our hope is that these results can enable learning paraphrastic sentence embeddings with powerful neural architectures across many languages and domains.

Acknowledgments

This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. We thank the developers of Theano (Theano Development Team, 2016) and NVIDIA Corporation for donating GPUs used in this research.

[8] Since the CC-CS data was the smallest dataset used to train the CS NMT system (See Table 3), we only used 100,000 pairs for the GRAN experiment. For AVG, we used the full 167,689.

[9] 800,011 pairs for GRAN and 1,341,188 for AVG.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. *Proceedings of SemEval*, pages 497–511.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of the International Conference on Learning Representations*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

Regina Barzilay and Kathleen R McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th annual meeting on Association for Computational Linguistics*, pages 50–57.

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovický, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. 2016. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In *Proceedings of 19th International Conference on Text, Speech, and Dialogue (TSD)*, pages 231–238.

William Coster and David Kauchak. 2011. Simple english wikipedia: a new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 665–669.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP*.

Andreas Eisele and Yu Chen. 2010. MultiUN: A multilingual corpus from united nation documents. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Juri Ganitkevitch and Chris Callison-Burch. 2014. The multilingual paraphrase database. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

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

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197.

Felix Hill, Kyunghyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014a. Embedding word similarity with neural machine translation. *arXiv preprint arXiv:1412.6448*.

Felix Hill, KyungHyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014b. Not all neural embeddings are born equal. *arXiv preprint arXiv:1410.0718*.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Yukun Zhu, Ruslan R. Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the 10th Machine Translation Summit*, pages 79–86.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 881–893.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Preslav Nakov, Francisco Guzman, and Stephan Vogel. 2012. Optimizing for sentence-level BLEU+1 yields short translations. In *Proceedings of COLING 2012*, pages 1979–1994.

Nghia The Pham, Germán Kruszewski, Angeliki Lazaridou, and Marco Baroni. 2015. Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016a. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016b. Towards universal paraphrastic sentence embeddings. In *Proceedings of International Conference on Learning Representations*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL (TACL)*.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Wei Xu, Chris Callison-Burch, and William B Dolan. 2015. SemEval-2015 task 1: Paraphrase and semantic similarity in Twitter (PIT). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval)*.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from Twitter. *Transactions of the Association for Computational Linguistics*, 2:435–448.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Shiqi Zhao, Haifeng Wang, Xiang Lan, and Ting Liu. 2010. Leveraging multiple MT engines for paraphrase generation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1326–1334.

Shiqi Zhao, Haifeng Wang, Ting Liu, and Sheng Li. 2008. Pivot approach for extracting paraphrase patterns from bilingual corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 780–788.

Joint Embeddings of Chinese Words, Characters, and Fine-grained Subcharacter Components

Jinxing Yu **Xun Jian** **Hao Xin** **Yangqiu Song**
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
{jyuat, xjian, hxinaa, yqsong}@cse.ust.hk

Abstract

Word embeddings have attracted much attention recently. Different from alphabetic writing systems, Chinese characters are often composed of subcharacter components which are also semantically informative. In this work, we propose an approach to jointly embed Chinese words as well as their characters and fine-grained subcharacter components. We use three likelihoods to evaluate whether the context words, characters, and components can predict the current target word, and collected 13,253 subcharacter components to demonstrate the existing approaches of decomposing Chinese characters are not enough. Evaluation on both word similarity and word analogy tasks demonstrates the superior performance of our model.

1 Introduction

Distributed word representation represents a word as a vector in a continuous vector space and can better uncover both the semantic and syntactic information over traditional one-hot representations. It has been successfully applied to many downstream natural language processing (NLP) tasks as input features, such as named entity recognition (Collobert et al., 2011), text classification (Joulin et al., 2016), sentiment analysis (Tang et al., 2014), and question answering (Zhou et al., 2015). Among many embedding methods (Bengio et al., 2003; Mnih and Hinton, 2009), CBOW and Skip-Gram models are very popular due to their simplicity and efficiency, making it feasible to learn good embeddings of words from large scale training corpora (Mikolov et al., 2013b,a).

Despite the success and popularity of word embeddings, most of the existing methods treat each word as the minimum unit, which ignores the morphological information of words. Rare words cannot be well represented when optimizing a cost function related to a rare word and its contexts. To address this issue, some recent studies (Luong et al., 2013; Qiu et al., 2014; Sun et al., 2016a; Wieting et al., 2016) have investigated how to exploit morphemes or character n-grams to learn better embeddings of English words.

Different from other alphabetic writing systems such as English, written Chinese is logosyllabic, i.e., a Chinese character can be a word on its own or part of a polysyllabic word[1]. The characters themselves are often composed of subcharacter components which are also semantically informative. The subword items of Chinese words, including characters and subcharacter components, contain rich semantic information. The characters composing a word can indicate the semantic meaning of the word and the subcharacter components, such as radicals and components themselves being a character, composing a character can indicate the semantic meaning of the character. The components of characters can be roughly divided into two types: semantic component and phonetic component. The semantic component indicates the meaning of a character while the phonetic component indicates the sound of a character. For example, 氵 (water) is the semantic component of characters 湖 (lake) and 海 (sea), 马 (horse) is the phonetic component of characters 妈 (mother) and 骂 (scold) where both 妈 and 骂 are pronounced similar to 马.

Leveraging the subword information such as characters and subcharacter components can enhance Chinese word embeddings with internal morphological semantics. Some methods have been proposed to incorporate the subword infor-

[1] https://en.wikipedia.org/wiki/Written_Chinese

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 286–291
Copenhagen, Denmark, September 7–11, 2017.

mation for Chinese word embeddings. Sun et al. (2014) and Li et al. (2015) proposed methods to enhance Chinese character embeddings with radicals based on C&W model (Collobert and Weston, 2008) and word2vec models (Mikolov et al., 2013a,b) respectively. Chen et al. (2015) used Chinese characters to improve Chinese word embeddings and proposed the CWE model to jointly learn *Chinese word and character* embeddings. Xu et al. (2016) extended the CWE model by exploiting the internal semantic similarity between a word and its characters in a cross-lingual manner. To combine both the *radical-character* and *character-word* compositions, Yin et al. (2016) proposed a multi-granularity embedding (MGE) model based on the CWE model, which represents the context as a combination of surrounding words, surrounding characters, and the radicals of the target word. Particularly, they developed a dictionary of 20,847 characters and 296 radicals.

However, all the above approaches still missed a lot of fine-grained components in Chinese characters. Formally and historically, radicals are character components used to index Chinese characters in dictionaries. Although many of the radicals are also semantic components, a character has only one radical, which cannot fully uncover the semantics and structure of the character. Besides over 200 radicals, there are more than 10,000 components which are also semantically meaningful or phonetically useful. For example, Chinese character 照 (illuminate, reflect, mirror, picture) has one radical 灬 (the corresponding traditional Chinese radical is 火, meaning fire) and three other components, i.e., 日 (sun), 刀 (knife), and 口 (mouth). Shi et al. (2015) proposed using WUBI input method to decompose the Chinese characters into components. However, WUBI input method uses rules to group Chinese characters into meaningless clusters which can fit the alphabet based keyboard. The semantics of the components are not straightforwardly meaningful.

In this work, we present a model to jointly learn the embeddings of Chinese words, characters, and subcharacter components. The learned Chinese word embeddings can leverage the external context co-occurrence information and incorporate rich internal subword semantic information. Experiments on both word similarity and word analogy tasks demonstrate the effectiveness of our model over previous works. The code and data are available at `https://github.com/HKUST-KnowComp/JWE`.

2 Joint Learning Word Embedding

In this section, we introduce our joint learning word embedding model (JWE), which combines words, characters, and subcharacter components information. Our model is based on CBOW model (Mikolov et al., 2013a). JWE uses the average of context word vectors, the average of context character vectors, and the average of context subcharacter vectors to predict the target word, and uses the sum of these three prediction losses as the objective function.

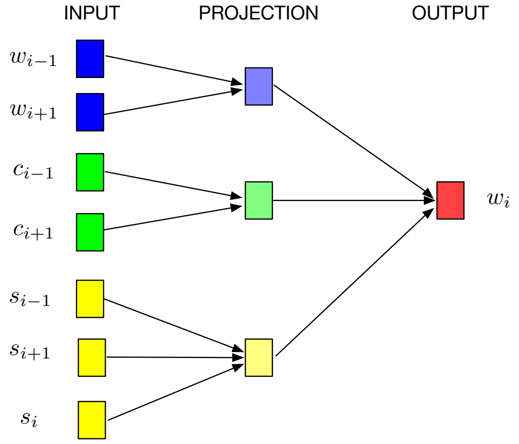


Figure 1: Illustration of JWE. w_i is the target word. w_{i-1} and w_{i+1} are the left word and right word of w_i respectively. c_{i-1} and c_{i+1} represent the characters in the context. s_{i-1} and s_{i+1} represent the subcharacters in the context, s_i represents the subcharacters of the target word w_i.

We denote D as the training corpus, $W = (w_1, w_2, \cdots, w_N)$ as the vocabulary of words, $C = (c_1, c_2, \cdots, c_M)$ as the vocabulary of characters, $S = (s_1, s_2, \cdots, s_K)$ as the vocabulary of subcharacters, and T as the context window size respectively. As illustrated in Figure 1, JWE aims to maximize the sum of log-likelihoods of three predictive conditional probabilities for a target word w_i:

$$L(w_i) = \sum_{k=1}^{3} \log P(w_i | h_{i_k}), \qquad (1)$$

where $h_{i_1}, h_{i_2}, h_{i_3}$ are the composition of context words, context characters, context subcharacters respectively. Let $\boldsymbol{v}_{w_i}, \boldsymbol{v}_{c_i}, \boldsymbol{v}_{s_i}$ be the "input" vectors of word w_i, character c_i, and subcharacter s_i

respectively, $\hat{\boldsymbol{v}}_{w_i}$ be the "output" vectors of word w_i. The conditional probability is defined by the softmax function as follows:

$$p(w_i|h_{i_k}) = \frac{\exp(\boldsymbol{h}_{i_k}^T \hat{\boldsymbol{v}}_{w_i})}{\sum_{j=1}^{N} \exp(\boldsymbol{h}_{i_k}^T \hat{\boldsymbol{v}}_{w_j})}, \quad \text{k} = 1, 2, 3, \tag{2}$$

where $\boldsymbol{h}_{i_1}$ is the average of the "input" vectors of words in the context, i.e.:

$$\boldsymbol{h}_{i_1} = \frac{1}{2T} \sum_{-T \le j \le T, j \ne 0} \boldsymbol{v}_{w_{i+j}}. \tag{3}$$

Similarly, $\boldsymbol{h}_{i_2}$ is the average of characters' "input" vectors in the context, $\boldsymbol{h}_{i_3}$ is the average of subcharacters' "input" vectors in the context or in the target word or all of them. Given a corpus D, JWE maximizes the overall log likelihood:

$$L(D) = \sum_{w_i \in D} L(w_i), \tag{4}$$

where the optimization follows the implementation of negative sampling used in CBOW model (Mikolov et al., 2013a).

This objective function is different from that of MGE (Yin et al., 2016). For a target word w_i, the objective function of MGE is almost equivalent to maximizing $P(w_i|\boldsymbol{h}_{i_1} + \boldsymbol{h}_{i_2} + \boldsymbol{h}_{i_3})$. During the backpropagation, the gradients of $\boldsymbol{h}_{i_1}, \boldsymbol{h}_{i_2}, \boldsymbol{h}_{i_3}$ can be different in our model while they are always same in MGE, so the gradients of the embeddings of words, characters, subcharacter components can be different in our model while they are same in MGE. Thus, the representations of words, characters, and subcharacter components are decoupled and can be better trained in our model. A similar decoupled objective function is used in (Sun et al., 2016a) to learn English word embeddings and phrase embeddings. Our model differs from theirs in that we combine the subwords of both the context words and target word to predict the target word while they use the morphemes of the target English word to predict it.

3 Experiments

We quantitatively evaluate the quality of word embeddings learned by our model on word similarity evaluation and word analogy tasks.

3.1 Experimental Settings

Model	Wordsim-240	Wordsim-295
CBOW	0.5009	0.5985
CWE	0.5133	0.5805
MGE	0.5128	0.5425
JWE+c+p1	0.5437	0.6549
JWE+c+p2	0.5476	0.6676
JWE+c+p3	0.5554	0.6533
JWE+r+p1	0.5478	0.6434
JWE+r+p2	**0.5619**	0.6621
JWE+r+p3	0.5273	0.6461
JWE-n	0.5476	**0.6710**

Table 1: Results on word similarity evaluation. For our JWE model, +c represents the components feature and +r represents the radicals feature; +p indicates which subcharacters are used to predict the target word; +p1 indicates using the surrounding words' subcharacter features; +p2 indicates using the target word's subcharacter features; +p3 indicates using the subcharacter features of both the surrounding words and the target word; -n indicates only using characters without either components or radicals.

Training Corpus. We adopt the Chinese Wikipedia Dump[2] as our training corpus. In preprocessing, pure digits and non Chinese characters are removed. We use THULAC[3] (Sun et al., 2016b) for Chinese word segmentation and POS tagging. We identify all entity names for CWE (Chen et al., 2015) and MGE (Yin et al., 2016) as they do not use the characters information for non-compositional words. Our model (JWE) does not use such a non-compositional word list. We obtained a 1GB training corpus with 153,071,899 tokens and 3,158,225 unique words.

Subcharacter Components. We crawled the components and radicals information of Chinese characters from HTTPCN[4]. We obtained 20,879 characters, 13,253 components and 218 radicals, of which 7,744 characters have more than one components, and 214 characters are equal to their radicals.

Parameter Settings. We compare our method with CBOW (Mikolov et al., 2013b)[5] , CWE (Chen et al., 2015)[6], and MGE (Yin et al., 2016)[7].

[2] http://download.wikipedia.com/zhwiki

[3] http://thulac.thunlp.org/

[4] http://tool.httpcn.com/zi/

[5] https://code.google.com/p/word2vec/

[6] https://github.com/Leonard-Xu/CWE

[7] We used the source code provided by the author. Our experimental results of baselines are different from that in MGE paper because we used a 1GB corpus while they used a 500MB corpus and we fixed the training iteration while they tried the training iteration in range [5, 200] and chose the best.

For all models, we used the same parameter settings. We fixed the word vector dimension to be 200, the window size to be 5, the training iteration to be 100, the initial learning rate to be 0.025, and the subsampling parameter to be 10^{-4}. Words with frequency less than 5 were ignored during training. We used 10-word negative sampling for optimization.

3.2 Word Similarity

This task evaluates the embedding's ability of uncovering the semantic relatedness of word pairs. We select two different Chinese word similarity datasets, wordsim-240 and wordsim-296 provided by (Chen et al., 2015) for evaluation. There are 240 pairs of Chinese words in wordsim-240 and 296 pairs of Chinese words in wordsim-296. Both datasets contain human-labeled similarity scores for each word pair. There is a word in wordsim-296 that did not appear in the training corpus, so we removed this from the gold-standard to produce wordsim-295. All words in wordsim-240 appeared in the training corpus. The similarity score for a word pair is computed as the cosine similarity of their embeddings generated by the learning model. We compute the Spearman correlation (Myers et al., 2010) between the human-labeled scores and similarity scores computed by embeddings. The evaluation results of our model and baseline methods on wordsim-240 and wordsim-295 are shown in Table 1.

From the results, we can see that JWE substantially outperforms CBOW, CWE, and MGE on the two word similarity datasets. JWE can better leverage the rich morphological information in Chinese words than CWE and MGE. It shows the benefits of decoupling the representation of words, characters, and subcharacter components as opposed to employing concatenation, sum, or average on all of them as the context.

We also observe that JWE with only characters can get competitive results on the word similarity task compared to JWE with characters and subcharacters. The reason may be that characters are enough to provide additional semantic information for computing the similarities of many word pairs in the two datasets. For example, the similarity of 法律 (law, statute) and 律师 (lawyer) in wordsim-295 can be directly inferred from the shared character 律 (law, rule).

3.3 Word Analogy

This task examines the quality of word embedding by its capacity of discovering linguistic regularities between pairs of words. For example, for a tuple like "罗马 (Rome): 意大利 (Italy):: 柏林 (Berlin): 德国 (Germany)", the model can answer correctly if the nearest vector representation to vec(意大利) - vec(罗马) + vec(柏林) is vec(德国) among all words except from 罗马, 意大利, and 柏林. More generally, given an analogy tuple "$a : b :: c : d$," the model answers the analogy question "$a : b :: c$:?" by finding x in the vocabulary such that

$$\arg \max_{x \neq a, x \neq b, x \neq c} cos(\vec{b} - \vec{a} + \vec{c}, \vec{x}).$$

We use accuracy as the evaluation metric. In this task, we use the Chinese word analogy dataset introduced by (Chen et al., 2015), which consists of 1,124 tuples of words and each tuple contains 4 words, coming from three different categories: "Capital" (677 tuples), "State" (175 tuples), and "Family" (272 tuples). Our training corpus covers all the testing words.

Model	Total	Capital	State	Family
CBOW	0.7954	0.8493	0.8857	0.6029
CWE	0.7553	0.8420	0.8743	0.4632
MGE	0.7696	0.8907	0.8857	0.3934
JWE+c+p1	0.7562	0.8272	0.8286	0.5331
JWE+c+p2	0.8407	0.8848	**0.9486**	**0.6618**
JWE+c+p3	**0.8505**	**0.9188**	0.9371	0.6250
JWE+r+p1	0.7553	0.8198	0.8171	0.5551
JWE+r+p2	0.8185	0.8656	0.9143	0.6397
JWE+r+p3	0.8416	0.9010	0.9200	0.6434
JWE-n	0.8229	0.8803	0.9028	0.6286

Table 2: Results on word analogy reasoning. The configurations are the same of the ones used in Table 1.

The results in Table 2 show that JWE outperforms the baselines on all categories' word analogy tasks. Different from the results on the word similarity task, JWE with components consistently performs better than JWE with radicals and JWE without either radicals or components. It demonstrates the necessary of delving deeper into fine-grained components for complex semantic reasoning tasks.

3.4 Case Studies

In addition to evaluating the benefits of incorporating subword information for Chinese word embeddings, it would be interesting to see the relationships of the embeddings of words, characters, and subcharacter components as they are embedded into a same continuous vector space.

照 (photograph)	照片 (photo) 相片 (photo) 拍照 (photograph) 护照 (passport) 照相 (photography)
河 (river)	黄河 (the Yellow River) 河流 (river) 河道 (watercourse) 运河 (canal) 河南 (Henan province)

Table 3: Closest words of characters 照 (photograph) and 河 (river).

Component	疒 (illness)
Closest characters	疗 (cure) 症 (symptom) 痛 (pain) 疮 (sore) 患 (suffer) 痒 (itch) 疳 (infantile malnutrition) 病 (disease) 肿 (swelling)
Closest words	治疗 (cure) 病症 (symptom) 复发 (recurrence) 疼痛 (pain) 症状 (symptom) 腹绞痛 (abdominal pain) 患者 (patients) 癫痫 (epilepsy) 疾病 (disease) 疗法 (therapy)

Table 4: Closest characters and closest words of the component 疒 (illness).

We evaluate the embeddings' abilities of uncovering the semantic relatedness of words, characters, and subcharacter components through case studies. The similarities between them are computed by the cosine similarities of their embeddings. Take two Chinese character 照 (photograph) and 河 (river) as examples, we list their closest words in Table 3. We can see that most of the closest words are semantically related to the corresponding character.

We further take the component 疒 (illness) as an example and list its closest characters and words in Table 4. All of the closest characters and words are semantically related to the component 疒 (illness). Most of them have the component 疒 (illness). 患 (suffer), 肿 (swelling), and 患者 (patients) do not have the component 疒 (illness), but they are also semantically related to 疒 (illness). It shows that JWE does not overuse the component information but leverages both the external context co-occurrence information and internal subword morphological information well.

4 Conclusion and Future Work

In this paper, we propose a model to jointly learn the embeddings of Chinese words, characters, and subcharacter components. Our approach makes full use of subword information to enhance Chinese word embeddings. Experiments show that our model substantially outperforms the baseline methods on Chinese word similarity computation and Chinese word analogy reasoning, and demonstrate the benefits of incorporating fine-grained components compared to just using characters.

There could be several directions to be explored for future work. First, we use the average operation to integrate the subcharacter components as the context to predict the target word. The structure of Chinese characters and the positions of components in the character may be considered to fully leverage the component information of Chinese characters. Second, for any target word, we simply use word context, character context, and subcharacter context to predict it and do not distinguish compositional words and non-compositional words. To solve this problem, attention models may be used to adaptively assign weights to word context, character context, and subcharacter context.

Acknowledgments

This paper was supported by HKUST initiation grant IGN16EG01, the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. 26206717), China 973 Fundamental R&D Program (No. 2014CB340304), and the LORELEI Contract HR0011-15-2-0025 with the US Defense Advanced Research Projects Agency (DARPA). Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. We also thank the anonymous reviewers for their valuable comments and suggestions that help improve the quality of this manuscript.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Joint learning of character and word embeddings. In *Proceedings of IJCAI*, pages 1236–1242.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

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

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Yanran Li, Wenjie Li, Fei Sun, and Sujian Li. 2015. Component-enhanced chinese character embeddings. In *Proceedings of EMNLP*, pages 829–834.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of CoNLL*, pages 104–113.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Proceedings of NIPS*, pages 1081–1088.

Jerome L Myers, Arnold Well, and Robert Frederick Lorch. 2010. *Research design and statistical analysis*. Routledge.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of COLING*, pages 141–150.

Xinlei Shi, Junjie Zhai, Xudong Yang, Zehua Xie, and Chao Liu. 2015. Radical embedding: Delving deeper to chinese radicals. In *Proceedings of ACL*, pages 594–598.

Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2016a. Inside out: Two jointly predictive models for word representations and phrase representations. In *Proceedings of AAAI*, pages 2821–2827.

Maosong Sun, Xinxiong Chen, Kaixu Zhang, Zhipeng Guo, and Zhiyuan Liu. 2016b. Thulac: An efficient lexical analyzer for chinese. *Technical Report*.

Yaming Sun, Lei Lin, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2014. Radical-enhanced chinese character embedding. In *International Conference on Neural Information Processing*, pages 279–286. Springer.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of ACL*, pages 1555–1565.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of EMNLP*, pages 1504–1515.

Jian Xu, Jiawei Liu, Liangang Zhang, Zhengyu Li, and Huanhuan Chen. 2016. Improve chinese word embeddings by exploiting internal structure. In *Proceedings of NAACL-HLT*, pages 1041–1050.

Rongchao Yin, Quan Wang, Peng Li, Rui Li, and Bin Wang. 2016. Multi-granularity chinese word embedding. In *Proceedings of EMNLP*, pages 981–986.

Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. 2015. Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of ACL*, pages 250–259.

Exploiting Morphological Regularities in Distributional Word Representations

Arihant Gupta* **Syed Sarfaraz Akhtar*** **Avijit Vajpayee***
Arjit Srivastava **Madan Gopal Jhanwar** **Manish Shrivastava**
{arihant.gupta, syed.akhtar, arjit.srivastava, madangopal.jhanwar}@research.iiit.ac.in,
avijit@inshorts.com,
manish.shrivastava@iiit.ac.in
Language Technologies Research Center(LTRC)
Kohli Center On Intelligent Systems (KCIS)
International Institute of Information Technology Hyderabad

Abstract

We present an unsupervised, language agnostic approach for exploiting morphological regularities present in high dimensional vector spaces. We propose a novel method for generating embeddings of words from their morphological variants using morphological transformation operators. We evaluate this approach on MSR word analogy test set (Mikolov et al., 2013d) with an accuracy of 85% which is 12% higher than the previous best known system.

1 Introduction

Vector representation of words are presently being used to solve a variety of problems like document classification (Sebastiani, 2002), question answering (Tellex et al., 2003) and chunking (Turian et al., 2010).

Word representations capture both syntactic and semantic properties (Mikolov et al., 2013d) of natural language. Soricut and Och (2015) exploited these regularities to generate prefix/suffix based morphological transformation rules in an unsupervised manner. These morphological transformations were represented as vectors in the same embedding space as the vocabulary.

Using Soricut's transformation rules, the major problem is identifying correct rule to apply to a word, i.e. if we have to generate an embedding for "runs", which rule to apply on "run". Experimental results showed that "walk - walks" gives better results than rules like "invent - invents" or "object - objects" in generating word embedding for "runs". In this paper, we try to explore if we can harness this morphological regularity in a much better way, than applying a single rule using vector arithmetic.

Hence, we tried to come up with a global transformation operator, which aligns itself with the source word, to give best possible word embedding for target word. We will have a single transformation operator for each rule, irrespective of the form of root word (like verb or a noun). Our transformation operator is in the form of a matrix, which when applied on a word embedding (cross product of vector representation of word with transformation matrix) gives us a word embedding for target word.

The intuition is not to solve for "invent is to invents as run is to ?" or "object is to objects as run is to ?", but instead we are solving for "walk is to walks, object is to objects, invent is to invents, ... as run is to ?". A transformation operator aims to be a unified transition function for different forms of the same transition.

The idea of projection learning has been applied to a multitude of tasks such as in the learning of cross lingual mappings for translation of English to Spanish (Mikolov et al., 2013b). Our approach has its basis on the same lines but with a different formulation and end goal to learn morphological rules rather than semantic associations and translational constraints (as done by Mikolov).

In summary, the main contributions of this paper is a new method to harness morphological regularities present in high dimensional word embeddings. Using this method, we present state of the art results on MSR word analogy dataset.

This paper is structured as follows. We first discuss the corpus used for training the transformation operators in section 2. In section 3, we discuss how these transformation operators are trained. Later in sections 4, we analyze and discuss the results of our experiments. We finish this paper with future scope of our work in section 5.

* These authors contributed equally to this work.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 292–297
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

2 Datasets

We are using word embeddings trained on Google News corpus (Mikolov et al., 2013c) for our experiments. For the model trained in this paper, we have used the Skip-gram (Mikolov et al., 2013a) algorithm. The dimensionality has been fixed at 300 with a minimum count of 5 along with negative sampling. As training set and for estimating the frequencies of words, we use the Wikipedia data (Shaoul, 2010). The corpus contains about 1 billion tokens.

The MSR dataset (Mikolov et al., 2013d) contains 8000 analogy questions. This data set has been used by us for testing our model. The relations portrayed by these questions are morpho-syntactic, and can be categorized according to parts of speech - adjectives, nouns and verbs. Adjective relations include comparative and superlative (good is to best as smart is to smartest). Noun relations include singular and plural, possessive and non-possessive (dog is to dog's as cat is to cat's). Verb relations are tense modifications (work is to worked as accept is to accepted).

For all the experiments, we have calculated the fraction of answers correctly answered by the system on MSR word analogy dataset.

3 Transformation Matrix

The thresholds mentioned in this section have been determined after empirical fine tuning.

To compute the transformation matrix of a rule, we first extract in an unsupervised way all the word pairs following that transition rule. For example, in case of the rule <null,s>, we find word pairs such as <boy,boys>, <object,objects> and <invent,invents>. In this paper, we used the data structure TRIE for computational optimization. However, we don't want cases which do not follow the general regularity of a rule. One such case may be <hat,hated> which does not follow the general trend of the transformation <null,ed>. For eliminating these cases, we set a threshold of cosine similarity of the word vectors of the two words of the pair at 0.2. Also, the frequency of both these words should be greater than 1000 (so that they are well trained). Since our transformation matrix is derived from all the word pairs following a particular transition rule, we carefully use only those word pairs which are of high frequency. We do so because highly frequent words have better trained word embeddings.

Suppose we get "N" highly frequent word pairs following the same regularity(transition rule). For our experiments, the lower threshold of "N" is set at 50. Dimensions of word embedding of a word in our model is "D". Using first word of our "N" chosen word pairs, we create a matrix "A" of dimensions N*D, where each row is vector representation of a word. Similarly, we create another matrix B, of similar dimensions as A, using second word of our chosen word pairs.

We now propose that a matrix "X" (our transformation matrix) exists such that,

$$
\begin{aligned}
A * X &= B \\
or, X &= A^{-1} * B
\end{aligned}
\tag{1}
$$

(all instances of A that we encountered were non-singular). Our matrix "X" will be of dimensions "D*D" and when applied to a word embedding (matrix of dimensions 1*D, it gives a matrix of dimensions 1*D as output), it results in the word embedding of the transformed form of the word.

Due to inverse property of a matrix, it accurately remembers the word pairs used for computing. The matrix also appears to align itself with the word embedding of other words (not used for its training) to transform them according to the rule that the matrix follows. Some interesting results are shown in table 1.

Word1	Word2	Word3	Operator	Word4	Cosine
decides	decided	studies	<s , d>	studied	0.89
reach	reaches	go	<null , es>	goes	1.0
ask	asks	reduce	<null , s>	reduces	0.91

Table 1: Some example results of transformation operators.

While testing, we extract the syntactic transition using the first two words of the analogy question. For example, for pairs like <reach, reached>, < walk, walked>, we are able to extract that they follow <null, ed> rule syntactically. But, for <go, went>, we are not able to find any transformation operator after syntactic analysis, and for such cases, we fall back on CosSum/CosMul (Levy et al., 2014) approaches as our backup. Mikolov et al. showed that relations between words are reflected to a large extent in the offsets between their vector embeddings (queen - king = woman - man), and thus the vector of the hidden word b^* will be similar to the vector $b - a + a^*$, suggesting that the

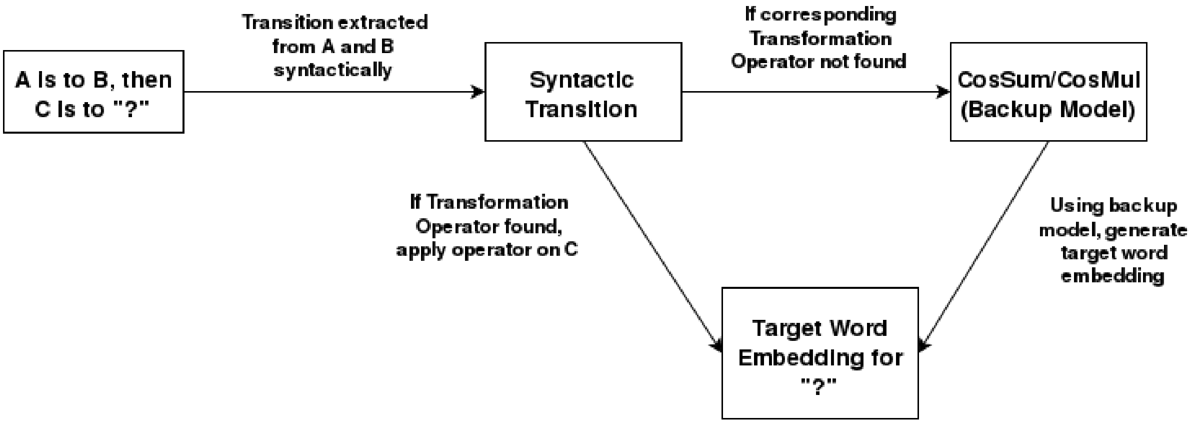

A is to B, then C is to "?"
Transition extracted from A and B syntactically
Syntactic Transition
If corresponding Transformation Operator not found
CosSum/CosMul (Backup Model)
If Transformation Operator found, apply operator on C
Using backup model, generate target word embedding
Target Word Embedding for "?"

Figure 1: System Workflow

analogy question can be solved by optimizing:

$$\underset{b^* \in V}{arg\ max}\,(sim(b^*, b - a + a^*)) \tag{2}$$

where V is the vocabulary and sim is a similarity measure. Specifically, they used the cosine similarity measure, defined as:

$$cos(u, v) = \frac{u\,.\,v}{||u||\,.\,||v||} \tag{3}$$

resulting in:

$$\underset{b^* \in V}{arg\ max}\,(cos(b^*, b - a + a^*)) \tag{4}$$

Equation 4 has been referred to as CosAdd model.

While experimenting, Omer Levy (Levy et al., 2014) found that for an analogy question “London is to England as Baghdad is to - ?”, using CosAdd model, they got *Mosul* - a large Iraqi city, instead of *Iraq* which is a country, as an answer. They were seeking for Iraq because of its similarity to England (both are countries), similarity to Baghdad (similar geography/culture) and dissimilarity to London (different geography/culture). While Iraq was much more similar to England than Mosul was (because both Iraq and England are countries), the sums were dominated by the geographic and cultural aspect of the analogy.

Hence to achieve better balancing among different aspects of similarity, they proposed a new model, where they moved from additive to multiplicative approach:

$$\underset{b^* \in V}{arg\ max}\,\frac{cos(b^*, b)\,.\,cos(b^*, a^*)}{cos(b^*, a) + \epsilon} \tag{5}$$

$$(\epsilon = 0.001\ to\ prevent\ division\ by\ zero)$$

This was equivalent to taking the logarithm of each term before summation, thus amplifying the differences between small quantities and reducing the differences between larger ones. This model has been referred to as CosMul model.

Even though our transformation operator can handle any sort of transformation, but if we are not able to detect the rule syntactically, we are not able to determine which transformation operator to use, and hence, we fall back on CosSum/CosMul. Like for the above mentioned examples, we will use transformation operator (if existing) for transformations like <reach, reached>, since we can find the rule syntactically, but for <go, went>, we can not, since we can not extract the corresponding rule itself - even if the matrix can handle such transitions.

If a transformation matrix exists for a transition rule, we apply the corresponding transformation matrix on the word embedding of the third word and search the whole vocabulary for the word with an embedding most similar to the transformed embedding (ignoring the third word itself). If the similarity of the resultant word's embedding with our transformed embedding is less than 0.68 (determined empirically) or the transformation matrix itself does not exist, we fall back on the CosSum/CosMul techniques.

Levy et. al. (2015) proposed the systems CosSum and CosMul in which they showed that tuning the hyperparameters has a significant impact on the performance. Hyperparameters are all the modifications and system design choices which are a part of the final algorithm.

Figure 1 gives an overview of how target word embedding is generated using transformation operators and our backup models.

4 Result and Analysis

Model	CosSum	CosSum w/ M	CosMul	CosMul w/ M
SGNS-L	0.69	-	**0.729**	-
Glove-L	0.628	-	0.685	-
SG	0.269	0.554	0.282	0.566
GN	0.646	0.718	0.67	**0.733**
GN-SG Hybrid	0.674	0.835	0.698	**0.85**

Table 2: Scores on MSR word analogy test set.

Word1	Word2	Word3	Operator	Word4	Cosine
decides	decided	studies	<s , d>	studied	0.89
reach	reaches	go	<null , es>	goes	1.0
member	members	school	<null , s>	schools	0.88
ask	asks	reduce	<null , s>	reduces	0.91
resident	residents	rate	<null , s>	rates	0.86
get	gets	show	<null , s>	shows	0.83
higher	highest	stricter	<r , st>	strictest	1.0
wild	wilder	harsh	<null , er>	harsher	0.91

Table 3: Example results of transformation operators for regular transformations.

Word1	Word2	Word3	Operator	Word4	Cosine
joined	joins	became	<ed , s>	becomes	0.68
turned	turns	said	<ed , s>	says	0.74
learn	learned	build	<null , ed>	built	0.80
support	supported	see	<null , ed>	saw	0.72

Table 4: Example results of transformation operators for irregular transformations.

In table 2, GN denotes the scores of Google-News word embeddings on the test set. SGNS-L and Glove-L (Levy et al., 2015) denote the results of Skip-gram with negative sampling and Glove word embeddings respectively, both trained on large datasets. SG denotes the scores of our word2vec trained model (on 1B tokens). "w/ M" implies that we have used matrix arithmetic (along with CosSum/CosMul as backup) for word analogy answering questions. Our model uses "CosSum" and "CosMul" as backup transformation method in case a transformation operator (matrix) does not exist. We see that the results of GN+Matrix are better than the previously used models.

However, one thing we noticed was that the model trained on Google-News did not contain words with apostrophe sign(s) and 1000 out of 8000 words in MSR word analogy test set contained apostrophe sign(s). Also, we noticed that in SG, the matrix approach was able to answer word analogy queries where words contained apostrophe sign(s), with an accuracy of 93.7% since it is a very common transformation - which resulted in well trained transformation matrix. So, we used SG as a backup for words which were not found in GN. The results of this hybrid model are denoted by GN-SG Hybrid. We see that this model performs considerably better than the existing state of the art system.

Word1	Word2	Word3	Operator	Word4	Cosine
reach	reached	go	<null , ed>	went	0.80
recognize	recognizes	be	<null , s>	is	0.70

Table 5: Example results of transformation operators for complete change of word form.

As we can see in table 3, our approach works really well for analogy questions where target word experiences regular transformation, i.e. the transformation type is simple addition/subtraction of suffix/prefix.

In table 4 and table 5 we observe that transformations are irregular transformations i.e there is slight change in word form while addition/subtraction of suffix/prefix or there is complete change in word form in the target word of our analogy question. This is an interesting observation, because even though our rule extraction (as explained above) is syntactic in nature, our method still learns and can apply transformation rules on words which undergo such irregular/complete transformations.

In operator "<null,s>", we see that our transformation matrix works pretty well irrespective of the form the word. For example, it works for "school-schools" and "reduce-reduces" which are noun and verb word pairs respectively. Our approach works by statistically creating global transformation operators and is agnostic in applying them (i.e. applied on a verb or a noun). Our transformation rules learn from both noun transitions and verb transitions and hence, even though we agree that linguistically there is a difference between noun and verb transitions, our approach performed better than previously existing systems

We also observed that in some cases, cosine similarity score is 1. This is mostly because "stricter-strictest" was used for training transformation matrix of "<r,st>" operator.

Although our cosine scores for irregular/complete transformations are not that high with respect to scores for regular transformations, our system still performs at par or better than previous known systems. It is still able to predict words with high accuracy using its limited training corpora.

These observations can also help us analyze how certain complex transformations (irregular/complete) still behave similar to their regular counterpart computationally, as is apparent from our transformation matrix - which has learnt itself from rules that were extracted via all possible prefix and suffix substitutions from w1 to w2, and thus irregular/complete transformations would not be present in training our transformation matrix (where w1 and w2 belong to our vocabulary V - the size of our corpus).

We conclude that our matrix is able to harness morphological regularities present in word pairs used for training.

5 Future Work

The main application of this approach lies in its ability to generate representations for unseen/unreliable words on the go. If we encounter a word such as "preparedness" for which we do not have a representation or our representation is not reliable, we can identify any reliable form of the word, say "prepared" and apply <null,ness> operator on it, resulting in a representation for "preparedness". In a similar case, we can generate embeddings for words such as "unpreparedness" from "prepared" by sequentially applying <null,ness> and a prefix operator trained in a similar manner - <null,un>. Overall, this results in a much larger vocabulary than of the model initially being used.

We observe that for a transformation matrix to exist, we need enough word pairs (frequent enough to be included) to train a transformation matrix, and to cover majority of the transformation rules. Since many languages face problem of data scarcity, we face problem of "missing" transformation matrix for a transformation rule.

We will also explore if we can generate better word embeddings for words that are not highly frequent (hence less reliable), by applying transformation operators on their morphological variants, which are highly frequent and hence more reliable. For example, if for word pair "<walked,walking>", "walked" was not highly frequent in our corpora, we would not include it in training our transformation matrix for operator "<ed,ing>" because "walked" didn't have a well trained word embedding (being less frequent). But, if we have walk as highly frequent, and a transformation matrix for rule "<null,ed>", we can generate a better word embedding for "walked", and in turn use "<walked,walking>" for training our transformation matrix for rule "<ed,ing>".

In our current approach, for problem "If A is to B, then C is to ?", we do syntactic analysis on "A" and "B" to find out the transformation rule (and hence the transformation matrix) to be applied on "C" to find our "?". Rather than doing syntactic analysis, we will focus on finding out the correct transformation matrix by applying all transformation matrices on "A" and then analyzing output of each matrix. The one which will give closest result to "B" can be safely assumed to be the correct rule (transformation matrix), and hence will be applied on "C" to find "?". We will also analyze this approach's impact in terms of space and time complexities with respect to our current system. This will also enable us to find transformation matrix for word pairs which do not follow a syntactic transformation.

References

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

Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. 2014. Linguistic regularities in sparse and explicit word representations. In *CoNLL*, pages 171–180.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013d. Linguistic regularities in continuous space word representations. In *hlt-Naacl*, volume 13, pages 746–751.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Cyrus Shaoul. 2010. The westbury lab wikipedia corpus. *Edmonton, AB: University of Alberta*.

Radu Soricut and Franz Josef Och. 2015. Unsupervised morphology induction using word embeddings. In *HLT-NAACL*, pages 1627–1637.

Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 41–47. ACM.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Exploiting Word Internal Structures for Generic Chinese Sentence Representation

Shaonan Wang[1,2], **Jiajun Zhang**[1,2], **Chengqing Zong**[1,2,3]
[1] National Laboratory of Pattern Recognition, CASIA, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] CAS Center for Excellence in Brain Science and Intelligence Technology, Shanghai, China
{shaonan.wang,jjzhang,cqzong}@nlpr.ia.ac.cn

Abstract

We introduce a novel mixed character-word architecture to improve Chinese sentence representations, by utilizing rich semantic information of word internal structures. Our architecture uses two key strategies. The first is a mask gate on characters, learning the relation among characters in a word. The second is a max-pooling operation on words, adaptively finding the optimal mixture of the atomic and compositional word representations. Finally, the proposed architecture is applied to various sentence composition models, which achieves substantial performance gains over baseline models on sentence similarity task.

1 Introduction

To understand the meaning of a sentence is a prerequisite to solve many natural language processing problems. Obviously, this requires a good representation of the meaning of a sentence. Recently, neural network based methods have shown advantage in learning task-specific sentence representations (Kalchbrenner et al., 2014; Tai et al., 2015; Chen et al., 2015a; Cheng and Kartsaklis, 2015) and generic sentence representations (Le and Mikolov, 2014; Hermann and Blunsom, 2014; Kiros et al., 2015; Kenter et al., 2016; Wang et al., 2017). To learn generic sentence representations that perform robustly across tasks as effective as word representations, Wieting et al. (2016b) proposes an architecture based on the supervision from the Paraphrase Database (Ganitkevitch et al., 2013).

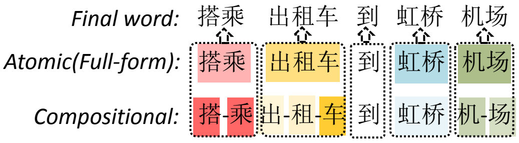


Figure 1: An example sentence that consists of five words as “搭乘(take) 出租车(taxi) 到(to) 虹桥(Hongqiao) 机场(airport)”. Most of these words are compositional, namely word “搭乘” consists of characters “搭(take)” and “乘(ride)”, word “出租车” constitutes characters “出(out)”, “租(rent)” and “车(car)”, and word “机场” is composed of characters “机(machine)” and “场(field)”. The color depth represents (1) contributions of each character to the compositional word meaning, and (2) contributions of the atomic (which ignore inner structures) and compositional word to the final word meaning. The deeper color means more contributions.

Despite the fact that Chinese has unique word internal structures, there is no work focusing on learning generic Chinese sentence representations. In contrast to English, Chinese characters contain rich information and are capable of indicating semantic meanings of words. As illustrated in Figure 1, the internal structures of Chinese words express two characteristics: (1) Each character in a word contribute differently to the compositional word meaning (Wong et al., 2009) such as the word “出租车(taxi)”. The first two characters “出租(rent)” are descriptive modifiers of the last character “车(car)”, and make the last character play the most important role in expressing word meaning. (2) The atomic and compositional representations contribute differently to different types of words (MacGregor and Shtyrov, 2013). For instance, the meaning of “机场(airport)”, a low-frequency word, can be better expressed by the compositional word representation, while the non-transparent word “虹桥(Hongqiao)” is better expressed by the atomic word representation.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 298–303
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

The word internal structures have been proven to be useful for Chinese word representations. Chen et al. (2015b) proposes a character-enhanced word representation model by adding the averaged character embeddings to the word embedding. Xu et al. (2016) extends this work by using weighted character embeddings. The weights are cosine similarities between embeddings of a word's English translation and its constituent characters' English translations. However, their work calculates weights based on a bilingual dictionary, which brings lots of mistakes because words in two languages do not mantain one-to-one relationship. Furthermore, they only consider the first characteristic of word internal structures, but ignore the contributions of the atomic and compositional word to the final word meaning. Similar ideas of adaptively utilizing character level informations have also been investigated in English recently (Hashimoto and Tsuruoka, 2016; Rei et al., 2016; Miyamoto and Cho, 2016). It should be noted that these studies are not focus on learning sentence embeddings.

In this paper, we explore word internal structures to learn generic sentence representations, and propose a mixed character-word architecture which can be integrated into various sentence composition models. In the proposed architecture, a mask gate is employed to model the relation among characters in a word, and pooling mechanism is leveraged to model the contributions of the atomic and compositional word embeddings to the final word representations. Experiments on sentence similarity (as well as word similarity) demonstrate the effectiveness of our method. In addition, as there are no publicly available Chinese sentence similarity datasets, we build a dataset to directly test the quality of sentence representations. The code and data will be publicly released.

2 Model Description

The problem of learning compositional sentence representations can be formulated as $g^{comp} = f(x)$, where f is the **composition function** which combines the **word representations** $x = \langle x_1, x_2, ..., x_n \rangle$ into the compositional sentence representation g^{comp}.

2.1 Mixed Character-Word Representation

In our method, the final word representation is a fusion of the atomic and compositional word embeddings. The atomic word representation is calculated by projecting word level inputs into a high-dimensional space by a look up table, while the compositional word representation is computed as a gated composition of character representations:

$$x_i^{comp} = \sum_{j=1}^{m} v_{ij} \cdot c_{ij}, \qquad (1)$$

where c_{ij} is the *j*-th character representation in the *i*-th word. The mask gate $v_{ij} \in \mathbb{R}^d$ controls the contribution of the *j*-th character in the *i*-th word. This is achieved by using a feed-forward neural network operated on the concatenation of a character and a word, under the assumption that the contribution of a character is correlated with both character itself and its relation with the corresponding word:

$$v_{ij} = tanh(W \cdot [c_{ij}; x_i]), \qquad (2)$$

where $W \in \mathbb{R}^{d \times 2d}$ is a trainable parameter. The proposed mask gate is a vector instead of a single value, which introduces more variations to character meaning in the composition process.

Then, the atomic and compositional word representations are mixed with max-pooling:

$$x_i^{final} = \max_{k=1}^{d}(x_{ik}^{atomic}, x_{ik}^{comp}), \qquad (3)$$

the *max* is an element-wise function to capture the most important features (i.e., the highest value in each dimension) in the two word representations.

2.2 Sentence Composition Model

Given word embeddings, we make a systematic comparison of five different composition models for sentence representations as follows:

1. $g = Average(x) = \frac{1}{n}\sum_{i=1}^{n} x_i$
2. $g = Matrix(x) = \frac{1}{n}\sum_{i=1}^{n} f(W_m x_i)$
3. $g = Dan(x) = f(W_d(\frac{1}{n}\sum_{i=1}^{n} x_i) + b)$
4. $g = RNN(x) = f(W_x x_i + W_h h_{i-1} + b)$
5. $g = LSTM(x) = o_t \odot f(c_i)$, where $c_i = f_i \cdot c_{i-1} + i_i \cdot \widetilde{c_i}$ and $\widetilde{c_i} = \sigma(W_{xc} x_i + W_{hc} h_{i-1})$

Average model, as the simplest composition model, represents sentences with averaged word vectors which are updated during training. The

Matrix and Dan models are proposed in Zanzotto et al. (2010) and Iyyer et al. (2015), respectively. By using matrix transformations and nonlinear functions, the two models represent sentence meaning in a more flexible way (Wang and Zong, 2017). We also include RNN and LSTM models, which are widely used in recent years. The parameters $\{i_t, f_t, o_t\} \in \mathbb{R}^d$ denote the input gate, the forget gate and the output gate, respectively. $c_t \in \mathbb{R}^d$ is the short-term memory state to store the history information. $\{W_m, W_d, W_x, W_h, W_{xc}, W_{hc}\} \in \mathbb{R}^{d \times d}$ are trainable parameters. h_{i-1} denotes representations in hidden layers. Sentence representations in RNN and LSTM models are hidden vectors of the last token.

2.3 Objective Function

This paper aims to learn the general-purpose sentence representations based on supervision from Chinese paraphrase pairs. Following the approach of Wieting et al. (2016b), we employ the max-margin objective function to train sentence representations by maximizing the distance between positive examples and negative examples.

3 Experimental Setting and Dataset

3.1 Experimental Setting

We construct four groups of models (G1˜G4) which serve as baselines to test the proposed mixed character-word models (*G5*). Group *G1* includes six baseline models, which have shown impressive performance in English. The first two are averaged word vectors and averaged character vectors. Followed by PV-DM model which uses auxiliary vectors to represent sentences and trains them together with word vectors, and FastSent model which utilizes a encoder-decoder model and encodes sentences as averaged word embeddings. The last two are Char-CNN model which is CNN model with character n-gram filters, and Charagram model which represents sentences with a character n-gram count vector. Group *G2* are the sentence representation models proposed by Wieting et al. (2016b), which utilize only word level information. We also compared our method with word representation models of Chen et al. (2015b) and Xu et al. (2016) in Group *G3* and *G4* respectively, by incorporate them into five sentence composition models in Section 2.2.

In all models, the word and character embeddings are initialized with 300-dimension vectors trained by Skip-gram model (Mikolov et al., 2013) on a corpus with 3 billion Chinese words. All models are implemented with Theano (Bergstra et al., 2010) and Lasagne (Dieleman et al., 2015), and optimized using Adam (Kingma and Ba, 2014). The hyper-parameters[1] are selected by testing different values and evaluating their effects on the development set. In this paper, we run all experiments 5 times and report the mean values.

3.2 Training Dataset

The training dataset is a set of paraphrase pairs in which two sentences in each pair represent the same meanings. Specifically, we extract Chinese paraphrases in machine translation evaluation corpora NIST2003[2] and CWMT2015[3]. Moreover, we select aligned sub-sentence pairs between paraphrases to enlarge the training corpus. Specifically, we first segment the sentences into sub-sentences according to punctuations of *comma, semicolon, colon, question mark, ellipses, and periods*. Then we pair all sub-sentences between a paraphrase and select sub-sentence pairs (s_1, s_2) which satisfy the following two constraints: (1) the number of overlapping words of sub-sentence s_1 and s_2 should meet the condition: $0.9 > len(overlap(s_1, s_2))/min(len(s_1), len(s_2)) > 0.2$, where $len(s)$ denotes the number of words in sentence s; (2) the relative length of sub-sentence should meet the condition: $max(len(s_1), len(s_2))/min(len(s_1), len(s_2)) <= 2$. Finally, we get 30,846 paraphrases (18,187 paraphrases from NIST including 11,413 sub-sentence pairs, and 12,659 paraphrases from CWMT which include 7,912 sub-sentence pairs).

3.3 Testing Dataset

We also build the testing dataset, which are sentence pairs collocated with human similarity ratings. We choose candidate sentences from the People's Daily and Baidu encyclopedia corpora. To assure sentence pairs to be representative of the full variation in semantic similarity, we choose

[1] We use a mini-batch of 25 and tune the initial learning rate over {0.001, 0.005, 0.0001, 0.0005}. For the Dan and the Matrix models, we tune over activation function (tanh or linear or rectified linear unit) and number of layers (1 or 2).

[2] which contains 1,100 English sentences with 4 Chinese translations and can be found at: https://catalog.ldc.upenn.edu/LDC2006T04

[3] which contains 1,859 English sentences with 4 Chinese translations and can be found at: http://www.ai-ia.ac.cn/cwmt2015/evaluation.html

high similarity sentence pairs[4] and then randomly pair the single sentences to construct low similarity sentence pairs. To collect human similarity ratings for sentence pairs, we use online questionnaire[5] and follow the gold standard[6] to guide the rating process of participants. The subjects are paid 7 cents for rating each sentence pair within a range of 0 5 score. In total, we obtain 104 valid questionnaires and every sentence pair is evaluated by average 8 persons. We use the average subjects' ratings for one paraphrase as its final similarity score, and the higher score means that the two sentences have more similar meaning. We then randomly partition the datasets into test and development splits in 9:1.

4 Results and Discussion

We use the Pearson's correlation coefficient to examine relationships between the averaged human ratings and the predicted cosine similarity scores of all models. Moreover, the Wilcoxon's test shows that significant difference ($p < 0.01$) exits between our models with baseline models.

From Table 1, we can see superiority of the proposed mixed character-word models (*G5*), which have significantly improved the performance over both word and character-word based models. This result indicates that it is important to find the appropriate way to fuse character and word level informations. Using mask gate alone and max pooling alone yield an improvement of 1.05 points and 0.83 points respectively, and using both strategies improves the averaged character-word models by 1.52 points. Another observation is that models with character level information (*G3, G4, G5*) perform better than word based models (*G2*), which indicates the great potential of Chinese characters in learning sentence representations. Comparing different composition functions, we can see that two simple models outperform others in all groups: the DAN model and the Matrix model. The simplest Average model achieves competitive results while the most complex LSTM model does not show advantages.

Group	Model	Test
G1: Baselines	Add (character)	0.6737
	Add (word)	0.7518
	PV-DM (Le and Mikolov, 2014)	0.7561
	FastSent (Hill et al., 2016)	0.7369
	Char-CNN (Kim et al., 2016)	0.8095
	Charagram(Wieting et al., 2016a)	0.8382
G2: Word level (Wieting et al., 2016b)	Average	0.8199
	Matrix	0.8382
	Dan	0.8385
	RNN	0.8121
	LSTM	0.7834
G3: Averaged Character-Word (Chen et al., 2015)	Average	0.8245
	Matrix	0.8427
	Dan	0.8407
	RNN	0.8185
	LSTM	0.7895
G4: Weighted Character-Word (Xu et al., 2016)	Average	0.8196
	Matrix	0.8428
	Dan	0.8413
	RNN	0.8344
	LSTM	0.7858
G5: Mixed Character-Word (Ours)	**Average**	**0.8471**
	Matrix	**0.8517**
	Dan	**0.8521**
	RNN	**0.8408**
	LSTM	**0.8000**

Table 1: Correlation coefficients of model predictions with subject similarity ratings on Chinese sentence similarity task. The bold data refers to best among models with same composition function.

4.1 Effects of Mask Gate and Max Pooling

The mask gate assigns different weights to characters in a word, hopefully leading to better word representations. To intuitively show effects of the mask gate, we check characters whose l2-norm increase after applying the mask gate approach. We find that characters like "罪(crime)" in "罪状(guilty)", "虎 (tiger)" in "美洲虎 (jaguar)" and "瓜 (melon)" in "黄瓜 (cucumber)" achieve more weights. The above results show that the mask gate approach successfully model the first characteristic of word internal structure (i.e., assigning more weights to key characters). To quantitatively display the results, we extract the word representations calculated by the five composition models in four different groups and evaluate their quality on WordSim-297 dataset[7] using the Pearson correlation method. As shown in Table 2, the mask gate approach significantly improves the quality of word representations.

[4] Here we choose high similarity sentence pairs by using edit distance and human post-processing.

[5] https://wj.qq.com/

[6] http://alt.qcri.org/semeval2015/task2/index.php?id=semantic-textual-similarity-for-english

[7] https://github.com/Leonard-Xu/CWE/tree/master/data

	G2	*G3*	*G4*	*G5*(Ours)
Average	0.4311	0.4584	0.4789	**0.5245**
Dan	0.4470	0.5410	0.5561	**0.5716**
Matrix	0.4496	0.5458	0.5548	**0.5694**
RNN	0.4562	0.5656	0.5550	**0.5674**
LSTM	0.4535	0.5674	0.5627	**0.5734**

Table 2: Correlation coefficients of model predictions with subject similarity ratings on Chinese word similarity task, where *G2* $\sim$ *G5* are the same as in Table 1.

The max-pooling approach is supposed to model different contributions of the atomic and compositional word vectors to the final word vector. To find out what have max-pooling method learned, we use contribution weights by calculating cosine similarities between the final word representation with the atomic and compositional word representations. The results show interesting relationships with word frequency. For high-frequency words, the contribution of compositional word representations are more dominant. While for low-frequency words, both high[8] and low contribution ratios of compositional word representations can be found. When looking into the words with the most lowest ratio, we find a large portion of English abbreviations like *NBA*, *BBC*, *GDP* etc., and a portion of metaphor words like "挂靴(retire, hanging boots)" and "扯皮(wrangle, pull skin)". Both kinds of these words are non-transparent, which indicates that the max-pooling method can successfully model the second characteristic of word internal structure and encode word transparency to some extent.

5 Conclusion and Further work

In this paper, we introduce a novel mixed character-word architecture to improve generic Chinese sentence representations by exploiting the complex internal structures of words. Extensive experiments and analyses have indicated that our models can encode word transparency and learn different semantic contributions across characters. We have also created a dataset to evaluate composition models of Chinese sentences, which could advance the research for related fields.

Future work includes applying the proposed method to other aspects of nominal semantics, such as understanding compound nouns in other languages, and to explore the compositionality of words and compounds.

Acknowledgement

The research work has been supported by the Natural Science Foundation of China under Grant No. 61333018 and No. 61403379.

References

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: A cpu and gpu math compiler in python. *In Proceedings of the Python for Scientific Computing Conference (SciPy).*

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Shiyu Wu, and Xuanjing Huang. 2015a. Sentence modeling with gated recursive neural network. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,*, pages 793–798.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huan-Bo Luan. 2015b. Joint learning of character and word embeddings. In *IJCAI*, pages 1236–1242.

Jianpeng Cheng and Dimitri Kartsaklis. 2015. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,*, pages 1531–1542.

Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, et al. 2015. Lasagne: First release. *Zenodo: Geneva, Switzerland.*

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *HLT-NAACL*, pages 758–764.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2016. Adaptive joint learning of compositional and non-compositional phrase embeddings. *Proceedings of the* 54^{th} *Annual Meeting of the Association for Computational Linguistics,*, pages 205–215.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. *Proceedings of the* 52^{nd} *Annual Meeting of the Association for Computational Linguistics,*, pages 58–68.

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

[8] The high ratio is more reasonable because low-frequency words generally learn poor atomic word representations.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. *Proceedings of the 53^{rd} Annual Meeting of the Association for Computational Linguistics*, pages 1681–1691.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Proceedings of the 52^{nd} Annual Meeting of the Association for Computational Linguistics,*, pages 655–665.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. *Processings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. *Advances in neural information processing systems*, pages 3294–3302.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *Proceedings of the 31^{st} International Conference on Machine Learning*, pages 1188–1196.

Lucy J MacGregor and Yury Shtyrov. 2013. Multiple routes for compound word processing in the brain: evidence from eeg. *Brain and language*, 126(2):217–229.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,*, pages 1992–1997.

Marek Rei, Gamal KO Crichton, and Sampo Pyysalo. 2016. Attending to characters in neural sequence labeling models. *Proceedings of COLING 2016, the 26^th International Conference on Computational Linguistics,*, pages 309–318.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Shaonan Wang, Jiajun Zhang, and Chengqing Zong. 2017. Learning sentence representation with guidance of human attention. *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*.

Shaonan Wang and Chengqing Zong. 2017. Comparison study on critical components in composition model for phrase representation. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(3):16.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016a. Charagram: Embedding words and sentences via character n-grams. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,*, pages 1504–1515.

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

Kam-Fai Wong, Wenjie Li, Ruifeng Xu, and Zhengsheng Zhang. 2009. Introduction to chinese natural language processing. *Synthesis Lectures on Human Language Technologies*, 2(1):1–148.

Jian Xu, Jiawei Liu, Liangang Zhang, Zhengyu Li, and Huanhuan Chen. 2016. Improve chinese word embeddings by exploiting internal structure. *Proceedings of NAACL-HLT 2016,*, pages 1041–1050.

Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. *Proceedings of the 23^{rd} International Conference on Computational Linguistics,*, pages 1263–1271.

High-risk learning:
acquiring new word vectors from tiny data

Aurélie Herbelot
Dept. of Translation and Language Sciences
Universitat Pompeu Fabra
aurelie.herbelot@cantab.net

Marco Baroni
Center for Mind/Brain Sciences
University of Trento
marco.baroni@unitn.it

Abstract

Distributional semantics models are known to struggle with small data. It is generally accepted that in order to learn 'a good vector' for a word, a model must have sufficient examples of its usage. This contradicts the fact that humans can guess the meaning of a word from a few occurrences only. In this paper, we show that a neural language model such as Word2Vec only necessitates minor modifications to its standard architecture to learn new terms from tiny data, using background knowledge from a previously learnt semantic space. We test our model on word definitions and on a nonce task involving 2-6 sentences' worth of context, showing a large increase in performance over state-of-the-art models on the definitional task.

1 Introduction

Distributional models (DS: Turney and Pantel (2010); Clark (2012); Erk (2012)), and in particular neural network approaches (Bengio et al., 2003; Collobert et al., 2011; Huang et al., 2012; Mikolov et al., 2013), do not fare well in the absence of large corpora. That is, for a DS model to learn a word vector, it must have seen that word a sufficient number of times. This is in sharp contrast with the human ability to perform *fast mapping*, i.e. the acquisition of a new concept from a single exposure to information (Lake et al., 2011; Trueswell et al., 2013; Lake et al., 2016).

There are at least two reasons for wanting to acquire vectors from very small data. First, some words are simply rare in corpora, but potentially crucial to some applications (consider, for instance, the processing of text containing technical terminology). Second, it seems that fast-mapping should be a prerequisite for any system pretending to cognitive plausibility: an intelligent agent with learning capabilities should be able to make educated guesses about new concepts it encounters.

One way to deal with data sparsity issues when learning word vectors is to use morphological structure as a way to overcome the lack of primary data (Lazaridou et al., 2013; Luong et al., 2013; Kisselew et al., 2015; Padó et al., 2016). Whilst such work has shown promising result, it is only applicable when there is transparent morphology to fall back on. Another strand of research has been started by Lazaridou et al. (2017), who recently showed that by using simple summation over the (previously learnt) contexts of a nonce word, it is possible to obtain good correlation with human judgements in a similarity task. It is important to note that both these strategies assume that rare words are special cases of the distributional semantics apparatus, and thus require separate approaches to model them.

Having different algorithms for modelling the same phenomenon means however that we need some meta-theory to know when to apply one or the other: it is for instance unclear at which frequency a rare word is not rare anymore. Further, methods like summation are naturally self-limiting: they create frustratingly strong baselines but are too simplistic to be extended and improved in any meaningful way. In this paper, our underlying assumption is thus that it would be desirable to build a single, all-purpose architecture to learn word representations from *any* amount of data. The work we present views fast-mapping as a component of an incremental architecture: the rare word case is simply the first part of the concept learning process, *regardless of how many times it will eventually be encountered.*

With the aim of producing such an incremen-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 304–309
Copenhagen, Denmark, September 7–11, 2017.

tal system, we demonstrate that the general architecture of neural language models like Word2Vec (Mikolov et al., 2013) is actually suited to modelling words from a few occurrences only, providing minor adjustments are made to the model itself and its parameters. Our main conclusion is that the combination of a heightened learning rate and greedy processing results in very reasonable one-shot learning, but that some safeguards must be in place to mitigate the high risks associated with this strategy.

2 Task description

We want to simulate the process by which a competent speaker encounters a new word in known contexts. That is, we assume an existing vocabulary (i.e. a previously trained semantic space) which can help the speaker 'guess' the meaning of the new word. To evaluate this process, we use two datasets, described below.

The definitional nonce dataset We build a novel dataset based on encyclopedic data, simulating the case where the context of the unknown word is supposedly maximally informative.[1] We first record all Wikipedia titles containing one word only (e.g. *Albedo, Insulin*). We then extract the first sentence of the Wikipedia page corresponding to each target title (e.g. *Insulin is a peptide hormone produced by beta cells in the pancreas.*), and tokenise that sentence using the Spacy toolkit.[2] Each occurrence of the target in the sentence is replaced with a slot (___).

From this original dataset, we only retain sentences with enough information (i.e. a length over 10 words), corresponding to targets which are frequent enough in the UkWaC corpus (Baroni et al. (2009), minimum frequency of 200). The frequency threshold allows us to make sure that we have a high-quality gold vector to compare our learnt representation to. We then randomly sample 1000 sentences, manually checking the data to remove instances that are, in fact, not definitional. We split the data into 700 training and 300 test instances.

On this dataset, we simulate first-time exposure to the nonce word by changing the label of the gold standard vector in the background semantic space, and producing a new, randomly initialised vector for the nonce. So for instance, *insulin* becomes *insulin_gold*, and a new random embedding is added to the input matrix for *insulin*. This setup allows us to easily measure the similarity of the newly learnt vector, obtained from one definition, to the vector produced by exposure to the whole Wikipedia. To measure the relative performance of various setups, we calculate the Reciprocal Rank (RR) of the gold vector in the list of all nearest neighbours to the learnt representation. We average RRs over the number of instances in the dataset, thus obtaining a single MRR figure (Mean Reciprocal Rank).

The Chimera dataset Our second dataset is the 'Chimera' dataset of (Lazaridou et al., 2017).[3] This dataset was specifically constructed to simulate a nonce situation where a speaker encounters a word for the first time in naturally-occurring (and not necessarily informative) sentences. Each instance in the data is a nonce, associated with 2-6 sentences showing the word in context. The novel concept is created as a 'chimera', i.e. a mixture of two existing and somewhat related concepts (e.g., a buffalo crossed with an elephant). The sentences associated with the nonce are utterances containing one of the components of the chimera, randomly extracted from a large corpus.

The dataset was annotated by humans in terms of the similarity of the nonce to other, randomly selected concepts. Fig. 1 gives an example of a data point with 2 sentences of context, with the nonce capitalised (*VALTUOR*, a combination of *cucumber* and *celery*). The sentences are followed by the 'probes' of the trial, i.e. the concepts that the nonce must be compared to. Finally, human similarity responses are given for each probe with respect to the nonce. Each chimera was rated by an average of 143 subjects. In our experiments, we simply replace all occurrences of the original nonce with a slot (___) and learn a representation for that slot. For each setting (2, 4 and 6 sentences), we randomly split the 330 instances in the data into 220 for training and 110 for testing.

Following the authors of the dataset, we evaluate by calculating the correlation between system and human judgements. For each trial, we calculate Spearman correlation (ρ) between the similarities given by the system to each nonce-probe pair, and the human responses. The overall result is the average Spearman across all trials.

[1] Data available at http://aurelieherbelot.net/resources/.

[2] https://spacy.io/

[3] Available at http://clic.cimec.unitn.it/Files/PublicData/chimeras.zip.

```
Sentences:
Canned sardines and VALTUOR between two slices of wholemeal bread and thinly spread Flora Original.
@@ Erm, VALTUOR, low fat dairy products, incidents of heart disease for those who have an olive oil rich diet.

Probes: rhubarb, onion, pear, strawberry, limousine, cushion
Human responses: 3, 2.86, 1.43, 2.14, 1.29, 1.71
```

Figure 1: An example chimera (VALTUOR).

3 Baseline models

We test two state-of-the art systems: a) Word2Vec (W2V) in its Gensim[4] implementation, allowing for update of a prior semantic space; b) the additive model of Lazaridou et al. (2017), using a background space from W2V.

We note that both models allow for some sort of incrementality. W2V processes input one context at a time (or several, if mini-batches are implemented), performing gradient descent after each new input. The network's weights in the input, which correspond to the created word vectors, can be inspected at any time.[5] As for addition, it also affords the ability to stop and restart training at any time: a typical implementation of this behaviour can be found in distributional semantics models based on random indexing (see e.g. QasemiZadeh et al., 2017). This is in contrast with so-called 'count-based' models calculated by computing a frequency matrix over a fixed corpus, which is then globally modified through a transformation such as Pointwise Mutual Information.

Word2Vec We consider W2V's 'skip-gram' model, which learns word vectors by predicting the context words of a particular target. The W2V architecture includes several important parameters, which we briefly describe below.

In W2V, predicting a word implies the ability to distinguish it from so-called *negative samples*, i.e. other words which are *not* the observed item. The number of negative samples to be considered can be tuned. What counts as a context for a particular target depends on the *window size* around that target. W2V features random resizing of the window, which has been shown to increase the model's performance. Further, each sentence passed to the model undergoes *subsampling*, a random process by which some words are dropped out of the input as a function of their overall frequency. Finally, the *learning rate* α measures how quickly the system learns at each training iteration. Traditionally, α is set low (0.025 for Gensim) in order not to overshoot the system's error minimum.

Gensim has an update function which allows us to save a W2V model and continue learning from new data: this lets us simulate prior acquisition of a background vocabulary and new learning from a nonce's context. As background vocabulary, we use a semantic space trained on a Wikipedia snapshot of $1.6B$ words with Gensim's standard parameters (initial learning rate of 0.025, 5 negative samples, a window of ± 5 words, subsampling $1e^{-3}$, 5 epochs). We use the skip-gram model with a minimum word count of 50 and vector dimensionality 400. This results in a space with $259,376$ word vectors. We verify the quality of this space by calculating correlation with the similarity ratings in the MEN dataset (Bruni et al., 2014). We obtain $\rho = 0.75$, indicating an excellent fit with human judgements.

Additive model Lazaridou et al. (2017) use a simple additive model, which sums the vectors of the context words of the nonce, taking as context the entire sentence where the target occurs. Their model operates on multimodal vectors, built over both text and images. In the present work, however, we use the semantic space described above, built on Wikipedia text only. We do not normalise vectors before summing, as we found that the system's performance was better than with normalisation. We also discard function words when summing, using a stopword list. We found that this step affects results very positively.

The results for our state-of-the-art models are shown in the top sections of Tables 1 and 2. W2V is run with the standard Gensim parameters, under the skip-gram model. It is clear from the results that W2V is unable to learn nonces from definitions ($MRR = 0.00007$). The additive model, on the other hand, performs well: an MRR of 0.03686 means that the median rank of the true vector is 861, out of a challenging

[4] Available at https://github.com/RaRe-Technologies/gensim.

[5] Technically speaking, standard W2V is not fully incremental, as it requires a first pass through the corpus to compute a vocabulary, with associated frequencies. As we show in §5, it however allows for an incremental interpretation, given minor modifications.

$259,376$ neighbours (the size of the vocabulary). On the Chimeras dataset, W2V still performs well under the sum model – although the difference is not as marked and possibly indicates that this dataset is more difficult (which we would expect, as the sentences are not as informative as in the encyclopedia case).

4 Nonce2Vec

Our system, Nonce2Vec (N2V),[6] modifies W2V in the following ways.

Initialisation: since addition gives a good approximation of the nonce word, we initialise our vectors to the sum of all known words in the context sentences (see §3). Note that this is not strictly equivalent to the pure sum model, as subsampling takes care of frequent word deletion in this setup (as opposed to a stopword list). In practice, this means that the initialised vectors are of slightly lesser quality than the ones from the sum model.

Parameter choice: we experiment with higher learning rates coupled with larger window sizes. That is, the model should take the risk of a) overshooting a minimum error; b) greedily considering irrelevant contexts in order to increase its chance to learn anything. We mitigate these risks through *selective training* and appropriate *parameter decay* (see below).

Window resizing: we suppress the random window resizing step when learning the nonce. This is because we need as much data as possible and accordingly need a large window around the target. Resizing would make us run the risk of ending up with a small window of a few words only, which would be uninformative.

Subsampling: With the goal of keeping most of our tiny data, we adopt a subsampling rate that only discards extremely frequent words.

Selective training: we only train the nonce. That is, we only update the weights of the network for the target. This ensures that, despite the high selected learning rate, the previously learnt vectors, associated with the other words in the sentence, will not be radically shifted towards the meaning expressed in that particular sentence.

Whilst the above modifications are appropriate to deal with the first mention of a word, we must ask in what measure they still are applicable when the term is encountered again (see §1). With a view to cater for incrementality, we introduce a notion of **parameter decay** in the system. We hypothesise that the initial high-risk strategy, combining high learning rate and greedy processing of the data, should only be used in the very first training steps. Indeed, this strategy drastically moves the initialised vector to what the system assumes is the right neighbourhood of the semantic space. Once this positioning has taken place, the system should refine its guess rather than wildly moving in the space. We thus suggest that the learning rate itself, but also the subsampling value and window size should be returned to more conventional standards as soon as it is desirable. To achieve this, we apply some exponential decay to the learning rate of the nonce, proportional to the number of times the term has been seen: every time t that we train a pair containing the target word, we set α to $\alpha_0 e^{-\lambda t}$, where α_0 is our initial learning rate. We also decrease the window size and increase subsampling rate on a per-sentence basis (see §5).

	MRR	Median rank
W2V	0.00007	111012
Sum	0.03686	861
N2V	**0.04907**	**623**

Table 1: Results on definitional dataset

	L2 ρ	L4 ρ	L6 ρ
W2V	0.1459	0.2457	0.2498
Sum	**0.3376**	0.3624	**0.4080**
N2V	0.3320	**0.3668**	0.3890

Table 2: Results on chimera dataset

5 Experiments

We first tune N2V's initial parameters on the training part of the definitional dataset. We experiment with a range of values for the learning rate ($[0.5, 0.8, 1, 2, 5, 10, 20]$), window size ($[5, 10, 15, 20]$), the number of negative samples ($[3, 5, 10]$), the number of epochs ($[1, 5]$) and the subsampling rate ($[500, 1000, 10000]$). Here, given the size of the data, the minimum frequency for a word to be considered is 1. The best performance is obtained for a window of 15 words, 3 negative samples, a learning rate of 1, a subsampling rate of 10000, an exponential decay where $\lambda = \frac{1}{70}$, and one single epoch (that is, the system truly implements fast-mapping). When applied to

[6] Code available at `https://github.com/minimalparts/nonce2vec`.

the test set, N2V shows a dramatic improvement in performance over the simple sum model, reaching $MMR = 0.04907$ (median rank 623).

On the training set of the Chimeras, we further tune the per-sentence decrease in window size and increase in subsampling. For the window size, we experiment with a reduction of $[1...6]$ words on either side of the target, not going under a window of ± 3 words. Further, we adjust each word's subsampling rate by a factor in the range $[1.1, 1.2...1.9, 2.0]$. Our results confirm that indeed, an appropriate change in those parameters is required: keeping them constant results in decreasing performance as more sentences are introduced. On the training set, we obtain our best performance (averaged over the 2-, 4- and 6-sentences datasets) for a per-sentence window size decrease of 5 words on either side of the target, and adjusting subsampling by a factor of 1.9. Table 2 shows results on the three corresponding test sets using those parameters. Unfortunately, on this dataset, N2V does not improve on addition.

The difference in performance between the definitional and the Chimeras datasets may be explained in two ways. First, the chimera sentences were randomly selected and thus, are not necessarily hugely informative about the nature of the nonce. Second, the most informative sentences are not necessarily at the beginning of the fragment, so the system heightens its learning rate on the wrong data: the risk does not pay off. This suggests that a truly intelligent system should adjust its parameters in a non-monotonic way, to take into account the quality of the information it is processing. This point seems to be an important general requirement for any architecture that claims incrementality: our results indicate very strongly that a notion of *informativeness* must play a role in the learning decisions of the system. This conclusion is in line with work in other domains, e.g. interactive word learning using dialogue, where performance is linked to the ability of the system to measure its own confidence in particular pieces of knowledge and ask questions with a high information gain (Yu et al., 2016). It also meets with general considerations on language acquisition, which accounts for the ability of young children to learn from limited 'primary linguistic data' by restricting explanatory models to those that provide such efficiency (Clark and Lappin, 2010).

6 Conclusion

We have proposed Nonce2Vec, a Word2Vec-inspired architecture to learn new words from tiny data. It requires a high-risk strategy combining heightened learning rate and greedy processing of the context. The particularly good performance of the system on definitions makes us confident that it is possible to build a unique, unified algorithm for learning word meaning from any amount of data. However, the less impressive performance on naturally-occurring sentences indicates that an ideal system should modulate its learning as a function of the informativeness of a context sentence, that is, take risks 'at the right time'.

As pointed out in the introduction, Nonce2Vec is designed with a view to be an essential component of an incremental concept learning architecture. In order to validate our system as a suitable, generic solution for word learning, we will have to test it on various data sizes, from the type of low- to middle-frequency terms found in e.g. the Rare Words dataset (Luong et al., 2013), to highly frequent words. We would like to systematically evaluate, in particular, how fast the system can gain an understanding of a concept which is fully equivalent to a vector built from big data. We believe that both quality and speed of learning will be strongly influenced by the ability of the algorithm to detect what we called *informative* sentences. Our future work will thus investigate how to capture and measure informativeness.

Acknowledgments

We are grateful to Katrin Erk for inspiring conversations about tiny data and fast-mapping, and to Raffaella Bernardi and Sandro Pezzelle for comments on an early draft of this paper. We also thank the anonymous reviewers for their time and valuable comments. We acknowledge ERC 2011 Starting Independent Research Grant No 283554 (COMPOSES). This project has also received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 751250.

References

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3:1137–1155.

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

Alexander Clark and Shalom Lappin. 2010. Computational learning theory and language acquisition. In Ruth M Kempson, Tim Fernando, and Nicholas Asher, editors, *Philosophy of linguistics*, pages 445–475. Elsevier.

Stephen Clark. 2012. Vector space models of lexical meaning. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantics – second edition*. Wiley-Blackwell.

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

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: a survey. *Language and Linguistics Compass*, 6:635–653.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL2012)*, pages 873–882.

Max Kisselew, Sebastian Padó, Alexis Palmer, and Jan Šnajder. 2015. Obtaining a Better Understanding of Distributional Models of German Derivational Morphology. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS2015)*, pages 58–63, London, UK.

Brenden M Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B Tenenbaum. 2011. One-shot learning of simple visual concepts. In *Proceedings of the 33rd Annual Meeting of the Cognitive Science Society (CogSci2012)*, Boston, MA.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2016. Building machines that learn and think like people. *arxiv*, abs/1604.00289.

Angeliki Lazaridou, Marco Marelli, and Marco Baroni. 2017. Multimodal word meaning induction from minimal exposure to natural text. *Cognitive Science*, 41(S4):677–705.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly Derived Representations of Morphologically Complex Words in Distributional Semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL2013)*, pages 1517–1526, Sofia, Bulgaria.

Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the 17th Conference on Computational Natural Language Learning (CoNLL2013)*, pages 104–113, Sofia, Bulgaria.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Sebastian Padó, Aurélie Herbelot, Max Kisselew, and Jan Šnajder. 2016. Predictability of distributional semantics in derivational word formation. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING2016)*, Osaka, Japan.

Behrang QasemiZadeh, Laura Kallmeyer, and Aurélie Herbelot. 2017. Non-Negative Randomized Word Embeddings. In *Proceedings of Traitement automatique des langues naturelles (TALN2017)*, Orléans, France.

John C Trueswell, Tamara Nicol Medina, Alon Hafri, and Lila R Gleitman. 2013. Propose but verify: Fast mapping meets cross-situational word learning. *Cognitive psychology*, 66(1):126–156.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Yanchao Yu, Arash Eshghi, and Oliver Lemon. 2016. Training an adaptive dialogue policy for interactive learning of visually grounded word meanings. In *Proceedings of the 17th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL2016)*, pages 339–349, Los Angeles,CA.

Word Embeddings based on Fixed-Size Ordinally Forgetting Encoding

Joseph Sanu[†], **Mingbin Xu**[†], **Hui Jiang**[†] and **Quan Liu**[‡]
[†]Department of Electrical Engineering and Computer Science,
York University, 4700 Keele Street, Toronto, Ontario, M3J 1P3, Canada
[‡]Department of EEIS, University of Science and Technology of China, Hefei, China
cse83186@cse.yorku.ca, xmb@cse.yorku.ca, hj@cse.yorku.ca, quanliu@mail.ustc.edu.cn

Abstract

In this paper, we propose to learn word embeddings based on the recent fixed-size ordinally forgetting encoding (FOFE) method, which can almost uniquely encode any variable-length sequence into a fixed-size representation. We use FOFE to fully encode the left and right context of each word in a corpus to construct a novel word-context matrix, which is further weighted and factorized using truncated SVD to generate low-dimension word embedding vectors. We have evaluated this alternative method in encoding word-context statistics and show the new FOFE method has a notable effect on the resulting word embeddings. Experimental results on several popular word similarity tasks have demonstrated that the proposed method outperforms many recently popular neural prediction methods as well as the conventional SVD models that use canonical count based techniques to generate word context matrices.

1 Introduction

Low dimensional vectors as word representations are very popular in NLP tasks such as inferring semantic similarity and relatedness. Most of these representations are based on either matrix factorization or context sampling described by (Baroni et al., 2014) as count or predict models. The basis for both models is the distributional hypothesis (Harris, 1954), which states that words that appear in similar contexts have similar meaning. Traditional context representations have been obtained by capturing co-occurrences of words from a fixed-size window relative to the focus word. This representation however does not encompass the entirety of the context surrounding the focus word. Therefore, the distributional hypothesis is not being taken advantage of to the fullest extent. In this work, we seek to capture these contexts through the fixed-size ordinally forgetting encoding (FOFE) method, recently proposed in (Zhang et al., 2015b). In addition to just capturing word co-occurrences, we attempt to use the FOFE to encode the full contexts of each focus word, including the order information of the context sequences. We believe the full encoding of contexts can enhance the resulting word embedding vectors, derived by factoring the corresponding word-context matrix. As argued in (Zhang et al., 2015b), the FOFE method can almost uniquely encode discrete sequences of varying lengths into a fixed-size code, and this encoding method was used to address the challenges of a limited size window when using deep neural networks for language modeling. The resulting algorithm fulfills the needs of keeping long term dependency while being fast. The word order in a sequence is modeled by FOFE using an ordinally-forgetting mechanism which encodes each position of every word in the sequence.

In this paper, we elaborate how to use the FOFE to fully encode context information of each focus word in text corpora, and present a new method to construct the word-context matrix for word embedding, which may be weighted and factorized as in traditional vector space models (Turney and Pantel, 2010). Next, we report our experimental results on several popular word similarity tasks, which demonstrate that the proposed FOFE-based approach leads to significantly better performance in these tasks, comparing with the conventional vector space models as well as the popular neural prediction methods, such as *word2vec*, *GloVe* and more recent *Swivel*. Finally, this paper will conclude with the analysis and prospects of com-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 310–315
Copenhagen, Denmark, September 7–11, 2017.

bining this approach with other methods.

2 Related Work

There has been some debate as to what the optimal length of a text should be for measuring word similarity. Word occurrences from a fixed context window of words can be used to represent a context (Lund and Burgess, 1996). The word co-occurrence frequencies are based on fixed windows spanning in both directions from the focus word. This is then used to create a word-context matrix from which row vectors can be used to measure word similarity. A weighting step is usually applied to highlight words with close association in the co-occurrence matrix, and the truncated SVD is used to factorize the weighted matrix to generate low-dimension word vectors. Recently, (Mikolov et al., 2013a) has introduced an alternative way to generate word embeddings using the skipgram model trained with stochastic gradient descent and negative sampling, named as SGNS. SGNS tries to maximize the dot product between $w \cdot c$ where both a word w and a context c are obtained from observed word-context pairs, and meanwhile it also tries to minimize the dot product between $w \cdot c'$ where c' is a negative sample representing some contexts that are not observed in the corpus. More recently, (Levy and Goldberg, 2014) has showed that the objective function of SGNS is essentially seeking to minimize the difference between the models estimate and the log of co-occurrence count. Their finding has shown that the optimal solution is a weighted factorization of a pointwise mutual information matrix shifted by the log of the number of negative samples.

SGNS and GloVe (Pennington et al., 2014) select a fixed window of usually 5 words or less around a focus word to encode its context and the word order information within the window is completely ignored. Other attempts to fully capture the contexts have been successful with the use of recurrent neural networks (RNNs) but these methods are much more expensive to run over large corpora when comparing with the proposed FOFE method in this paper. Some previous approaches to encode order information, such as such as BEAGLE (Jones and Mewhort, 2007) and Random Permutations (Sahlgren et al., 2008), typically require the use of expensive operations such as convolution and permutation to process all n-grams within a context window to memorize order information for a given word. On the contrary, the FOFE methods only use a simple recursion to process a sentence once to memorize both context and order information for all words in the sentence.

3 FOFE based Embedding

To capture the full essence of the distributional hypothesis, we need to fully encode the left and right context of each focus word in the text, and further take into accounts that words closer to the focus word should play a bigger role in representing the context relevant to the focus word than other words locating much farther away. Traditional co-occurrence word-context matrixes fail to address these concerns of context representation.

In this work, we propose to make use of the fixed-size ordinally-forgetting encoding (FOFE) method, proposed in (Zhang et al., 2015b) as a unique encoding method for any variable-length sequence of discrete words.

Given a vocabulary of size K, FOFE uses 1-of-K one-hot representation to represent each word. To encode any variable-length sequence of words, FOFE generates the code using a simple recursive formula from the first word (w_1) to the last one (w_T) of the sequence: (assume $z_0 = 0$)

$$z_t = \alpha \cdot z_{t-1} + e_t \; (1 \leq t \leq T) \qquad (1)$$

where z_t denotes the FOFE code for the partial sequence up to word w_t, α is a constant forgetting factor, and e_t denotes the one-hot vector representation of word w_t. In this case, the code z_T may be viewed as a fixed-size representation of any sequence of $\{w_1, w_2, \cdots, w_T\}$. For example, assume we have three symbols in vocabulary, e.g., *A*, *B*, *C*, whose 1-of-K codes are $[1, 0, 0]$, $[0, 1, 0]$ and $[0, 0, 1]$ respectively. When calculating from left to right, the FOFE code for the sequence $\{ABC\}$ is $[\alpha^2, \alpha, 1]$, and that of $\{ABCBC\}$ is $[\alpha^4, \alpha + \alpha^3, 1 + \alpha^2]$.

The uniqueness of the FOFE code is made evident if the original sequence can be unequivocally recovered from the given FOFE code. According to (Zhang et al., 2015b), FOFE codes have some nice theoretical properties to ensure the uniqueness, as exemplified by the following two theorems [1]:

Theorem 1 *If the forgetting factor* α *satisfies* $0 < \alpha \leq 0.5$, FOFE *is unique for any* K *and* T.

[1] See (Zhang et al., 2015a) for the proof of these two theorems.

Theorem 2 *For* $0.5 < \alpha < 1$*, given any finite values of* K *and* T*,* FOFE *is almost unique everywhere for* $\alpha \in (0.5, 1.0)$*, except only a finite set of countable choices of* α*.*

Finally, for alpha values less than or equal to 0.5 and greater than 0, the FOFE is unique for any sequence. For alpha values greater than 0.5, the chance of collision is extremely low and the FOFE is unique in almost all cases. Too find more about the theoretical correctness of FOFE, please refer to (Zhang et al., 2015b). In other words, the FOFE codes can almost uniquely encode any sequences, serving as a fixed-size but theoretically lossless representation for any variable-length sequences.

In this work, we propose to use FOFE to encode the full context where each focus word appears in text. As shown in Figure 1, the left context of a focus word, i.e., *bank*, may be viewed as a sequence and encoded as a FOFE code L from the left to right while its right context is encoded as another FOFE code R from right to left. When a proper forgetter factor α is chosen, the two FOFE codes can almost fully represent the context of the focus word. If the focus word appears multiple times in text, a pair of FOFE codes $[L, R]$ is generated for each occurrence. Next, a mean vector is calculated for each word from all of its occurrences in text. Finally, as shown in Figure 1, we may line up these mean vectors (one word per row) to form a new word-context matrix, called the FOFE matrix here.

4 PMI-based Weighting and SVD-based Matrix Factorization

We further weight the above FOFE matrix using the standard positive pointwise mutual information (PMI) (Church and Hanks, 1990) which has been shown to be of benefit for regular word-context matrices (Pantel and Lin, 2002). PMI is used as a measure of association between a word and a context. PMI tries to compute the association probabilities based on co-occurrence frequencies. Positive pointwise mutual information is a commonly adopted approach where all negative values in the PMI matrix are replaced with zero. The PMI-based weighting function is critical here since it helps to highlight the more surprising events in original word-context matrix.

There are significant benefits in working with low-dimensional dense vectors, as noted by (Deerwester et al., 1990) with the use of truncated singular value decomposition (SVD). Here, we also use truncated SVD to factorize the above weighted FOFE matrix as the product of three dense matrices U, Σ, V^T, where U and V^T have orthonormal columns and Σ is a diagonal matrix consisting of singular values. If we select Σ to be of rank d, its diagonal values represent the top d singular values, and U_d can be used to represent all word embeddings with d dimensions where each row represents a word vector.

5 Experiments

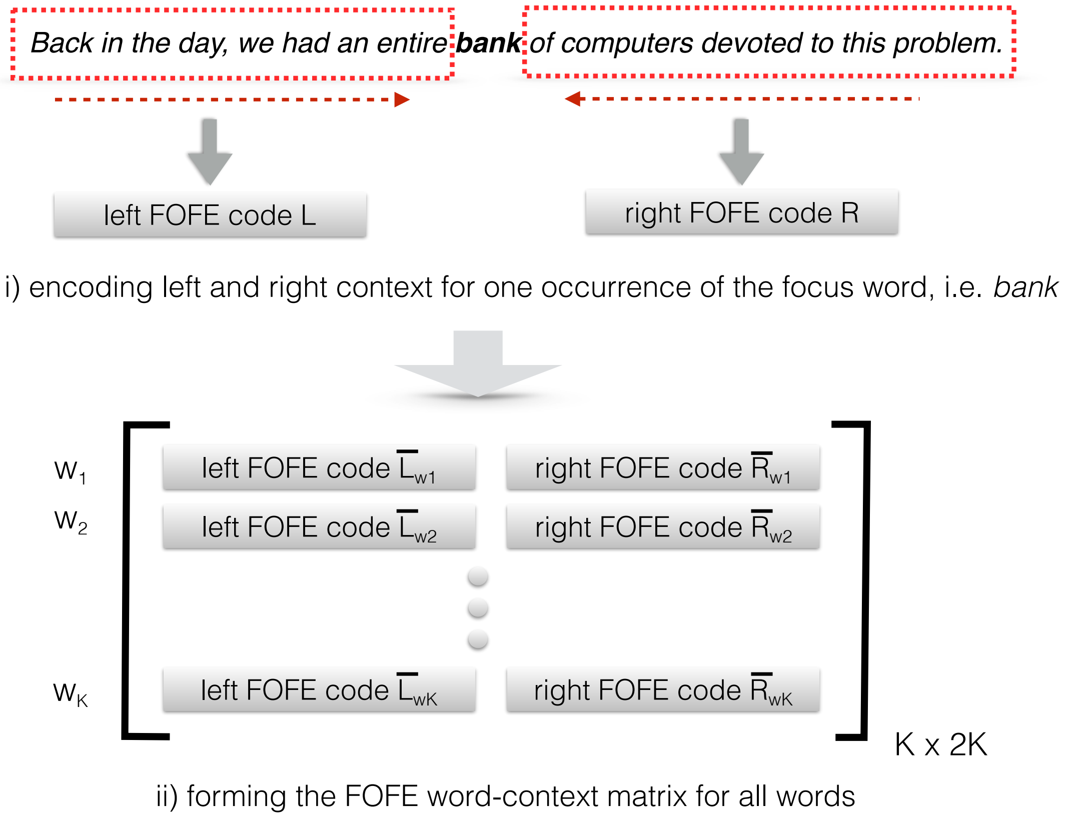
We conducted experiments on several popular word similarity data sets and compare our FOFE method with other existing word embedding models in these tasks. In this work, we opt to use five data sets: *WordSim353* (Finkelstein et al., 2001), *MEN* (Bruni et al., 2012), *Mechanical Turk* (Radinsky et al., 2011), *Rare Words* (Luong et al., 2013) and *SimLex-999* (Hill et al., 2015). The word similarity performance is evaluated based on the Spearman rank correlation coefficient obtained by comparing cosine distance between word vectors and human assigned similarity scores.

For our training data, we use the standard *enwiki9* corpus which contains 130 million words. The pre-processing stage includes discarding extremely long sentences, tokenizing, lowercasing and splitting each sentence as a context. Our vocabulary size is chosen to be 80,000 for the most frequent words in the corpus. All words not in the vocabulary are replaced with the token <*unk*>. In this work, we use a python-based library, called *scipy* [2], to perform truncated SVD to factorize all word-context matrices.

5.1 Experimental Setup

Our first baseline is the conventional vector space model (VSM) (Turney and Pantel, 2010), relying on the PMI-weighted co-occurrence matrix with dimensionality reduction performed using truncated SVD. The dimension of word vectors is chosen to be 300 and this number is kept the same for all models examined in this paper. Our main goal is to outperform VSM as the model proposed in this paper also uses SVD based matrix factorization. This allows for appropriate comparisons between the different word encoding methods.

[2] See `http://docs.scipy.org/doc/scipy/reference/`.


Back in the day, we had an entire bank of computers devoted to this problem.
left FOFE code L
right FOFE code R
w1
w2
wK
left FOFE code $\bar{L}_{w1}$
right FOFE code $\bar{R}_{w1}$
left FOFE code $\bar{L}_{w2}$
right FOFE code $\bar{R}_{w2}$
left FOFE code $\bar{L}_{wK}$
right FOFE code $\bar{R}_{wK}$
K x 2K

i) encoding left and right context for one occurrence of the focus word, i.e. *bank*

ii) forming the FOFE word-context matrix for all words

Figure 1: i) encoding left and right contexts of each focus word with FOFE and ii) forming the FOFE word-context matrix.

For the purpose of completeness, the other non-SVD based embedding models, mainly the more recent neural prediction methods, are also compared in our experiments. As a result, we build the second baseline using the skip-gram model provided by the `word2vec` software package (Mikolov et al., 2013a), denoted as SGNS. The word embeddings are generated using the recommended hyper-parameters from (Levy et al., 2015). Their findings show a larger number of negative samples is preferable and increments on the window size have minimal improvements on word similarity tasks. In our experiments the number of negative samples is set to 5 and the window size is set to 5. In addition, we set the subsampling rate to 10^{-4} and run 3 iterations for training. In adition to SGNS, we also obtained results for CBOW, GloVe (Pennington et al., 2014) and Swivel (Shazeer et al., 2016) models using similar recommended settings. While the window size has a fixed limit in the baseline models, our model does not have a window size parameter as the entire sentence is fully captured as well as distinctions between left and right contexts when generating the FOFE codes. The impact of closer context words is further highlighted by the use of the forgetting factor which is unique to the FOFE based word embedding.

Finally, we use the FOFE codes to construct the word-context matrix and generate word embedding as described in sections 3 and 4. Throughout our experiments, we have chosen to use a constant forgetting factor $\alpha = 0.7$. There is no significant difference in word similarity scores after experimenting with different α values between $[0.6, 0.9]$ when generating FOFE codes.

We have applied the same hyperparameters to both VSM and FOFE methods and fine-tune them based on the recommended settings provided in (Levy et al., 2015). Although it has been previously reported that context distribution smoothing (Mikolov et al., 2013b) can provide a net positive effect, it did not yield significant gains in our experiments. On the other hand, the eigenvalue

Table 1: The best achieved performance of various word embedding models on all five examined word similarity tasks.

Method	WordSim353	MEN	Mech Turk	Rare Words	SimLex-999
VSM+SVD	0.7109	0.7130	0.6258	0.4813	**0.3866**
CBOW	0.6763	0.6768	0.6621	0.4280	0.3549
GloVe	0.5873	0.6350	0.5831	0.3934	0.2883
SGNS	0.7028	0.6689	0.6187	0.4360	0.3709
Swivel	0.7303	0.7246	**0.7024**	0.4430	0.3323
FOFE+SVD	**0.7580**	**0.7637**	0.6525	**0.5002**	**0.3866**

weighting parameter tuning (Caron, 2001) proved to be incredibly effective for some datasets but ineffectual in others. The net benefit however is palpable and we include it for both VSM and FOFE methods.

5.2 Results and Discussion

The best results of all word embedding models are summarized in Table 1 for all five examined data sets, which include the the traditional count based VSM with SVD alongside SGNS using `word2vec` and our proposed FOFE word embeddings. The most discernible piece of information from the table is that the FOFE method significantly outperforms the traditional count based VSM method on most of these word similarity tasks. The results in Table 1 show that substantial gains are obtained by FOFE in *WordSim353*, *MEN* and *Rare Words* data sets. The *MEN* dataset shows a 7% relative improvement over the conventional VSM.

Among all of these five data sets, the proposed FOFE word embedding significantly outperforms VSM in four tasks while yielding similar performance as VSM in the last data set, i.e. *SimLex-999*. FOFE also outperforms all the other models except Swivel in the *Mech Turk* dataset. It is important to note that this paper does not state that SVD is obligatory to obtain the best model. The FOFE method can be complemented with other models such as Swivel in place of count based encoding methods. It is also theoretically guaranteed that the original sentence is perfectly recoverable from this FOFE code. This theoretical guarantee is clearly missing in previous methods to encode word order information, such as both BEAGLE and Random Permutations. It is evident that overall the FOFE encoding method does achieve significant gains in performance in these word similarity tests over the traditional VSM method that applies the same factorization method. This is substantial as (Levy et al., 2015) demonstrates that larger window sizes when using SVD does not payoff and the optimal context window is 2. We establish that we can indeed encode more information into our embedding with the FOFE codes.

In summary, our experimental results show great promise in using the FOFE encoding to represent word contexts for traditional matrix factorization methods. As for future work, the FOFE encoding method may be combined with other popular algorithms, such as Swivel, to replace the co-occurrence statistics based on a fixed window size.

6 Conclusion

The ability to capture the full context without restriction can play a crucial factor in generating superior word embeddings that excel in NLP tasks. The fixed-size ordinally forgetting encoding (FOFE) has the ability to seize large contexts while discriminating contexts that are farther away as being less significant. Conventional embeddings are derived from ambiguous co-occurrence statistics that fail to adequately discriminate contexts words even within the fixed-size window. The FOFE encoding technique trumps other approaches in its ability to procure the state of the art results in several word similarity tasks when combined with prominent factorization practices.

Acknowledgments

This work is partially supported by a Discovery Grant from Natural Sciences and Engineering Research Council (NSERC) of Canada, and a research donation from iFLYTEK Co., Hefei, China.

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs.

context-predicting semantic vectors. In *ACL (1)*, pages 238–247.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 136–145. Association for Computational Linguistics.

John Caron. 2001. Experiments with lsa scoring: Optimal rank and basis. In *Proceedings of the SIAM Computational Information Retrieval Workshop*, pages 157–169.

Kenneth W. Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

M.N Jones and D.J.K Mewhort. 2007. Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114(1):1–37.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185.

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

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical cooccurrence. *Behavior Research Methods, Instruments, and Computers*, 28(2):203–208.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113. Citeseer.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

Magnus Sahlgren, Anders Holst, and Kanerva Pentti. 2008. Permutations as a means to encode order in word space. In *In Proceedings of the 30th Annual Conference of the Cognitive Science Society*, pages 1300–1305.

Noam Shazeer, Ryan Doherty, Colin Evans, and Chris Waterson. 2016. Swivel: Improving embeddings by noticing whats missing. *arXiv preprint arXiv:1602.02215*.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015a. A fixed-size encoding method for variable-length sequences with its application to neural network language models. *arXiv preprint arXiv:1505.01504*.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015b. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 495–500, Beijing, China. Association for Computational Linguistics.

VecShare: A Framework for Sharing Word Representation Vectors

Jared Fernandez and **Zhaocheng Yu** and **Doug Downey**
Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208
{jared.fern|zhaochengyu2017}@u.northwestern.edu
ddowney@eecs.northwestern.edu

Abstract

Many Natural Language Processing (NLP) models rely on distributed vector representations of words. Because the process of training word vectors can require large amounts of data and computation, NLP researchers and practitioners often utilize pre-trained embeddings downloaded from the Web. However, finding the best embeddings for a given task is difficult, and can be computationally prohibitive. We present a framework, called VecShare, that makes it easy to share and retrieve word embeddings on the Web. The framework leverages a public data-sharing infrastructure to host embedding sets, and provides automated mechanisms for retrieving the embeddings most similar to a given corpus. We perform an experimental evaluation of VecShare's similarity strategies, and show that they are effective at efficiently retrieving embeddings that boost accuracy in a document classification task. Finally, we provide an open-source Python library for using the VecShare framework.[1]

1 Introduction

Word embeddings capture syntactic and semantic properties of words, and are a key component of many modern NLP models (Turian et al., 2010). However, high-quality embeddings can be expensive to train. As a result, rather than training their own embeddings, NLP researchers and practitioners often download pre-trained embeddings from the Web, e.g. (Limsopatham and Collier, 2016; Cheng et al., 2016).

However, existing methods for sharing embeddings on the Web are suboptimal. Current practice primarily consists of contributors posting embedding sets to their own Web sites. No central embedding repository exists, and it is difficult for users to know which embedding sets are available. Furthermore, determining the utility of an embedding set for a particular NLP task entails significant time and computational costs, as users must manually download and evaluate multiple complete embedding sets. Methods exist for automatically scoring an embedding set, but these are limited to specific tasks and lack integration with existing code bases (Faruqui and Dyer, 2014).

Our goal in this paper is to introduce a framework that makes sharing word embeddings easier for NLP researchers and practitioners. It should be simple and fast to post, browse, and retrieve embeddings from a public data store. Additionally, integration with existing NLP codebases should be more seamless: software libraries should automatically identify and download the particular embeddings that are likely to be relevant to a user's corpus.

This paper presents VecShare, a framework for sharing word embeddings. As its data store, it uses an existing public data-sharing platform, which provides searching and browsing capability. To solve the critical challenge of helping users quickly find relevant embeddings, we introduce embedding *indexers*. The indexers compute and share compact representations, called *signatures*, for each embedding set. Users employ a software library that downloads the signatures and compares them against the user's corpus. Using the signatures, the library efficiently evaluates the utility of each shared embedding set and determines which sets are most likely to be relevant. The library can then automatically download the relevant embeddings and make them available

[1] https://github.com/JaredFern/VecShare

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 316–320
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

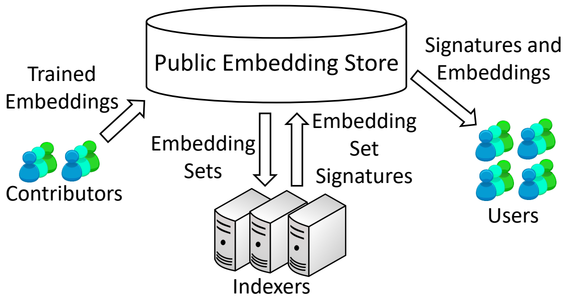


Figure 1: The VecShare Framework.

within the user's code. Finally, VecShare is an open source framework: new embeddings, indexers, and signature methods can be independently added at any time.

We perform experiments evaluating different signature methods in selecting embeddings for document classification tasks, and demonstrate that an ensemble signature based on simple features (e.g. vocabulary overlap with the user's corpus, or similarities between a sample of embedding pairs) can select helpful embeddings. We release a Python library for NLP researchers and practitioners that can query the embedding library, and automatically select and download relevant embeddings for a given corpus.

2 The VecShare Framework

The VecShare framework is illustrated in Figure 1. Contributors upload embedding sets to a *Public Embedding Store*, hosted on a public data-sharing platform. *Indexers* periodically poll the embedding store, detecting and indexing newly uploaded embedding sets. For each embedding set, indexers store a *signature* that compactly represents the content of the embedding set. The indexer uses these signatures to return relevant embedding sets to users. A *user* can then retrieve embeddings for their particular corpus from the data store, using a software library built for this purpose.

The current VecShare implementation uses the public data sharing website data.world as its embedding store.[2] We chose data.world for its ease of use and robustness, but any publicly-available data share that allows search by tags and programmatic access is sufficient to house VecShare. Below, we describe the details of the framework.

2.1 Contributors

A contributor who has computed a set of word embeddings adds the embeddings to VecShare by uploading the data to the share, following a simple standard format with $n + 1$ fields where the first field is a word, and the remaining fields give the n-dimensional embedding of the word. The contributor tags the data set with a designated tag so that indexers can automatically identify that the data set is an indexable word embedding set. For embedding sets to be applicable to certain kinds of signatures, contributors can also elect to upload additional metadata with their embeddings (in our initial implementation, metadata includes a frequency ranking of terms in the corpus, and the total number of tokens used to construct the embeddings).

2.2 Indexers and Signatures

Indexers periodically poll the embedding store, looking for new embedding sets uploaded by contributors. For each embedding set, the indexer computes and stores a signature designed to capture characteristics of the embeddings. To estimate the relevance of each embedding set for the user's task, these signatures are later compared to corresponding signatures created from the user's corpus. Thus, each signature method has an associated *similarity measure*, which takes in a pair of signatures (one from a VecShare embedding set, and the other from the user's corpus) and outputs a numeric similarity score for the pair.

We explore two primary signature methods in this paper. The first, *VocabRk*, consists of (up to) the T_v most frequent words in the embedding corpus, excluding a set of stop words. The similarity method is the negative average rank of the signature words within the user's frequency-ordered vocabulary. Words not in the user's corpus are assigned a rank of T. Thus, the most similar embedding set under the *VocabRk* signature is thus the one with the lowest average rank.

The *VocabRk* signature method relies only on vocabulary overlap, and entirely ignores the embeddings themselves. We also experiment with a second signature method that does utilize the embeddings. The *SimCorr* signature for a set of embeddings E consists of the embeddings for the T_s most frequent words in E's corpus (again excluding stop words). To estimate the similarity between the user's corpus C and the embeddings E,

[2] http://data.world/

the *SimCorr* method first computes a set of embeddings on the user's corpus. For all words found in both E's signature and in the user's corpus, *SimCorr* computes all pairwise cosine similarities between pairs of E embeddings, and pairs of C embeddings. The Pearson correlation coefficient between the E-based cosine similarities and the C-based ones is taken as the *SimCorr* similarity measure. Thus, two embedding sets that estimate similarity of terms in a similar manner will be deemed similar according the *SimCorr* measure, even if the vector values of the embeddings differ substantially between the two sets.

The indexers make their signatures available to users by simply sharing a signature data set on the public data store, which is read by the VecShare software library. Like the other components of VecShare, the indexers of the framework are extensible – new indexers, signature and similarity measures can be created at any time.

2.3 Libraries

Users access the VecShare framework using a code library. Embeddings can be requested by name if the user desires a particular embedding (e.g. "300 dimensional Google News word2vec embeddings"), or the user can query an indexer to find embedding sets most likely to be useful for the user's task. Importantly, the software library performs embedding selection locally on the user's machine, using signature similarity methods described previously. The user's corpus does not need to be uploaded or shared.

Once the target embedding set has been identified, the library downloads the target embeddings for the particular words in the user's vocabulary. Thus, if the user's vocabulary is much smaller than that of the embedding set, this download can be much more compact than the full embedding set.

A contribution of this paper is the release of a Python library for the framework, which implements the *VocabRk* similarity computation. With this library, leveraging the framework to select and retrieve the top-ranked embeddings for a given corpus requires just one line of code.

3 Experiments

We now evaluate the effectiveness of the signature methods described in the previous section at identifying high-quality embedding sets for a given corpus, for the task of text classification. We also quantify the improvement in efficiency when using VecShare rather than following the current practice of downloading and testing multiple embedding sets.

In our experiments, we set the parameters $T_v = 5,000$ and $T_s = 1,000$, and we discard the top 100 most frequent words as stopwords. When training embeddings on the user's corpus, we use word2vec.

3.1 Experimental Methodology

We perform experiments in two settings: first with *large-corpus embeddings*, where we use word2vec and GloVe embedding sets trained on billions of tokens. The large-corpus embeddings are representative of state-of-the-art models, but are trained over very broad-topic corpora (billions of tokens of newswire, Web or social media text). To better measure whether VecShare can harness more specific, targeted embedding sets, we also evaluate over *small-corpus* embeddings.

For the large-corpus embeddings, we utilize three sets of GloVe embeddings (Pennington et al., 2014): **wik+**, 100-dimensional embeddings trained on six billion tokens of Wikipedia and the Gigaword corpus; **web**, 300-dimensional embeddings trained on 42 billion tokens of the Common Crawl Web dataset; and **twtr**, 100-dimensional embeddings trained on 27 billion tokens of Twitter posts. We also utilize **gnws**, 300-dimensional word2vec embeddings trained on three billion tokens of Google News data.[3]

For the small corpus embeddings, we created a topically diverse collection of subsets of the New York Times corpus (Sandhaus, 2008), across seven categories (agriculture, arts, books, economics, government, movies, and weather). We then trained word2vec embeddings on each subset, to create seven distinct similarly-sized, small corpus embedding sets.

For our experiments, we utilize the embeddings as features for document classification within a convolutional neural network (Chollet, 2017). We evaluate on four document classification tasks: Reuters-21578 newswire topic classification (Lewis, 1997), subjectivity classification (Pang and Lee, 2004), IMDB movie review classification (Maas et al., 2011), and the 20news classification task.[4]

[3] https://github.com/mmihaltz/word2vec-GoogleNews-vectors.

[4] http://qwone.com/~jason/20Newsgroups/

	Reuters			Subjectivity			IMDB			20news			Average	
	ρ	Sel.	Acc.	ρ	Sel.	Acc.	ρ	Sel.	Acc.	ρ	Sel.	Acc.	ρ	Acc
Random	-	-	0.862	-	-	0.688	-	-	0.868	-	-	0.763	-	0.795
MaxTkn	**0.63**	web	**0.888**	0.19	web	0.728	0.38	web	0.881	**0.97**	web	**0.863**	**0.54**	0.840
VocabRk	0.46	gnws	0.882	0.02	gnws	**0.759**	0.40	gnws	**0.886**	0.20	gnws	0.719	0.27	0.812
SimCorr	-0.65	wik+	0.84	**0.81**	gnws	**0.759**	0.45	gnws	**0.886**	0.60	twtr	0.748	0.30	0.808
All	0.26	gnws	0.882	0.43	gnws	**0.759**	**0.49**	gnws	**0.886**	0.87	web	**0.863**	0.51	**0.848**
Oracle	-	web	0.888	-	gnws	0.759	-	gnws	0.886	-	web	0.863	-	0.85

Table 1: Experimental results using large-corpus embeddings. All of the signature methods outperform the random baseline, and the *All* method performs best in terms of both correlation ρ and text classification accuracy.

	Reuters			Subjectivity			IMDB			20news			Average	
	ρ	Sel.	Acc.	ρ	Sel.	Acc.	ρ	Sel.	Acc.	ρ	Sel.	Acc.	ρ	Acc
Random	-	-	0.844	-	-	0.667	-	-	0.829	-	-	0.610	-	0.738
MaxTkn	0.62	govt	0.856	-0.64	govt	0.568	-0.02	govt	0.763	**0.82**	govt	**0.647**	0.20	0.709
VocabRk	0.74	econ	**0.880**	0.51	mov	0.686	0.89	mov	0.835	0.64	book	0.629	**0.70**	0.758
SimCorr	0.51	econ	**0.880**	**0.62**	book	**0.706**	**0.93**	book	0.842	-0.25	agri	0.551	0.45	0.745
All	**0.82**	econ	**0.880**	0.16	book	**0.706**	0.87	book	**0.842**	0.67	book	0.629	0.63	**0.764**
Oracle	-	econ	0.880	-	book	0.706	-	book	0.842	-	govt	0.647	-	0.769

Table 2: Experimental results using small-corpus embeddings. The *VocabRk* and *SimCorr* methods outperform the baselines, and the *All* method performs best in terms of both correlation ρ and text classification accuracy.

In addition to the *VocabRk* and *SimCorr* methods described in the previous section, we also evaluate against two simple baselines: *Random*, which selects an embedding set at random, and *MaxTkn*, which adopts a "bigger is better" strategy, always selecting the embedding set trained over the largest text corpus.

Finally, as discussed below our experimental results reveal that the different signature methods have distinct strengths. Thus, we also evaluate *All*, a simple ensemble of the *VocabRk, SimCorr*, and *MaxTkn* methods. *All* simply takes an even average of the rankings output by its three constituent signature methods. The *All* method selects the single embedding set with the lowest average ranking, breaking ties in favor of the *VocabRk* method.

3.2 Results

Our results are shown in Tables 1 and 2. For each classification task and each method, "Sel." indicates the embedding set that the method selects (that is, the one the method ranks most similar to the text classification data set). We report two measures of the quality of a signature method: ρ, the Pearson correlation between the similarity scores assigned by the method and the set's accuracy on the classification task; and "Acc.," the accuracy of the embeddings selected by the method.

The results show that for the small-corpus embeddings, the *VocabRk* and *SimCorr* signature methods perform well, beating the random baseline overall. By contrast, for the large-corpus embeddings, the *MaxTkn* method performs the best of the individual methods (primarily due to its strong performance on the idiosyncratic 20news data set). The *All* ensemble method achieves performance nearly as high as the best possible embedding selection (represented as the *Oracle* method in the tables).

An alternative to using pre-trained embeddings is to train embeddings on the evaluation corpus itself. We found that this approach achieved an average accuracy of 0.746 across our four data sets, lower than our results using the *All* method. Utilizing the pre-trained embeddings, especially those computed over large corpora, provides a significant boost in text classification accuracy.

3.3 Efficiency Experiment

We also evaluated the relative gain in time and space efficiency that VecShare provides over the current practice of manually evaluating each embedding set and selecting the embedding that performs best. The efficiency experiment was performed on a single machine with a 2.3 GHz quad-core CPU and 8GB of main memory, using a test framework containing 11 embedding sets.

Embedding selection was performed using both the *VocabRk* signature method on VecShare and the conventional method of selecting embeddings, which trains models for each embedding set and then evaluates those models on the test corpus. The conventional approach required an average of 177 minutes to train, evaluate, and select an embedding set for each test corpus. Whereas, the *VocabRk* signature method on the VecShare framework required an average of 38 seconds to select an embedding for each test corpus, an average speedup of 280x. Additionally, VecShare substantially reduces space cost: the total size of the signatures in the experiments is 4-5 orders of magnitude smaller than the full embedding sets.

4 Conclusions and Future Work

We presented VecShare, a framework for sharing word vector representations. The VecShare framework uses signatures to help researchers and practitioners quickly identify helpful embeddings for their task. We released a Python library that allows practitioners to access the framework. We also performed experiments quantifying the accuracy and efficiency of VecShare's embedding selection approach on text classification. Further experiments on additional data sets and NLP tasks are necessary.

In future work, we wish to explore embedding signatures that leverage richer knowledge of the practitioner's corpus and task. Finally, we hope to extend VecShare's embedding selection methods to consider syntheses of multiple distinct embedding sets, tailored to the practitioner's task and corpus.

Acknowledgments

This research was supported in part by NSF grant IIS-1351029 and the Allen Institute for Artificial Intelligence. We thank Mohammed Alam, Dave Demeter, Thanapon Noraset, and Zheng Yuan for helpful comments.

References

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *EMNLP*.

Franois Chollet. 2017. keras. `https://github.com/fchollet/keras`.

Manaal Faruqui and Chris Dyer. 2014. Community evaluation and exchange of word vectors at wordvectors.org. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, USA.

David D Lewis. 1997. Reuters-21578 text categorization test collection, distribution 1.0. *http://www. research. att. com/~ lewis/reuters21578. html* .

Nut Limsopatham and Nigel Collier. 2016. Modelling the combination of generic and target domain embeddings in a convolutional neural network for sentence classification. Association for Computational Linguistics.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 142–150.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 271.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia* 6(12):e26752.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics, pages 384–394.

Word Re-Embedding via Manifold Dimensionality Retention

Souleiman Hasan and **Edward Curry**
Lero- The Irish Software Research Centre
National University of Ireland, Galway
{souleiman.hasan, edward.curry}@lero.ie

Abstract

Word embeddings seek to recover a Euclidean metric space by mapping words into vectors, starting from words co-occurrences in a corpus. Word embeddings may underestimate the similarity between nearby words, and overestimate it between distant words in the Euclidean metric space. In this paper, we re-embed pre-trained word embeddings with a stage of manifold learning which retains dimensionality. We show that this approach is theoretically founded in the metric recovery paradigm, and empirically show that it can improve on state-of-the-art embeddings in word similarity tasks $0.5 - 5.0\%$ points depending on the original space.

1 Introduction

Concepts have been hypothesized in the cognitive psychometric literature as points in a Euclidean metric space, with empirical support from human judgement experiments (Rumelhart and Abrahamson, 1973; Sternberg and Gardner, 1983). Word embeddings, such as GloVe (Pennington et al., 2014a) and Word2Vec (Mikolov et al., 2013), harvest observed features of the latent Euclidean space such as words co-occurrence counts in a corpus and turn words into dense vectors of a few hundred dimensions. Word embeddings have proved useful in downstream NLP tasks such as Part of Speech Tagging (Collobert, 2011), Named Entity Recognition (Turian et al., 2010), and Machine Translation (Devlin et al., 2014). However, the potential of word embeddings and further improvements remain a research question.

When comparing word pairs similarities obtained from word embeddings, to word pairs similarities obtained from human judgement, it is observed that word embeddings slightly underestimate the similarity between similar words, and overestimate the similarity between distant words. For example, in the WS353 (Finkelstein et al., 2001) word similarity ground truth:

$$\begin{aligned} sim(\text{“}shore\text{”}, \text{“}woodland\text{”}) &= 3.08 \\ < sim(\text{“}physics\text{”}, \text{“}proton\text{”}) &= 8.12 \end{aligned}$$

However, the use of GloVe 42B 300d embedding with cosine similarity (see Section 4) yields the opposite order:

$$\begin{aligned} sim(\text{“}shore\text{”}, \text{“}woodland\text{”}) &= 0.36 \\ > sim(\text{“}physics\text{”}, \text{“}proton\text{”}) &= 0.33 \end{aligned}$$

Re-embedding the space using a manifold learning stage can rectify this. Manifold learning works by estimating the distance between nearby words using direct similarity assignment in a local neighbourhood, while distance between faraway words is approximated by multiple neighbourhoods based on the manifold shape. This observation forms the basis for the rest of this paper.

For instance, using Locally Linear Embedding (LLE) (Roweis and Saul, 2000) on top of GloVe, as described in this paper, can recover the right pairs order yielding:

$$\begin{aligned} sim(\text{“}shore\text{”}, \text{“}woodland\text{”}) &= 0.08 \\ < sim(\text{“}physics\text{”}, \text{“}proton\text{”}) &= 0.25 \end{aligned}$$

Hashimoto et al. (Hashimoto et al., 2016) put word embeddings under a paradigm which seeks to recover the underlying Euclidean metric semantic space. In this paradigm, word embeddings land into a space where a Euclidean metric can be used. They show that co-occurrence counts are the results of random walk sequences in the metric space, corresponding to sentences in a corpus.

Hashimoto et al. link this to manifold learning which also seeks to recover a Euclidean space

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 321–326
Copenhagen, Denmark, September 7–11, 2017.

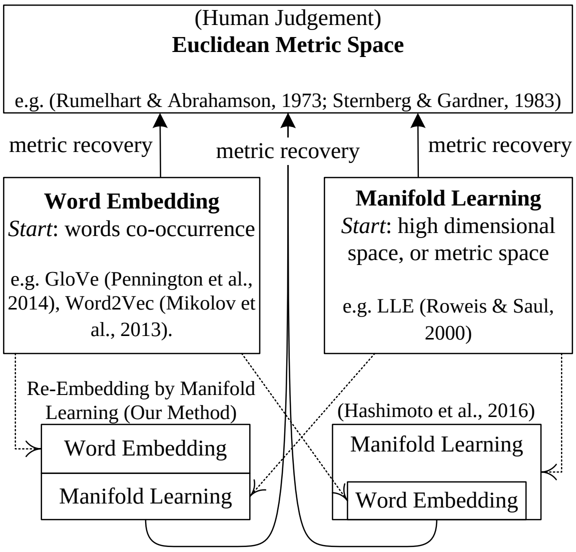

(Human Judgement)
Euclidean Metric Space
e.g. (Rumelhart & Abrahamson, 1973; Sternberg & Gardner, 1983)
metric recovery
metric recovery
metric recovery
Word Embedding
Start: words co-occurrence
e.g. GloVe (Pennington et al., 2014), Word2Vec (Mikolov et al., 2013).
Manifold Learning
Start: high dimensional space, or metric space
e.g. LLE (Roweis & Saul, 2000)
Re-Embedding by Manifold Learning (Our Method)
Word Embedding
Manifold Learning
(Hashimoto et al., 2016)
Manifold Learning
Word Embedding

Figure 1: Methodology and Related Work.

but starting from local neighbourhoods of objects, such as images or words. Global distances are built by adding up small local neighbourhoods. The authors show that word embedding algorithms can be used to solve manifold learning by generating random walks, aka sentences, on the manifold neighbourhood graph, and then embedding them.

In this work we follow a methodology which adheres to this paradigm and adopt a different angle, as per Figure 1. We start from an off-the-shelf word embedding, then we take a sample of it and feed it into manifold learning which leverages local word neighbourhoods formed in the original embedding space, learns the manifold, and embeds it into a new Euclidean space. The resulting re-embedding space is a recovery of a Euclidean metric space that is empirically better than the original word embedding when tested on word similarity tasks.

These results show that word embeddings can be improved in estimating the latent metric. Such an approach can provide new opportunities to improve our understanding of embedding methods, their properties, and limits. It also allows us to reuse and re-embed off-the-shelf pre-trained embeddings, saving time on training, while aiming at improved results in downstream NLP tasks, and other data processing tasks (Hasan and Curry, 2014; Hasan, 2017; Freitas and Curry, 2014).

Section 2 discusses the related literature to this work. Section 3 details the proposed approach. Sections 4 and 5 discuss the experiments and results. The paper concludes with Section 6.

2 Related Work

The relationship to related work is depicted in Figure 1. Word embeddings are unsupervised methods based on word co-occurrence counts which can be directly observed in a corpus. Mikolov et al. presents a neural network-based architecture which learns a word representation by learning to predict its context words (Mikolov et al., 2013). Pennington et al. proposed GloVe, which directly leverages nonzero word-word co-occurrences in a global manner (Pennington et al., 2014a).

The idea of embedding objects from a high dimensional space, e.g. images, into a smaller dimensional space constitute the area of manifold learning. For instance, Roweis and Saul present the Locally Linear Embedding (LLE) algorithm and show that pixel-based distance between images is meaningful only at a local neighbourhood scale (Roweis and Saul, 2000). Reconstructions can capture the underlying manifold of the data, and can embed the high dimensional objects, into a lower dimensional Euclidean space while preserving neighbourhoods. Other methods exist such as Isomap (Balasubramanian and Schwartz, 2002) and t-SNE (Maaten and Hinton, 2008).

Hashimoto et al. show that word embeddings and manifold learning are both methods to recover a Euclidean metric using co-occurrence counts and high dimensional features respectively (Hashimoto et al., 2016). They show that word embeddings can be used to solve manifold learning when starting from a high dimensional space. In this paper we start from a trained word embedding space, and learn a manifold from it to improve results. We do not use manifold learning to reduce dimensionality, but to transform between two equally-dimensional coordinate systems.

Other related work comes from word embedding post-processing. Labutov and Lipson use a supervised model to re-embed words for a target task (Labutov and Lipson, 2013). Lee et al. filter out abnormal dimensions from a GloVe space according to their histograms and show a slight improvement in performance (Lee et al., 2016). Mu at al. perform similar post-processing through the removal of the mean vector and vectors re-projection (Mu et al., 2017). We see manifold learning as a generic, unsupervised, nonlinear, and theoretically-founded model for post-processing that can cover linear post-processing such as PCA and normalization of vectors.

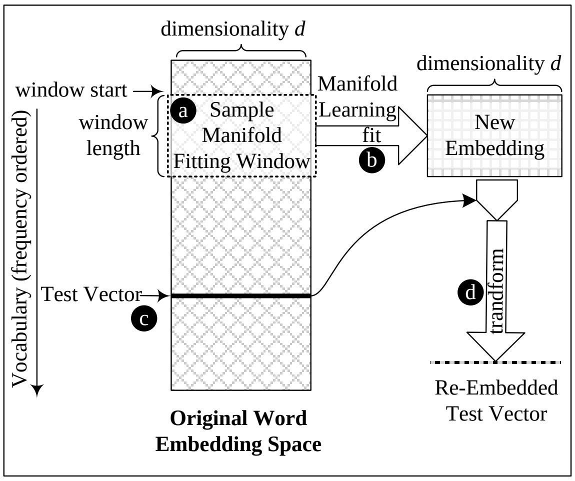


Figure 2: Re-Embedding via Manifold Learning.

3 Approach

Figure 2 illustrates our re-embedding method. We start from an original embedding space with vectors ordered by words frequencies. In step (a), we pick a sample window of vectors from this space to be used for learning the manifold. In step (b), we fit the manifold learning model to the selected sample using an algorithm such as LLE. We retain the dimensionality at this stage. In step (c), an arbitrary test vector can be selected from the original space. In step (d), the resulting fitted model serves as a transformation which can be used to transform the test vector into a vector which lives in the new re-embedding space, and used in downstream tasks.

In step (a), a sample subset of the words is used based on word frequency rank. The rational is that word embedding attempts to recover a metric space and frequent words co-occurrences can represent a better sampling of the underlying space due to their frequent usage, rather than being handled equally with other points, thus can better recover the manifold shape. Experimenting with subsets from all the vocabulary or non-frequent words, may yield no improvement. Additionally, manifold learning on all points is computationally expensive. The sampling used here follows a sliding sample window to study the effect of its start position and size. Various ways to choose a sample, e.g. random sampling, can be followed, but word frequency should remain a factor in where the sample is taken from.

In step (b), the sample is used to fit a manifold. For LLE (Saul and Roweis, 2000), that is done through learning the weights which can reconstruct each word vector from the sample X through its K-nearest neighbours in the sample, by minimizing the error function:

$$\mathcal{E}(W) = \sum_i \left| \vec{X}_i - \sum_j W_{ij}\vec{X}_j \right|^2 \qquad (1)$$

such that $W_{ij} = 0$ if $\vec{X}_j$ is not in the K-nearest neighbours of $\vec{X}_i$. The weights are then used to construct a new embedding Y of the sample X via a neighbourhood-preserving mapping through minimizing the cost function:

$$\Phi(Y) = \sum_i \left| \vec{Y}_i - \sum_j W_{ij}\vec{Y}_j \right|^2 \qquad (2)$$

In steps (c) and (d), to transform an arbitrary vector $\vec{x}$, the weights are first constructed from only the K-nearest neighbours of $\vec{x}$ in the sample X, by minimizing the function:

$$\mathcal{E}(W^x) = \left| \vec{x} - \sum_j W^x_j \vec{X}_j \right|^2 \qquad (3)$$

such that $W^x_j = 0$ if $\vec{X}_j$ is not in the K-nearest neighbours of $\vec{x}$. The weights are then used along with the new embedding Y to transform $\vec{x}$ into $\vec{y}$ which lives in the new embedding space through the equation:

$$\vec{y} = \sum_j W^x_j \vec{Y}_j \qquad (4)$$

where $\vec{Y}_j$ is the transform, from step (b), of $\vec{X}_j$ that is in the K-nearest neighbours of $\vec{x}$.

4 Experiments

Original Embedding Spaces. The original word embeddings used are pre-trained GloVe models: Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, 50d, 100d, 200d, & 300d vectors), and Common Crawl (42B tokens, 1.9M vocab, 300d vectors) (Pennington et al., 2014b). The vectors are ordered by the frequency of their corresponding words, so the vector representing the word 'the' comes first in the space.

Task. We use similarity tasks WS353 (Finkelstein et al., 2001) and RG65 (Rubenstein and Goodenough, 1965).

Space	Task	GloVe	Re-Embedding
6B 50d	WS353	**61.2**	56.6
6B 50d	RG65	**60.2**	53.0
6B 100d	WS353	**64.5**	64.3
6B 100d	RG65	65.3	**67.3**
6B 200d	WS353	68.5	**69.7**
6B 200d	RG65	75.5	**76.0**
6B 300d	WS353	65.8	**70.3**
6B 300d	RG65	75.5	**80.5**
42B 300d	WS353	75.2	**78.4**
42B 300d	RG65	80.0	**83.4**

Table 1: Average performance on similarity tasks. (Window start $\in [5000, 15000]$, Number of LLE local neighbours =1000, Window length = 1001, Manifold dimensionality = Space dimensionality.)

Baseline. We use the performance by the original word embeddings on the tasks. For each original space, we normalize features using their minimum and maximum values to $[-1, +1]$, and then normalize vectors to unit norms. For each pair of words in the similarity task, we get the normalized vectors and measure the cosine similarity. We finally compute the Spearman Rank Correlation with human judgements.

Approach. For a given original embedding, we normalize vectors to unit norms, then we conduct Manifold (Mfd) Re-Embedding using LLE as explained in Section 3. For each similarity task, we transform the vectors of test words into the re-embedding space before computing the cosine similarity, and the final Spearman score. We vary relevant parameters and see what effect they have on the performance, so we can understand the effectiveness of the approach and its limits.

5 Results and Discussion

Average Performance. Table 1 shows that the re-embedding method outperforms the baseline in most cases with improvements from 0.5% to 5.0%. These results are achieved for effective manifold training windows which start anywhere between 5000 and 15000. The table also shows that improvements are over spaces with underlying bigger corpora and vectors, i.e. good quality vectors which facilitate the embedding.

Manifold Dimensionality Retention. Figure 3 shows that for a given window, the re-embedding performs better when the dimensionality of the learned manifold is chosen to be closer to the original space dimensionality. In other words, dimensional reduction on the original space will bare a cost in performance.

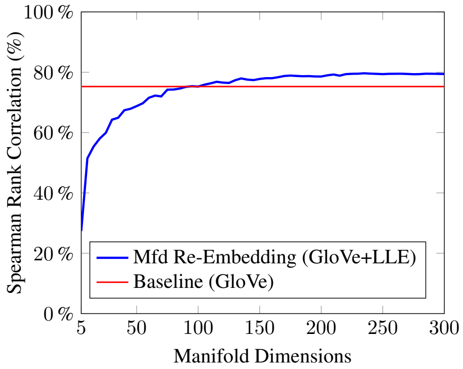


Figure 3: Accuracy on WS353 similarity task as a function of manifold dimensionality. (Space is GloVe 42B 300d. Window start = 7000, LLE local neighbours =1000, Window length = 1001.)

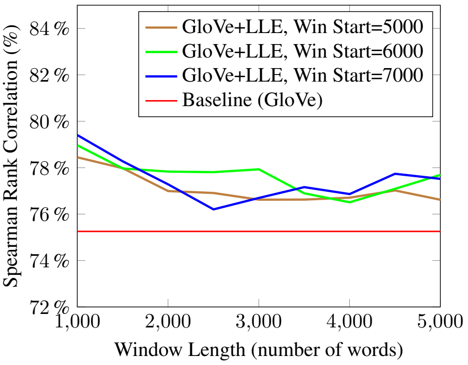
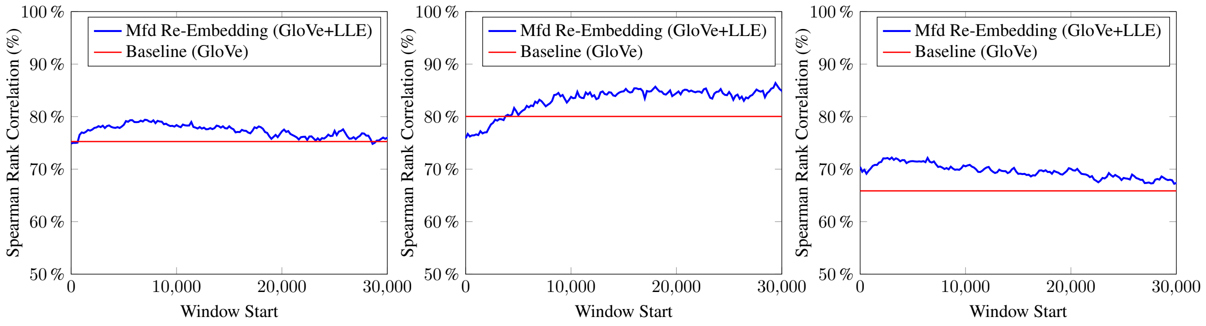


Figure 4: Accuracy on WS353 as a function of window length. (GloVe 42B 300d, LLE local neighbours =1000. Manifold dimensions =300.)

Manifold learning typically starts from a high-dimensional raw space, such as pixels, and aims to reduce the dimensionality. In our method we start from a word embedding which is already a good embedding of the raw word co-occurrences. So, dimensionality shall be retained, as suggested by Figure 3, or otherwise information can be lost during eigenvectors computation and selection in the manifold learning.

Effect of Window Length. Figure 4 shows that the best window length to choose is as close as possible to the number of local neighbours used by the manifold learning. Performance drops slightly with higher values of window length, but becomes stable after an initial drop.

Effect of Window Start. Figure 5 shows that the performance is first modest when the manifold is trained on the most frequent word vectors (i.e. stop words), but then picks up and outperforms the baseline for most cases. Performance drops grad-



(a) 42B 300d, WS353

(b) 42B 300d, RG65

(c) 6B 300d, WS353

Figure 5: Accuracy on similarity tasks as a function of window start. (a) Original space GloVe 42B 300d, with WS353. (b) 42B 300d, with RG65. (c) 6B 300d, with WS353. (LLE local neighbours =1000, Window length = 1001, Manifold dimensionality = 300.)

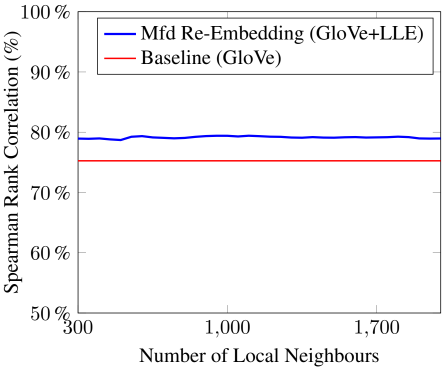


Figure 6: Accuracy on WS353 as a function of the number of manifold local neighbours. (42B 300d, Window start = 7000, Manifold dimensionality = 300, Window length = local neighbours+1.

ually as the manifold is trained on relatively less frequent word vectors.

Effect of the Number of Local Neighbours. Figure 6 shows that the performance is generally stable with variation in the number of local neighbours that the manifold is learned upon. Generally lower numbers of local neighbours mean faster manifold learning.

Discussion. The above results show that word re-embedding based on manifold learning can help the original space recover the Euclidean metric, and thus improves performance on word similarity tasks. The ability of re-embedding to achieve improved results depends on the quality of the vectors in the original space. It also depends on the choice of the window used to learn the manifold. The window start is the most influential variable, and it should be chosen just after the stop words in the original space. The choice of other parameters is relatively easier: the length of the window should be close or equal to the number of local neighbours, which in turn can be chosen from a wide range with no significant difference. The dimensionality of the original embedding space should be retained and used for learning the manifold to guarantee the best re-embedding.

6 Conclusions and Future Work

In this paper we presented a new method to re-embed words from off-the-shelf embeddings based on manifold learning. We showed that such an approach is theoretically founded in the metric recovery paradigm and can empirically improve the performance of state-of-the-art embeddings in word similarity tasks. In future work we intend to extend the experiments to include other original pre-trained embeddings, and other algorithms for manifold learning. We also intend to extend the experiments to other NLP tasks in addition to word similarity such as word analogies.

Acknowledgements

This work was supported with the financial support of the Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero - the Irish Software Research Centre (www.lero.ie).

References

Mukund Balasubramanian and Eric L Schwartz. 2002. The isomap algorithm and topological stability. *Science*, 295(5552):7–7.

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *AISTATS*, volume 15, pages 224–232.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *ACL*, pages 1370–1380. Citeseer.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

André Freitas and Edward Curry. 2014. Natural language queries over heterogeneous linked data graphs: A distributional-compositional semantics approach. In *Proceedings of the 19th international conference on Intelligent User Interfaces*, pages 279–288. ACM.

Souleiman Hasan. 2017. Nosym: Non-symbolic databases for data decoupling. In *the Conference on Innovative Data Systems Research (CIDR)*.

Souleiman Hasan and Edward Curry. 2014. Thematic event processing. In *Proceedings of the 15th International Middleware Conference*, Middleware '14, pages 109–120, New York, NY, USA. ACM.

Tatsunori B Hashimoto, David Alvarez-Melis, and Tommi S Jaakkola. 2016. Word embeddings as metric recovery in semantic spaces. *Transactions of the Association for Computational Linguistics*, 4:273–286.

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *ACL*, pages 489–493.

Yang-Yin Lee, Hao Ke, Hen-Hsen Huang, and Hsin-Hsi Chen. 2016. Less is more: Filtering abnormal dimensions in glove. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 71–72. International World Wide Web Conferences Steering Committee.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

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

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017. All-but-the-top: Simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014a. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014b. Glove resources. *Available at: http://nlp.stanford.edu/projects/glove/*.

Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326.

Herbert Rubenstein and John B Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

David E Rumelhart and Adele A Abrahamson. 1973. A model for analogical reasoning. *Cognitive Psychology*, 5(1):1–28.

Lawrence K Saul and Sam T Roweis. 2000. An introduction to locally linear embedding. *Available at: www.cs.toronto.edu/%7Eroweis/lle/publications.html*.

Robert J Sternberg and Michael K Gardner. 1983. Unities in inductive reasoning. *Journal of Experimental Psychology: General*, 112(1):80.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

MUSE: Modularizing Unsupervised Sense Embeddings

Guang-He Lee[⋆†] **Yun-Nung Chen**[†]
[⋆]Massachusetts Institute of Technology, Cambridge, MA
[†]National Taiwan University, Taipei, Taiwan
guanghe@mit.edu y.v.chen@ieee.org

Abstract

This paper proposes to address the word sense ambiguity issue in an unsupervised manner, where word sense representations are learned along a word sense selection mechanism given contexts. Prior work focused on designing a single model to deliver both mechanisms, and thus suffered from either coarse-grained representation learning or inefficient sense selection. The proposed modular approach, *MUSE*, implements flexible modules to optimize distinct mechanisms, achieving the first purely sense-level representation learning system with linear-time sense selection. We leverage reinforcement learning to enable joint training on the proposed modules, and introduce various exploration techniques on sense selection for better robustness. The experiments on benchmark data show that the proposed approach achieves the state-of-the-art performance on synonym selection as well as on contextual word similarities in terms of MaxSimC.

1 Introduction

Recently, deep learning methodologies have dominated several research areas in natural language processing (NLP), such as machine translation, language understanding, and dialogue systems. However, most of applications usually utilize word-level embeddings to obtain semantics. Considering that natural language is highly ambiguous, the standard word embeddings may suffer from polysemy issues. Neelakantan et al. (2014) pointed out that, due to triangle inequality in vector space, if one word has two different senses but is restricted to *one embedding*, the sum of the distances between the word and its synonym in each sense would upper-bound the distance between the respective synonyms, which may be mutually irrelevant, in embedding space[1]. Due to the theoretical inability to account for polysemy using a single embedding representation per word, multi-sense word representations are proposed to address the ambiguity issue using multiple embedding representations for different senses in a word (Reisinger and Mooney, 2010; Huang et al., 2012).

This paper focuses on unsupervised learning from the unannotated corpus. There are two key mechanisms for a multi-sense word representation system in such scenario: 1) a sense selection (decoding) mechanism infers the most probable sense for a word given its context and 2) a sense representation mechanism learns to embed word senses in a continuous space.

Under this framework, prior work focused on designing a single model to deliver both mechanisms (Neelakantan et al., 2014; Li and Jurafsky, 2015; Qiu et al., 2016). However, the previously proposed models introduce side-effects: 1) mixing word-level and sense-level tokens achieves efficient sense selection but introduces ambiguous word-level tokens during the representation learning process (Neelakantan et al., 2014; Li and Jurafsky, 2015), and 2) pure sense-level tokens prevent ambiguity from word-level tokens but require exponential time complexity when decoding a sense sequence (Qiu et al., 2016).

Unlike the prior work, this paper proposes *MUSE*[2]—a novel modularization framework incorporating sense selection and representation learning models, which implements flexible modules to optimize distinct mechanisms. Specifically,

[1] $d(\text{rock}, \text{stone}) + d(\text{rock}, \text{shake}) \geq d(\text{stone}, \text{shake})$

[2] The trained models and code are available at https://github.com/MiuLab/MUSE.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 327–337
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

MUSE enables linear time sense identity decoding with a *sense selection module* and purely sense-level representation learning with a *sense representation module*.

With the modular design, we propose a novel joint learning algorithm on the modules by connecting to a reinforcement learning scenario, which achieves the following advantages. First, the decision making process under reinforcement learning better captures the sense selection mechanism than probabilistic and clustering methods. Second, our reinforcement learning algorithm realizes the first single objective function for modular unsupervised sense representation systems. Finally, we introduce various exploration techniques under reinforcement learning on sense selection to enhance robustness.

In summary, our contributions are five-fold:

- *MUSE* is the first system that maintains purely sense-level representation learning with linear-time sense decoding.
- We are among the first to leverage reinforcement learning to model the sense selection process in sense representations system.
- We are among the first to propose a single objective for modularized unsupervised sense embedding learning.
- We introduce a sense exploration mechanism for the sense selection module to achieve better flexibility and robustness.
- Our experimental results show the state-of-the-art performance for synonym selection and contextual word similarities in terms of MaxSimC.

2 Related Work

There are three dominant types of approaches for learning multi-sense word representations in the literature: 1) clustering methods, 2) probabilistic modeling methods, and 3) lexical ontology based methods. Our reinforcement learning based approach can be loosely connected to clustering methods and probabilistic modeling methods.

Reisinger and Mooney (2010) first proposed multi-sense word representations on the vector space based on clustering techniques. With the power of deep learning, some work exploited neural networks to learn embeddings with sense selection based on clustering (Huang et al., 2012; Neelakantan et al., 2014). Chen et al. (2014) replaced the clustering procedure with a word sense disambiguation model using WordNet (Miller, 1995). Kågebäck et al. (2015) and Vu and Parker (2016) further leveraged a weighting mechanism and interactive process in the clustering procedure. Moreover, Guo et al. (2014) leveraged bilingual resources for clustering. However, most of the above approaches separated the clustering procedure and the representation learning procedure without a joint objective, which may suffer from the error propagation issue. Instead, the proposed approach, MUSE, enables joint training on sense selection and representation learning.

Instead of clustering, probabilistic modeling methods have been applied for learning multi-sense embeddings in order to make the sense selection more flexible, where Tian et al. (2014) and Jauhar et al. (2015) conducted probabilistic modeling with EM training. Li and Jurafsky (2015) exploited Chinese Restaurant Process to infer the sense identity. Furthermore, Bartunov et al. (2016) developed a non-parametric Bayesian extension on the skip-gram model (Mikolov et al., 2013b). Despite reasonable modeling on sense selection, all above methods mixed word-level and sense-level tokens during representation learning—unable to conduct representation learning in the pure sense level due to the complicated computation in their EM algorithms.

Recently, Qiu et al. (2016) proposed an EM algorithm to learn purely sense-level representations, where the computational cost is high when decoding the sense identity sequence, because it takes exponential time to search all sense combination within a context window. Our modular design addresses such drawback, where the sense selection module decodes a sense sequence with linear-time complexity, while the sense representation module remains representation learning in the pure sense level.

Unlike a lot of relevant work that requires additional resources such as the lexical ontology (Pilehvar and Collier, 2016; Rothe and Schütze, 2015; Jauhar et al., 2015; Chen et al., 2015; Iacobacci et al., 2015) or bilingual data (Guo et al., 2014; Ettinger et al., 2016; Šuster et al., 2016), which may be unavailable in some language, our model can be trained using only an unlabeled corpus. Also, some prior work proposed to learn topical embeddings and word embeddings jointly in order to consider the contexts (Liu et al., 2015a,b), whereas this paper focuses on learning multi-sense

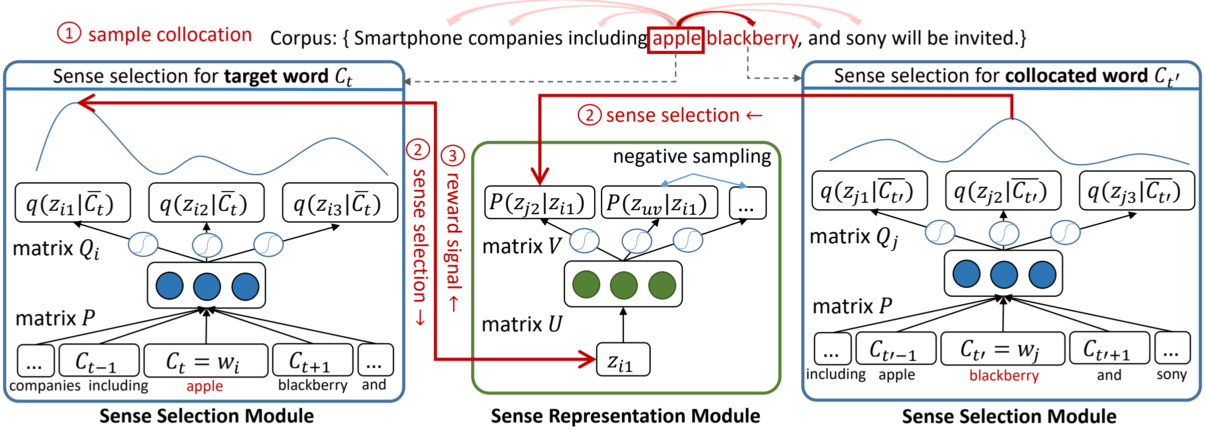


Figure 1: The *MUSE* architecture with a 3-step learning algorithm: 1) collocation sampling, 2) sense selection for sense representation learning, and 3) optimizing sense selection with a reward signal from sense representation. Reward signal is only passed to the target word to stabilize model training due to directional architecture in the sense representation module.

word embeddings.

3 Proposed Approach: MUSE

This work proposes a framework to modularize two key mechanisms for multi-sense word representations: a *sense selection module* and a *sense representation module*. The sense selection module decides which sense to use given a text context, whereas the sense representation module learns meaningful representations based on its statistical characteristics. Unlike prior work that must suffer from either inefficient sense selection (Qiu et al., 2016) or coarse-grained representation learning (Neelakantan et al., 2014; Li and Jurafsky, 2015; Bartunov et al., 2016), the proposed modularized framework is capable of performing efficient sense selection and learning representations in pure sense level simultaneously.

To learn sense-level representations, a sense selection model should be first established for sense identity decoding. On the other hand, the sense embeddings should guide the sense selection model when decoding a sense identity sequence. Therefore, these two modules should be tangled. This indicates that a naive two-stage algorithm or two separate learning algorithms proposed by prior work are not optimal.

By connecting the proposed formulation with reinforcement learning literature, we design a novel joint training algorithm. Besides, taking advantage of the form of reinforcement learning, we are among the first to investigate various exploration techniques in sense selection for unsupervised sense embedding learning.

3.1 Model Architecture

Our model architecture is illustrated in Figure 1, where there are two modules in optimization.

3.1.1 Sense Selection Module

Formally speaking, given a corpus C, vocabulary W, and the t-th word $C_t = w_i \in W$, we would like to find the most probable sense $z_{ik} \in Z_i$, where Z_i is the set of senses in word w_i. Assuming that a word sense is determined by the local context, we exploit a local context $\bar{C}_t = \{C_{t-m}, \cdots, C_{t+m}\}$ for sense selection according to the Markov assumption, where m is the size of a context window. Then we can either formulate a probabilistic policy $\pi(z_{ik} \mid \bar{C}_t)$ about sense selection or estimate the *individual* likelihood $q(z_{ik} \mid \bar{C}_t)$ for each sense identity.

To ensure efficiency, here we exploit a linear neural architecture that takes word-level input tokens and outputs sense-level identities. The architecture is similar to continuous bag-of-words (CBOW) (Mikolov et al., 2013a). Specifically, given a *word* embedding matrix P, the local context can be modeled as the summation of word embeddings from its context $\bar{C}_t$. The output can be formulated with a 3-mode tensor Q, whose dimensions denote words, senses, and latent variables. Then we can model $\pi(z_{ik} \mid \bar{C}_t)$ or $q(z_{ik} \mid \bar{C}_t)$ correspondingly. Here we model $\pi(\cdot)$ as a *categorical* distribution using a softmax layer:

$$\pi(z_{ik} \mid \bar{C}_t) = \frac{\exp(Q_{ik}^T \sum_{j \in \bar{C}_t} P_j)}{\sum_{k' \in Z_i} \exp(Q_{ik'}^T \sum_{j \in \bar{C}_t} P_j)}. \tag{1}$$

On the other hand, the likelihood of selecting *distinct* sense identities, $q(z_{ik} \mid \bar{C}_t)$, is modeled as a *Bernoulli* distribution with a sigmoid function $\sigma(\cdot)$:

$$q(z_{ik} \mid \bar{C}_t) = \sigma(Q_{ik}^T \sum_{j \in \bar{C}_t} P_j). \tag{2}$$

Different modeling approaches require different learning methods, especially for the unsupervised setting. We leave the corresponding learning algorithms in § 3.2. Finally, with a built sense selection module, we can apply any selection algorithm such as a greedy selection strategy to infer the sense identity z_{ik} given a word w_i with its context C_t.

We note that modularized model enables efficient sense selection by leveraging word-level tokens, while remaining purely sense-level tokens in the representation module. Specifically, if n denotes $\max_k |Z_k|$, decoding L words takes $O(nL)$ senses to be searched due to independent sense selection. The prior work using a single model with purely sense-level tokens (Qiu et al., 2016) requires exponential time to calculate the collocation energy for every possible combination of sense identities within a context window, $O(n^{2m})$, for a *single target sense*. Further, Qiu et al. (2016) took an additional sequence decoding step with quadratic time complexity $O(n^{4m}L)$, based on an exponential number n^{2m} in the base unit. It demonstrates the achievement about sense inference efficiency in our proposed model.

3.1.2 Sense Representation Module

The semantic representation learning is typically formulated as a maximum likelihood estimation (MLE) problem for collocation likelihood. In this paper, we use the skip-gram formulation (Mikolov et al., 2013b) considering that it requires less training time, where only two sense identities are required for stochastic training. Other popular candidates, like GloVe (Pennington et al., 2014) and CBOW (Mikolov et al., 2013a), require more sense identities to be selected as input and thus not suitable for our scenario. For example, GloVe (Pennington et al., 2014) takes computationally expensive collocation *counting* statistics for each token in a corpus as input, which requires sense selection for every occurrence of the target word across the whole corpus for a single optimization step.

To learn the representations, we first create input sense representation matrix U and collocation estimation matrix V as the learning targets. Given a target word w_i and collocated word w_j with corresponding local contexts, we map them to their sense identities as z_{ik} and z_{jl} by the sense selection module, and maximize the sense collocation log likelihood $\log \mathcal{L}(\cdot)$. A natural choice of the likelihood function is formulated as a categorical distribution over all possible collocated senses given the target sense z_{ik}:

$$\max_{U,V} \ \log \mathcal{L}(z_{jl} \mid z_{ik}) = \log \frac{\exp(U_{z_{ik}}^T V_{z_{jl}})}{\sum_{z_{uv}} \exp(U_{z_{ik}}^T V_{z_{uv}})}. \tag{3}$$

Instead of enumerating all possible collocated senses which is computationally expensive, we use the skip-gram objective (4) (Mikolov et al., 2013b) to approximate (3) as shown in the green block of Figure 1.

$$\max_{U,V} \ \log \bar{\mathcal{L}}(z_{jl} \mid z_{ik}) = \log \sigma(U_{z_{ik}}^T V_{z_{jl}}) \tag{4}$$
$$+ \sum_{v=1}^{M} \mathbb{E}_{z_{uv} \sim p_{neg}(z)} [\log \sigma(-U_{z_{ik}}^T V_{z_{uv}})],$$

where $p_{neg}(z)$ is the distribution over all senses for negative samples. In our experiment with $|Z_i|$ senses for word w_i, we use $(1/|Z_i|)$ word-level unigram as sense-level unigram for efficiency and the 3/4-th power trick in Mikolov et al. (2013b).

We note that our modular framework can easily maintain purely sense-level tokens with an arbitrary representation learning model. In contrast, most related work using probabilistic modeling (Tian et al., 2014; Jauhar et al., 2015; Li and Jurafsky, 2015; Bartunov et al., 2016) binded sense representations with the sense selection mechanism, so efficient sense selection by leveraging word-level tokens can be achieved only at the cost of mixing word-level and sense-level tokens in their representation learning process.

3.2 Learning

Without the supervised signal for the proposed modules, it is desirable to connect two modules in a way where they can improve each other by their own estimations. First, a trivial way is to forward the prediction of the sense selection module to the representation module. Then we cast the estimated collocation likelihood as a reward signal for the selected sense for effective learning.

To realize the above procedure, we cast the learning problem a one-step Markov decision process (MDP) (Sutton and Barto, 1998), where the state, action, and reward correspond to context $\bar{C}_t$, sense z_{ik}, and collocation log likelihood $\log \bar{\mathcal{L}}(\cdot)$, respectively. Based on different modeling methods ((1) or (2)) in the sense selection module, we connect the model to respective reinforcement learning algorithms to solve the MDP. Specifically, we refer (1) to policy distribution and refer (2) to Q-value estimation in the reinforcement learning literature.

The proposed MDP framework embodies several nuances of sense selection. First, the decision of a word sense is Markov: taking the whole corpus into consideration is not more helpful than a handful of necessary local contexts. Second, the decision making in MDP exploits a hard decision for selecting sense identity, which captures the sense selection process more naturally than a joint probability distribution among senses (Qiu et al., 2016). Finally, we exploit the reward mechanism in MDP to enable joint training: the estimation of sense representation is treated as a reward signal to guide sense selection. In contrast, the decision making under clustering (Huang et al., 2012; Neelakantan et al., 2014) considers the similarity within clusters instead of the outcome of a decision using a reward signal as MDP.

3.2.1 Policy Gradient Method

Because (1) fits a valid probability distribution, an intuitive optimization target is the expectation of resulting collocation likelihood among each sense. In addition, as the skip-gram formulation in (4) is unidirectional ($\bar{\mathcal{L}}(z_{ik} \mid z_{jl}) \neq \bar{\mathcal{L}}(z_{jl} \mid z_{ik})$), we perform one-side optimization for the target sense z_{ik} to stabilize model training[3]. That is, for the target word w_i and the collocated word w_j given respective contexts $\bar{C}_t$ and $\bar{C}_{t'}$ $(0 < |t - t'| \leq m)$, we first draw a sense z_{jl} for w_j from the policy $\pi(\cdot \mid \bar{C}_{t'})$ and optimize the expected collocation likelihood for the target sense z_{ik} as follows,

$$\max_{P,Q} \; \mathbb{E}_{z_{ik} \sim \pi(\cdot \mid \bar{C}_t)} [\log \bar{\mathcal{L}}(z_{jl} \mid z_{ik})]. \quad (5)$$

Note that (4) can be merged into (5) as a single objective. The objective is differentiable and supports stochastic optimization (Lei et al., 2016), which uses a stochastic sample z_{ik} for optimization.

However, there are two possible disadvantages in this formulation. First, because the policy assumes the probability distribution in (1), optimizing the selected sense must affect the estimation of the other senses. Second, if applying stochastic gradient ascent to optimizing (5), it would always lower the probability estimation for the selected sense z_{ik} even if the model accurately selects the right sense. The detailed proof is in Appendix A.

3.2.2 Value-Based Method

To address the above issues, we apply the Q-learning algorithm (Mnih et al., 2013). Instead of maintaining a probabilistic policy for sense selection, Q-learning estimates the Q-value (resulting collocation log likelihood) for each sense candidate directly and independently. Thus, the estimation of unselected senses may not be influenced by the selected one. Note that in one-step MDP, the reward is equivalent to the Q-value, so we will use reward and Q-value interchangeably, hereinafter, based on the context.

We further follow the convention of recent neural reinforcement learning by reducing the reward range to aid model training (Mnih et al., 2013). Specifically, we replace the *log likelihood* $\log \bar{\mathcal{L}}(\cdot) \in (-\inf, 0]$ with the *likelihood* $\bar{\mathcal{L}}(\cdot) \in [0, 1]$ as the reward function. Due to the monotonic operation in $\log()$, the relative ordering of the reward remains the same.

Furthermore, we exploit the probabilistic nature of likelihood for Q-learning. To elaborate, as Q-learning is used to approximate the Q-value for each action in typical reinforcement learning, most literature adopted square loss to characterize the discrepancy between the target and estimated Q-values (Mnih et al., 2013). In our setting where the Q-value/reward is a likelihood function, our model exploits cross-entropy loss to better capture the characteristics of probability distribution.

Given that the collocation likelihood in (4) is an *approximation* to the original categorical distribution with a softmax function shown in (3) (Mikolov et al., 2013b), we revise the formulation by omitting the negative sampling term. The resulting formulation $\hat{\mathcal{L}}(\cdot)$ is a Bernoulli distribution indicating whether z_{jl} collocates or not given z_{ik}:

$$\hat{\mathcal{L}}(z_{jl} \mid z_{ik}) = \sigma(U_{z_{ik}}^T V_{z_{jl}}). \quad (6)$$

[3] We observe about 4% performance drop by optimizing input selection z_{ik} and output selection z_{jl} simultaneously.

There are three advantages about using $\hat{\mathcal{L}}(\cdot)$ instead of approximated $\bar{\mathcal{L}}(\cdot)$ and original $\mathcal{L}(\cdot)$. First, regarding the variance of estimation, $\hat{\mathcal{L}}(\cdot)$ better captures $\mathcal{L}(\cdot)$ than $\bar{\mathcal{L}}(\cdot)$ because $\bar{\mathcal{L}}(\cdot)$ involves sampling:

$$Var(\bar{\mathcal{L}}(\cdot)) \geq Var(\hat{\mathcal{L}}(\cdot)) = Var(\mathcal{L}(\cdot)) = 0. \quad (7)$$

Second, regarding the relative ordering of estimation, for any two collocated senses z_{jl} and $z_{jl'}$ with a target sense z_{ik}, the following equivalence holds:

$$\begin{aligned} & \mathcal{L}(z_{jl} \mid z_{ik}) < \mathcal{L}(z_{jl'} \mid z_{ik}) \\ \Leftrightarrow \quad & \bar{\mathcal{L}}(z_{jl} \mid z_{ik}) < \bar{\mathcal{L}}(z_{jl'} \mid z_{ik}) \\ \Leftrightarrow \quad & \hat{\mathcal{L}}(z_{jl} \mid z_{ik}) < \hat{\mathcal{L}}(z_{jl'} \mid z_{ik}) \end{aligned} \quad (8)$$

Third, for collocation computation, $\mathcal{L}(\cdot)$ requires all sense identities and $\bar{\mathcal{L}}(\cdot)$ requires $(M+1)$ sense identities, whereas $\hat{\mathcal{L}}(\cdot)$ only requires 1 sense identity. In sum, the proposed $\hat{\mathcal{L}}(\cdot)$ approximates $\mathcal{L}(\cdot)$ with no variance, no "bias" (in terms of relative ordering), and significantly less computation.

Finally, because both target distribution $\hat{\mathcal{L}}(\cdot)$ and estimated distribution $q(\cdot)$ in (2) are Bernoulli distributions, we follow the last section to conduct one-side optimization by fixing a collocated sense z_{jl} and optimize the selected sense z_{ik} with cross entropy as

$$\min_{P,Q} \; H(\hat{\mathcal{L}}(z_{ik} \mid z_{jl}), q(z_{ik} \mid \bar{C}_t)). \quad (9)$$

3.2.3 Joint Training

To jointly train sense selection and sense representation modules, we first select a pair of the collocated senses, z_{ik} and z_{jl}, based on the sense selection module with any selecting strategy (e.g. greedy), and then optimize the sense representation module and the sense selection module using the above derivations. Algorithm 1 describes the proposed MUSE model training procedure.

As modular frameworks, the major distinction between our modular framework and two-stage clustering-representation learning framework (Neelakantan et al., 2014; Vu and Parker, 2016) is that we establish a reward signal from the sense representation to the sense selection module to enable immediate and joint optimization.

3.3 Sense Selection Strategy

Given a fitness estimation for each sense, exploiting the greedy sense is the most popular strategy for clustering algorithms (Neelakantan et al., 2014; Kågebäck et al., 2015) and hard-EM algorithms (Qiu et al., 2016; Jauhar et al., 2015) in literature. However, there are two incentives to conduct exploration. First, in the early training stage when the fitness is not well estimated, it is desirable to explore underestimated senses. Second, due to high ambiguity in natural language, sometimes multiple senses in a word would fit in the same context. The dilemma between exploring sub-optimal choices and exploiting the optimal choice is called exploration-exploitation trade-off in reinforcement learning (Sutton and Barto, 1998).

Algorithm 1: Learning Algorithm

```
for $w_i = C_t \in C$ do
    sample $w_j = C_{t'} (0 < |t' - t| \leq m)$;
    $z_{ik} =$ select$(C_t, w_i)$;
    $z_{jl} =$ select$(C_{t'}, w_j)$;
    optimize $U, V$ by (4) for the sense
      representation module;
    optimize $P, Q$ by (5) or (9) for the sense
      selection module;
```

We introduce exploration mechanisms for sense selection for both policy gradient and Q-learning. For policy gradient, we sample the policy distribution to approximate the expectation in (5). Because of the flexible formulation of Q-learning, the following classic exploration mechanisms are applied to sense selection:

- *Greedy*: selects the sense with the largest Q-value (no exploration).
- ϵ-*Greedy*: selects a random sense with ϵ probability, and adopts the greedy strategy otherwise (Mnih et al., 2013).
- *Boltzmann*: samples the sense based on the Boltzmann distribution modeled by Q-value. We directly use (1) as the Boltzmann distribution for simplicity.

We note that Q-learning with Boltzmann sampling yields the same sampling process as policy gradient but different optimization objectives. To our best knowledge, we are among the first to explore several exploration strategies for unsupervised sense embedding learning.

In the following sections, MUSE-Policy denotes the proposed MUSE model with policy learning and MUSE-Greedy denotes the model using corresponding sense selection strategy for Q-learning.

4 Experiments

We evaluate our proposed MUSE model in both quantitative and qualitative experiments.

4.1 Experimental Setup

Our model is trained on the April 2010 Wikipedia dump (Shaoul and Westbury, 2010), which contains approximately 1 billion tokens. For fair comparison, we adopt the same vocabulary set as Huang et al. (2012) and Neelakantan et al. (2014). For preprocessing, we convert all words to their lower cases, apply the Stanford tokenizer and the Stanford sentence tokenizer (Manning et al., 2014), and remove all sentences with less than 10 tokens. The number of senses per word in Q is set to 3 as the prior work (Neelakantan et al., 2014).

In the experiments, the context window size is set to 5 ($|\bar{C}_t| = 11$). Subsampling technique introduced by `word2vec` (Mikolov et al., 2013b) is applied to accelerate the training process. The learning rate is set to 0.025. The embedding dimension is 300. We initialize Q and V as zeros, and P and U from uniform distribution $[-\sqrt{1/100}, \sqrt{1/100}]$ such that each embedding has unit length in expectation (Lei et al., 2015). Our model uses 25 negative senses for negative sampling in (4). We use $\epsilon = 5\%$ for ϵ-Greedy sense selection strategy

In optimization, we conduct mini-batch training with 2048 batch size using the following procedure: 1) select senses in the batch; 2) optimize U, V using stochastic training within the batch for efficiency; 3) optimize P, Q using mini-batch training for robustness.

4.2 Experiment 1: Contextual Word Similarity

To evaluate the quality of the learned sense embeddings, we compute the similarity score between each word pair given their respective local contexts and compare with the human-judged score using Stanford's Contextual Word Similarities (SCWS) dataset (Huang et al., 2012). Specifically, given a list of word pairs with corresponding contexts, $S = \{(w_i, \bar{C}_t, w_j, \bar{C}_{t'})\}$, we calculate the Spearman's rank correlation ρ between human-judged similarity and model similarity estimations[4]. Two major contextual similarity estimations are introduced by Reisinger and Mooney (2010): AvgSimC and MaxSimC. AvgSimC is a *soft* measurement that addresses the contextual information with a probability estimation:

Method	MaxSimC	AvgSimC
Huang et al. (2012)	26.1	65.7
Neelakantan et al. (2014)	60.1	**69.3**
Tian et al. (2014)	63.6	65.4
Li and Jurafsky (2015)	66.6	66.8
Bartunov et al. (2016)	53.8	61.2
Qiu et al. (2016)	64.9	66.1
MUSE-Policy	66.1	67.4
MUSE-Greedy	66.3	68.3
MUSE-ϵ-Greedy	67.4†	68.6
MUSE-Boltzmann	**67.9**†	68.7

Table 1: Spearman's rank correlation ρ x100 on the SCWS dataset. † denotes superior performance to all unsupervised competitors.

$$\text{AvgSimC}(w_i, \bar{C}_t, w_j, \bar{C}_{t'}) = \sum_{k=1}^{|Z_i|}\sum_{l=1}^{|Z_j|} \pi(z_{ik}|\bar{C}_t)\pi(z_{jl}|\bar{C}_{t'})d(z_{ik}, z_{jl}),$$

where $d(z_{ik}, z_{jl})$ refers to the cosine similarity between $U_{z_{ik}}$ and $U_{z_{jl}}$. AvgSimC weights the similarity measurement of each sense pair z_{ik} and z_{jl} by their probability estimations. On the other hand, MaxSimC is a *hard* measurement that only considers the most probable senses:

$$\begin{aligned}\text{MaxSimC}(w_i, \bar{C}_t, w_j, \bar{C}_{t'}) &= d(z_{ik}, z_{jl}),\\ z_{ik} &= \arg\max_{z_{ik'}} \pi(z_{ik'}|\bar{C}_t),\\ z_{jl} &= \arg\max_{z_{jl'}} \pi(z_{jl'}|\bar{C}_{t'}).\end{aligned}$$

The baselines for comparison include classic clustering methods (Huang et al., 2012; Neelakantan et al., 2014), EM algorithms (Tian et al., 2014; Qiu et al., 2016; Bartunov et al., 2016), and Chinese Restaurant Process (Li and Jurafsky, 2015)[5], where all approaches are trained on the same corpus except Qiu et al. (2016) used more recent Wikipedia dumps. The embedding sizes of all baselines are 300, except 50 in Huang et al. (2012). For every competitor with multiple settings, we report the best performance in each similarity measurement setting and show in Table 1.

[4] For example, human-judged similarity between "... east **bank** of the Des Moines River ..." and "... basis of all **money** laundering ..." is 2.5 out of 10.0 in SCWS dataset (Huang et al., 2012).

[5] We run Li and Jurafsky (2015)'s released code on our corpus for fair comparison.

Method	**ESL-50**	**RD-300**	**TOEFL-80**
1) Conventional Word Embedding			
Global Context	47.73	45.07	60.87
Skip-Gram	52.08	55.66	66.67
2) Word Sense Disambiguation			
IMS+SG	41.67	53.77	66.67
3) Unsupervised Sense Embeddings			
EM	27.08	33.96	40.00
MSSG	57.14	58.93	78.26
CRP	50.00	55.36	82.61
MUSE-Policy	52.38	51.79	79.71
MUSE-Greedy	57.14	58.93	79.71
MUSE-ϵ-Greedy	61.90†	62.50†	84.06†
MUSE-Boltzmann	**64.29**†	**66.07**†	**88.41**†
4) Supervised Sense Embeddings			
Retro-GC	63.64	66.20	71.01
Retro-SG	56.25	65.09	73.33

Table 2: Accuracy on synonym selection. † denotes superior performance to all unsupervised competitors.

Our MUSE model achieves the state-of-the-art performance on MaxSimC, demonstrating superior quality on independent sense embeddings. On the other hand, MUSE achieves comparable performance with the best competitor in terms of AvgSimC (68.7 vs. 69.3), while MUSE outperforms the same competitor significantly in terms of MaxSimC (67.9 vs. 60.1). The results demonstrate not only the high quality of sense representations but also accurate sense selection.

From the application perspective, MaxSimC refers to a typical scenario using single embedding per word, while AvgSimC employs multiple sense vectors simultaneously per word, which not only brings computational overhead but changes existing neural architecture for NLP. Hence, we argue that MaxSimC better characterize practical usage of a sense representation system than AvgSimC.

Among various learning methods for MUSE, policy gradient performs worst, echoing our argument in § 3.2.1. On the other hand, the superior performance of Boltzmann sampling and ϵ-Greedy over Greedy selection demonstrates the effectiveness of exploration.

Finally, replacing $\bar{\mathcal{L}}(\cdot)$ with $\hat{\mathcal{L}}(\cdot)$ as the reward signal yields 2.3 times speedup for MUSE-ϵ-Greedy and 1.3 times speedup for MUSE-Boltzmann to reach 67.0 in MaxSimC, which demonstrates the efficacy of proposed approximation $\hat{\mathcal{L}}(\cdot)$ over typical $\bar{\mathcal{L}}(\cdot)$ in terms of convergence.

4.3 Experiment 2: Synonym Selection

We further evaluate our model on synonym selection using multi-sense word representations (Jauhar et al., 2015). Three standard synonym selection datasets, ESL-50 (Turney, 2001), RD-300 (Jarmasz and Szpakowicz, 2004), and TOEFL-80 (Landauer and Dumais, 1997), are performed. In the datasets, each question consists of a question word w_Q and four answer candidates $\{w_A, w_B, w_C, w_D\}$, and the goal is to select the most semantically synonymous choice among the four candidates. For example, in the TOEFL-80 dataset, a question shows {(Q) enormously, (A) appropriately, (B) uniquely, (C) tremendously, (D) decidedly}, and the answer is (C). For multi-sense representations system, it selects the synonym of the question word w_Q using the maximum sense-level cosine similarity as a proxy of the semantic similarity (Jauhar et al., 2015).

Our model is compared with the following baselines: 1) conventional word embeddings: global context vectors (Huang et al., 2012) and skip-gram (Mikolov et al., 2013b); 2) applying supervised word sense disambiguation using the IMS system and then applying skip-gram on disambiguated corpus (IMS+SG) (Zhong and Ng, 2010); 3) unsupervised sense embeddings: EM algorithm (Jauhar et al., 2015), multi-sense skip-gram (MSSG) (Neelakantan et al., 2014), Chinese restaurant process (CRP) (Li and Jurafsky, 2015), and the MUSE models; 4) supervised sense embeddings with WordNet (Miller, 1995): retrofitting global context vectors (Retro-GC) and retrofitting skip-gram (Retro-SG) (Jauhar et al., 2015).

Among unsupervised sense embedding approaches, CRP and MSSG refer to the baselines with highest MaxSimC and AvgSimC in Table 1 respectively. Here we report the setting for baselines based on the best average performance in this task. We also show the performance of supervised sense embeddings as an upperbound of unsupervised methods due to the usage of additional supervised information from WordNet.

The results are shown in Table 2, where our MUSE-ϵ-Greedy and MUSE-Boltzmann significantly outperform all unsupervised sense embeddings methods, echoing the superior quality of our

Context	k-NN Senses
··· braves finish the season in **tie** with the los angeles dodgers ···	scoreless otl shootout 6-6 hingis 3-3 7-7 0-0
··· his later years proudly wore **tie** with the chinese characters for ···	pants trousers shirt juventus blazer socks anfield
··· of the mulberry or the **blackberry** and minos sent him to ···	cranberries maple vaccinium apricot apple
··· of the large number of **blackberry** users in the us federal ···	smartphones sap microsoft ipv6 smartphone
··· shells and/or high explosive squash **head** hesh and/or anti-tank ···	venter thorax neck spear millimeters fusiform
··· head was shaven to prevent **head** lice serious threat back then ···	shaved thatcher loki thorax mao luthor chest
··· appoint john pope republican as **head** of the new army of ···	multi-party appoints unicameral beria appointed

Table 3: Different word senses are selected by MUSE according to different contexts. The respective k-NN (sorted by collocation likelihood) senses are shown to indicate respective semantic meanings.

sense vectors in last section. MUSE-Boltzmann also outperforms the supervised sense embeddings except 1 setting without any supervised signal during training. Finally, the MUSE methods with proper exploration outperform all unsupervised baselines consistently, demonstrating the importance of exploration.

4.4 Qualitative Analysis

We further conduct qualitative analysis to check the semantic meanings of different senses learned by MUSE with k-nearest neighbors (k-NN) using sense representations. In addition, we provide contexts in the training corpus where the sense will be selected to validate the sense selection module. Table 3 shows the results. The learned sense embeddings of the words “tie”, “blackberry”, and “head” clearly correspond to correct senses under different contexts.

Since we address an unsupervised setting that learns sense embeddings from unannotated corpus, the discovered senses highly depend on the training corpus. From our manual inspection, it is common for our model to discover only two senses in a word, like “tie” and “blackberry”. However, we maintain our effort in developing unsupervised sense embeddings learning methods in this work, and the number of discovered sense is not a focus.

5 Conclusion

This paper proposes a novel modularized framework for unsupervised sense representation learning, which supports not only the flexible design of modular tasks but also joint optimization among modules. The proposed model is the first work that implements purely sense-level representation learning with linear-time sense selection, and achieves the state-of-the-art performance on benchmark contextual word similarity and synonym selection tasks. In the future, we plan to investigate reinforcement learning methods to incorporate multi-sense word representations for downstream NLP tasks.

Acknowledgements

We would like to thank reviewers for their insightful comments on the paper. The authors are supported by the Ministry of Science and Technology of Taiwan under the contract number 105-2218-E-002-033, Institute for Information Industry, and MediaTek Inc..

References

Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry Vetrov. 2016. Breaking sticks and ambiguities with adaptive skip-gram. *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, page 130–138.

Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. 2015. Improving distributed representation of word sense via wordnet gloss composition and context clustering. Association for Computational Linguistics.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *EMNLP*, pages 1025–1035. Citeseer.

Allyson Ettinger, Philip Resnik, and Marine Carpuat. 2016. Retrofitting sense-specific word vectors using parallel text. In *Proceedings of NAACL-HLT*, pages 1378–1383.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning sense-specific word embeddings by exploiting bilingual resources. In *COLING*, pages 497–507.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple

Word Prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *ACL (1)*, pages 95–105.

Mario Jarmasz and Stan Szpakowicz. 2004. Roget's thesaurus and semantic similarity. *Recent Advances in Natural Language Processing III: Selected Papers from RANLP*, 2003:111.

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

Mikael Kågebäck, Fredrik Johansson, Richard Johansson, and Devdatt Dubhashi. 2015. Neural context embeddings for automatic discovery of word senses. In *Proceedings of NAACL-HLT*, pages 25–32.

Thomas K Landauer and Susan T Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. *EMNLP*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2015a. Learning context-sensitive word embeddings with neural tensor skip-gram model. In *IJCAI*, pages 1284–1290.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015b. Topical word embeddings. In *AAAI*, pages 2418–2424.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

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

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

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *NIPS Deep Learning Workshop*.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. volume 14, pages 1532–1543.

Mohammad Taher Pilehvar and Nigel Collier. 2016. De-conflated semantic representations. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Lin Qiu, Kewei Tu, and Yong Yu. 2016. Context-dependent sense embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the ACL*.

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

Simon Šuster, Ivan Titov, and Gertjan van Noord. 2016. Bilingual learning of multi-sense embeddings with discrete autoencoders. *NAACL-HLT 2016*.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING*, pages 151–160.

Peter D Turney. 2001. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *European Conference on Machine Learning*, pages 491–502. Springer.

Thuy Vu and D Stott Parker. 2016. K-embeddings: Learning conceptual embeddings for words using context. In *Proceedings of NAACL-HLT*, pages 1262–1267.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83. Association for Computational Linguistics.

A Doubly Stochastic Gradient

To derive doubly stochastic gradient for equation (5), we first denote (5) as $J(\Theta)$ with $\Theta = \{P, Q\}$ and resolve the expectation form as:

$$\begin{aligned} J(\theta) &= \mathbb{E}_{z_{ik} \sim \pi(\cdot|\bar{C}_t)}[\log \bar{\mathcal{L}}(z_{jl} \mid z_{ik})] \\ &= \sum_k \pi(z_{ik} \mid \bar{C}_t) \log \bar{\mathcal{L}}(z_{jl}|z_{ik}). \end{aligned}$$

Denote $\Theta = \{P, Q\}$ as the parameter set for policy π. The gradient with respect to Θ should be:

$$\frac{\partial J(\theta)}{\partial \Theta} = \frac{\partial}{\partial \Theta} \sum_k \pi(z_{ik} \mid \bar{C}_t) \log \bar{\mathcal{L}}(z_{jl}|z_{ik})$$

$$= \sum_k \log \bar{\mathcal{L}}(z_{jl}|z_{ik}) \frac{\partial \pi(z_{ik} \mid \bar{C}_t)}{\partial \Theta}$$

$$= \sum_k \log \bar{\mathcal{L}}(z_{jl}|z_{ik}) (\frac{\partial \log \pi(z_{ik} \mid \bar{C}_t)}{\partial \Theta})(\pi(z_{ik} \mid \bar{C}_t))$$

$$= \mathbb{E}_{z_{ik} \sim \pi(\cdot|\bar{C}_t)}[\log \bar{\mathcal{L}}(z_{jl} \mid z_{ik}) \frac{\partial \log \pi(z_{ik} \mid \bar{C}_t)}{\partial \Theta}]$$

Accordingly, if we conduct typical stochastic gradient ascent training on $J(\Theta)$ with respect to Θ from samples z_{ik} with a learning rate η, the update formula will be:

$$\Theta = \Theta + \eta \log \bar{\mathcal{L}}(z_{jl} \mid z_{ik}) \frac{\partial \log \pi(z_{ik} \mid \bar{C}_t)}{\partial \Theta}.$$

However, the collocation log likelihood should always be non-positive: $\log \bar{\mathcal{L}}(z_{jl} \mid z_{ik}) \leq 0$. Therefore, as long as the collocation log likelihood $\log \bar{\mathcal{L}}(z_{jl} \mid z_{ik})$ is negative, the update formula is to minimize the likelihood of choosing z_{ik}, despite the fact that z_{ik} may be good choices. On the other hand, if the log likelihood reaches 0, according to (4), it indicates:

$$\begin{aligned} &\log \bar{\mathcal{L}}(z_{jl} \mid z_{ik}) = 0 \Rightarrow \quad \bar{\mathcal{L}}(z_{jl} \mid z_{ik}) = 1 \\ \Rightarrow \quad & U_{z_{ik}}^T V_{z_{jl}} \to \infty, \ U_{z_{ik}}^T V_{z_{uv}} \to \infty, \ \forall z_{uv}, \end{aligned}$$

which leads to computational overflow from an infinity value.

Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging

Nils Reimers and **Iryna Gurevych**
Ubiquitous Knowledge Processing Lab (UKP) and Research Training Group AIPHES
Department of Computer Science, Technische Universität Darmstadt
Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research
www.ukp.tu-darmstadt.de

Abstract

In this paper we show that reporting a single performance score is insufficient to compare non-deterministic approaches. We demonstrate for common sequence tagging tasks that the seed value for the random number generator can result in *statistically significant* ($p < 10^{-4}$) differences for state-of-the-art systems. For two recent systems for NER, we observe an absolute difference of one percentage point F_1-score depending on the selected seed value, making these systems perceived either as *state-of-the-art* or *mediocre*. Instead of publishing and reporting single performance scores, we propose to compare score distributions based on multiple executions. Based on the evaluation of 50.000 LSTM-networks for five sequence tagging tasks, we present network architectures that produce both superior performance as well as are more stable with respect to the remaining hyperparameters. The full experimental results are published in (Reimers and Gurevych, 2017).[1] The implementation of our network is publicly available.[2]

1 Introduction

Large efforts are spent in our community on developing new *state-of-the-art* approaches. To document that those approaches are better, they are applied to unseen data and the obtained performance score is compared to previous approaches. In order to make results comparable, a provided split between train, development and test data is often used, for example from a former shared task.

In recent years, deep neural networks were shown to achieve state-of-the-art performance for a wide range of NLP tasks, including many sequence tagging tasks (Ma and Hovy, 2016), dependency parsing (Andor et al., 2016), and machine translation (Wu et al., 2016). The training process for neural networks is highly non-deterministic as it usually depends on a random weight initialization, a random shuffling of the training data for each epoch, and repeatedly applying random dropout masks. The error function of a neural network is a highly non-convex function of the parameters with the potential for many distinct local minima (LeCun et al., 1998; Erhan et al., 2010). Depending on the seed value for the pseudo-random number generator, the network will converge to a different local minimum.

Our experiments show that these different local minima have vastly different characteristics on unseen data. For the recent NER system by Ma and Hovy (2016) we observed that, depending on the random seed value, the performance on unseen data varies between 89.99% and 91.00% F_1-score. The difference between the best and worst performance is statistically significant ($p < 10^{-4}$) using a randomization test[3]. In conclusion, whether this newly developed approach is perceived as *state-of-the-art* or as *mediocre*, largely depends on which random seed value is selected. This issue is not limited to this specific approach, but potentially applies to all approaches with non-deterministic training processes.

This large dependence on the random seed value creates several challenges when evaluating new approaches:

[1] https://arxiv.org/abs/1707.06799

[2] https://github.com/UKPLab/emnlp2017-bilstm-cnn-crf

[3] 1 Million iterations. p-value adapted using the Bonferroni correction to take the 86 tested seed values into account.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 338–348
Copenhagen, Denmark, September 7–11, 2017.

- Observing a (statistically significant) improvement through a new non-deterministic approach might not be the result of a superior approach, but the result of having a more favorable sequence of random numbers.

- Promising approaches might be rejected too early, as they fail to deliver an outperformance simply due to a less favorable sequence of random numbers.

- Reproducing results is difficult.

To study the impact of the random seed value on the performance we will focus on five linguistic sequence tagging tasks: POS-tagging, Chunking, Named Entity Recognition, Entity Recognition[4], and Event Detection. Further we will focus on Long-Short-Term-Memory (LSTM) Networks (Hochreiter and Schmidhuber, 1997b), as those demonstrated state-of-the-art performance for a wide variety of sequence tagging tasks (Ma and Hovy, 2016; Lample et al., 2016; Søgaard and Goldberg, 2016).

Fixing the random seed value would solve the issue with the reproducibility, however, there is no justification for choosing one seed value over another seed value. Hence, instead of reporting and comparing a single performance, we show that comparing score distributions can lead to new insights into the functioning of algorithms.

Our main contributions are:

1. Showing the implications of non-deterministic approaches on the evaluation of approaches and the requirement to compare score distributions instead of single performance scores.

2. Comparison of two recent, state-of-the-art systems for NER and showing that reporting a single performance score can be misleading.

3. In-depth analysis of different LSTM-architectures for five sequence tagging tasks with respect to: superior performance, stability of results, and importance of tuning parameters.

2 Background

Validating and reproducing results is an important activity in science to manifest the correctness of previous conclusions and to gain new insights into the presented approaches. Fokkens et al. (2013) show that reproducing results is not always straightforward, as factors like preprocessing (e.g. tokenization), experimental setup (e.g. splitting data), the version of components, the exact implementation of features, and the treatment of ties can have a major impact on the achieved performance and sometimes on the drawn conclusions.

For approaches with non-deterministic training procedures, like neural networks, reproducing exact results becomes even more difficult, as randomness can play a major role in the outcome of experiments. The error function of a neural network is a highly non-convex function of the parameters with the potential for many distinct local minima (LeCun et al., 1998; Erhan et al., 2010). The sequence of random numbers plays a major role to which minima the network converges during the training process. However, not all minima generalize equally well to unseen data. Erhan et al. (2010) showed for the MNIST handwritten digit recognition task that different random seeds result in largely varying performances. They noted further that with increasing depth of the neural network, the probability of finding poor local minima increases.

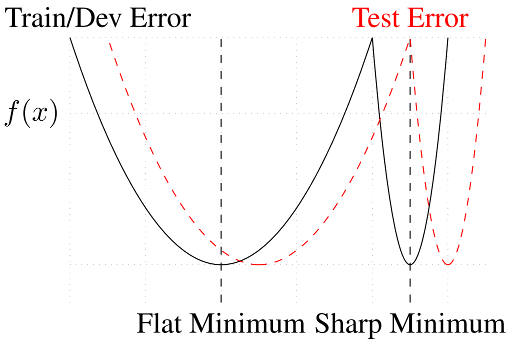


Figure 1: A conceptual sketch of flat and sharp minima from Keskar et al. (2016). The Y-axis indicates values of the error function and the X-axis the weight-space.

As (informally) defined by Hochreiter and Schmidhuber (1997a), a minimum can be flat, where the error function remains approximately constant for a large connected region in weight-space, or it can be sharp, where the error function increases rapidly in

[4] Entity Recognition labels all tokens that refer to an entity in a sentence, also generic phrases like *U.S. president*.

a small neighborhood of the minimum. A conceptual sketch is given in Figure 1. The error functions for training and testing are typically not perfectly synced, i.e. the local minima on the train or development set are not the local minima for the held-out test set. A sharp minimum usually depicts poorer generalization capabilities, as a slight variation results in a rapid increase of the error function. On the other hand, flat minima generalize better on new data (Keskar et al., 2016). Keskar et al. observe for the MNIST, TIMIT, and CIFAR dataset, that the generalization gap is not due to *over-fitting* or *over-training*, but due to different generalization capabilities of the local minima the networks converge to.

A priori it is unknown to which type of local minimum a neural network will converge. Some methods like the weight initialization (Erhan et al., 2010; Glorot and Bengio, 2010) or small-batch training (Keskar et al., 2016) help to avoid bad (e.g. sharp) minima. Nonetheless, the non-deterministic behavior of approaches must be considered when they are evaluated.

3 Impact of Randomness in the Evaluation of Neural Networks

Two recent, state-of-the-art systems for NER are proposed by Ma and Hovy (2016)[5] and by Lample et al. (2016)[6]. Lample et al. report an F_1-score of 90.94% and Ma and Hovy report an F_1-score of 91.21%. Ma and Hovy draw the conclusion that their system achieves a significant improvement over the system by Lample et al.

We re-ran both implementations multiple times, each time only changing the seed value of the random number generator. We ran the Ma and Hovy system 86 times and the Lample et al. system, due to its high computational requirement, for 41 times. The score distribution is depicted as a violin plot in Figure 2. Using a Kolmogorov-Smirnov significance test (Massey, 1951), we observe a statistically significant difference between these two distributions ($p < 0.01$). The plot reveals that the quartiles for the Lample et al. system are above those of the Ma and Hovy system. Further it reveals a smaller standard deviation σ of the F_1-scores for the Lample et al. system. Using a Brown-Forsythe test, the standard deviations are different with $p < 0.05$. Table 1 shows the minimum, the maximum, and the median performance for the test performances.

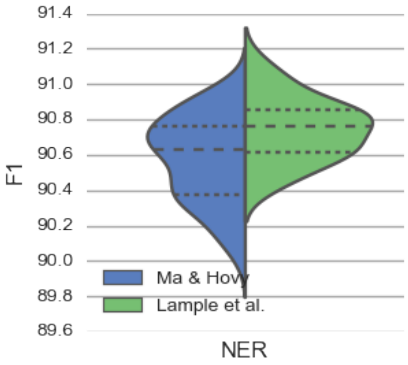


Figure 2: Distribution of scores for re-running the system by Ma and Hovy (left) and Lample et al. (right) multiple times with different seed values. Dashed lines indicate quartiles.

Based on this observation, we draw the conclusion that the system by Lample et al. outperforms the system by Ma and Hovy, as their implementation achieves a higher score distribution and shows a lower standard deviation.

In a usual setup, approaches would be compared on a development set and the run with the highest development score would be used for unseen data, i.e. be used to report the test performance. For the Lample et al. system we observe a Spearman's rank correlation between the development and the test score of $\rho = 0.229$. This indicates a weak correlation and that the performance on the development set is not a reliable indicator. Using the run with the best development score (94.44%) would yield a test performance of mere 90.31%. Using the second best run on development set (94.28%), would yield state-of-the-art performance with 91.00%. This difference is statistically significant ($p < 0.002$). In conclusion, a development set will not necessarily solve the issue with bad local minima.

The main difference between these two approaches is in the generation of character-based representations: Ma and Hovy uses a Convolutional Neural Network (CNN) (LeCun et al., 1989), while Lample et al. uses an LSTM-network. As our experiments in section 6.4 show, both approaches perform comparably if all other parameters were kept the same. Further, we could only observe a

[5] https://github.com/XuezheMax/LasagneNLP

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

System	Reported F_1	# Seed values	Min. F_1	Median F_1	Max. F_1	σ
Ma and Hovy	91.21%	86	89.99%	90.64%	91.00%	0.00241
Lample et al.	90.94%	41	90.19%	90.81%	91.14%	0.00176

Table 1: The system by Ma and Hovy (2016) and Lample et al. (2016) were run multiple times with different seed values.

Task	Dataset	# Configs	Median Difference	95th percentile	Max. Difference
POS	Penn Treebank	269	0.17%	0.78%	1.55%
Chunking	CoNLL 2000	385	0.17%	0.50%	0.81%
NER	CoNLL 2003	406	0.38%	1.08%	2.59%
Entities	ACE 2005	405	0.72%	2.10%	8.23%
Events	TempEval 3	365	0.43%	1.23%	1.73%

Table 2: The table depicts the median, the 95th percentile and the maximum difference between networks with the same hyperparameters but different random seed values.

statistically significant improvement for the tasks POS, Chunking and Event Detection. For NER and Entity Recognition, the difference was statistically not significant given the number of tested hyperparameters.

In the next step, we evaluated the impact of the random seed value for the five sequence tagging tasks described in section 4. We sampled randomly 1830 different configurations, for example different numbers of recurrent units, and ran the network twice, each time with a different seed value. The results are depicted in Table 2.

The largest difference was observed for the ACE 2005 Entities dataset: Using one seed value, the network achieved an F_1 performance of 82.5% while using another seed value, the network achieved a performance of only 74.3%. Even though this is a rare extreme case, the median difference between different weight initializations is still large. For example for the CoNLL 2003 NER dataset, the median difference is at 0.38% and the 95th percentile is at 1.08%.

In conclusion, if the fact of different local minima is not taken care of and single performance scores are compared, there is a high chance of drawing false conclusions and either rejecting promising approaches or selecting weaker approaches due to a more or less favorable sequence of random numbers.

4 Experimental Setup

In order to find LSTM-network architectures that perform robustly on different tasks, we selected five classical NLP tasks as benchmark tasks: Part-of-Speech tagging (*POS*), Chunking, Named Entity Recognition (*NER*), Entity Recognition (*Entities*) and Event Detection (*Events*).

For Part-of-Speech tagging, we use the benchmark setup described by Toutanova et al. (2003). Using the full training set for POS tagging would hinder our ability to detect design choices that are consistently better than others. The error rate for this dataset is approximately 3% (Marcus et al., 1993), making all improvements above 97% accuracy likely the result of chance. A 97.24% accuracy was achieved by Toutanova et al. (2003). Hence, we reduced the training set size from over 38.000 sentences to the first 500 sentences. This decreased the accuracy to about 95%.

For Chunking, we use the CoNLL 2000 shared task setup. For Named Entity Recognition (NER), we use the CoNLL 2003 setup. The ACE 2005 entity recognition task annotated not only named entities, but all words referring to an entity, e.g. the phrase *U.S. president*. We use the same data split as Li et al. (2013). For the Event Detection task, we use the TempEval3 Task B setup. There, the smallest extent of text, usually a single word, that expresses the occurrence of an event, is annotated.

For the POS-task, we report accuracy and for the other tasks we report the F_1-score.

4.1 Model

We use a BiLSTM-network for sequence tagging as described in (Huang et al., 2015; Ma and Hovy, 2016; Lample et al., 2016). To be able to evaluate a large number of different network configurations, we optimized our implementation for efficiency, reducing by a factor of 6 the time required per epoch compared to Ma and Hovy (2016).

4.2 Evaluated Parameters

We evaluate the following design choices and hyperparameters:

Pre-trained Word Embeddings. We evaluate the Google News embeddings (*G. News*)[7] from Mikolov et al. (2013), the Bag of Words (*Le. BoW*) as well as the dependency based embeddings (*Le. Dep.*)[8] by Levy and Goldberg (2014), three different GloVe embeddings[9] from Pennington et al. (2014) trained either on Wikipedia 2014 + Gigaword 5 (*GloVe1* with 100 dimensions and *GloVe2* with 300 dimensions) or on Common Crawl (*GloVe3*), and the Komninos and Manandhar (2016) embeddings (*Komn.*)[10]. We also evaluate the approach of Bojanowski et al. (2016) (*FastText*), which trains embeddings for n-grams with length 3 to 6. The embedding for a word is defined as the sum of the embeddings of the n-grams.

Character Representation. We evaluate the approaches of Ma and Hovy (2016) using Convolutional Neural Networks (CNN) as well as the approach of Lample et al. (2016) using LSTM-networks to derive character-based representations.

Optimizer. Besides Stochastic Gradient Descent (*SGD*), we evaluate *Adagrad* (Duchi et al., 2011), *Adadelta* (Zeiler, 2012), *RMSProp* (Hinton, 2012), *Adam* (Kingma and Ba, 2014), and *Nadam* (Dozat, 2015), an Adam variant that incorporates Nesterov momentum (Nesterov, 1983) as optimizers.

Gradient Clipping and Normalization. Two common strategies to deal with the exploding gradient problem are *gradient clipping* (Mikolov, 2012) and *gradient normalization* (Pascanu et al., 2013). Gradient clipping involves clipping the gradient's components element-wise if it exceeds a defined threshold. Gradient normalization has a better theoretical justification and rescales the gradient whenever the norm goes over a threshold.

Tagging schemes. We evaluate the *BIO* and *IOBES* schemes for tagging segments.

Dropout. We compare *no dropout*, *naive dropout*, and *variational dropout* (Gal and Ghahramani, 2016). Naive dropout applies a new dropout mask at every time step of the LSTM-layers. Variational dropout applies the same dropout mask for all time steps in the same sentence. Further, it applies dropout to the recurrent units. We evaluate the dropout rates $\{0.05, 0.1, 0.25, 0.5\}$.

Classifier. We evaluate a Softmax classifier as well as a CRF classifier as the last layer of the network.

Number of LSTM-layers. We evaluated *1*, *2*, and *3* stacked BiLSTM-layers.

Number of recurrent units. For each LSTM-layer, we selected independently a number of recurrent units from the set $\{25, 50, 75, 100, 125\}$.

Mini-batch sizes. We evaluate the mini-batch sizes 1, 8, 16, 32, and 64.

5 Robust Model Evaluation

We have shown in section 3 that re-running non-deterministic approaches multiple times and comparing score distributions is essential to draw correct conclusions. However, to truly understand the capabilities of an approach, it is interesting to test the approach with different sets of hyperparameters for the complete network.

Training and tuning a neural network can be time consuming, sometimes taking multiple days to train a single instance of a network. A priori it is hard to know which hyperparameters will yield the best performance and the selection of the parameters often makes the difference between *mediocre* and *state-of-the-art* performance (Hutter et al., 2014). If an approach yields good performance only for a narrow set of parameters, it might be difficult to adapt the approach to new tasks, new domains

[7] https://code.google.com/archive/p/word2vec/

[8] https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/

[9] http://nlp.stanford.edu/projects/glove/

[10] https://www.cs.york.ac.uk/nlp/extvec/

or new languages, as a large range of possible parameters must be evaluated, each time requiring a significant amount of training time. Hence it is desirable, that the approach yields stable results for a wide range of parameters.

In order to find approaches that result in high performance and are robust against the remaining parameters, we decided to randomly sample several hundred network configurations from the set described in section 4.2. For each sampled configuration, we compare different options, e.g. different options for the last layer of the network. For example, we sampled in total 975 configurations and each configuration was trained with a Softmax classifier as well as with a CRF classifier, totaling to 1950 trained networks.

Dataset	# Configs	Softmax	CRF
POS $\Delta Acc.$	111	18.9% -0.20%	**81.1%**
Chunking ΔF_1	229	4.8% -0.38%	**95.2%**
NER ΔF_1	232	9.5% -0.66%	**90.5%**
Entities ΔF_1	210	13.3% -0.84%	**86.7%**
Events ΔF_1	202	**61.9%**	38.1% -0.15%
Average		21.7%	**78.3%**

Table 3: Percentages of configurations where Softmax or CRF classifiers demonstrated a higher test performance.

Our results are presented in Table 3. The table shows that for the NER task 232 configurations were sampled randomly and for 210 of the 232 configurations (90.5%), the CRF setup achieved a better test performance than the setup with a Softmax classifier. To measure the difference between these two options, we compute the median of the absolute differences: Let S_i be the test performance (F_1-measure) for the Softmax setup for configuration i and C_i the test performance for the CRF setup. We then compute $\Delta F_1 = \text{median}(S_1 - C_1, S_2 - C_2, \ldots, S_{232} - C_{232})$. For the NER task, the median difference was $\Delta F_1 = -0.66\%$, i.e. the setup with a Softmax classifier achieved on average an F_1-score of 0.66 percentage points below that of the CRF setup.

We also evaluated the standard deviation of the F_1-scores to detect approaches that are less dependent on the remaining hyperparameters and the random number generator. The standard deviation σ for the CRF-classifier is with 0.0060 significantly lower ($p < 10^{-3}$ using Brown-Forsythe test) than for the Softmax classifier with $\sigma = 0.0082$.

6 Results

This section highlights our main insights in the evaluation of different design choices for BiLSTM architectures. We limit the number of results we present for reasons of brevity. Detailed information can be found in (Reimers and Gurevych, 2017).[11]

6.1 Classifier

Table 3 shows a comparison between using a Softmax classifier as a last layer and using a CRF classifier. The BiLSTM-CRF architecture by Huang et al. (2015) achieves a better performance on 4 out of 5 tasks. For the NER task it further achieves a 27% lower standard deviation (statistically significant with $p < 10^{-3}$), indicating that it is less sensitive to the remaining configuration of the network.

The CRF classifier only fails for the Event Detection task. This task has nearly no dependency between tags, as often only a single token is annotated as an event trigger in a sentence.

We studied the differences between these two classifiers in terms of number of LSTM-layers. As Figure 3 shows, a Softmax classifier profits from a deep LSTM-network with multiple stacked layers. On the other hand, if a CRF classifier is used, the effect of additional LSTM-layers is much smaller.

6.2 Optimizer

We evaluated six optimizers with the suggested default configuration from their respective papers. We observed that SGD is quite sensitive towards the selection of the learning rate and it failed in many instances to converge. For the optimizers SGD, Adagrad and Adadelta we observed a large standard deviation in terms of test performance,

[11] https://public.ukp.informatik.tu-darmstadt.de/reimers/Optimal_Hyperparameters_for_Deep_LSTM-Networks.pdf

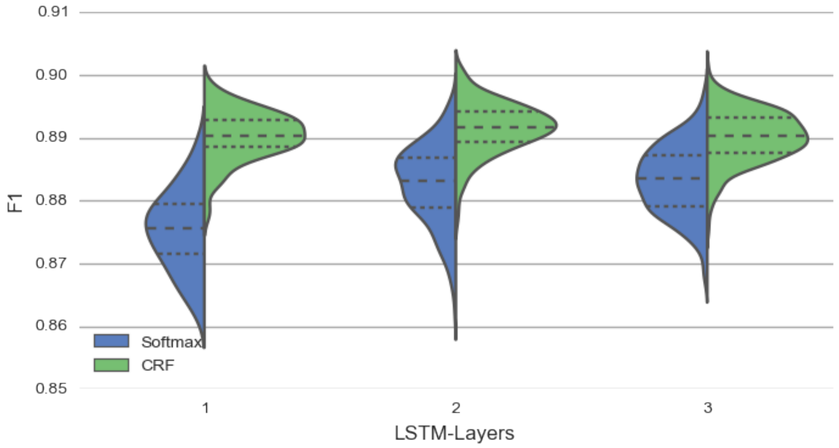


Figure 3: Difference between Softmax and CRF classifier for different number of BiLSTM-layers for the CoNLL 2003 NER dataset.

which was for the NER task at 0.1328 for SGD, 0.0139 for Adagrad, and 0.0138 for Adadelta. The optimizers RMSProp, Adam, and Nadam on the other hand produced much more stable results. Not only were the medians for these three optimizers higher, but also the standard deviation was with 0.0096, 0.0091, and 0.0092 roughly 35% smaller in comparison to Adagrad. A large standard deviation indicates that the optimizer is sensitive to the hyperparameters as well as to the random initialization and bears the risk that the optimizer produces subpar results.

The best result was achieved by Nadam. For 453 out of 882 configurations (51.4%), it yielded the highest performance out of the six tested optimizers. For the NER task, it produced on average a 0.82 percentage points better performance than Adagrad.

Besides test performance, the convergence speed is important in order to reduce training time. Here, Nadam had the best convergence speed. For the NER dataset, Nadam converged on average after 9 epochs, whereas SGD required 42 epochs.

6.3 Word Embeddings

The pre-trained word embeddings had a large impact on the performance as shown in Table 4. The embeddings by Komninos and Manandhar (2016) resulted in the best performance for the POS, the Entities and the Events task. For the Chunking task, the dependency-based embeddings of Levy and Goldberg (2014) are slightly ahead of the Komninos embeddings, the significance level is at $p = 0.025$. For NER, the GloVe embeddings trained on common crawl perform on par with the Komninos embeddings ($p = 0.391$).

We observe that the underlying word embeddings have a large impact on the performance for all tasks. Well suited word embeddings are especially critical for datasets with small training sets. For the POS task we observe a median difference of 4.97% between the Komninos embeddings and the GloVe2 embeddings.

Note we only evaluated the pre-trained embeddings provided by different authors, but not the underlying algorithms to generate these embeddings. The quality of word embeddings depends on many factors, including the size, the quality, and the preprocessing of the data corpus. As the corpora are not comparable, our results do not allow concluding that one approach is superior for generating word embeddings.

6.4 Character Representation

We evaluate the approaches of Ma and Hovy (2016) using Convolutional Neural Networks (CNN) as well as the approach of Lample et al. (2016) using LSTM-networks to derive character-based representations.

Table 5 shows that character-based representations yield a statistically significant difference only for the POS, the Chunking, and the Events task. For NER and Entity Recognition, the difference to not using a character-based representation is not significant ($p > 0.01$).

Dataset	Le. Dep.	Le. BoW	GloVe1	GloVe2	GloVe3	Komn.	G. News	FastText
POS $\Delta Acc.$	6.5% -0.39%	0.0% -2.52%	0.0% -4.14%	0.0% -4.97%	0.0% -2.60%	**93.5%**	0.0% -1.95%	0.0% -2.28%
Chunking ΔF_1	**60.8%**	0.0% -0.52%	0.0% -1.09%	0.0% -1.50%	0.0% -0.93%	37.1% -0.10%	2.1% -0.48%	0.0% -0.75%
NER ΔF_1	4.5% -0.85%	0.0% -1.17%	22.7% -0.15%	0.0% -0.73%	**43.6%**	27.3% -0.08%	1.8% -0.75%	0.0% -0.89%
Entities ΔF_1	4.2% -0.92%	7.6% -0.89%	0.8% -1.50%	0.0% -2.24%	6.7% -0.80%	**57.1%**	21.8% -0.33%	1.7% -1.13%
Events ΔF_1	12.9% -0.55%	4.8% -0.78%	0.0% -2.77%	0.0% -3.55%	0.0% -2.55%	**71.8%**	9.7% -0.67%	0.8% -1.36%
Average	17.8%	2.5%	4.7%	0.0%	10.1%	**57.4%**	7.1%	0.5%

Table 4: Randomly sampled configurations were evaluated with 8 possible word embeddings. 108 configurations were sampled for POS, 97 for Chunking, 110 for NER, 119 for Entities, and 124 for Events.

The difference between the CNN approach by Ma and Hovy (2016) and the LSTM approach by Lample et al. (2016) to derive a character-based representations is statistically insignificant for all tasks. This is quite surprising, as both approaches have fundamentally different properties: The CNN approach from Ma and Hovy (2016) takes only trigrams into account. It is also position independent, i.e. the network will not be able to distinguish between trigrams at the beginning, in the middle, or at the end of a word, which can be crucial information for some tasks. The BiLSTM approach from Lample et al. (2016) takes all characters of the word into account. Further, it is position aware, i.e. it can distinguish between characters at the start and at the end of the word. Intuitively, one would think that the LSTM approach by Lample et al. would be superior.

Task	No	CNN	LSTM
POS $\Delta Acc.$	4.9% -0.90%	**58.2%**	36.9% -0.05%
Chunking ΔF_1	13.3% -0.20%	43.2% -0.00%	**43.6%**
NER ΔF_1	27.2% -0.11%	**36.4%**	36.4% -0.01%
Entities ΔF_1	26.8% -0.07%	36.0% 0.00%	**37.3%**
Events ΔF_1	20.5% -0.44%	35.6% -0.04%	**43.8%**
Average	18.5%	**41.9%**	39.6%

Table 5: Comparison of not using character-based representations and using CNNs (Ma and Hovy, 2016) or LSTMs (Lample et al., 2016) to derive character-based representations. 225 configurations were sampled for POS, 241 for Chunking, 217 for NER, 228 for Entities, and 219 for Events.

6.5 Gradient Clipping and Normalization

For *gradient clipping* (Mikolov, 2012) we couldn't observe any improvement for the thresholds of 1, 3, 5, and 10 for any of the five tasks.

Gradient normalization has a better theoretical justification (Pascanu et al., 2013) and we can confirm with our experiments that it performs better. Not normalizing the gradient was the best option only for 5.6% of the 492 evaluated configurations (under null-hypothesis we would expect 20%). Which threshold to choose, as long as it is not too small or too large, is of lower importance. In most cases, a threshold of 1 was the best option (30.5% of the cases).

We observed a large performance increase compared to not normalizing the gradient. The median increase was between 0.29 percentage points F_1-score for the Chunking task and 0.82 percentage points for the POS task.

6.6 Dropout

Dropout is a popular method to deal with overfitting for neural networks (Srivastava et al., 2014). We could observe that *variational dropout* (Gal and Ghahramani, 2016) clearly outperforms *naive dropout* and not using dropout. It was the best option in 83.5% of the 479 evaluated configurations. The median performance increase in comparison to not using dropout was between 0.31 percentage points for the POS-task and 1.98 for the Entities task. We also observed a large improvement in comparison to naive dropout between 0.19 percentage points for the POS task and 1.32 percentage points for the Entities task. Variational dropout showed the smallest standard deviation, indicating that it is less dependent on the remaining hyperparameters and the random number sequence.

We further evaluated whether variational dropout should be applied to the output units of the LSTM-network, to the recurrent units, or to both. We observed that applying dropout to both dimensions gave in most cases (62.6%) the best results. The median performance increase was between 0.05 percentage points and 0.82 percentage points.

6.7 Further Evaluated Parameters

The tagging schemes `BIO` and `IOBES` performed on par for 4 out of 5 tasks. For the Entities task, the `BIO` scheme significantly outperformed the `IOBES` scheme for 88.7% of the tested configurations. The median difference was $\Delta F_1 = -1.01\%$.

For the evaluated tasks, 2 stacked LSTM-layers achieved the best performance. For the POS-tagging task, 1 and 2 layers performed on par. For flat networks with a single LSTM-layer, around 150 recurrent units yielded the best performance. For networks with 2 or 3 layers, around 100 recurrent units per network yielded the best performance. However, the impact of the number of recurrent units was extremely small.

For tasks with small training sets, smaller mini-batch sizes of 1 up to 16 appears to be a good choice. For larger training sets sizes of 8 - 32 appears to be a good choice. Mini-batch sizes of 64 usually performed worst.

7 Conclusion

In this paper, we demonstrated that the sequence of random numbers has a *statistically significant* impact on the test performance and that wrong conclusions can be made if performance scores based on single runs are compared. We demonstrated this for the two recent state-of-the-art NER systems by Ma and Hovy (2016) and Lample et al. (2016). Based on the published performance scores, Ma and Hovy draw the conclusion of a significant improvement over the approach of Lample et al. Re-executing the provided implementations with different seed values however showed that the implementation of Lample et al. results in a superior score distribution generalizing better to unseen data.

Comparing score distributions reduces the risk of rejecting promising approaches or falsely accepting weaker approaches. Further it can lead to new insights on the properties of an approach. We demonstrated this for ten design choices and hyperparameters of LSTM-networks for five tasks.

By studying the standard deviation of scores, we estimated the dependence on hyperparameters and on the random seed value for different approaches. We showed that SGD, Adagrad and Adadelta have a higher dependence than RMSProp, Adam or Nadam. We have shown that variational dropout also reduces the dependence on the hyperparameters and on the random seed value. As future work, we will investigate if those methods are either less dependent on the hyperparameters or are less dependent on the random seed value, e.g. if they avoid converging to bad local minima.

By testing a large number of configurations, we showed that some choices consistently lead to superior performance and are less dependent on the remaining configuration of the network. Thus, there is a good chance that these configurations require less tuning when applied to new tasks or domains.

Acknowledgements

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1. Calculations for this research were conducted on the Lichtenberg high performance computer of the TU Darmstadt.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *CoRR*, abs/1603.06042.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*.

Timothy Dozat. 2015. Incorporating Nesterov Momentum into Adam.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.*, 12:2121–2159.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why Does Unsupervised Pre-training Help Deep Learning? *Journal of Machine Learning Research*, 11:625–660.

Antske Fokkens, Marieke van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. 2013. Offspring from Reproduction Problems: What Replication Failure Teaches Us. In *ACL (1)*, pages 1691–1701. The Association for Computer Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1019–1027.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*.

Geoffrey Hinton. 2012. Neural Networks for Machine Learning - Lecture 6a - Overview of mini-batch gradient descent.

Sepp Hochreiter and Jürgen Schmidhuber. 1997a. Flat Minima. *Neural Computation*, 9(1):1–42.

Sepp Hochreiter and Jürgen Schmidhuber. 1997b. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *CoRR*, abs/1508.01991.

Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. 2014. An Efficient Approach for Assessing Hyperparameter Importance. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages I–754–I–762. JMLR.org.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *CoRR*, abs/1609.04836.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency based embeddings for sentence classification tasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1490–1500, San Diego, California. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *CoRR*, abs/1603.01360.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1989. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551.

Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. 1998. Efficient BackProp. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pages 9–50, London, UK, UK. Springer-Verlag.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 302–308.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint Event Extraction via Structured Prediction with Global Features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82, Sofia, Bulgaria. Association for Computational Linguistics.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. *CoRR*, abs/1603.01354.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Comput. Linguist.*, 19(2):313–330.

Frank J. Massey. 1951. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78.

Tomáš Mikolov. 2012. *Statistical language models based on neural networks*. Ph.D. thesis, Brno University of Technology.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781.

Yurii Nesterov. 1983. A method of solving a convex programming problem with convergence rate O(1/sqr(k)). *Soviet Mathematics Doklady*, 27:372–376.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the Difficulty of Training Recurrent Neural Networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1310–III–1318. JMLR.org.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Nils Reimers and Iryna Gurevych. 2017. Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. *arXiv preprint arXiv:1707.06799*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235, Berlin, Germany. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL 2003, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Learning What's Easy: Fully Differentiable Neural Easy-First Taggers

André F. T. Martins
Unbabel
& Instituto de Telecomunicações
Lisbon, Portugal
andre.martins@unbabel.com

Julia Kreutzer[*]
Computational Linguistics
Heidelberg University, Germany
kreutzer@cl.uni-heidelberg.de

Abstract

We introduce a novel neural easy-first decoder that learns to solve sequence tagging tasks in a flexible order. In contrast to previous easy-first decoders, our models are end-to-end differentiable. The decoder iteratively updates a "sketch" of the predictions over the sequence. At its core is an attention mechanism that controls which parts of the input are strategically the best to process next. We present a new constrained softmax transformation that ensures the same cumulative attention to every word, and show how to efficiently evaluate and backpropagate over it. Our models compare favourably to BILSTM taggers on three sequence tagging tasks.

1 Introduction

In the last years, neural models have led to major advances in several structured NLP problems, including sequence tagging (Plank et al., 2016; Lample et al., 2016), sequence-to-sequence prediction (Sutskever et al., 2014), and sequence-to-tree (Dyer et al., 2015). Part of the success comes from clever architectures such as (bidirectional) long-short term memories (BILSTMs; Hochreiter and Schmidhuber (1997); Graves et al. (2005)) and attention mechanisms (Bahdanau et al., 2015), which are able to select the pieces of context relevant for prediction.

A noticeable aspect about many of the systems above is that they typically decode from left to right, greedily or with a narrow beam. While this is computationally convenient and reminiscent of the way humans process spoken language, the combination of unidirectional decoding and greediness leads to error propagation and suboptimal classification performance. This can partly be mitigated by globally normalized models (Andor et al., 2016) and imitation learning (Daumé et al., 2009; Ross et al., 2011; Bengio et al., 2015), however these techniques still have a left-to-right bias.

Easy-first decoders (Tsuruoka and Tsujii, 2005; Goldberg and Elhadad, 2010, §2) are an interesting alternative: instead of a fixed decoding order, these methods schedule their own actions by prefering "easier" decisions over more difficult ones. A disadvantage is that these models are harder to learn, due to the factorial number of orderings leading to correct predictions. Usually, gradients are *not* backpropagated over this combinatorial latent space (Kiperwasser and Goldberg, 2016a), or a separate model is used to determine the easiest next move (Clark and Manning, 2016).

In this paper, we develop novel, **fully differentiable, neural easy-first sequence taggers** (§3). Instead of taking discrete actions, our decoders use an attention mechanism to decide (in a soft manner) which word to focus on for the next tagging decision. Our models are able to learn their own sense of "easiness": the words receiving focus may not be the ones the model is most confident about, but the best to avoid error propagation in the long run. To make sure that all words receive the same cumulative attention, we further contribute with a new **constrained softmax transformation** (§4). This transformation extends the softmax by permitting upper bound constraints on the amount of probability a word can receive. We show how to evaluate this transformation and backpropagate its gradients.

We run experiments in three sequence tagging tasks: multilingual part-of-speech (POS) tagging, named entity recognition (NER), and word-level quality estimation (§5). We complement our findings with a visual analysis of the attention distribu-

[*] This research was partially carried out during an internship at Unbabel.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 349–362
Copenhagen, Denmark, September 7–11, 2017.

Algorithm 1 Easy-First Sequence Tagging

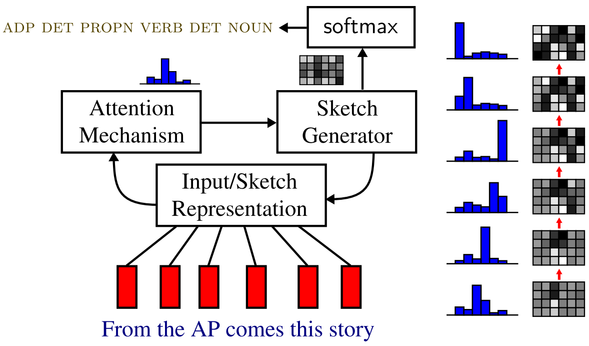
```
Input: input sequence $x_{1:L}$
Output: tagged sequence $\widehat{y}_{1:L}$
1: initialize $\mathcal{B} = \mathcal{S} = \varnothing$
2: while $\mathcal{B} \neq \{1, \dots, L\}$ do
3:     for each non-covered position $i \notin \mathcal{B}$ do
4:         compute scores $f(i, y_i; x_{1:L}, \mathcal{S}),\ \forall y_i$
5:     end for
6:     $(j, \widehat{y}_j) = \mathsf{argmax}_{i,y_i}\ f(i, y_i; x_{1:L}, \mathcal{S})$
7:     $\mathcal{S} \leftarrow \mathcal{S} \cup \{(j, \widehat{y}_j)\}, \mathcal{B} \leftarrow \mathcal{B} \cup \{j\}$
8: end while
```

tions produced by the decoder, to help understand what tagging decisions the model finds the easiest.

2 Easy-First Decoders

The idea behind easy-first decoding is to perform "easier" and less risky decisions before committing to more difficult ones (Tsuruoka and Tsujii, 2005; Goldberg and Elhadad, 2010; Ma et al., 2013). Alg. 1 shows the overall procedure for a sequence tagging problem (the idea carries out to other structured problems). Let $x_{1:L}$ be an input sequence (e.g. words in a sentence) and $y_{1:L}$ be the corresponding tag sequence (e.g. their POS tags). The algorithm assigns tags one position i at the time, maintaining a set $\mathcal{B}$ of **covered positions**. It also maintains a set $\mathcal{S}$ of pairs $(i, \widehat{y}_i)$, storing the tags that have already been predicted at those positions. We can regard this set as a **sketch** of the output sequence, built incrementally while the algorithm is executed. At each time step, the model computes a score $f(i, y_i; x_{1:L}, \mathcal{S})$ for each position $i \notin \mathcal{B}$ and each candidate tag y_i, taking into account the current "sketch" $\mathcal{S}$, which provides useful contextual information. The "easiest" position and the corresponding tag are then jointly obtained by maximizing this score (line 6). The algorithm terminates when all positions are covered.

Previous work has trained easy-first systems with variants of the perceptron algorithm (Goldberg and Elhadad, 2010; Ma et al., 2013) or with a gradient-based method (Kiperwasser and Goldberg, 2016a)—but without backpropagating information about the best ordering chosen by the algorithm (only tag mistakes). In fact, doing so directly would be hard, since the space of possible orderings is combinatorial—the argmax in line 6 is not continuous, let alone differentiable. In the next section, we introduce a fully differentiable easy-first system that sidesteps this problem by working with a "continuous" space of actions.



ADP DET PROPN VERB DET NOUN
softmax
Attention Mechanism
Sketch Generator
Input/Sketch Representation
From the AP comes this story

Figure 1: A neural easy-first system applied to a POS tagging problem. Given the current input/sketch representation, an attention mechanism decides where to focus (see bar plot) and is used to generate the next sketch. Right: A sequence of sketches $(\mathbf{S}^n)_{n=1}^N$ generated along the way.

3 Neural Easy-First Sequence Taggers

Let $\Delta^{L-1} := \{\boldsymbol{\alpha} \in \mathbb{R}^L \,|\, \mathbf{1}^\top \boldsymbol{\alpha} = 1, \boldsymbol{\alpha} \geq \mathbf{0}\}$ be the probability simplex. Our **neural easy-first decoders** depart from Alg. 1 in the following key points:

- Instead of picking the position with the largest score at each step (line 6 in Alg. 1), we compute a (continuous) attention distribution $\boldsymbol{\alpha} \in \Delta^{L-1}$ over word positions.
- Instead of a set of covered positions $\mathcal{B}$, we maintain a (continuous) **cumulative attention vector $\boldsymbol{\beta}$** $\in \mathbb{R}^L$ (ideally in $[0,1]^L$) over the L positions in the sequence.
- The sketch set $\mathcal{S}$ is replaced by a **sketch matrix $\mathbf{S}$** $\in \mathbb{R}^{D_s \times L}$, whose columns are D_s-dimensional vector representations of the output labels to be predicted.

The high-level procedure is shown in Figure 1. We describe two models that implement this procedure: a **single-state model** and a **full-state model**. They differ in the way they update the sketch matrix: the single-state model applies a rank-one update, while the full-state model does a full update.

3.1 Single-State Model (NEF-S)

Let $\mathsf{concat}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_K) \in \mathbb{R}^{\sum_{k=1}^K D_k}$ be the concatenation of the vectors $\boldsymbol{x}_k \in \mathbb{R}^{D_k}$. We use the shorthand $\mathsf{affine}(\boldsymbol{x}) := \mathbf{W}\boldsymbol{x} + \boldsymbol{b}$ to denote an affine transformation of $\boldsymbol{x}$, where $\mathbf{W}$ is a weight matrix and $\boldsymbol{b}$ is a bias vector.

Alg. 2 shows the overall procedure. We start by encoding the input sequence $x_{1:L}$ as a matrix

Algorithm 2 Neural Easy-First Sequence Tagging

```
Input: input sequence $x_{1:L}$, sketch steps $N$
Output: tagged sequence $\widehat{y}_{1:L}$
 1: initialize $\boldsymbol{\beta}^0 = \mathbf{0}$ and $\boldsymbol{s}_i^0 = \mathbf{0}$, $\forall i \in [L]$
 2: encode sequence as $[\boldsymbol{h}_1, \ldots, \boldsymbol{h}_L]$
 3: for $n = 1, 2, \ldots, N$ do
 4:     for each position $i \in [L]$ do
 5:         compute $\boldsymbol{c}_i^n$ and $z_i^n$ (Eqs. 1–2)
 6:     end for
 7:     compute attention $\boldsymbol{\alpha}^n = \rho(\boldsymbol{z}^n; \boldsymbol{\beta}^{n-1})$
 8:     for each position $i \in [L]$ do
 9:         refine sketch $\boldsymbol{s}_i^n$ from $\alpha_i^n$ and $\boldsymbol{c}_i^n$, via Eq. 4 (single-state) or Eq. 7 (full-state)
10:     end for
11:     $\boldsymbol{\beta}^n = \boldsymbol{\beta}^{n-1} + \boldsymbol{\alpha}^n$
12: end for
13: for each position $i \in [L]$ do
14:     $\boldsymbol{p}_i = \mathsf{softmax}(\mathsf{affine}(\mathsf{concat}(\boldsymbol{h}_i, \boldsymbol{s}_i^N)))$
15:     predict $\widehat{y}_i = \mathsf{argmax}_{y_i}\, p_i(y_i)$
16: end for
```

$\mathbf{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_L] \in \mathbb{R}^{D_h \times L}$ (line 2). Our model is completely agnostic about this encoding step. In our experiments, we compute $\mathbf{H}$ by composing a lookup embedding layer with a BILSTM (Hochreiter and Schmidhuber, 1997; Graves et al., 2005), as this strategy was successful in similar structured prediction tasks. However, other choices are possible, for example using convolutional layers.

As stated above, our algorithm maintains a cumulative attention vector $\boldsymbol{\beta} \in \mathbb{R}^L$ and a sketch matrix $\mathbf{S} \in \mathbb{R}^{D_s \times L}$, both with all entries initialized to zero. It then performs N sketching steps, which progressively refine this sketch matrix, producing versions $\mathbf{S}^1, \ldots, \mathbf{S}^N$. At the nth step, the following operations are performed:

Input-Sketch Contextual Representation. For each word $i \in [L]$, we compute a state $\boldsymbol{c}_i^n$ summarizing the surrounding local information about other words and sketches. We use a simple vector concatenation over a w-wide context window:

$$\boldsymbol{c}_i^n = \mathsf{concat}(\boldsymbol{h}_{i-w}, \ldots, \boldsymbol{h}_{i+w}, \boldsymbol{s}_{i-w}^{n-1}, \ldots, \boldsymbol{s}_{i+w}^{n-1}), \tag{1}$$

where we denote by $\boldsymbol{s}_j^{n-1}$ the jth column of $\mathbf{S}^{n-1}$. The intuition is that the current sketches provide valuable information about the neighboring words' predictions that can influence the prediction for the ith word. In the vanilla easy-first algorithm, this was assumed in the score computation (line 4 of Alg. 1).

Attention Mechanism. We then use an attention mechanism to decide what is the "best" word to focus on next. This is done in a similar way as the feedforward attention proposed by Bahdanau et al. (2015). We first compute a score for each word $i \in [L]$ based on its contextual representation,

$$z_i^n = \boldsymbol{v}^\top \mathsf{tanh}(\mathsf{affine}(\boldsymbol{c}_i^n)), \tag{2}$$

where $\boldsymbol{v} \in \mathbb{R}^{D_z}$ is a model parameter. Then, we aggregate these scores in a vector $\boldsymbol{z}^n \in \mathbb{R}^L$ and apply a transformation ρ to map them to a probability distribution $\boldsymbol{\alpha}^n \in \Delta^{L-1}$ (optionally taking into account the past cumulative attention $\boldsymbol{\beta}^{n-1}$):

$$\boldsymbol{\alpha}^n = \rho(\boldsymbol{z}^n; \boldsymbol{\beta}^{n-1}). \tag{3}$$

The "standard" choice for ρ is the softmax transformation. However, in this work, we consider other possible transformations (to be described in §4). After this, the cumulative attention is updated via $\boldsymbol{\beta}^n = \boldsymbol{\beta}^{n-1} + \boldsymbol{\alpha}^n$.

Sketch Generation. Now that we have a distribution $\boldsymbol{\alpha}^n \in \Delta^{L-1}$ over word positions, it remains to generate a sketch for those words. We first compute a **single-state** vector representation of the entire sentence $\bar{\boldsymbol{c}}^n = \sum_{i=1}^L \alpha_i^n \boldsymbol{c}_i^n$ as the weighted average of the word states defined in Eq. 1. Then, we update each column of the sketch matrix as:[1]

$$\boldsymbol{s}_i^n = \boldsymbol{s}_i^{n-1} + \alpha_i^n \cdot \mathsf{tanh}(\mathsf{affine}(\bar{\boldsymbol{c}}^n)), \;\; \forall i \in [L]. \tag{4}$$

The intuition for this update is the following: in the extreme case where the attention distribution is peaked on a single word (say, the kth word, $\boldsymbol{\alpha}^n = \boldsymbol{e}_k$), we obtain $\bar{\boldsymbol{c}}^n = \boldsymbol{c}_k^n$ and the sketch update only affects that word, i.e.,

$$\boldsymbol{s}_i^n = \begin{cases} \boldsymbol{s}_i^{n-1} + \mathsf{tanh}(\mathsf{affine}(\boldsymbol{c}_k^n)) & \text{if } i = k \\ \boldsymbol{s}_i^{n-1} & \text{if } i \neq k. \end{cases} \tag{5}$$

This is similar to the sketch update in the original easy-first algorithm (line 7 in Alg. 1).

The three operations above are repeated N times (or "sketch steps"). The standard choice is $N = L$ (one step per word), which mimics the vanilla easy-first algorithm. However, it is possible to have fewer steps (or more, if we want the decoder to be able to "self-correct"). After completing the N sketch steps, we obtain the final sketch matrix $\mathbf{S}^N = [\boldsymbol{s}_1^N, \ldots, \boldsymbol{s}_L^N]$. Then, we compute a tag probability for every word as follows:[2]

$$\boldsymbol{p}_i = \mathsf{softmax}(\mathsf{affine}(\mathsf{concat}(\boldsymbol{h}_i, \boldsymbol{s}_i^N))). \tag{6}$$

[1] Note that this is a rank-one update, as it can be written in matrix notation as $\mathbf{S}^n = \mathbf{S}^{n-1} + \mathsf{tanh}(\mathsf{affine}(\bar{\boldsymbol{c}}^n)) \cdot \boldsymbol{\alpha}^{n\top}$.

[2] We found the concatenation with $\boldsymbol{h}_i$ in Eq. 6 beneficial to transfer input information directly to the output layer, avoiding the need of flowing this information through the sketches.

In §5, we compare this to a BILSTM tagger, which predicts according to $\boldsymbol{p}_i = \mathsf{softmax}(\mathsf{affine}(\boldsymbol{h}_i))$.

3.2 Full-State Model (NEF-F)

The full-state model differs from the single-state model in §3.1 by computing a full matrix for every sketch step, instead of a single vector. Namely, instead of Eq. 4, it does the following sketch update for every word $i \in [L]$:

$$\boldsymbol{s}_i^n = \boldsymbol{s}_i^{n-1} + \alpha_i^n \cdot \mathsf{tanh}(\mathsf{affine}(\boldsymbol{c}_i^n)). \quad (7)$$

Note that the only difference is in replacing the single vector $\bar{\boldsymbol{c}}^n$ by the word-specific vector $\boldsymbol{c}_i^n$. As a result, this is no longer a rank-one update of the sketch matrix, but a full update. In the extreme case where the attention is peaked on a single word, the sketch update reduces to the same form as in the single-state model (Eq. 5). However, the full-state model is generally more flexible and allows processing words in parallel, since it allows different sketch updates for multiple words receiving attention, instead of trying to force those words to receive the same update. We will see in the experiments (§5) that this flexibility can be important in practice.

3.3 Computational Complexity

For both models, assuming that the $\rho(\boldsymbol{z}; \boldsymbol{\beta})$ transformation in Eq. 3 takes $O(L)$ time to compute, the total runtime of Alg. 2 is $O((N+K)L)$ (where K is the number of tags), which becomes $O(L^2)$ if $K \leq N = L$. This is so because the input-sketch representation, the attention mechanism, and the sketch generation step all have $O(L)$ complexity, and the final softmax layer requires $O(KL)$ operations. This is the same runtime of the vanilla easy-first algorithm, though the latter can be reduced to $O(KL\log L)$ with caching and a heap, if the scores in line 4 depend only on local sketches (Goldberg and Elhadad, 2010). By comparison, a standard BILSTM tagger has runtime $O(KL)$.

4 Constrained Softmax Attention

An important part of our models is their attention component (line 7 in Alg. 2). To keep the "easy-first" intuition, we would like the transformation ρ in Eq. 3 to have a couple of properties:

1. **Sparsity:** being able to generate sparse distributions $\boldsymbol{\alpha}^n$ (ideally, peaked on a single word).
2. **Evenness:** over iterations, spreading attention evenly over the words. Ideally, the cumulative attention should satisfy $\boldsymbol{\beta}^n \in [0,1]^L$ for every $n \in [N]$ and $\boldsymbol{\beta}^N = \mathbf{1}$.

The standard choice for attention mechanisms is the **softmax** transformation, $\boldsymbol{\alpha}^n = \mathsf{softmax}(\boldsymbol{z}^n)$. However, the softmax does not satisfy either of the properties above. For the first requirement, we could incorporate a "temperature" parameter in the softmax to push for more peaked distributions. However, this does not guarantee sparsity (only "hard attention" in the limit) and we found it numerically unstable when plugged in Alg. 2. For the second one, we could add a penalty before the softmax transformation, $\boldsymbol{\alpha}^n = \mathsf{softmax}(\boldsymbol{z}^n - \lambda\boldsymbol{\beta}^{n-1})$, where $\lambda \geq 0$ is a tunable hyperparameter. This strategy was found effective to prevent a word to receive too much attention, but it made the model less accurate.

An alternative is the **sparsemax** transformation (Martins and Astudillo, 2016):

$$\mathsf{sparsemax}(\boldsymbol{z}) = \underset{\boldsymbol{\alpha} \in \Delta^{L-1}}{\mathsf{argmin}} \|\boldsymbol{\alpha} - \boldsymbol{z}\|^2. \quad (8)$$

The sparsemax maintains most of the appealing properties of the softmax (efficiency to evaluate and backpropagate), and it is able to generate truly sparse distributions. However, it still does not satisfy the "evenness" property.

Instead, we propose a novel **constrained softmax** transformation that satisfies both requirements. It resembles the standard softmax, but it allows imposing hard constraints on the maximal probability assigned to each word. Let us start by writing the (standard) softmax in the following variational form (**?**):

$$\begin{aligned} \mathsf{softmax}(\boldsymbol{z}) &= \underset{\boldsymbol{\alpha} \in \Delta^{L-1}}{\mathsf{argmin}}\ \mathbf{KL}(\boldsymbol{\alpha} \| \mathsf{softmax}(\boldsymbol{z})) \\ &= \underset{\boldsymbol{\alpha} \in \Delta^{L-1}}{\mathsf{argmin}}\ -\mathbf{H}(\boldsymbol{\alpha}) - \boldsymbol{z}^\top \boldsymbol{\alpha}, \quad (9) \end{aligned}$$

where **KL** and **H** denote the Kullback-Leibler divergence and the entropy, respectively. Based on this observation, we define the constrained softmax transformation as follows:

$$\begin{aligned} \mathsf{csoftmax}(\boldsymbol{z}; \boldsymbol{u}) &= \underset{\boldsymbol{\alpha} \in \Delta^{L-1}}{\mathsf{argmin}}\ -\mathbf{H}(\boldsymbol{\alpha}) - \boldsymbol{z}^\top \boldsymbol{\alpha} \\ &\qquad \text{s.t.} \quad \boldsymbol{\alpha} \leq \boldsymbol{u}, \quad (10) \end{aligned}$$

where $\boldsymbol{u} \in \mathbb{R}^L$ is a vector of upper bounds. Note that, if $\boldsymbol{u} \geq \mathbf{1}$, all constraints are loose and this

Algorithm 3 Constrained Softmax Forward

```
Input: $\boldsymbol{z}, \boldsymbol{u}$
Output: $\boldsymbol{\alpha} = \mathsf{csoftmax}(\boldsymbol{z}; \boldsymbol{u})$
initialize $s := 0, \mathcal{A} := [L], Z := \sum_{i=1}^{K} \mathsf{exp}(z_i)$
sort $q_{i_1} \geq \ldots \geq q_{i_L}$, where $q_i = \mathsf{exp}(z_i)/u_i, \forall i \in [L]$
for $k = 1$ to $L$ do
    $\alpha_{i_k} := \mathsf{exp}(z_{i_k})(1-s)/Z$
    if $\alpha_{i_k} > u_{i_k}$ then
        $Z := Z - \mathsf{exp}(z_{i_k})$
        $\alpha_{i_k} := u_{i_k}, \quad s := s + u_{i_k}, \quad \mathcal{A} := \mathcal{A} \setminus \{i_k\}$
    end if
end for
```

reduces to the standard softmax; on the contrary, if $\boldsymbol{u} \in \Delta^{L-1}$, they are tight and we must have $\boldsymbol{\alpha} = \boldsymbol{u}$ due to the normalization constraint. Thus, we propose the following for Eq. 3:

$$\boldsymbol{\alpha}^n = \mathsf{csoftmax}(\boldsymbol{z}^n; \mathbf{1} - \boldsymbol{\beta}^{n-1}). \tag{11}$$

The constraints guarantee $\boldsymbol{\beta}^n = \boldsymbol{\beta}^{n-1} + \boldsymbol{\alpha}^n \leq \mathbf{1}$. Since $\mathbf{1}^\top \boldsymbol{\beta}^N = \sum_{n=1}^{N} \mathbf{1}^\top \boldsymbol{\alpha}^n = N$, they also ensure that $\boldsymbol{\beta}^N = \mathbf{1}$, hence the "evenness" property is fully satisfied. Intuitively, each word gets a credit of one unit of attention that is consumed during the execution of the algorithm. When this credit expires, all subsequent attention weights for that word will be zero.

The next proposition shows how to evaluate the constrained softmax and compute its gradients.

Proposition 1 *Let* $\boldsymbol{\alpha} = \mathsf{csoftmax}(\boldsymbol{z}; \boldsymbol{u})$*, and define the set* $\mathcal{A} = \{i \in [L] \mid \alpha_i < u_i\}$ *of the constraints in Eq. 10 that are met strictly. Then:*

- **Forward propagation.** *The solution of Eq. 10 can be written in closed form as* $\alpha_i = \min\{\mathsf{exp}(z_i)/Z, u_i\}$*, where* $Z = \frac{\sum_{i \in \mathcal{A}} \mathsf{exp}(z_i)}{1 - \sum_{i \notin \mathcal{A}} u_i}$.
- **Gradient backpropagation.** *Let* $L(\boldsymbol{\theta})$ *be a loss function,* $\mathrm{d}\boldsymbol{\alpha} = \nabla_{\boldsymbol{\alpha}} L(\boldsymbol{\theta})$ *be the output gradient, and* $\mathrm{d}\boldsymbol{z} = \nabla_{\boldsymbol{z}} L(\boldsymbol{\theta})$ *and* $\mathrm{d}\boldsymbol{u} = \nabla_{\boldsymbol{u}} L(\boldsymbol{\theta})$ *be the input gradients. Then, we have:*

$$\mathrm{d}z_i = \mathbb{1}(i \in \mathcal{A})\alpha_i(\mathrm{d}\alpha_i - m) \tag{12}$$

$$\mathrm{d}u_i = \mathbb{1}(i \notin \mathcal{A})(\mathrm{d}\alpha_i - m), \tag{13}$$

where $m = (\sum_{i \in \mathcal{A}} \alpha_i \, \mathrm{d}\alpha_i)/(1 - \sum_{i \notin \mathcal{A}} u_i)$.

Proof: See App. A (supplementary material). ■

Algs. 3–4 turn the results in Prop. 1 into concrete procedures for evaluating csoftmax and for backpropagating its gradient. Their runtimes are respectively $O(L \log L)$ and $O(L)$.

Algorithm 4 Constrained Softmax Backprop

```
Input: $\boldsymbol{z}, \boldsymbol{u}, \mathrm{d}\boldsymbol{\alpha}$ (and cached $\boldsymbol{\alpha}, \mathcal{A}, s$ from Alg. 3)
Output: $\mathrm{d}\boldsymbol{z}, \mathrm{d}\boldsymbol{u}$
$m := \sum_{i \in \mathcal{A}} \alpha_i \, \mathrm{d}\alpha_i/(1-s)$
for $i \in [L]$ do
    if $i \in \mathcal{A}$ then
        $\mathrm{d}z_i := \alpha_i(\mathrm{d}\alpha_i - m), \quad \mathrm{d}u_i := 0$
    else
        $\mathrm{d}z_i := 0, \quad \mathrm{d}u_i := \mathrm{d}\alpha_i - m$
    end if
end for
```

5 Experiments

We evaluate our neural easy-first models in three sequence tagging tasks: POS tagging, NER, and word quality estimation.

5.1 Part-of-Speech Tagging

We ran POS tagging experiments in 12 languages from the Universal Dependencies project v1.4 (Nivre et al., 2016), using the standard splits. The datasets contain 17 universal tags.[3]

We implemented Alg. 2 in DyNet (Neubig et al., 2017), which we extended with the constrained softmax operator (Algs. 3–4).[4] We used 64-dimensional word embeddings, initialized with pre-trained Polyglot vectors (Al-Rfou et al., 2013). Apart from the words, we embedded prefix and suffix character n-grams with $n \leq 4$. We set the affix embedding size to 50 and summed all these embeddings; in the end, we obtained a 164-dimensional representation for each word (words, prefixes, and suffixes). We then fed these embeddings into a BILSTM (with 50 hidden units in each direction) to obtain the encoder states $[\boldsymbol{h}_1, \ldots, \boldsymbol{h}_L] \in \mathbb{R}^{100 \times L}$. The other hyperparameters were set as follows: we used a context size $w = 2$, set the pre-attention size D_z and the sketch size D_s to 50, and applied dropout with a probability of 0.2 after the embedding and BILSTM layers and before the final softmax output layer.[5] We ran 20 epochs of Adagrad to minimize the cross-entropy loss, with a stepsize of 0.1, and gradient

[3] Our choice of languages covers 8 families: Romance (French, Portuguese, Spanish), Germanic (English, German), Slavic (Czech, Russian), Semitic (Arabic), Indo-Iranian (Hindi), Austronesian (Indonesian), Sino-Tibetan (Chinese), and Japonic (Japanese). For all languages, we used the datasets that have the canonical language name, except for Russian, where we used the (much larger) SynTagRus corpus. We prefered large datasets over small ones, to reduce the impact of overfitting in our analysis.

[4] The code is available at `https://github.com/Unbabel/neural-easy-first`.

[5] These hyperparameters were tuned in the English development set, and kept fixed across languages.

	Ara.	Chi.	Cze.	Eng.	Fre.	Ger.	Hin.	Ind.	Jap.	Por.	Rus.	Spa.	**Avg.**
Gillick et al. (2016)†	–	–	*98.44*	*94.00*	*95.17*	*92.34*	–	*91.03*	–	–	–	*95.26*	–
Plank et al. (2016)†	*98.91*	–	*98.24*	*95.16*	*96.11*	*93.38*	*97.10*	*93.41*	–	*97.90*	–	*95.74*	–
Linear (TurboTagger)	95.18	91.38	98.07	94.43	96.48	91.92	95.98	93.12	89.35	96.63	97.32	**95.44**	94.88
BILSTM	95.60	92.73	98.46	94.94	96.37	**92.97**	96.80	93.79	92.72	97.01	97.83	95.27	95.39
Vanilla Easy-First	95.62	92.78	98.50	94.78	96.35	92.91	96.81	93.68	92.68	**97.12**	97.93	95.31	95.42
NEF-S, csoftmax	**95.63**	92.87	98.50	94.91	96.41	92.86	96.79	93.65	**92.93**	96.57	97.91	95.42	95.42
NEF-F, softmax	95.61	92.92	98.47	**95.15**	96.52	92.96	96.75	93.67	92.57	**97.12**	97.92	95.41	95.43
NEF-F, sparsemax	95.14	92.86	97.75	93.97	91.79	90.27	89.40	93.48	92.61	95.27	96.32	95.36	93.88
NEF-F, csoftmax	95.53	**92.99**	**98.53**	95.01	**96.70**	92.93	**96.98**	**93.81**	92.72	97.04	**97.94**	95.42	**95.47**

Table 1: POS tagging accuracies. The average in the rightmost column is over the words of each treebank. †Note that Gillick et al. (2016) and Plank et al. (2016) are not strictly comparable, since they use older versions of the treebanks (UD1.1 and UD1.2, respectively).

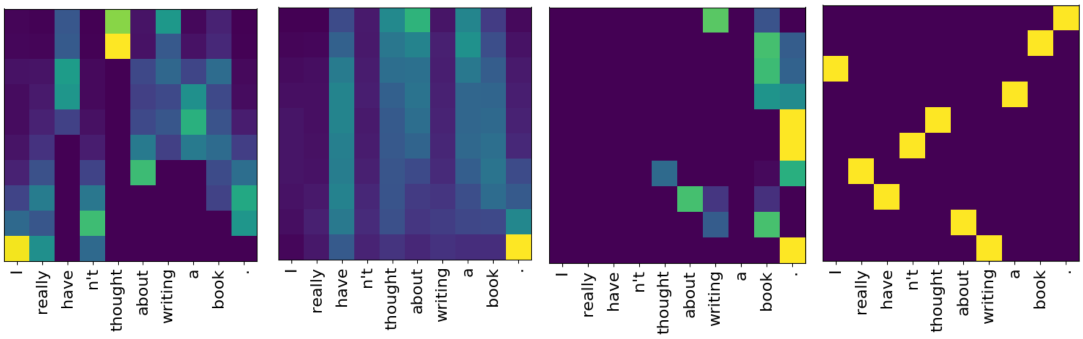


Figure 2: Attention visualization for English POS tagging. From the left: constrained softmax, softmax, sparsemax, vanilla. Rows correspond to attention vectors (high values in yellow, low ones in dark blue).

clipping of 5 (DyNet's default). We excluded from the training set sentences longer than 50 words.

Table 1 compares several variants of our neural easy-first system—the single-state model (NEF-S), the full-state model (NEF-F), and the latter with softmax, sparsemax, and csoftmax attention. We used as many sketch steps as the number of words, $N = L$. As baselines, we used:

- A feature-based linear model (TurboTagger, Martins et al. (2013)).
- A BILSTM tagger identical to our system, but without sketch steps ($N = 0$).
- A vanilla easy-first tagger (Alg. 1), using the argument of the softmax in Eq. 6 as the scoring function. This uses the same sketch representations as the neural easy-first systems, but replaces the attention mechanism by "hard" attention placed on the highest scored word.

For comparison, we also show the accuracies reported by Gillick et al. (2016) for their byte-to-span system (trained separately on each language) and by Plank et al. (2016) for their state-of-the-art multi-task BILSTM tagger (these results are not fully comparable though, due to different treebank versions).

Among the neural easy-first systems, we observe that NEF-F with csoftmax attention generally outperforms the others, but the differences are very slight (excluding the sparsemax attention system, which performed substancially worse). This system wins over the linear system for all languages but Spanish, and over the BILSTM baseline for 9 out of 12 languages (loses in Arabic and German, and ties in Japanese). Note, however, that the differences are small (95.47% against 95.39%, averaged across treebanks). We conjecture that this is due to the fact that the BILSTM already captures most of the relevant context in its encoder. Our NEF-F system with csoftmax also wins over the vanilla easy-first system for 10 out of 12 languages (arguably due to its ability to backpropagate the gradients through the soft attention mechanism), but the difference in the average score is again small (95.47% against 95.42%).

Figure 2 depicts some patterns learned by the NEF-F model with various attention types. With the csoftmax, the model learns to move left and right, and the main verb *"thought"* is the most prominent candidate for the easiest decision. In fact, in 57% of the test sentences, the model focuses first on a verb. The "raindrop" appearance of the plot is due to the evenness property of csoftmax, which causes the attention over a word to increase gradually until the cumulative attention is exhausted. This constrasts with the softmax attention (less diverse and non-sparse) and the sparsemax (sparse, but not even). We show for comparison the (hard) decisions made by the vanilla easy-first decoder.

5.2 Named Entity Recognition

Next, we applied our model to NER. We used the official datasets from the CoNLL 2002-3 shared tasks (Sang, 2002; Sang and De Meulder, 2003), which tag names, locations, and organizations using a BIO scheme, and cover four languages (Dutch, English, German, and Spanish). We made two experiments: one using the exact same BILSTM and NEF models with a standard softmax output layer, as in §5.1 (which does not guarantee valid segmentations), and another one replacing the output softmax layer by a sequential CRF layer, which requires learning $O(K^2)$ additional parameters for pairs of consecutive tags (Huang et al., 2015; Lample et al., 2016). We used the same hyperparameters as in the POS tagging experiments, except the dropout probability, which was set to 0.3 (tuned on the validation set). For English, we used pre-trained 300-dimensional GloVe-840B embeddings (Pennington et al., 2014); for Spanish and German, we used the 64-dimensional word embeddings from Lample et al. (2016); for Dutch we used the aforementioned Polyglot vectors. All embeddings are fine-tuned during training. Since many words are not entities, and those receive a default "outside" tag, we expect that fewer sketch steps are necessary to achieve top performance.

Table 2 shows the results, which confirm this hypothesis. We compare the same BILSTM baseline to our NEF-S and NEF-F models with csoftmax attention (with and without the CRF output layer), varying the maximum number of sketch steps. We also compare against the byte-to-span model of Gillick et al. (2016) and the state-of-the-art character-based LSTM-CRF system of Lample et al. (2016).[6] We can see that, for all languages, the NEF-CRF-F model with 5 steps is consistently better than the BILSTM-CRF and, with the exception of English, the NEF-CRF-S model. The same holds for the BILSTM and NEF-F models without the CRF output layer. With the exception of German, increasing the number of steps did not make a big difference.

	Dut.	Eng.	Ger.	Spa.
Gillick et al. (2016)	78.08	84.57	72.08	81.83
Lample et al. (2016)	**81.74**	**90.94**	**78.76**	**85.75**
BILSTM	76.56	85.74	70.05	77.00
NEF-S, $N = 5$	77.37	86.40	72.27	75.80
NEF-F, $N = 5$	77.96	86.11	72.60	**79.22**
NEF-F, $N = 10$	77.52	**87.11**	**72.96**	78.99
NEF-F, $N = L$	**78.46**	86.36	72.35	78.87
BILSTM-CRF	79.00	86.96	72.98	80.44
NEF-CRF-S, $N = 5$	78.89	**88.33**	72.37	80.21
NEF-CRF-F, $N = 5$	80.03	88.01	73.45	**81.00**
NEF-CRF-F, $N = 10$	79.86	87.51	73.63	80.35
NEF-CRF-F, $N = L$	**80.29**	87.58	**74.75**	80.71

Table 2: F_1 scores for NER, computed by the CoNLL 2002 evaluation script.

Figure 3 shows the attention distributions over the sketch steps for an English sentence, for full-state models trained with $N \in \{5, L\}$. The model with L sketch steps learned that it is easiest to focus on the beginning of a named entity, and then to move to the right to identify the full span. The model with only 5 sketch steps learns to go straight to the point, placing most attention on the entity words and ignoring most of the O-tokens.

5.3 Word-Level Quality Estimation

Finally, we evaluate our model's performance on word-level translation quality estimation. The goal is to evaluate a translation system's quality without access to reference translations (Blatz et al., 2004; Specia et al., 2013). Given a sentence pair (a source sentence and its machine translated sentence in a target language), a word-level system classifies each target word as OK or BAD. We used the official English-German dataset from the WMT16 shared task (Bojar et al., 2016).

This task differs from the previous ones in which its input is a sentence pair and not a single

[6] The current state of the art on these datasets (Gillick et al., 2016; Lample et al., 2016; Ma and Hovy, 2016) is achieved by more sophisticated systems with more parameters, mixing character and word-based models, sharing a model across languages, or combining CRFs with convolutional and recurrent layers. We used simpler models in our experiments since our goal is to assess how much the neural easy-first systems can bring in addition to a BILSTM system, rather than building a state-of-the-art system.

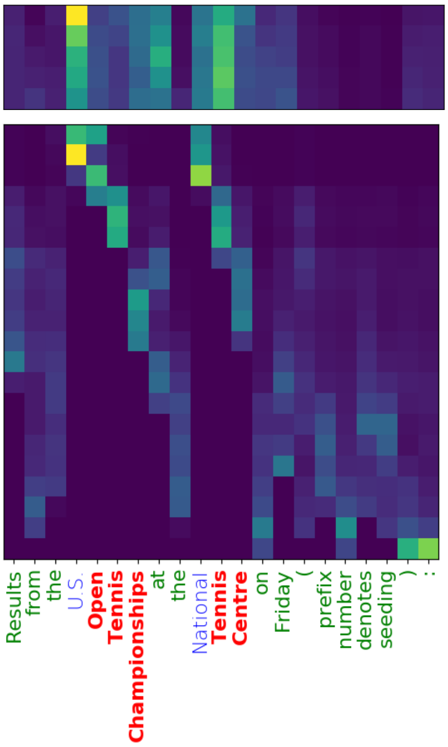


Figure 3: Attention visualization for English NER, for 5 (top) and L (bottom) sketch steps. Words tagged as B-* are marked in light blue, those with I-* tags, in bold red, and with O-* tags, in green.

sentence. We replaced the affix embeddings by the concatenation of the 64-dimensional embeddings of the target words with those of the aligned source words (we used the alignments provided in the shared task), yielding 128-dimensional representations. We used the same hyperparameters as in the POS tagging task, except the dropout probability, set to 0.1. We followed prior work (Kreutzer et al., 2015) and upweighted the BAD words in the loss function to make the model more pessimistic; we used a weight of 5 (tuned in the validation set).

BILSTM	NEF-S		NEF-F	
	$N = 5$	$N = L$	$N = 5$	$N = L$
39.71	40.91	40.99	**41.18**	40.84

Table 3: F_1-MULT scores (product of F_1 for OK and BAD words) for word-level quality estimation, computed by the official shared task script.

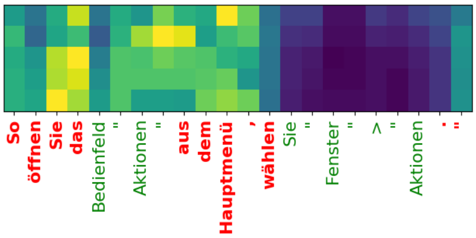


Figure 4: Example for word-level quality estimation. The source sentence is *"To open the Actions panel, from the main menu, choose Window > Actions."* BAD words are red (bold font), OK words are green.

Table 3 shows the results. We see that all our NEF-S and NEF-F models outperform the BILSTM, and that the NEF-F model with 5 sketch steps achieved the best results.[7] Figure 4 illustrates the attention over the target words for 5 sketches. We observe that the attention focuses early in the areas predicted BAD and moves left and right within these areas, not wasting attention on the OK part of the sentence. This block-wise focus makes sense for quality estimation, since often complete phrases are BAD.

5.4 Ablation Study

To better understand our proposed model, we carried out an ablation study for NER on the English dataset. The following alternate configurations were tried and compared against the NEF-CRF-F model with csoftmax attention and 5 sketch steps:

- A NEF-CRF-F model for which the final concatenation in Eq. 6 was removed, being replaced by $\boldsymbol{p}_i = \mathsf{softmax}(\mathsf{affine}(\boldsymbol{s}_i^N))$. The goal was to see if the sketches retain enough information about the input to make a final prediction without requiring the states $\boldsymbol{h}_i$.
- A model for which the attention mechanism applied at each sketch step was replaced by a uniform distribution over the input words.
- A vanilla easy-first system (Alg. 1). Since this system can only focus on one word at the time (unlike the models with soft attention), we tried both $N = 5$ and $N = L$ sketch steps.
- A left-to-right and right-to-left model, which replaces the attention mechanism by one of these two prescribed orders.

Table 4 shows the results. As expected, the neural easy-first system was the best performing one, although the difference with respect to the ablated

[7] Our best system would rank third in the shared task, out of 13 submissions. The winner system, which achieved 49.52 F_1-MULT, was considerably more complex than ours, using an ensemble of three neural networks with a linear system (Martins et al., 2016).

NEF-CRF-F, $N = 5$	**88.01**
NEF-CRF-F w/out concat, $N = 5$	87.47
Uniform Attention, $N = 5$	87.46
Vanilla EF + CRF, $N = 5$	87.17
Vanilla EF + CRF, $N = L$	87.46
Left-to-right + CRF, $N = L$	87.57
Right-to-left + CRF, $N = L$	87.53

Table 4: Ablation experiments. Reported are F_1 scores for NER in the English test set.

systems is relatively small. Removing the concatenation in Eq. 6 is harmful, which suggests that there is information about the input not retained in the sketches. The uniform attention performs surprisingly well, and so do the left-to-right and right-to-left models, but they are still about half a point behind. The vanilla easy-first system has the worst performance with $N = 5$. This is due to the fact that the vanilla model is uncapable of processing words "in parallel" in the same sketch step, a disadvantage with respect to the neural easy-first models, which have this capability due to their soft attention mechanisms (see the top image in Fig. 3).

6 Related Work

Vanilla easy-first decoders have been used in POS tagging (Tsuruoka and Tsujii, 2005; Ma et al., 2013), dependency parsing (Goldberg and Elhadad, 2010), and coreference resolution (Stoyanov and Eisner, 2012), being related to cyclic dependency networks and guided learning (Toutanova et al., 2003; Shen et al., 2007). More recent works compute scores with a neural network (Socher et al., 2011; Clark and Manning, 2016; Kiperwasser and Goldberg, 2016a), but they still operate in a discrete space to pick the easiest actions (the non-differentiable argmax in line 6 of Alg. 1). Generalizing this idea to "continuous" operations is at the very core of our paper, allowing gradients to be fully backpropagated. In a different context, building differentiable computation structures has also been addressed by Graves et al. (2014); Grefenstette et al. (2015).

An important contribution of our paper is the constrained softmax transformation. Others have proposed alternatives to softmax attention, including the sparsemax (Martins and Astudillo, 2016) and multi-focal attention (Globerson et al., 2016). The latter computes a KL projection onto a budget polytope to focus on multiple words. Our constrained softmax also corresponds to a KL projection, but (i) it involves box constraints instead of a budget, (ii) it is normalized to 1, and (iii) we also backpropagate the gradient over the constraint variables. It also achieves sparsity (see the "raindrop" plots in Figures 2–4), and is suitable for sequentially computing attention distributions when diversity is desired (e.g. soft 1-to-1 alignments). Recently, Chorowski and Jaitly (2016) developed an heuristic with a threshold on the total attention as a "coverage criterion" (see their Eq. 11), however their heuristic is non-differentiable.

Our sketch generation step is similar in spirit to the "deep recurrent attentive writer" (DRAW, Gregor et al. (2015)) which generates images by iteratively refining sketches with a recurrent neural network (RNN). However, our goal is very different: instead of generating images, we generate vectors that lead to a final sequence tagging prediction.

Finally, the visualization provided in Figures 2–4 brings up the question how to understand and rationalize predictions by neural network systems, addressed by Lei et al. (2016). Their model, however, uses a form of stochastic attention and it does not perform any iterative refinement like ours.

7 Conclusions

We introduced novel fully-differentiable easy-first taggers that learn to make predictions over sequences in an order that is adapted to the task at hand. The decoder iteratively updates a sketch of the predictions by interacting with an attention mechanism. To spread attention evenly through all words, we introduced a new constrained softmax transformation, along with an algorithm to backpropagate its gradients. Our neural-easy first decoder consistently outperformed a BILSTM on a range of sequence tagging tasks.

A natural direction for future work is to go beyond sequence tagging (which we regard as a simple first step) toward other NLP structured prediction problems, such as sequence-to-sequence prediction. This requires replacing the sketch matrix in Alg. 2 by a dynamic memory structure.

Acknowledgments

We thank the Unbabel research team and the reviewers for their comments. This work was supported by the Fundação para a Ciência e Tecnologia through contracts UID/EEA/50008/2013, PTDC/EEI-SII/7092/2014 (LearnBig), and CMU-PERI/TIC/0046/2014 (GoLocal).

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. *arXiv preprint arXiv:1307.1662* .

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*. pages 2442–2452.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*. pages 1171–1179.

John Blatz, Erin Fitzgerald, George Foster, Simona Gandrabur, Cyril Goutte, Alex Kulesza, Alberto Sanchis, and Nicola Ueffing. 2004. Confidence estimation for machine translation. In *Proc. of the International Conference on Computational Linguistics*. page 315.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*. pages 131–198.

Jan Chorowski and Navdeep Jaitly. 2016. Towards better decoding and language model integration in sequence to sequence models. *arXiv preprint arXiv:1612.02695* .

Kevin Clark and Christopher D Manning. 2016. Improving coreference resolution by learning entity-level distributed representations. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*.

H. Daumé, J. Langford, and D. Marcu. 2009. Search-based structured prediction. *Machine learning* 75(3):297–325.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Beijing, China, pages 334–343.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proc. of the Annual Meeting of the North-American Chapter of the Association for Computational Linguistics*.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*.

Y. Goldberg and M. Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Proc. of Annual Conference of the North American Chapter of the Association for Computational Linguistics*. pages 742–750.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*. Springer, pages 799–804.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing Machines. *arXiv preprint arXiv:1410.5401* .

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. 2015. Learning to Transduce with Unbounded Memory. In *Advances in Neural Information Processing Systems*. pages 1819–1827.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. 2015. Draw: A recurrent neural network for image generation. In *Proc. of the International Conference on Machine Learning*. pages 1462–1471.

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

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

Eliyahu Kiperwasser and Yoav Goldberg. 2016a. Easy-first dependency parsing with hierarchical tree lstms. *arXiv preprint arXiv:1603.00375* .

Eliyahu Kiperwasser and Yoav Goldberg. 2016b. Simple and accurate dependency parsing using bidirectional lstm feature representations. *arXiv preprint arXiv:1603.04351* .

Julia Kreutzer, Shigehiko Schamoni, and Stefan Riezler. 2015. Quality estimation from scratch (quetch): Deep learning for word-level translation quality estimation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. pages 316–322.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of the Annual Meeting of the North-American Chapter of the Association for Computational Linguistics*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proc. of Empirical Methods for Natural Language Processing*.

Ji Ma, Tong Xiao, and Nan Yang. 2013. Easy-first pos tagging and dependency parsing with beam search. In *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*.

André F. T Martins, Miguel B. Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*.

André F. T. Martins and Ramón Astudillo. 2016. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. In *Proc. of the International Conference on Machine Learning*.

André F. T. Martins, Ramón Astudillo, Chris Hokamp, and Fábio Kepler. 2016. Unbabel's participation in the wmt16 word-level translation quality estimation shared task. In *Proceedings of the First Conference on Machine Translation*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Riyaz Ahmad Bhat, Eckhard Bick, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Fabricio Chalub, Çar Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Claudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökrmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Jan Hajič, Linh Hà M, Dag Haug, Barbora Hladká, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Jessica Kenney, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lucia Lam, Phng Lê Hng, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Keiko Sophie Mori, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Lng Nguyn Th, Huyn Nguyn Th Minh, Vitaly Nikolaev, Hanna Nurmi, Petya Osenova, Robert Östling, Lilja Øvrelid, Valeria Paiva, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalnia, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Baiba Saulīte, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Carolyn Spadine, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jing Xian Wang, Jonathan North Washington, Mats Wirén, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2016. Universal dependencies 1.4. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University in Prague. http://hdl.handle.net/11234/1-1827.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)* 12:1532–1543.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (short papers)*.

Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*. volume 1, page 6.

E.F.T.K. Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proc. of International Conference on Natural Language Learning*.

E.F.T.K. Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proc. of International Conference on Natural Language Learning*.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *ACL*. volume 7, pages 760–767.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. pages 129–136.

Lucia Specia, Kashif Shah, Jose G.C. de Souza, and Trevor Cohn. 2013. QuEst - a translation quality estimation framework. In *Proc. of the Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. pages 79–84. http://www.aclweb.org/anthology/P13-4014.

Veselin Stoyanov and Jason Eisner. 2012. Easy-first coreference resolution. In *COLING*. pages 2519–2534.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. pages 3104–3112.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 173–180.

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*. Association for Computational Linguistics, pages 467–474.

M. Wainwright and M. Jordan. 2008. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers.

A Proof of Proposition 1

We provide here a detailed proof of Proposition 1.

A.1 Forward Propagation

The optimization problem is

$$\mathsf{csoftmax}(\boldsymbol{z}, \boldsymbol{u}) = \mathsf{argmin} \quad -H(\boldsymbol{\alpha}) - \boldsymbol{z}^\top \boldsymbol{\alpha} \\ \text{s.t.} \quad \begin{cases} \mathbf{1}^\top \boldsymbol{\alpha} = 1 \\ \mathbf{0} \leq \boldsymbol{\alpha} \leq \boldsymbol{u}. \end{cases}$$

The Lagrangian function is:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}, \lambda, \boldsymbol{\mu}, \boldsymbol{\nu}) &= -H(\boldsymbol{\alpha}) - \boldsymbol{z}^\top \boldsymbol{\alpha} + \lambda(\mathbf{1}^\top \boldsymbol{\alpha} - 1) \\ &\quad -\boldsymbol{\mu}^\top \boldsymbol{\alpha} + \boldsymbol{\nu}^\top (\boldsymbol{\alpha} - \boldsymbol{u}). \end{aligned} \tag{14}$$

To obtain the solution, we invoke the Karush-Kuhn-Tucker conditions. From the stationarity condition, we have $\mathbf{0} = \log(\boldsymbol{\alpha}) + \mathbf{1} - \boldsymbol{z} + \lambda\mathbf{1} - \boldsymbol{\mu} + \boldsymbol{\nu}$, which due to the primal feasibility condition implies that the solution is of the form:

$$\boldsymbol{\alpha} = \exp(\boldsymbol{z} + \boldsymbol{\mu} - \boldsymbol{\nu})/Z, \tag{15}$$

where Z is a normalization constant. From the complementarity slackness condition, we have that $0 < \alpha_i < u_i$ implies that $\mu_i = \nu_i = 0$ and therefore $\alpha_i = \exp(z_i)/Z$. On the other hand, $\nu_i > 0$ implies $\alpha_i = u_i$. Hence the solution can be written as $\alpha_i = \min\{\exp(z_i)/Z, u_i\}$, where Z is determined such that the distribution normalizes:

$$Z = \frac{\sum_{i \in \mathcal{A}} \exp(z_i)}{1 - \sum_{i \notin \mathcal{A}} u_i}, \tag{16}$$

with $\mathcal{A} = \{i \in [L] \mid \alpha_i < u_i\}$.

A.2 Gradient Backpropagation

We now turn to the problem of backpropagating the gradients through the constrained softmax transformation. For that, we need to compute its Jacobian matrix, i.e., the derivatives $\frac{\partial \alpha_i}{\partial z_j}$ and $\frac{\partial \alpha_i}{\partial u_j}$ for $i, j \in [L]$. Let us first express $\boldsymbol{\alpha}$ as

$$\alpha_i = \begin{cases} \frac{\exp(z_i)(1-s)}{\sum_{j \in \mathcal{A}} \exp(z_j)}, & i \in \mathcal{A} \\ u_i, & i \notin \mathcal{A}, \end{cases} \tag{17}$$

where $s = \sum_{j \notin \mathcal{A}} u_j$. Note that we have $\partial s / \partial z_j = 0, \ \forall j$, and $\partial s / \partial u_j = \mathbb{1}(j \notin \mathcal{A})$. To compute the entries of the Jacobian matrix, we need to consider several cases.

Case 1: $\boxed{i \in \mathcal{A}.}$ In this case, the evaluation of Eq. 17 goes through the first branch. Let us first compute the derivative with respect to u_j. Two things can happen: if $j \in \mathcal{A}$, then s does not depend on u_j, hence $\frac{\partial \alpha_i}{\partial u_j} = 0$. Else, if $j \notin \mathcal{A}$, we have

$$\frac{\partial \alpha_i}{\partial u_j} = \frac{-\exp(z_i) \frac{\partial s}{\partial u_j}}{\sum_{k \in \mathcal{A}} \exp(z_k)} = -\alpha_i/(1-s).$$

Now let us compute the derivative with respect to z_j. Three things can happen: if $j \in \mathcal{A}$ and $i \neq j$, we have

$$\begin{aligned} \frac{\partial \alpha_i}{\partial z_j} &= \frac{-\exp(z_i)\exp(z_j)(1-s)}{\left(\sum_{k \in \mathcal{A}} \exp(z_k)\right)^2} \\ &= -\alpha_i \alpha_j/(1-s). \end{aligned} \tag{18}$$

If $j \in \mathcal{A}$ and $i = j$, we have

$$\begin{aligned}\frac{\partial \alpha_i}{\partial z_i} &= (1-s) \times \\ & \quad \frac{\mathsf{exp}(z_i) \sum_{k \in \mathcal{A}} \mathsf{exp}(z_k) - \mathsf{exp}(z_i)^2}{\left(\sum_{k \in \mathcal{A}} \mathsf{exp}(z_k)\right)^2} \\ &= \alpha_i - \alpha_i^2/(1-s). \end{aligned} \tag{19}$$

Finally, if $j \notin \mathcal{A}$, we have $\frac{\partial \alpha_i}{\partial z_j} = 0$.

Case 2: $\boxed{i \notin \mathcal{A}.}$ In this case, the evaluation of Eq. 17 goes through the second branch, which means that $\frac{\partial \alpha_i}{\partial z_j} = 0$, always. Let us now compute the derivative with respect to u_j. This derivative is always zero unless $i = j$, in which case $\frac{\partial \alpha_i}{\partial u_j} = 1$.

To sum up, we have:

$$\frac{\partial \alpha_i}{\partial z_j} = \begin{cases} \mathbb{1}(i=j)\alpha_i - \frac{\alpha_i \alpha_j}{1-s}, & \text{if } i, j \in \mathcal{A} \\ 0, & \text{otherwise,} \end{cases} \tag{20}$$

and

$$\frac{\partial \alpha_i}{\partial u_j} = \begin{cases} -\frac{\alpha_i}{1-s}, & \text{if } i \in \mathcal{A}, j \notin \mathcal{A} \\ 1, & \text{if } i, j \notin \mathcal{A}, i = j \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

Therefore, we obtain:

$$\begin{aligned} \mathrm{d}z_j &= \sum_i \frac{\partial \alpha_i}{\partial z_j} \mathrm{d}\alpha_i \\ &= \mathbb{1}(j \in \mathcal{A}) \left(\alpha_j \mathrm{d}\alpha_j - \frac{\alpha_j \sum_{i \in \mathcal{A}} \alpha_i \mathrm{d}\alpha_i}{1-s} \right) \\ &= \mathbb{1}(j \in \mathcal{A}) \alpha_j (\mathrm{d}\alpha_j - m), \end{aligned} \tag{22}$$

and

$$\begin{aligned} \mathrm{d}u_j &= \sum_i \frac{\partial \alpha_i}{\partial u_j} \mathrm{d}\alpha_i \\ &= \mathbb{1}(j \notin \mathcal{A}) \left(\mathrm{d}\alpha_j - \frac{\sum_{i \in \mathcal{A}} \alpha_i \mathrm{d}\alpha_i}{1-s} \right) \\ &= \mathbb{1}(j \notin \mathcal{A}) (\mathrm{d}\alpha_j - m), \end{aligned} \tag{23}$$

where $m = \frac{\sum_{i \in \mathcal{A}} \alpha_i \mathrm{d}\alpha_i}{1-s}$.

Incremental Skip-gram Model with Negative Sampling

Nobuhiro Kaji and **Hayato Kobayashi**
Yahoo Japan Corporation
{nkaji,hakobaya}@yahoo-corp.jp

Abstract

This paper explores an incremental training strategy for the skip-gram model with negative sampling (SGNS) from both empirical and theoretical perspectives. Existing methods of neural word embeddings, including SGNS, are multi-pass algorithms and thus cannot perform incremental model update. To address this problem, we present a simple incremental extension of SGNS and provide a thorough theoretical analysis to demonstrate its validity. Empirical experiments demonstrated the correctness of the theoretical analysis as well as the practical usefulness of the incremental algorithm.

1 Introduction

Existing methods of neural word embeddings are typically designed to go through the entire training data multiple times. For example, negative sampling (Mikolov et al., 2013b) needs to pre-compute the noise distribution from the entire training data before performing Stochastic Gradient Descent (SGD). It thus needs to go through the training data at least twice. Similarly, hierarchical soft-max (Mikolov et al., 2013b) has to determine the tree structure and GloVe (Pennington et al., 2014) has to count co-occurrence frequencies before performing SGD.

The fact that those existing methods are multi-pass algorithms means that they cannot perform incremental model update when additional training data is provided. Instead, they have to re-train the model on the old and new training data from scratch.

However, the re-training is obviously inefficient since it has to process the entire training data received thus far whenever new training data is provided. This is especially problematic when the amount of the new training data is relatively smaller than the old one. One such situation is that the embedding model is updated on a small amount of training data that includes newly emerged words for instantly adding them to the vocabulary set. Another situation is that the word embeddings are learned from ever-evolving data such as news articles and microblogs (Peng et al., 2017) and the embedding model is periodically updated on newly generated data (*e.g.*, once in a week or month).

This paper investigates an incremental training method of word embeddings with a focus on the skip-gram model with negative sampling (SGNS) (Mikolov et al., 2013b) for its popularity. We present a simple incremental extension of SGNS, referred to as *incremental SGNS*, and provide a thorough theoretical analysis to demonstrate its validity. Our analysis reveals that, under a mild assumption, the optimal solution of incremental SGNS agrees with the original SGNS when the training data size is infinitely large. See Section 4 for the formal and strict statement. Additionally, we present techniques for the efficient implementation of incremental SGNS.

Three experiments were conducted to assess the correctness of the theoretical analysis as well as the practical usefulness of incremental SGNS. The first experiment empirically investigates the validity of the theoretical analysis result. The second experiment compares the word embeddings learned by incremental SGNS and the original SGNS across five benchmark datasets, and demonstrates that those word embeddings are of comparable quality. The last experiment explores the training time of incremental SGNS, demonstrating that it is able to save much training time by avoiding expensive re-training when additional training data is provided.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 363–371
Copenhagen, Denmark, September 7–11, 2017.

2 SGNS Overview

As a preliminary, this section provides a brief overview of SGNS.

Given a word sequence, $w_1, w_2, \ldots, w_n$, for training, the skip-gram model seeks to minimize the following objective to learn word embeddings:

$$\mathcal{L}_{\text{SG}} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{\substack{|j| \leq c \\ j \neq 0}} \log p(w_{i+j} \mid w_i),$$

where w_i is a target word and w_{i+j} is a context word within a window of size c. $p(w_{i+j} \mid w_i)$ represents the probability that w_{i+j} appears within the neighbor of w_i, and is defined as

$$p(w_{i+j} \mid w_i) = \frac{\exp(\mathbf{t}_{w_i} \cdot \mathbf{c}_{w_{i+j}})}{\sum_{w \in \mathcal{W}} \exp(\mathbf{t}_{w_i} \cdot \mathbf{c}_w)}, \qquad (1)$$

where $\mathbf{t}_w$ and $\mathbf{c}_w$ are w's embeddings when it behaves as a target and context, respectively. $\mathcal{W}$ represents the vocabulary set.

Since it is too expensive to optimize the above objective, Mikolov et al. (2013b) proposed negative sampling to speed up skip-gram training. This approximates Eq. (1) using sigmoid functions and k randomly-sampled words, called *negative samples*. The resulting objective is given as

$$\mathcal{L}_{\text{SGNS}} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{\substack{|j| \leq c \\ j \neq 0}} \psi^{+}_{w_i, w_{i+j}} + k \mathbb{E}_{v \sim q(v)} [\psi^{-}_{w_i, v}],$$

where $\psi^{+}_{w,v} = \log \sigma(\mathbf{t}_w \cdot \mathbf{c}_v)$, $\psi^{-}_{w,v} = \log \sigma(-\mathbf{t}_w \cdot \mathbf{c}_v)$, and $\sigma(x)$ is the sigmoid function. The negative sample v is drawn from a smoothed unigram probability distribution referred to as *noise distribution*: $q(v) \propto f(v)^{\alpha}$, where $f(v)$ represents the frequency of a word v in the training data and α is a smoothing parameter ($0 < \alpha \leq 1$).

The objective is optimized by SGD. Given a target-context word pair (w_i and w_{i+j}) and k negative samples ($v_1, v_2, \ldots, v_k$) drawn from the noise distribution, the gradient of $-\psi^{+}_{w_i, w_{i+j}} - k \mathbb{E}_{v \sim q(v)} [\psi^{-}_{w_i, v}] \approx -\psi^{+}_{w_i, w_{i+j}} - \sum_{k'=1}^{k} \psi^{-}_{w_i, v_{k'}}$ is computed. Then, the gradient descent is performed to update $\mathbf{t}_{w_i}$, $\mathbf{c}_{w_{i+j}}$, and $\mathbf{c}_{v_1}, \ldots, \mathbf{c}_{v_k}$.

SGNS training needs to go over the entire training data to pre-compute the noise distribution $q(v)$ before performing SGD. This makes it difficult to perform incremental model update when additional training data is provided.

3 Incremental SGNS

This section explores incremental training of SGNS. The incremental training algorithm (Section 3.1), its efficient implementation (Section 3.2), and the computational complexity (Section 3.3) are discussed in turn.

3.1 Algorithm

Algorithm 1 presents *incremental SGNS*, which goes through the training data in a single-pass to update word embeddings incrementally. Unlike the original SGNS, it does not pre-compute the noise distribution. Instead, it reads the training data word by word[1] to incrementally update the word frequency distribution and the noise distribution while performing SGD. Hereafter, the original SGNS (*c.f.*, Section 2) is referred to as *batch SGNS* to emphasize that the noise distribution is computed in a batch fashion.

The learning rate for SGD is adjusted by using AdaGrad (Duchi et al., 2011). Although the linear decay function has widely been used for training batch SGNS (Mikolov, 2013), adaptive methods such as AdaGrad are more suitable for the incremental training since the amount of training data is unknown in advance or can increase unboundedly.

It is straightforward to extend the incremental SGNS to the mini-batch setting by reading a subset of the training data (or mini-batch), rather than a single word, at a time to update the noise distribution and perform SGD (Algorithm 2). Although this paper primarily focuses on the incremental SGNS, the mini-batch algorithm is also important in practical terms because it is easier to be multithreaded.

Alternatives to Algorithms 2 might be possible. Other possible approaches include computing the noise distribution separately on each subset of the training data, fixing the noise distribution after computing it from the first (possibly large) subset, and so on. We exclude such alternatives from our investigation because it is considered difficult to provide them with theoretical justification.

3.2 Efficient implementation

Although the incremental SGNS is conceptually simple, implementation issues are involved.

[1] In practice, Algorithm 1 buffers a sequence of words $w_{i-c}, \ldots, w_{i+c}$ (rather than a single word w_i) at each step, as it requires an access to the context words w_{i+j} in line 7. This is not a practical problem because the window size c is usually small and independent from the training data size n.

Algorithm 1 Incremental SGNS

```
$f(w) \leftarrow 0$ for all $w \in \mathcal{W}$
for $i = 1, \ldots, n$ do
    $f(w_i) \leftarrow f(w_i) + 1$
    $q(w) \leftarrow \frac{f(w)^\alpha}{\sum_{w' \in \mathcal{W}} f(w')^\alpha}$ for all $w \in \mathcal{W}$
    for $j = -c, \ldots, -1, 1, \ldots, c$ do
        draw $k$ negative samples from $q(w)$: $v_1, \ldots, v_k$
        use SGD to update $\mathbf{t}_{w_i}$, $\mathbf{c}_{w_{i+j}}$, and $\mathbf{c}_{v_1}, \ldots, \mathbf{c}_{v_k}$
    end for
end for
```

Algorithm 2 Mini-batch SGNS

```
for each subset $\mathcal{D}$ of the training data do
    update the noise distribution using $\mathcal{D}$
    perform SGD over $\mathcal{D}$
end for
```

3.2.1 Dynamic vocabulary

One problem that arises when training incremental SGNS is how to maintain the vocabulary set. Since new words emerge endlessly in the training data, the vocabulary set can grow unboundedly and exhaust a memory.

We address this problem by dynamically changing the vocabulary set. The Misra-Gries algorithm (Misra and Gries, 1982) is used to approximately keep track of top-m frequent words during training, and those words are used as the dynamic vocabulary set. This method allows the maximum vocabulary size to be explicitly limited to m, while being able to dynamically change the vocabulary set.

3.2.2 Adaptive unigram table

Another problem is how to generate negative samples efficiently. Since k negative samples per target-context pair have to be generated by the noise distribution, the sampling speed has a significant effect on the overall training efficiency.

Let us first examine how negative samples are generated in batch SGNS. In a popular implementation (Mikolov, 2013), a word array (referred to as a *unigram table*) is constructed such that the number of a word w in it is proportional to $q(w)$. See Table 1 for an example. Using the unigram table, negative samples can be efficiently generated by sampling the table elements uniformly at random. It takes only $O(1)$ time to generate one negative sample.

w	a	b	c
$q(w)$	0.5	0.3	0.2

$T = (a, a, a, a, a, b, b, b, c, c)$

Table 1: Example noise distribution $q(w)$ for the vocabulary set $\mathcal{W} = \{a, b, c\}$ (left) and the corresponding unigram table T of size 10 (right).

Algorithm 3 Adaptive unigram table.

```
1:  $f(w) \leftarrow 0$ for all $w \in \mathcal{W}$
2:  $z \leftarrow 0$
3:  for $i = 1, \ldots, n$ do
4:      $f(w_i) \leftarrow f(w_i) + 1$
5:      $F \leftarrow f(w_i)^\alpha - (f(w_i) - 1)^\alpha$
6:      $z \leftarrow z + F$
7:      if $|T| < \tau$ then
8:          add $F$ copies of $w_i$ to $T$
9:      else
10:         for $j = 1, \ldots, \tau$ do
11:             $T[j] \leftarrow w_i$ with probability $\frac{F}{z}$
12:         end for
13:     end if
14: end for
```

The above method assumes that the noise distribution is fixed and thus cannot be used directly for the incremental training. One simple solution is to reconstruct the unigram table whenever new training data is provided. However, such a method is not effective for the incremental SGNS, because the unigram table reconstruction requires $\mathcal{O}(|\mathcal{W}|)$ time.[2]

We propose a reservoir-based algorithm for efficiently updating the unigram table (Vitter, 1985; Efraimidis, 2015) (Algorithm 3). The algorithm incrementally update the unigram table T while limiting its maximum size to τ. In case $|T| < \tau$, it can be easily confirmed that the number of a word w in T is $f(w)^\alpha (\propto q(w))$. In case $|T| = \tau$, since $z = \sum_{w \in \mathcal{W}} f(w)^\alpha$ is equal to the normalization factor of the noise distribution, it can be proven by induction that, for all j, $T[j]$ is a word w with probability $q(w)$. See (Vitter, 1985; Efraimidis, 2015) for reference.

Note on implementation In line 8, F copies of w_i are added to T. When F is not an integer, the copies are generated so that their expected number becomes F. Specifically, $\lceil F \rceil$ copies are added to T with probability $F - \lfloor F \rfloor$, and $\lfloor F \rfloor$ copies are added otherwise.

The loop from line 10 to 12 becomes expensive if implemented straightforwardly because the maximum table size τ is typically set large (*e.g.*, $\tau = 10^8$ in `word2vec` (Mikolov, 2013)). For acceleration, instead of checking all elements in the unigram table, randomly chosen $\frac{\tau F}{z}$ elements are substituted with w_i. Note that $\frac{\tau F}{z}$ is the expected

[2] This overhead is amortized in mini-batch SGNS if the mini-batch size is sufficiently large. Our discussion here is dedicated to efficiently perform the incremental training irrespective of the mini-batch size.

number of table elements to be substituted in the original algorithm. This approximation achieves great speed-up because we usually have $F \ll z$. In fact, it can be proven that it takes $O(1)$ time when $\alpha = 1.0$. See Appendix[3] A for more discussions.

3.3 Computational complexity

Both incremental and batch SGNS have the same space complexity, which is independent of the training data size n. Both require $\mathcal{O}(|\mathcal{W}|)$ space to store the word embeddings and the word frequency counts, and $\mathcal{O}(|T|)$ space to store the unigram table.

The two algorithms also have the same time complexity. Both require $\mathcal{O}(n)$ training time when the training data size is n. Although incremental SGNS requires extra time for updating the dynamic vocabulary and adaptive unigram table, these costs are practically negligible, as will be demonstrated in Section 5.3.

4 Theoretical Analysis

Although the extension from batch to incremental SGNS is simple and intuitive, it is not readily clear whether incremental SGNS can learn word embeddings as well as the batch counterpart. To answer this question, in this section we examine incremental SGNS from a theoretical point of view.

The analysis begins by examining the difference between the objectives optimized by batch and incremental SGNS (Section 4.1). Then, probabilistic properties of their difference are investigated to demonstrate the relationship between batch and incremental SGNS (Sections 4.2 and 4.3). We shortly touch the mini-batch SGNS at the end of this section (Section 4.4).

4.1 Objective difference

As discussed in Section 2, batch SGNS optimizes the following objective:

$$\mathcal{L}_{\mathrm{B}}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{\substack{|j|\le c\\ j\neq 0}} \psi^{+}_{w_i,w_{i+j}} + k\mathbb{E}_{v\sim q_n(v)}[\psi^{-}_{w_i,v}],$$

where $\theta = (\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{|\mathcal{W}|}, \mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{|\mathcal{W}|})$ collectively represents the model parameters[4] (*i.e.*, word embeddings) and $q_n(v)$ represents the noise distribution. Note that the noise distribution is represented in a different notation than Section 2 to make its dependence on the whole training data explicit. The function $q_i(v)$ is defined as $q_i(v) = \frac{f_i(v)^{\alpha}}{\sum_{v'\in\mathcal{W}} f_i(v')^{\alpha}}$, where $f_i(v)$ represents the word frequency in the first i words of the training data.

In contrast, incremental SGNS computes the gradient of $-\psi^{+}_{w_i,w_{i+j}} - k\mathbb{E}_{v\sim q_i(v)}[\psi^{-}_{w_i,v}]$ at each step to perform gradient descent. Note that the noise distribution does not depend on n but rather on i. Because it can be seen as a sample approximation of the gradient of

$$\mathcal{L}_{\mathrm{I}}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{\substack{|j|\le c\\ j\neq 0}} \psi^{+}_{w_i,w_{i+j}} + k\mathbb{E}_{v\sim q_i(v)}[\psi^{-}_{w_i,v}],$$

incremental SGNS can be interpreted as optimizing $\mathcal{L}_{\mathrm{I}}(\theta)$ with SGD.

Since the expectation terms in the objectives can be rewritten as $\mathbb{E}_{v\sim q_i(v)}[\psi^{-}_{w_i,v}] = \sum_{v\in\mathcal{W}} q_i(v)\psi^{-}_{w_i,v}$, the difference between the two objectives can be given as

$$\begin{aligned}
\Delta\mathcal{L}(\theta) &= \mathcal{L}_{\mathrm{B}}(\theta) - \mathcal{L}_{\mathrm{I}}(\theta)\\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{\substack{|j|\le c\\ j\neq 0}} k \sum_{v\in\mathcal{W}} (q_i(v) - q_n(v))\psi^{-}_{w_i,v}\\
&= \frac{2ck}{n}\sum_{i=1}^{n}\sum_{v\in\mathcal{W}} (q_i(v) - q_n(v))\psi^{-}_{w_i,v}\\
&= \frac{2ck}{n}\sum_{w,v\in\mathcal{W}}\sum_{i=1}^{n} \delta_{w_i,w}(q_i(v) - q_n(v))\psi^{-}_{w,v}
\end{aligned}$$

where $\delta_{w,v} = \delta(w = v)$ is the delta function.

4.2 Unsmoothed case

Let us begin by examining the objective difference $\Delta\mathcal{L}(\theta)$ in the unsmoothed case, $\alpha = 1.0$.

The technical difficulty in analyzing $\Delta\mathcal{L}(\theta)$ is that it is dependent on the word order in the training data. To address this difficulty, we assume that the words in the training data are generated from some stationary distribution. This assumption allows us to investigate the property of $\Delta\mathcal{L}(\theta)$ from a probabilistic perspective. Regarding the validity of this assumption, we want to note that this assumption is already taken by the original SGNS: the probability that the target and context words co-occur is assumed to be independent of their position in the training data.

[3] The appendices are in the supplementary material.

[4] We treat words as integers and thus $\mathcal{W} = \{1, 2, \ldots |\mathcal{W}|\}$.

We below introduce some definitions and notations as the preparation of the analysis.

Definition 1. Let $\mathrm{X}_{i,w}$ be a random variable that represents $\delta_{w_i,w}$. It takes 1 when the i-th word in the training data is $w \in \mathcal{W}$ and 0 otherwise.

Remind that we assume that the words in the training data are generated from a stationary distribution. This assumption means that the expectation and (co)variance of $\mathrm{X}_{i,w}$ do not depend on the index i. Hereafter, they are respectively denoted as $\mathbb{E}[\mathrm{X}_{i,w}] = \mu_w$ and $\mathbb{V}[\mathrm{X}_{i,w}, \mathrm{X}_{j,v}] = \rho_{w,v}$.

Definition 2. Let $\mathrm{Y}_{i,w}$ be a random variable that represents $q_i(w)$ when $\alpha = 1.0$. It is given as $\mathrm{Y}_{i,w} = \frac{1}{i}\sum_{i'=1}^{i} \mathrm{X}_{i',w}$.

4.2.1 Convergence of the first and second order moments of $\Delta\mathcal{L}(\theta)$

It can be shown that the first order moment of $\Delta\mathcal{L}(\theta)$ has an analytical form.

Theorem 1. *The first order moment of $\Delta\mathcal{L}(\theta)$ is given as*

$$\mathbb{E}[\Delta\mathcal{L}(\theta)] = \frac{2ck(H_n - 1)}{n} \sum_{w,v\in\mathcal{W}} \rho_{w,v}\psi^-_{w,v},$$

where H_n is the n-th harmonic number.

Sketch of proof. Notice that $\mathbb{E}[\Delta\mathcal{L}(\theta)]$ can be written as

$$\frac{2ck}{n} \sum_{w,v\in\mathcal{W}} \sum_{i=1}^{n} \big(\mathbb{E}[\mathrm{X}_{i,w}\mathrm{Y}_{i,v}] - \mathbb{E}[\mathrm{X}_{i,w}\mathrm{Y}_{n,v}]\big)\psi^-_{w,v}.$$

Because we have, for any i and j such that $i \leq j$,

$$\mathbb{E}[\mathrm{X}_{i,w}\mathrm{Y}_{j,v}] = \sum_{j'=1}^{j} \mathbb{E}[\mathrm{X}_{i,w}\frac{\mathrm{X}_{j',v}}{j}] = \mu_w\mu_v + \frac{\rho_{w,v}}{j},$$

plugging this into $\mathbb{E}[\Delta\mathcal{L}(\theta)]$ proves the theorem. See Appendix B.1 for the complete proof. □

Theorem 1 readily gives the convergence property of the first order moment of $\Delta\mathcal{L}(\theta)$:

Theorem 2. *The first-order moment of $\Delta\mathcal{L}(\theta)$ decreases in the order of $\mathcal{O}(\frac{\log(n)}{n})$:*

$$\mathbb{E}[\Delta\mathcal{L}(\theta)] = \mathcal{O}\bigg(\frac{\log(n)}{n}\bigg),$$

and thus converges to zero in the limit of infinity:

$$\lim_{n\to\infty} \mathbb{E}[\Delta\mathcal{L}(\theta)] = 0.$$

Proof. We have $H_n = \mathcal{O}(\log(n))$ from the upper integral bound, and thus Theorem 1 gives the proof. □

A similar result to Theorem 2 can be obtained for the second order moment of $\Delta\mathcal{L}(\theta)$ as well.

Theorem 3. *The second-order moment of $\Delta\mathcal{L}(\theta)$ decreases in the order of $\mathcal{O}(\frac{\log(n)}{n})$:*

$$\mathbb{E}[\Delta\mathcal{L}(\theta)^2] = \mathcal{O}\bigg(\frac{\log(n)}{n}\bigg),$$

and thus converges to zero in the limit of infinity:

$$\lim_{n\to\infty} \mathbb{E}[\Delta\mathcal{L}(\theta)^2] = 0.$$

Proof. Omitted. See Appendix B.2. □

4.2.2 Main result

The above theorems reveal the relationship between the optimal solutions of the two objectives, as stated in the next lemma.

Lemma 4. *Let θ^* and $\hat{\theta}$ be the optimal solutions of $\mathcal{L}_B(\theta)$ and $\mathcal{L}_I(\theta)$, respectively: $\theta^* = \arg\min_\theta \mathcal{L}_B(\theta)$ and $\hat{\theta} = \arg\min_\theta \mathcal{L}_I(\theta)$. Then,*

$$\lim_{n\to\infty} \mathbb{E}[\mathcal{L}_B(\hat{\theta}) - \mathcal{L}_B(\theta^*)] = 0, \tag{2}$$

$$\lim_{n\to\infty} \mathbb{V}[\mathcal{L}_B(\hat{\theta}) - \mathcal{L}_B(\theta^*)] = 0. \tag{3}$$

Proof. The proof is made by the squeeze theorem. Let $l = \mathcal{L}_\mathrm{B}(\hat{\theta}) - \mathcal{L}_\mathrm{B}(\theta^*)$. The optimality of θ^* gives $0 \leq l$. Also, the optimality of $\hat{\theta}$ gives

$$\begin{aligned} l &= \mathcal{L}_\mathrm{B}(\hat{\theta}) - \mathcal{L}_\mathrm{I}(\theta^*) + \mathcal{L}_\mathrm{I}(\theta^*) - \mathcal{L}_\mathrm{B}(\theta^*) \\ &\leq \mathcal{L}_\mathrm{B}(\hat{\theta}) - \mathcal{L}_\mathrm{I}(\hat{\theta}) + \mathcal{L}_\mathrm{I}(\theta^*) - \mathcal{L}_\mathrm{B}(\theta^*) \\ &= \Delta\mathcal{L}(\hat{\theta}) - \Delta\mathcal{L}(\theta^*). \end{aligned}$$

We thus have $0 \leq \mathbb{E}[l] \leq \mathbb{E}[\Delta\mathcal{L}(\hat{\theta}) - \Delta\mathcal{L}(\theta^*)]$. Since Theorem 2 implies that the right hand side converges to zero when $n \to \infty$, the squeeze theorem gives Eq. (2). Next, we have

$$\begin{aligned} \mathbb{V}[l] = \mathbb{E}[l^2] - \mathbb{E}[l]^2 &\leq \mathbb{E}[l^2] \\ &\leq \mathbb{E}[(\Delta\mathcal{L}(\hat{\theta}) - \Delta\mathcal{L}(\theta^*))^2] \\ &\leq \mathbb{E}[(\Delta\mathcal{L}(\hat{\theta}) - \Delta\mathcal{L}(\theta^*))^2] \\ &\quad + \mathbb{E}[(\Delta\mathcal{L}(\hat{\theta}) + \Delta\mathcal{L}(\theta^*))^2] \\ &= 2\mathbb{E}[\Delta\mathcal{L}(\hat{\theta})^2] + 2\mathbb{E}[\Delta\mathcal{L}(\theta^*)^2]. \end{aligned} \tag{4}$$

Theorem 3 suggests that Eq. (4) converges to zero when $n \to \infty$. Also, the non-negativity of the variance gives $0 \leq \mathbb{V}[l]$. Therefore, the squeeze theorem gives Eq. (3). □

We are now ready to provide the main result of the analysis. The next theorem shows the convergence of $\mathcal{L}_{\mathrm{B}}(\hat{\theta})$.

Theorem 5. *$\mathcal{L}_B(\hat{\theta})$ converges in probability to $\mathcal{L}_B(\theta^*)$:*

$$\forall \epsilon > 0, \lim_{n\to\infty} \Pr\Big[|\mathcal{L}_B(\hat{\theta}) - \mathcal{L}_B(\theta^*)| \geq \epsilon\Big] = 0.$$

Sketch of proof. Let again $l = \mathcal{L}_{\mathrm{B}}(\hat{\theta}) - \mathcal{L}_{\mathrm{B}}(\theta^*)$. Chebyshev's inequality gives, for any $\epsilon_1 > 0$,

$$\lim_{n\to\infty} \frac{\mathbb{V}[l]}{\epsilon_1^2} \geq \lim_{n\to\infty} \Pr\Big[|l - \mathbb{E}[l]| \geq \epsilon_1\Big].$$

Remember that Eq. (2) means that for any $\epsilon_2 > 0$, there exists n' such that if $n' \leq n$ then $|\mathbb{E}[l]| < \epsilon_2$. Therefore, we have

$$\lim_{n\to\infty} \frac{\mathbb{V}[l]}{\epsilon_1^2} \geq \lim_{n\to\infty} \Pr\Big[|l| \geq \epsilon_1 + \epsilon_2\Big] \geq 0.$$

The arbitrary property of ϵ_1 and ϵ_2 allows $\epsilon_1 + \epsilon_2$ to be rewritten as ϵ. Also, Eq. (3) implies that $\lim_{n\to\infty} \frac{\mathbb{V}[l]}{\epsilon_1^2} = 0$. This completes the proof. See Appendix B.3 for the detailed proof. □

Informally, this theorem can be interpreted as suggesting that the optimal solutions of batch and incremental SGNS agree when n is infinitely large.

4.3 Smoothed case

We next examine the smoothed case ($0 < \alpha < 1$). In this case, the noise distribution can be represented by using the ones in the unsmoothed case:

$$q_i(w) = \frac{f_i(w)^\alpha}{\sum_{w' \in \mathcal{W}} f_i(w')^\alpha} = \frac{\left(\frac{f_i(w)}{F_i}\right)^\alpha}{\sum_{w' \in \mathcal{W}} \left(\frac{f_i(w')}{F_i}\right)^\alpha}$$

where $F_i = \sum_{w' \in \mathcal{W}} f_i(w')$ and $\frac{f_i(w)}{F_i}$ corresponds to the unsmoothed noise distribution.

Definition 3. Let $\mathrm{Z}_{i,w}$ be a random variable that represents $q_i(w)$ in the smoothed case. Then, it can be written by using $\mathrm{Y}_{i,w}$:

$$\mathrm{Z}_{i,w} = g_w(\mathrm{Y}_{i,1}, \mathrm{Y}_{i,2}, \ldots, \mathrm{Y}_{i,|\mathcal{W}|})$$

where $g_w(x_1, x_2, \ldots, x_{|\mathcal{W}|}) = \frac{x_w^\alpha}{\sum_{w' \in \mathcal{W}} x_{w'}^\alpha}$.

Because $\mathrm{Z}_{i,w}$ is no longer a linear combination of $\mathrm{X}_{i,w}$, it becomes difficult to derive similar proofs to the unsmoothed case. To address this difficulty, $\mathrm{Z}_{i,w}$ is approximated by the first-order Taylor expansion around

$$\mathbb{E}[(\mathrm{Y}_{i,1}, \mathrm{Y}_{i,2}, \ldots, \mathrm{Y}_{i,|\mathcal{W}|})] = (\mu_1, \mu_2, \ldots, \mu_{|\mathcal{W}|}).$$

The first-order Taylor approximation gives

$$\mathrm{Z}_{i,w} \approx g_w(\mu) + \sum_{v \in \mathcal{W}} M_{w,v}(\mathrm{Y}_{i,v} - g_v(\mu))$$

where $\mu = (\mu_1, \mu_2, \ldots, \mu_{|\mathcal{W}|})$ and $M_{w,v} = \frac{\partial g_w(x)}{\partial x_v}|_{x=\mu}$. Consequently, it can be shown that the first and second order moments of $\Delta\mathcal{L}(\theta)$ have the order of $\mathcal{O}(\frac{\log(n)}{n})$ in the smoothed case as well. See Appendix C for the details.

4.4 Mini-batch SGNS

The same analysis result can also be obtained for the mini-batch SGNS. We can prove Theorems 2 and 3 in the mini-batch case as well (see Appendix D for the proof). The other part of the analysis remains the same.

5 Experiments

Three experiments were conducted to investigate the correctness of the theoretical analysis (Section 5.1) and the practical usefulness of incremental SGNS (Sections 5.2 and 5.3). Details of the experimental settings that do not fit into the paper are presented in Appendix E.

5.1 Validation of theorems

An empirical experiment was conducted to validate the result of the theoretical analysis. Since it is difficult to assess the main result in Section 4.2.2 directly, the theorems in Sections 4.2.1, from which the main result is readily derived, were investigated. Specifically, the first and second order moments of $\Delta\mathcal{L}(\theta)$ were computed on datasets of increasing sizes to empirically investigate the convergence property.

Datasets of various sizes were constructed from the English Gigaword corpus (Napoles et al., 2012). The datasets made up of n words were constructed by randomly sampling sentences from the Gigaword corpus. The value of n was varied over $\{10^3, 10^4, 10^5, 10^6, 10^7\}$. $10,000$ different datasets were created for each size n to compute the first and second order moments.

Figure 1 (top left) shows log-log plots of the first order moments of $\Delta\mathcal{L}(\theta)$ computed on the different sized datasets when $\alpha = 1.0$. The crosses

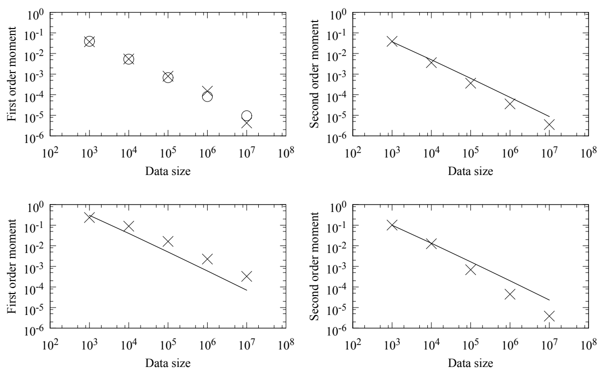


Figure 1: Log-log plots of the first and second order moments of $\Delta\mathcal{L}(\theta)$ on the different sized datasets when $\alpha = 1.0$ (top left and top right) and $\alpha = 0.75$ (bottom left and bottom right).

and circles represent the empirical values and theoretical values obtained by Theorem 1, respectively. Figure 1 (top right) similarly illustrates the second order moments of $\Delta\mathcal{L}(\theta)$. Since Theorem 3 suggests that the second order moment decreases in the order of $\mathcal{O}(\frac{\log(n)}{n})$, the graph $y \propto \frac{\log(x)}{x}$ is also shown. The graph was fitted to the empirical data by minimizing the squared error.

The top left figure demonstrates that the empirical values of the first order moments fit the theoretical result very well, providing a strong empirical evidence for the correctness of Theorem 1. In addition, the two figures show that the first and second order moments decrease almost in the order of $\mathcal{O}(\frac{\log(n)}{n})$, converging to zero as the data size increases. This result validates Theorems 2 and 3.

Figures 1 (bottom left) and (bottom right) show similar results when $\alpha = 0.75$. Since we do not have theoretical estimates of the first order moment when $\alpha \neq 1.0$, the graphs $y \propto \frac{\log(n)}{n}$ are shown in both figures. From these, we can again observe that the first and second order moments decrease almost in the order of $\mathcal{O}(\frac{\log(n)}{n})$. This indicates the validity of the investigation in Section 4.3. The relatively larger deviations from the graphs $y \propto \frac{\log(n)}{n}$, compared with the top right figure, are considered to be attributed to the first-order Taylor approximation.

5.2 Quality of word embeddings

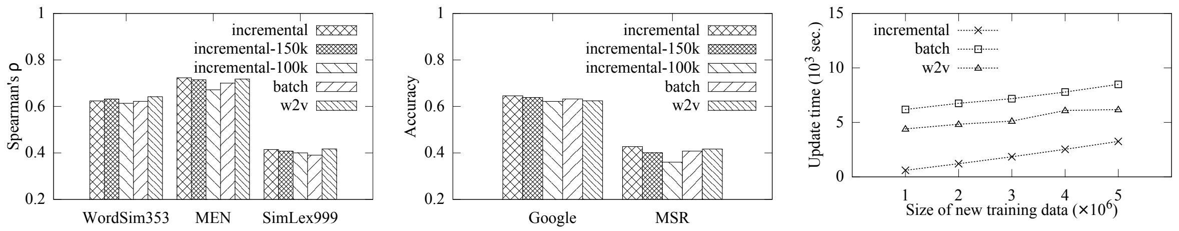
The next experiment investigates the quality of the word embeddings learned by incremental SGNS through comparison with the batch counterparts.

The Gigaword corpus was used for the training. For the comparison, both our own implementation of batch SGNS as well as WORD2VEC (Mikolov et al., 2013c) were used (denoted as **batch** and **w2v**). The training configurations of the three methods were set the same as much as possible, although it is impossible to do so perfectly. For example, incremental SGNS (denoted as **incremental**) utilized the dynamic vocabulary (*c.f.*, Section 3.2.1) and thus we set the maximum vocabulary size m to control the vocabulary size. On the other hand, we set a frequency threshold to determine the vocabulary size of **w2v**. We set $m = 240$k for **incremental**, while setting the frequency threshold to 100 for **w2v**. This yields vocabulary sets of comparable sizes: $220,389$ and $246,134$.

The learned word embeddings were assessed on five benchmark datasets commonly used in the literature (Levy et al., 2015): WordSim353 (Agirre et al., 2009), MEN (Bruni et al., 2013), SimLex-999 (Hill et al., 2015), the MSR analogy dataset (Mikolov et al., 2013c), the Google analogy dataset (Mikolov et al., 2013a). The former three are for a semantic similarity task, and the remaining two are for a word analogy task. As evaluation measures, Spearman's ρ and prediction accuracy were used in the two tasks, respectively.

Figures 2 (a) and (b) represent the results on the similarity datasets and the analogy datasets. We see that the three methods (**incremental**, **batch**, and **w2v**) perform equally well on all of the datasets. This indicates that incremental SGNS can learn as good word embeddings as the batch counterparts, while being able to perform incremental model update. Although **incremental** performs slightly better than the batch methods in some datasets, the difference seems to be a product of chance.

The figures also show the results of incremental SGNS when the maximum vocabulary size m was reduced to 150k and 100k (**incremental-150k** and **incremental-100k**). The resulting vocabulary sizes were $135,447$ and $86,993$, respectively. We see that **incremental-150k** and **incremental-100k** perform comparatively well with **incremental**, although relatively large performance drops are observed in some datasets (MEN and MSR). This demonstrates that the Misra-Gries algorithm can effectively control the vocabulary size.



(a) (b) (c)

Figure 2: (a): Spearman's ρ on the word similarity datasets. (b): Accuracy on the analogy datasets. (c): Update time when new training data is provided.

5.3 Update time

The last experiment investigates how much time incremental SGNS can save by avoiding re-training when updating the word embeddings.

In this experiment, **incremental** was first trained on the initial training data of size[5] n_1 and then updated on the new training data of size n_2 to measure the update time. For comparison, **batch** and **w2v** were re-trained on the combination of the initial and new training data. We fixed $n_1 = 10^7$ and varied n_2 over $\{1\times10^6, 2\times10^6, \ldots, 5\times10^6\}$. The experiment was conducted on Intel® Xeon® 2GHz CPU. The update time was averaged over five trials.

Figure 2 (c) compares the update time of the three methods across various values of n_2. We see that **incremental** significantly reduces the update time. It achieves 10 and 7.3 times speed-up compared with **batch** and **w2v** (when $n_2 = 10^6$). This represents the advantage of the incremental algorithm, as well as the time efficiency of the dynamic vocabulary and adaptive unigram table. We note that **batch** is slower than **w2v** because it uses AdaGrad, which maintains different learning rates for different dimensions of the parameter, while **w2v** uses the same learning rate for all dimensions.

6 Related Work

Word representations based on distributional semantics have been common (Turney and Pantel, 2010; Baroni and Lenci, 2010). The distributional methods typically begin by constructing a word-context matrix and then applying dimension reduction techniques such as SVD to obtain high-quality word meaning representations. Although some studies investigated incremental updating of the word-context matrix (Yin et al., 2015; Goyal and Daume III, 2011), they did not explore the reduced representations. On the other hand, neural word embeddings have recently gained much popularity as an alternative. However, most previous studies have not explored incremental strategies (Mikolov et al., 2013a,b; Pennington et al., 2014).

Peng et al. (2017) proposed an incremental learning method of hierarchical soft-max. Because hierarchical soft-max and negative sampling have different advantages (Peng et al., 2017), the incremental SGNS and their method are complementary to each other. Also, their updating method needs to scan not only new but also old training data, and thus is not an incremental algorithm in a strict sense. As a consequence, it potentially incurs the same time complexity as the re-training. Another consequence is that their method has to retain the old training data and thus wastes space, while incremental SGNS can discard old training examples after processing them.

Very recently, May et al. (2017) also proposed an incremental algorithm for SGNS. However, their work differs from ours in that their algorithm is not designed to use smoothed noise distribution (*i.e.*, the smoothing parameter α is assumed fixed as $\alpha = 1.0$ in their method), which is a key to learn high-quality word embeddings. Another difference is that they did not provide theoretical justification for their algorithm.

There are publicly available implementations for training SGNS, one of the most popular being WORD2VEC (Mikolov, 2013). However, it does not support an incremental training method. GENSIM (Řehůřek and Sojka, 2010) also offers SGNS training. Although GENSIM allows the incremental updating of SGNS models, it is done in an ad-hoc manner. In GENSIM, the vocabulary set as well as the unigram table are fixed once trained, meaning that new words cannot be added. Also,

[5] The number of sentences here.

they do not provide any theoretical accounts for the validity of their training method. Finally, we want to note that most of the existing implementations can be easily extended to support the incremental (or mini-batch) SGNS by simply keep updating the noise distribution.

7 Conclusion and Future Work

This paper proposed incremental SGNS and provided thorough theoretical analysis to demonstrate its validity. We also conducted experiments to empirically demonstrate its effectiveness. Although the incremental model update is often required in practical machine learning applications, only a little attention has been paid to learning word embeddings incrementally. We consider that incremental SGNS successfully addresses this situation and serves as an useful tool for practitioners.

The success of this work suggests several research directions to be explored in the future. One possibility is to explore extending other embedding methods such as GloVe (Pennington et al., 2014) to incremental algorithms. Such studies would further extend the potential of word embedding methods.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of NAACL*, pages 19–27.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computatoinal Linguistics*, 36:673–721.

E. Bruni, N. K. Tran, and M. Baroni. 2013. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–49.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Pavlos S. Efraimidis. 2015. Weighted random sampling over data streams. ArXiv:1012.0256.

Amit Goyal and Hal Daume III. 2011. Approximate scalable bounded space sketch for large data nlp. In *Proceedings of EMNLP*, pages 250–261.

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

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

Chandler May, Kevin Duh, Benjamin Van Durme, and Ashwin Lall. 2017. Streaming word embeddings with the space-saving algorithm. ArXiv:1704.07463.

Tomas Mikolov. 2013. word2vec. https://code.google.com/archive/p/word2vec.

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

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

Tomas Mikolov, Wen-Tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL*, pages 746–751.

Jayadev Misra and David Gries. 1982. Finding repeated elements. *Science of Computer Programming*, 2(2):143–152.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated english gigaword ldc2012t21.

Hao Peng, Jianxin Li, Yangqiu Song, and Yaopeng Liu. 2017. Incrementally learning the hierarchical softmax function for neural language models. In *Proceedings of AAAI*, pages 3267–3273.

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

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Jeffrey S. Vitter. 1985. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11:37–57.

Wenpeng Yin, Tobias Schnabel, and Hinrich Schütze. 2015. Online updating of word representations for part-of-speech tagging. In *Proceedings of EMNLP*, pages 1329–1334.

Learning to select data for transfer learning with Bayesian Optimization

Sebastian Ruder♠♣ **Barbara Plank**♡
♡Center for Language and Cognition, University of Groningen, The Netherlands
♠Insight Research Centre, National University of Ireland, Galway
♣Aylien Ltd., Dublin, Ireland
sebastian@ruder.io,b.plank@rug.nl

Abstract

Domain similarity measures can be used to gauge adaptability and select suitable data for transfer learning, but existing approaches define ad hoc measures that are deemed suitable for respective tasks. Inspired by work on curriculum learning, we propose to *learn* data selection measures using Bayesian Optimization and evaluate them across models, domains and tasks. Our learned measures outperform existing domain similarity measures significantly on three tasks: sentiment analysis, part-of-speech tagging, and parsing. We show the importance of complementing similarity with diversity, and that learned measures are—to some degree—transferable across models, domains, and even tasks.

1 Introduction

Natural Language Processing (NLP) models suffer considerably when applied in the wild. The distribution of the test data is typically very different from the data used during training, causing a model's performance to deteriorate substantially. Domain adaptation is a prominent approach to transfer learning that can help to bridge this gap; many approaches were suggested so far (Blitzer et al., 2007; Daumé III, 2007; Jiang and Zhai, 2007; Ma et al., 2014; Schnabel and Schütze, 2014). However, most work focused on one-to-one scenarios. Only recently research considered using multiple sources. Such studies are rare and typically rely on specific model transfer approaches (Mansour, 2009; Wu and Huang, 2016).

Inspired by work on curriculum learning (Bengio et al., 2009; Tsvetkov et al., 2016), we instead propose—to the best of our knowledge—the first model-agnostic *data selection* approach to transfer learning. Contrary to curriculum learning that aims at speeding up learning (see §6), we aim at *learning to select* the most relevant data from multiple sources using data metrics. While several measures have been proposed in the past (Moore and Lewis, 2010; Axelrod et al., 2011; Van Asch and Daelemans, 2010; Plank and van Noord, 2011; Remus, 2012), prior work is limited in studying metrics mostly in isolation, using only the notion of similarity (Ben-David et al., 2007) and focusing on a single task (see §6). Our hypothesis is that different tasks or even different domains demand different notions of similarity. In this paper we go beyond prior work by i) studying a range of similarity metrics, including diversity; and ii) testing the robustness of the learned weights across models (e.g., whether a more complex model can be approximated with a simpler surrogate), domains and tasks (to delimit the transferability of the learned weights).

The contributions of this work are threefold. First, we present the first model-independent approach to *learn* a data selection measure for transfer learning. It outperforms baselines across three tasks and multiple domains and is competitive with state-of-the-art domain adaptation approaches. Second, prior work on transfer learning mostly focused on similarity. We demonstrate empirically that diversity is as important as—and complements—domain similarity for transfer learning. Finally, we show—for the first time—to what degree learned measures *transfer* across models, domains and tasks.

2 Background: Transfer learning

Transfer learning generally involves the concepts of a domain and a task (Pan and Yang, 2010). A domain $\mathcal{D}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$ over $\mathcal{X}$,

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 372–382
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

where $X = \{x_1, \cdots, x_n\} \in \mathcal{X}$. For document classification with a bag-of-words, $\mathcal{X}$ is the space of all document vectors, x_i is the i-th document vector, and X is a sample of documents.

Given a domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task $\mathcal{T}$ consists of a label space $\mathcal{Y}$ and a conditional probability distribution $P(Y|X)$ that is typically learned from training data consisting of pairs $\{x_i, y_i\}$, where $x_i \in X$ and $y_i \in Y$.

Finally, given a source domain $\mathcal{D}_S$, a corresponding source task $\mathcal{T}_S$, as well as a target domain $\mathcal{D}_T$ and a target task $\mathcal{T}_T$, transfer learning seeks to facilitate the learning of the target conditional probability distribution $P(Y_T|X_T)$ in $\mathcal{D}_T$ with the information gained from $\mathcal{D}_S$ and $\mathcal{T}_S$ where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$. We will focus on the scenario where $\mathcal{D}_S \neq \mathcal{D}_T$ assuming that $\mathcal{T}_S = \mathcal{T}_T$, commonly referred to as domain adaptation. We investigate transfer across tasks in §5.3.

Existing research in domain adaptation has generally focused on the scenario of one-to-one adaptation: Given a set of source domains A and a set of target domains B, a model is evaluated based on its ability to adapt between all pairs (a, b) in the Cartesian product $A \times B$ where $a \in A$ and $b \in B$ (Remus, 2012). However, adaptation between two dissimilar domains is often undesirable, as it may lead to negative transfer (Rosenstein et al., 2005). Only recently, many-to-one adaptation (Mansour, 2009; Wu and Huang, 2016) has received some attention, as it replicates the realistic scenario of multiple source domains where performance on the target domain is the foremost objective.

3 Data selection model

In order to select training data for adaptation for a task $\mathcal{T}$, existing approaches rank the available n training examples $X = \{x_1, x_2, \cdots, x_n\}$ of k source domains $D = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_k\}$ according to a domain similarity measure $\mathcal{S}$ and choose the top m samples for training their algorithm. While this has been shown to work empirically (Moore and Lewis, 2010; Axelrod et al., 2011; Plank and van Noord, 2011; Van Asch and Daelemans, 2010; Remus, 2012), using a pre-existing metric leaves us unable to adapt to the characteristics of our task $\mathcal{T}$ and target domain $\mathcal{D}_T$ and foregoes additional knowledge that may be gleaned from the interaction of different metrics. For this reason, we propose to *learn* the following linear domain similarity measure $\mathcal{S}$ as a linear combination of feature values:

$$\mathcal{S} = \phi(X) \cdot w^\intercal \tag{1}$$

where $\phi(X) \in \mathbb{R}^{n \times l}$ are the similarity and diversity features further described in §3.2 for each training example, with l being the number of features, while $w \in \mathbb{R}^l$ are the weights learned by Bayesian Optimization.

We aim to learn weights w in order to optimize the objective function J of the respective task $\mathcal{T}$ on a small number of validation examples of the corresponding target domain $\mathcal{D}_T$.

3.1 Bayesian Optimization for data selection

As the learned measure $\mathcal{S}$ should be agnostic of the particular objective function J, we cannot use gradient-based methods for optimization. Similar to Tsvetkov et al. (2016), we use Bayesian Optimization (Brochu et al., 2010), which has emerged as an efficient framework to optimize any function. For instance, it has repeatedly found better settings of neural network hyperparameters than domain experts (Snoek et al., 2012).

Given a black-box function $f : \mathbb{X} \rightarrow \mathbb{R}$, Bayesian Optimization aims to find an input $\hat{x} \in \arg\min_{x \in \mathbb{X}} f(x)$ that globally minimizes f. For this, it requires a prior $p(f)$ over the function and an acquisition function $a_{p(f)} : \mathbb{X} \rightarrow \mathbb{R}$ that calculates the *utility* of any evaluation at any x.

Bayesian Optimization then proceeds iteratively. At iteration t, 1) it finds the most promising input $x_t \in \arg\max a_p(x)$ through numerical optimization; 2) it then evaluates the surrogate function $y_t \sim f(x_t) + \mathcal{N}(0, \sigma^2)$ on this input and adds the resulting data point (x_t, y_t) to the set of observations $\mathcal{O}_{t-1} = (x_j, y_j)_{j=1 \dots t-1}$; 3) finally, it updates the prior $p(f|\mathcal{O}_t)$ and the acquisition function $a_{p(f|\mathcal{O}_t)}$.

For data selection, the black-box function f looks as follows: 1) It takes as input a set of weights w that should be evaluated; 2) the training examples of all source domains are then scored and sorted according to Equation 1; 3) the model for the respective task $\mathcal{T}$ is trained on the top n samples; 4) the model is evaluated on the validation set according to the evaluation measure J and the value of J is returned.

Gaussian Processes (GP) are a popular choice for $p(f)$ due to their descriptive power (Rasmussen, 2006). We use GP with Monte Carlo acquistion and Expected Improvement (EI)

(Močkus, 1974) as acquisition function as this combination has been shown to outperform comparable approaches (Snoek et al., 2012).[1]

3.2 Features

Existing work on data selection for domain adaptation selects data based on its *similarity* to the target domain. Several measures have been proposed in the literature (Van Asch and Daelemans, 2010; Plank and van Noord, 2011; Remus, 2012), but were so far only used in isolation.

Only selecting training instances with respect to the target domain also fails to account for instances that are richer and better suited for knowledge acquisition. For this reason, we consider—to our knowledge for the first time—whether intrinsic qualities of the training data accounting for *diversity* are of use for domain adaptation in NLP.

Similarity We use a range of similarity metrics. Some metrics might be better suited for some tasks, while different measures might capture complementary information. We thus use the following measures as features for learning a more effective domain similarity metric.

We define similarity features over probability distributions in accordance with existing literature (Plank and van Noord, 2011). Let P be the representation of a source training example, while Q is the corresponding target domain representation. Let further $M = \frac{1}{2}(P + Q)$, i.e. the average distribution between P and Q and let $D_{KL}(P||Q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$, i.e., the KL divergence between the two domains. We do not use D_{KL} as a feature as it is undefined for distributions where some event $q_i \in Q$ has probability 0, which is common for term distributions. Our features are:

- Jensen-Shannon divergence (Lin, 1991): $\frac{1}{2}[D_{KL}(P||M) + D_{KL}(Q||M)]$. Jensen-Shannon divergence is a smoothed, symmetric variant of D_{KL} that has been successfully used for domain adaptation (Plank and van Noord, 2011; Remus, 2012).
- Rényi divergence (Rényi, 1961): $\frac{1}{\alpha-1} \log(\sum_{i=1}^{n} \frac{p_i^{\alpha}}{q_i^{\alpha-1}})$. Rényi divergence reduces to D_{KL} if $\alpha = 1$. We set $\alpha = 0.99$ following Van Asch and Daelemans (2010).
- Bhattacharyya distance (Bhattacharya, 1943): $\ln(\sum_i \sqrt{P_i Q_i})$
- Cosine similarity (Lee, 2001): $\frac{P \cdot Q}{\|P\| \, \|Q\|}$. We can treat the distributions alternatively as vectors and consider geometrically motivated distance functions such as cosine similarity as well as the following.
- Euclidean distance (Lee, 2001): $\sqrt{\sum_i (P_i - Q_i)^2}$.
- Variational dist. (Lee, 2001): $\sum_i |P_i - Q_i|$.

We consider three different representations for calculating the above domain similarity measures:

- Term distributions (Plank and van Noord, 2011): $t \in \mathbb{R}^{|V|}$ where t_i is the probability of the i-th word in the vocabulary V.
- Topic distributions (Plank and van Noord, 2011): $t \in \mathbb{R}^n$ where t_i is the probability of the i-th topic as determined by an LDA model (Blei et al., 2003) trained on the data and n is the number of topics.
- Word embeddings (Mikolov et al., 2013): $\frac{1}{n} \sum_i v_{w_i} \sqrt{\frac{a}{p(w_i)}}$ where n is the number of words with embeddings in the document, v_{w_i} is the pre-trained embedding of the i-th word, $p(w_i)$ its probability, and a is a smoothing factor used to discount frequent probabilities. A similar weighted sum has recently been shown to outperform supervised approaches for other tasks (Arora et al., 2017). As embeddings may be negative, we use them only with the latter three geometric features above.

Diversity For each training example, we calculate its diversity based on the words in the example. Let p_i and p_j be probabilities of the word types t_i and t_j in the training data and $\cos(v_{t_i}, v_{t_j})$ the cosine similarity between their word embeddings. We employ measures that have been used in the past for measuring diversity (Tsvetkov et al., 2016):

- Number of word types: $\#types$.
- Type-token ratio: $\frac{\#types}{\#tokens}$.
- Entropy (Shannon, 1948): $-\sum_i p_i \ln(p_i)$.
- Simpson's index (Simpson, 1949): $-\sum_i p_i^2$.
- Rényi entropy (Rényi, 1961): $\frac{1}{\alpha-1} \log(\sum_i p_i^{\alpha})$
- Quadratic entropy (Rao, 1982): $\sum_{i,j} \cos(v_{t_i}, v_{t_j}) p_i p_j$.

[1] We also experimented with FABOLAS (Klein et al., 2017), but found its ability to adjust the training set size during optimization to be inconclusive for our relatively small training sets.

4 Experiments

4.1 Tasks, datasets, and models

We evaluate our approach on three tasks: sentiment analysis, part-of speech (POS) tagging, and dependency parsing. We use the n examples with the highest score as determined by the learned data selection measure for training our models.[2] We show statistics for all datasets in Table 1.

Sentiment Analysis For sentiment analysis, we evaluate on the Amazon reviews dataset (Blitzer et al., 2006). We use tf-idf-weighted unigram and bigram features and a linear SVM classifier (Blitzer et al., 2007). We set the vocabulary size to 10,000 and the number of training examples $n = 1600$ to conform with existing approaches (Bollegala et al., 2011) and stratify the training set.

POS tagging For POS tagging and parsing, we evaluate on the coarse-grained POS data (12 universal POS) of the SANCL 2012 shared task (Petrov and McDonald, 2012). Each domain—except for WSJ—contains around 2000-5000 labeled sentences and more than 100,000 unlabeled sentences. In the case of WSJ, we use its dev and test data as labeled samples and treat the remaining sections as unlabeled. We set $n = 2000$ for POS tagging and parsing to retain enough examples for the most-similar-domain baseline.

To evaluate the impact of model choice, we compare two models: a Structured Perceptron (in-house implementation with commonly used features pertaining to tags, words, case, prefixes, as well as prefixes and suffixes) trained for 5 iterations with a learning rate of 0.2; and a state-of-the-art Bi-LSTM tagger (Plank et al., 2016) with word and character embeddings as input. We perform early stopping on the validation set with patience of 2 and use otherwise default hyperparameters[3] as provided by the authors.

Parsing For parsing, we evaluate the state-of-the-art Bi-LSTM parser by Kiperwasser and Goldberg (2016) with default hyperparameters.[4] We use the same domains as used for POS tagging, i.e., the dependency parsing data with gold POS as made available in the SANCL 2012 shared task.[5]

$\mathcal{T}$	Domain	# labeled	# unlabeled
Sentiment	Book	2000	4465
	DVD	2000	3586
	Electronics	2000	5681
	Kitchen	2000	5945
POS/Parsing	Answers	3489	27274
	Emails	4900	1194173
	Newsgroups	2391	1000000
	Reviews	3813	1965350
	Weblogs	2031	524834
	WSJ	2976	30060

Table 1: Number of labeled and unlabeled sentences for each domain in the Amazon Reviews dataset (Blitzer et al., 2006) (above) and the SANCL 2012 dataset (Petrov and McDonald, 2012) for POS tagging and parsing (below).

4.2 Training details

In practice, as feature values occupy different ranges, we have found it helpful to apply z-normalisation similar to Tsvetkov et al. (2016). We moreover constrain the weights w to $[-1, 1]$.

For each dataset, we treat each domain as target domain and all other domains as source domains. Similar to Bousmalis et al. (2016), we chose to use a small number (100) target domain examples as validation set. We optimize each similarity measure using Bayesian Optimization with 300 iterations according to the objective measure J of each task (accuracy for sentiment analysis and POS tagging; LAS for parsing) with respect to the validation set of the corresponding target domain.

Unlabeled data is used in addition to calculate the representation of the target domain and to calculate the source domain representation for the most similar domain baseline. We train an LDA model (Blei et al., 2003) with 50 topics and 10 iterations for topic distribution-based representations and use GloVe embeddings (Pennington et al., 2014) trained on 42B tokens of Common Crawl data[6] for word embedding-based representations.

For sentiment analysis, we conduct 10 runs of each feature set for every domain and report mean and variance. For POS tagging and parsing, we observe that variance is low and perform one run while retaining random seeds for reproducibility.

[2] All code is available at https://github.com/sebastianruder/learn-to-select-data.

[3] https://github.com/bplank/bilstm-aux

[4] https://github.com/elikip/bist-parser

[5] We leave investigating the effect of the adapted taggers on parsing for future work.

[6] https://nlp.stanford.edu/projects/glove/

		Target domains			
	Feature set	**Book**	**DVD**	**Electronics**	**Kitchen**
Base	Random	71.17 (± 4.41)	70.51 (± 3.33)	76.75 (± 1.77)	77.94 (± 3.72)
	Jensen-Shannon divergence – examples	72.51 (± 0.42)	68.21 (± 0.34)	76.51 (± 0.63)	77.47 (± 0.44)
	Jensen-Shannon divergence – domain	75.26 (± 1.25)	73.74 (± 1.36)	72.60 (± 2.19)	80.01 (± 1.93)
Learned measures	Similarity (word embeddings)	75.06 (± 1.38)	74.96 (± 2.12)	80.79 (± 1.31)	83.45 (± 0.96)
	Similarity (term distributions)	75.39 (± 0.98)	76.25 (± 0.96)	81.91 (± 0.57)	83.39 (± 0.84)
	Similarity (topic distributions)	76.04 (± 1.10)	75.89 (± 0.81)	81.69 (± 0.96)	83.09 (± 0.95)
	Diversity	76.03 (± 1.28)	77.48 (± 1.33)	81.15 (± 0.67)	83.94 (± 0.99)
	Sim (term dists) + sim (topic dists)	75.76 (± 1.30)	76.62 (± 0.95)	81.73 (± 0.63)	83.43 (± 0.75)
	Sim (word embeddings) + diversity	75.52 (± 0.98)	77.50 (± 0.61)	80.97 (± 0.83)	84.28 (± 1.02)
	Sim (term dists) + diversity	76.20 (± 1.45)	77.60 (± 1.01)	**82.67** (± 0.73)	**84.98** (± 0.60)
	Sim (topic dists) + diversity	**77.16** (± 0.77)	**79.00** (± 0.93)	81.92 (± 1.32)	84.29 (± 1.00)
	All source data (6,000 training examples)	70.86 (± 0.51)	68.74 (± 0.32)	77.39 (± 0.32)	73.09 (± 0.41)

Table 2: Accuracy scores for data selection for sentiment analysis domain adaptation on the Amazon reviews dataset (Blitzer et al., 2006). Best: bold; second-best: underlined.

4.3 Baselines and features

We compare the learned measures to three baselines: i) a random baseline that randomly selects n training samples from all source domains (*random*); ii) the top n examples selected using Jensen-Shannon divergence (*JS – examples*), which outperformed other measures in previous work (Plank and van Noord, 2011; Remus, 2012); iii) n examples randomly selected from the most similar source domain determined by Jensen-Shannon divergence (*JS – domain*). We additionally compare against training on all available source data (6,000 examples for sentiment analysis; 14,700-17,569 examples for POS tagging and parsing depending on the target domain).

We optimize data selection using Bayesian Optimization with every feature set: similarity features respectively based on i) word embeddings, ii) term distributions, and iii) topic distributions; and iv) diversity features. In addition, we investigate how well different representations help each other by using similarity features with the two best-performing representations, term distributions and topic distributions. Finally, we explore whether diversity and similarity-based features complement each other by in turn using each similarity-based feature set together with diversity features.

5 Results

Sentiment analysis We show results for sentiment analysis in Table 2. First of all, the baselines show that the sentiment review domains are clearly delimited. Adapting between two similar domains such as Book and DVD is more productive than adaptation between dissimilar domains, e.g. Books and Electronics, as shown in previous work (Blitzer et al., 2007). This explains the strong performance of the most-similar-domain baseline. In contrast, selecting individual examples based on a domain similarity measure performs only as good as chance. Thus, when domains are more clear-cut, selecting from the closest domain is a stronger baseline than selecting from the entire pool of source data.

If we learn a data selection measure using Bayesian Optimization, we are able to outperform the baselines with almost all feature sets. Performance gains are considerable for all domains with individual feature sets (term similarity, word embeddings similarity, diversity and topic similarity), except for Books were improvements for some single feature sets are smaller. Term distributions and topic distributions are the best-performing representations for calculating similarity, with term distributions performing slightly better across all domains. Combining term distribution-based and topic distribution-based features only provides marginal gains over the individual feature sets, demonstrating that most of the information is contained in the similarity features rather than the representations.

Diversity features perform comparatively to the best similarity features and outperform them on two domains. Furthermore, the combination of diversity and similarity features yields another sizable gain of around 1 percentage point for almost all domains over the best similarity features, which shows that diversity and similarity features capture complementary information. Term distribution and topic distribution-based similarity features in

	Trg domains →	Answers			Emails			Newsgroups			Reviews			Weblogs			WSJ		
	Task →	POS		Pars	POS		Pars	POS		Pars	POS		Pars	POS		Pars	POS		Pars
	Feat ↓ Model →	P	B	BIST	P	B	BIST	P	B	BIST	P	B	BIST	P	B	BIST	P	B	BIST
Base	Random	91.34	92.55	81.02	91.80	93.25	79.09	92.50	93.26	80.61	92.08	92.12	82.30	92.76	93.03	82.39	91.08	92.54	78.31
	JS – examples	92.42	93.16	82.80	91.75	93.77	80.53	92.96	94.29	83.25	92.77	93.32	84.35	94.33	94.92	85.36	92.85	94.08	82.43
	JS – domain	90.84	91.13	80.37	91.64	93.16	79.93	92.23	92.67	81.77	92.27	92.67	82.11	93.19	94.34	83.44	91.20	92.99	80.61
Learned measures	W2v sim	92.53	93.22	82.74	92.94	94.14	81.18	93.41	94.09	81.62	93.51	93.30	82.98	94.41	94.83	84.30	93.02	94.66	81.57
	Term sim	**93.13**	93.43	**83.79**	92.96	94.04	81.09	93.58	**94.55**	82.68	93.53	**93.73**	**84.66**	94.42	**95.09**	84.85	**93.44**	94.11	82.57
	Topic sim	92.50	93.16	82.87	92.70	**94.48**	81.43	93.97	94.09	82.07	93.21	93.22	83.98	94.42	93.71	84.98	93.09	94.02	**82.90**
	Diversity	92.33	92.58	83.01	93.08	93.56	80.93	**94.37**	93.97	83.98	93.33	93.05	83.92	94.62	94.94	85.84	93.33	93.44	82.80
	Term+topic sim	92.80	**93.69**	82.87	92.70	92.28	81.13	93.57	93.76	82.97	**93.56**	93.61	84.65	94.41	94.23	84.43	93.07	**94.68**	82.43
	W2v sim+div	92.76	92.38	82.34	**93.51**	94.19	80.77	93.96	94.10	**84.26**	93.45	93.39	84.47	94.36	94.95	85.53	93.32	93.20	82.32
	Term sim+div	92.73	93.46	83.72	92.90	93.81	**81.60**	94.03	93.47	82.80	93.47	93.29	84.62	**94.76**	95.06	85.44	93.32	93.68	82.87
	Topic sim+div	92.93	93.62	82.60	92.62	93.93	80.83	93.85	94.06	84.04	93.16	93.59	84.45	94.42	94.45	**85.89**	93.38	94.23	82.33
	All source data	94.30	95.16	86.34	94.34	95.90	85.57	95.40	95.90	87.18	94.90	95.03	87.51	95.53	95.79	88.23	94.19	95.64	85.20

Table 3: Results for data selection for part-of-speech tagging and parsing domain adaptation on the SANCL 2012 shared task dataset (Petrov and McDonald, 2012). POS: Part-of-speech tagging. Pars: Parsing. POS tagging models: Structured Perceptron (P); Bi-LSTM tagger (B) (Plank et al., 2016). Parsing model: Bi-LSTM parser (BIST) (Kiperwasser and Goldberg, 2016). Evaluation metrics: Accuracy (POS tagging); Labeled Attachment Score (parsing). Best: bold; second-best: underlined.

conjunction with diversity features finally yield the best performance, outperforming the baselines by 2-6 points in absolute accuracy.

Finally, we compare data selection to training on all available source data (in this setup, 6,000 instances). The result complements the findings of the most-similar baseline: as domains are dissimilar, training on all available sources is detrimental.

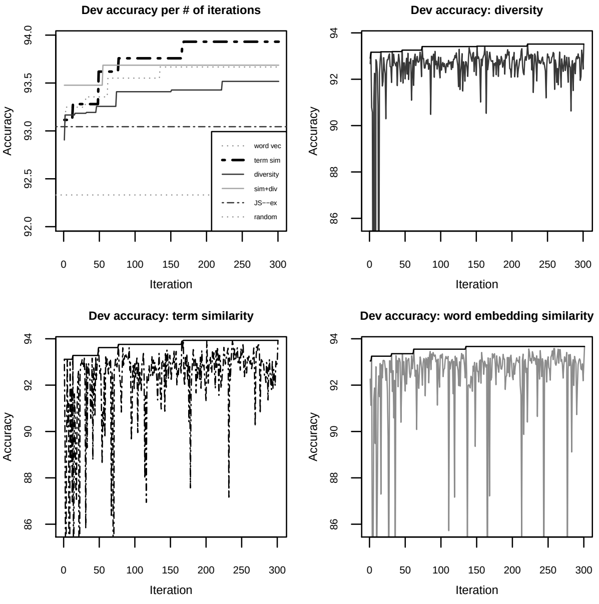


Figure 1: Dev accuracy curves of Bayes Optimization for POS tagging on the Reviews domain. Best dev acc for different feature sets (top-left). Best dev acc vs. exploration (top-right, bottom).

POS tagging Results for POS tagging are given in Table 3. Using Bayesian Optimization, we are able to outperform the baselines with almost all feature sets, except for a few cases (e.g., diversity and word embeddings similarity, topic and term distributions). Overall term distribution-based similarity emerges as the most powerful individual feature. Combining it with diversity does not prove as beneficial as in the sentiment analysis case, however, often yields the second-best results.

Notice that for POS tagging/parsing, in contrast to sentiment analysis, the most-similar domain baseline is not effective, it often performs only as good as chance, or even hurts. In contrast, the baseline that selects instances (*JS – examples*) rather than a domain performs better. This makes sense as in SA topically closer domains express sentiment in more similar ways, while for POS tagging having more varied training instances is intuitively more beneficial. In fact, when inspecting the domain distribution of our approach, we find that the best SA model chooses more instances from the closest domain, while for POS tagging instances are more balanced across domains. This suggests that the Web treebank domains are less clear-cut. In fact, training a model on all sources, which is considerably more and varied data (in this setup, 14-17.5k training instances) is beneficial. This is in line with findings in machine translation (Mirkin and Besacier, 2014), which show that similarity-based selection works best if domains are very different. Results are thus less pronounced for POS tagging, and we leave experimenting with larger n for future work.

To gain some insight into the optimization procedure, Figure 1 shows the development accuracy for the Structured Perceptron for an example domain. The top-right and bottom graphs show the hypothesis space exploration of Bayesian Optimization for different single feature sets, while the

	Target domains											
	Answers		Emails		Newsgroups		Reviews		Weblogs		WSJ	
Feature set ↓ $\mathcal{M}_S$ →	**B**	**P**$_{proxy}$	**B**	**P**$_{proxy}$	**B**	**P**$_{proxy}$	**B**	**P**$_{proxy}$	**B**	**P**$_{proxy}$	**B**	**P**$_{proxy}$
Term similarity	93.43	93.67	94.04	93.88	94.55	93.77	**93.73**	93.54	**95.09**	95.06	94.11	94.30
Diversity	92.58	93.19	93.56	**94.40**	93.97	**94.96**	93.05	93.52	94.94	94.91	93.44	**94.14**
Term similarity+diversity	**93.46**	93.18	93.81	94.29	93.47	94.28	93.29	93.35	95.06	94.67	93.68	93.92

Table 4: Accuracy scores for cross-model transfer of learned data selection weights for part-of-speech tagging from Structured Perceptron (P$_{proxy}$) to Bi-LSTM tagger (B) (Plank et al., 2016) on the SANCL 2012 shared task dataset (Petrov and McDonald, 2012). Data selection weights are learned using model $\mathcal{M}_S$; Bi-LSTM tagger (B) is then trained using the learned weights. Better than baselines: underlined.

top-left graph displays the overall best dev accuracy for different features. We observe again that term similarity is among the best feature sets and results in a larger explored space (more variance), in contrast to the diversity features whose development accuracy increases less and results in an overall less explored space. Exploration plots for other features/models looks similar.

Parsing The results for parsing are given in Table 3. Diversity features are stronger than for POS tagging and outperform the baselines for all except the Reviews domain. Similarly to POS tagging, term distribution-based similarity as well as its combination with diversity features yield the best results across most domains.

5.1 Transfer across models

In addition, we are interested how well the metric learned for one target domain transfers to other settings. We first investigate its ability to transfer to another model. In practice, a metric can be learned using a model that is cheap to evaluate and serves as proxy for a state-of-the-art model, in a way similar to uptraining (Petrov et al., 2010). For this, we employ the data selection features learned using the Structured Perceptron model for POS tagging and use them to select data for the Bi-LSTM tagger. The results in Table 4 indicate that cross-model transfer is indeed possible, with most transferred feature sets achieving similar results or even outperforming features learned with the Bi-LSTM. In particular, transferred diversity significantly outperforms its in-model equivalent. This is encouraging, as it allows to learn a data selection metric using less complex models.

5.2 Transfer across domains

We explore whether data selection parameters learned for one target domain transfer to other target domains. For each domain, we use the weights with the highest performance on the validation set and use them for data selection with the remaining domains as target domains. We conduct 10 runs for the best-performing feature sets for sentiment analysis and report the average accuracy scores in Table 5 (for POS tagging, see Table 6).

		Target domains			
Feature	$\mathcal{D}_S$	**B**	**D**	**E**	**K**
Sim	B	**75.39**	75.22	80.74	80.41
Sim	D	75.30	76.25	**82.68**	82.29
Sim	E	74.55	76.65	81.91	82.23
Sim	K	73.64	**76.66**	81.09	**83.39**
Div	B	**76.03**	75.16	80.16	80.01
Div	D	75.68	**77.48**	65.74	72.48
Div	E	74.69	76.60	**81.15**	81.97
Div	K	75.03	76.23	80.71	**83.94**
Sim+div	B	**76.20**	64.81	65.06	79.87
Sim+div	D	74.17	77.60	**83.26**	**85.19**
Sim+div	E	74.14	**79.32**	82.67	84.53
Sim+div	K	75.54	76.11	78.72	84.98
SDAMS	-	78.29	79.13	84.06	86.29

Table 5: Accuracy scores for cross-domain transfer of learned data selection weights on Amazon reviews (Blitzer et al., 2006). $\mathcal{D}_S$: target domain used for learning metric $\mathcal{S}$. B: Book. D: DVD. E: Electronics. K: Kitchen. Sim: term distribution-based similarity. Div: diversity. Best per feature set: bold. In-domain results: gray. SDAMS (Wu and Huang, 2016) listed as comparison.

The transfer of the weights learned with Bayesian Optimization is quite robust in most cases. Feature sets like Similarity or Diversity trained on Books outperform the strong JS – $\mathcal{D}$ baseline in all 6 cases, for Electronics and Kitchen in 4/6 cases (off-diagonals for box 2 and 3 in Table 5). In some cases, the transferred weights even outperform the data selection metric learned for the respective domain, such as on D->E with sim and sim+div features and by almost 2 pp on E->D.

		Target domains					
Feature set	$\mathcal{D}_{\mathcal{S}}$	**Answers (A)**	**Emails (E)**	**Newsgroups (N)**	**Reviews (R)**	**Weblogs (W)**	**WSJ**
Term similarity	A	**93.13**	91.60	93.94	**93.63**	94.26	92.42
Term similarity	E	92.35	**92.96**	93.42	**93.63**	93.75	92.24
Term similarity	N	92.48	92.28	93.58	93.35	93.95	93.00
Term similarity	R	92.06	92.18	93.38	93.53	94.26	91.88
Term similarity	W	92.69	92.12	**93.65**	93.12	**94.42**	92.63
Term similarity	WSJ	92.50	92.51	93.53	93.00	94.29	**93.44**
Diversity	A	92.33	92.14	93.46	92.00	94.01	92.56
Diversity	E	92.11	**93.08**	93.81	92.67	94.16	93.13
Diversity	N	**92.67**	92.22	**94.37**	92.44	94.05	92.96
Diversity	R	92.65	92.72	93.67	**93.33**	94.18	93.28
Diversity	W	92.19	92.31	93.31	92.20	**94.62**	92.04
Diversity	WSJ	92.26	92.31	93.75	92.70	94.32	**93.33**
Term similarity+diversity	A	92.73	92.63	93.16	92.58	93.88	92.23
Term similarity+diversity	E	92.55	92.90	93.78	92.73	93.54	92.57
Term similarity+diversity	N	92.47	92.27	**94.03**	92.63	94.30	93.14
Term similarity+diversity	R	**92.80**	**93.11**	93.92	93.47	93.79	92.99
Term similarity+diversity	W	92.61	92.45	93.44	**93.52**	**94.76**	93.26
Term similarity+diversity	WSJ	91.82	92.37	93.52	92.63	94.17	**93.32**

Table 6: Accuracy scores for cross-domain transfer of learned data selection weights for part-of-speech tagging with the Structured Perceptron model on the SANCL 2012 shared task dataset (Petrov and McDonald, 2012). $\mathcal{D}_{\mathcal{S}}$: target domain used for learning metric $\mathcal{S}$. Best: bold. In-domain results: gray.

		Target tasks		
Feature set	$\mathcal{T}_{\mathcal{S}}$	**POS**	**Pars**	**SA**
Sim	POS	93.51	83.11	74.19
Sim	Pars	92.78	83.27	72.79
Sim	SA	86.13	67.33	79.23
Div	POS	93.51	83.11	69.78
Div	Pars	93.02	83.41	68.45
Div	SA	90.52	74.68	79.65
Sim+div	POS	93.54	83.24	69.79
Sim+div	Pars	93.11	83.51	72.27
Sim+div	SA	89.80	75.17	80.36

Table 7: Results of cross-task transfer of learned data selection weights. $\mathcal{T}_{\mathcal{S}}$: task used for learning metric $\mathcal{S}$. POS: Part-of-speech tagging. Pars: Parsing. SA: sentiment analysis. Accuracy scores for SA and POS; LAS Attachment Score for parsing. Models: Structured Perceptron (POS tagging); Bi-LSTM parser (Kiperwasser and Goldberg, 2016) (Pars). Same features as in Table 5. In-task results: gray. Better than base: underlined.

Transferred similarity+diversity features mostly achieve higher performance than other feature sets, but the higher number of parameters runs the risk of overfitting to the domain as can be observed with two instances of negative transfer with sim+div features.

As a reference, we also list the performance of the state-of-the-art multi-domain adaptation approach (Wu and Huang, 2016), which shows that task-independent data selection is in fact competitive with a task-specific, heuristic state-of-the-art domain adaptation approach. In fact our *transferred* similarity+diversity feature (E->D) outperforms the state-of-the-art (Wu and Huang, 2016) on DVD. This is encouraging as previous work (Remus, 2012) has shown that data selection and domain adaptation can be complementary.

5.3 Transfer across tasks

We finally investigate whether data selection is task-specific or whether a metric learned on one task can be transferred to another one. For each feature set, we use the learned weights for each domain in the source task (for sentiment analysis, we use the best weights on the validation set; for POS tagging, we use the Structured Perceptron model) and run experiments with them for all domains in the target task.[7] We report the averaged accuracy scores for transfer across all tasks in Table 7.

Transfer is productive between related tasks, i.e. POS tagging and parsing results are similar to those obtained with data selection learned for the particular task. We observe large drops in performance for transfer between unrelated tasks, such

[7] E.g., for SA->POS, for each feature set, we obtain one set of weights for each of 4 SA domains, which we use to select data for the 6 POS domains, yielding $4 \cdot 6 = 24$ results.

as sentiment analysis and POS tagging, which is expected since these are very different tasks. Between related tasks, the combination of similarity and diversity features achieves the most robust transfer and outperforms the baselines in both cases. This suggests that even in the absence of target task data, we only require data of a related task to learn a successful data selection measure.

6 Related work

Most prior work on data selection for transfer learning focuses on phrase-based machine translation. Typically language models are leveraged via perplexity or cross-entropy scoring to select target data (Moore and Lewis, 2010; Axelrod et al., 2011; Duh et al., 2013; Mirkin and Besacier, 2014). A recent study investigates data selection for neural machine translation (van der Wees et al., 2017). Perplexity was also used to select training data for dependency parsing (Søgaard, 2011), but has been found to be less suitable for tasks such as sentiment analysis (Ruder et al., 2017). In general, there are fewer studies on data selection for other tasks, e.g., constituent parsing (McClosky et al., 2010), dependency parsing (Plank and van Noord, 2011; Søgaard, 2011) and sentiment analysis (Remus, 2012). Work on predicting task accuracy is related, but can be seen as complementary (Ravi et al., 2008; Van Asch and Daelemans, 2010).

Many domain similarity metrics have been proposed. Blitzer et al. (2007) show that proxy $\mathcal{A}$ distance can be used to measure the adaptability between two domains in order to determine examples for annotation. Van Asch and Daelemans (2010) find that Rényi divergence outperforms other metrics in predicting POS tagging accuracy, while Plank and van Noord (2011) observe that topic distribution-based representations with Jensen-Shannon divergence perform best for data selection for parsing. Remus (2012) apply Jensen-Shannon divergence to select training examples for sentiment analysis. Finally, Wu and Huang (2016) propose a similarity metric based on a sentiment graph. We test previously explored similarity metrics and complement them with diversity.

Very recently interest emerged in *curriculum learning* (Bengio et al., 2009). It is inspired by human active learning by providing easier examples at initial learning stages (e.g., by curriculum strategies such as growing vocabulary size). Curriculum learning employs a range of data metrics, but aims at altering the order in which the entire training data is selected, rather than *selecting* data. In contrast to us, curriculum learning is mostly aimed at speeding up the learning, while we focus on learning metrics for transfer learning. Other related work in this direction include using Reinforcement Learning to learn what data to select during neural network training (Fan et al., 2017).

There is a long history of research in adaptive data selection, with early approaches grounded in information theory using a Bayesian learning framework (MacKay, 1992). It has also been studied extensively as active learning (El-Gamal, 1991). Curriculum learning is related to active learning (Settles, 2012), whose view is different: active learning aims at finding the most difficult instances to label, examples typically close to the decision boundary. Confidence-based measures are prominent, but as such are less widely applicable than our model-agnostic approach.

The approach most similar to ours is by Tsvetkov et al. (2016) who use Bayesian Optimization to learn a curriculum for training word embeddings. Rather than ordering data (in their case, paragraphs), we use Bayesian Optimization for learning to *select* relevant training instances that are useful for transfer learning in order to prevent negative transfer (Rosenstein et al., 2005). To the best of our knowledge there is no prior work that uses this strategy for transfer learning.

7 Conclusion

We propose to use Bayesian Optimization to learn data selection measures for transfer learning. Our results outperform existing domain similarity metrics on three tasks (sentiment analysis, POS tagging and parsing), and are competitive with a state-of-the-art domain adaptation approach. More importantly, we present the first study on the transferability of such measures, showing promising results to port them across models, domains and related tasks.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. Sebastian is supported by Irish Research Council Grant Number EBPPG/2014/30 and Science Foundation Ireland Grant Number SFI/12/RC/2289. Barbara is supported by NVIDIA corporation and the Computing Center of the University of Groningen.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A Simple But Tough-to-Beat Baseline for Sentence Embeddings. In *ICLR 2017*.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. ACM.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. *Advances in Neural Information Processing Systems*, 19:137–144.

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. 2009. Curriculum learning. In *International Conference on Machine Learning, ICML*.

Anil Bhattacharya. 1943. On a measure of divergence between two statistical population defined by their population distributions. *Bulletin Calcutta Mathematical Society 35.99-109 (1943): 28.*, 35(99-109):28.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *Annual Meeting-Association for Computational Linguistics*, 45(1):440.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain Adaptation with Structural Correspondence Learning. *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP '06)*, pages 120–128.

Danushka Bollegala, David Weir, and John Carroll. 2011. Using Multiple Sources to Construct a Sentiment Sensitive Thesaurus for Cross-Domain Sentiment Classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 132–141.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain Separation Networks. *NIPS*.

E Brochu, V M Cora, and N De Freitas. 2010. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. In *CoRR*.

Hal Daumé III. 2007. Frustratingly Easy Domain Adaptation. *Association for Computational Linguistic (ACL)*, (June):256–263.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation Data Selection using Neural Language Models: Experiments in Machine Translation. *ACL-2013*, (1):678–683.

M. A El-Gamal. 1991. The role of priors in active Bayesian learning in the sequential statistical decision framework. In *Maximum Entropy and Bayesian Methods*, pages 33–38. Springer Netherlands.

Yang Fan, Fei Tian, Tao Qin, Jiang Bian, and Tie-Yan Liu. 2017. Learning What Data to Learn. In *Workshop track - ICLR 2017*.

Jing Jiang and ChengXiang Zhai. 2007. Instance Weighting for Domain Adaptation in NLP. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, (October):264–271.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Aaron Klein, Stefan Falkner, and Frank Hutter. 2017. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017)*.

Lillian Lee. 2001. On the Effectiveness of the Skew Divergence for Statistical Language Analysis. *AISTATS (Artificial Intelligence and Statistics)*, pages 65–72.

J Lin. 1991. Divergence Measures Based on the Shannon Entropy. *IEEE Transactions on Information theory*, 37(1):145–151.

Ji Ma, Yue Zhang, and Jingbo Zhu. 2014. Tagging The Web: Building A Robust Web Tagger with Neural Network. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 144–154.

David J. C. MacKay. 1992. Information-Based Objective Functions for Active Data Selection. *Neural Computation*, 4(4):590–604.

Yishay Mansour. 2009. Domain Adaptation with Multiple Sources. *Neural Information Processing Systems Conference (NIPS 2009)*.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 28–36. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *NIPS*.

Shachar Mirkin and Laurent Besacier. 2014. Data selection for compact adapted smt models. In *Eleventh Conference of the Association for Machine Translation in the Americas (AMTA)*.

Jonas Močkus. 1974. On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference Novosibirsk*.

Robert C Moore and William D Lewis. 2010. Intelligent Selection of Language Model Training Data. *ACL-2010: 48th Annual Meeting of the Association for Computational Linguistics*, (July):220–224.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

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

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 705–713. Association for Computational Linguistics.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, 59.

Barbara Plank and Gertjan van Noord. 2011. Effective Measures of Domain Similarity for Parsing. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 1:1566–1576.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

C. Radhakrishna Rao. 1982. Diversity and dissimilarity coefficients: a unified approach. *Theoretical population biology*, 21(1):24–43.

Carl Edward Rasmussen. 2006. *Gaussian processes for machine learning*. MIT Press.

Sujith Ravi, Kevin Knight, and Radu Soricut. 2008. Automatic prediction of parser accuracy. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 887–896. Association for Computational Linguistics.

Robert Remus. 2012. Domain adaptation using Domain Similarity- and Domain Complexity-based Instance Selection for Cross-Domain Sentiment Analysis. In *IEEE ICDM SENTIRE-2012*.

Alfréd Rényi. 1961. On measures of entropy and information. In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, volume 1.

Michael T. Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G. Dietterich. 2005. To Transfer or Not To Transfer. *NIPS Workshop on Inductive Transfer*.

Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. 2017. Data Selection Strategies for Multi-Domain Sentiment Analysis. In *arXiv preprint arXiv:1702.02426*.

Tobias Schnabel and Hinrich Schütze. 2014. FLORS: Fast and Simple Domain Adaptation for Part-of-Speech Tagging. *TACL*, 2:15–26.

Burr Settles. 2012. *Active learning literature survey*. Morgan and Claypool.

Claude E Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(July 1928):379–423.

Edward H. Simpson. 1949. Measurement of diversity. *Nature*.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. *Neural Information Processing Systems Conference (NIPS 2012)*.

Andres Søgaard. 2011. Data Point Selection for Cross-Language Adaptation of Dependency Parsers. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT '11): Short Papers*, pages 682–686.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, and Chris Dyer. 2016. Learning the Curriculum with Bayesian Optimization for Task-Specific Word Representation Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Vincent Van Asch and Walter Daelemans. 2010. Using Domain Similarity for Performance Estimation. *Computational Linguistics*, (July):31–36.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017. Dynamic data selection for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Fangzhao Wu and Yongfeng Huang. 2016. Sentiment Domain Adaptation with Multiple Sources. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 301–310.

Unsupervised Pretraining for Sequence to Sequence Learning

Prajit Ramachandran and **Peter J. Liu** and **Quoc V. Le**
Google Brain
{prajit, peterjliu, qvl}@google.com

Abstract

This work presents a general unsupervised learning method to improve the accuracy of sequence to sequence (seq2seq) models. In our method, the weights of the encoder and decoder of a seq2seq model are initialized with the pretrained weights of two language models and then fine-tuned with labeled data. We apply this method to challenging benchmarks in machine translation and abstractive summarization and find that it significantly improves the subsequent supervised models. Our main result is that pretraining improves the generalization of seq2seq models. We achieve state-of-the-art results on the WMT English→German task, surpassing a range of methods using both phrase-based machine translation and neural machine translation. Our method achieves a significant improvement of 1.3 BLEU from the previous best models on both WMT'14 and WMT'15 English→German. We also conduct human evaluations on abstractive summarization and find that our method outperforms a purely supervised learning baseline in a statistically significant manner.

1 Introduction

Sequence to sequence (*seq2seq*) models (Sutskever et al., 2014; Cho et al., 2014; Kalchbrenner and Blunsom, 2013; Allen, 1987; Ñeco and Forcada, 1997) are extremely effective on a variety of tasks that require a mapping between a variable-length input sequence to a variable-length output sequence. The main weakness of sequence to sequence models, and deep networks in general, lies in the fact that they can easily overfit when the amount of supervised training data is small.

In this work, we propose a simple and effective technique for using unsupervised pretraining to improve seq2seq models. Our proposal is to initialize both encoder and decoder networks with pretrained weights of two language models. These pretrained weights are then fine-tuned with the labeled corpus. During the fine-tuning phase, we jointly train the seq2seq objective with the language modeling objectives to prevent overfitting.

We benchmark this method on machine translation for English→German and abstractive summarization on CNN and Daily Mail articles. Our main result is that a seq2seq model, with pretraining, exceeds the strongest possible baseline in both neural machine translation and phrase-based machine translation. Our model obtains an improvement of 1.3 BLEU from the previous best models on both WMT'14 and WMT'15 English→German. On human evaluations for abstractive summarization, we find that our model outperforms a purely supervised baseline, both in terms of correctness and in avoiding unwanted repetition.

We also perform ablation studies to understand the behaviors of the pretraining method. Our study confirms that among many other possible choices of using a language model in seq2seq with attention, the above proposal works best. Our study also shows that, for translation, the main gains come from the improved generalization due to the pretrained features. For summarization, pretraining the encoder gives large improvements, suggesting that the gains come from the improved optimization of the encoder that has been unrolled for hundreds of timesteps. On both tasks, our proposed method always improves generalization on the test sets.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 383–391
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

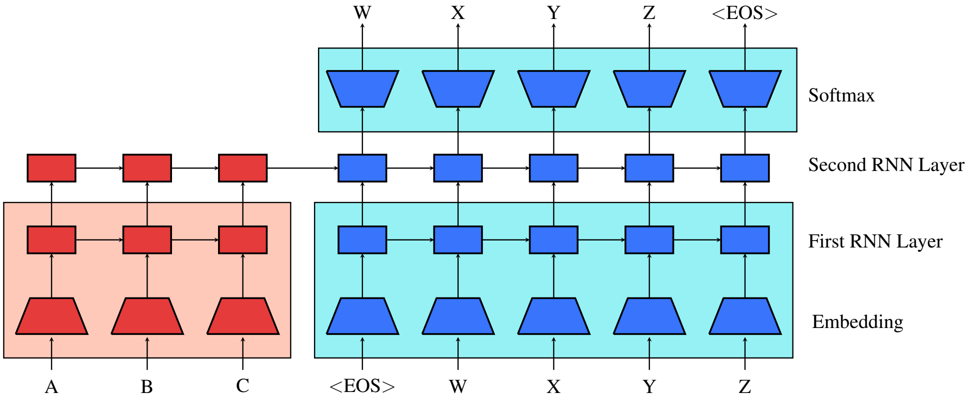


Figure 1: Pretrained sequence to sequence model. The red parameters are the encoder and the blue parameters are the decoder. All parameters in a shaded box are pretrained, either from the source side (light red) or target side (light blue) language model. Otherwise, they are randomly initialized.

2 Methods

In the following section, we will describe our basic unsupervised pretraining procedure for sequence to sequence learning and how to modify sequence to sequence learning to effectively make use of the pretrained weights. We then show several extensions to improve the basic model.

2.1 Basic Procedure

Given an input sequence $x_1, x_2, ..., x_m$ and an output sequence $y_n, y_{n-1}, ..., y_1$, the objective of sequence to sequence learning is to maximize the likelihood $p(y_n, y_{n-1}, ..., y_1 | x_1, x_2, ..., x_m)$. Common sequence to sequence learning methods decompose this objective as $p(y_n, y_{n-1}, ..., y_1 | x_1, x_2, ..., x_m) = \prod_{t=1}^{n} p(y_t | y_{t-1}, ..., y_1; x_1, x_2, ..., x_m)$.

In sequence to sequence learning, an RNN encoder is used to represent $x_1, ..., x_m$ as a hidden vector, which is given to an RNN decoder to produce the output sequence. Our method is based on the observation that without the encoder, the decoder essentially acts like a language model on y's. Similarly, the encoder with an additional output layer also acts like a language model. Thus it is natural to use trained languages models to initialize the encoder and decoder.

Therefore, the basic procedure of our approach is to pretrain both the seq2seq encoder and decoder networks with language models, which can be trained on large amounts of unlabeled text data. This can be seen in Figure 1, where the parameters in the shaded boxes are pretrained. In the following we will describe the method in detail using machine translation as an example application.

First, two monolingual datasets are collected, one for the source side language, and one for the target side language. A language model (*LM*) is trained on each dataset independently, giving an LM trained on the source side corpus and an LM trained on the target side corpus.

After two language models are trained, a multi-layer seq2seq model M is constructed. The embedding and first LSTM layers of the encoder and decoder are initialized with the pretrained weights. To be even more efficient, the softmax of the decoder is initialized with the softmax of the pretrained target side LM.

2.2 Monolingual language modeling losses

After the seq2seq model M is initialized with the two LMs, it is fine-tuned with a labeled dataset. However, this procedure may lead to *catastrophic forgetting*, where the model's performance on the language modeling tasks falls dramatically after fine-tuning (Goodfellow et al., 2013). This may hamper the model's ability to generalize, especially when trained on small labeled datasets.

To ensure that the model does not overfit the labeled data, we regularize the parameters that were pretrained by continuing to train with the monolingual language modeling losses. The seq2seq and language modeling losses are weighted equally.

In our ablation study, we find that this technique is complementary to pretraining and is important in achieving high performance.

2.3 Other improvements to the model

Pretraining and the monolingual language modeling losses provide the vast majority of improvements to the model. However in early experimentation, we found minor but consistent improvements with two additional techniques: a) residual connections and b) multi-layer attention (see Figure 2).

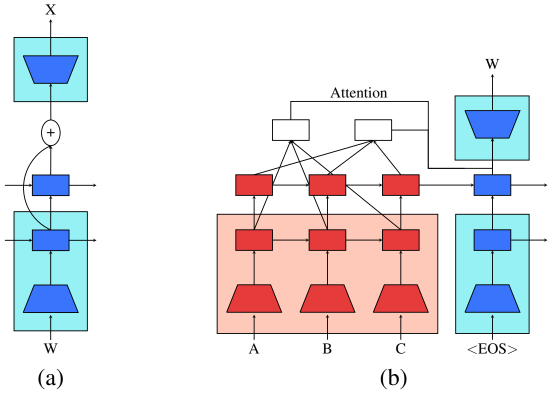


Figure 2: Two small improvements to the baseline model: (a) residual connection, and (b) multi-layer attention.

Residual connections: As described, the input vector to the decoder softmax layer is a random vector because the high level (non-first) layers of the LSTM are randomly initialized. This introduces random gradients to the pretrained parameters. To avoid this, we use a residual connection from the output of the first LSTM layer directly to the input of the softmax (see Figure 2-a).

Multi-layer attention: In all our models, we use an attention mechanism (Bahdanau et al., 2015), where the model attends over both top and first layer (see Figure 2-b). More concretely, given a query vector q_t from the decoder, encoder states from the first layer $h_1^1, \ldots, h_T^1$, and encoder states from the last layer $h_1^L, \ldots, h_T^L$, we compute the attention context vector c_t as follows:

$$\alpha_i = \frac{\exp(q_t \cdot h_i^N)}{\sum_{j=1}^T \exp(q_t \cdot h_j^N)} \qquad c_t^1 = \sum_{i=1}^T \alpha_i h_i^1$$

$$c_t^N = \sum_{i=1}^T \alpha_i h_i^N \qquad c_t = [c_t^1; c_t^N]$$

3 Experiments

In the following section, we apply our approach to two important tasks in seq2seq learning: machine translation and abstractive summarization. On each task, we compare against the previous best systems. We also perform ablation experiments to understand the behavior of each component of our method.

3.1 Machine Translation

Dataset and Evaluation: For machine translation, we evaluate our method on the WMT English→German task (Bojar et al., 2015). We used the WMT 14 training dataset, which is slightly smaller than the WMT 15 dataset. Because the dataset has some noisy examples, we used a language detection system to filter the training examples. Sentences pairs where either the source was not English or the target was not German were thrown away. This resulted in around 4 million training examples. Following Sennrich et al. (2015a), we use subword units (Sennrich et al., 2015b) with 89500 merge operations, giving a vocabulary size around 90000. The validation set is the concatenated newstest2012 and newstest2013, and our test sets are newstest2014 and newstest2015. Evaluation on the validation set was with case-sensitive BLEU (Papineni et al., 2002) on tokenized text using `multi-bleu.perl`. Evaluation on the test sets was with case-sensitive BLEU on detokenized text using `mteval-v13a.pl`. The monolingual training datasets are the News Crawl English and German corpora, each of which has more than a billion tokens.

Experimental settings: The language models were trained in the same fashion as (Jozefowicz et al., 2016) We used a 1 layer 4096 dimensional LSTM with the hidden state projected down to 1024 units (Sak et al., 2014) and trained for one week on 32 Tesla K40 GPUs. Our seq2seq model was a 3 layer model, where the second and third layers each have 1000 hidden units. The monolingual objectives, residual connection, and the modified attention were all used. We used the Adam optimizer (Kingma and Ba, 2015) and train with asynchronous SGD on 16 GPUs for speed. We used a learning rate of 5e-5 which is multiplied by 0.8 every 50K steps after an initial 400K steps, gradient clipping with norm 5.0 (Pascanu et al., 2013), and dropout of 0.2 on non-recurrent connections (Zaremba et al., 2014). We used early stopping on validation set perplexity. A beam size of 10 was used for decoding. Our ensemble is con-

System	*ensemble?*	*BLEU* *newstest2014*	 *newstest2015*
Phrase Based MT (Williams et al., 2016)	-	21.9	23.7
Supervised NMT (Jean et al., 2015)	single	-	22.4
Edit Distance Transducer NMT (Stahlberg et al., 2016)	single	21.7	24.1
Edit Distance Transducer NMT (Stahlberg et al., 2016)	ensemble 8	22.9	25.7
Backtranslation (Sennrich et al., 2015a)	single	22.7	25.7
Backtranslation (Sennrich et al., 2015a)	ensemble 4	23.8	26.5
Backtranslation (Sennrich et al., 2015a)	ensemble 12	**24.7**	27.6
No pretraining	single	21.3	24.3
Pretrained seq2seq	single	**24.0**	**27.0**
Pretrained seq2seq	ensemble 5	**24.7**	**28.1**

Table 1: English→German performance on WMT test sets. Our pretrained model outperforms all other models. Note that the model without pretraining uses the LM objective.

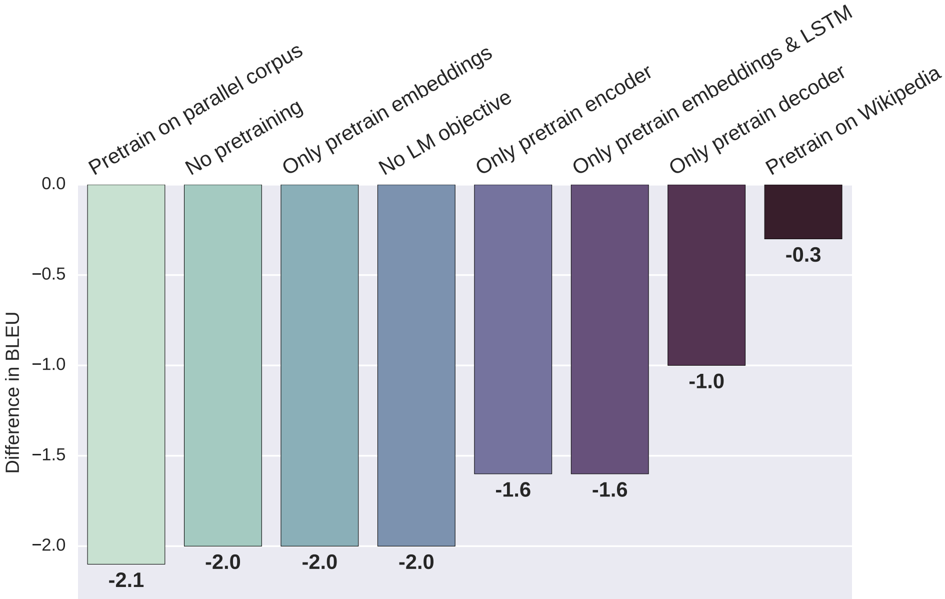


Figure 3: English→German ablation study measuring the difference in validation BLEU between various ablations and the full model. More negative is worse. The full model uses LMs trained with monolingual data to initialize the encoder and decoder, plus the language modeling objective.

structed with the 5 best performing models on the validation set, which are trained with different hyperparameters.

Results: Table 1 shows the results of our method in comparison with other baselines. Our method achieves a new state-of-the-art for single model performance on both newstest2014 and newstest2015, significantly outperforming the competitive semi-supervised *backtranslation* technique (Sennrich et al., 2015a). Equally impressive is the fact that our best single model outperforms the previous state of the art ensemble of 4 models. Our ensemble of 5 models matches or exceeds the previous best ensemble of 12 models.

Ablation study: In order to better understand the effects of pretraining, we conducted an ablation study by modifying the pretraining scheme. We were primarily interested in varying the pretraining scheme and the monolingual language modeling objectives because these two techniques produce the largest gains in the model. Figure 3 shows the drop in validation BLEU of various ablations compared with the full model. The *full model* uses LMs trained with monolingual data to initialize the encoder and decoder, in addition to the language modeling objective. In the following, we interpret the findings of the study. Note that some findings are specific to the translation task.

Given the results from the ablation study, we can make the following observations:

- Only pretraining the decoder is better than only pretraining the encoder: Only pretraining the encoder leads to a 1.6 BLEU point drop while only pretraining the decoder leads to a 1.0 BLEU point drop.

- Pretrain as much as possible because the benefits compound: given the drops of no pretraining at all (-2.0) and only pretraining the encoder (-1.6), the additive estimate of the drop of only pretraining the decoder side is $-2.0 - (-1.6) = -0.4$; however the actual drop is -1.0 which is a much larger drop than the additive estimate.

- Pretraining the softmax is important: Pretraining only the embeddings and first LSTM layer gives a large drop of 1.6 BLEU points.

- The language modeling objective is a strong regularizer: The drop in BLEU points of pretraining the entire model and not using the LM objective is as bad as using the LM objective without pretraining.

- Pretraining on a lot of unlabeled data is essential for learning to extract powerful features: If the model is initialized with LMs that are pretrained on the source part and target part of the *parallel* corpus, the drop in performance is as large as not pretraining at all. However, performance remains strong when pretrained on the large, non-news Wikipedia corpus.

To understand the contributions of unsupervised pretraining vs. supervised training, we track the performance of pretraining as a function of dataset size. For this, we trained a a model with and without pretraining on random subsets of the English→German corpus. Both models use the additional LM objective. The results are summarized in Figure 4. When a 100% of the labeled data is used, the gap between the pretrained and no pretrain model is 2.0 BLEU points. However, that gap grows when less data is available. When trained on 20% of the labeled data, the gap becomes 3.8 BLEU points. This demonstrates that the pretrained models degrade less as the labeled dataset becomes smaller.

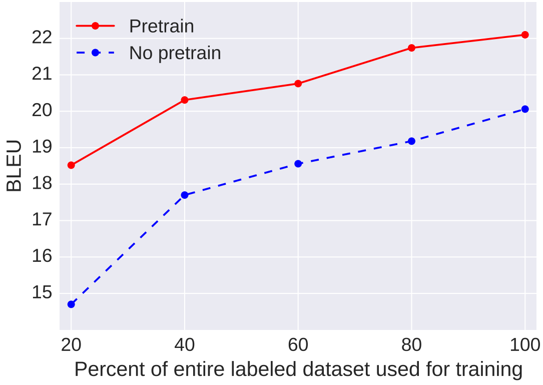


Figure 4: Validation performance of pretraining vs. no pretraining when trained on a subset of the entire labeled dataset for English→German translation.

3.2 Abstractive Summarization

Dataset and Evaluation: For a low-resource abstractive summarization task, we use the CNN/Daily Mail corpus from (Hermann et al., 2015). Following Nallapati et al. (2016), we modify the data collection scripts to restore the bullet point summaries. The task is to predict the bullet point summaries from a news article. The dataset has fewer than 300K document-summary pairs. To compare against Nallapati et al. (2016), we used the anonymized corpus. However, for our ablation study, we used the non-anonymized corpus.[1] We evaluate our system using full length ROUGE (Lin, 2004). For the anonymized corpus in particular, we considered each highlight as a separate sentence following Nallapati et al. (2016). In this setting, we used the English Gigaword corpus (Napoles et al., 2012) as our larger, unlabeled "monolingual" corpus, although all data used in this task is in English.

Experimental settings: We use subword units (Sennrich et al., 2015b) with 31500 merges, resulting in a vocabulary size of about 32000. We use up to the first 600 tokens of the document and

[1] We encourage future researchers to use the non-anonymized version because it is a more realistic summarization setting with a larger vocabulary. Our numbers on the non-anonymized test set are 35.56 ROUGE-1, 14.60 ROUGE-2, and 25.08 ROUGE-L. We did not consider highlights as separate sentences.

System	*ROUGE-1*	*ROUGE-2*	*ROUGE-L*
Seq2seq + pretrained embeddings (Nallapati et al., 2016)	32.49	11.84	29.47
+ temporal attention (Nallapati et al., 2016)	**35.46**	**13.30**	**32.65**
Pretrained seq2seq	32.56	11.89	29.44

Table 2: Results on the anonymized CNN/Daily Mail dataset.

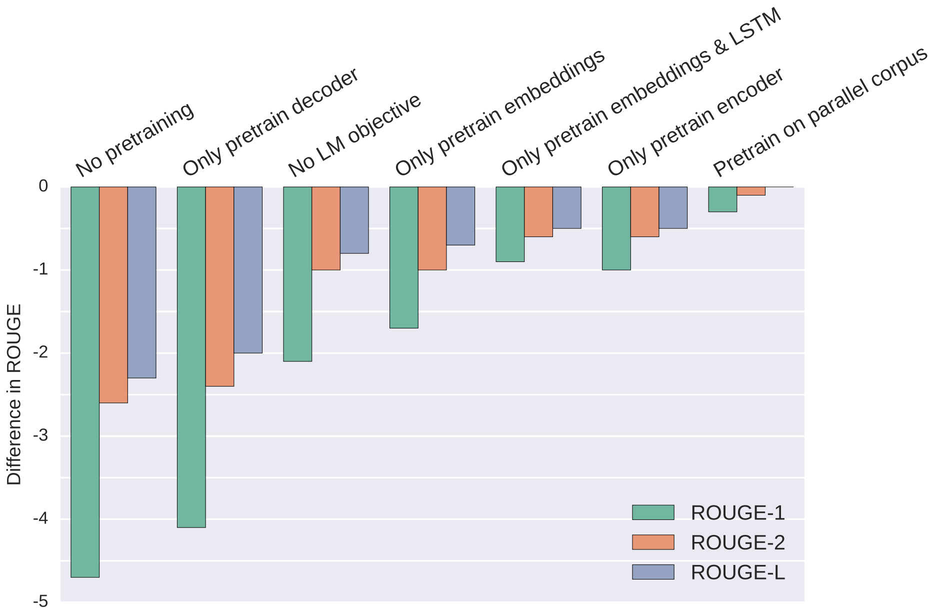


Figure 5: Summarization ablation study measuring the difference in validation ROUGE between various ablations and the full model. More negative is worse. The full model uses LMs trained with unlabeled data to initialize the encoder and decoder, plus the language modeling objective.

predict the entire summary. Only one language model is trained and it is used to initialize both the encoder and decoder, since the source and target languages are the same. However, the encoder and decoder are not tied. The LM is a one-layer LSTM of size 1024 trained in a similar fashion to Jozefowicz et al. (2016). For the seq2seq model, we use the same settings as the machine translation experiments. The only differences are that we use a 2 layer model with the second layer having 1024 hidden units, and that the learning rate is multiplied by 0.8 every 30K steps after an initial 100K steps.

Results: Table 2 summarizes our results on the anonymized version of the corpus. Our pretrained model is only able to match the previous baseline seq2seq of Nallapati et al. (2016). Interestingly, they use pretrained word2vec (Mikolov et al., 2013) vectors to initialize their word embeddings. As we show in our ablation study, just pretraining the embeddings itself gives a large improvement. Furthermore, our model is a unidirectional LSTM while they use a bidirectional LSTM. They also use a longer context of 800 tokens, whereas we used a context of 600 tokens due to GPU memory issues.

Ablation study: We performed an ablation study similar to the one performed on the machine translation model. The results are reported in Figure 5. Here we report the drops on ROUGE-1, ROUGE-2, and ROUGE-L on the non-anonymized validation set.

Given the results from our ablation study, we can make the following observations:

- Pretraining appears to improve optimization: in contrast with the machine translation model, it is more beneficial to only pretrain the encoder than only the decoder of the summarization model. One interpretation is that pretraining enables the gradient to flow much further back in time than randomly initialized weights. This may also explain why pretraining on the parallel corpus is no worse than pretraining on a larger monolingual corpus.

- The language modeling objective is a strong regularizer: A model without the LM objective has a significant drop in ROUGE scores.

Human evaluation: As ROUGE may not be able to capture the quality of summarization, we also performed a small qualitative study to understand the human impression of the summaries produced by different models. We took 200 random documents and compared the performance of a pretrained and non-pretrained system. The document, gold summary, and the two system outputs were presented to a human evaluator who was asked to rate each system output on a scale of 1-5 with 5 being the best score. The system outputs were presented in random order and the evaluator did not know the identity of either output. The evaluator noted if there were repetitive phrases or sentences in either system outputs. Unwanted repetition was also noticed by Nallapati et al. (2016).

Table 3 and 4 show the results of the study. In both cases, the pretrained system outperforms the system without pretraining in a statistically significant manner. The better optimization enabled by pretraining improves the generated summaries and decreases unwanted repetition in the output.

NP > *P*	*NP* = *P*	*NP* < *P*
29	88	83

Table 3: The count of how often the no pretrain system (*NP*) achieves a higher, equal, and lower score than the pretrained system (*P*) in the side-by-side study where the human evaluator gave each system a score from 1-5. The sign statistical test gives a p-value of < 0.0001 for rejecting the null hypothesis that there is no difference in the score obtained by either system.

		No pretrain	
		No repeats	Repeats
Pretrain	No repeats	67	65
	Repeats	24	44

Table 4: The count of how often the pretrain and no pretrain systems contain repeated phrases or sentences in their outputs in the side-by-side study. McNemar's test gives a p-value of < 0.0001 for rejecting the null hypothesis that the two systems repeat the same proportion of times. The pretrained system clearly repeats less than the system without pretraining.

4 Related Work

Unsupervised pretraining has been intensively studied in the past years, most notably is the work by Dahl et al. (2012) who found that pretraining with deep belief networks improved feedforward acoustic models. More recent acoustic models have found pretraining unnecessary (Xiong et al., 2016; Zhang et al., 2016; Chan et al., 2015), probably because the reconstruction objective of deep belief networks is too easy. In contrast, we find that pretraining language models by next step prediction significantly improves seq2seq on challenging real world datasets.

Despite its appeal, unsupervised learning has not been widely used to improve supervised training. Dai and Le (2015); Radford et al. (2017) are amongst the rare studies which showed the benefits of pretraining in a semi-supervised learning setting. Their methods are similar to ours except that they did not have a decoder network and thus could not apply to seq2seq learning. Similarly, Zhang and Zong (2016) found it useful to add an additional task of sentence reordering of source-side monolingual data for neural machine translation. Various forms of transfer or multitask learning with seq2seq framework also have the flavors of our algorithm (Zoph et al., 2016; Luong et al., 2015; Firat et al., 2016).

Perhaps most closely related to our method is the work by Gulcehre et al. (2015), who combined a language model with an already trained seq2seq model by fine-tuning additional deep output layers. Empirically, their method produces small improvements over the supervised baseline. We suspect that their method does not produce significant gains because (i) the models are trained independently of each other and are not fine-tuned (ii) the LM is combined with the seq2seq model after the last layer, wasting the benefit of the low level LM features, and (iii) only using the LM on the decoder side. Venugopalan et al. (2016) addressed (i) but still experienced minor improvements. Using pretrained GloVe embedding vectors (Pennington et al., 2014) had more impact.

Related to our approach in principle is the work by Chen et al. (2016) who proposed a two-term, theoretically motivated unsupervised objective for unpaired input-output samples. Though they did not apply their method to seq2seq learning, their framework can be modified to do so. In that case, the first term pushes the output to be highly probable under some scoring model, and the second term ensures that the output depends on the input. In the seq2seq setting, we interpret the first term as a pretrained language model scoring the output sequence. In our work, we fold the pretrained language model into the decoder. We believe that using the pretrained language model only for scoring is less efficient that using all the pretrained weights. Our use of labeled examples satisfies the second term. These connections provide a theoretical grounding for our work.

In our experiments, we benchmark our method on machine translation, where other unsupervised methods are shown to give promising results (Sennrich et al., 2015a; Cheng et al., 2016). In backtranslation (Sennrich et al., 2015a), the trained model is used to decode unlabeled data to yield extra labeled data. One can argue that this method may not have a natural analogue to other tasks such as summarization. We note that their technique is complementary to ours, and may lead to additional gains in machine translation. The method of using autoencoders in Cheng et al. (2016) is promising, though it can be argued that autoencoding is an easy objective and language modeling may force the unsupervised models to learn better features.

5 Conclusion

We presented a novel unsupervised pretraining method to improve sequence to sequence learning. The method can aid in both generalization and optimization. Our scheme involves pretraining two language models in the source and target domain, and initializing the embeddings, first LSTM layers, and softmax of a sequence to sequence model with the weights of the language models. Using our method, we achieved state-of-the-art machine translation results on both WMT'14 and WMT'15 English to German. A key advantage of this technique is that it is flexible and can be applied to a large variety of tasks.

References

Robert B. Allen. 1987. Several studies on natural language and back-propagation. *IEEE First International Conference on Neural Networks*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*.

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2015. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*.

Jianshu Chen, Po-Sen Huang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Unsupervised learning of predictors from unpaired input-output samples. abs/1606.04646.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. *arXiv preprint arXiv:1606.04596*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

G. E. Dahl, D. Yu, L. Deng, and A. Acero. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman-Vural, and Kyunghyun Cho. 2016. Zero-resource translation with multilingual neural machine translation. *arXiv preprint arXiv:1606.04164*.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal neural machine translation systems for WMT'15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. In *ICLR*.

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

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016. Sequence-to-sequence RNNs for text summarization. *arXiv preprint arXiv:1602.06023*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. ACL.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *ACL*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML*.

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

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Hasim Sak, Andrew W. Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015b. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Felix Stahlberg, Eva Hasler, and Bill Byrne. 2016. The edit distance transducer in action: The university of cambridge english-german system at wmt16. In *Proceedings of the First Conference on Machine Translation*, pages 377–384, Berlin, Germany. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Subhashini Venugopalan, Lisa Anne Hendricks, Raymond Mooney, and Kate Saenko. 2016. Improving LSTM-based video description with linguistic knowledge mined from text. *arXiv preprint arXiv:1604.01729*.

Philip Williams, Rico Sennrich, Maria Nadejde, Matthias Huck, Barry Haddow, and Ondřej Bojar. 2016. Edinburgh's statistical machine translation systems for wmt16. In *Proceedings of the First Conference on Machine Translation*, pages 399–410, Berlin, Germany. Association for Computational Linguistics.

Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. 2016. Achieving human parity in conversational speech recognition. abs/1610.05256.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *EMNLP*.

Yu Zhang, William Chan, and Navdeep Jaitly. 2016. Very deep convolutional networks for end-to-end speech recognition. abs/1610.03022.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *EMNLP*.

Ramon P. Ñeco and Mikel L. Forcada. 1997. Asynchronous translations with recurrent neural nets. *Neural Networks*.

Efficient Attention using a Fixed-Size Memory Representation

Denny Britz[*] and **Melody Y. Guan**[*] and **Minh-Thang Luong**
Google Brain
dennybritz,melodyguan,thangluong@google.com

Abstract

The standard content-based attention mechanism typically used in sequence-to-sequence models is computationally expensive as it requires the comparison of large encoder and decoder states at each time step. In this work, we propose an alternative attention mechanism based on a fixed size memory representation that is more efficient. Our technique predicts a compact set of K attention contexts during encoding and lets the decoder compute an efficient lookup that does not need to consult the memory. We show that our approach performs on-par with the standard attention mechanism while yielding inference speedups of 20% for real-world translation tasks and more for tasks with longer sequences. By visualizing attention scores we demonstrate that our models learn distinct, meaningful alignments.

1 Introduction

Sequence-to-sequence models (Sutskever et al., 2014; Cho et al., 2014) have achieved state of the art results across a wide variety of tasks, including Neural Machine Translation (NMT) (Bahdanau et al., 2014; Wu et al., 2016), text summarization (Rush et al., 2015; Nallapati et al., 2016), speech recognition (Chan et al., 2015; Chorowski and Jaitly, 2016), image captioning (Xu et al., 2015), and conversational modeling (Vinyals and Le, 2015; Li et al., 2015).

The most popular approaches are based on an encoder-decoder architecture consisting of two recurrent neural networks (RNNs) and an attention mechanism that aligns target to source tokens (Bahdanau et al., 2014; Luong et al., 2015). The typical attention mechanism used in these architectures computes a new attention context at each decoding step based on the current state of the decoder. Intuitively, this corresponds to looking at the source sequence after the output of every single target token.

Inspired by how humans process sentences, we believe it may be unnecessary to look back at the entire original source sequence at each step.[1] We thus propose an alternative attention mechanism (section 3) that leads to smaller computational time complexity. Our method predicts K attention context vectors while reading the source, and learns to use a weighted average of these vectors at each step of decoding. Thus, we avoid looking back at the source sequence once it has been encoded. We show (section 4) that this speeds up inference while performing on-par with the standard mechanism on both toy and real-world WMT translation datasets. We also show that our mechanism leads to larger speedups as sequences get longer. Finally, by visualizing the attention scores (section 5), we verify that the proposed technique learns meaningful alignments, and that different attention context vectors specialize on different parts of the source.

2 Background

2.1 Sequence-to-Sequence Model with Attention

Our models are based on an encoder-decoder architecture with attention mechanism (Bahdanau et al., 2014; Luong et al., 2015). An encoder function takes as input a sequence of source tokens $\mathbf{x} = (x_1,...,x_m)$ and produces a sequence of states $\mathbf{s} = (s_1,...,s_m)$ .The decoder is an RNN that predicts the probability of a target sequence $\mathbf{y} = (y_1,...,y_T \,|\, \mathbf{s})$. The probability of each target token $y_i \in \{1,...,|V|\}$ is predicted based on the recurrent state in the decoder RNN, h_i, the previous words, $y_{<i}$, and a context vector c_i. The context vector c_i, also referred to as the attention vector, is calculated as a weighted average of the source states.

[*] Equal Contribution. Author order alphabetical.

[1] Eye-tracking and keystroke logging data from human translators show that translators generally do not reread previously translated source text words when producing target text (Carl et al., 2011).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 392–400
Copenhagen, Denmark, September 7–11, 2017.

$$c_i = \sum_j \alpha_{ij} s_j \tag{1}$$

$$\alpha_i = \text{softmax}(f_{att}(h_i, \mathbf{s})) \tag{2}$$

Here, $f_{att}(h_i, \mathbf{s})$ is an attention function that calculates an unnormalized alignment score between the encoder state s_j and the decoder state h_i. Variants of f_{att} used in Bahdanau et al. (2014) and Luong et al. (2015) are:

$$f_{att}(h_i, s_j) = \begin{cases} v_a^T \tanh(W_a[h_i, s_j]), & \textit{Bahdanau} \\ h_i^T W_a s_j & \textit{Luong} \end{cases}$$

where W_a and v_a are model parameters learned to predict alignment. Let $|S|$ and $|T|$ denote the lengths of the source and target sequences respectively and D denoate the state size of the encoder and decoder RNN. Such content-based attention mechanisms result in inference times of $O(D^2|S||T|)$[2], as each context vector depends on the current decoder state h_i and all encoder states, and requires an $O(D^2)$ matrix multiplication.

The decoder outputs a distribution over a vocabulary of fixed-size $|V|$:

$$P(y_i | y_{<i}, \mathbf{x}) = \text{softmax}(W[s_i; c_i] + b) \tag{3}$$

The model is trained end-to-end by minimizing the negative log likelihood of the target words using stochastic gradient descent.

3 Memory-Based Attention Model

Our proposed model is shown in Figure 1. During encoding, we compute an attention matrix $C \in \mathbb{R}^{K \times D}$, where K is the number of attention vectors and a hyperparameter of our method, and D is the dimensionality of the top-most encoder state. This matrix is computed by predicting a score vector $\alpha_t \in \mathbb{R}^K$ at each encoding time step t. C is then a linear combination of the encoder states, weighted by α_t:

$$C_k = \sum_{t=0}^{|S|} \alpha_{tk} s_t \tag{4}$$

$$\alpha_t = \text{softmax}(W_\alpha s_t), \tag{5}$$

where W_α is a parameter matrix in $\mathbb{R}^{K \times D}$.

The computational time complexity for this operation is $O(KD|S|)$. One can think of C as compact fixed-length memory that the decoder will perform attention over. In contrast, standard approaches use a variable-length set of encoder states for attention. At each decoding step, we similarly predict K scores $\beta \in \mathbb{R}^K$. The final attention context c is a linear combination of the rows in C weighted by the scores. Intuitively, each decoder step predicts how important each of the K attention vectors is.

$$c = \sum_{i=0}^{K} \beta_i C_i \tag{6}$$

$$\beta = \text{softmax}(W_\beta h) \tag{7}$$

Here, h is the current state of the decoder, and W_β is a learned parameter matrix. Note that we do not access the encoder states at each decoder step. We simply take a linear combination of the attention matrix C pre-computed during encoding - a much cheaper operation that is independent of the length of the source sequence. The time complexity of this computation is $O(KD|T|)$ as multiplication with the K attention matrices needs to happen at each decoding step.

Summing $O(KD|S|)$ from encoding and $O(KD|T|)$ from decoding, we have a total linear computational complexity of $O(KD(|S| + |T|)$. As D is typically very large, 512 or 1024 units in most applications, we expect our model to be faster than the standard attention mechanism running in $O(D^2|S||T|)$. For long sequences (as in summarization, where —S— is large), we also expect our model to be faster than the cheaper dot-based attention mechanism, which needs $O(D|S||T|)$ computation time and requires encoder and decoder states sizes to match.

We also experimented with using a sigmoid function instead of the softmax to score the encoder and decoder attention scores, resulting in 4 possible combinations. We call this choice the *scoring function*. A softmax scoring function calculates normalized scores, while the sigmoid scoring function results in unnormalized scores that can be understood as gates.

3.1 Model Interpretations

Our memory-based attention model can be understood intuitively in two ways. We can interpret it as ”predicting” the set of attention contexts produced by a standard attention mechanism during encoding. To see this, assume we set $K \approx |T|$. In this case, we predict all $|T|$ attention contexts during the encoding stage and learn to choose the right one during decoding. This is cheaper than computing contexts one-by-one based on the decoder and encoder content. In fact, we could enforce this objective by first training

[2] An exception is the dot-attention from Luong et al. (2015), which is $O(D|S||T|)$, which we discuss further in Section 3.

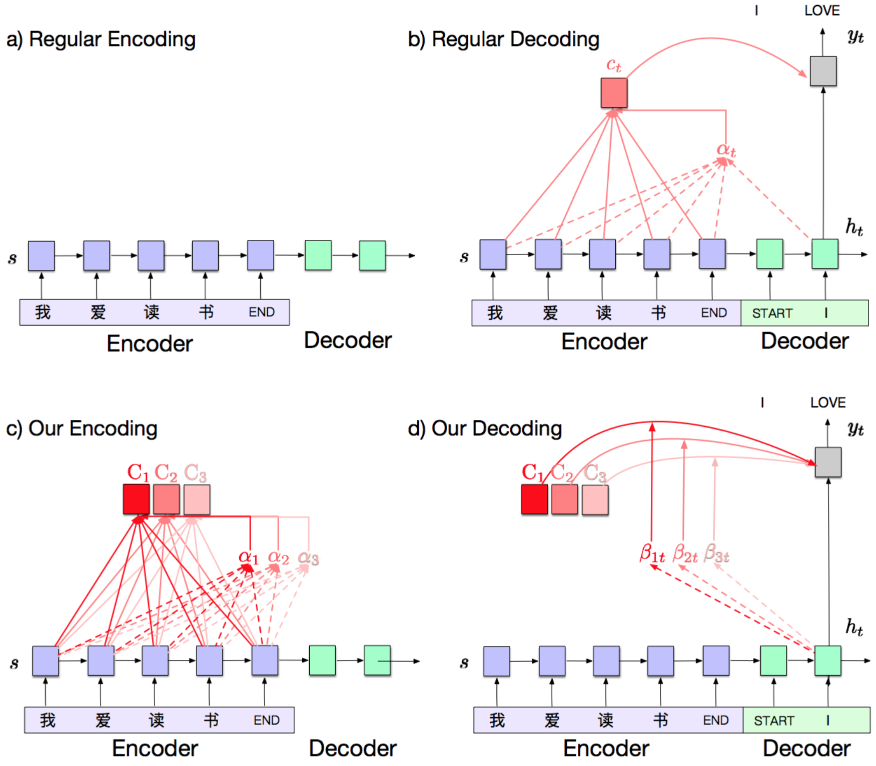


Figure 1: Memory Attention model architecture. K attention vectors are predicted during encoding, and a linear combination is chosen during decoding. In our example, $K = 3$.

a regular attention model and adding a regularization term to force the memory matrix C to be close to the $T \times D$ vectors computed by the standard attention. We leave it to future work to explore such an objective.

Alternatively, we can interpret our mechanism as first predicting a compact $K \times D$ memory matrix, a representation of the source sequence, and then performing *location-based* attention on the memory by picking which row of the matrix to attend to. Standard location-based attention mechanism, by contrast, predicts a location in the source sequence to focus on (Luong et al., 2015; Xu et al., 2015).

3.2 Position Encodings (PE)

In the above formulation, the predictions of attention contexts are symmetric. That is, C_i is not forced to be different from $C_{j \neq i}$. While we would hope for the model to learn to generate distinct attention contexts, we now present an extension that pushes the model into this direction. We add *position encodings* to the score matrix that forces the first few context vector $C_1, C_2, ...$ to focus on the beginning of the sequence and the last few vectors $..., C_{K-1}, C_K$ to focus on the end (thereby encouraging in-between vectors to focus on the middle).

Explicitly, we multiply the score vector α with position encodings $l_s \in \mathbb{R}^K$:

$$C^{PE} = \sum_{s=0}^{|S|} \alpha^{PE} h_s \tag{8}$$

$$\alpha_s^{PE} = \text{softmax}(W_\alpha h_s \circ l_s) \tag{9}$$

To obtain l_s we first calculate a constant matrix L where we define each element as

$$L_{ks} = (1 - k/K)(1 - s/\mathcal{S}) + \frac{k}{K}\frac{s}{\mathcal{S}}, \tag{10}$$

adapting a formula from (Sukhbaatar et al., 2015). Here, $k \in \{1, 2, ..., K\}$ is the context vector index and $\mathcal{S}$ is the maximum sequence length across all source sequences. The manifold is shown graphically in Figure 2. We can see that earlier encoder states are upweighted in the first context vectors, and later states are upweighted in later vectors. The symmetry of the manifold and its stationary point having value 0.5 both follow from Eq. 10. The elements of the matrix that fall beyond the sequence lengths are then masked out and the remaining elements are renormalized across the timestep dimension. This results in the jagged array of position encodings $\{l_{ks}\}$.

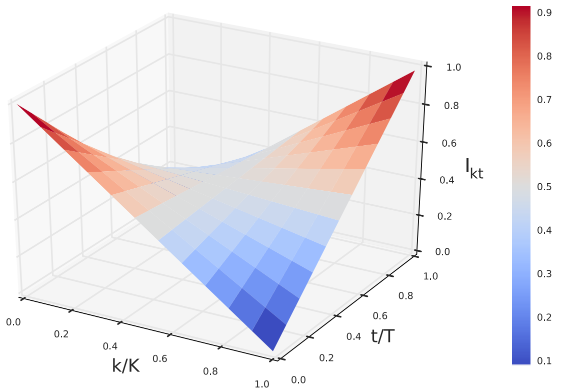


Figure 2: Surface for the position encodings.

4 Experiments

4.1 Toy Copying Experiment

Due to the reduction of computational time complexity we expect our method to yield performance gains especially for longer sequences and tasks where the source can be compactly represented in a fixed-size memory matrix. To investigate the trade-off between speed and performance, we compare our technique to standard models with and without attention on a Sequence Copy Task of varying length like in Graves et al. (2014). We generated 4 training datasets of 100,000 examples and a validation dataset of 1,000 examples. The vocabulary size was 20. For each dataset, the sequences had lengths randomly chosen between 0 to L, for $L \in \{10,50,100,200\}$ unique to each dataset.

4.1.1 Training Setup

All models are implemented using TensorFlow based on the seq2seq implementation of Britz et al. (2017)[3] and trained on a single machine with a Nvidia K40m GPU. We use a 2-layer 256-unit, a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) encoder, a 2-layer 256-unit LSTM decoder, and 256-dimensional embeddings. For the attention baseline, we use the standard parametrized attention (Bahdanau et al., 2014). Dropout of 0.2 (0.8 keep probability) is applied to the input of each cell and we optimize using Adam (Kingma and Ba, 2014) at a learning rate of 0.0001 and batch size 128. We train for at most 200,000 steps (see Figure 3 for sample learning curves). BLEU scores are calculated on tokenized data using the *multi-bleu.perl* script in Moses.[4] We decode using beam search with a beam size of 10 (Wiseman and Rush, 2016).

Length	Model	BLEU	Time (s)
20	No Att	99.93	2.03
	$K=1$	99.52	2.12
	$K=4$	99.56	2.25
	$K=16$	99.56	2.21
	$K=32$	99.57	2.59
	$K=64$	99.75	2.86
	Att	99.98	2.86
50	No Att	97.37	3.90
	$K=1$	98.86	4.33
	$K=4$	99.95	4.48
	$K=16$	99.96	4.58
	$K=32$	99.96	5.35
	$K=64$	99.97	5.84
	Att	99.94	6.46
100	No Att	73.99	6.33
	$K=1$	87.42	7.32
	$K=4$	99.81	7.47
	$K=16$	99.97	7.50
	$K=32$	99.99	7.65
	$K=64$	100.00	7.77
	Att	100.00	11.00
200	No Att	32.64	9.10
	$K=1$	44.22	9.30
	$K=4$	98.54	9.49
	$K=16$	99.98	9.53
	$K=32$	100.00	9.59
	$K=64$	100.00	9.78
	Att	100.00	14.28

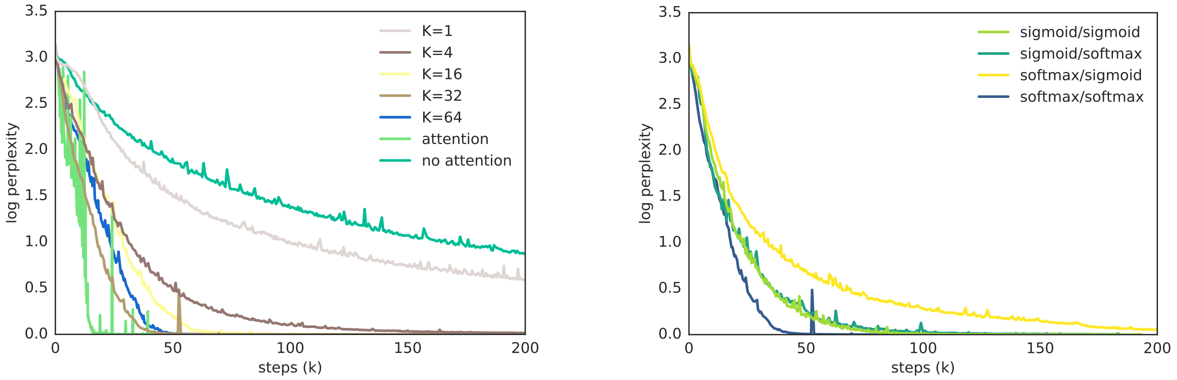
Table 1: BLEU scores and computation times with varying K and sequence length compared to baseline models with and without attention.

4.1.2 Results

Table 1 shows the BLEU scores of our model on different sequence lengths while varying K. This is a study of the trade-off between computational time and representational power. A large K allows us to compute complex source representations, while a K of 1 limits the source representation to a single vector. We can see that performance consistently increases with K up to a point that depends on the data length, with longer sequences requiring more complex representations. The results with and without position encodings are almost identical on the toy data. Our technique learns to fit the data as well as the standard attention mechanism despite having less representational power. Both beat the non-attention baseline by a significant margin.

[3] http://github.com/google/seq2seq

[4] http://github.com/moses-smt/mosesdecoder



(a) Comparison of varying K for copying sequences of length 200 on evaluation data, showing that large K leads to faster convergence and small K performs similarly to the non-attentional baseline.

(b) Comparison of sigmoid and softmax functions for choosing the encoder and decoder attention scores on evaluation data, showing that choice of gating/normalization matters.

Figure 3: Training Curves for the Toy Copy task

That we are able to represent the source sequence with a fixed size matrix with fewer than $|S|$ rows suggests that traditional attention mechanisms may be representing the source with redundancies and wasting computational resources. This makes intuitive sense for the toy task, which should require a relatively simple representation.

The last column shows that our technique significantly speeds up the inference process. The gap in inference speed increases as sequences become longer. We measured inference time on the full validation set of 1,000 examples, not including data loading or model construction times.

Figure 3a shows the learning curves for sequence length 200. We see that $K=1$ is unable to fit the data distribution, while $K \in \{32,64\}$ fits the data almost as quickly as the attention-based model. Figure 3b shows the effect of varying the encoder and decoder scoring functions between softmax and sigmoid. All combinations manage to fit the data, but some converge faster than others. In section 5 we show that distinct alignments are learned by different function combinations.

4.2 Machine Translation

Next, we explore if the memory-based attention mechanism is able to fit complex real-world datasets. For this purpose we use 4 large machine translation datasets of WMT'17[5] on the following language pairs: English-Czech (en-cs, 52M examples), English-German (en-de, 5.9M examples), English-Finish (en-fi, 2.6M examples), and English-Turkish (en-tr, 207,373 examples). We used the newly available preprocessed datasets for the WMT'17 task.[6] Note that our scores may not be directly comparable to other work that performs their own data pre-processing. We learn shared vocabularies of 16,000 subword units using the BPE algorithm (Sennrich et al., 2016). We use newstest2015 as a validation set, and report BLEU on newstest2016.

4.2.1 Training Setup

We use a similar setup to the Toy Copy task, but use 512 RNN and embedding units, train using 8 distributed workers with 1 GPU each, and train for at most 1M steps. We save checkpoints every 30 minutes during training, and choose the best based on the validation BLEU score.

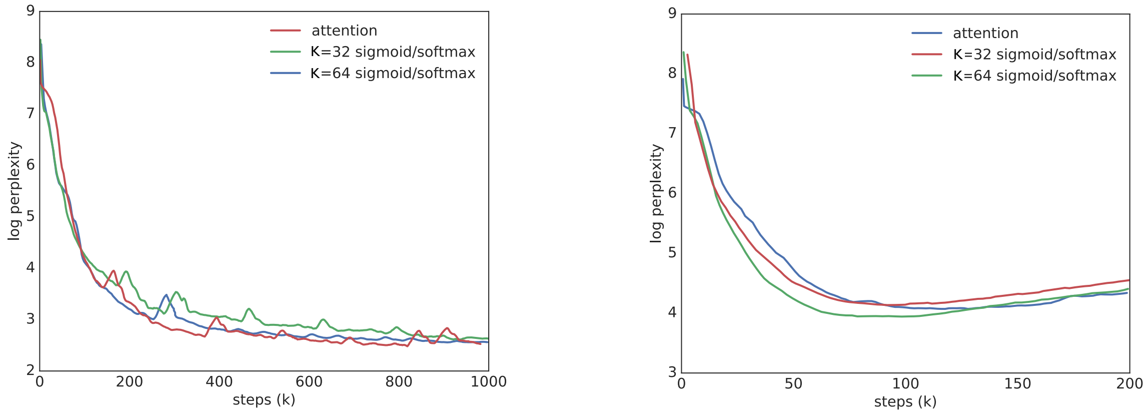
4.2.2 Results

Table 2 compares our approach with and without position encodings, and with varying values for hyperparameter K, to baseline models with regular attention mechanism. Learning curves are shown in Figure 4. We see that our memory attention model with sufficiently high K performs on-par with, or slightly better, than the attention-based baseline model despite its simpler nature. Across the board, models with $K = 64$ performed better than corresponding models with $K=32$, suggesting that using a larger number of attention vectors can capture a richer understanding of source sequences. Position encodings also seem to consistently improve model performance.

Table 3 shows that our model results in faster decoding time even on a complex dataset with a large

[5] statmt.org/wmt17/translation-task.html

[6] http://data.statmt.org/wmt17/translation-task/preprocessed



(a) Training curves for en-fi

(b) Training curves for en-tr

Figure 4: Comparing training curves for en-fi and en-tr with sigmoid encoder scoring and softmax decoder scoring and position encoding. Note that en-tr curves converged very quickly.

Model	Dataset	K	en-cs	en-de	en-fi	en-tr
Memory Attention	Test	32	19.37	28.82	15.87	-
		64	19.65	29.53	16.49	-
	Valid	32	19.20	26.20	15.90	12.94
		64	19.63	26.39	16.35	13.06
Memory Attention + PE	Test	32	19.45	29.53	15.86	-
		64	**20.36**	30.61	17.03	-
	Valid	32	19.35	26.22	16.31	12.97
		64	19.73	27.31	16.91	13.25
Attention	Test	-	19.19	**30.99**	**17.34**	-
	Valid	-	18.61	28.13	17.16	13.76

Table 2: BLEU scores on WMT'17 translation datasets from the memory attention models and regular attention baselines. We picked the best out of the four scoring function combinations on the validation set. Note that en-tr does not have an official test set. Best test scores on each dataset are highlighted.

Model	**Decoding Time (s)**
$K=32$	26.85
$K=64$	27.13
Attention	33.28

Table 3: Decoding time, averaged across 10 runs, for the en-de validation set (2169 examples) with average sequence length of 35. Results are similar for both PE and non-PE models.

vocabulary of 16k. We measured decoding time over the full validation set, not including time used for model setup and data loading, averaged across 10 runs. The average sequence length for examples in this data was 35, and we expect more significant speedups for tasks with longer sequences, as suggested by our experiments on toy data. Note that in our NMT examples/experiments, $K \approx T$, but we obtain computational savings from the fact that $K \ll D$. We may be able to set $K \ll T$, as in toy copying, and still get very good performance in other tasks. For instance, in summarization the source is complex but the representation of the source required to perform the task is "simple" (i.e. all that is needed to generate the abstract).

Figure 5 shows the effect of using sigmoid and softmax function in the encoders and decoders. We found that softmax/softmax consistently performs badly, while all other combinations perform about equally well. We report results for the best combination only (as chosen on the validation set), but we found this choice to only make a minor difference.

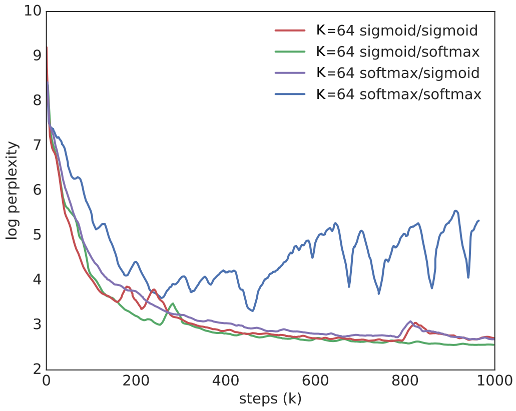


Figure 5: Comparing training curves for en-fi for different encoder/decoder scoring functions for our models at $K = 64$.

5 Visualizing Attention

A useful property of the standard attention mechanism is that it produces meaningful alignment between source and target sequences. Often, the attention mechanism learns to progressively focus on the next source token as it decodes the target. These visualizations can be an important tool in debugging and evaluating seq2seq models and are often used for unknown token replacement.

This raises the question of whether or not our proposed memory attention mechanism also learns to generate meaningful alignments. Due to limiting the number of attention contexts to a number that is generally less than the sequence length, it is not immediately obvious what each context would learn to focus on. Our hope was that the model would learn to focus on multiple alignments at the same time, within the same attention vector. For example, if the source sequence is of length 40 and we have $K = 10$ attention contexts, we would hope that C_1 roughly focuses on tokens 1 to 4, C_2 on tokens 5 to 8, and so on. Figures 6 and 7 show that this is indeed the case. To generate this visualization we multiply the attention scores α and β from the encoder and decoder. Figure 8 shows a sample translation task visualization.

Figure 6 suggests that our model learns distinct ways to use its memory depending on the encoder and decoder functions. Interestingly, using softmax normalization results in attention maps typical of those derived from using standard attention, i.e. a relatively linear mapping between source and target tokens. Meanwhile, using sigmoid gating results in what seems to be a distributed representation of the source sequences across encoder time steps, with multiple contiguous attention contexts being accessed at each decoding step.

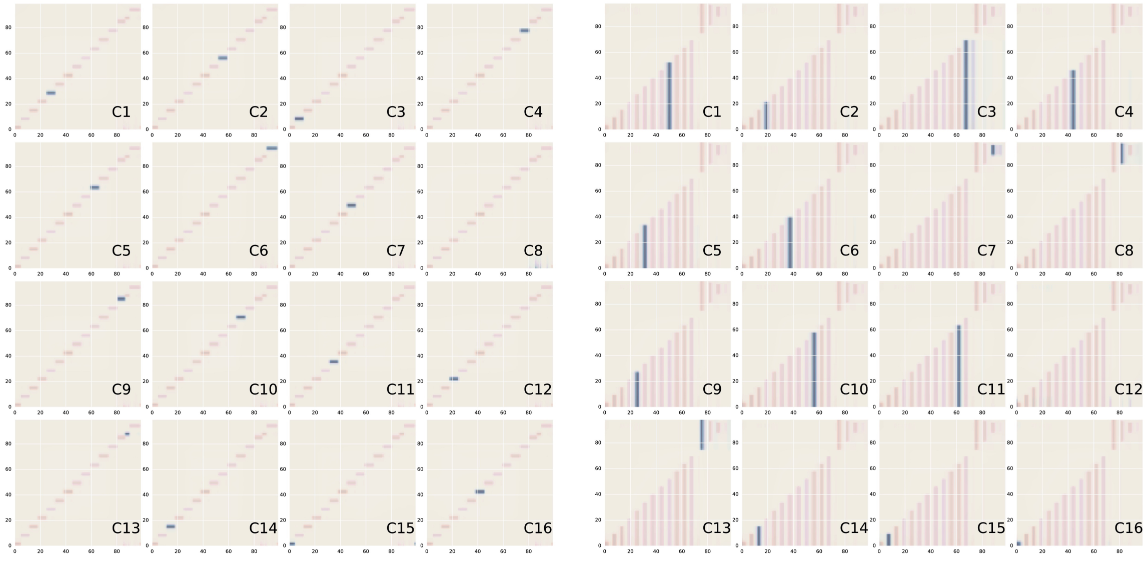
6 Related Work

Our contributions build on previous work in making seq2seq models more computationally efficient. Luong et al. (2015) introduce various attention mechanisms that are computationally simpler and perform as well or better than the original one presented in Bahdanau et al. (2014). However, these typically still require $O(D^2)$ computation complexity, or lack the flexibility to look at the full source sequence. Efficient location-based attention (Xu et al., 2015) has also been explored in the image recognition domain.

Wu et al. (2016) presents several enhancements to the standard seq2seq architecture that allow more efficient computation on GPUs, such as only attending on the bottom layer. Kalchbrenner et al. (2016) propose a linear time architecture based on stacked convolutional neural networks. Gehring et al. (2016) also propose the use of convolutional encoders to speed up NMT. de Brébisson and Vincent (2016) propose a linear attention mechanism based on covariance matrices applied to information retrieval. Raffel et al. (2017) enable online linear time attention calculation by enforcing that the alignment between input and output sequence elements be monotonic. Previously, monotonic attention was proposed for morphological inflection generation by Aharoni and Goldberg (2016).

7 Conclusion

In this work, we propose a novel memory-based attention mechanism that results in a linear computational time of $O(KD(|S| + |T|))$ during decoding in seq2seq models. Through a series of experiments, we demonstrate that our technique leads to consistent inference speedups as sequences get longer, and can fit complex data distributions such as those found in Neural Machine Translation. We show that our attention mechanism learns meaningful alignments despite being constrained to a fixed representation after encoding. We encourage future work that explores the optimal values of K for various language tasks and examines whether or not it is possible to predict K based on the task at hand. We also encourage evaluating our models on other tasks that must deal with long sequences but have compact representations, such as summarization and question-answering, and further exploration of their effect on memory and training speed.



(a) softmax normalization functions, no position encoding

(b) sigmoid normalization functions, no position encoding

Figure 6: Attention scores at each step of decoding for on a sample from the sequence length 100 toy copy dataset. Individual attention vectors are highlighted in blue. (y-axis: source tokens; x-axis: target tokens)

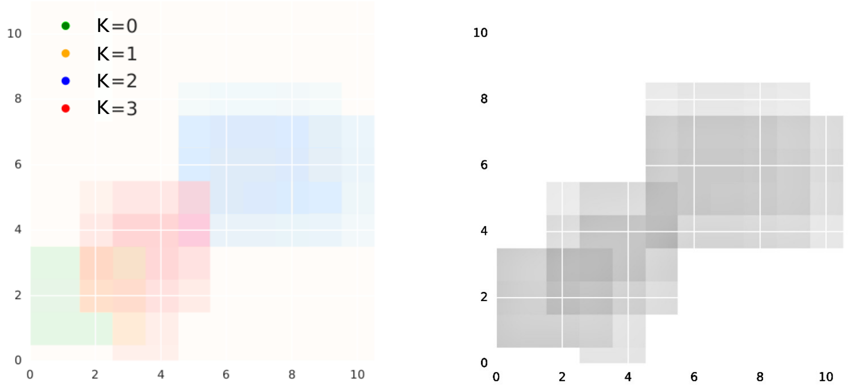


Figure 7: Attention scores at each step of decoding for $K = 4$ on a sample with sequence length 11. The subfigure on the left color codes each individual attention vector. (y-axis: source; x-axis: target)

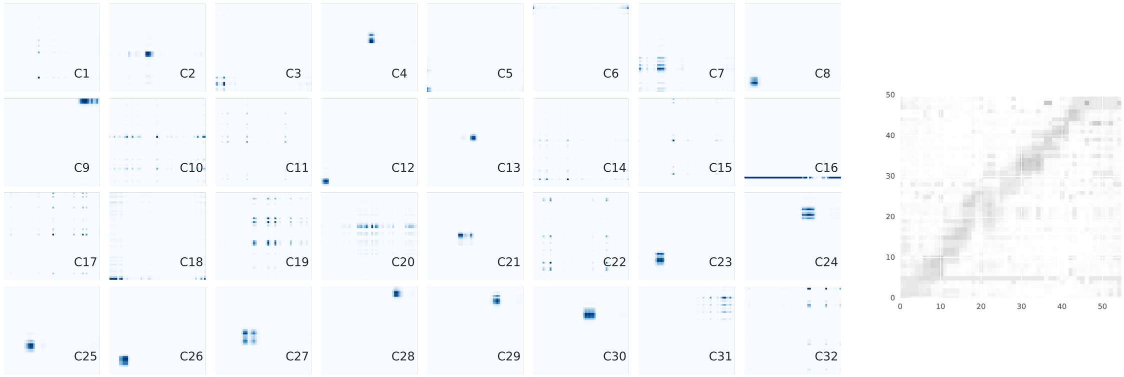


Figure 8: Attention scores at each step of decoding for en-de WMT translation task using model with sigmoid scoring functions and $K = 32$. The left subfigure displays each individual attention vector separately while the right subfigure displays the full combined attention. (y-axis: source; x-axis: target)

References

Roee Aharoni and Yoav Goldberg. 2016. Morphological inflection generation with hard monotonic attention. *CoRR* abs/1611.01487. http://arxiv.org/abs/1611.01487.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Denny Britz, Anna Goldie, Thang Luong, and Quoc Le. 2017. Massive Exploration of Neural Machine Translation Architectures. *CoRR* abs/1703.03906. http://arxiv.org/abs/1703.03906.

Michael Carl, Barbara Dragsted, and Arnt Lykke Jakobsen. 2011. A taxonomy of human translation styles. *Translation Journal* 16(2).

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2015. Listen, attend and spell. *CoRR* abs/1508.01211. http://arxiv.org/abs/1508.01211.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Jan Chorowski and Navdeep Jaitly. 2016. Towards better decoding and language model integration in sequence to sequence models. *CoRR* abs/1612.02695. http://arxiv.org/abs/1612.02695.

Alexandre de Brébisson and Pascal Vincent. 2016. A cheap linear attention mechanism with fast lookups and fixed-size representations. *CoRR* abs/1609.05866. http://arxiv.org/abs/1609.05866.

Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. 2016. A convolutional encoder model for neural machine translation. *CoRR* abs/1611.02344. http://arxiv.org/abs/1611.02344.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR* abs/1410.5401. http://arxiv.org/abs/1410.5401.

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

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *CoRR* abs/1610.10099. http://arxiv.org/abs/1610.10099.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *CoRR* abs/1510.03055. http://arxiv.org/abs/1510.03055.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR* abs/1508.04025. http://arxiv.org/abs/1508.04025.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016. Sequence-to-sequence rnns for text summarization. *CoRR* abs/1602.06023. http://arxiv.org/abs/1602.06023.

Colin Raffel, Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. *CoRR* abs/1704.00784. http://arxiv.org/abs/1704.00784.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *CoRR* abs/1509.00685. http://arxiv.org/abs/1509.00685.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. Weakly supervised memory networks. *CoRR* abs/1503.08895. http://arxiv.org/abs/1503.08895.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR* abs/1506.05869. http://arxiv.org/abs/1506.05869.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *CoRR* abs/1606.02960. http://arxiv.org/abs/1606.02960.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144. http://arxiv.org/abs/1609.08144.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *CoRR* abs/1502.03044. http://arxiv.org/abs/1502.03044.

Rotated Word Vector Representations and their Interpretability

Sungjoon Park and **JinYeong Bak** and **Alice Oh**
Department of Computing, KAIST, Republic of Korea
{sungjoon.park, jy.bak}@kaist.ac.kr, alice.oh@kaist.edu

Abstract

Vector representation of words improves performance in various NLP tasks, but the high-dimensional word vectors are very difficult to interpret. We apply several rotation algorithms to the vector representation of words to improve the interpretability. Unlike previous approaches that induce sparsity, the rotated vectors are interpretable while preserving the expressive performance of the original vectors. Furthermore, any pre-built word vector representation can be rotated for improved interpretability. We apply rotation to skip-grams and glove and compare the expressive power and interpretability with the original vectors and the sparse overcomplete vectors. The results show that the rotated vectors outperform the original and the sparse overcomplete vectors for interpretability and expressiveness tasks.

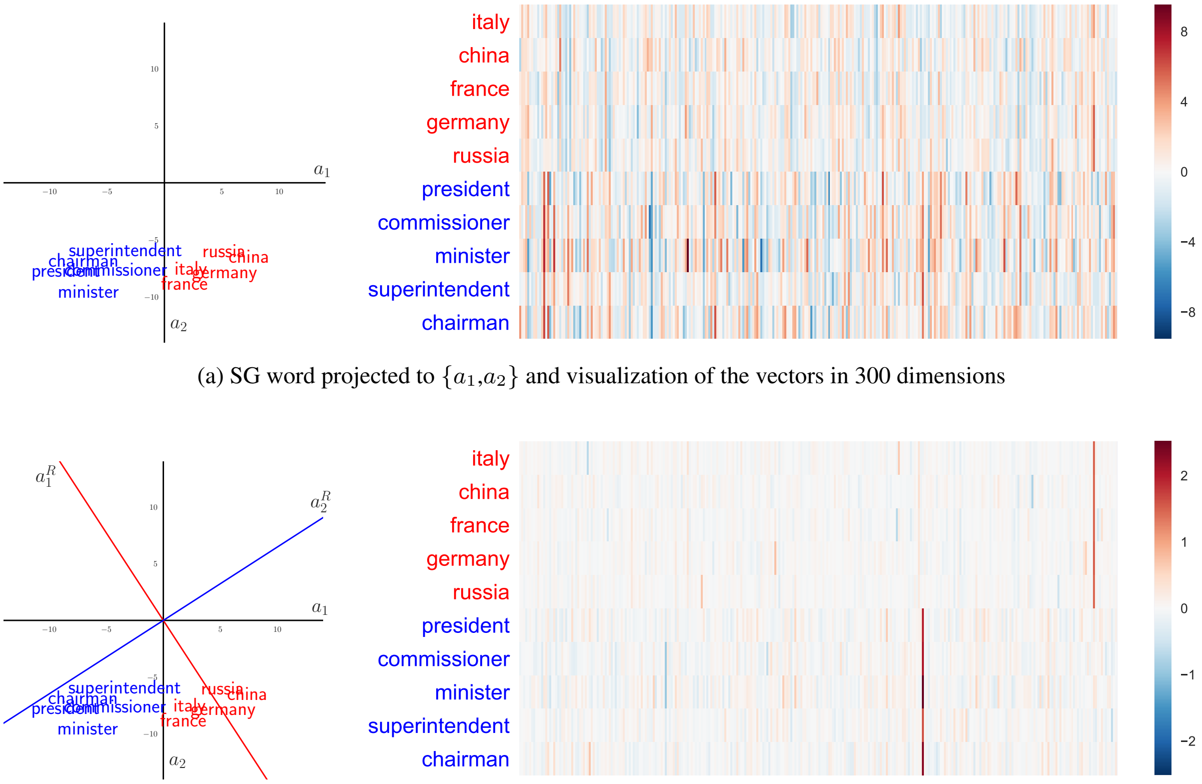
1 Introduction

Vector representations of words contain rich semantic and syntactic information and thus improve the performance of numerous natural language processing tasks. The vectors also play a basic role as an embedding layer in deep learning models for NLP, affecting the expressive performance of the model (Iyyer et al., 2014; Tai et al., 2015; Yang et al., 2016). However, the many dimensions comprising the vector representation are not amenable to interpretation.

Previous research on vector representation of words has proposed improving interpretability while keeping the expressive performance by inducing sparsity in word vector dimensions (Murphy et al., 2012; Fyshe et al., 2014). Recent research has proposed to build sparse vector representations from a large corpus and added the non-negativity constraint using improved projected gradient (Luo et al., 2015), while (Sun et al., 2016) learns l1-regularised vectors. But, these models cannot be learned over pre-trained word vectors based on skip-gram (Mikolov et al., 2013) or glove (Pennington et al., 2014) which are widely used. Faruqui et al. proposes an alternative approach to stand-alone models by forming sparse representations based on the pre-trained models. To do this, they use overcomplete vectors, which are much higher in dimensionality than the original vectors.

Unlike these sparsity-inducing approaches, we construct an interpretable word vector representation by using the pre-trained word vectors as input and using a basis rotation algorithm from the Exploratory Factor Analysis (EFA) literature used in developing psychological scales (Osborne and Costello, 2009). Like the word vector representation, every single item in the scale is represented as a numeric vector in the latent factor space. The set of item vectors are represented in a factor loading matrix, and the matrix is rotated such that the factors (i.e., dimensions) become interpretable. The rotation achieves a *Simple Structure* (Thurstone, 1947) through minimizing the row and the column complexity of the matrix (Crawford and Ferguson, 1970). We elaborate on this process in the next section. As in EFA, we rotate the word vector representation matrix to obtain dimension-wise interpretability while retaining the number of dimensions the same. For example, Figure 1 shows the rotated skip-gram vectors for two groups of words. These words are top five words of two dimensions from rotated Word2Vec.

Our main contribution is applying the matrix rotation algorithm from psychometric analysis to word vector representation models to improve the interpretability of the vector. This approach gives an answer to the question why and how word vec-



(a) SG word projected to $\{a_1,a_2\}$ and visualization of the vectors in 300 dimensions

(b) Rotated word vectors in $\{a_1^R,a_2^R\}$ and visualization of the vectors in 300 dimensions

Figure 1: Overview of rotating word vectors dimensions. We plot (a) unrotated and (b) rotated skip-gram word vectors in 2-D projected embedding space using PCA (left), and visualization of the vectors in original 300 dimensional space (right). Colors of words indicates the meaning of countries (Red) and positions (Blue). As in (b), after the dimensions are rotated, interpretability for each dimensions is improved having meaning of countries and positions.

tor representations work well by revealing a hidden structure of the original word vectors. That is, it is meaningful to transform the hard-to-interpret dimensions of the pre-built word vectors, which are widely used, to more interpretable vectors. We also show that the rotated vectors retain their effectiveness with respect to downstream tasks without re-building the vector representations.

Our method can be applied to any type of word vectors as a post-processing method such that it does not require a large corpus to be trained. In addition, it does not require additional number of dimensions so it does not increase the complexity of the model. Furthermore, we explore the characteristics of the rotated word vectors.

2 Factor Rotation

We take the rotation algorithm from the exploratory factor analysis (EFA) conducted to verify the construct validity of the psychological scale in development. For example, when validating a scale measuring respondents' latent factors, such as "Engineering problem solving" and "Interest in engineering", items should be similar within a factor, and distinguished between factors. As shown in Table 1, EFA projects every item into the latent factor space as an unrotated factor loading matrix. However, since it is unclear what the factor means, factor rotation is applied to the matrix that produces the rotated factor loading matrix which enhances the interpretability of the dimensions (Osborne, 2015).

2.1 Rotating Factors

The rotation algorithm transforms factor loading matrix to the *simple structure* which is much easier to interpret (Thurstone, 1947). It involves post-multiplication of a $p \times m$ input matrix A by an $m \times m$ square matrix T, to compute the rotated matrix Λ,

$$\Lambda = AT \tag{1}$$

Latent Factors	Items	Unrotated Factor Matrix		Rotated Factor Matrix	
		Factor 1	Factor 2	Factor 1	Factor 2
Engineering Problem Solving	How well did you feel prepared for:				
	(1) Defining what the problem really is	.759	-.389	**.830**	.153
	(2) Thinking up potential solutions to the problem	.784	-.392	**.861**	.157
	(3) Detailing how to implement the solution to the problem	.798	-.416	**.888**	.146
Interest in Engineering	(1) I find many topics in engineering to be interesting	.630	.521	.194	**.793**
	(2) Solving engineering problems is interesting to me	.660	.630	.149	**.901**
	(3) Engineering fascinates me	.669	.627	.158	**.906**

Table 1: An example of the factor rotation process to verify the construct validity of the psychological scale and its intended latent factor (left) in development. Items and loadings are from (Osborne, 2015).

which minimizes the cost function $f(\Lambda)$, also known as the rotation criterion. The function minimizes the *complexity* of the matrix, to make the rotated matrix have a few large values in a row or a column.

Minimizing the complexity allows non-binary values in the vector, and thus a more complex solution that the perfect simple structure. This is a more realistic solution since a solution with binary vectors may be misleading in representing the factor of interest (Yates, 1988; Browne, 2001). More details are described in the next subsection.

The intuition behind this approach is that inducing interpretability by factor rotation reforms the word embedding matrix to have a *simple structure* by linear transformation. It encourages each word vector (row) and dimension (column) to have a few large values, leading to more interpretable dimensions as shown in Fig 1.

2.2 Crawford-Ferguson Rotation Family

The rotation criterion introduced in Crawford and Ferguson is a family of complexity functions as follows:

$$\begin{aligned} f(\Lambda) = (1-\kappa)\ \Sigma_{i=1}^{p}\Sigma_{j=1}^{m}\Sigma_{l\neq j,l=1}^{m}\lambda_{ij}^{2}\lambda_{il}^{2} \\ +\ \kappa\ \Sigma_{j=1}^{m}\Sigma_{i=1}^{p}\Sigma_{l\neq i,l=1}^{m}\lambda_{ij}^{2}\lambda_{lj}^{2} \end{aligned} \tag{2}$$

where λ_{ij} is an element of Λ. The first term represents the row (item) complexity, and the second term represents the column (factor) complexity. The ratio between the two is adjusted by the parameter κ ($0 \leq k \leq 1$). The criterion is a generalized version of the widely used criteria, the orthomax family (Harman, 1960) which includes quartimax (Carroll, 1953; Ferguson, 1954; Neuhaus and Wrigley, 1954), varimax (Kaiser, 1958), and direct quartimin (Carroll, 1960). It effectively reflects the simple structure as well (Browne, 2001). In this work, we apply the following representative κ values in Table 2 (Sass and Schmitt, 2010).

	Quartimax	Varimax	Parsimax	FacParsim
κ	0	$\frac{1}{p}$	$\frac{m-1}{p+m-2}$	1

Table 2: Representative κ values used. As (Sass and Schmitt, 2010), we use 4 criterion referred to as CF-Quartimax, CF-Varimax, CF-Parsimax, CF-FacParsim. We omit 'CF-' for simplicity and do not separate the name of the kappa condition whether it is orthogonal or oblique. FacParsim stands for factor parsimony.

In addition, the constraints for the rotation matrix T can be applied in general. We can categorize the rotation as orthogonal and oblique based on the constraint. Orthogonal rotation assumes the correlation between the rotated dimensions is zero. Hence, the matrix should be an orthogonal matrix that with $m(m-1)/2$ constraints, satisfies:

$$T'T = I \tag{3}$$

Oblique rotations allow the correlation between dimension to be non-zero, resulting in m constraints satisfying:

$$diag(T^{-1}T^{-1'}) = I \tag{4}$$

The solution for the input matrix is computed by using the gradient projection algorithm (Jennrich, 2001, 2002). The algorithm minimizes equation 2 while satisfying the constraints of the rotation matrix.

3 Experimental Settings

We choose the Wikipedia English articles[1] to train the word vector models. The corpus contains 5.3M articles, 83M sentences and 1,676M tokens. For

[1] https://dumps.wikimedia.org/enwiki/20170120/

preprocessing, we leave only the alphanumeric tokens and apply lowercase to all words. Then we remove the words with frequency less than 50, and the size of the remaining vocabulary is 306,491.

We train skip-gram[2] (Mikolov et al., 2013) and glove[3] (Pennington et al., 2014) based on the corpus by using existing implementations. We set the window size to 5 for both skip-gram and glove. We set the number of negative samples to 5 and the number of dimensions to 300. We use the default values for the other hyperparameters. The size of the resulting word vector matrix is (306,491, 300).

We compare our model with two baseline models: sparse overcomplete vector representations (SOV) and the non-negative version of the SOV. We set the hyperparameters of these models as $\lambda = .5$, $\tau = 10^{-5}$, $K = 3000$ for SG, and $\lambda = 1.0$, $\tau = 10^{-5}$, $K = 3000$ for Glove (Faruqui et al., 2015). We excluded methods as baselines that construct interpretable word vectors using huge training corpora because our method works with pre-trained vectors.

We apply four rotation algorithms for each orthogonal and oblique rotation, listed in Table 2. Since we have two original word vector representations, we have 16 (4 x 2 x 2) rotated vectors in total. We implement the algorithm through TensorFlow (Abadi et al., 2016), and it is publicly available on GitHub[4].

4 Interpretability

In this section, we show how the rotation of word vectors results in improved dimension-wise interpretability using the word intrusion task. (Murphy et al., 2012; Faruqui et al., 2015; Sun et al., 2016).

4.1 Word Intrusion

Word intrusion task seeks to measure the semantic coherence of a set of words. For example, consider a set of words consists of ('daughter', 'wife', 'sister', 'mother', 'son') and add an 'intruder' word ('bigram') to the set. Since the words except intruder has similar meanings to each other, we can easily pick out the intruder to conclude that the five words are sharing coherent meanings.

We apply this task to measure interpretability of every word vector dimensions. If we choose the words with the highest embedding values for each of the dimensions (top words for that dimension) and add an random (intruder) word and see whether the intruder can be easily identified, then we can conclude the dimension is semantically coherent. In this way, we can measure the extent of interpretability of a dimension in vector representations by this task. Note that we pick top words for a dimension by looking only for the value of that dimension, ignoring values in the other dimensions.

Specifically, we first choose the top five words in each dimension, and then we choose an intruder word based on two criteria: 1) it is in the lower half of that dimension, and 2) it is in the top 10% in some other dimension. Also, we follow the settings of the measure ($k = 5$, top 10%) from previous works. We see similar results when we run experiments with larger k. (Murphy et al., 2012; Sun et al., 2016)

In the standard word intrusion task, human evaluators pick out the intruder words, and the results report the accuracy of the evaluators (Chang et al., 2009). But this approach would be impractical to use for all experimental conditions with 300 dimensions and the baselines, so we use the following distance ratio (DR) metric as an alternative approach in (Sun et al., 2016) with slight modifications. Another advantage of our metric is that it can be used to quantify the distance between the intruder and the non-intruder words. We define the overall metric as the average of the ratio between D^a_{inter} and D^a_{intra} over d dimensions as

$$DR_{overall} = \frac{1}{d} \frac{\Sigma^d_{a=1} D^a_{inter}}{\Sigma^d_{a=1} D^a_{intra}} \qquad (5)$$

where D^a_{intra} is the average distance of every pair among the top k words in dimension a

$$D^a_{intra} = \frac{\Sigma_{w_i} \Sigma_{w_j} dist(w_i, w_j)}{k(k-1)}, \qquad (6)$$

and D^a_{inter} is the average distance between the intruder word and each of the top k words in dimension a

$$D^a_{inter} = \frac{\Sigma_{w_i} dist(w_i, w_{intruder})}{k}. \qquad (7)$$

We define $dist(w_j, w_k)$ as the cosine distance between w_j and w_k. We set $k = 5$ and repeat this three times for each dimension a and use the average to compute $DR_{overall}$.

[2] https://radimrehurek.com/gensim
[3] https://nlp.stanford.edu/projects/glove
[4] https://github.com/SungjoonPark/factor_rotation

	SG	Glove
Original	1.258	1.095
SOV	1.089	1.050
SOV (non-neg)	1.081	1.074
Quartimax (orthogonal)	1.479	1.248
Varimax (orthogonal)	1.477	**1.289**
Parsimax (orthogonal)	**1.596**	1.261
FacParsim (orthogonal)	1.300	1.102
Quartimax (oblique)	1.385	**1.225**
Varimax (oblique)	**1.398**	1.222
Parsimax (oblique)	1.386	1.174
FacParsim (oblique)	1.145	1.081

Table 3: Overall distance ratio ($DR_{overall}$) of the original, sparse overcomplete vectors, and the rotated (orthogonal and oblique) vector representations. Rotated vectors show improved interpretability over SOV and the original.

4.2 Results

Table 3 shows the results of word intrusion in terms of the distance ratio metric. Overall, the results of the rotated vector representations show improvements over SOV and the original word vector representations. For skip-grams, orthogonal parsimax shows the best result while for Glove, orthogonal varimax outperforms the others. Among oblique rotation, varimax and quartimax show better performance than factor parsimony.

In general, interpretability varies with different values of κ. It increases when κ is close to zero and decreases when κ is close to one, putting more weight on the column complexity. Also, orthogonal rotation shows better performance than oblique rotation when κ is controlled.

4.3 Qualitative Examples

We present the top words of five dimensions for skip-gram and rotated skip-gram (parsimax-orthogonal) in Table 4. The dimensions shown are randomly selected for both conditions.

Overall, the top words in each dimension of skip-gram do not clearly show a common topic among them. Only a few dimensions out of 300 are interpretable, such as the second row in the table which is related to numbers. The overall distance ratio of the original vectors is slightly higher than one.

For the rotated word vectors, the top words show clear semantic coherence. The first row shows words about social network services, the second row is about biology, the third row is about geographical locations in the US, and the fourth is about paintings. As the last row shows, some of these dimensions represent syntactic features.

Model	Topwords
SG	householder, asked, indicted, there, ethnic
	score, two, best, three, four
	mining, footballer, population, laps, settled
	density, census, fourier, editor, photos
	money, toured, season, announced, banned
Rot. SG	twitter, facebook, youtube, myspace, internet
	receptors, receptor, neurons, apoptosis, neuronal
	pennsylvania,ohio,maryland,philadelphia,illinois
	paintings, portraits, painting, drawings, painter
	that, which, when, where, but

Table 4: 5 top words for the original and the rotated skip-gram word representations. The rotated vectors show common semantic or syntactic coherence while the original vectors do not.

5 Expressive Performance

We evaluate the expressive power of word vector representations on the following tasks and report Spearman's correlation coefficient for the first task, and accuracy for the other tasks. Table 5 shows the results.

5.1 Evaluation

We briefly describe the seven benchmark tasks: word similarity and semantic/syntactic analogy, and four classification tasks. For the classification tasks, we average the word vectors in each training sentence or phrase to use them as features. SVM and random forest classifier are trained to predict the target values, and hyperparameters are tuned on the validation set.

Word Similarity (Simil.) SimLex-999 (Hill et al., 2016) presented to evaluate the similarity of word pairs, rather than relatedness. We compute the cosine similarity between the given word pairs, and report the Spearman's correlation coefficient as a measure of consistency between the similarity and human ratings.

Semantic and Syntactic Analogies (Analg. sem, syn). The second and third tasks are word analogy tasks proposed by (Mikolov et al., 2013). The semantic task includes 8,869 questions (sem) and the syntactic task includes 10,675 questions (syn).

Sentiment Analysis (Sent.) The first classification task is sentiment classification on the movie reviews (Socher et al., 2013). This dataset contains

	# dims	Simil.	Analg. (sem)	Analg. (syn)	Sent.	Ques.	Topics (Sp.)	NP brckt.
Skip-Gram	300	.374	.668	**.652**	.741	.920	**.960**	.812
SOV	3000	.390	.640	.594	.751	.910	.955	**.836**
SOV (non-neg)	3000	.384	.566	.480	**.761**	.918	**.960**	.829
Quartimax (orthogonal)	300	.374	.668	**.652**	.744	.922	.956	.822
Varimax (orthogonal)	300	.374	.668	**.652**	.744	.922	.956	.822
Parsimax (orthogonal)	300	.374	.668	**.652**	.744	.922	.956	.819
FacParsim (orthogonal)	300	.374	.668	**.652**	.744	.922	.956	.822
Quartimax (oblique)	300	**.422**	**.673**	.624	.755	**.932**	.955	.820
Varimax (oblique)	300	**.422**	**.673**	.624	.755	**.932**	.955	.820
Parsimax (oblique)	300	.421	.671	.623	.752	**.932**	.956	.826
FacParsim (oblique)	300	.417	.660	.620	.751	.928	.952	.820

Table 5: Evaluation results of the original skip-gram, sparse overcomplete vectors (SOV), and the rotated (orthogonal and oblique) word vectors on various tasks. The left three columns show tasks based on cosine similarity, and the right four columns show classification tasks using average word vectors as features. Overall, the rotated word vectors show higher or comparable performance to that of the SOV and the original. We observe a similar pattern in Glove as well.

6,920, 872, 1,821 sentences for training, development, and test, respectively. The goal of this task is to predict positive or negative sentiment of the reviews.

Question Classification (Ques.) Next, we use TREC dataset to classify categories of the questions (Faruqui et al., 2015). We divide the dataset into 4,952, 500, 500 for training, development, and test. The dataset has six types of questions including about person, location, etc.

Topic Classification (Topics: Sp.) Next, we obtain the 20 newsgroup dataset to classify Sports (baseball vs. hockey) topics (Yogatama and Smith, 2014; Faruqui et al., 2015). The dataset consists of 958, 239, 796 for training, development, and test.

NP bracketing (NP brckt.) The final task is classifying noun phrases in terms of bracketing (Lazaridou et al., 2013; Faruqui et al., 2015). Each phrase consists of three words, and the task is to predict the correct bracketing to match the similar words. We compute the average of NPs and perform ten-fold cross-validation over 2,227 phrases. The classifiers are trained and the hyperparameters are tuned for every fold.

5.2 Results

Word Similarity and Analogies We observe improved performance of oblique rotation of word vectors compared to the original and the SOV in word similarity and semantic analogy tasks. In the syntactic analogy, orthogonal rotation shows the same performance as the original. Note that the orthogonal rotations preserve the cosine-based expressive performances because the cosine similarity between any two vectors does not change after the orthogonal rotation.

Classification Tasks The SOV models show slightly higher performance except the question classification task. However, we can observe the rotated word vectors have improved performance over the original vectors. We observe a similar pattern in Glove as well. In conclusion, the rotated representations preserve the expressive power of the original word vectors, and it is quite close to that of the sparse representation with 10 times larger dimensionality.

6 Understanding Rotated Word Vectors

In this section, we perform several experiments to understand the characteristics of the rotated word vector representations.

6.1 Directionality

One conventional approach to make the word vectors to be more interpretable is by forcing the representation to have non-negative values (Faruqui et al., 2015; Luo et al., 2015). However, the dimensions in the rotated vectors are not non-negative, spread in both directions. Hence, we investigate the relationship between the directionality (positive / negative) and interpretability.

	(A)	(B)	(C)	(D)	(E)	(F)	(G)
	Desc	Asc	Cor	Cor	Cor	Cor	DR
	(Hi)	(Lo)	(Hi, Lo)	(abs, DR)	(abs, intra)	(abs, inter)	(abs)
Quartimax (orthogonal)	1.479	1.507	-.452***	.843***	-.835***	.204***	2.045
Varimax (orthogonal)	1.477	1.478	-.431***	.847***	-.840***	.205***	2.004
Parsimax (orthogonal)	1.596	1.499	-.729***	.845***	-.836***	.216***	2.442
FacParsim (orthogonal)	1.300	1.309	-.114*	.536***	-.549***	.056	1.384
Quartimax (oblique)	1.385	1.464	-.692***	.879***	-.880***	.276***	1.997
Varimax (oblique)	1.398	1.465	-.684***	.879***	-.878***	.204***	2.022
Parsimax (oblique)	1.386	1.463	-.696***	.886***	-.883***	.279***	1.993
FacParsim (oblique)	1.145	1.152	.006	.382***	-.369***	.037	1.171

Table 6: Overall distance ratio based on the top words extracted from the values in word vectors sorted by descending order (Hi) and ascending order (Lo). Cor(Hi, Lo) is correlation between two distance ratios based on both directions. Next three columns present correlation between the absolute word vector values of the top words and distance ratios. The last columns shows selective distance ratio measure. The results implies generally both direction is interpretable, one direction is more interpretable than the other within a dimension, and larger absolute value in a dimension means higher interpretability. (* $p < .05$, ** $p < .01$, *** $p < .001$)

Overall Interpretability of both directions The first two columns (A) and (B) in table 6 show the overall distance ratio computed over the top words extracted by descending order and ascending order, respectively. In other words, (A) refers to the top words having the highest positive values in each dimension, while (B) uses the lowest negative values. Note that we used descending order in word intrusion task in the previous section.

Interestingly, the overall distance ratios in both directions are comparable to each other. On average, both sides of a dimension are more interpretable than the unrotated vector representations except the oblique factor parsimony rotation.

Interpretability of both directions within a dimension Next, we compare the interpretability of both directions within a dimension. We first define the distance ratio of an individual dimension a as follows:

$$DR^a = \frac{D^a_{inter}}{D^a_{intra}} \tag{8}$$

We compute the ratio by using top words extracted from positive and negative directions for every dimension, and compute Spearman's correlation of the distance ratio pairs. Table 6 column (C) shows the results. All of the rotation conditions except the oblique factor parsimony shows significant ($p < .05$) negative correlation, meaning that both directions are hard to be highly interpretable within a dimension simultaneously.

Dir.	Topwords
+	depends, depend, rely, focused, focuses
-	on, upon, onto, again, until
+	years, month, weeks, days, decades
-	many, several, ago, numerous, various
+	that, which, when, where, but
-	consists, includes, provides, contains, serves
+	criticizes, excelled, tended, much, criticized
-	october, july, april, september, june
+	were, hoc, recently, their, had
-	largest, oldest, longest, biggest, tallest

Table 7: Examples of top words in both directions. The words are extracted from a part of the orthogonal parsimax rotated skip-gram word vectors.

Case Study We present the top words in both directions for some dimensions of orthogonal parsimax rotated word vectors. As shown in table 7, some dimensions show a relationship between the opposite directions that they consist of consecutively used words, such as ”rely on”, ”depends upon”, ”which includes”, ”that contains”, ”many years”, ”weeks ago”. However, other dimensions show that one direction is relatively more interpretable than the other direction.

6.2 Selecting the Direction

Next, it is natural to question whether the larger absolute value in word vectors means higher interpretability, regardless of its directionality. We verify the relation between them by investigating the size of the absolute value in a dimension and the individual distance ratios.

Relation to distance ratio Table 6 column (D) presents Spearman's correlation between individual distance ratio and the mean absolute vector value of top words for that dimension. The fifth column (E) also shows the correlation between the intra-distance among the top words and the mean absolute value, and the sixth column (F) is the relationship of the inter-distance among the top words and the intruder and the mean absolute value.

Correlation coefficients show that the larger mean absolute value means higher interpretability for that dimension. In detail, there exists tendencies that larger mean absolute value of dimension reduces the intra-distances among the top words while increasing the inter-distances among the top words and the intruder.

Overall, we summarize our findings as follows: 1) generally both directions are somewhat interpretable, 2) one direction is usually more interpretable than the other within a dimension, and 3) a larger absolute value in a dimension means higher interpretability of the dimension.

Selective Distance Ratio We can select a more interpretable direction for each dimension through inspecting the mean absolute value of the top words in both directions. If we choose a direction that has a larger mean absolute value among the top words, each dimension should be easier to interpret.

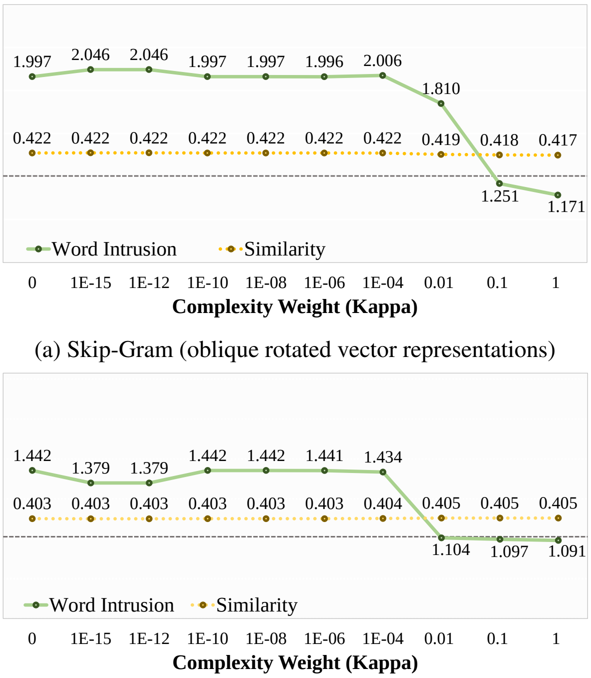
Table 6 column (G) presents this distance ratio computed on the rotated vectors, resulting in increased distance ratio values. We name this ratio as the overall selective distance ratio. This measure could be effectively used when vector representation is interpretable in both directions.

6.3 Effect of κ

We explore the effect on performance of the ratio between the row and the column complexity of the rotation criteria. As shown in section 4, choosing an appropriate κ is important for interpretability.

We set the κ value from zero to one and the numbers divided on a log scale. We run the word similarity task and the word intrusion to evaluate the performance. We present Spearman's correlation and the selective overall distance ratio.

Figure 2 shows that the performance of the similarity task tends not to change regardless of κ, however, the selective distance ratio starts to decrease when $\kappa > .01$. Considering the ratio between the number of rows and columns of the word vector matrix, giving too much weight to the column complexity results in degraded interpretability.



(a) Skip-Gram (oblique rotated vector representations)

(b) Glove (oblique rotated vector representations)

Figure 2: Spearman's correlation of the word similarity and the selective distance ratio of word intrusion changes over κs, computed over oblique rotated (a) skip-gram and (b) glove vectors. Dashed line is original performance for each task. Word similarity does not change regardless of κs, while the distance ratio falls when κ is larger than 1e-4.

In our experiments, κ values of the quartimax, varimax, and parsimax rotation are computed as 0, 3e-06, 1e-04 respectively. Based on the results, our selection of kappas have shown interpretability improvement effectively, compared to factor parsimony ($\kappa = 1$). We observe these tendencies in orthogonal rotations as well.

6.4 Effect of the Number of Dimensions

To investigate the effect of the number of dimensions to interpretability of dimensions, we also measure the overall distance ratio ($DR_{overall}$) on 50, 100 and 200 dimensions of unrotated skip-gram and parsimax (orthogonal) and varimax (oblique) rotated word vectors.

Figure 3 shows the results. For all settings, the rotated vectors orthogonal (parsimax) and oblique (varimax) show higher $DR_{overall}$ score than the original skip-gram vectors.

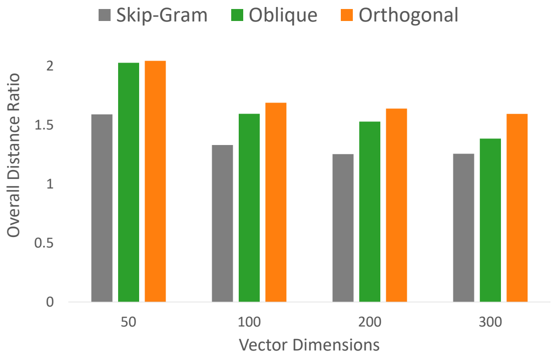


Figure 3: Overall distance ratio ($DR_{overall}$) over word vector dimensions. The rotated vectors (parsimax-orthogonal and varimax-oblique) show higher $DR_{overall}$ score than the original skip-gram vectors.

7 Related Work

Since distributed representations play an important role in various NLP tasks, they are applied to semantics (Herbelot and Vecchi, 2015; Qiu et al., 2015; Woodsend and Lapata, 2015), with incorporating external information to them (Tian et al., 2016; Nguyen et al., 2016). In addition, finding interpretable regularities from the representations is often conducted through non-negative and sparse coding (Murphy et al., 2012; Faruqui et al., 2015; Luo et al., 2015; Kober et al., 2016), and regularization (Sun et al., 2016). Instead, our approach is using rotation, showing better results in terms of interpretability.

Meanwhile, various rotation methods are proposed such as CF-family (Crawford and Ferguson, 1970), Infomax (McKeon, 1968), Minimum Entropy (Jennrich, 2006), Geomin (Yates, 1988), procrustues (Hurley and Cattell, 1962), and promax rotation criteria. (Hendrickson and White, 1964). Incorporating prior knowledge about rotated matrix is possible through target rotations (Harman, 1960; Browne, 1972a,b) are proposed as well. There are various ways to rotated dimensions, we select a CF-family that covers frequently used rotation methods in practice.

8 Conclusion and Discussions

In this paper, we applied the rotation algorithm to improve interpretability of distributed representation of words. We applied quartimax, varimax, parsimax and factor parsimony rotation by using the Crawford-Ferguson rotation criteria, then we constructed the rotated word vector representations. We evaluated the expressive performance and interpretability for the rotated word vectors by word similarity, analogy, classification, and word intrusion task. The results show that the rotated word vector representations are highly interpretable with preserving expressive performance.

In addition, we explored the characteristics of the rotated word vectors: we observed 1) increased interpretability in both directions and 2) the positive relation between absolute value of the dimension and interpretability. Based on these observations, we proposed the selective distance ratio to measure and maximize the interpretability when the vector representation has interpretable meaning in both directions. We expect that the rotation algorithm can be easily applied to other word vector representations.

Our results imply that a rotated word vector can be used to understand what the word vectors are comprised of. Since a lexicon can be decomposed into morphemes, a word can have multiple meaning as a polysemy, contain information of syntactic structure in its meaning (Carpenter et al., 1995; MacDonald et al., 1994; Trueswell et al., 1994), or it can be divided into a variety of sub-components. Hence, we can investigate the lexical semantics of words by exploring the dimensions for which a word has higher values.

In addition, there are practical implications of interpreting the dimensions as well. Based on the meanings, we can remove irrelevant dimensions for a specific task of interest, in order to secure more efficient storage of the vectors and decrease the complexity of downstream NLP models. We will examine the issues in future work.

We plan to explore following issues. First, we apply target rotation (Harman, 1960; Browne, 1972a,b) to incorporate prior knowledge when constructing the rotated word vector representations. Second, we will investigate the interpretability of hidden structures of neural networks for NLP tasks such as (Yang et al., 2016; Li et al., 2016), when the rotated word vectors are used as an embedding layer.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2016R1A2B4016048)

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of OSDI*.

Michael W Browne. 2001. An overview of analytic rotation in exploratory factor analysis. *Multivariate Behavioral Research*, 36(1):111–150.

MW Browne. 1972a. Oblique rotation to a partially specified target. *British Journal of Mathematical and Statistical Psychology*, 25(2):207–212.

MW Browne. 1972b. Orthogonal rotation to a partially specified target. *British Journal of Mathematical and Statistical Psychology*, 25(1):115–120.

Patricia A Carpenter, Akira Miyake, and Marcel Adam Just. 1995. Language comprehension: Sentence and discourse processing. *Annual review of psychology*, 46(1):91–120.

JB Carroll. 1960. Ibm 704 program for generalized analytic rotation solution in factor analysis. *Unpublished manuscript, Harvard University*, 9:324.

John B Carroll. 1953. An analytical solution for approximating simple structure in factor analysis. *Psychometrika*, 18(1):23–38.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L. Boyd-graber, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proceedings of NIPS*, pages 288–296.

Charles B Crawford and George A Ferguson. 1970. A general rotation criterion and its use in orthogonal rotation. *Psychometrika*, 35(3):321–332.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah Smith. 2015. Sparse overcomplete word vector representations. In *Proceedings of ACL*.

George A Ferguson. 1954. The concept of parsimony in factor analysis. *Psychometrika*, 19(4):281–290.

Alona Fyshe, Partha P Talukdar, Brian Murphy, and Tom M Mitchell. 2014. Interpretable semantic vectors from a joint model of brain-and text-based meaning. In *Proceedings of ACL*.

Harry H Harman. 1960. *Modern factor analysis*. University of Chicago Press.

Alan E Hendrickson and Paul Owen White. 1964. Promax: A quick method for rotation to oblique simple structure. *British journal of statistical psychology*, 17(1):65–70.

Aurélie Herbelot and Eva Maria Vecchi. 2015. Building a shared world: mapping distributional to model-theoretic semantic spaces. In *Proceedings of EMNLP*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

John R Hurley and Raymond B Cattell. 1962. The procrustes program: Producing direct rotation to test a hypothesized factor structure. *Systems Research and Behavioral Science*, 7(2):258–262.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proceedings of ACL*.

Robert I Jennrich. 2001. A simple general procedure for orthogonal rotation. *Psychometrika*, 66(2):289–306.

Robert I Jennrich. 2002. A simple general method for oblique rotation. *Psychometrika*, 67(1):7–19.

Robert I Jennrich. 2006. Rotation to simple loadings using component loss functions: The oblique case. *Psychometrika*, 71(1):173–191.

Henry F Kaiser. 1958. The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3):187–200.

Thomas Kober, Julie Weeds, Jeremy Reffin, and David Weir. 2016. Improving sparse word representations with distributional inference for semantic composition. In *Proceedings of EMNLP*.

Angeliki Lazaridou, Eva Maria Vecchi, and Marco Baroni. 2013. Fish transporters and miracle homes: How compositional distributional semantics can help np parsing. In *Proceedings of EMNLP*.

Shaohua Li, Tat-Seng Chua, Jun Zhu, and Chunyan Miao. 2016. Generative topic embedding: a continuous representation of documents. In *Proceedings of ACL*.

Hongyin Luo, Zhiyuan Liu, Huan-Bo Luan, and Maosong Sun. 2015. Online learning of interpretable word embeddings. In *Proceedings of EMNLP*.

Maryellen C MacDonald, Neal J Pearlmutter, and Mark S Seidenberg. 1994. The lexical nature of syntactic ambiguity resolution. *Psychological review*, 101(4):676.

JJ McKeon. 1968. Rotation for maximum association between factors and tests. *Unpublished manuscript, Biometric Laboratory, George Washington University*.

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

Brian Murphy, Partha Pratim Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceddings of COLING*.

Jack O Neuhaus and Charles Wrigley. 1954. The quartimax method. *British Journal of Statistical Psychology*, 7(2):81–91.

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

Jason W Osborne. 2015. What is rotating in exploratory factor analysis. *Practical Assessment, Research & Evaluation*, 20(2):2.

Jason W Osborne and Anna B Costello. 2009. Best practices in exploratory factor analysis: Four recommendations for getting the most from your analysis. *Pan-Pacific Management Review*, 12(2):131–146.

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

Likun Qiu, Yue Zhang, and Yanan Lu. 2015. Syntactic dependencies and distributed word representations for chinese analogy detection and mining. In *Proceedings of EMNLP*.

Daniel A Sass and Thomas A Schmitt. 2010. A comparative investigation of rotation criteria within exploratory factor analysis. *Multivariate Behavioral Research*, 45(1):73–103.

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

Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2016. Sparse word embeddings using l1 regularized online learning. In *Proceedings of IJCAI*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*.

Louis Leon Thurstone. 1947. *Multiple factor analysis*. University of Chicago Press.

Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2016. Learning semantically and additively compositional distributional representations. In *Proceedings of ACL*.

John C Trueswell, Michael K Tanenhaus, and Susan M Garnsey. 1994. Semantic influences on parsing: Use of thematic role information in syntactic ambiguity resolution. *Journal of memory and language*, 33(3):285.

Kristian Woodsend and Mirella Lapata. 2015. Distributed representations for unsupervised semantic role labeling. In *Proceedings of EMNLP*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL-HLT*.

Allen Yates. 1988. *Multivariate exploratory data analysis: A perspective on exploratory factor analysis*. Suny Press.

Dani Yogatama and Noah A Smith. 2014. Linguistic structured sparsity in text categorization. In *Proceedings of ACL*.

A causal framework for explaining the predictions of black-box sequence-to-sequence models

David Alvarez-Melis and **Tommi S. Jaakkola**
CSAIL, MIT
{davidam, tommi}@csail.mit.edu

Abstract

We interpret the predictions of any black-box structured input-structured output model around a specific input-output pair. Our method returns an "explanation" consisting of groups of input-output tokens that are causally related. These dependencies are inferred by querying the black-box model with perturbed inputs, generating a graph over tokens from the responses, and solving a partitioning problem to select the most relevant components. We focus the general approach on sequence-to-sequence problems, adopting a variational autoencoder to yield meaningful input perturbations. We test our method across several NLP sequence generation tasks.

1 Introduction

Interpretability is often the first casualty when adopting complex predictors. This is particularly true for structured prediction methods at the core of many natural language processing tasks such as machine translation (MT). For example, deep learning models for NLP involve a large number of parameters and complex architectures, making them practically black-box systems. While such systems achieve state-of-the-art results in MT (Bahdanau et al., 2014), summarization (Rush et al., 2015) and speech recognition (Chan et al., 2015), they remain largely uninterpretable, although attention mechanisms (Bahdanau et al., 2014) can shed some light on how they operate.

Stronger forms of interpretability could offer several advantages, from trust in model predictions, error analysis, to model refinement. For example, critical medical decisions are increasingly being assisted by complex predictions that should lend themselves to easy verification by human experts. Without understanding how inputs get mapped to the outputs, it is also challenging to diagnose the source of potential errors. A slightly less obvious application concerns model improvement (Ribeiro et al., 2016) where interpretability can be used to detect biases in the methods.

Interpretability has been approached primarily from two main angles: *model interpretability*, i.e., making the architecture itself interpretable, and *prediction interpretability*, i.e., explaining particular predictions of the model (cf. (Lei et al., 2016)). Requiring the model itself to be transparent is often too restrictive and challenging to achieve. Indeed, prediction interpretability can be more easily sought *a posteriori* for black-box systems including neural networks.

In this work, we propose a novel approach to prediction interpretability with only oracle access to the model generating the prediction. Following (Ribeiro et al., 2016), we turn the local behavior of the model around the given input into an interpretable representation of its operation. In contrast to previous approaches, we consider structured prediction where both inputs and outputs are combinatorial objects, and our explanation consists of a summary of operation rather than a simpler prediction method.

Our method returns an "explanation" consisting of sets of input and output tokens that are causally related under the black-box model. Causal dependencies arise from analyzing perturbed versions of inputs that are passed through the black-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 412–421
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

box model. Although such perturbations might be available in limited cases, we generate them automatically. For sentences, we adopt a variational autoencoder to produce semantically related sentence variations. The resulting inferred causal dependencies (interval estimates) form a dense bipartite graph over tokens from which explanations can be derived as robust min-cut k-partitions.

We demonstrate quantitatively that our method can recover known dependencies. As a starting point, we show that a grapheme-to-phoneme dictionary can be largely recovered if given to the method as a black-box model. We then show that the explanations provided by our method closely resemble the attention scores used by a neural machine translation system. Moreover, we illustrate how our summaries can be used to gain insights and detect biases in translation systems. Our main contributions are:

- We propose a general framework for explaining structured black-box models
- For sequential data, we propose a variational autoencoder for controlled generation of input perturbations required for causal analysis
- We evaluate the explanations produced by our framework on various sequence-to-sequence prediction tasks, showing they can recover known associations and provide insights into the workings of complex systems.

2 Related Work

There is a wide body of work spanning various fields centered around the notion of "interpretability". This term, however, is underdetermined, so the goals, methods and formalisms of these approaches are often non-overlapping (Lipton, 2016). In the context of machine learning, perhaps the most visible line of work on interpretability focuses on medical applications (Caruana et al., 2015), where trust can be a decisive factor on whether a model is used or not. With the ever-growing success and popularity of deep learning methods for image processing, recent work has addressed interpretability in this setting, usually requiring access to the method's activations and gradients (Selvaraju et al., 2016), or directly modeling how influence propagates (Bach et al., 2015). For a broad overview of interpretability in machine learning, we refer the reader to the recent survey by Doshi-Velez and Kim (2017).

Most similar to this work are the approaches of Lei et al. (2016) and Ribeiro et al. (2016). The former proposes a model that justifies its predictions in terms of fragments of the input. This approach formulates explanation generation as part of the learning problem, and, as most previous work, only deals with the case where predictions are scalar or categorical. On the other hand, Ribeiro et al. (2016) propose a framework for explaining the predictions of black-box classifiers by means of locally-faithful interpretable models. They focus on sparse linear models as explanations, and rely on local perturbations of the instance to explain. Their model assumes the input directly admits a fixed size interpretable representation in euclidean space, so their framework operates directly on this vector-valued representation.

Our method differs from—and can be thought of as generalizing—these approaches in two fundamental aspects. First, our framework considers both inputs and outputs to be structured objects thus extending beyond the classification setting. This requires rethinking the notion of explanation to adapt it to variable-size combinatorial objects. Second, while our approach shares the locality and model-agnostic view of Ribeiro et al. (2016), generating perturbed versions of structured objects is a challenging task by itself. We propose a solution to this problem in the case of sequence-to-sequence learning.

3 Interpreting structured prediction

Explaining predictions in the structured input-structured output setting poses various challenges. As opposed to scalar or categorical prediction, structured predictions vary in size and complexity. Thus, one must decide not only how to explain the prediction, but also what parts of it to explain. Intuitively, the "size" of an explanation should grow with the size of the input and output. A good explanation would ideally also decompose into *cognitive chunks* (Doshi-Velez and Kim, 2017): basic units of explanation which are a priori bounded in size. Thus, we seek a framework that naturally decomposes an explanation into (potentially several) *explaining components*, each of which justifies, from the perspective of the black-box model, parts of the output relative to the parts of the input.

Formally, suppose we have a black-box model $F : \mathcal{X} \to \mathcal{Y}$ that maps a structured input $\mathbf{x} \in \mathcal{X}$ to a structured output $\mathbf{y} \in \mathcal{Y}$. We make no assumptions on the spaces $\mathcal{X}, \mathcal{Y}$, except that their elements admit a feature-set representation $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$, $\mathbf{y} = \{y_1, y_2, \dots, y_m\}$. Thus, $\mathbf{x}$ and $\mathbf{y}$ can be sequences, graphs or images. We refer to the elements x_i and y_j as units or "tokens" due to our motivating application of sentences, though everything in this work holds for other combinatorial objects.

For a given input output pair $(\mathbf{x}, \mathbf{y})$, we are interested in obtaining an *explanation* of $\mathbf{y}$ in terms of $\mathbf{x}$. Following (Ribeiro et al., 2016), we seek explanations via *interpretable representations* that are both i) *locally faithful*, in the sense that they approximate how the model behaves in the vicinity of $\mathbf{x}$, and ii) *model agnostic*, that is, that do not require any knowledge of F. For example, we would like to identify whether token x_i is a likely cause for the occurrence of y_j in the output when the input context is $\mathbf{x}$. Our assumption is that we can summarize the behavior of F around $\mathbf{x}$ in terms of a weighted bipartite graph $G = (V_x \cup V_y, E)$, where the nodes V_x and V_y correspond to the elements in $\mathbf{x}$ and $\mathbf{y}$, respectively, and the weight of each edge E_{ij} corresponds to the influence of the occurrence of token x_i on the appearance of y_j. The bipartite graph representation suggests naturally that the explanation be given in terms of explaining components. We can formalize these components as subgraphs $G^k = (V_x^k \cup V_y^k, E^k)$, where the elements in V_x^k are likely causes for the elements in V_y^k. Thus, we define an explanation of $\mathbf{y}$ as a collection of such components: $E_{x \to y} = \{G^1, \dots, G^k\}$.

Our approach formalizes this framework through a pipeline (sketched in Figure 1) consisting of three main components, described in detail in the following section: a perturbation model for exercising F locally, a causal inference model for inferring associations between inputs and predictions, and a selection step for partitioning and selecting the most relevant sets of associations. We refer to this framework as a *structured-output causal rationalizer* (SocRat).

A note on alignment models When the inputs and outputs are sequences such as sentences, one might envision using an alignment model, such as those used in MT, to provide an explanation. This differs from our approach in several respects. Specifically, we focus on explaining the behavior of the "black box" mapping F only locally, around the current input context, not globally. Any global alignment model would require access to substantial parallel data to train and would have varying coverage of the local context around the specific example of interest. Any global model would likely also suffer from misspecification in relation to F. A more related approach to ours would be an alignment model trained locally based on the same perturbed sentences and associated outputs that we generate.

4 Building blocks

4.1 Perturbation Model

The first step in our approach consists of obtaining *perturbed* versions of the input: semantically similar to the original but with potential changes in elements and their order. This is a major challenge with any structured inputs. We propose to do this using a variational autoencoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014). VAEs have been successfully used with fixed dimensional inputs such as images (Rezende and Mohamed, 2015; Sønderby et al., 2016) and recently also adapted to generating sentences from continuous representations (Bowman et al., 2016). The goal is to introduce the perturbation in the continuous latent representation rather than directly on the structured inputs.

A VAE is composed of a probabilistic encoder $\textsc{Enc} : \mathcal{X} \to \mathbb{R}^d$ and a decoder $\textsc{Dec} : \mathbb{R}^d \to \mathcal{X}$. The encoder defines a distribution over latent codes $q(\mathbf{z}|\mathbf{x})$, typically by means of a two-step procedure that first maps $\mathbf{x} \mapsto (\boldsymbol{\mu}, \boldsymbol{\sigma})$ and then samples $\mathbf{z}$ from a gaussian distribution with these parameters. We can leverage this stochasticity to obtain perturbed versions of the input

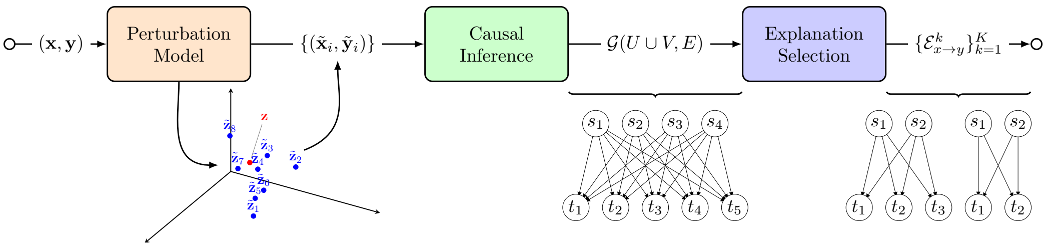

$(\mathbf{x}, \mathbf{y})$
Perturbation Model
$\{(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i)\}$
Causal Inference
$\mathcal{G}(U \cup V, E)$
Explanation Selection
$\{\mathcal{E}^k_{x \to y}\}_{k=1}^K$
$\mathbf{z}$
$\tilde{\mathbf{z}}_1$ $\tilde{\mathbf{z}}_2$ $\tilde{\mathbf{z}}_3$ $\tilde{\mathbf{z}}_4$ $\tilde{\mathbf{z}}_5$ $\tilde{\mathbf{z}}_6$ $\tilde{\mathbf{z}}_7$ $\tilde{\mathbf{z}}_8$
s_1 s_2 s_3 s_4
t_1 t_2 t_3 t_4 t_5
s_1 s_2 s_1 s_2
t_1 t_2 t_3 t_1 t_2

Figure 1: A schematic representation of the proposed prediction interpretability method.

by sampling repeatedly from this distribution, and then mapping these back to the original space using the decoder. The training regime for the VAE ensures approximately that a small perturbation of the hidden representation maintains similar semantic content while introducing small changes in the decoded surface form. We emphasize that the approach would likely fail with an ordinary autoencoder where small changes in the latent representation can result in large changes in the decoded output. In practice, we ensure diversity of perturbations by scaling the variance term $\boldsymbol{\sigma}$ and sampling points $\tilde{\mathbf{z}}$ and different resolutions. We provide further details of this procedure in the supplement. Naturally, we can train this perturbation model in advance on (unlabeled) data from the input domain $\mathcal{X}$, and then use it as a subroutine in our method. After this process is complete, we have N pairs of perturbed input-output pairs: $\{(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i)\}_{i=1}^N$ which exercise the mapping F around semantically similar inputs.

4.2 Causal model

The second step consists of using the perturbed input-output pairs $\{(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i)\}_{i=1}^N$ to infer causal dependencies between the original input and output tokens. A naive approach would consider 2x2 contingency tables representing presence/absence of input/output tokens together with a test statistic for assessing their dependence. Instead, we incorporate *all* input tokens simultaneously to predict the occurrence of a single output token via logistic regression. The quality of these dependency estimators will depend on the frequency with which each input and output token occurs in the perturbations. Thus, we are interested in obtaining uncertainty estimates for these predictions, which can be naturally done with a Bayesian approach to logistic regression. Let $\phi_{\mathbf{x}}(\tilde{\mathbf{x}}) \in \{0,1\}^{|\mathbf{x}|}$ be a binary vector encoding the presence of the original tokens $x_1, \ldots, x_n$ from $\mathbf{x}$ in the perturbed version $\tilde{\mathbf{x}}$. For each target token $y_j \in \mathbf{y}$, we estimate a model:

$$P(y_j \in \tilde{\mathbf{y}} \mid \tilde{\mathbf{x}}) = \sigma(\boldsymbol{\theta}_j^T \phi_{\mathbf{x}}(\tilde{\mathbf{x}})) \qquad (1)$$

where $\sigma(z) = (1 + \exp(-z))^{-1}$. We use a Gaussian approximation for the logarithm of the logistic function together with the prior $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}_0, \mathbf{H}_0^{-1})$ (Murphy, 2012). Since in our case all tokens are guaranteed to occur at least once (we include the original example pair as part of the set), we use $\boldsymbol{\theta}_0 = \alpha\mathbf{1}, \mathbf{H}_0 = \beta\mathbf{I}$, with $\alpha, \beta > 0$. Upon completion of this step, we have dependency coefficients between all original input and output tokens $\{\theta_{ij}\}$, along with their uncertainty estimates.

4.3 Explanation Selection

The last step in our interpretability framework consists of selecting a set explanations for $(\mathbf{x}, \mathbf{y})$. The steps so far yield a dense bipartite graph between the input and output tokens. Unless $|\mathbf{x}|$ and $|\mathbf{y}|$ are small, this graph itself may not be sufficiently interpretable. We are interested in selecting *relevant* components of this dependency graph, i.e., partition the vertex set of $\mathcal{G}$ into disjoint subsets so as to minimize the weight of omitted edges (i.e. the k-cut value of the partition).

Graph partitioning is a well studied NP-complete problem (Garey et al., 1976). The usual setting assumes deterministic edge weights, but in our case we are interested in incorporating the uncertainty of the dependency estimates—resulting from their finite sample estimation—into the partitioning problem. For this, we rely on the approach of Fan et al. (2012) designed for interval estimates of edge weights. At a high level, this is a robust optimization formulation which seeks to minimize worst case cut values, and can be cast as a Mixed Integer Programming (MIP) problem. Specifically, for a bipartite graph $G = (U, V, E)$

Algorithm 1 Structured-output causal rationalizer

```
procedure SOCRAT($\mathbf{x}, \mathbf{y}, F$)
    $(\boldsymbol{\mu}, \boldsymbol{\sigma}) \leftarrow$ ENCODE($\mathbf{x}$)            } Perturbation
    for $i = 1$ to $N$ do                                  } Model.
        $\tilde{\mathbf{z}}_i \leftarrow$ SAMPLE($\boldsymbol{\mu}, \boldsymbol{\sigma}$)
        $\tilde{\mathbf{x}}_i \leftarrow$ DECODE($\tilde{\mathbf{z}}_i$)
        $\tilde{\mathbf{y}}_i \leftarrow F(\tilde{\mathbf{x}}_i)$
    end for
    $G \leftarrow$ CAUSAL($\mathbf{x}, \mathbf{y}, \{\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i\}_{i=1}^N$)
    $E_{x \mapsto y} \leftarrow$ BIPARTITION($G$)
    $E_{x \mapsto y} \leftarrow$ SORT($E_{x \mapsto y}$)          ▷ By cut capacity
    return $E_{x \mapsto y}$
end procedure
```

with edge weights given as uncertainty intervals $\theta_{ij} \pm \hat{\theta}_{ij}$, the partitioning problem is given by

$$\min_{(x^u_{ik}, x^v_{jk}, y_{ij}) \in Y} \sum_{i=1}^{n} \sum_{j=1}^{m} \theta_{ij} y_{ij} + \max_{\substack{S: S \subseteq V, |S| \leq \Gamma \\ (i_t, j_t) \in V \setminus S}} \sum_{(i,j) \in S} \hat{\theta}_{ij} y_{ij} + (\Gamma - \lfloor \Gamma \rfloor) \hat{\theta}_{i_t, j_t} y_{i_t, j_t} \tag{2}$$

where x^u_{ik}, x^v_{jk} are binary variables indicating subset belonging for elements of U and V respectively, y_{ij} are binary auxiliary variables indicating whether i and j are in different partitions, and Y is a set of constraints that ensure the K-partition is valid. Γ is a parameter in $[0, |V|]$ which adjusts the robustness of the partition (the number of deviations from the mean edge values). See the supplement for further explanation of this objective.

If $|\mathbf{x}|$ and $|\mathbf{y}|$ are small, the number of clusters K will also be small, so we can simply return all the partitions (i.e. the *explanation chunks*) $E^k_{x \to y} := (V^k_x \cup V^k_y)$. However, when K is large, one might wish to entertain only the κ most relevant explanations. The graph partitioning framework provides us with a natural way to score the importance of each chunk. Intuitively, subgraphs that have few high-valued edges connecting them to other parts of the graph (i.e. low *cut-capacity*) can be thought of as *self-contained* explanations, and thus more relevant for interpretability. We can therefore define the importance score an atom as:

$$\text{importance}(E^k_{x \to y}) := - \sum_{(i,j) \in X_k} \theta_{ij} \tag{3}$$

where X_k is the cut-set implied by $E^k_{x \to y}$:

$$X_k = \{(i,j) \in E \mid i \in E^k_{x \to y}, \ j \in V \setminus E^k_{x \to y}\}$$

The full interpretability method is succinctly expressed in Algorithm 1.

5 Experimental Framework

5.1 Training and optimization

For the experiments involving sentence inputs, we train in advance the VAE described in Section 4.1. We use symmetric encoder-decoders consisting of recurrent neural networks with an intermediate variational layer. In our case, however, we use L stacked RNN's on both sides, and a stacked variational layer. Training variational autoencoders for text is notoriously hard. In addition to dropout and KLD annealing (Bowman et al., 2016), we found that slowly scaling the variance sampled from the normal distribution from 0 to 1 made training much more stable.

For the partitioning step we compare the robust formulation described above with two classical approaches to bipartite graph partitioning which do not take uncertainty into account: the coclustering method of Dhillon (2001) and the biclustering method of Kluger et al. (2003). For these two, we use off-the-shelf implementations,[1] while we solve the MIP problem version of (2) with the optimization library `gurobi`.[2]

5.2 Recovering simple mappings

Before using our interpretability framework in real tasks where quantitative evaluation of explanations is challenging, we test it in a simplified setting where the "black-box" is simple and fully known. A reasonable minimum expectation on our method is that it should be able to infer many of these simple dependencies. For this purpose, we use the CMU Dictionary of word pronunciations,[3] which is based on the ARPAbet symbol set and consists of about 130K word-to-phoneme pairs. Phonemes are expressed as tokens of 1 to 3 characters. An example entry in this dictionary is the pair *vowels* ↦ `V AW1 AH0 L Z`. Though the mapping is simple, it is not one-to-one (a group of characters can correspond to a single phoneme) nor deterministic (the same character can map to different phonemes depending on the context). Thus, it provides a reasonable testbed

[1] http://scikit-learn.org/stable/modules/biclustering.html

[2] http://www.gurobi.com/

[3] www.speech.cs.cmu.edu/cgi-bin/cmudict

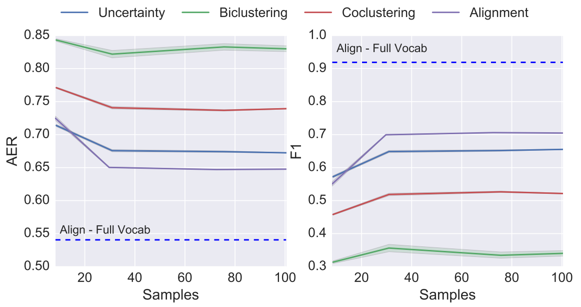


Figure 2: Arpabet test results as a function of number of perturbations used. Shown are mean plus confidence bounds over 5 repetitions. **Left**: Alignment Error Rate, **Right**: F1 over edge prediction.

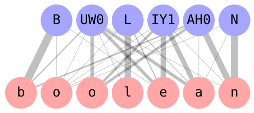
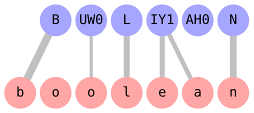
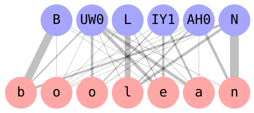
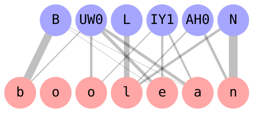
Raw Dependencies		Explanation Graph
	→	
	→	

Table 1: Inferred dependency graphs before (left) and after (right) explanation selection for the prediction: *boolean* $\mapsto$ `B UW0 L IY1 AH0 N`, in independent runs with large (top) and small (bottom) clustering parameter k.

for our method. The setting is as follows: given an input-output pair from the `cmudict` "black-box", we use our method to infer dependencies between characters in the input and phonemes in the output. Since locality in this context is morphological instead of semantic, we produce perturbations selecting n words randomly from the intersection of the `cmudict` vocabulary and the set of words with edit distance at most 2 from the original word.

To evaluate the inferred dependencies, we randomly selected 100 key-value pairs from the dictionary and manually labeled them with character-to-phoneme alignments. Even though our framework is not geared to produce pairwise alignments, it should nevertheless be able to recover them to a certain extent. To provide a point of reference, we compare against a (strong) baseline that is tailored to such a task: a state-of-the-art unsupervised word alignment method based on Monte Carlo inference (Tiedemann and Östling, 2016). The results in Figure 2 show that the version of our method that uses the uncertainty clustering performs remarkably close to the alignment system, with an alignment error rate only ten points above an oracle version of this system that was trained on the *full* arpabet dictionary (dashed line). The raw and partitioned explanations provided by our method for an example input-output pair are shown in Table 1, where the edge widths correspond to the estimated strength of dependency. Throughout this work we display the nodes in the same lexical order of the inputs/outputs to facilitate reading, even if that makes the explanation chunks less visibly discernible. Instead, we sometimes provide an additional (sorted) heatplot of dependency values to show these partitions.

5.3 Machine Translation

In our second set of experiments we evaluate our explanation model in a relevant and popular sequence-to-sequence task: machine translation. As black-boxes, we use three different methods for translating English into German: (i) Azure's Machine Translation system, (ii) a Neural MT model, and (iii) a human (native speaker of German). We provide details on all three systems in the supplement. We translate the same English sentences with all three methods, and explain their predictions using SOCRAT. To be able to generate sentences with similar language and structure as those used to train the two automatic systems, we use the monolingual English side of the WMT14 dataset to train the variational autoencoder described in Section 4.1. For every explanation instance, we sample $S = 100$ perturbations and use the black-boxes to translate them. In all cases, we use the same default SOCRAT configurations, including the robust partitioning method.

In Figure 3, we show the explanations provided by our method for the predictions of each of the three systems on the input sentence *"Students said they looked forward to his class"*. Although the three black-boxes all provided different translations, the explanations show a mostly consistent clustering around the two phrases in the sentence, and in all three cases the cluster with the highest cut value (i.e. the most relevant explanative chunk) is the one containing the subject. Interestingly, the

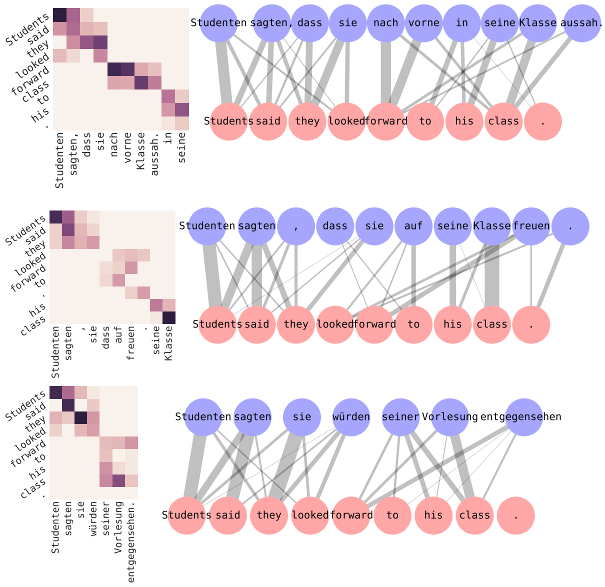


Figure 3: Explanations for the predictions of three Black-Box translators: Azure (top), NMT (middle) and human (bottom). Note that the rows and columns of the heatmaps are permuted to show explanation *chunks* (clusters).

dependency coefficients are overall higher for the human than for the other systems, suggesting more coherence in the translations (potentially because the human translated sentences in context, while the two automatic systems carry over no information from one example to the next).

The NMT system, as opposed to the other two, is not truly a black-box. We can *open the box* to get a glimpse on the true dependencies on the inputs used by the system at prediction time (the attention weights) and compare them to the explanation graph. The attention matrix, however, is dense and not normalized over target tokens, so it is not directly comparable to our dependency scores. Nevertheless, we can partition it with the coclustering method described in Section 4.3 to enforce group structure and make it easier to compare. Figure 4 shows the attention matrix and the explanation for an example sentence of the test set. Their overall cluster structure agrees, though our method shows conservatism with respect to the dependencies of the function words (*to*, *for*). Interestingly, our method is able to figure out that the <unk> token was likely produced by the word "appeals", as shown by the explanation graph.

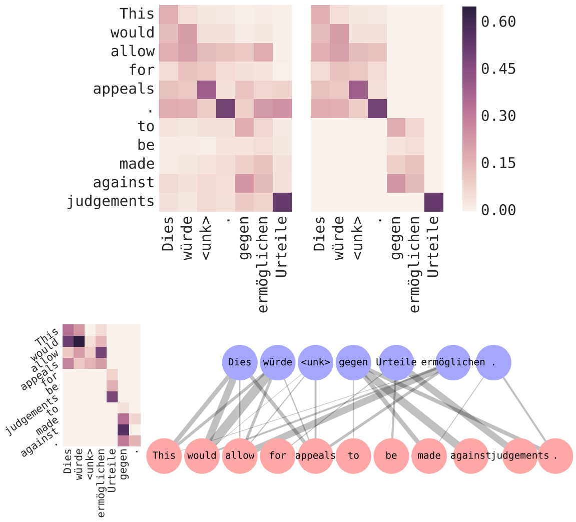


Figure 4: **Top**: Original and clustered attention matrix of the NMT system for a given translation. **Bottom**: Dependency estimates and explanation graph generated by SocRat with with $S = 100$.

It must be emphasized that although we display attention scores in various experiments in this work, we do so only for qualitative evaluation purposes. Our model-agnostic framework can be used on top of models that do not use attention mechanisms or for which this information is hard to extract. Even in cases where it is available, the explanation provided by SocRat might be complementary or even preferable to attention scores because: (a) being normalized on both directions (as opposed to only over source tokens) and partitioned, it is often more interpretable than a dense attention matrix, and (b) it can be retrieved *chunk*-by-*chunk* in decreasing order of relevance, which is especially important when explaining large inputs and/or outputs.

5.4 A (mediocre) dialogue system

So far we have used our method to explain (mostly) correct predictions of meaningful models. But we can use it to gain insights into the workings of flawed black-box systems too. To test this, we train a simple dialogue system on the OpenSubtitle corpus (Tiedemann, 2009), consisting of ∼14M two-step movie dialogues. As before, we use a sequence-to-sequence model with attention, but now we constrain the quality of the model, using only two layers, hidden state dimension of 1000 and no hyper-parameter tuning.

Input	Prediction
What do you mean it doesn't matter?	*I don't know*
Perhaps have we met before?	*I don't think so*
Can I get you two a cocktail?	*No, thanks.*

Table 2: "Good" dialogue system predictions.

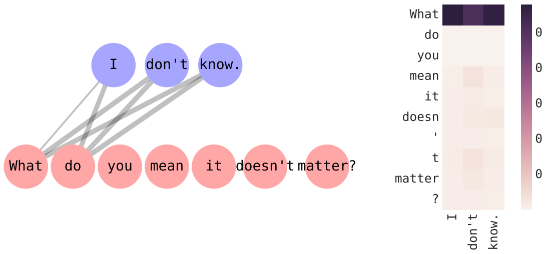


Figure 5: Explanation with $S = 50$ (left) and attention (right) for the first prediction in Table 2.

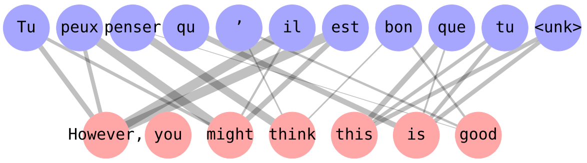


Figure 6: Explanation with $S = 50$ for the prediction of the biased translator.

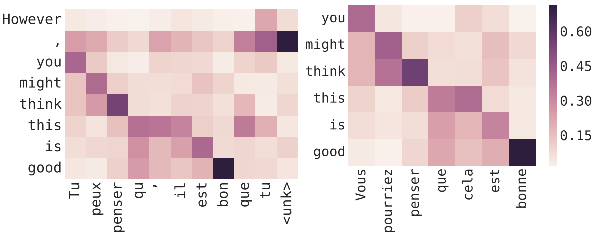


Figure 7: Attention scores on similar sentences by the biased translator.

Although most of the predictions of this model are short and repetitive (*Yes*/*No*/<unk> answers), some of them are seemingly meaningful, and might—if observed in isolation—lead one to believe the system is much better than it actually is. For example, the predictions in Table 2 suggest a complex use of the input to generate the output. To better understand this model, we rationalize its predictions using SocRat. The explanation graph for one such "good" prediction, shown in Figure 5, suggests that there is little influence of anything except the tokens *What* and *you* on the output. Thus, our method suggests that this model is using only partial information of the input and has probably memorized the connection between question words and responses. This is confirmed upon inspecting the model's attention scores for this prediction (same figure, right pane).

5.5 Bias detection in parallel corpora

Natural language processing methods that derive semantics from large corpora have been shown to incorporate biases present in the data, such as archaic stereotypes of male/female occupations (Caliskan et al., 2017) and sexist adjective associations (Bolukbasi et al., 2016). Thus, there is interest in methods that can detect and address those biases. For our last set of experiments, we use our approach to diagnose and explain biased translations of MT systems, first on a simplistic but *verifiable* synthetic setting, where we inject a pre-specified spurious association into an otherwise normal parallel training corpus, and then on an industrial-quality black-box system.

We simulate a biased corpus as follows. Starting from the WMT14 English-French dataset, we identify French sentences written in the informal register (e.g. containing the singular second person *tu*) and prepend their English translation with the word *However*. We obtain about 6K examples this way, after which we add an additional 1M examples that do not contain the word *however* on the English side. The purpose of this is to attempt to induce a (false) association between this adverb and the informal register in French. We then train a sequence-to-sequence model on this polluted data, and we use it to translate adversarially-chosen sentences containing the contaminating token. For example, given the input sentence "*However, you might think this is good*", the method predicts the translation "*Tu peux penser qu ' il est bon que tu <unk>*", which, albeit far from perfect, seems reasonable. However, using SocRat to explain this prediction (cf. Figure 6) raises a red flag: there is an inexplicable strong dependency between the function word *however* and tokens in the output associated with the informal register (*tu*, *peux*), and a lack of dependency between the second *tu* and the source-side pronoun *you*. The model's attention for this prediction (shown in Figure 7, left) confirms that it has picked up this spurious association. Indeed, translating the English sentence now without the prepended adverb

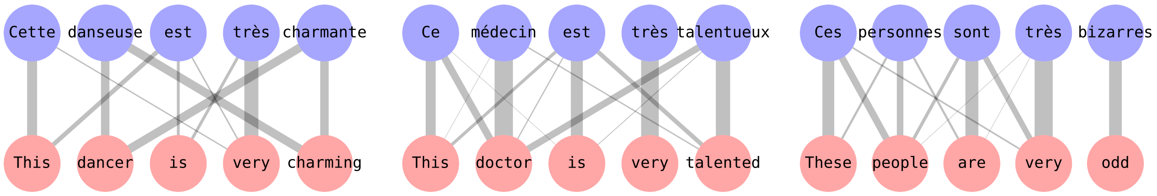


Figure 8: Explanations for biased translations of similar gender-neutral English sentences into French generated with Azure's MT service. The first two require gender declination in the target (French) language, while the third one, in plural, does not. The dependencies in the first two shed light on the cause of the biased selection of gender in the output sentence.

results in a switch to the formal register, as shown in the second plot in Figure 7.

Although somewhat contrived, this synthetic setting works as a litmus test to show that our method is able to detect *known* artificial biases from a model's predictions. We now move to a real setting, where we investigate biases in the predictions of an industrial-quality translation system. We use Azure's MT service to translate into French various simple sentences that lack gender specification in English, but which require gender-declined words in the output. We choose sentences containing occupations and adjectives previously shown to exhibit gender biases in linguistic corpora (Bolukbasi et al., 2016). After observing the choice of gender in the translation, we use SoCRat to explain the output.

In line with previous results, we observe that this translation model exhibits a concerning preference for the masculine grammatical gender in sentences containing occupations such as *doctor, professor* or adjectives such as *smart, talented*, while choosing the feminine gender for *charming, compassionate* subjects who are *dancers* or *nurses*. The explanation graphs for two such examples, shown in Figure 8 (left and center), suggest strong associations between the gender-neutral but stereotype-prone source tokens (*nurse, doctor, charming*) and the gender-carrying target tokens (i.e. the feminine-declined *cette, danseuse, charmante* in the first sentence and the masculine *ce, médecin, talenteux* in the second). While it is not unusual to observe interactions between multiple source and target tokens, the strength of dependence in some of these pairs (*charming*→*danseuse*, *doctor*→*ce*) is unexplained from a grammatical point of view. For comparison, the third example—a sentence in the plural form that does not involve choice of grammatical gender in French—shows comparatively much weaker associations across words in different parts of the sentence.

6 Discussion

Our model-agnostic framework for prediction interpretability with structured data can produce reasonable, coherent, and often insightful explanations. The results on the machine translation task demonstrate how such a method yields a partial view into the inner workings of a black-box system. Lastly, the results of the last two experiments also suggest potential for improving existing systems, by questioning seemingly correct predictions and explaining those that are not.

The method admits several possible modifications. Although we focused on sequence-to-sequence tasks, SocRat generalizes to other settings where inputs and outputs can be expressed as sets of features. An interesting application would be to infer dependencies between textual and image features in image-to-text prediction (e.g. image captioning). Also, we used a VAE-based sampling for object perturbations but other approaches are possible depending on the nature of the domain or data.

Acknowledgments

We thank the anonymous reviewers for their helpful suggestions regarding presentation and additional experiments, and Dr. Chantal Melis for valuable feedback. DAM gratefully acknowledges support from a CONACYT fellowship and the MIT-QCRI collaboration.

References

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLoS One*, 10(7):1–46.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation By Jointly Learning To Align and Translate. *Iclr 2015*, pages 1–15.

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings. *NIPS*, (Nips):4349—-4357.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating Sentences from a Continuous Space. *Iclr*, pages 1–13.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science (80-.).*, 356(6334):183–186.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. Intelligible Models for HealthCare : Predicting Pneumonia Risk and Hospital 30-day Readmission. *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. - KDD '15*, pages 1721–1730.

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2015. Listen, attend and spell. *arXiv Prepr.*, pages 1–16.

Inderjit s. Dhillon. 2001. Co-clustering documents and words using Bipartite spectral graph partitioning. *Proc 7th ACM SIGKDD Conf*, pages 269–274.

Finale Doshi-Velez and Been Kim. 2017. A Roadmap for a Rigorous Science of Interpretability. *ArXiv e-prints*, (Ml):1–12.

Neng Fan, Qipeng P. Zheng, and Panos M. Pardalos. 2012. Robust optimization of graph partitioning involving interval uncertainty. In *Theor. Comput. Sci.*, volume 447, pages 53–61.

M. R. Garey, D. S. Johnson, and L. Stockmeyer. 1976. Some simplified NP-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267.

Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. *Iclr*, (Ml):1–14.

G. Klein, Y. Kim, Y. Deng, J. Senellert, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Yuval Kluger, Ronen Basri, Joseph T. Chang, and Mark Gerstein. 2003. Spectral biclustering of microarray data: Coclustering genes and conditions.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing Neural Predictions. In *EMNLP 2016, Proc. 2016 Conf. Empir. Methods Nat. Lang. Process.*, pages 107–117.

Zachary C Lipton. 2016. The Mythos of Model Interpretability. *ICML Work. Hum. Interpret. Mach. Learn.*, (Whi).

Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective.*

D J Rezende, S Mohamed, and D Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *Proc. 31st ...*, 32:1278–1286.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational Inference with Normalizing Flows. *Proc. 32nd Int. Conf. Mach. Learn.*, 37:1530–1538.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proc. 22Nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, KDD '16, pages 1135–1144, New York, NY, USA. ACM.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. *Proc. Conf. Empir. Methods Nat. Lang. Process.*, (September):379–389.

Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. 2016. Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization. (Nips):1–5.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. 2016. Ladder Variational Autoencoders. *NIPS*, (Nips).

Jörg Tiedemann. 2009. News from OPUS - A Collection of Multilingual Parallel Corpora with Tools and Interfaces. In N. Nicolov, G. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Adv. Nat. Lang. Process.*, pages 237—-248. John Benjamins, Amsterdam/Philadelphia.

Jörg Tiedemann and Robert Östling. 2016. Efficient Word Alignment with Markov Chain Monte Carlo. *Prague Bull. Math. Linguist.*, (106):125–146.

Piecewise Latent Variables for Neural Variational Text Processing

Iulian V. Serban[1*] and **Alexander G. Ororbia II**[2*] and **Joelle Pineau**[3] and **Aaron Courville**[1]
[1] Department of Computer Science and Operations Research, Universite de Montreal
[2]College of Information Sciences & Technology, Penn State University
[3]School of Computer Science, McGill University
iulian [DOT] vlad [DOT] serban [AT] umontreal [DOT] ca
ago109 [AT] psu [DOT] edu
jpineau [AT] cs [DOT] mcgill [DOT] ca
aaron [DOT] courville [AT] umontreal [DOT] ca

Abstract

Advances in neural variational inference have facilitated the learning of powerful directed graphical models with continuous latent variables, such as variational autoencoders. The hope is that such models will learn to represent rich, multi-modal latent factors in real-world data, such as natural language text. However, current models often assume simplistic priors on the latent variables — such as the uni-modal Gaussian distribution — which are incapable of representing complex latent factors efficiently. To overcome this restriction, we propose the simple, but highly flexible, piecewise constant distribution. This distribution has the capacity to represent an exponential number of modes of a latent target distribution, while remaining mathematically tractable. Our results demonstrate that incorporating this new latent distribution into different models yields substantial improvements in natural language processing tasks such as document modeling and natural language generation for dialogue.

1 Introduction

The development of the variational autoencoder framework (Kingma and Welling, 2014; Rezende et al., 2014) has paved the way for learning large-scale, directed latent variable models. This has led to significant progress in a diverse set of machine learning applications, ranging from computer vision (Gregor et al., 2015; Larsen et al., 2016) to natural language processing tasks (Mnih and Gregor, 2014; Miao et al., 2016; Bowman et al., 2015; Serban et al., 2017b). It is hoped that this framework will enable the learning of generative processes of real-world data — including text, audio and images — by disentangling and representing the underlying latent factors in the data. However, latent factors in real-world data are often highly complex. For example, topics in newswire text and responses in conversational dialogue often posses latent factors that follow non-linear (non-smooth), multi-modal distributions (i.e. distributions with multiple local maxima).

Nevertheless, the majority of current models assume a simple prior in the form of a multivariate Gaussian distribution in order to maintain mathematical and computational tractability. This is often a highly restrictive and unrealistic assumption to impose on the structure of the latent variables. First, it imposes a strong uni-modal structure on the latent variable space; latent variable samples from the generating model (prior distribution) all cluster around a single mean. Second, it forces the latent variables to follow a perfectly symmetric distribution with constant kurtosis; this makes it difficult to represent asymmetric or rarely occurring factors. Such constraints on the latent variables increase pressure on the down-stream generative model, which in turn is forced to carefully partition the probability mass for each latent factor throughout its intermediate layers. For complex, multi-modal distributions — such as the distribution over topics in a text corpus, or natural language responses in a dialogue system — the uni-modal Gaussian prior inhibits the model's ability to extract and represent important latent structure in the data. In order to learn more expressive latent variable models, we therefore need more flexible, yet tractable, priors.

In this paper, we introduce a simple, flexible

[*] The first two authors contributed equally.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 422–432
Copenhagen, Denmark, September 7–11, 2017.

prior distribution based on the piecewise constant distribution. We derive an analytical, tractable form that is applicable to the variational autoencoder framework and propose a differentiable parametrization for it. We then evaluate the effectiveness of the distribution when utilized both as a prior and as approximate posterior across variational architectures in two natural language processing tasks: document modeling and natural language generation for dialogue. We show that the piecewise constant distribution is able to capture elements of a target distribution that cannot be captured by simpler priors — such as the uni-modal Gaussian. We demonstrate state-of-the-art results on three document modeling tasks, and show improvements on a dialogue natural language generation. Finally, we illustrate qualitatively how the piecewise constant distribution represents multi-modal latent structure in the data.

2 Related Work

The idea of using an artificial neural network to approximate an inference model dates back to the early work of Hinton and colleagues (Hinton and Zemel, 1994; Hinton et al., 1995; Dayan and Hinton, 1996). Researchers later proposed Markov chain Monte Carlo methods (MCMC) (Neal, 1992), which do not scale well and mix slowly, as well as variational approaches which require a tractable, factored distribution to approximate the true posterior distribution (Jordan et al., 1999). Others have since proposed using feed-forward inference models to initialize the mean-field inference algorithm for training Boltzmann architectures (Salakhutdinov and Larochelle, 2010; Ororbia II et al., 2015). Recently, the variational autoencoder framework (VAE) was proposed by Kingma and Welling (2014) and Rezende et al. (2014), closely related to the method proposed by Mnih and Gregor (2014). This framework allows the joint training of an inference network and a directed generative model, maximizing a variational lower-bound on the data log-likelihood and facilitating exact sampling of the variational posterior. Our work extends this framework.

With respect to document modeling, neural architectures have been shown to outperform well-established topic models such as Latent Dirichlet Allocation (LDA) (Hofmann, 1999; Blei et al., 2003). Researchers have successfully proposed several models involving discrete latent variables (Salakhutdinov and Hinton, 2009; Hinton and Salakhutdinov, 2009; Srivastava et al., 2013; Larochelle and Lauly, 2012; Uria et al., 2014; Lauly et al., 2016; Bornschein and Bengio, 2015; Mnih and Gregor, 2014). The success of such discrete latent variable models — which are able to partition probability mass into separate regions — serves as one of our main motivations for investigating models with more flexible continuous latent variables for document modeling. More recently, Miao et al. (2016) proposed to use continuous latent variables for document modeling.

Researchers have also investigated latent variable models for dialogue modeling and dialogue natural language generation (Bangalore et al., 2008; Crook et al., 2009; Zhai and Williams, 2014). The success of discrete latent variable models in this task also motivates our investigation of more flexible continuous latent variables. Closely related to our proposed approach is the Variational Hierarchical Recurrent Encoder-Decoder (*VHRED*, described below) (Serban et al., 2017b), a neural architecture with latent multivariate Gaussian variables.

Researchers have explored more flexible distributions for the latent variables in VAEs, such as autoregressive distributions, hierarchical probabilistic models and approximations based on MCMC sampling (Rezende et al., 2014; Rezende and Mohamed, 2015; Kingma et al., 2016; Ranganath et al., 2016; Maaløe et al., 2016; Salimans et al., 2015; Burda et al., 2016; Chen et al., 2017; Ruiz et al., 2016). These are all complimentary to our approach; it is possible to combine them with the piecewise constant latent variables. In parallel to our work, multiple research groups have also proposed VAEs with discrete latent variables (Maddison et al., 2017; Jang et al., 2017; Rolfe, 2017; Johnson et al., 2016). This is a promising line of research, however these approaches often require approximations which may be inaccurate when applied to larger scale tasks, such as document modeling or natural language generation. Finally, discrete latent variables may be inappropriate for certain natural language processing tasks.

3 Neural Variational Models

We start by introducing the neural variational learning framework. We focus on modeling discrete output variables (e.g. words) in the context of natural language processing applications. However, the framework can easily be adapted to handle continuous output variables.

3.1 Neural Variational Learning

Let $w_1, \ldots, w_N$ be a sequence of N tokens (words) conditioned on a continuous latent variable z. Further, let c be an additional observed variable which conditions both z and $w_1, \ldots, w_N$. Then, the distribution over words is:

$$P_\theta(w_1, \ldots, w_N|c) = \int \prod_{n=1}^{N} P_\theta(w_n|w_{<n}, z, c) P_\theta(z|c) dz,$$

where θ are the model parameters. The model first generates the higher-level, continuous latent variable z conditioned on c. Given z and c, it then generates the word sequence $w_1, \ldots, w_N$. For unsupervised modeling of documents, the c is excluded and the words are assumed to be independent of each other, when conditioned on z:

$$P_\theta(w_1, \ldots, w_N) = \int \prod_{n=1}^{N} P_\theta(w_n|z) P_\theta(z) dz.$$

Model parameters can be learned using the variational lower-bound (Kingma and Welling, 2014):

$$\begin{aligned} &\log P_\theta(w_1, \ldots, w_N|c) \\ &\quad \geq \quad \mathrm{E}_{z \sim Q_\psi(z|w_1,\ldots,w_N,c)}[\log P_\theta(w_n|w_{<n}, z, c)] \\ &\qquad - \mathrm{KL}\left[Q_\psi(z|w_1, \ldots, w_N, c) || P_\theta(z|c)\right], \qquad (1) \end{aligned}$$

where we note that $Q_\psi(z|w_1, \ldots, w_N, c)$ is the approximation to the intractable, true posterior $P_\theta(z|w_1, \ldots, w_N, c)$. Q is called the *encoder*, or sometimes the *recognition model* or *inference model*, and it is parametrized by ψ. The distribution $P_\theta(z|c)$ is the prior model for z, where the only available information is c. The VAE framework further employs the re-parametrization trick, which allows one to move the derivative of the lower-bound inside the expectation. To accomplish this, z is parametrized as a transformation of a fixed, parameter-free random distribution $z = f_\theta(\epsilon)$, where ϵ is drawn from a random distribution. Here, f is a transformation of ϵ, parametrized by θ, such that $f_\theta(\epsilon) \sim P_\theta(z|c)$. For example, ϵ might be drawn from a standard Gaussian distribution and f might be defined as $f_\theta(\epsilon) = \mu + \sigma\epsilon$, where μ and σ are in the parameter set θ. In this case, z is able to represent any Gaussian with mean μ and variance σ^2.

Model parameters are learned by maximizing the variational lower-bound in eq. (1) using gradient descent, where the expectation is computed using samples from the approximate posterior.

The majority of work on VAEs propose to parametrize z as multivariate Gaussian distributions. However, this unrealistic assumption may critically hurt the expressiveness of the latent variable model. See Appendix A for a detailed discussion. This motivates the proposed piecewise constant latent variable distribution.

3.2 Piecewise Constant Distribution

We propose to learn latent variables by parametrizing z using a piecewise constant probability density function (PDF). This should allow z to represent complex aspects of the data distribution in latent variable space, such as non-smooth regions of probability mass and multiple modes.

Let $n \in \mathbb{N}$ be the number of piecewise constant components. We assume z is drawn from PDF:

$$P(z) = \frac{1}{K} \sum_{i=1}^{n} 1_{\left(\frac{i-1}{n} \leq z \leq \frac{i}{n}\right)} a_i, \qquad (2)$$

where $1_{(x)}$ is the indicator function, which is one when x is true and otherwise zero. The distribution parameters are $a_i > 0$, for $i = 1, \ldots, n$. The normalization constant is:

$$K = \sum_{i=1}^{n} K_i, \text{ where } K_0 = 0, K_i = \frac{a_i}{n}, \text{ for } i = 1, \ldots, n.$$

It is straightforward to show that a piecewise constant distribution with more than $n > 2$ pieces is capable of representing a bi-modal distribution. When $n > 2$, a vector z of piecewise constant variables can represent a probability density with $2^{|z|}$ modes. Figure 1 illustrates how these variables help model complex, multi-modal distributions.

In order to compute the variational bound, we need to draw samples from the piecewise constant distribution using its inverse cumulative distribution function (CDF). Further, we need to compute the KL divergence between the prior and posterior. The inverse CDF and KL divergence quantities are both derived in Appendix B. During training we must compute derivatives of the variational bound in eq. (1). These expressions involve derivatives of indicator functions, which have derivatives zero everywhere except for the changing points where the derivative is undefined. However, the probability of sampling the value exactly at its changing

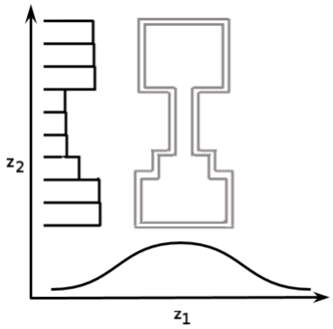


Figure 1: Joint density plot of a pair of Gaussian and piecewise constant variables. The horizontal axis corresponds to z_1, which is a univariate Gaussian variable. The vertical axis corresponds to z_2, which is a piecewise constant variable.

point is effectively zero. Thus, we fix these derivatives to zero. Similar approximations are used in training networks with rectified linear units.

4 Latent Variable Parametrizations

In this section, we develop the parametrization of both the Gaussian variable and our proposed piecewise constant latent variable.

Let x be the current output sequence, which the model must generate (e.g. $w_1, \ldots, w_N$). Let c be the observed conditioning information. If the task contains additional conditioning information this will be embedded by c. For example, for dialogue natural language generation c represents an embedding of the dialogue history, while for document modeling $c = \emptyset$.

4.1 Gaussian Parametrization

Let μ^{prior} and $\sigma^{2,\text{prior}}$ be the prior mean and variance, and let μ^{post} and $\sigma^{2,\text{post}}$ be the approximate posterior mean and variance. For Gaussian latent variables, the prior distribution mean and variances are encoded using linear transformations of a hidden state. In particular, the prior distribution covariance is encoded as a diagonal covariance matrix using a softplus function:

$$\begin{aligned}\mu^{\text{prior}} &= H_\mu^{\text{prior}}\text{Enc}(c) + b_\mu^{\text{prior}},\\ \sigma^{2,\text{prior}} &= \text{diag}(\log(1 + \exp(H_\sigma^{\text{prior}}\text{Enc}(c) + b_\sigma^{\text{prior}}))),\end{aligned}$$

where $\text{Enc}(c)$ is an embedding of the conditioning information c (e.g. for dialogue natural language generation this might, for example, be produced by an LSTM encoder applied to the dialogue history), which is shared across all latent variable dimensions. The matrices $H_\mu^{\text{prior}}, H_\sigma^{\text{prior}}$ and vectors $b_\mu^{\text{prior}}, b_\sigma^{\text{prior}}$ are learnable parameters. For the posterior distribution, previous work has shown it is better to parametrize the posterior distribution as a linear interpolation of the prior distribution mean and variance and a new estimate of the mean and variance based on the observation x (Fraccaro et al., 2016). The interpolation is controlled by a gating mechanism, allowing the model to turn on/off latent dimensions:

$$\begin{aligned}\mu^{\text{post}} =& (1 - \alpha_\mu)\mu^{\text{prior}} + \alpha_\mu \left(H_\mu^{\text{post}}\text{Enc}(c, x) + b_\mu^{\text{post}}\right),\\ \sigma^{2,\text{post}} =& (1 - \alpha_\sigma)\sigma^{2,\text{prior}}\\ &+ \alpha_\sigma \text{diag}(\log(1 + \exp(H_\sigma^{\text{post}}\text{Enc}(c, x) + b_\sigma^{\text{post}}))),\end{aligned}$$

where $\text{Enc}(c, x)$ is an embedding of both c and x. The matrices $H_\mu^{\text{post}}, H_\sigma^{\text{post}}$ and the vectors $b_\mu^{\text{post}}, b_\sigma^{\text{post}}, \alpha_\mu, \alpha_\sigma$ are parameters to be learned. The interpolation mechanism is controlled by α_μ and α_σ, which are initialized to zero (i.e. initialized such that the posterior is equal to the prior).

4.2 Piecewise Constant Parametrization

We parametrize the piecewise prior parameters using an exponential function applied to a linear transformation of the conditioning information:

$$a_i^{\text{prior}} = \exp(H_{a,i}^{\text{prior}}\text{Enc}(c) + b_{a,i}^{\text{prior}}), \quad i = 1, \ldots, n,$$

where matrix H_a^{prior} and vector b_a^{prior} are learnable. As before, we define the posterior parameters as a function of both c and x:

$$a_i^{\text{post}} = \exp(H_{a,i}^{\text{post}}\text{Enc}(c, x) + b_{a,i}^{\text{post}}), \quad i = 1, \ldots, n,$$

where H_a^{post} and b_a^{post} are parameters.

5 Variational Text Modeling

We now introduce two classes of VAEs. The models are extended by incorporating the Gaussian and piecewise latent variable parametrizations.

5.1 Document Model

The neural variational document model (*NVDM*) model has previously been proposed for document modeling (Mnih and Gregor, 2014; Miao et al., 2016), where the latent variables are Gaussian. Since the original *NVDM* uses Gaussian latent variables, we will refer to it as *G-NVDM*. We propose two novel models building on *G-NVDM*. The first model we propose uses piecewise constant latent variables instead of Gaussian latent variables.

We refer to this model as *P-NVDM*. The second model we propose uses a combination of Gaussian and piecewise constant latent variables. The models sample the Gaussian and piecewise constant latent variables independently and then concatenates them together into one vector. We refer to this model as *H-NVDM*.

Let V be the vocabulary of document words. Let W represent a document matrix, where row w_i is the 1-of-$|V|$ binary encoding of the i'th word in the document. Each model has an encoder component $Enc(W)$, which compresses a document vector into a continuous distributed representation upon which the approximate posterior is built. For document modeling, word order information is not taken into account and no additional conditioning information is available. Therefore, each model uses a bag-of-words encoder, defined as a multi-layer perceptron (MLP) $Enc(c = \emptyset, x) = Enc(x)$. Based on preliminary experiments, we choose the encoder to be a two-layered MLP with parametrized rectified linear activation functions (we omit these parameters for simplicity). For the approximate posterior, each model has the parameter matrix W_a^{post} and vector b_a^{post} for the piecewise latent variables, and the parameter matrices $W_\mu^{\text{post}}, W_\sigma^{\text{post}}$ and vectors $b_\mu^{\text{post}}, b_\sigma^{\text{post}}$ for the Gaussian means and variances. For the prior, each model has parameter vector b_a^{prior} for the piecewise latent variables, and vectors $b_\mu^{\text{prior}}, b_\sigma^{\text{prior}}$ for the Gaussian means and variances. We initialize the bias parameters to zero in order to start with centered Gaussian and piecewise constant priors. The encoder will adapt these priors as learning progresses, using the gating mechanism to turn on/off latent dimensions.

Let z be the vector of latent variables sampled according to the approximate posterior distribution. Given z, the decoder $Dec(w, z)$ outputs a distribution over words in the document:

$$Dec(w, z) = \frac{\exp\left(-w^{\mathrm{T}} R z + b_w\right)}{\sum_{w'} \exp\left(-w^{\mathrm{T}} R z + b_{w'}\right)},$$

where R is a parameter matrix and b is a parameter vector corresponding to the bias for each word to be learned. This output probability distribution is combined with the KL divergences to compute the lower-bound in eq. (1). See Appendix C.

Our baseline model *G-NVDM* is an improvement over the original *NVDM* proposed by Mnih and Gregor (2014) and Miao et al. (2016). We learn the prior mean and variance, while these were fixed to a standard Gaussian in previous work. This increases the flexibility of the model and makes optimization easier. In addition, we use a gating mechanism for the approximate posterior of the Gaussian variables. This gating mechanism allows the model to turn off latent variable (i.e. fix the approximate posterior to equal the prior for specific latent variables) when computing the final posterior parameters. Furthermore, Miao et al. (2016) alternated between optimizing the approximate posterior parameters and the generative model parameters, while we optimize all parameters simultaneously.

5.2 Dialogue Model

The variational hierarchical recurrent encoder-decoder (*VHRED*) model has previously been proposed for dialogue modeling and natural language generation (Serban et al., 2017b, 2016a). The model decomposes dialogues using a two-level hierarchy: sequences of utterances (e.g. sentences), and sub-sequences of tokens (e.g. words). Let $\mathbf{w}_n$ be the n'th utterance in a dialogue with N utterances. Let $w_{n,m}$ be the m'th word in the n'th utterance from vocabulary V given as a 1-of-$|V|$ binary encoding. Let M_n be the number of words in the n'th utterance. For each utterance $n = 1, \dots, N$, the model generates a latent variable z_n. Conditioned on this latent variable, the model then generates the next utterance:

$$P_\theta(\mathbf{w}_1, z_1, \dots, \mathbf{w}_N, z_N) = \prod_{n=1}^{N} P_\theta(z_n | \mathbf{w}_{<n}) \times \prod_{m=1}^{M_n} P_\theta(w_{n,m} | w_{n,<m}, \mathbf{w}_{<n}, z_n),$$

where θ are the model parameters. *VHRED* consists of three RNN modules: an *encoder* RNN, a *context* RNN and a *decoder* RNN. The *encoder* RNN computes an embedding for each utterance. This embedding is fed into the *context* RNN, which computes a hidden state summarizing the dialogue context before utterance n: h_{n-1}^{con}. This state represents the additional conditioning information, which is used to compute the prior distribution over z_n:

$$P_\theta(z_n \mid \mathbf{w}_{<n}) = f_\theta^{\text{prior}}(z_n; h_{n-1}^{con}),$$

where f^{prior} is a PDF parametrized by both θ and h_{n-1}^{con}. A sample is drawn from this distribution: $z_n \sim P_\theta(z_n | \mathbf{w}_{<n})$. This sample is given as input

to the *decoder* RNN, which then computes the output probabilities of the words in the next utterance. The model is trained by maximizing the variational lower-bound, which factorizes into independent terms for each sub-sequence (utterance):

$$\begin{aligned}
&\log P_\theta(\mathbf{w}_1, \ldots, \mathbf{w}_N) \\
&\geq \sum_{n=1}^{N} - \mathrm{KL}\left[Q_\psi(z_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_n) || P_\theta(z_n \mid \mathbf{w}_{<n})\right] \\
&\qquad + \mathbb{E}_{Q_\psi(z_n \mid \mathbf{w}_1, \ldots, \mathbf{w}_n)}\left[\log P_\theta(\mathbf{w}_n \mid z_n, \mathbf{w}_{<n})\right],
\end{aligned}$$

where distribution Q_ψ is the approximate posterior distribution with parameters ψ, computed similarly as the prior distribution but further conditioned on the *encoder* RNN hidden state of the next utterance.

The original *VHRED* model (Serban et al., 2017b) used Gaussian latent variables. We refer to this model as *G-VHRED*. The first model we propose uses piecewise constant latent variables instead of Gaussian latent variables. We refer to this model as *P-VHRED*. The second model we propose takes advantage of the representation power of both Gaussian and piecewise constant latent variables. This model samples both a Gaussian latent variable z_n^{gaussian} and a piecewise latent variable $z_n^{\text{piecewise}}$ independently conditioned on the *context* RNN hidden state:

$$\begin{aligned}
P_\theta(z_n^{\text{gaussian}} \mid \mathbf{w}_{<n}) &= f_\theta^{\text{prior, gaussian}}(z_n^{\text{gaussian}}; h_{n-1}^{con}), \\
P_\theta(z_n^{\text{piecewise}} \mid \mathbf{w}_{<n}) &= f_\theta^{\text{prior, piecewise}}(z_n^{\text{piecewise}}; h_{n-1}^{con}),
\end{aligned}$$

where $f^{\text{prior, gaussian}}$ and $f^{\text{prior, piecewise}}$ are PDFs parametrized by independent subsets of parameters θ. We refer to this model as *H-VHRED*.

6 Experiments

We evaluate the proposed models on two types of natural language processing tasks: document modeling and dialogue natural language generation. All models are trained with back-propagation using the variational lower-bound on the log-likelihood or the exact log-likelihood. We use the first-order gradient descent optimizer Adam (Kingma and Ba, 2015) with gradient clipping (Pascanu et al., 2012)[1]

Model	**20-NG**	**RCV1**	**CADE**
LDA	1058	−−	−−
docNADE	896	−−	−−
NVDM	836	−−	−−
G-NVDM	651	905	339
H-NVDM-3	607	865	**258**
H-NVDM-5	**566**	**833**	294

Table 1: Test perplexities on three document modeling tasks: 20-NewGroup (20-NG), Reuters corpus (RCV1) and CADE12 (CADE). Perplexities were calculated using 10 samples to estimate the variational lower-bound. The *H-NVDM* models perform best across all three datasets.

6.1 Document Modeling

Tasks We use three different datasets for document modeling experiments. First, we use the 20 News-Groups (20-NG) dataset (Hinton and Salakhutdinov, 2009). Second, we use the Reuters corpus (RCV1-V2), using a version that contained a selected 5,000 term vocabulary. As in previous work (Hinton and Salakhutdinov, 2009; Larochelle and Lauly, 2012), we transform the original word frequencies using the equation $\log(1 + \text{TF})$, where TF is the original word frequency. Third, to test our document models on text from a non-English language, we use the Brazilian Portuguese CADE12 dataset (Cardoso-Cachopo, 2007). For all datasets, we track the validation bound on a subset of 100 vectors randomly drawn from each training corpus.

Training All models were trained using minibatches with 100 examples each. A learning rate of 0.002 was used. Model selection and early stopping were conducted using the validation lower-bound, estimated using five stochastic samples per validation example. Inference networks used 100 units in each hidden layer for 20-NG and CADE, and 100 for RCV1. We experimented with both 50 and 100 latent random variables for each class of models, and found that 50 latent variables performed best on the validation set. For *H-NVDM* we vary the number of components used in the PDF, investigating the effect that 3 and 5 pieces had on the final quality of the model. The number

[1] Code and scripts are available at `https://github.com/ago109/piecewise-nvdm-emnlp-2017` and `https://github.com/julianser/hred-latent-piecewise`.

G-NVDM	H-NVDM-3	H-NVDM-5
environment	project	science
project	gov	built
flight	major	high
lab	based	technology
mission	earth	world
launch	include	form
field	science	scale
working	nasa	sun
build	systems	special
gov	technical	area

Table 2: Word query similarity test on 20 News-Groups: for the query 'space", we retrieve the top 10 nearest words in word embedding space based on Euclidean distance. *H-NVDM-5* associates multiple meanings to the query, while *G-NVDM* only associates the most frequent meaning.

of hidden units was chosen via preliminary experimentation with smaller models. On 20-NG, we use the same set-up as (Hinton and Salakhutdinov, 2009) and therefore report the perplexities of a topic model (*LDA*, (Hinton and Salakhutdinov, 2009)), the document neural auto-regressive estimator (*docNADE*, (Larochelle and Lauly, 2012)), and a neural variational document model with a fixed standard Gaussian prior (*NVDM*, lowest reported perplexity, (Miao et al., 2016)).

Results In Table 1, we report the test document perplexity: $\exp(-\frac{1}{D}\sum_n \frac{1}{L_n} \log P_\theta(x_n)$. We use the variational lower-bound as an approximation based on 10 samples, as was done in (Mnih and Gregor, 2014). First, we note that the best baseline model (i.e. the *NVDM*) is more competitive when both the prior and posterior models are learnt together (i.e. the *G-NVDM*), as opposed to the fixed prior of (Miao et al., 2016). Next, we observe that integrating our proposed piecewise variables yields even better results in our document modeling experiments, substantially improving over the baselines. More importantly, in the 20-NG and Reuters datasets, increasing the number of pieces from 3 to 5 further reduces perplexity. Thus, we have achieved a new state-of-the-art perplexity on 20 News-Groups task and — to the best of our knowledge – better perplexities on the CADE12 and RCV1 tasks compared to using a state-of-the-art model like the *G-NVDM*. We also evaluated the converged models using an non-parametric inference procedure, where a separate approximate posterior is learned for each test example in order to tighten the variational lower-bound. *H-NVDM* also performed best in this evaluation across all three datasets, which confirms that the performance improvement is due to the piecewise components. See appendix for details.

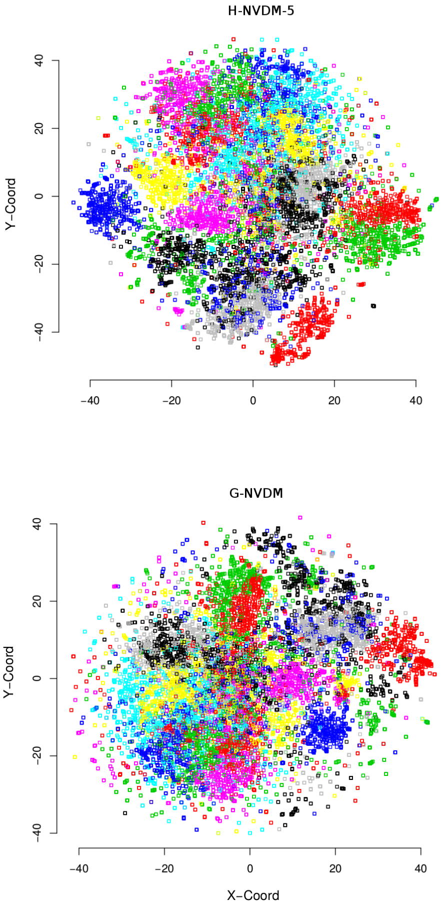


Figure 2: Latent variable approximate posterior means t-SNE visualization on 20-NG for *G-NVDM* and *H-NVDM-5*. Colors correspond to the topic labels assigned to each document.

In Table 2, we examine the top ten highest ranked words given the query term "space", using the decoder parameter matrix. The piecewise variables appear to have a significant effect on what is uncovered by the model.In the case of "space", the hybrid with 5 pieces seems to value two senses of the word–one related to "outer space" (e.g., "sun", "world", etc.) and another related to the dimensions of depth, height, and width within which things may exist and move (e.g., "area", "form", "scale", etc.). On the other hand, *G-NVDM* appears to only capture the "outer space" sense of

Model	Activity	Entity
HRED	4.77	2.43
G-VHRED	**9.24**	2.49
P-VHRED	5	2.49
H-VHRED	8.41	**3.72**

Table 3: Ubuntu evaluation using F1 metrics w.r.t. activities and entities. *G-VHRED*, *P-VHRED* and *H-VHRED* all outperform the baseline *HRED*. *G-VHRED* performs best w.r.t. activities and *H-VHRED* performs best w.r.t. entities.

the word. More examples are in the appendix.

Finally, we visualized the means of the approximate posterior latent variables on 20-NG through a t-SNE projection. As shown in Figure 2, both *G-NVDM* and *H-NVDM-5* learn representations which disentangle the topic clusters on 20-NG. However, *G-NVDM* appears to have more dispersed clusters and more outliers (i.e. data points in the periphery) compared to *H-NVDM-5*. Although it is difficult to draw conclusions based on these plots, these findings could potentially be explained by the Gaussian latent variables fitting the latent factors poorly.

6.2 Dialogue Modeling

Task We evaluate *VHRED* on a natural language generation task, where the goal is to generate responses in a dialogue. This is a difficult problem, which has been extensively studied in the recent literature (Ritter et al., 2011; Lowe et al., 2015; Sordoni et al., 2015; Li et al., 2016; Serban et al., 2016a,b). Dialogue response generation has recently gained a significant amount of attention from industry, with high-profile projects such as Google SmartReply (Kannan et al., 2016) and Microsoft Xiaoice (Markoff and Mozur, 2015). Even more recently, Amazon has announced the Alexa Prize Challenge for the research community with the goal of developing a natural and engaging chatbot system (Farber, 2016).

We evaluate on the technical support response generation task for the Ubuntu operating system. We use the well-known Ubuntu Dialogue Corpus (Lowe et al., 2015, 2017), which consists of about 1/2 million natural language dialogues extracted from the #Ubuntu Internet Relayed Chat (IRC) channel. The technical problems discussed span a wide range of software-related and hardware-related issues. Given a dialogue history — such as a conversation between a user and a technical support assistant — the model must generate the next appropriate response in the dialogue. For example, when it is the turn of the technical support assistant, the model must generate an appropriate response helping the user resolve their problem.

We evaluate the models using the activity- and entity-based metrics designed specifically for the Ubuntu domain (Serban et al., 2017a). These metrics compare the *activities* and *entities* in the model generated responses with those of the reference responses; activities are verbs referring to high-level actions (e.g. *download*, *install*, *unzip*) and entities are nouns referring to technical objects (e.g. *Firefox*, *GNOME*). The more activities and entities a model response overlaps with the reference response (e.g. expert response) the more likely the response will lead to a solution.

Training The models were trained to maximize the log-likelihood of training examples using a learning rate of 0.0002 and mini-batches of size 80. We use a variant of truncated backpropagation. We terminate the training procedure for each model using early stopping, estimated using one stochastic sample per validation example. We evaluate the models by generating dialogue responses: conditioned on a dialogue context, we fix the model latent variables to their median values and then generate the response using a beam search with size 5. We select model hyperparameters based on the validation set using the F1 activity metric, as described earlier.

It is often difficult to train generative models for language with stochastic latent variables (Bowman et al., 2015; Serban et al., 2017b). For the latent variable models, we therefore experiment with reweighing the KL divergence terms in the variational lower-bound with values 0.25, 0.50, 0.75 and 1.0. In addition to this, we linearly increase the KL divergence weights starting from zero to their final value over the first 75000 training batches. Finally, we weaken the *decoder* RNN by randomly replacing words inputted to the decoder RNN with the unknown token with 25% probability. These steps are important for effectively training the models, and the latter two have been used in previous work by Bowman et al. (2015) and Serban et al. (2017b).

HRED (Baseline): We compare to the *HRED* model (Serban et al., 2016a): a sequence-to-sequence model, shown to outperform other established models on this task, such as the LSTM RNN language model (Serban et al., 2017a). The *HRED* model's *encoder* RNN uses a bidirectional GRU RNN encoder, where the forward and backward RNNs each have 1000 hidden units. The context RNN is a GRU encoder with 1000 hidden units, and the decoder RNN is an LSTM decoder with 2000 hidden units.[2] The encoder and context RNNs both use layer normalization (Ba et al., 2016).[3] We also experiment with an additional rectified linear layer applied on the inputs to the decoder RNN. As with other hyper-parameters, we choose whether to include this additional layer based on the validation set performance. *HRED*, as well as all other models, use a word embedding dimensionality of size 400.

G-HRED: We compare to *G-VHRED*, which is *VHRED* with Gaussian latent variables (Serban et al., 2017b). *G-VHRED* uses the same hyper-parameters for the encoder, context and decoder RNNs as the HRED model. The model has 100 Gaussian latent variables per utterance.

P-HRED: The first model we propose is *P-VHRED*, which is *VHRED* model with piecewise constant latent variables. We use $n = 3$ number of pieces for each latent variable. *P-VHRED* also uses the same hyper parameters for the encoder, context and decoder RNNs as the *HRED* model. Similar to *G-VHRED*, *P-VHRED* has 100 piecewise constant latent variables per utterance.

H-HRED: The second model we propose is *H-VHRED*, which has 100 piecewise constant (with $n = 3$ pieces per variable) and 100 Gaussian latent variables per utterance. *H-VHRED* also uses the same hyper-parameters for the encoder, context and decoder RNNs as *HRED*.

Results: The results are given in Table 3. All latent variable models outperform *HRED* w.r.t. both activities and entities. This strongly suggests that the high-level concepts represented by the latent variables help generate meaningful, goal-directed responses. Furthermore, each type of latent variable appears to help with a different aspects of the generation task. *G-VHRED* performs best w.r.t. activities (e.g. *download*, *install* and so on), which occur frequently in the dataset. This suggests that the Gaussian latent variables learn useful latent representations for frequent actions. On the other hand, *H-VHRED* performs best w.r.t. entities (e.g. *Firefox*, *GNOME*), which are often much rarer and mutually exclusive in the dataset. This suggests that the combination of Gaussian and piecewise latent variables help learn useful representations for entities, which could not be learned by Gaussian latent variables alone. We further conducted a qualitative analysis of the model responses, which supports these conclusions. See Appendix G.[4]

7 Conclusions

In this paper, we have sought to learn rich and flexible multi-modal representations of latent variables for complex natural language processing tasks. We have proposed the piecewise constant distribution for the variational autoencoder framework. We have derived closed-form expressions for the necessary quantities required for in the autoencoder framework, and proposed an efficient, differentiable implementation of it. We have incorporated the proposed piecewise constant distribution into two model classes — *NVDM* and *VHRED* — and evaluated the proposed models on document modeling and dialogue modeling tasks. We have achieved state-of-the-art results on three document modeling tasks, and have demonstrated substantial improvements on a dialogue modeling task. Overall, the results highlight the benefits of incorporating the flexible, multi-modal piecewise constant distribution into variational autoencoders. Future work should explore other natural language processing tasks, where the data is likely to arise from complex, multi-modal latent factors.

Acknowledgments

The authors acknowledge NSERC, Canada Research Chairs, CIFAR, IBM Research, Nuance Foundation and Microsoft Maluuba for funding. Alexander G. Ororbia II was funded by a NACME-Sloan scholarship. The authors thank Hugo Larochelle for sharing the NewsGroup 20 dataset. The authors thank Laurent Charlin, Sungjin Ahn, and Ryan Lowe for constructive feedback. This research was enabled in part by support provided by Calcul Qubec (`www.calculquebec.ca`) and Compute Canada (`www.computecanada.ca`).

[2] Since training lasted between 1-3 weeks for each model, we had to fix the number of hidden units during preliminary experiments on the training and validation datasets.

[3] We did not apply layer normalization to the decoder RNN, because several of our colleagues have found that this may hurt the performance of generative language models.

[4] Results on a Twitter dataset are given in the appendix.

References

J. L. Ba, J. R. Kiros, and G. E. Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

S. Bangalore, G. Di Fabbrizio, and A. Stent. 2008. Learning the structure of task-driven human–human dialogs. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(7):1249–1259.

D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent dirichlet allocation. *JAIR*, 3:993–1022.

J. Bornschein and Y. Bengio. 2015. Reweighted wake-sleep. In *ICLR*.

S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. 2015. Generating sentences from a continuous space. In *Conference on Computational Natural Language Learning*.

Y. Burda, R. Grosse, and R. Salakhutdinov. 2016. Importance weighted autoencoders. *ICLR*.

A. Cardoso-Cachopo. 2007. Improving Methods for Single-label Text Categorization. PdD Thesis, Instituto Superior Tecnico, Universidade Tecnica de Lisboa.

X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. 2017. Variational lossy autoencoder. In *ICLR*.

N. Crook, R. Granell, and S. Pulman. 2009. Unsupervised classification of dialogue acts using a dirichlet process mixture model. In *Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 341–348.

P. Dayan and G. E. Hinton. 1996. Varieties of helmholtz machine. *Neural Networks*, 9(8):1385–1403.

L. Devroye. 1986. Sample-based non-uniform random variate generation. In *Proceedings of the 18th conference on Winter simulation*, pages 260–265. ACM.

M. Farber. 2016. Amazon's 'Alexa Prize' Will Give College Students Up To $2.5M To Create A Socialbot. *Fortune*.

M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther. 2016. Sequential neural models with stochastic layers. In *NIPS*, pages 2199–2207.

K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. 2015. DRAW: A recurrent neural network for image generation. In *ICLR*.

G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. 1995. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161.

G. E. Hinton and R. Salakhutdinov. 2009. Replicated softmax: an undirected topic model. In *NIPS*, pages 1607–1614.

G. E. Hinton and R. S. Zemel. 1994. Autoencoders, minimum description length and helmholtz free energy. In *NIPS*, pages 3–10. NIPS.

T. Hofmann. 1999. Probabilistic latent semantic indexing. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57. ACM.

E. Jang, S. Gu, and B. Poole. 2017. Categorical reparameterization with gumbel-softmax. In *ICLR*.

M. Johnson, D. K. Duvenaud, A. Wiltschko, R. P. Adams, and S. R. Datta. 2016. Composing graphical models with neural networks for structured representations and fast inference. In *NIPS*, pages 2946–2954.

M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

A. Kannan, K. Kurach, et al. 2016. Smart Reply: Automated Response Suggestion for Email. In *KDD*.

D. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

D. P. Kingma, T. Salimans, and M. Welling. 2016. Improving variational inference with inverse autoregressive flow. *NIPS*, pages 4736–4744.

D. P. Kingma and M. Welling. 2014. Auto-encoding variational Bayes. *ICLR*.

H. Larochelle and S. Lauly. 2012. A neural autoregressive topic model. In *NIPS*, pages 2708–2716.

A. B. Lindbo Larsen, S. K. Sønderby, and O. Winther. 2016. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, pages 1558–1566.

S. Lauly, Y. Zheng, A. Allauzen, and H. Larochelle. 2016. Document neural autoregressive distribution estimation. *arXiv preprint arXiv:1603.05962*.

J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *The North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 110–119.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Ryan T. Lowe, Nissan Pow, Iulian V. Serban, Laurent Charlin, Chia-Wei Liu, and Joelle Pineau. 2017. Training End-to-End Dialogue Systems with the Ubuntu Dialogue Corpus. *Dialogue & Discourse*, 8(1).

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. 2016. Auxiliary deep generative models. In *ICML*, pages 1445–1453.

C. J. Maddison, A. Mnih, and Y. W. Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*.

J. Markoff and P. Mozur. 2015. For Sympathetic Ear, More Chinese Turn to Smartphone Program. *New York Times*.

Y. Miao, L. Yu, and P. Blunsom. 2016. Neural variational inference for text processing. In *ICML*, pages 1727–1736.

A. Mnih and K. Gregor. 2014. Neural variational inference and learning in belief networks. In *ICML*, pages 1791–1799.

R. M. Neal. 1992. Connectionist learning of belief networks. *Artificial intelligence*, 56(1):71–113.

A. G. Ororbia II, C. L. Giles, and D. Reitter. 2015. Online semi-supervised learning with deep hybrid boltzmann machines and denoising autoencoders. *arXiv preprint arXiv:1511.06964*.

R. Pascanu, T. Mikolov, and Y. Bengio. 2012. On the difficulty of training recurrent neural networks. *ICML*, 28:1310–1318.

Rajesh Ranganath, Dustin Tran, and David Blei. 2016. Hierarchical variational models. In *ICML*, pages 324–333.

D. J. Rezende and S. Mohamed. 2015. Variational inference with normalizing flows. In *ICML*, pages 1530–1538.

D. J. Rezende, S. Mohamed, and D. Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, pages 1278–1286.

A. Ritter, C. Cherry, and W. B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 583–593.

J. T. Rolfe. 2017. Discrete variational autoencoders. In *ICLR*.

Francisco R Ruiz, Michalis Titsias RC AUEB, and David Blei. 2016. The generalized reparameterization gradient. In *NIPS*, pages 460–468.

R. Salakhutdinov and G. E. Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978.

R. Salakhutdinov and H. Larochelle. 2010. Efficient learning of deep boltzmann machines. In *AISTATs*, pages 693–700.

T. Salimans, D. P Kingma, and M. Welling. 2015. Markov chain monte carlo and variational inference: Bridging the gap. In *ICML*, pages 1218–1226.

R. Sennrich, B. Haddow, and A. Birch. 2016. Neural machine translation of rare words with subword units. In *Association for Computational Linguistics (ACL)*.

I. V. Serban, T. Klinger, G. Tesauro, K. Talamadupula, B. Zhou, Y. Bengio, and A. Courville. 2017a. Multiresolution recurrent neural networks: An application to dialogue response generation. In *Thirty-First AAAI Conference (AAAI)*.

I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau. 2016a. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference (AAAI)*.

I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio. 2017b. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference (AAAI)*.

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2016b. Generative deep neural networks for dialogue: A short review. In *NIPS, Let's Discuss: Learning Methods for Dialogue Workshop*.

A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J. Nie, J. Gao, and B. Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2015)*, pages 196–205.

N. Srivastava, R. R Salakhutdinov, and G. E. Hinton. 2013. Modeling documents with deep boltzmann machines. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 616–624.

B. Uria, I. Murray, and H. Larochelle. 2014. A deep and tractable density estimator. In *ICML*, pages 467–475.

K. Zhai and J. D. Williams. 2014. Discovering latent structure in task-oriented dialogues. In *Association for Computational Linguistics (ACL)*, pages 36–46.

Learning the Structure of Variable-Order CRFs: a Finite-State Perspective

Thomas Lavergne and **François Yvon**
LIMSI, CNRS, Univ. Paris-Sud, Université Paris Saclay
Campus Universitaire, F-91 403 Orsay, France
{lavergne,yvon}@limsi.fr

Abstract

The computational complexity of linear-chain Conditional Random Fields (CRFs) makes it difficult to deal with very large label sets and long range dependencies. Such situations are not rare and arise when dealing with morphologically rich languages or joint labelling tasks. We extend here recent proposals to consider variable order CRFs. Using an effective finite-state representation of variable-length dependencies, we propose new ways to perform feature selection at large scale and report experimental results where we outperform strong baselines on a tagging task.

1 Introduction

Conditional Random Fields (CRFs) (Lafferty et al., 2001; Sutton and McCallum, 2006) are a method of choice for many sequence labelling tasks such as Part of Speech (PoS) tagging, Text Chunking, or Named Entity Recognition. Linear-chain CRFs are easy to train by solving a convex optimization problem, can accomodate rich feature patterns, and enjoy polynomial exact inference procedures. They also deliver state-of-the-art performance for many tasks, sometimes surpassing seq2seq neural models (Schnober et al., 2016).

A major issue with CRFs is the complexity of training and inference procedures, which are quadratic in the number of possible output labels for first order models and grow exponentially when higher order dependencies are considered. This is problematic for tasks such as precise PoS tagging for Morphologically Rich Languages (MRLs), where the number of morphosyntactic labels is in the thousands (Hajič, 2000; Müller et al., 2013). Large label sets also naturally arise when *joint labelling tasks* (eg. simultaneous PoS tagging and text chunking) are considered, For such tasks, processing first-order models is demanding, and full size higher-order models are out of the question. Attempts to overcome this difficulty are based on a greedy approach which starts with first-order dependencies between labels and iteratively increases the scope of dependency patterns under the constraint that a high-order dependency is selected only if it extends an existing lower order feature (Müller et al., 2013). As a result, feature selection may only choose only few higher-order features, motivating the need for an effective variable-order CRF (voCRF) training procedure (Ye et al., 2009).[1] The latest implementation of this idea (Vieira et al., 2016) relies on (structured) sparsity promoting regularization (Martins et al., 2011) and on finite-state techniques, handling high-order features at a small extra cost (see § 2). In this approach, the sparse set of label dependency patterns is represented in a finite-state automaton, which arises as the result of the feature selection process.

In this paper, we somehow reverse the perspective and consider VoCRF training mostly as an automaton inference problem. This leads us to consider alternative techniques for learning the finite-state machine representing the dependency structure of sparse VoCRFs (see § 3). Two lines of enquiries are explored: (a) to take into account the internal structure of large tag sets in order to learn better and/or leaner feature sets; (b) to detect unconditional structural dependencies in label sequences in order to speed-up the discovery of useful features. These ideas are implemented in 6 feature selection strategies, allowing us to explore a large set of dependency structures. Relying on lazy finite-state operations, we train VoCRFs up to order 5, and achieve PoS tagging performance that

[1] This is reminiscent of *variable order HMMs*, introduced eg. in (Schütze and Singer, 1994; Ron et al., 1996).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 433–439
Copenhagen, Denmark, September 7–11, 2017.

surpass strong baselines for two MRLs (see § 4).

2 Variable order CRFs

In this section, we recall the basics of CRFs and VoCRFs and introduce some notations.

2.1 Basics

First-order CRFs use the following model:

$$p_\theta(\mathbf{y}|\mathbf{x}) = Z_\theta(\mathbf{x})^{-1} \exp(\theta^T F(\mathbf{x}, \mathbf{y})) \quad (1)$$

where $\mathbf{x} = (x_1, \ldots, x_T)$ and $\mathbf{y} = (y_1, \ldots, y_T)$ are the input (in $\mathcal{X}^T$) and output (in $\mathcal{Y}^T$) sequences and $Z_\theta(\mathbf{x})$ is a normalizer. Each component $F_j(\mathbf{x}, \mathbf{y})$ of the global feature vector decomposes as a sum of local features $\sum_{t=1}^{T} f_j(y_{t-1}, y_t, x_t)$ and is associated to parameter θ_j. Local features typically use binary tests and take the form:

$$f_{u,g}(y_{t-1}, y_t, x, t) = \mathbb{I}(y_t = u \wedge g(x,t))$$
$$f_{uv,g}(y_{t-1}, y_t, x, t) = \mathbb{I}(y_{t-1} y_t = uv \wedge g(x,t))$$

where $\mathbb{I}()$ is an indicator function and $g()$ tests a local property of $\mathbf{x}$ around x_t. In this setting, the number of parameters is $|\mathcal{Y}|^2 \times |\mathcal{X}|_{\text{train}}$, where $|A|$ is the cardinality of A and $|\mathcal{X}|_{\text{train}}$ is the number of values of $g(x,t)$ observed in the training set. Even in moderate size applications, the parameter set can be very large and contain dozen of millions of features, due to the introduction of sequential dependencies in the model.

Given N i.i.d. sequences $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$, estimation is based on the minimization of the negated conditional log-likelihood $l(\theta)$. Optimizing this objective requires to compute its gradient and to repeatedly evaluate the conditional expectation of the feature vector. This can be done using a forward-backward algorithm having a complexity that grows quadratically with $|\mathcal{Y}|$. $l(\theta)$ is usually complemented with a regularization term so as to avoid overfitting and stabilize the optimization. Common regularizers use the ℓ_1- or the ℓ_2-norm of the parameter vector, the former having the benefit to promote sparsity, thereby performing automatic feature selection (Tibshirani, 1996).

2.2 Variable order CRFs (VoCRFs)

When the label set is large, many pairs of labels never occur in the training data and the sparsity of label ngrams quickly increases with the order p of the model. In the *variable order CRF* model, it is assumed that only a small number of ngrams (out of $|\mathcal{Y}|^p$) are associated with a non-zero parameter value. Denoting $\mathcal{W}$ the set of such ngrams and $w \in \mathcal{W}$, a generic feature function is then $f_{w,g}(w, x, t) = \mathbb{I}(y_{t-s} \ldots y_t = w \wedge g(x,t))$.

Algorithm 1: Building $A[\mathcal{W}]$

```
$\mathcal{W}$ : list of patterns, $A[\mathcal{W}]$ initially empty
$\mathcal{U}$ = Pref($\mathcal{W}$)
foreach $w \in \mathcal{W}$ do
    TrieInsert($w, A[\mathcal{W}]$)
// Add missing transitions
foreach $u = vy \in \mathcal{U}$ do
    new FailureTrans($u$, LgSuff($v$,$\mathcal{U}$))
```

In (order-p) VoCRFs, the computational cost of training and inference is proportional to the size of a finite-state automaton $A[\mathcal{W}]$ encoding the patterns in $\mathcal{W}$,[2] which can be much less than $|\mathcal{Y}|^p$. Our procedure for building $A[\mathcal{W}]$ is sketched in Algorithm 1, where `TrieInsert` inserts a string in a trie, `Pref`$(\mathcal{W})$ computes the set of prefixes of the strings in $\mathcal{W}$,[3] `LgSuff`$(v,\mathcal{U})$ returns the longest suffix of v in $\mathcal{U}$, and `FailureTrans` is a special ε-transition used only when no labelled transition exists (Allauzen et al., 2003).[4] Each state (or pattern prefix) v in $A[\mathcal{W}]$ is associated with a set of feature functions $\{f_{u,g}, \forall u \in$ `Suff`$(v), g\}$.[5] The forward step of the gradient computation maintains one value $\alpha(v,t)$ per state and time step, which is recursively accumulated over all paths ending in v at time t.

The next question is to identify $\mathcal{W}$. The simplest method keeps all the ngrams viewed in training, additionally filtering rare patterns (Cuong et al., 2014). However, frequency based feature selection does not take interactions into account and is not the best solution. Ideally, one would like to train a complete order-p model with a sparsity promoting penalty, a technique that only works for small label sets.[6] The greedy algorithm of

[2] More precisely, Vieira et al. (2016) consider $\overline{\mathcal{W}}$, the closure of $\mathcal{W}$ under suffix and last character substitution, which factors as $\overline{\mathcal{W}} = \mathcal{H} \times \mathcal{Y}$. The complexity of training depends on the size of the finite-state automaton representing $\overline{\mathcal{W}}$.

[3] A trie has one state for each prefix.

[4] This was also suggested by Cotterell and Eisner (2015) as a way to build a more compact pattern automaton.

[5] Upon reaching a state v, we need to access the features that fire for that pattern, and also for all its suffixes. Each state thus stores a set of pattern; each pattern is associated with a set of tests on the observation (cf. 2.1).

[6] Recall that the size of parameter set is exponential wrt. the model order.

Schmidt and Murphy (2010); Vieira et al. (2016) is more scalable: it starts with all unigram patterns and iteratively grows $\mathcal{W}$ by extending the ngrams that have been selected in the simpler model. At each round of training, feature selection is performed using a ℓ_1 penalty and identifies the patterns that will be further augmented.

3 Learning patterns

We introduce now several alternatives for learning $\mathcal{W}$. Our motivation for doing so is twofold: (a) to take the internal structure of large label sets into account; (b) to identify more abstract patterns in label sequences, possibly containing gaps or iterations, which could yield smaller $A[\mathcal{W}]$. As discussed below, both motivations can be combined.

3.1 Greedy ℓ_1

The greedy strategy iteratively grows patterns up to order p. Considering all possible unigram and bigram patterns, we train a sparse model to select a first set of useful bigrams. In subsequent iterations, each pattern w selected at order k is extended in all possible ways to specify the pattern set at order $k + 1$, which will be filtered during the next training round. This approach is close, yet simpler, than the group lasso approach of Vieira et al. (2016) and experimentally yields slightly smaller pattern sets (see Table 2). This is because we do not enforce closure under last-character replacement: once pattern w is pruned, longer patterns ending in w are never considered.[7]

3.2 Component-wise training

Large tag sets often occur in joint tasks, where multiple levels of information are encoded in one compound tag. For instance, the fine grain labels in the Tiger corpus (Brants et al., 2002) combine PoS and morphological information in tags such as `NN.Dat.Sg.Fem` for a feminine singular dative noun. In the sequel, we refer to each piece of information as a tag *component*. We assume that all tags contain the same components, using a "non-applicable" value whenever needed. Using features that test arbitrary combinations of tag components would make feature selection much more difficult, as the number of possible patterns grows combinatorially with the number of components. We keep things simple by allowing features to only evaluate one single component at a time: this allows us to identify dependencies of different orders for each component.

Assuming that each tag y contains K components $y = [z_1, z_2 \dots, z_K]$, with $z_k \in \mathcal{Y}_k$, $\mathcal{W}$ is then computed as in § 3.1, except that we now consider one distinct set of patterns $\mathcal{W}_k$ for each component k. At each training round, each set $\mathcal{W}_k$ is extended and pruned independently from the others. Note that all these automata are trained simultaneously using a common set of features. This process results in K automata, which are intersected on the fly[8] using "lazy" composition. In our experiments, we also consider the case where we additionally combine the automaton representing complete tag sequences: this has the beneficial effect to restrict the combinations of subtags to values that actually exist in the data.

3.3 Pruned language models

Another approach for computing $\mathcal{W}$ assumes that useful dependencies between tags can be identified using an auxiliary language model (LM) trained *without paying any attention to observation sequences*. A pattern w will then be deemed useful for the labelling task only if w is a useful history in a LM of tag sequences. This strategy was implemented by first training a compact p-gram LM with entropy pruning[9] (Stolcke, 1998) and including all the surviving histories in $\mathcal{W}$. In a second step, we train the complete CRF as usual, with all observation features and ℓ_1 penalty to further prune the parameter set.

	cz	de
train set	38,727	40,474
development set	5,228	5,000
test set	4,213	5,000
# PoS	12	54
# attributes	13	8
# full tags	1,924	781

Table 1: Corpus description

3.4 Maximum entropy language models

Another technique, which combines the two previous ideas, relies on Maximum Entropy LMs

[7] cf. the discussion in (Vieira et al., 2016, § 4).

[8] Formally, each $A[\mathcal{W}_k]$ has transitions labelled with elements of $\mathcal{Y}_k$; lazy intersection operates on "generalized" transitions, where each label z is replaced with $[?, \dots, z, \dots, ?]$, where ? matches any symbol. $A[\mathcal{W}]$ is the intersection $\bigcap_k A[\mathcal{W}_k]$ and is labelled with completely specified tags.

[9] Starting with a full back-off n-gram language model, this approach discards n-grams if their removal causes a sufficiently small drop in cross-entropy. We used the implementation of Stolcke (2002).

(MELMs) (Rosenfeld, 1996). MELMs decompose the probabililty of a sequence $y_1 \dots y_T$ using the chain rule, where each term $p_\lambda(y_t|y_{<t})$ is a locally normalized exponential model including all possible ngram features up to order p:

$$p(y_t|y_{<t}; \lambda) = Z(\lambda)^{-1} \exp \lambda^T G(y_1 \dots y_t)$$

In contrast to globally normalized models, the complexity of training remains linear wrt. $|\mathcal{Y}|$, irrespective of p. It it also straightforward both to (a) use a ℓ_1 penalty to perform feature selection; (b) include features that only test specific components of a complex tag. For an order p model, our feature functions evaluate all n-grams (for $n \leq p$) of complete tags or of one specific component:

$$\begin{aligned} G_w(y_1, \dots, y_t) =& \mathbb{I}(y_{t-n+1} \dots y_t = w) \\ G_u(y_1, \dots, y_t) =& \mathbb{I}(z_{k,t-n+1} \dots z_{k,t} = u) \end{aligned}$$

Once a first round of feature selection has been performed,[10] we compute $A[\mathcal{W}]$ as explained above. The last step of training reintroduces the observations and estimates the CRF paramaters. A variant of this approach adds extra *gappy* features to the n-gram features. Gappy features at order p test whether some label u occurs in the remote past anywhere between position $t - p + 1$[11] and $t - n$. They take the following form:

$$\begin{aligned} G_{w,u}(y_1, \dots, y_t) =& \mathbb{I}(y_{t-n+1} \dots y_t = w \wedge \\ & u \in \{y_{t-p+1} \dots y_{t-n}\}), \end{aligned}$$

and likewise for features testing components.

4 Experiments

4.1 Training protocol

The following protocol is used throughout: (a) identify $\mathcal{W}$ (§3) - note that this may imply to tune a regularization parameter; (b) train a full model (including tests on the observations for each pattern in $\mathcal{W}$) using ℓ_1 regularization and a very small ℓ_2 term to stabilize convergence. The best regularization in (a) and (b) is selected on development data and targets either perplexity (for LMs) or label accuracy (for CRFs).

4.2 Datasets and Features

Experiments are run on two MRLs: for Czech, we use the CoNLL 2009 data set (Hajič et al., 2009) and for German, the Tiger Treebank with the split of Fraser et al. (2013)). Both datasets include rich morphological attributes (cf. Table 1).

All the patterns in $\mathcal{W}$ are combined with *lexical* features testing the current word x_t, its prefixes and suffixes of length 1 to 4, its capitalization and the presence of digit or punctuation symbols. Additional contextual features also test words in a local window around position t. These tests greatly increase the feature count and are not provided for all label patterns: for unigram patterns, we test the presence of all unigrams and bigrams of words in a window of 5 words; for bigrams patterns we only test for all unigrams in a window of 3 words. Contextual features are not used for larger patterns.

4.3 Results

We consider several baselines: Maxent and MEMM models, neither of which considers label dependencies in training, a linear chain CRF[12] and our own implementation of the group lasso of Vieira et al. (2016). For the latter, we contrast two setups: one where each pattern in $\mathcal{W}$ gives rise to one single feature, and one where it is conjoined with tests on the observation.[13] All scores in Table 2 are label accuracies on unseen test data.

As expected, Maxent and MEMM are outperformed by almost all variants of CRFs, and their scores are only reported for completeness. `Group lasso` results demonstrate the effectiveness of using contextual information with high order features: the gain is ≈ 0.7 points for both languages and all values of p. `Greedy` ℓ_1 achieves accuracy results similar to `group lasso`, suggesting that ℓ_1 penalty alone is effective to select high-order features. It also yields slighly smaller models and very comparable training time across the board: indeed, greedy parameter selection strategies imply multiple rounds of training which are overall quite costly, due to the size of the full label set. Testing individual subtags (§ 3.2) results in a slight improvement (≈+0.3) in accuracy over `Greedy` ℓ_1. When using an additional automata for the full tag, we get a larger gain of ≈ 0.6 points for Czech, slightly less for German: including a model for complete tags also prevents to gener-

[10] As the LM building step only look at labels, we tune the regularization to optimize the perplexity of the LM on a development set.

[11] We use $p = 6$ in our experiments.

[12] Using the implementation of Lavergne et al. (2010).

[13] As suggested by the authors themselves in fn 4.

	cz				de			
	$p = 2$	$p = 3$	$p = 4$	$p = 5$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
Maxent	90.01% 1924 191min	91.12% 1924 219min	91.17% 1924 286min	91.14% 1924 349min	85.62% 781 142min	85.84% 781 193min	85.96% 781 252min	86.02% 781 297min
MEMM	90.96% 1924 191min	92.09% 1924 219min	92.13% 1924 286min	92.12% 1924 349min	86.48% 781 142min	86.88% 781 193min	87.13% 781 252min	87.19% 781 297min
Linear Chain CRF	91.93% 3.7e6 657min	–	–	–	86.95% 6.1e5 447min	–	–	–
Group lasso	91.91% 9.6e5 421min	92.27% 4.3e7 1656min	92.41% 1.2e8 3067min	–	86.92% 1.8e5 305min	87.24% 9.2e6 1134min	87.48% 5.4e7 2101min	–
Group lasso + ctx	92.51% 9.2e5 520min	92.95% 4.1e7 1632min	93.03% 1.2e8 3285min	–	87.48% 1.7e5 349min	87.92% 7.8e6 1218min	87.96% 5.3e7 2398min	–
Greedy ℓ_1	92.47% 8.4e5 462min	92.94% 4.1e7 1759min	93.01% 1.1e8 3300min	–	87.43% 1.7e5 340min	87.87% 7.1e6 1239min	87.96% 5.0e7 2357min	–
Component-wise	92.76% 6.2e4 247min	93.24% 2.8e5 370min	93.36% 8.2e5 1179min	93.28% 3.7e6 2224min	87.47% 2.4e4 173min	88.16% 7.2e4 268min	88.26% 3.7e5 836min	88.29% 1.4e6 1483min
Component-wise + Full	92.97% 8.7e5 463min	93.41% 2.8e7 1569min	93.69% 8.3e7 3162min	93.65% 4.6e8 4321min	87.39% 1.4e5 311min	88.36% 5.2e6 1097min	88.59% 2.1e7 2181min	88.60% 1.3e8 3249min
Pruned LM	92.98% 3.2e5 233min	93.27% 8.2e6 487min	93.51% 1.1e7 1210min	93.53% 8.6e7 2519min	87.43% 1.3e5 163min	88.12% 8.9e5 372min	88.25% 9.1e6 896min	88.21% 5.3e7 1894min
MELM	93.02% 4.6e5 303min	93.33% 1.7e7 545min	93.81% 2.3e7 1478min	93.63% 1.4e8 2559min	87.41% 1.4e5 206min	88.61% 2.9e6 407min	88.76% 1.4e7 1063min	88.74% 9.8e7 1924min
MELM + Gaps	93.52% 4.5e5 289min	93.68% 1.5e7 658min	93.79% 1.9e7 1751min	–	88.38% 1.4e5 217min	88.70% 2.3e6 439min	88.78% 1.1e7 1297min	–

Table 2: Experimental results. Each cell reports accuracy, number of states in $A[\mathcal{W}]$ and total training time. Group lasso is our reimplementation of Vieira et al. (2016) (+Ctx = +context features) ; Greedy ℓ_1 is described in section 3.1, Component-wise is the decomposition approach of § 3.2, PrunedLM and MELM (+Gaps) were described in § 3.3 and § 3.4.

ate invalid combinations of subtags. These models represent different tradeoffs between accuracy and training time: the 4-gram `Component-wise` experiment only took 14 hrs to complete on German data and outperforms the corresponding `Greedy` ℓ_1 setup while containing approximately 100 times less features. `Component-wise+Full` is more comparable in size and training time to `Greedy` ℓ_1, but yields a larger improvement in performance. The last sets of experiments with LMs yields even better operating points, as the first stage of pattern selection is performed with a cheap model. They are our best trade-off to date, yielding the best performance for all values of p.

5 Conclusion

In this work, we have explored ways to take advantage of the flexibility offered by implementations of VoCRFs based on finite-state techniques. We have proposed strategies to include tests on subparts of complex tags, as well as to select useful label patterns with auxiliary unconditional LMs. Experiments with two MRLs with large tagsets yielded consistent improvements (≈ +0.8 points) over strong baselines. They offer new perspectives to perform feature selection in high order CRFs. In our future work, we intend to also explore how to complement ℓ_1 penalties with terms penalizing more explicitely the processing time; we also wish to study how these ideas can be used in combination with neural models.

Acknowledgements

The authors wish to thank the reviewers for their useful comments and suggestions. This work has been partly funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 645452 (QT21).

References

Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sapporo, Japan, pages 40–47. https://doi.org/10.3115/1075096.1075102.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist.* 22(1):39–71.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the workshop on treebanks and linguistic theories*. pages 24–41.

Ryan Cotterell and Jason Eisner. 2015. Penalized expectation propagation for graphical models over strings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 932–942. https://doi.org/10.3115/v1/N15-1094.

Nguyen Viet Cuong, Nan Ye, Wee Sun Lee, and Hai Leong Chieu. 2014. Conditional Random Field with High-order Dependencies for Sequence Labeling and Segmentation. *Journal of Machine Learning Research* 15:981–1009. http://jmlr.org/papers/v15/cuong14a.html.

Alexander Fraser, Helmut Schmid, Richárd Farkas, Renjing Wang, and Hinrich Schütze. 2013. Knowledge sources for constituent parsing of german, a morphologically rich and less-configurational language. *CL* 39(1):57–85.

Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*. Seattle, WA, pages 94–101.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*. CoNLL '09, pages 1–18.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, Williamstown, MA, (ICML'01), pages 282–289.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden, pages 504–513.

Andre Martins, Noah Smith, Mario Figueiredo, and Pedro Aguiar. 2011. Structured sparsity in structured prediction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. pages 1500–1511.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, EMNLP'15, pages 2268–2274.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA, EMNLP'15, pages 322–332.

Chris Pal, Charles Sutton, and Andrew McCallum. 2006. Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*. volume 5, pages V–V. https://doi.org/10.1109/ICASSP.2006.1661342.

Dana Ron, Yoram Singer, and Naftali Tishby. 1996. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning* 25(2-3):117–149.

Ronald Rosenfeld. 1996. A maximum entropy approach to adaptive statistical learning modeling. *Computer, Speech and Language* 10:187 – 228.

Mark W. Schmidt and Kevin P. Murphy. 2010. Convex structure learning in log-linear models: Beyond pairwise potentials. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics,*. Chia Laguna Resort, Sardinia, Italy, AISTATS, pages 709–716.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 1703–1714.

Hinrich Schütze and Yoram Singer. 1994. Part-of-speech tagging using a variable memory Markov model. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*. Las Cruces, New Mexico, pages 181–187.

Nobuyuki Shimizu and Andrew Haas. 2006. Exact decoding for jointly labeling and chunking sequences. In *Proceedings of COLING/ACL*. pages 763–770.

Noah A. Smith, David A. Smith, and Roy W. Tromble. 2005. Context-based morphological disambiguation with random fields. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, British Columbia, Canada, pages 475–482.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*. Lansdowne, VA, pages 270–274.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Langage Processing (ICSLP)*. Denver, CO, volume 2, pages 901–904.

Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. The MIT Press, Cambridge, MA.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B* 58(1):267–288.

Kristina Toutanova and Colin Cherry. 2009. A global model for joint lemmatization and part-of-speech prediction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, pages 486–494. http://aclweb.org/anthology/P09-1055.

Tim Vieira, Ryan Cotterell, and Jason Eisner. 2016. Speed-accuracy tradeoffs in tagging with variable-order crfs and structured sparsity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. EMNLP, pages 1973–1978.

Nan Ye, Wee S. Lee, Hai L. Chieu, and Dan Wu. 2009. Conditional random fields with high-order features for sequence labeling. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*. Curran Associates, Inc., pages 2196–2204. http://papers.nips.cc/paper/3815-conditional-random-fields-with-high-order-features-for-sequence-labeling.pdf.

Sparse Communication for Distributed Gradient Descent

Alham Fikri Aji and **Kenneth Heafield**
School of Informatics, University of Edinburgh
10 Crichton Street
Edinburgh EH8 9AB
Scotland, European Union
a.fikri@ed.ac.uk, kheafiel@inf.ed.ac.uk

Abstract

We make distributed stochastic gradient descent faster by exchanging sparse updates instead of dense updates. Gradient updates are positively skewed as most updates are near zero, so we map the 99% smallest updates (by absolute value) to zero then exchange sparse matrices. This method can be combined with quantization to further improve the compression. We explore different configurations and apply them to neural machine translation and MNIST image classification tasks. Most configurations work on MNIST, whereas different configurations reduce convergence rate on the more complex translation task. Our experiments show that we can achieve up to 49% speed up on MNIST and 22% on NMT without damaging the final accuracy or BLEU.

1 Introduction

Distributed computing is essential to train large neural networks on large data sets (Raina et al., 2009). We focus on data parallelism: nodes jointly optimize the same model on different parts of the training data, exchanging gradients and parameters over the network. This network communication is costly, so prior work developed two ways to approximately compress network traffic: 1-bit quantization (Seide et al., 2014) and sending sparse matrices by dropping small updates (Strom, 2015; Dryden et al., 2016). These methods were developed and tested on speech recognition and toy MNIST systems. In porting these approximations to neural machine translation (NMT) (Ñeco and Forcada, 1996; Bahdanau et al., 2014), we find that translation is less tolerant to quantization.

Throughout this paper, we compare neural machine translation behavior with a toy MNIST system, chosen because prior work used a similar system (Dryden et al., 2016). NMT parameters are dominated by three large embedding matrices: source language input, target language input, and target language output. These matrices deal with vocabulary words, only a small fraction of which are seen in a mini-batch, so we expect skewed gradients. In contrast, MNIST systems exercise every parameter in every mini-batch. Additionally, NMT systems consist of multiple parameters with different scales and sizes, compared to MNIST's 3-layers network with uniform size. More formally, gradient updates have positive skewness coefficient (Zwillinger and Kokoska, 1999): most are close to zero but a few are large.

2 Related Work

An orthogonal line of work optimizes the SGD algorithm and communication pattern. Zinkevich et al. (2010) proposed an asynchronous architecture where each node can push and pull the model independently to avoid waiting for the slower node. Chilimbi et al. (2014) and Recht et al. (2011) suggest updating the model without a lock, allowing race conditions. Additionally, Dean et al. (2012) run multiple minibatches before exchanging updates, reducing the communication cost. Our work is a more continuous version, in which the most important updates are sent between minibatches. Zhang et al. (2015) downweight gradients based on stale parameters.

Approximate gradient compression is not a new idea. 1-Bit SGD (Seide et al., 2014), and later Quantization SGD (Alistarh et al., 2016), work by converting the gradient update into a 1-bit matrix, thus reducing data communication significantly. Strom (2015) proposed threshold quantiza-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 440–445
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

tion, which only sends gradient updates that larger than a predefined constant threshold. However, the optimal threshold is not easy to choose and, moreover, it can change over time during optimization. Dryden et al. (2016) set the threshold so as to keep a constant number of gradients each iteration.

3 Distributed SGD

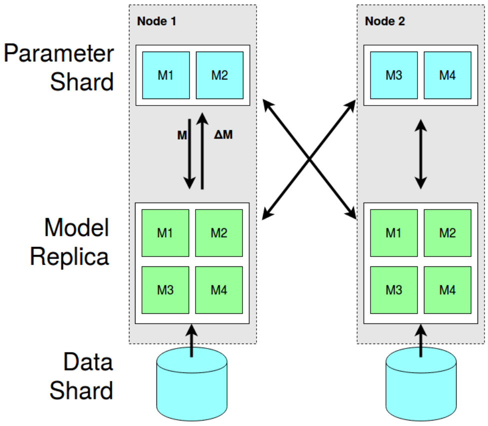


Figure 1: Distributed SGD architecture with parameter sharding.

We used distributed SGD with parameter sharding (Dean et al., 2012), shown in Figure 1. Each of the N workers is both a client and a server. Servers are responsible for $1/N$th of the parameters.

Clients have a copy of all parameters, which they use to compute gradients. These gradients are split into N pieces and pushed to the appropriate servers. Similarly, each client pulls parameters from all servers. Each node converses with all N nodes regarding $1/N$th of the parameters, so bandwidth per node is constant.

4 Sparse Gradient Exchange

We sparsify gradient updates by removing the R% smallest gradients by absolute value, dubbing this Gradient Dropping. This approach is slightly different from Dryden et al. (2016) as we used a single threshold based on absolute value, instead of dropping the positive and negative gradients separately. This is simpler to execute and works just as well.

Small gradients can accumulate over time and we find that zeroing them damages convergence. Following Seide et al. (2014), we remember residuals (in our case dropped values) locally and add them to the next gradient, before dropping again.

Algorithm 1 Gradient dropping algorithm given gradient ∇ and dropping rate R.

```
function GRADDROP($\nabla$, $R$)
    $\nabla + = residuals$
    Select $threshold$: R% of $|\nabla|$ is smaller
    $dropped \leftarrow 0$
    $dropped[i] \leftarrow \nabla[i] \forall i : |\nabla[i]| > threshold$
    $residuals \leftarrow \nabla - dropped$
    return $sparse(dropped)$
end function
```

Gradient Dropping is shown in Algorithm 1. This function is applied to all data transmissions, including parameter pulls encoded as deltas from the last version pulled by the client. To compute these deltas, we store the last pulled copy server-side. Synchronous SGD has one copy. Asynchronous SGD has a copy per client, but the server is responsible for $1/N$th of the parameters for N clients so memory is constant.

Selection to obtain the threshold is expensive (Alabi et al., 2012). However, this can be approximated. We sample 0.1% of the gradient and obtain the threshold by running selection on the samples.

Parameters and their gradients may not be on comparable scales across different parts of the neural network. We can select a threshold locally to each matrix of parameters or globally for all parameters. In the experiments, we find that layer normalization (Lei Ba et al., 2016) makes a global threshold work, so by default we use layer normalization with one global threshold. Without layer normalization, a global threshold degrades convergence for NMT. Prior work used global thresholds and sometimes column-wise quantization.

5 Experiment

We experiment with an image classification task based on MNIST dataset (LeCun et al., 1998) and Romanian→English neural machine translation system.

For our image classification experiment, we build a fully connected neural network with three 4069-neuron hidden layers. We use AdaGrad with an initial learning rate of 0.005. The mini-batch size of 40 is used. This setup is identical to Dryden et al. (2016).

Our NMT experiment is based on Sennrich et al. (2016), which won first place in the 2016 Workshop on Machine Translation. It is based on an attentional encoder-decoder LSTM with

Drop Ratio	words/sec (NMT)	images/sec (MNIST)
0%	13100	2489
90%	14443	3174
99%	14740	3726
99.9%	14786	3921

Table 1: Training speed with various drop ratios.

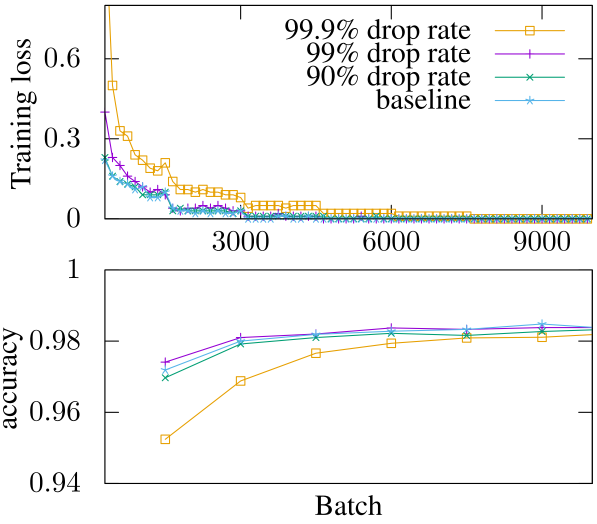


Figure 2: MNIST: Training loss and accuracy for different dropping ratios.

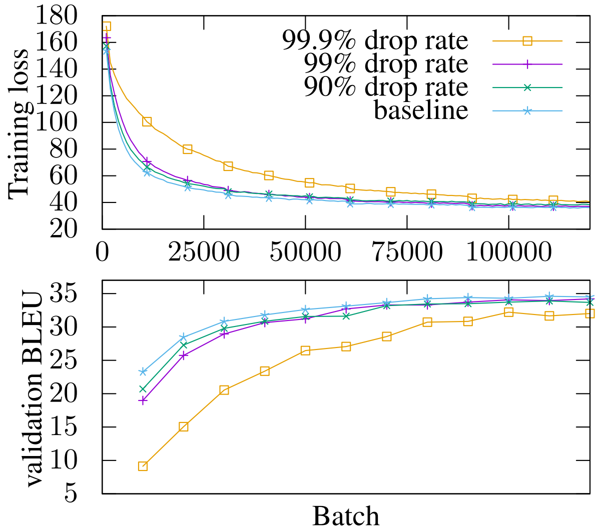


Figure 3: NMT: Training loss and validation BLEU for different dropping ratios.

119M parameters. The default batch size is 80. We save and validate every 10000 steps. We pick 4 saved models with the highest validation BLEU and average them into the final model. AmuNMT (Junczys-Dowmunt et al., 2016) is used for decoding with a beam size of 12. Our test system has PCI Express 3.0 x16 for each of 4 NVIDIA Pascal Titan Xs. All experiments used asynchronous SGD, though our method applies to synchronous SGD as well.

5.1 Drop Ratio

To find an appropriate dropping ratio $R\%$, we tried 90%, 99%, and 99.9% then measured performance in terms of loss and classification accuracy or translation quality approximated by BLEU (Papineni et al., 2002) for image classification and NMT task respectively.

Figure 3 shows that the model still learns after dropping 99.9% of the gradients, albeit with a worse BLEU score. However, dropping 99% of the gradient has little impact on convergence or BLEU, despite exchanging 50x less data with offset-value encoding. The x-axis in both plots is batches, showing that we are not relying on speed improvement to compensate for convergence.

Dryden et al. (2016) used a fixed dropping ratio of 98.4% without testing other options. Switching to 99% corresponds to more than a 1.5x reduction in network bandwidth.

For MNIST, gradient dropping oddly improves accuracy in early batches. The same is not seen for NMT, so we caution against interpreting slight gains on MNIST as regularization.

5.2 Local vs Global Threshold

Parameters may not be on a comparable scale so, as discussed in Section 4, we experiment with local thresholds for each matrix or a global threshold for all gradients. We also investigate the effect of layer normalization. We use a drop ratio of 99% as suggested previously. Based on the results and due to the complicated interaction with sharding, we did not implement locally thresholded pulling, so only locally thresholded pushing is shown.

The results show that layer normalization has no visible impact on MNIST. On the other side, our NMT system performed poorly as, without layer normalization, parameters are on various scales and global thresholding underperforms. Furthermore, our NMT system has more parameter categories compared to MNIST's 3-layer network.

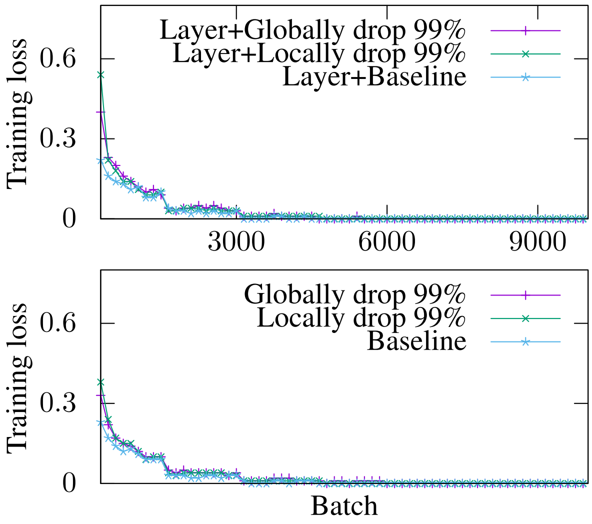


Figure 4: MNIST: Comparison of local and global thresholds with and without layer normalization.

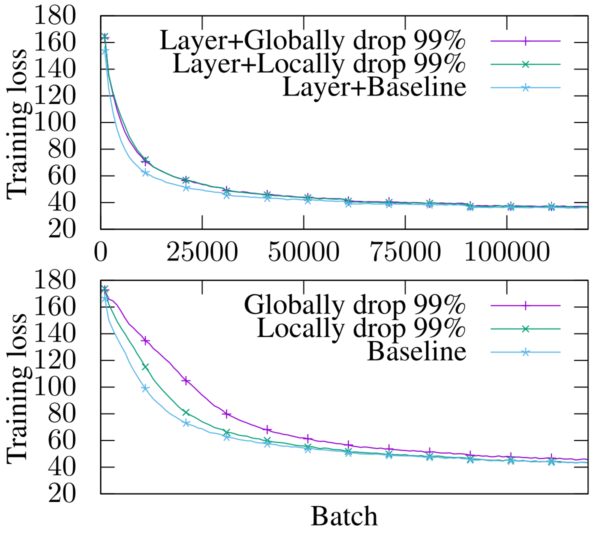


Figure 5: NMT: Comparison of local and global thresholds with and without layer normalization.

5.3 Convergence Rate

While dropping gradients greatly reduces the communication cost, it is shown in Table 1 that overall speed improvement is not significant for our NMT experiment. For our NMT experiment with 4 Titan Xs, communication time is only around 13% of the total training time. Dropping 99% of the gradient leads to 11% speed improvement. Additionally, we added an extra experiment of NMT with batch-size of 32 to give more communication cost ratio. In this scenario, communication is 17% of the total training time and we see a 22% average speed improvement. For MNIST, communication is 41% of the total training time and we see a 49% average speed improvement. Computation got faster by reducing multitasking.

We investigate the convergence rate: the combination of loss and speed. For MNIST, we train the model for 20 epochs as mentioned in Dryden et al. (2016). For NMT, we tested this with batch sizes of 80 and 32 and trained for 13.5 hours.

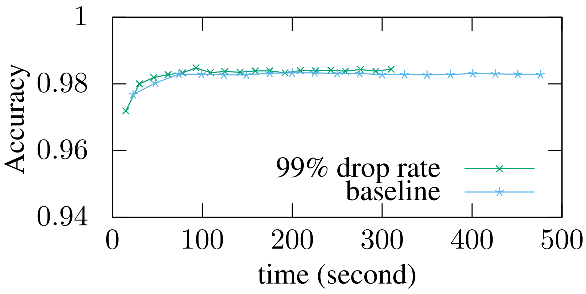


Figure 6: MNIST classification accuracy over time.

As shown in Figure 6, our baseline MNIST experiment reached 99.28% final accuracy, and reached 99.42% final accuracy with a 99% drop rate. It also shown that it has better convergence rate in general with gradient dropping.

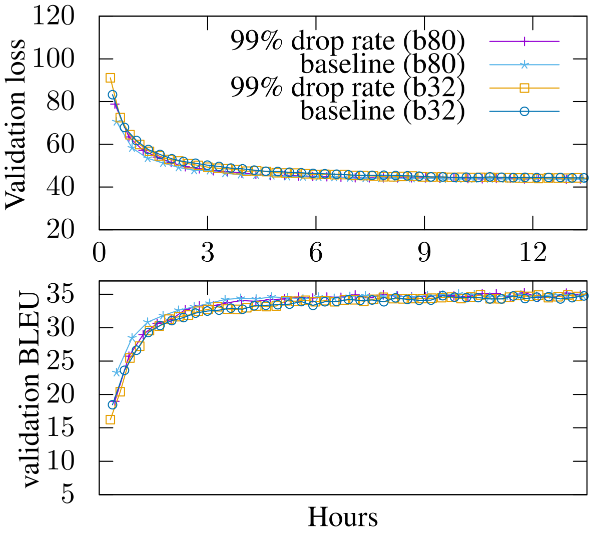


Figure 7: NMT validation BLEU and loss over time.

Our NMT experiment result is shown in Table 2. Final BLEU scores are essentially unchanged.

Experiment	Final %BLEU	Time to reach 33% BLEU
batch-size 80		
+ baseline	34.51	2.6 hours
+ 99% grad-drop	34.40	2.7 hours
batch-size 32		
+ baseline	34.16	4.2 hours
+ 99% grad-drop	34.08	3.2 hours

Table 2: Summary of BLEU score obtained.

Our algorithm converges 23% faster than the baseline when the batch size is 32, and nearly the same with a batch size of 80. This in a setting with fast communication: 15.75 GB/s theoretical over PCI express 3.0 x16.

5.4 1-Bit Quantization

We can obtain further compression by applying 1-bit quantization after gradient dropping. Strom (2015) quantized simply by mapping all surviving values to the dropping threshold, effectively the minimum surviving absolute value. Dryden et al. (2016) took the averages of values being quantized, as is more standard. They also quantized at the column level, rather than choosing centers globally. We tested 1-bit quantization with 3 different configurations: threshold, column-wise average, and global average. The quantization is applied after gradient dropping with a 99% drop rate, layer normalization, and a global threshold.

Figure 8 shows that 1-bit quantization slows down the convergence rate for NMT. This differs from prior work (Seide et al., 2014; Dryden et al., 2016) which reported no impact from 1-bit quantization. Yet, we agree with their experiments: all tested types of quantization work on MNIST. This emphasizes the need for task variety in experiments.

NMT has more skew in its top 1% gradients, so it makes sense that 1-bit quantization causes more loss. 2-bit quantization is sufficient.

6 Conclusion and Future Work

Gradient updates are positively skewed: most are close to zero. This can be exploited by keeping 99% of gradient updates locally, reducing communication size to 50x smaller with a coordinate-value encoding.

Prior work suggested that 1-bit quantization can be applied to further compress the communication.

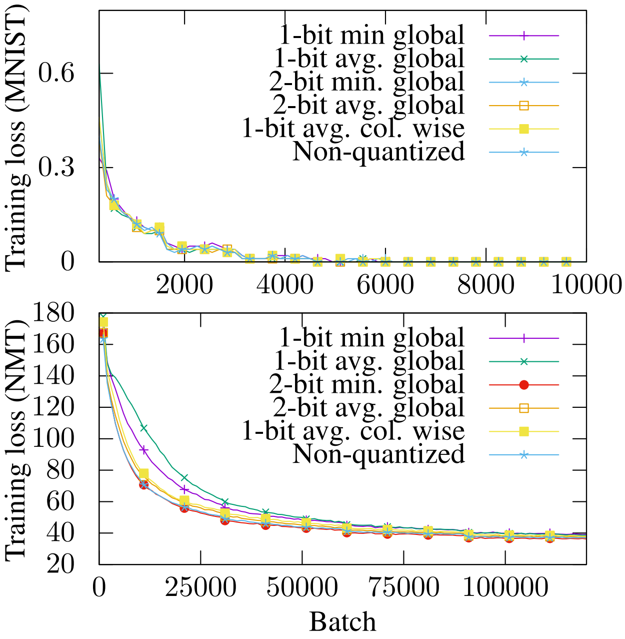


Figure 8: Training loss for different quantization methods.

However, we found out that this is not true for NMT. We attribute this to skew in the embedding layers. However, 2-bit quantization is likely to be sufficient, separating large movers from small changes. Additionally, our NMT system consists of many parameters with different scales, thus layer normalization or using local threshold per-parameter is necessary. On the hand side, MNIST seems to work with any configurations we tried.

Our experiment with 4 Titan Xs shows that on average only 17% of the time is spent communicating (with batch size 32) and we achieve 22% speed up. Our future work is to test this approach on systems with expensive communication cost, such as multi-node environments.

Acknowledgments

Alham Fikri Aji is funded by the Indonesia Endowment Fund for Education scholarship scheme. Marcin Junczys-Dowmunt wrote the baseline distributed code and consulted on this implementation. We thank School of Informatics computing and finance staff for working around NVIDIA's limit of two Pascal Titan Xs per customer. Computing was funded by the Amazon Academic Research Awards program and by Microsoft's donation of Azure time to the Alan Turing Institute. This work was supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1.

References

Tolu Alabi, Jeffrey D Blanchard, Bradley Gordon, and Russel Steinbach. 2012. Fast k-selection algorithms for graphics processing units. *Journal of Experimental Algorithmics (JEA)* 17:4–2.

Dan Alistarh, Jerry Li, Ryota Tomioka, and Milan Vojnovic. 2016. QSGD: randomized quantization for communication-optimal stochastic gradient descent. *CoRR* abs/1610.02132. http://arxiv.org/abs/1610.02132.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Trishul M Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. 2014. Project adam: Building an efficient and scalable deep learning training system. In *OSDI*. volume 14, pages 571–582.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. 2012. Large scale distributed deep networks. In *Advances in neural information processing systems*. pages 1223–1231.

Nikoli Dryden, Sam Ade Jacobs, Tim Moon, and Brian Van Essen. 2016. Communication quantization for data-parallel training of deep neural networks. In *Proceedings of the Workshop on Machine Learning in High Performance Computing Environments*. IEEE Press, pages 1–8.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? A case study on 30 translation directions. In *Program of the 13th International Workshop on Spoken Language Translation (IWSLT 2016)*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

J. Lei Ba, J. R. Kiros, and G. E. Hinton. 2016. Layer Normalization. *ArXiv e-prints* .

Ramón P Ñeco and Mikel L Forcada. 1996. Beyond mealy machines: Learning translators with recurrent neural networks. In *Proceedings of the 1996 International Neural Network Society Annual Meeting*. San Diego, California, USA.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evalution of machine translation. In *Proceedings 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, PA, pages 311–318.

Rajat Raina, Anand Madhavan, and Andrew Y Ng. 2009. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*. ACM, pages 873–880.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. 2011. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, Curran Associates, Inc., pages 693–701. http://papers.nips.cc/paper/4390-hogwild-a-lock-free-approach-to-parallelizing-stochastic-gradient-descent.pdf.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 2014. 1-bit stochastic gradient descent and application to data-parallel distributed training of speech DNNs. In *Interspeech*. https://www.microsoft.com/en-us/research/publication/1-bit-stochastic-gradient-descent-and-application-to-data-parallel-distributed-training-of-speech-dnns/.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the ACL 2016 First Conference on Machine Translation (WMT16)*.

Nikko Strom. 2015. Scalable distributed dnn training using commodity gpu cloud computing. In *INTERSPEECH*. volume 7, page 10.

Wei Zhang, Suyog Gupta, Xiangru Lian, and Ji Liu. 2015. Staleness-aware async-sgd for distributed deep learning. *CoRR* abs/1511.05950. http://arxiv.org/abs/1511.05950.

Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. 2010. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*. pages 2595–2603.

Daniel Zwillinger and Stephen Kokoska. 1999. *CRC standard probability and statistics tables and formulae*. CRC Press.

Why ADAGRAD Fails for Online Topic Modeling

You Lu
Computer Science (CS)
University of Colorado Boulder
Boulder, CO
you.lu@colorado.edu

Jeffrey Lund
Computer Science (CS)
Brigham Young University
Provo, UT
jefflund@byu.edu

Jordan Boyd-Graber
CS, iSchool, LSC, and UMIACS
University of Maryland
College Park, MD
jbg@umiacs.umd.edu

Abstract

Online topic modeling, i.e., topic modeling with stochastic variational inference, is a powerful and efficient technique for analyzing large datasets, and ADAGRAD is a widely-used technique for tuning learning rates during online gradient optimization. However, these two techniques do not work well together. We show that this is because ADAGRAD uses accumulation of previous gradients as the learning rates' denominators. For online topic modeling, the magnitude of gradients is very large. It causes learning rates to shrink very quickly, so the parameters cannot fully converge until the training ends.

Probabilistic topic models (Blei, 2012) are popular algorithms for uncovering hidden thematic structure in text. They have been widely used to help people understand and navigate document collections (Blei et al., 2003), multilingual collections (Hu et al., 2014), images (Chong et al., 2009), networks (Chang and Blei, 2009; Yang et al., 2016), etc. Probabilistic topic modeling usually requires computing a posterior distribution over thousands or millions of latent variables, which is often intractable. Variational inference (Blei et al., 2016, VI) approximates posterior distributions. Stochastic variational inference (Hoffman et al., 2013, SVI) is its natural online extension and enables the analysis of large datasets.

Online topic models (Hoffman et al., 2010; Bryant and Sudderth, 2012; Paisley et al., 2015) optimize the global parameters of interest using stochastic gradient ascent. At each iteration, they sample data points to estimate the gradient. In practice, the sample has only a small percentage of the vocabulary. The resulting sparse gradients hurt performance. ADAGRAD (Duchi et al., 2011) is designed for high dimensional online optimization problems and adjusts learning rates for each dimension, favoring rare features. This makes ADAGRAD well-suited for tasks with sparse gradients such as distributed deep networks (Dean et al., 2012), forward-backward splitting (Duchi and Singer, 2009), and regularized dual averaging methods (Xiao, 2010).

Thus, it may seem reasonable to apply ADAGRAD to optimize online topic models. However, ADAGRAD is not suitable for online topic models (Section 1). This is because to get a topic model, the training algorithm must break the symmetry between parameters of words that are highly related to the topic and words that are not related to the topic. Before the algorithm converges, the magnitude of gradients of the parameters are very large. Since ADAGRAD uses the accumulation of previous gradients as learning rates' denominators, the learning rates shrink very quickly. Thus, the algorithm cannot break the symmetry quickly. We provide solutions for this problem. Two alternative learning rate methods, i.e., ADADELTA (Zeiler, 2012) and ADAM (Kingma and Ba, 2014), can address this incompatibility with online topic models. When the dataset is small enough, e.g., a corpus with only hundreds of documents, ADAGRAD can still work.

1 Buridan's Optimizer

Latent Dirichlet allocation (Blei et al., 2003, LDA) is perhaps the most well known topic model. In this section, we analyze problems with ADAGRAD for online LDA (Hoffman et al., 2010), and provide some solutions. Our analysis is easy to generalize to other online topic models, e.g., online Hierarchical Dirichlet Process (Wang et al., 2011, HDP).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 446–451
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

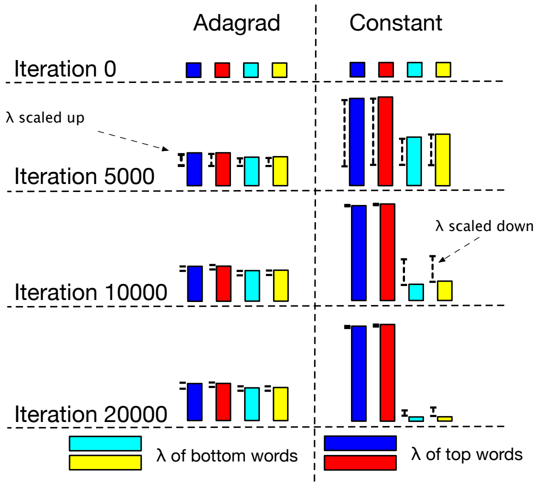


Figure 1: Illustration of ADAGRAD's problem. Initially, the topic does not favor particular words over others, so the training algorithm incorrectly increases the parameters of bottom words. Then, ADAGRAD learning rates decrease too quickly, leaving the tie between top and bottom unbroken. Thus, the algorithm fails to form appropriate topics. A constant rate easily breaks the tie. When the tie is broken, the algorithm decreases the parameters of bottom words and increases the parameters of top words until convergence.

1.1 Online LDA

To train LDA, we want to compute the posterior

$$p(\beta,\theta,z\,|\,w,\alpha,\eta) \propto \prod_{k=1}^{K} p(\beta_k\,|\,\eta)\cdot \prod_{d=1}^{D} p(\theta_d\,|\,\alpha) \prod_{n=1}^{N_d} p(z_{dn}\,|\,\theta_d)p(w_{dn}\,|\,\beta_{z_{dn}}),$$

where β_k is the topic-word distribution for the k^{th} of K topics, θ_d is the document-topic distribution for the d^{th} of D document, z_{dn} is the topic assignment for the n^{th} of N_d words in in the d^{th} document, w_{dn} is the word type of the n^{th} word in the d^{th} document, with α and η the Dirichlet priors over the document-topic and topic-word distributions.

However, this is intractable. Stochastic variational inference (SVI) is a popular approach for approximation. It first posits a mean field variational distribution

$$q(\beta,\theta,z\,|\,\lambda,\gamma,\phi) = \prod_{k=1}^{K} q(\beta_k\,|\,\lambda_k)\cdot \prod_{d=1}^{D} q(\theta_d\,|\,\gamma_d) \prod_{n=1}^{N_d} q(z_{dn}\,|\,\phi_{dn}),$$

where γ (Dirichlet) and ϕ (multinomial) are local parameters and λ (Dirichlet) is a global parameter. SVI then optimizes the variational parameters to minimize the KL divergence between the variational distribution and the true posterior.

At iteration t, SVI samples a document d from the corpus and updates the local parameters:

$$\phi^d_{vk} \propto \exp\left\{\Psi\left(\gamma_{dk}\right) + \Psi\left(\lambda^{(t)}_{kv}\right) - \Psi\left(\sum_i \lambda^{(t)}_{ki}\right)\right\}, \tag{1}$$

$$\gamma^{(t)}_k = \alpha + \sum_v n_v \phi^d_{vk}, \tag{2}$$

where n_v is the number of words v in d, and $\Psi\left(.\right)$ is the digamma function. After finding ϕ^d and γ^d, SVI optimizes the global parameters using stochastic gradient ascent,

$$\begin{aligned}\lambda^{(t+1)}_{kv} &= (1-\rho^{(t)}_{kv})\lambda^{(t)}_{kv} + \rho^{(t)}_{kv}(\eta + D\phi^d_{vk}n_{dv})\\ &= (1-\rho^{(t)}_{kv})\lambda^{(t)}_{kv} + \rho^{(t)}_{kv}\hat{\lambda}^{(t)}_{kv}\\ &= \lambda^{(t)}_{kv} + \rho^{(t)}_{kv}g^{(t)}_{kv},\end{aligned} \tag{3}$$

where $\rho^{(t)}$ is the learning rate, $\hat{\lambda}^{(t)}_{kv} = \eta + D\phi^d_{vk}n_{dv}$ is the intermediate parameter and $g^{(t)}_{kv} = -\lambda^{(t)}_{kv} + \hat{\lambda}^{(t)}_{kv}$ is the gradient.

1.2 ADAGRAD for Online LDA

In general, $\rho^{(t)}_{kv} = \kappa^{(t)}$, for all $v \in 1,..,V$ and $k \in 1,...,K$, where $\kappa^{(t)}$ can be a decreasing rate (Hoffman et al., 2013), a small constant (Collobert et al., 2011) or an adaptive rate (Ranganath et al., 2013). These three methods are all global learning rate methods, which cannot adaptively adjust learning rate for each dimension of the parameter, or address the problems caused by sparse gradients.

ADAGRAD is a popular learning rate method designed for online optimization problems with high dimension and sparse gradients. Thus, it seems reasonable to apply ADAGRAD to update learning rates for online topic models. When using ADAGRAD (Duchi et al., 2011) with online LDA, the update rule for the each learning rate is

$$\rho^{(t)}_{kv} = \frac{\rho_0}{\sqrt{\epsilon + \sum_{i=0}^{t}\left(g^{(i)}_{kv}\right)^2}}, \tag{4}$$

where ρ_0 is a constant, and a very small ϵ guarantees that the learning rates are non-zero.

1.3 ADAGRAD's Indecision

A philosophical thought experiment provides us with the story of Buridan's ass (Bayle, 1826): situated between two piles of equally tasty hay, the poor animal starved to death. ADAGRAD faces a similar problem in breaking the symmetries of common variational inference initializations. For convenience, we unfold an example with a single document at each iteration. Our analysis generalizes to mini-batches.

Initially, the topics $\beta_{1:K}$ do not favor particular words over others as inference cannot know *a priori* which words will have high probability in a particular topic. The algorithm must break ties between parameters of the top and bottom words in a topic. Unfortunately, the momentum of ADAGRAD fails for topic models. We now explain why this is.

ADAGRAD looks to the gradient for clues about what features will be important. This is because before the equilibrium is broken, the values of different λ_{kv} are close, so Equation 1 will be approximately seen as $\phi_{vk}^{d} \propto \exp\{\Psi(\gamma_{dk})\}$, which implicates that λ has very small influence on the optimization of ϕ. If some topics are prevalent in the sampled document d, large probability will be assigned to the corresponding $\phi_{\cdot k}$, meaning that all words in document d are treated as top words. The initial clues are at best random and at words counter productive.

However, ADAGRAD uses these cues to prefer some dimensions over others. Let λ^* be the optimum; the topic ADAGRAD should find at convergence: $\lambda_{kv}^* \approx \mathbb{E}\left[\hat{\lambda}_{kv}^{(t)}\right]$. By definition, once the algorithm converges, λ_{kv}^* for top words will have very large values while λ_{kv}^* for bottom words will be small. After using noisy momentum terms, it must overcome initial faulty signals.

We now show the lower and upper bounds of $\mathbb{E}\left[\hat{\lambda}_{kv}^{(t)}\right]$ to show how big of an uphill battle ADAGRAD faces. Expanding the update rule,

$$\begin{aligned}\mathbb{E}\left[\hat{\lambda}_{kv}^{(t)}\right] &= \mathbb{E}\left[\eta + D\phi_{vk}^{d} n_{dv}\right] \\ &= \eta + D\bar{n}_v \mathbb{E}\left[\phi_{vk}\right],\end{aligned}$$

where $\bar{n}_v = \sum_{i=1}^{D} n_{iv}/D$, and ϕ_{vk} is the probability that word v is assigned to topic k. For a bottom word, $\phi_{vk} \to 0$. For a top word, $\phi_{vk} \geq 1/K$. After convergence, for a bottom word $\mathbb{E}\left[\phi_{vk}\right] \approx \eta$. For a top word, $1/K \leq \mathbb{E}\left[\phi_{vk}\right] \leq 1$. Thus, the lower and upper bounds of $\mathbb{E}\left[\hat{\lambda}_{kv}^{(t)}\right]$ are

$$\eta + (1/K)D\bar{n}_v \leq \mathbb{E}\left[\hat{\lambda}_{kv}^{(t)}\right] \leq \eta + D\bar{n}_v.$$

For a large datasets, $D\bar{n}_v$ should be large. Thus for top words, λ_{kv}^* will converge to a large value: quite a large hill to climb.

How quickly the algorithm climbs the hill is inversely proportional to the gradient size. We next show that the magnitude of gradients of top words are very large before the algorithm converges. Let g^* be the gradient after convergence. We show the bounds of $|g_{kv}|$, where $|.|$ is the absolute value, in the following:

$$\begin{aligned}|\,g_{kv}^*\,| &= |-\lambda_{kv}^* + \eta + D\phi_{vk}^{d} n_{dv}\,| \\ &\approx |-\eta - D\bar{n}_v E[\phi_{vk}] + \eta + D\phi_{vk}^{d} n_{dv}\,| \\ &\approx \mathbb{E}\left[\phi_{vk}\right] * D\,|\,n_{dv} - \bar{n}_v\,|\,.\end{aligned}$$

Thus,

$$(D/K)\,|\,n_{dv} - \bar{n}_v\,| \leq |\,g_{kv}^*\,| \leq D\,|\,n_{dv} - \bar{n}_v\,|\,.$$

Only when $n_{dv} = \bar{n}_v$, does $|\,g_{kv}^{(t)}\,| = 0$. Otherwise, due to the large D, $|\,g_{kv}^*\,|$ will be large. However, in practice, n_{dv} varies largely from document to document, which leads to large values of $|\,g_{kv}^*\,|$. Based on the gradient's property, when λ_{kv} is far away from the optimum, $|\,g_{kv}^{(t)}\,| \geq |\,g_{kv}^*\,|$. Thus, the values of $|\,g_{kv}^{(t)}\,|$ for the top words are very large before convergence.

ADAGRAD uses the accumulations of previous gradients as learning rates' denominators. Because of these large gradients in the first several iterations, learning rates soon decrease to small values; even if a topic has gathered a few words, ADAGRAD lacks the momentum to move other words into the topic. These small learning rates slows the updates of λ.

In sum, the initial gradient signals confuse the algorithm, the gradients are large enough to impede progress later, and large datasets imply a very large hill the algorithm must climb. Since the update progresses slowly, online LDA needs more iterations to break the equilibrium. Because the gradients of all words are still very large, the learning rates decrease quickly, which makes the update progress slower. When the update progresses more slowly, online LDA needs more iterations to break the tie. This cycle repeats, until some learning rates decrease to zero and learning effectively stops. Thus, the algorithm will never break the tie or infer good topics. Figure 1 illustrates the problem of online LDA with ADAGRAD.

1.4 Alternative Solutions

ADADELTA (Zeiler, 2012) and ADAM (Kingma and Ba, 2014) are extensions to ADAGRAD. ADADELTA does not have guaranteed convergence on convex optimization problems. Even though ADAM has a theoretical bound on its convergence rate, it is controlled by and sensitive to several learning rate parameters. For good performance with ADAM, manual adjustment is necessary. In addition, since ADADELTA computes the moving average of updates, and ADAM needs to compute the bias-corrected gradient estimate, they require more intricate implementations. Consequently, these two methods are not as popular as ADAGRAD for beginners. However, for SVI latent variable models, they can address the problems with ADAGRAD.

ADADELTA updates the learning rates with the following rule:

$$\rho_{kv}^{(t)} = \frac{\sqrt{\mathbb{E}\left[(\lambda_{kv}^{(t)} - \lambda_{kv}^{(t-1)})\right] + \varepsilon}}{\sqrt{\mathbb{E}\left[g_{kv}^{(t)}\right] + \varepsilon}}, \qquad (5)$$

where $\mathbb{E}\left[x^{(t)}\right] = \rho_0 \mathbb{E}\left[x^{(t-1)}\right] + (1-\rho_0)(x^{(t)})^2$, ρ_0 is a decay constant, and ε is for numerical stability.

ADAM's update rule is determined based on estimates of first and second moments of the gradients:

$$\begin{aligned} m_{kv}^{(t)} &= b_m m_{kv}^{(t-1)} + (1-b_m) g_{kv}^{(t)}, \\ u_{kv}^{(t)} &= b_u u_{kv}^{(t-1)} + (1-b_u)(g_{kv}^{(t)})^2, \\ \hat{m}_{kv}^{(t)} &= \frac{m_{kv}^{(t)}}{1-b_m^t}, \hat{u}_{kv}^{(t)} = \frac{u_{kv}^{(t)}}{1-b_u^t}, \\ \lambda_{kv}^{(t+1)} &= \lambda_{kv}^{(t)} + \rho_0 \hat{m}_{kv}^{(t)} / (\sqrt{\hat{u}_{kv}^{(t)}} + \varepsilon), \quad (6) \end{aligned}$$

where ρ_0 is a constant, b controls the decay rate.

Both ADADELTA and ADAM use the moving average of gradients as the denominator of learning rates. The learning rates will not monotonically decrease, but vary in a certain range. This property prevents online topic models from being trapped and breaks the tie between top words and bottom topic words. ADAM in particular uses bias-corrected estimate of gradient $\hat{m}_{kv}$, rather than the original stochastic gradient g_{kv} to guide direction for the optimization and therefore achieves better results.

In addition, the magnitude of gradients is proportional to the dataset's size. Thus, when the dataset is small enough, ADAGRAD will still work.

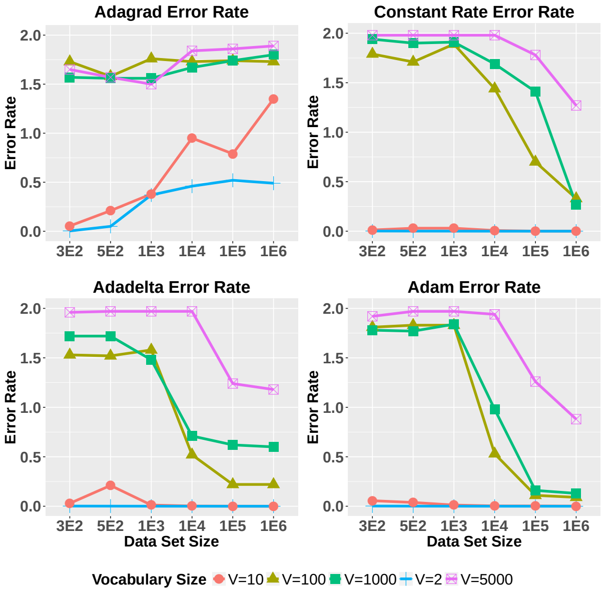


Figure 2: Experimental results on synthetic data sets. We vary the vocabulary size V, and the number of documents D. ADADELTA, ADAM and constant rate perform better with more data, while ADAGRAD only does well with small values of D.

2 Empirical Study

We study three datasets: **synthetic data**, **Wikipedia** and **SMS spam corpus**.[1] We use the generative process of LDA to generate synthetic data. We vary the vocabulary size $V \in \{2, 10, 100, 1000, 5000\}$, and the number of documents $D \in \{300, 500, 10^3, 10^4, 10^5, 10^6\}$. The **Wikipedia** dataset consists of 1M articles collected from Wikipedia.[2] The vocabulary is the same as (Hoffman et al., 2010). The SMS corpus is a small corpus containing 1084 documents.

2.1 Metrics and Settings

Error rate: For experiments on synthetic data set, we use error rate

$$\text{Error}(\hat{\beta}) = \frac{1}{K} \sum\nolimits_{k=1}^{K} \min_i ||\hat{\beta}_i - \beta_k||_1 \qquad (7)$$

to measure the difference between the estimated $\hat{\beta}$ and the known β. The min greedily matches each $\hat{\beta}_k$ to its best fit. While an uncommon metric for unsupervised algorithms, on the synthetic data we have the true β.

[1] http://www.esp.uem.es/jmgomez/smsspamcorpus/
[2] http://www.wikipedia.org

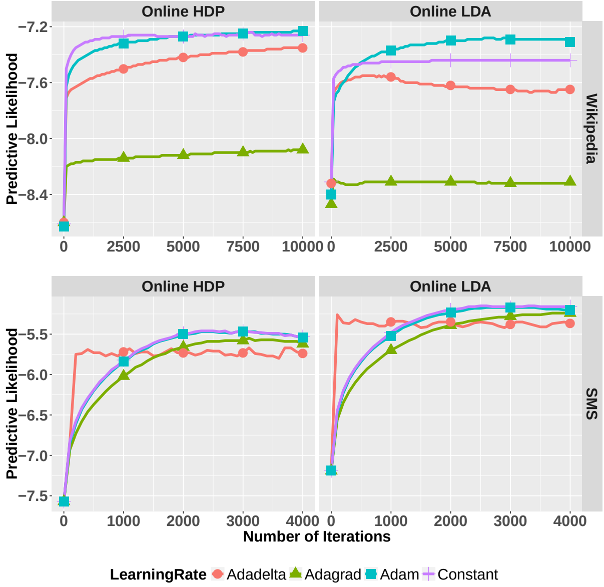


Figure 3: Experimental results on real corpora. Larger predictive likelihood is better. On Wikipedia, ADAGRAD has does worse than other methods. On SMS corpus, ADAGRAD is competitive.

Predictive likelihood: For experiments on real data sets, we use per-word likelihood (Hoffman et al., 2013) to evaluate the model quality. We randomly hold out 10K documents and 100 documents on Wikipedia and SMS respectively.

Settings: In the experiments on synthetic data, we use online LDA (Hoffman et al., 2010), since the data is generated by LDA. In the experiments on real datasets, we use online LDA and online HDP (Wang et al., 2011). In the experiments on Wikipedia, we set the number of topics $K = 100$ and the mini-batch size $M = 100$. In the experiments on SMS corpus, we set $K = 10$ and $M = 20$. For ADAM, we use the default setting of b, and set $\rho_0 = 10$ and $\epsilon = 1000$. For ADADELTA, we set $\epsilon = 1000$. For ADAGRAD, we set $\rho_0 = \epsilon = 1$. These are best settings for these three methods. The best constant rate is 10^{-3}.

2.2 Experimental Results

Figure 2 illustrates the experimental results on synthetic datasets. ADAGRAD only works well with small datasets. When the number of documents increases, ADAGRAD performance degrades. Conversely, other methods can handle more documents.

Figure 3 illustrates experimental results on real corpora. ADAGRAD gets competitive results to the other algorithms on the small SMS corpus. However on very large Wikipedia corpus, ADAGRAD fails to infer good topics, and its predictive ability is worse than the other methods. While ADADELTA and ADAM work well on Wikipedia, ADAM is the clear winner between the two.

3 Conclusion

ADAGRAD is a simple and popular technique for online learning, but is not compatible with traditional initializations and objective functions for online topic models. We show that practitioners are best off using simpler online learning techniques or ADADELTA and ADAM, which are two variants of ADAGRAD, which use the moving average of gradients as denominator. These two methods avoid ADAGRAD's problem. In particular, ADAM performs much better for prediction.

We would like to build a deeper understanding of which aspects of an unsupervised objective, near-uniform initialization, and non-identifiability contribute to these issues and to discover other learning problems that may share these issues.

Acknowledgments

We thank the anonymous reviewers, Stephan Mandt, Alp Kucukelbir, Bill Foland, Forough Poursabzi-Sangdeh and Alvin Grissom II for their insightful comments. Boyd-Graber and Lu's contribution is supported by NSF grants NCSE-1422492 and IIS-1409287, (UMD). Boyd-Graber is also supported by IIS-1564275 and IIS-1652666. Lund is supported by collaborative NSF Grant IIS-1409739 (BYU). Any opinions, findings, results, or recommendations expressed here are of the authors and do not necessarily reflect the view of the sponsor.

References

Pierre Bayle. 1826. *An historical and critical dictionary, selected and abridged*. Number 1 in An historical and critical dictionary, selected and abridged. https://books.google.com/books?id=cDsN3xOyO-oC.

David M Blei. 2012. Probabilistic topic models. *Communications of the ACM* 55(4):77–84.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. 2016. Variational inference: A review for statisticians. *arXiv preprint arXiv:1601.00670* .

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Michael Bryant and Erik B Sudderth. 2012. Truly nonparametric online variational inference for hierarchical Dirichlet processes. In *Proceedings of Advances in Neural Information Processing Systems*.

Jonathan Chang and David M Blei. 2009. Relational topic models for document networks. In *Proceedings of Artificial Intelligence and Statistics*. volume 9, pages 81–88.

Wang Chong, David Blei, and Fei-Fei Li. 2009. Simultaneous image classification and annotation. In *Computer Vision and Pattern Recognition*. IEEE.

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

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. 2012. Large scale distributed deep networks. In *Proceedings of Advances in Neural Information Processing Systems*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159.

John Duchi and Yoram Singer. 2009. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research* 10:2899–2934.

Matthew Hoffman, Francis R Bach, and David M Blei. 2010. Online learning for latent Dirichlet allocation. In *Proceedings of Advances in Neural Information Processing Systems*.

Matthew D Hoffman, David M Blei, Chong Wang, and John William Paisley. 2013. Stochastic variational inference. *Journal of Machine Learning Research* 14:1303–1347.

Yuening Hu, Ke Zhai, Vlad Eidelman, and Jordan Boyd-Graber. 2014. Polylingual tree-based topic models for translation domain adaptation. In *Proceedings of the Association for Computational Linguistics*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

John Paisley, Chong Wang, David M Blei, and Michael I Jordan. 2015. Nested hierarchical Dirichlet processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37:256–270.

Rajesh Ranganath, Chong Wang, David M Blei, and Eric P Xing. 2013. An adaptive learning rate for stochastic variational inference. In *Proceedings of the International Conference of Machine Learning*.

Chong Wang, John William Paisley, and David M Blei. 2011. Online variational inference for the hierarchical Dirichlet process. In *Proceedings of Artificial Intelligence and Statistics*.

Lin Xiao. 2010. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research* 11:2543–2596.

Weiwei Yang, Jordan Boyd-Graber, and Philip Resnik. 2016. A discriminative topic model using document network structure. In *Association for Computational Linguistics*.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv: 1212.5701* .

Recurrent Attention Network on Memory for Aspect Sentiment Analysis

Peng Chen **Zhongqian Sun** **Lidong Bing**[*] **Wei Yang**
AI Lab
Tencent Inc.
{patchen, sallensun, lyndonbing, willyang}@tencent.com

Abstract

We propose a novel framework based on neural networks to identify the sentiment of opinion targets in a comment/review. Our framework adopts multiple-attention mechanism to capture sentiment features separated by a long distance, so that it is more robust against irrelevant information. The results of multiple attentions are non-linearly combined with a recurrent neural network, which strengthens the expressive power of our model for handling more complications. The weighted-memory mechanism not only helps us avoid the labor-intensive feature engineering work, but also provides a tailor-made memory for different opinion targets of a sentence. We examine the merit of our model on four datasets: two are from SemEval2014, i.e. reviews of restaurants and laptops; a twitter dataset, for testing its performance on social media data; and a Chinese news comment dataset, for testing its language sensitivity. The experimental results show that our model consistently outperforms the state-of-the-art methods on different types of data.

1 Introduction

The goal of aspect sentiment analysis is to identify the sentiment polarity (i.e., negative, neutral, or positive) of a specific opinion target expressed in a comment/review by a reviewer. For example, in “I bought a mobile phone, its camera is wonderful but the battery life is short”, there are three opinion targets, “camera”, “battery life”, and “mobile phone”. The reviewer has a positive sentiment on the “camera”, a negative sentiment on the “battery life”, and a mixed sentiment on the “mobile phone”. Sentence-oriented sentiment analysis methods (Socher et al., 2011; Appel et al., 2016) are not capable to capture such fine-grained sentiments on opinion targets.

In order to identify the sentiment of an individual opinion target, one critical task is to model appropriate context features for the target in its original sentence. In simple cases, the sentiment of a target is identifiable with a syntactically nearby opinion word, e.g. “wonderful” for “camera”. However, there are many cases in which opinion words are enclosed in more complicated contexts. E.g., “Its camera is not wonderful enough” might express a neutral sentiment on “camera”, but not negative. Such complications usually hinder conventional approaches to aspect sentiment analysis.

To model the sentiment of the above phrase-like word sequence (i.e. “not wonderful enough”), LSTM-based methods are proposed, such as target dependent LSTM (TD-LSTM) (Tang et al., 2015). TD-LSTM might suffer from the problem that after it captures a sentiment feature far from the target, it needs to propagate the feature word by word to the target, in which case it's likely to lose this feature, such as the feature “cost-effective” for “the phone” in “My overall feeling is that the phone, after using it for three months and considering its price, is really cost-effective”.[1] Attention mechanism, which has been successfully used in machine translation (Bahdanau et al., 2014), can enforce a model to pay more attention to the important part of a sentence. There are already some works using attention in sentiment analysis to exploit this advantage (Wang et al., 2016; Tang et al., 2016). Another observation is that some types of

[*] Corresponding author.

[1] Although LSTM could keep information for a long distance by preventing the vanishing gradient problem, it usually requires a large training corpus to capture the flexible usage of parenthesis.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 452–461
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

sentence structures are particularly challenging for target sentiment analysis. For example, in "Except Patrick, all other actors don't play well", the word "except" and the phrase "don't play well" produce a positive sentiment on "Patrick". It's hard to synthesize these features just by LSTM, since their positions are dispersed. Single attention based methods (e.g. (Wang et al., 2016)) are also not capable to overcome such difficulty, because attending multiple words with one attention may hide the characteristic of each attended word.

In this paper, we propose a novel framework to solve the above problems in target sentiment analysis. Specifically, our framework first adopts a bidirectional LSTM (BLSTM) to produce the memory (i.e. the states of time steps generated by LSTM) from the input, as bidirectional recurrent neural networks (RNNs) were found effective for a similar purpose in machine translation (Bahdanau et al., 2014). The memory slices are then weighted according to their relative positions to the target, so that different targets from the same sentence have their own tailor-made memories. After that, we pay multiple attentions on the position-weighted memory and nonlinearly combine the attention results with a recurrent network, i.e. GRUs. Finally, we apply softmax on the output of the GRU network to predict the sentiment on the target.

Our framework introduces a novel way of applying multiple-attention mechanism to synthesize important features in difficult sentence structures. It's sort of analogous to the cognition procedure of a person, who might first notice part of the important information at the beginning, then notices more as she reads through, and finally combines the information from multiple attentions to draw a conclusion. For the above sentence, our model may attend the word "except" first, and then attends the phrase "don't play well", finally combines them to generate a positive feature for "Patrick". Tang et al. (2016) also adopted the idea of multiple attentions, but they used the result of a previous attention to help the next attention attend more accurate information. Their vector fed to softmax for classification is only from the final attention, which is essentially a linear combination of input embeddings (they did not have a memory component). Thus, the above limitation of single attention based methods also holds for (Tang et al., 2016). In contrast, our model combines the results of multiple attentions with a GRU network, which has different behaviors inherited from RNNs, such as forgetting, maintaining, and non-linearly transforming, and thus allows a better prediction accuracy.

We evaluate our approach on four datasets: the first two come from SemEval 2014 (Pontiki et al., 2014), containing reviews of restaurant domain and laptop domain; the third one is a collection of tweets, collected by (Dong et al., 2014); to examine whether our framework is language-insensitive (since languages show differences in quite a few aspects in expressing sentiments), we prepared a dataset of Chinese news comments with people mentions as opinion targets. The experimental results show that our model performs well for different types of data, and consistently outperforms the state-of-the-art methods.

2 Related Work

The task of aspect sentiment classification belongs to entity-level sentiment analysis. Conventional representative methods for this task include rule-based methods (Ding et al., 2008) and statistic-based methods (Jiang et al., 2011; Zhao et al., 2010). Ganapathibhotla and Liu (2008) extracted 2-tuples of (opinion target, opinion word) from comments and then identified the sentiment of opinion targets. Deng and Wiebe (2015) adopted Probabilistic Soft Logic to handle the task. There are also statistic-based approaches which employ SVM (Jiang et al., 2011) or MaxEnt-LDA (Zhao et al., 2010). These methods need either laborious feature engineering work or massive extra-linguistic resources.

Neural Networks (NNs) have the capability of fusing original features to generate new representations through multiple hidden layers. Recursive NN (Rec-NN) can conduct semantic compositions on tree structures, which has been used for syntactic analysis (Socher et al., 2010) and sentence sentiment analysis (Socher et al., 2013). (Dong et al., 2014; Nguyen and Shirai, 2015) adopted Rec-NN for aspect sentiment classification, by converting the opinion target as the tree root and propagating the sentiment of targets depending on the context and syntactic relationships between them. However, Rec-NN needs dependency parsing which is likely ineffective on nonstandard texts such as news comments and tweets. (Chen et al., 2016) employed Convolution NNs to identify the senti-

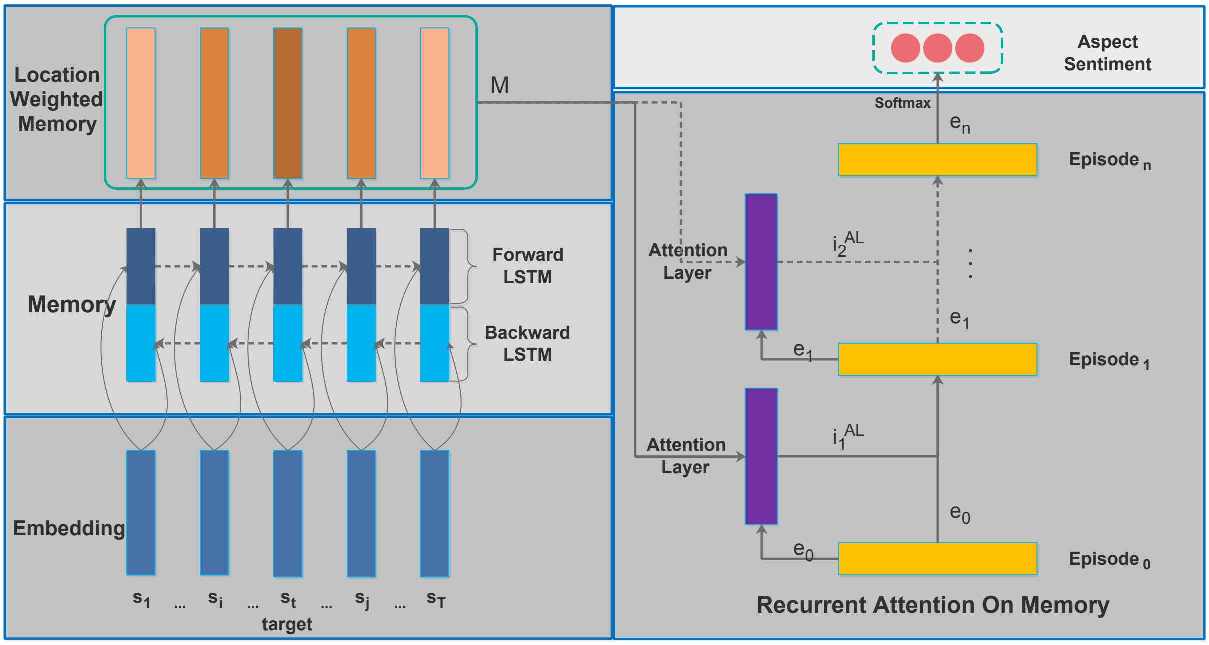


Figure 1: Model architecture. The dotted lines on the right indicate a layer may or may not be added.

ment of a clause which is then used to infer the sentiment of the target. The method has an assumption that an opinion word and its target lie in the same clause. TD-LSTM (Tang et al., 2015) utilizes LSTM to model the context information of a target by placing the target in the middle and propagating the state word by word from the beginning and the tail to the target respectively to capture the information before and after it. Nevertheless, TD-LSTM might not work well when the opinion word is far from the target, because the captured feature is likely to be lost ((Cho et al., 2014) reported similar problems of LSTM-based models in machine translation).

(Graves et al., 2014) introduced the concept of memory for NNs and proposed a differentiable process to read and write memory, which is called Neural Turing Machine (NTM). Attention mechanism, which has been used successfully in many areas (Bahdanau et al., 2014; Rush et al., 2015), can be treated as a simplified version of NTM because the size of memory is unlimited and we only need to read from it. Single attention or multiple attentions were applied in aspect sentiment classification in some previous works (Wang et al., 2016; Tang et al., 2016). One difference between our method and (Tang et al., 2016) is that we introduce a memory module between the attention module and the input module, thus our method can synthesize features of word sequences such as sentiment phrases (e.g. “not wonderful enough”). More importantly, we combine the results of attentions in a nonlinear way. (Wang et al., 2016) only uses one attention, while our model uses multiple attentions. The effectiveness of multiple attentions was also investigated in QA task (Kumar et al., 2015), which shows that multiple attentions allow a model to attend different parts of the input during each pass. (Kumar et al., 2015) assigns attention scores to memory slices independently and their attention process is more complex, while we produce a normalized attention distribution to attend information from the memory.

3 Our Model

The architecture of our model is shown in Figure 1, which consists of five modules: input module, memory module, position-weighted memory module, recurrent attention module, and output module. Suppose the input sentence is $\mathbf{s} = \{s_1, \ldots, s_{\tau-1}, s_\tau, s_{\tau+1}, \ldots, s_T\}$, the goal of our model is to predict the sentiment polarity of the target s_τ. For simplicity, we notate a target as one word here, where necessary, we will elaborate how to handle phrase-form targets, e.g. “battery life”.

3.1 Input Embedding

Let $\mathbb{L} \in \mathbb{R}^{d \times |V|}$ be an embedding lookup table generated by an unsupervised method such as GloVe (Pennington et al., 2014) or CBOW

(Mikolov et al., 2013), where d is the dimension of word vectors and $|V|$ is the vocabulary size. The input module retrieves the word vectors from $\mathbb{L}$ for an input sequence and gets a list of vectors $\{v_1, \ldots, v_t, \ldots, v_T\}$ where $v_t \in \mathbb{R}^d$. $\mathbb{L}$ may or may not be tuned in the training of our framework. If it is not tuned, the model can utilize the words' similarity revealed in the original embedding space. If it is tuned, we expect the model would capture some intrinsic information that is useful for the sentiment analysis task.

3.2 BLSTM for Memory Building

MemNet (Tang et al., 2016) simply used the sequence of word vectors as memory, which cannot synthesize phrase-like features in the original sentence. It is straightforward to achieve the goal with the models of RNN family. In this paper, we use Deep Bidirectional LSTM (DBLSTM) to build the memory which records all information to be read in the subsequent modules.

At each time step t, the forward LSTM not only outputs the hidden state $\overrightarrow{h}_t^l$ at its layer l ($\overrightarrow{h}_t^0 = v_t$) but also maintains a memory $\overrightarrow{c}_t^l$ inside its hidden cell. The update process at time t is as follows:

$$i = \sigma(\overrightarrow{W}_i \overrightarrow{h}_t^{l-1} + \overrightarrow{U}_i \overrightarrow{h}_{t-1}^l) \tag{1}$$

$$f = \sigma(\overrightarrow{W}_f \overrightarrow{h}_t^{l-1} + \overrightarrow{U}_f \overrightarrow{h}_{t-1}^l) \tag{2}$$

$$o = \sigma(\overrightarrow{W}_o \overrightarrow{h}_t^{l-1} + \overrightarrow{U}_o \overrightarrow{h}_{t-1}^l) \tag{3}$$

$$g = \tanh(\overrightarrow{W}_g \overrightarrow{h}_t^{l-1} + \overrightarrow{U}_g \overrightarrow{h}_{t-1}^l) \tag{4}$$

$$\overrightarrow{c}_t^l = f \odot \overrightarrow{c}_{t-1}^l + i \odot g \tag{5}$$

$$\overrightarrow{h}_t^l = o \odot \tanh(\overrightarrow{c}_t^l) \tag{6}$$

where σ and $\tanh$ are sigmoid and hyperbolic tangent functions, $\overrightarrow{W}_i, \overrightarrow{W}_f, \overrightarrow{W}_o, \overrightarrow{W}_g \in \mathbb{R}^{\overrightarrow{d}_l \times \overrightarrow{d}_{l-1}}$, $\overrightarrow{U}_i, \overrightarrow{U}_f, \overrightarrow{U}_o, \overrightarrow{U}_g \in \mathbb{R}^{\overrightarrow{d}_l \times \overrightarrow{d}_l}$, and $\overrightarrow{d}_l$ is the number of hidden cells at the layer l of the forward LSTM. The gates $i, f, o \in \mathbb{R}^{\overrightarrow{d}_l}$ simulate binary switches that control whether to update the information from the current input, whether to forget the information in the memory cells, and whether to reveal the information in memory cells to the output, respectively. The backward LSTM does the same thing, except that its input sequence is reversed. If there are L layers stacked in the BLSTM, the final memory generated in this module is $M^* = \{m_1^*, \ldots, m_t^*, \ldots, m_T^*\}$, where $m_t^* = (\overrightarrow{h}_t^L, \overleftarrow{h}_t^L) \in \mathbb{R}^{\overrightarrow{d}_L + \overleftarrow{d}_L}$. In our framework, we use 2 layers of BLSTM to build the memory, as it generally performs well in NLP tasks (Karpathy et al., 2015).

3.3 Position-Weighted Memory

The memory generated in the above module is the same for multiple targets in one comment, which is not flexible enough for predicting respective sentiments of these targets. To ease this problem, we adopt an intuitive method to edit the memory to produce a tailor-made input memory for each target. Specifically, the closer to the target a word is, the higher its memory slide is weighted. We define the distance as the number of words between the word and the target. One might want to use the length of the path from the specific word to the target in the dependency tree as the distance, which is a worthwhile option to try in the future work, given the condition that dependency parsing on the input text is effective enough. Precisely, the weight for the word at position t is calculated as:

$$w_t = 1 - \frac{|t - \tau|}{t_{max}} \tag{7}$$

where t_{max} is truncation length of the input. We also calculate $u_t = \frac{t-\tau}{t_{max}}$ to memorize the relative offset between each word and the target. If the target is a phrase, the distance (i.e. $t - \tau$) is calculated with its left or right boundary index according to which side w_t locates. The final position-weighted memory of a target is $M = \{m_1, \ldots, m_t, \ldots, m_T\}$ where $m_t = (w_t \cdot m_t^*, u_t) \in \mathbb{R}^{\overrightarrow{d}_L + \overleftarrow{d}_L + 1}$. The weighted memory is designed to up-weight nearer sentiment words, and the recurrent attention module, discussed below, attends long-distance sentiment words. Thus, they work together to expect a better prediction accuracy.

3.4 Recurrent Attention on Memory

To accurately predict the sentiment of a target, it is essential to: (1) correctly distill the related information from its position-weighted memory; and (2) appropriately manufacture such information as the input of sentiment classification. We employ multiple attentions to fulfil the first aspect, and a recurrent network for the second which non-linearly combines the attention results with GRUs (since GRUs have less number of parameters). For example, "except" and "don't play well" in "Except Patrick, all other actors don't play well" are attended by different attentions, and combined to produce a positive sentiment on "Patrick".

Particularly, we employ a GRU to update the episode e after each attention. Let e_{t-1} denote the episode at the previous time and i_t^{AL} is the current information attended from the memory M, and the process of updating e_t is as follows:

$$r = \sigma(W_r i_t^{AL} + U_r e_{t-1}) \tag{8}$$

$$z = \sigma(W_z i_t^{AL} + U_z e_{t-1}) \tag{9}$$

$$\tilde{e}_t = \tanh(W_x i_t^{AL} + W_g(r \odot e_{t-1})) \tag{10}$$

$$e_t = (1-z) \odot e_{t-1} + z \odot \tilde{e}_t \tag{11}$$

where $W_r, W_z \in \mathbb{R}^{H\times(\overrightarrow{d}_L+\overleftarrow{d}_L+1)}$, $U_r, U_z \in \mathbb{R}^{H\times H}$, $W_g \in \mathbb{R}^{H\times(\overrightarrow{d}_L+\overleftarrow{d}_L+1)}$, $W_x \in \mathbb{R}^{H\times H}$, and H is the hidden size of GRU. As we can see from Equations (10) and (11), the state of episode e_t is the interpolation of e_{t-1} and the candidate hidden vector $\tilde{e}_t$. A vector of 0's is used as e_0.

For calculating the attended information i_t^{AL} at t, the input of an attention layer (AL for short) includes the memory slices $m_j (1 \leq j \leq T)$ and the previous episode e_{t-1}. We first calculate the attention score of each memory slice as follows:

$$g_j^t = W_t^{AL}(m_j, e_{t-1}[, v_\tau]) + b_t^{AL}, \tag{12}$$

where $[, v_\tau]$ indicates when the attention result relies on particular aspects such as those of products, we also add the target vector v_τ because different product aspects have different preference on opinion words; when the target is a person, there is no need to do so. If the target is a phrase, v_τ takes the average of word embeddings. We utilize the previous episode for the current attention, since it can guide the model to attend different useful information. (Tang et al., 2016) also adopts multiple attentions, but they don't combine the results of different attentions.

Then we calculate the normalized attention score of each memory slice as:

$$\alpha_j^t = \frac{\exp(g_j^t)}{\sum_k \exp(g_k^t)}. \tag{13}$$

Finally, the inputs to a GRU (i.e. Eqs. 8 to 11) at time t are the episode e_{t-1} at time $t-1$ and the content i_t^{AL}, which is read from the memory as:

$$i_t^{AL} = \sum_{j=1}^{T} \alpha_j^t m_j. \tag{14}$$

3.5 Output and Model Training

After N-time attentions on the memory, the final episode e_N serves as the feature and is fed into a softmax layer to predict the target sentiment.

The model is trained by minimizing the cross entropy plus an L_2 regularization term:

$$\mathcal{L} = \sum_{(x,y)\in D} \sum_{c\in C} y^c \log f^c(x;\theta) + \lambda \parallel \theta \parallel^2 \tag{15}$$

where C is the sentiment category set, D is the collection of training data, $y \in \mathbb{R}^{|C|}$ is a one-hot vector where the element for the true sentiment is 1, $f(x;\theta)$ is the predicted sentiment distribution of the model, λ is the weight of L_2 regularization term. We also adopt dropout and early stopping to ease overfitting.

4 Experiments

4.1 Experimental Setting

We conduct experiments on four datasets, as shown in Table 1. The first two are from SemEval 2014 (Pontiki et al., 2014), containing reviews of restaurant and laptop domains, which are widely used in previous works. The third one is a collection of tweets, collected by (Dong et al., 2014). The last one is prepared by us for testing the language sensitivity of our model, which contains Chinese news comments and has politicians and entertainers as opinion targets. We purposely add more negation, contrastive, and question comments to make it more challenging. Each comment is annotated by at least two annotators, and only if they agree with each other, the comment will be added into our dataset. Moreover, we replace each opinion target (i.e. word/phrase of pronoun or person name) with a placeholder, as did in (Dong et al., 2014). For the first two datasets, we removed a few examples having the "conflict label", e.g., "Certainly not the best sushi in New York, however, it is always fresh" (Pontiki et al., 2014).

We use 300-dimension word vectors pre-trained by GloVe (Pennington et al., 2014) (whose vocabulary size is 1.9M[2]) for our experiments on the English datasets, as previous works did (Tang et al., 2016). Some works employed domain-specific training corpus to learn embeddings for better performance, such as TD-LSTM (Tang et al., 2015) on the tweet dataset. In contrast, we prefer to use

[2] http://nlp.stanford.edu/projects/glove/

Dataset		Negative	Neutral	Positive
Laptop reviews	Training	858	454	980
	Testing	128	171	340
Restaurant reviews	Training	800	632	2,159
	Testing	195	196	730
Tweets	Training	1,563	3,127	1,567
	Testing	174	346	174
News comments	Training	6,001	8,403	5,633
	Testing	838	1,054	732

Table 1: Details of the experimental datasets.

the general embeddings from (Pennington et al., 2014) for all datasets, so that the experimental results can better reveal the model's capability and the figures are directly comparable across different papers. The embeddings for Chinese experiments are trained with a corpus of 1.4 billion tokens with CBOW[3].

4.2 Compared Methods

We compare our proposed framework of Recurrent Attention on Memory (**RAM**) with the following methods:

- Average Context: There are two versions of this method. The first one, named **AC-S**, averages the word vectors before the target and the word vectors after the target separately. The second one, named **AC**, averages the word vectors of the full context.
- SVM (Kiritchenko et al., 2014): The traditional state-of-the-art method using SVMs on surface features, lexicon features and parsing features, which is the best team in SemEval 2014.
- Rec-NN (Dong et al., 2014): It firstly uses rules to transform the dependency tree and put the opinion target at the root, and then performs semantic composition with Recursive NNs for sentiment prediction.
- TD-LSTM (Tang et al., 2015): It uses a forward LSTM and a backward LSTM to abstract the information before and after the target. Finally, it takes the hidden states of LSTM at last time step to represent the context for prediction. We reproduce its results on the tweet dataset with our embeddings, and also run it for the other three datasets.
- TD-LSTM-A: We developed TD-LSTM to make it have one attention on the outputs of forward and backward LSTMs, respectively.
- MemNet (Tang et al., 2016): It applies attention multiple times on the word embeddings, and the last attention's output is fed to softmax for prediction, without combining the results of different attentions. We produce its results on all four datasets with the code released by the authors.[4]

For each method, the maximum number of training iterations is 100, and the model with the minimum training error is utilized for testing. We will discuss different settings of RAM later.

4.3 Main Results

The first evaluation metric is Accuracy, which is used in (Tang et al., 2016). Because the datasets have unbalanced classes as shown in Table 1, Macro-averaged F-measure is also reported, as did in (Dong et al., 2014; Tang et al., 2015). As shown by the results in Table 2, our RAM consistently outperforms all compared methods on these four datasets. AC and AC-S perform poorly, because averaging context is equivalent to paying identical attention to each word which would hide the true sentiment word. Rec-NN is better than TD-LSTM but not as good as our method. The advantage of Rec-NN is that it utilizes the result of dependency parsing which might shorten the distance between the opinion target and the related opinion word. However, dependency parsing is not guaranteed to work well on irregular texts such as tweets, which may still result in long path between the opinion word and its target, so that the opinion features would also be lost while being propagated. TD-LSTM performs less competitive than our method on all the datasets, particularly on the tweet dataset, because in this dataset sentiment words are usually far from person names,

[3] https://github.com/svn2github/word2vec

[4] http://ir.hit.edu.cn/∼dytang

Method	Laptop		Restaurant		Tweet		Comments	
	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1
AC	0.6729	0.6186	0.7504	0.6396	0.6228	0.5912	0.6231	0.6182
AC-S	0.6839	0.6217	0.7585	0.6379	0.6329	0.6009	0.6425	0.6376
SVM	0.7049*	NA	0.8016*	NA	0.6340♮	0.6330♮	0.6524	0.6499
Rec-NN	NA	NA	NA	NA	0.6630*	0.6590*	NA	NA
TD-LSTM	0.7183	0.6843	0.7800	0.6673	0.6662	0.6401	0.7275	0.7260
TD-LSTM-A	0.7214	0.6745	0.7889	0.6901	0.6647	0.6404	0.7206	0.7195
MemNet	0.7033	0.6409	0.7816	0.6583	0.6850	0.6691	0.6247	0.6117
RAM	**0.7449**	**0.7135**	**0.8023**	**0.7080**	**0.6936**	**0.6730**	**0.7389**	**0.7385**

Table 2: Main results. The results with '*' are retrieved from the papers of compared methods, and those with '♮' are retrieved from Rec-NN paper.

No. of AL	Laptop	Restaurant	Tweet	Comments
RAM-1AL	0.7074	0.7996	0.6864	0.7336
RAM-2AL	**0.7465**	0.7889	0.6922	0.7363
RAM-3AL	0.7449	0.8023	**0.6936**	**0.7389**
RAM-4AL	0.7293	**0.8059**	0.6879	0.7325
RAM-5AL	0.7293	0.7960	0.6864	0.7325

Table 3: The impacts of attention layers. (Word embeddings are not tuned in the training stage.)

for which case the multiple-attention mechanism is designed to work. TD-LSTM-A also performs worse than our method, because its two attentions, i.e. one for the text before the target and the other for the after, cannot tackle some cases where more than one features being attended are at the same side of the target.

Our method steadily performs better than MemNet on all four datasets, particularly on the News comment dataset, its improvement is more than 10%. MemNet adopts multiple attentions in order to improve the attention results, given the assumption that the result of an attention at a later hop should be better than that at the beginning. MemNet doesn't combine the results of multiple attentions, and the vector fed to softmax is the result of the last attention, which is essentially the linear combination of word embeddings. As we described before, attending too many words in one time may hide the characteristic of each word, moreover, the sentiment transition usually combines features in a nonlinear way. Our model overcomes this shortcoming with a GRU network to combine the results of multiple attentions. The feature-based SVM, which needs labor-intensive feature engineering works and a mass of extra linguistic resources, doesn't display its advantage, because the features for aspect sentiment analysis cannot be extracted as easily as for sentence or document level sentiment analysis.

4.4 Effects of Attention Layers

One major setting that affects the performance of our model is the number of attention layers. We evaluate our framework with 1 to 5 attention layers, and the results are given in Table 3, where NAL means using N attentions. In general, our model with 2 or 3 attention layers works better, but the advantage is not always there for different datasets. For example, for the Restaurant dataset, our model with 4 attention layers performs the best. Using 1 attention is always not as good as using more, which shows that one-time attention might not be sufficient to capture the sentiment features in complicated cases. One the other hand, the performance is not monotonically increasing with respect to the number of attentions. RAM-4AL is generally not as good as RAM-3AL, it is because as the model's complexity increases, the model becomes more difficult to train and less generalizable.

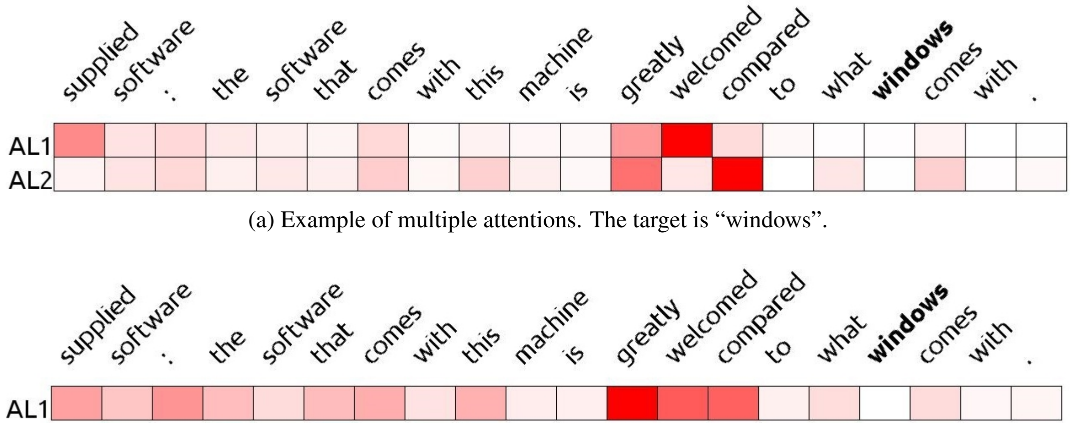
4.5 Effects of Embedding Tuning

The compared embedding tuning strategies are:

- RAM-3AL-T-R: It does not pre-train word embeddings, but initializes embeddings randomly and then tunes them in the supervised

Embedding	Laptop	Restaurant	Tweet	Comment
RAM-3AL-T-R	0.5806	0.7129	0.6272	0.6749
RAM-3AL-T	0.6854	0.7522	0.6402	0.7283
RAM-3AL-NT	**0.7449**	**0.8023**	**0.6936**	**0.7389**

Table 4: The impact of different embedding tuning strategies.



(a) Example of multiple attentions. The target is "windows".

(b) Example of single attention. The target is "windows".

Figure 2: Comparison of single attention and multiple attentions. Attention score by Eq. 13 is used as the color-coding.

training stage.

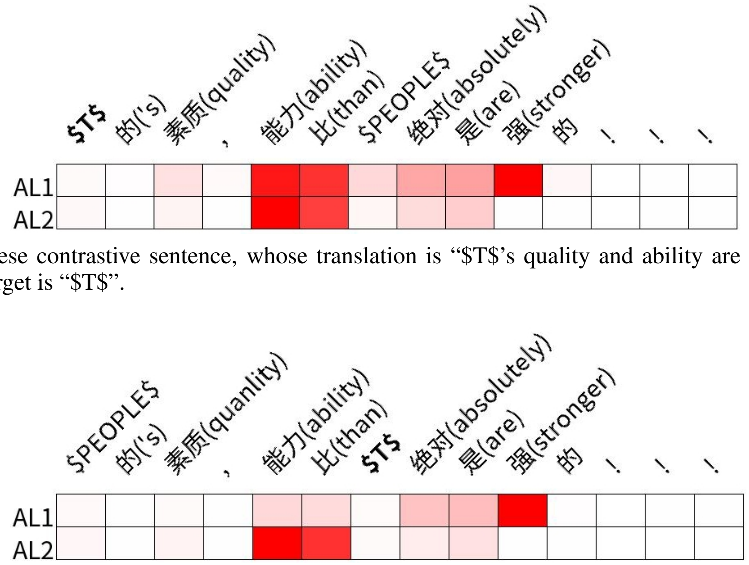
- RAM-3AL-T: Using the pre-trained embeddings initially, and they are also tuned in the training.
- RAM-3AL-NT: The pre-trained embeddings are not tuned in the training.

From Table 4, we can see that RAM-3AL-T-R performs very poorly, especially when the size of training data is smaller. The reason could be threefold: (1) The amount of labelled samples in the four experimental datasets is too small to tune reliable embeddings from scratch for the in-vocabulary words (i.e. existing in the training data); (2) A lot of out-of-vocabulary (OOV) words, i.e. absent from the training data, but exist in the testing data; (3) It increases the risk of overfitting after adding the embedding parameters to the solution space (it requires the embeddings not only to fit model parameters, but also to capture the similarity among words). During training, we indeed observed that the training error converges too fast in RAM-3AL-T-R. RAM-3AL-T can utilize the embedding similarity among words at the beginning of training, but fine tuning will destroy this similarity during training. On the other hand, the initial embeddings of OOV words in the testing data are not tuned, so that their similarity with vocabulary words are also destroyed. In addition, RAM-3AL-T also suffers from the risk of overfitting. RAM-3AL-NT performs the best on all four datasets, and we also observe that the training error converges gradually while the model parameters are being updated with the error signal from the output layer.

4.6 Case Study

We pick some testing examples from the datasets and visualize their attention results. To make the visualized results comprehensible, we remove the BLSTM memory module to make the attention module directly work on the word embeddings, thus we can check whether the attention results conform with our intuition. The visualization results are shown in Figures 2 and 3.

Figures 2a and 2b present the differences between using two attentions and using one attention, which show that multiple attentions are useful to attend correct features. As shown in Figure 2a, in order to identify the sentiment of "windows", the model firstly notices "welcomed" and secondly notices "compared" before the aspect target "windows". Finally it combines them with the GRU network, and generates a negative sentiment because the compared item (i.e. "windows") after a positive sentiment word (i.e. "welcomed") is less preferred. While the attention result of the model



(a) Example of a Chinese contrastive sentence, whose translation is "T's quality and ability are absolutely stronger than $PEOPLE$!!!". The target is "T".

(b) The sentence from 3a with a different target, i.e. "$PEOPLE$'s quality and ability are absolutely stronger than T!!!".

Figure 3: Example of multiple opinion targets. Attention score by Eq. 13 is used as the color-coding.

with only one attention, as shown in Figure 2a, is a sort of uniform distribution and mingles too many word vectors in a linear way, which would ruin the characteristic of each word.

Figures 3a and 3b present a case that there are more than one opinion targets in a comment, which cannot be analyzed with sentence-level sentiment analysis methods properly. Specifically, it's a comparative sentence in which the reviewer has a positive sentiment on the first commented person, but a negative sentiment on the second person, and our model predicts both of them correctly. Although all useful information (e.g. "than" and "stronger") is attended in both cases, the attention procedures of them show some interesting differences. They mainly attend important information after the target T in the first attention layer AL1. After that, Figure 3b attends more information before T in AL2. Since the same words in Figures 3a and 3b have different memory slices due to position weighting and augmented offset feature, as described in Section 3.3, our model predicts opposite sentiments on the two persons. For example in Figure 3b, the model first attends a positive word "stronger" and then attends "than" before the target, so it reverses the sentiment and finally predicts a negative sentiment.

5 Conclusions and Future Work

In this paper, we proposed a framework to identify the sentiment of opinion targets. The model first runs through the input to generate a memory, in the process of which it can synthesize the word sequence features. And then, the model pays multiple attentions on the memory to pick up important information to predict the final sentiment, by combining the features from different attentions non-linearly. We demonstrated the efficacy of our model on four datasets, and the results show that it can outperform the state-of-the-art methods.

Although multiple-attention mechanism has the potential to synthesize features in complicated sentences, enforcing the model to pay a fix number of attentions to the memory is unnatural and even sort of unreasonable for some cases. Therefore, we need a mechanism to stop the attention process automatically if no more useful information can be read from the memory. We may also try other memory weighting strategies to distinguish multiple targets in one comment more clearly.

References

Orestes Appel, Francisco Chiclana, Jenny Carter, and Hamido Fujita. 2016. A hybrid approach to the sentiment analysis problem at the sentence level. *Knowl.-Based Syst.*, 108:110–124.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Peng Chen, Bing Xu, Muyun Yang, and Sheng Li. 2016. Clause sentiment identification based on convolutional neural network with context embedding.

In *Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016 12th International Conference on*, pages 1532–1538.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Lingjia Deng and Janyce Wiebe. 2015. Joint prediction for entity/event-level sentiment analysis using probabilistic soft logic models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Xiaowen Ding, Bing Liu, and Philip S Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 international conference on web search and data mining*, pages 231–240.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. 2014. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Murthy Ganapathibhotla and Bing Liu. 2008. Mining opinions in comparative sentences. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 241–248.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *CoRR*, abs/1410.5401.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 151–160.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078.

Svetlana Kiritchenko, Xiaodan Zhu, Colin Cherry, and Saif Mohammad. 2014. Nrc-canada-2014: Detecting aspects and sentiment in customer reviews. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 437–442.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285.

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

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Phrasernn: Phrase recursive neural network for aspect-based sentiment analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 27–35.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685.

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011*, pages 151–161.

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

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Target-dependent sentiment classification with long short term memory. *CoRR*, abs/1512.01100.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Wayne Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. 2010. Jointly modeling aspects and opinions with a maxent-lda hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 56–65.

A Cognition Based Attention Model for Sentiment Analysis

Yunfei Long[1], **Qin Lu**[1], **Rong Xiang**[2], **Minglei Li**[1] and **Chu-Ren Huang**[3]
[1]Department of Computing, The Hong Kong Polytechnic University
csylong,csluqin,csmli@comp.polyu.edu.hk
[2]Advanced Micro Devices, Inc.(Shanghai)
Rong.Xiang@amd.com
[3]Department of Chinese and Bilingual Studies, The Hong Kong Polytechnic University
churen.huang@polyu.edu.hk

Abstract

Attention models are proposed in sentiment analysis because some words are more important than others. However, most existing methods either use local context based text information or user preference information. In this work, we propose a novel attention model trained by cognition grounded eye-tracking data. A reading prediction model is first built using eye-tracking data as dependent data and other features in the context as independent data. The predicted reading time is then used to build a cognition based attention (CBA) layer for neural sentiment analysis. As a comprehensive model, We can capture attentions of words in sentences as well as sentences in documents. Different attention mechanisms can also be incorporated to capture other aspects of attentions. Evaluations show the CBA based method outperforms the state-of-the-art local context based attention methods significantly. This brings insight to how cognition grounded data can be brought into NLP tasks.

1 Introduction

Sentiment analysis is critical for many applications such as sentimental product recommendation (Dong et al., 2013), public opinion detection (Pang et al., 2008), and human-machine interaction (Clavel and Callejas, 2016), etc.Sentiment analysis has been well-explored (Pang et al., 2002; Vanzo et al., 2014; Tang et al., 2015a; Chen et al., 2016; Maas et al., 2011).Recently, deep learning based methods have further elevated the performance of sentiment analysis without the need for labor intensive feature engineering.

Attention models are incorporated into sentiment analysis because not all words are created equal. Some words are more important than others in conveying the message in a sentence. Similarly, some sentences are more important than others in a document. Although the overall reading time as a cognitive process may reflect the syntax and discourse complexity, reading time of individual words is also an indicator of their semantic importance in text (Roseman, 2001; Demberg and Keller, 2008). Previous attention models are built using information embedded in text including users, products and text in local context for sentiment classification (Tang et al., 2015b; Yang et al., 2016; Chen et al., 2016; Gui et al., 2016). However, attention models using local context based text through distributional similarity lack theoretical foundation to reflect the cognitive basis. But, the key in sentiment analysis lies in its cognitive basis. Thus, we envision that cognition grounded data obtained in text reading should be helpful in building an attention model.

In this paper, we propose a novel cognition based attention(CBA) model for sentiment analysis learned from cognition grounded eye-tracking data. Eye-tracking is the process of measuring either the point of gaze or the motion of an eye relative to the head[1]. In psycho-linguistics experiments, Barrett(2016) shows that readers are less likely to fixate on close-class words that are predictable from context. Readers also fixate longer on words which play significant semantic roles (Demberg and Keller, 2008) in addition to infrequent words, ambiguous words, and morphological complex words (Rayner, 1998). Since reading time can be learned from an eye-tracking dataset, predicted reading time of words in its context can be used as indicators of attention weights.

[1] https://en.wikipedia.org/wiki/Eye-tracking

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 462–471
Copenhagen, Denmark, September 7–11, 2017.

We first build a regression model to map syntax, and context features of a word to its reading time based on eye-tracking data. We then apply the model to sentiment analysis text to obtain the estimated reading time of words at the sentence level. The estimated reading time can then be used as the attention weights in its context to build the attention layer in a neural network based sentiment analysis model. Evaluation on the four sentiment analysis benchmark datasets (IMDB, Yelp 13, Yelp 14 and IMDB2) show that our proposed model can significantly improve the performance compared to the state-of-the-art attention methods.

To sum up, we have two major contributions: (1) We propose a novel cognition grounded attention model to improve the state-of-the-art neural network based sentiment analysis models by learning attention information from eye-tracking data. This is one of the first attempts to use cognition grounded data in sentiment analysis. The CBA model not only can capture attention of words at the sentence level, it can also be aggregated to work at the document level. (2) Evaluation on several real-world datasets in sentiment analysis shows that our method outperforms other state-of-the-art methods significantly. This work validates the effectiveness of cognition grounded data in building attention models.

2 Related works

The basic task in sentiment analysis can be formulated as a classification problem. Class labels can either be binary (positive/negative) or polarity either as intensity by continuous values or as ratings in certain range such as 0 to 5 or 1 to 10, etc..

In recent years, deep learning based methods have greatly improved the performance of sentiment analysis. Commonly used models include Convolutional Neural Networks (Socher et al., 2011), Recursive Neural Network (Socher et al., 2013), and Recurrent Neural Networks (Irsoy and Cardie, 2014). RNN naturally benefits sentiment classification because of its ability to capture sequential information in text. However, standard RNN suffers from the gradient vanishing problem (Bengio et al., 1994) where gradients may grow or decay exponentially over long sequences. To address this problem, Long-Short Term Memory model (LSTM) is introduced by adding a gated mechanism to keep long term memory. Each LSTM layer is generally followed by mean pooling and then feed into the next layer. Experiments in datasets which contain long documents and sentences demonstrate that the LSTM model outperforms the traditional RNN (Tang et al., 2015a,c).

Not all words contribute equally to the semantics of a sentence (Hahn and Keller, 2016). Attention based neural networks are proposed to highlight their difference in contribution (Yang et al., 2016). In document level sentiment classification, both sentence level attention and document level attention are proposed. In the sentence level attention layer, an attention mechanism identifies words that are important. Those informative words are aggregated as attention weights to form sentence embedding representation. This method is generally called local context attention method. Similarly, some sentences can also be highlighted to indicate their importance in a document.

Apart from local context attention, user/product attentions are also included in deep learning based methods either in a separate network (Gui et al., 2016) or a unified network (Tang et al., 2015c; Gui et al., 2016). Some feature engineering method to some specific datasets can also achieve very good result(Sadeghian and Sharafat, 2015). However, they are not suited for other genre of text as user-product information are not generally available.

Attention models can be built not only from local text or user/product information but also from cognitive grounded data, especially eye-tracking data (Rayner, 1998; Allopenna et al., 1998). Joshi (2014) proposes a novel metric called Sentiment Annotation Complexity for measuring sentiment annotation complexity based on eye-tracking data. Mishra (2014) presents a cognitive study of sentiment detection from the perspective of AI where readers are tested as sentiment readers. Mishra (Mishra et al., 2016b) recently proposes a model in sentiment analysis and sarcasm detection by using eye-tracking data as a feature in addition to text features using Naive-Bayes and SVM classifiers.

In other NLP tasks, Joshi (2013) shows that Word-Sense-Disambiguation can make use of simultaneous eye-tracking. Eye-tracking data are also used to measure the difficulty in translation annotation (Mishra et al., 2013). Barrett (2016) finds that gaze patterns during reading are strongly influenced by the role a word plays in terms of syntax, semantic, and discourse.

Among different available eye-tracking datasets, the Dundee corpus, GECO (the Ghent

Eye-Tracking Corpus), and Mishra et al. (Mishra et al., 2016b) are considered high-quality resources (Kennedy, 2003; Cop et al., 2016; Mishra et al., 2016b). The Dundee corpus contains eye movement data from English and French newspapers (Kennedy, 2003). Measurements were taken while 10 participants read 20 newspaper articles. GECO is an English-Dutch bilingual corpus with eye-tracking data from 17 participants collected from reading the complete novel *The Mysterious Affair at Styles*. The corpus has 4,934 sentences, 774,015 tokens, and 9,876 words. The Mishra(Mishra et al., 2016a) dataset contains 994 text snippets with 383 positive and 611 negative examples from newspaper clippings, sampled from seven native speakers.

To predict reading time using eye-tracking data, Tomanek et al. (2010) proposes a regression model using linguistic features related to syntax and semantics for calibration. Hahn (2016) proposes a novel approach to model both skipping and reading using unsupervised method which combines neural attention with auto-encoding trained on raw text using reinforcement learning.

3 Our proposed CBA model

The basic idea of our method is to add a CBA model into a neural-network based LSTM sentiment classifier. Let D be a collection of documents. A document d_k, $d_k \in D$, has m number of sentences $S_1, S_2, ...S_j, ..., S_m$. A sentence S_j is formed by a sequence of words $S_j = w_1^j w_2^j ... w_{l_j}^j$, where l_j is the length of S_j. The features of a word $w_i \in D$ form a feature vector $\vec{v}^{w_i} = [{F_1}^{w_i}, {F_2}^{w_i} .... {F_n}^{w_i}]$ where n is the feature space size. The purpose of document level sentiment classification is to project a document d_k into the target space of L class labels. Similarly, at the sentence level, the purpose is to project a sentence S_j into the target class space.

To build the CBA model, we need to first build a reading time prediction model for words within each sentence. Reading time is predicted based on word features and text features calibrated by eye-tracking data. Note that reading time from an eye-tracking dataset cannot be used directly because the text of any eye-tracking dataset is too small for sufficient coverage. Consequently, our method has four tasks: (1) to predict the reading time of words using eye-tracking data and $\vec{v}_{w_i}$ as features; (2) to build attention models based on predicted reading time at sentence level and document level; (3) to integrate attentions from other attention models; and (4) to add the attention model into the LSTM based sentiment classifier.

3.1 Modeling of reading time

To learn the reading time of words in a sentence, our method is based on regression analysis using eye-tracking data as dependent variables and context information in $\vec{v}_{w \in S_j}$ as independent variables. In the eye-tracking process, a number of different time measures such as *first fixation duration*, *gaze duration*, and *total reading time*. In this work, we only use *the total reading time*.

Since a document set is always available for sentiment analysis, we use features extracted from these documents to train the regression model. We select features based on the works from Demberg(2008) and Tomanek (2010) to include word features such as word length and POS tags as well as context level syntax and semantic features such as the total number of dominated nodes in a dependency parsing three, the maximum dependency distance, semantic category etc..

Given a word w in a sentence S_j, $w \in S_j$, and its feature vector $\vec{v}_{w \in S_j} = [F_1^w, F_2^w, ..., F_n^w]$ where n is the dimension size in feature space, the regression model on eye-tracking data is a mapping function g between reading time $t_{w \in S_j}$ and $\vec{v}_{w \in S_j}$ as defined below:

$$t_{w \in S_j} = g(\alpha_1 F_1^w + \alpha_2 F_2^w + ... + \alpha_n F_n^w + b), \tag{1}$$

where $t_{w \in S_j}$ is the predicted reading time for w, α_i is the weight of feature F_i^w, and b is a constant. Note that the set of $\alpha_i (i = 1...n)$ forms the weight vector $\vec{\alpha}_w$ for $t_{w \in S_j}$. When $\vec{v}_{w \in S_j}$ takes scalar values, g can be an identity function and thus this model becomes a typical linear regression model. When $t_{w \in S_j}$ takes discrete values, g can be a logistic function and this model becomes a typical logistic regression model.

we set g to be the identity function. The objective function then becomes:

$$\min_{\vec{\alpha}} \sum_{a_i \in \vec{\alpha}}^{n} ||t_{w \in S_j} - y_{w \in S_j}||_2^2 + \lambda R(\vec{\alpha}), \tag{2}$$

where $y_{w \in S_j}$ is the true eye-tracking values of reading time, $R(\vec{\alpha})$ is the regularization of $\vec{\alpha}$, and λ is the regularization weight. When $\lambda = 0$, the model degrades to a linear regression function. In this work, we evaluate the use of both the linear regression model and the Ridge regression model.

3.2 Building the attention based model

Once we have predicted reading time for words used in sentences, the attention model can be built with two components. The first component works at the sentence level to give different words different emphasis in a sentence. The second component works at the document level to give different sentences different emphasis in a document.

For a sentence $S_j = w_1 w_2 ... w_i ... w_{l_j}$ with length l_j, each word w_i in S_j has a corresponding reading time t_{w_i}. Let t_{S_j} denote the total reading time of S_j. Then,

$$t_{S_j} = \sum_{i=1, w_i \in S_j}^{l_j} t_{w_i}. \tag{3}$$

For sentence level attention, the CBA weight for w_i in S_j, denoted as $A_{S_j:w_i}$, can be defined as:

$$A_{S_j:w_i} = \frac{t_{w_i}}{t_{S_j}}. \tag{4}$$

This sentence level attention model defined above gives more weights to words that have longer reading time relative to the total reading time of the sentence.

Let a document d_k, $d_k \in D$, be formed by a set of sentences $S_j = w_1 w_2 ... w_i ... w_{l_j}$. Now the CBA weight for a sentence S_j in d_k is defined as:

$$A_{d_k:S_j} = \frac{t_{s_j}}{\sum_{i=1}^{m} t_{S_i}}. \tag{5}$$

This aggregated document level attention model gives more weights to the sentences that have longer reading time relative to the total reading time of the document. Let $\vec{A}_{d_k}$ denote the document level attention weight vector. The size of $\vec{A}_{d_k}$ should be m, the number of sentences in d_k.

Let $\vec{S}_j$ denote the embedding of S_j in N dimensional space, where $S_j \in d_k$. Then, the set of sentence representations for d_k should be a matrix of size $m \times N$, denoted by $\hat{S}_{d_k}$. After the inclusion of the attention model, $\hat{S}_{d_k}$ should be:

$$\hat{S}_{d_k} = \vec{A}_{d_k} \vec{S}_j^T. \tag{6}$$

Let $\vec{d}_k$ denote the document embedding of d_k. Since $\vec{d}_k$ is an N dimensional vector, $\vec{d}_k$ can now be defined by the adjusted attention model as

$$(\vec{d}_k)_i = \sum_{j=1}^{m} (\hat{S}_{d_k})_{i,j}. \tag{7}$$

3.3 Incorporation of other attention models

Since document embedding representation allows the combined use of multiple attention mechanisms, it is to our advantage to incorporate different attention mechanisms which may help to capture different aspects of attentions. Generally speaking, different attention mechanisms can be incorporated either serially or in parallel.

In principle, any number of attention models can be included. As an an example to illustrate the capability of our proposed method, we choose one state-of-the-art local attention model(shorthanded as LA). The model is a semantic-based local attention model proposed by Yang (2016) and also used by Chen (2016). For inclusion serially, the attention weight is formulated as follows:

$$A^s{}_{S_j}{:}w_i = LA_{S_j:w_i} * A_{S_j:w_i}, \tag{8}$$

where $LA_{s_j:w_i}$ the sentence level attention model by the local attention model. To incorporate LA in parallel mode, the attention weight can be formulated by:

$$A^p{}_{S_j}{:}w_i = LA_{S_j}{:}w_i + A_{S_j}{:}w_i. \tag{9}$$

Similar methods can be used at document level.

3.4 General sentiment analysis model

We take the neural network based LSTM sentiment classifier (Gers, 2001) to be applied in both the sentence level and the document level because of its excellent performance on long sentences (Tang et al., 2015a). The basic LSTM model has five internal vectors for a node i including an input gate $\vec{i_i}$, a forget gate $\vec{f_i}$, an output gate $\vec{o_i}$, a candidate memory cell $\vec{c_i'}$, and a memory cell $\vec{c_i}$, and $\vec{i_i}\vec{f_i}$ and $\vec{o_i}$ are used to indicate which values will be updated, forget or for keeping in the LSTM model. $\vec{c_i'}$ and $\vec{c_i}$ are used to keep the candidate features and the actual accepted features, respectively.

At the sentence level, each word w_i in a sentence S_j is represented by its word embedding $\vec{w_i}$ in the N dimensional space. The LSTM cell state $\vec{c_i}$ and the hidden state $\vec{h}_{S_j:w_i}$ can be updated in two steps. In the first step, the previous hidden state $\vec{h}_{S_j:w_{i-1}}$ uses a hyperbolic function to form $\vec{c_i'}$ as defined below.

$$\vec{c_i'} = tanh(\hat{W}_c * [\vec{h}_{S_j:w_{i-1}} * \vec{w_i}] + \hat{b}), \qquad (10)$$

where $\hat{W}_c$ is a parameter matrix, $\vec{h}_{S_j:w_{i-1}}$ is the previous hidden state and $\vec{w_i}$ is the word vector. $\hat{b}$ is the regularization parameter matrix. In the second step, $\vec{c_i}$ is updated by $\vec{c_i'}$ and its previous state $\vec{c}_{i-1}$ to form $\vec{c_i}$ according to the below formula:

$$\vec{c_i} = \vec{f_i} \odot \vec{c}_{i-1} + \vec{i_i} \odot \vec{c_i'}. \qquad (11)$$

The hidden state of w_i can be obtained by

$$\vec{h}_{S_j:w_i} = \vec{o_i} tanh(\vec{f_i} \odot \vec{c_i}). \qquad (12)$$

The forget gate $\vec{f_i}$ is designed to keep the long term memory. A series of hidden states $\vec{h_1}\vec{h_2}...\vec{h_i}$ can serve as input to the attention layer to obtain sentence representation $\vec{S_j}$. In the document level, similar method is used to get the sentence matrix $\hat{S}$ in the document level LSTM layer to obtain the final document representation $\vec{d_k}$.

In our work, the final document representation $\vec{d_k}$ encodes both the sentence level information and the document level information. In the LSTM model, we use a hidden layer to project the final document vector $\vec{d_k^f}$ through a hyperbolic function.

$$\vec{d_k^f} = tanh(\hat{W}_h \vec{d_k} + \hat{b}_h), \qquad (13)$$

where $\hat{W}_h$ is the hidden layer weight matrix and $\hat{b}_h$ is the regularization matrix.

Finally, sentiment prediction for any label $l \epsilon L$ obtained by the softmax function defined below:

$$P(y = l|\vec{d_k^f}) = \frac{e^{\vec{d_k^f}^T \vec{W_l}}}{\sum_{l=1}^{L} e^{\vec{d_k^f}^T \vec{W_l}}} \qquad (14)$$

where $\vec{W_l}$ is the softmax weight for each label.

4 Performance evaluation

Our proposed CBA for sentiment classification is evaluated on four document sets: The first three datasets IMDB, Yelp 13, and Yelp14 which are review texts including user/product information developed by Tang (2015a). The last dataset IMDB2 is a plain text by Maas (2011). All four datasets are tokenized through the Stanford NLP tool (Manning et al., 2014).

Table 1 list the statistics of the datasets including number of classes, number of documents, and average length of sentence. We split train/development/test set in the rate of 8:1:1. The best configuration of the development dataset is used in the test set to obtain the final result.

Data	#class	#doc	#user	#pro	#len*[2]
IMDB	10	84,919	1,310	1,635	24.56
Yelp14	5	231,163	4,818	4,194	17.25
Yelp13	5	78,966	1,631	1,631	17.37
IMDB2	2	50,000	N/A	N/A	20.10

Table 1: Statistics of three benchmark datasets

Two commonly used performance evaluation metrics are used. The first one is accuracy and the second one is rooted mean square error (RMSE)[3]. Let GR_i be the golden sentiment ratings, PR_i be the predicted sentiment rating, and T be the number of documents where $GR_i = PR_i$. Accuracy is then defined by

$$Accuracy = \frac{T}{N}, \qquad (15)$$

and RMSE is defined by

$$RMSE = \sqrt{\sum_{i=1}^{N}(GR_i - PR_i)^2 * \frac{1}{N}}. \qquad (16)$$

We train the skip-gram word embedding (Mikolov et al., 2013) on each dataset separately to initialize the word vectors. All embedding sizes on the model are set to 200, a commonly used size.

Three sets of experiments are conducted. The first is on the selection of the regression model for reading time prediction. The second set of experiments compares our proposed CBA with another sentiment analysis method which use text only. The third set of experiments evaluates the effectiveness of combining different attention models.

4.1 Reading time prediction

The training for the regression model for reading time prediction using eye-tracking data requires the learning from text and context features as discussed in Section 3.1. We compare our regression model with more complex deep learning based regression models in each of the three eye-tracking datasets.[4]

[3] Normally accuracy is a problematic measure in highly unbalanced data sets. But in In IMDB, the largest class only takes less than 20% of all instances out of classes. The most imbalanced data are Yelp 13 whose largest class is 41% among 5 classes and second largest is about 30%. IMDB has a 50/50 split for 2-classes.

[4] Mishra et.al (Mishra et al., 2016a) only provides fixation time. So, fixation time is used when training by this set of eye-tracking data.

We take the first 90% of sentences as training data and the rest 10% as test data. The configuration that performs the best is selected and predicated on the document sentiment analysis dataset to obtain estimated reading time. Ideally, an eye-tracking corpus built from on-line reviews is more suitable for our experiments. But, we can only work with what is available.

In addition to the linear regression model(LL) and the Ridge regression model(RR), we also choose the Recurrent Neural Network (RNN) model and the Long Short Time Memory (LSTM) model for regression learning. For both models, there are two versions. The basic version inputs the extracted feature sets as word representation, labeled as RNN-1 and LSTM-1, respective. The second version takes word embedding (Pennington et al., 2014) as the initial word representation input, labeled as RNN-2 and LSTM-2, respectively. The RMSE results are listed in Table 2.

	GECO	DUNDEE	Mishra
LR	72.47	73.52	87.25
RR	69.47	70.52	84.22
RNN-1	75.47	83.52	96.23
LSTM-1	79.47	84.52	114.25
RNN-2	79.57	86.47	101.25
LSTM-2	83.88	95.88	122.27

Table 2: RMSE for reading time prediction(Unit:Milliseconds)

Note that Ridge Regression(RR) has the best performance on all the three datasets because regularization in RR reduces over-fitting problem.In three eye tracking datasets, the RR can achieve coefficient of determination[5] of 0.32, 0.30 and 0.27 in three eye tracking datasets. The features, their types and the corresponding coefficients in RR are shown in Table 3.

The more complicated deep learning models suffer from serious over-fitting problem. And the result of Deep learning model with word embedding initialization partly supports the fact that the reading time are more depend on the micro level syntax and semantic feature for the word, such as number of letters in word and complexity score of the word instead of the deep level context features.

4.2 Comparison of different sentiment classification methods

Because the features used in our model are all text based, we compare CBA with two groups of baseline methods which also only use review text for learning. Group 1 methods include commonly known linguistic and context features for SVM classifiers. Group 2 includes recent sentiment classification algorithms which are top performers using review text for training including one method that uses local attention model. Below is the list of Group1 methods.

Feature_Name	Type	Cofficient
Number of letters	Num	22.441
Start with capital letter	Bool	1.910
Capital letters only	Bool	161.580
Have alphanumeric letters	Bool	6.020
Is punctuation	Bool	-8.930
Is abbreviation	Bool	10.551
Is entity-critical word	Bool	7.612
Number of dominated nodes	Num	0.980
Max dependency distance	Num	1.982
Inverse document frequency	Num	-9.291
Number of senses in wordnet	Num	7.494
Complexity score	Num	57.240
Constant	Num	239.910

Table 3: Major features used by the Ridge Regression Model

- **Majority** — A simple majority based classifier based on sentence labels.
- **Trigram** — A SVM classifier using unigrams/bigrams/trigram as features.
- **Text feature** — A SVM classifier using word level and context level features, such as n-gram and sentiment lexicons.
- **AvgWordvec** — A SVM classifier that takes the average of word embeddings in Word2Vec as document embedding.

Here is a list of Group 2 methods:

- **SSWE** (Tang et al., 2014) — A SVM classifier using sentiment specific word embedding.
- **RNTN+RNN** (Socher et al., 2013) — A Recursive Neural Tensor Network(RNTN) to represent sentences and trained using RNN.
- **Paragraph vector** (Le and Mikolov, 2014) — A SVM classifier using document embedding as features.
- **LSTM+LA** (Chen et al., 2016) — State-of-the-art LSTM using local context as attention mechanism in both sentence level and document level.

[5] $https://en.wikipedia.org/wiki/Coefficient_of_determination$

		IMDB		Yelp13		Yelp14	
		ACC	RMSE	ACC	RMSE	ACC	RMSE
General baseline (Group 1)	Majority	0.196	2.495	0.411	1.060	0.392	1.097
	Trigram	0.399	1.783	0.569	0.814	0.577	0.804
	TextFeature	0.402	1.793	0.556	0.845	0.572	0.801
	AveWord2vec	0.304	1.985	0.526	0.898	0.531	0.893
Recently developed methods (Group 2)	SSWE+SVM	0.312	1.973	0.549	0.849	0.557	0.851
	Paragraph Vector	0.314	1.814	0.554	0.832	0.564	0.802
	RNTN+RNN	0.401	1.764	0.574	0.804	0.582	0.821
	CLSTM	0.421	1.549	0.592	0.769	0.594	0.766
	B-CLSTM	0.462	1.453	0.619	0.705	0.592	0.741
	LSTM	0.443	1.465	0.627	0.701	0.637	0.686
	LSTM+LA	0.487	1.381	0.631	0.706	0.631	0.715
CBA based models	LSTM+CBA^M	0.447	1.495	0.610	0.746	0.613	0.768
	LSTM+CBA^D	0.468	1.419	0.623	0.706	0.628	0.702
	LSTM+CBA^G	**0.489**	**1.365**	**0.638**	**0.697**	**0.641**	**0.678**

Table 4: Evaluation on sentiment classification using review text for training

- **CLSTM** (Xu et al., 2016) — Cached LSTM to capture the overall semantic information in long text. The two variations include regular **CLSTM** and bi-directional **B-CLSTM**.

- **LSTM+UPA** (Chen et al., 2016) — State-of-the-art LSTM including LA as well as user/product as attention mechanism at both sentence level and document level.

Our proposed CBA model has several variations as explained below.

- **LSTM+CBA** — The LSTM classifier using only CBA model at sentence level and document level. Based on the three eye-tracking datasets(GECO, DUNDEE and Mishra's) for reading time prediction, we label the same model by different training data as **LSTM+CBA**^{G},**LSTM+CBA**^{D} and **LSTM+CBA**^{M}.

- **LSTM+CBA+LA**^{G} — The LSTM based classifier using both the CBA model and the local text context based attention model(LA) (Chen et al., 2016). Since combining method can either be serial or in parallel, there are actually two corresponding variations: LSTM+CBA+LA$_s^G$ and LSTM+CBA+LA$_p^G$.

- **LSTM+CBA+UPA**^{G} — The same framework to **LSTM+CBA+LA**^{G} with additional user/product attention. The two corresponding variations are LSTM+CBA+UPA$_s^G$ and LSTM+CBA+UPA$_p^G$.

Table 4 shows the performance of the three groups using review text without user/product information on only the first three datasets methods in Group 1 and Group 2 do not have evaluations on IMDB2. Among all the reference methods that do not use any attention mechanism including all methods in Group 1 and Group 2(except LSTM+LA), LSTM is the best performer. LSTM+LA (2016), which is the state-of-the-art method, uses local attention mechanism to improve performance significantly. Among our CBA based variations, using the GECO dataset gives the best result outperforming LSTM+LA in all three datasets. LSTM+CBA^G has significant improvement over LSTM+LA with p values of $p < 0.016$ on IMDB, $p < 0.0019$ on Yelp 13, and $p < 0.00023$ on Yelp 14. LSTM+CBA^G has the best result compared to the other two variations because GECO has larger participant size. Its text genre is also closer to the review datasets for sentiment analysis.

In the third set of experiment, we compare our LSTM+CBA model with the combination of other attention models including the LA model and the UPA model as shown in Table 5. In the second set of experiment, since the GECO dataset gives the best performance, Table 5 shows the performance of LSTM+CBA using only the GECO dataset including LSTM+CBA^G, LSTM+CBA+LA$_s^G$, LSTM+CBA+LA$_p^G$, LSTM+CBA+UPA$_s^G$,and LSTM+CBA+UPA$_p^G$. Note that UPA is build based on user/product information. So it works

	IMDB		Yelp13		Yelp14		IMDB2	
	ACC	RMSE	ACC	RMSE	ACC	RMSE	ACC	RMSE
LSTM+LA	0.487	1.381	0.631	0.706	0.631	0.715	0.885	0.337
LSTM+CBA^G	0.489	1.365	0.638	0.697	0.641	0.678	0.894	0.332
LSTM+CBA+LA$_s^G$	0.488	1.369	0.633	0.706	0.643	0.672	0.898	0.328
LSTM+CBA+LA$_p^G$	0.492	1.362	0.639	0.696	0.639	0.675	**0.901**	**0.322**
LSTM+UPA	**0.533**	1.281	0.650	0.692	0.667	0.654	N/A	N/A
LSTM+CBA+UPA$_s^G$	0.523	**1.277**	0.654	0.693	0.664	0.645	N/A	N/A
LSTM+CBA+UPA$_p^G$	0.521	1.278	**0.655**	**0.685**	**0.668**	**0.644**	N/A	N/A

Table 5: Evaluation on sentiment classification on using dual attention

only if user/product information is available. Such data is provided in the first three sets of data.

Table 5 shows that among all three single attention models, UPA outperforms both LA and CBA in the first three datasets. This is easier to understand as UPA already included LA and it has more explicit information from users and products for its attention model compared to CBA which needs to learn hidden attention information. The combined method of CBA with UPA can still further improve performance. When CBA+UPA are combined in parallel, it has the best performance for both Yelp13 and Yelp14 (with p value of 0.027 and 0.032 respectively compare to LSTM+UPA). In the IMDB dataset, however, UPA has the best performance. This may be because user/product information is more effective in movie review IMDB dataset which is more subjective.

However, the UPA model works only if user and product information is available. Thus for IMDB2 where user/product information is not available, only CBA and LA models work and the combined use of CBA+LA gives the best performance.

4.3 Case study

A random sentence sample *'The Shelton hotel is lucky to receive 2stars from me considering ...'* is taken from the Yelp13 dataset to demonstrate the difference in the two attention mechanisms, i.e. local text(LA), and cognition-based(CBA). Figure 1 shows visually the difference in attention weights of the two models.

The attention weights of words in the LA model does not change much. CBA, on the other hand, gives higher weights to the sentiment linked word *2stars* and the verb *receive*. This two words do play significant roles as an indirect object and a main verb, respectively. This case shows that CBA does a better job in capturing micro level information in the sentence level. This support the experimental results in Table 4 and Table 5.

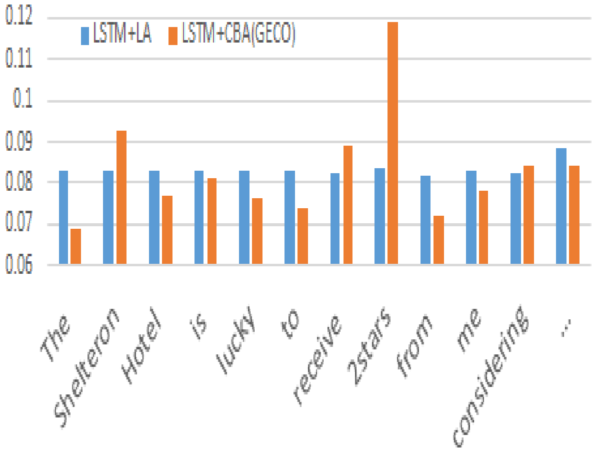


Figure 1: Case Study on attention weights

5 Conclusion and future works

In this paper, we propose a novel cognition based attention model to improve the state-of-the-art neural sentiment analysis model through cognition grounded eye-tracking data. A simple and effective regression model is used to predict reading time using both eye-tracking data and local text features. The predicted reading time is then used to build an attention layer in neural sentiment analysis models. The attention model considers both reading time and other syntactic and context features. It works in both the sentence level and the document level sentiment analysis.

Evaluation on benchmarking datasets validates the effectiveness of our method in sentiment analysis as our method clearly outperforms other state-of-the-art methods that use local context information to build their attention models. Our CBA mechanism can also be combined with other attention mechanisms to provide room for further improvement. Future work includes using other eye-tracking information such as saccade and fixation. The incorporation of other information such as user-product information can also be explored.

Acknowledgments

The work is partially supported by the research grants from Hong Kong Polytechnic University (PolyU RTVU) and GRF grant(CERG PolyU 15211/14E).

References

Paul D Allopenna, James S Magnuson, and Michael K Tanenhaus. 1998. Tracking the time course of spoken word recognition using eye movements: Evidence for continuous mapping models. *Journal of memory and language*, 38(4):419–439.

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

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. EMNLP.

Chloé Clavel and Zoraida Callejas. 2016. Sentiment analysis: from opinion mining to human-agent interaction. *IEEE Transactions on affective computing*, 7(1):74–93.

Uschi Cop, Nicolas Dirix, Denis Drieghe, and Wouter Duyck. 2016. Presenting geco: An eyetracking corpus of monolingual and bilingual sentence reading. *Behavior research methods*, pages 1–14.

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

Ruihai Dong, Michael P O'Mahony, Markus Schaal, Kevin McCarthy, and Barry Smyth. 2013. Sentimental product recommendation. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 411–414. ACM.

Felix Gers. 2001. *Long short-term memory in recurrent neural networks*. Ph.D. thesis, Universität Hannover.

Lin Gui, Ruifeng Xu, Yulan He, Qin Lu, and Zhongyu Wei. 2016. Intersubjectivity and sentiment: from language to knowledge. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 2789–2795.

Michael Hahn and Frank Keller. 2016. Modeling human reading with neural attention. *arXiv preprint arXiv:1608.05604*.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *EMNLP*, pages 720–728.

Aditya Joshi, Abhijit Mishra, Nivvedan Senthamilselvan, and Pushpak Bhattacharyya. 2014. Measuring sentiment annotation complexity of text. In *ACL (2)*, pages 36–41.

Salil Joshi, Diptesh Kanojia, and Pushpak Bhattacharyya. 2013. More than meets the eye: Study of human cognition in sense annotation. In *HLT-NAACL*, pages 733–738.

Michael Hahn Frank Keller. 2016. Modeling human reading with neural attention. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, volume 85, page 95.

A Kennedy. 2003. The dundee corpus [cd-rom]. *Psychology Department, University of Dundee*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Abhijit Mishra, Pushpak Bhattacharyya, Michael Carl, and IBC CRITT. 2013. Automatically predicting sentence translation difficulty. In *ACL (2)*, pages 346–351.

Abhijit Mishra, Aditya Joshi, and Pushpak Bhattacharyya. 2014. A cognitive study of subjectivity extraction in sentiment annotation. *ACL 2014*, page 142.

Abhijit Mishra, Diptesh Kanojia, and Pushpak Bhattacharyya. 2016a. Predicting readers' sarcasm understandability by modeling gaze behavior. In *AAAI*, pages 3747–3753.

Abhijit Mishra, Diptesh Kanojia, Seema Nagar, Kuntal Dey, and Pushpak Bhattacharyya. 2016b. Leveraging cognitive features for sentiment analysis. *CoNLL 2016*, page 156.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

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

Ira J Roseman. 2001. A model of appraisal in the emotion system: Integrating theory, research, and applications.

Amir Sadeghian and Ali Reza Sharafat. 2015. Bag of words meets bags of popcorn.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Duyu Tang, Bing Qin, and Ting Liu. 2015a. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

Duyu Tang, Bing Qin, and Ting Liu. 2015b. Learning semantic representations of users and products for document level sentiment classification. In *Proc. ACL*.

Duyu Tang, Bing Qin, and Ting Liu. 2015c. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1014–1023, Beijing, China. Association for Computational Linguistics.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL (1)*, pages 1555–1565.

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

Andrea Vanzo, Danilo Croce, and Roberto Basili. 2014. A context-based model for sentiment analysis in twitter. In *COLING*, pages 2345–2354.

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuangjing Huang. 2016. Cached long short-term memory neural networks for document-level sentiment classification. *arXiv preprint arXiv:1610.04989*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Author-aware Aspect Topic Sentiment Model to Retrieve Supporting Opinions from Reviews

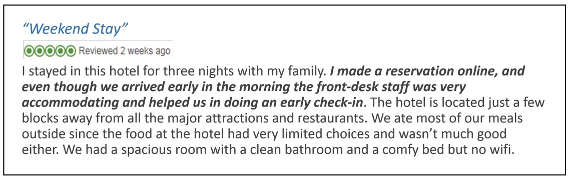
Lahari Poddar, Wynne Hsu, Mong Li Lee
School of Computing
National University of Singapore
{lahari, whsu, leeml}@comp.nus.edu.sg

Abstract

User generated content about products and services in the form of reviews are often diverse and even contradictory. This makes it difficult for users to know if an opinion in a review is prevalent or biased. We study the problem of searching for supporting opinions in the context of reviews. We propose a framework called SURF, that first identifies opinions expressed in a review, and then finds similar opinions from other reviews. We design a novel probabilistic graphical model that captures opinions as a combination of aspect, topic and sentiment dimensions, takes into account the preferences of individual authors, as well as the quality of the entity under review, and encodes the flow of thoughts in a review by constraining the aspect distribution dynamically among successive review segments. We derive a similarity measure that considers both lexical and semantic similarity to find supporting opinions. Experiments on TripAdvisor hotel reviews and Yelp restaurant reviews show that our model outperforms existing methods for modeling opinions, and the proposed framework is effective in finding supporting opinions.

1 Introduction

In order to make an informed decision when booking a hotel online, a user will often read through its reviews looking for specific feedbacks. For example, if he or she plans to do an early check-in and comes across a review that mentions a hassle-free early check-in as shown in Figure 1, it will be helpful to know whether other guests had similar experiences. If a review complains about bed bugs or noise from construction nearby, then it is important to know if that was an occasional problem based on a single user's experience or happens frequently. However, it is impossible for an individual to go through the large volume of reviews to verify whether an opinion is prevalent.



"Weekend Stay"

●●●●● Reviewed 2 weeks ago

I stayed in this hotel for three nights with my family. ***I made a reservation online, and even though we arrived early in the morning the front-desk staff was very accommodating and helped us in doing an early check-in***. The hotel is located just a few blocks away from all the major attractions and restaurants. We ate most of our meals outside since the food at the hotel had very limited choices and wasn't much good either. We had a spacious room with a clean bathroom and a comfy bed but no wifi.

Figure 1: A sample hotel review

In this work, we study the problem of finding supporting sentences from reviews that corroborate the opinions expressed in a target review sentence. This is useful as it enables users to easily look for appropriate comments on the specific issues they are interested in.

A review is a collection of sentences where each sentence may have multiple segments separated by punctuations or conjunctions. Each segment expresses an opinion that can be represented as a combination of aspect, topic and sentiment. An aspect refers to the overall theme of a segment, a topic is the specific subject or issue discussed and the sentiment for each topic can be neutral, positive or negative. Table 1 shows the segments and the possible latent aspect, topics and sentiment for a sentence of the review in Figure 1.

Review Sentence	Segments	Aspect	Topic	Sentiment
We had a big room with clean bathroom and a comfy bed, but no wifi	We had a big room with clean bathroom	room	room	positive
			bathroom	positive
	a comfy bed	room	bed	positive
	no wifi	amenities	wifi	negative

Table 1: Opinion structure for a review sentence

Given an opinion (in a target segment), we say that a review supports the opinion, if it contains some segment whose aspect, topic and sentiment are similar to those in the target segment. Finding such supporting reviews is a challenge since

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 472–481
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

reviews are typically short unstructured text and discuss a wide range of topics on various aspects with differing sentiments and vocabulary used.

Topic modeling have been widely used to reduce the effect of huge vocabulary by grouping words in topics. However, the fundamental assumption of topic models is the independence of topics even in the same document. This fails to capture the natural coherence present in reviews, which rarely consist of isolated, unrelated sentences, but are composed of collocated, structured and coherent groups of sentences (Hovy, 1993). We observe that an author's train of thoughts when writing a review is often linear, i.e., he or she will finish discussing one aspect before moving on to the next. In Figure 1, we see that the user first commented on *Service* (*"front-desk staff was very accommodating"*), then the *Location* aspect, followed by the comment on *Food*, and finally moved on to *Room*. This shows that aspects discussed in a review are not chosen from a simple *independent mixture*, but rather, words in close proximity tend to discuss the same aspect and within a review the aspects discussed in the current segment will affect the possible aspects for the successive segments.

We explicitly model this by constraining aspect transition between segments using a review specific Markov chain. Each segment is assumed to discuss a single aspect and possible aspects for a segment are made dependent on the aspects of the previous segments. By tracking aspects of previous segments we are able to ensure constrained aspect sampling for accurate modeling of a review structure. This non-iterative nature of discourse has not been considered by existing works.

For opinion modeling, capturing the sentiment expressed for an aspect is important. Recent works (Kim et al., 2013; Jo and Oh, 2011; Moghaddam and Ester, 2011; Wang et al., 2010; Titov and McDonald, 2008a,b) have developed models to capture aspect and sentiment. However, they do not consider the preferences of authors, or the inherent quality of the entity for the aspect. In a hotel review, the sentiment expressed for *service* depends on both the service standard of the hotel (evident from the sentiment distribution of *service* of all reviews for the hotel) and the expectation of the author for *service* (evident from the sentiment distribution of the author on *service* across all hotels) (Poddar et al., 2017). We take this into account by making the sentiment distribution of a review dependent on both entity and author.

We propose an Author-aware Aspect Topic Sentiment model (Author-ATS) to capture the diverse opinions, taking into account user preferences and thought patterns. The model considers a word to be generated from a hierarchy of aspect, topic and sentiment and encodes the coherent structural property of a review by dynamically constraining aspect distributions. We also develop a non-parametric version of Author-ATS based on Dirichlet Process called Author-ATS (DP).

We develop a SUpporting Review Framework (SURF) that utilizes the Author-ATS model to compute the lexical and semantic similarity of an opinion in a target segment to those in the review corpus, and returns the top-k supporting reviews. Experiments on real world review datasets show the effectiveness of Author-ATS in modeling opinions compared to existing topic models. Furthermore, SURF outperforms keyword-based approaches and word embedding based similarity measures in finding supporting opinions. To the best of our knowledge, this is the first work to find supporting reviews for an opinion expressed in user generated contents.

2 Related Work

There has been substantial research to mine online reviews using topic models (Paul and Girju, 2010; Trabelsi and Zaiane, 2014; Lin and He, 2009; Jo and Oh, 2011; Mukherjee and Liu, 2012; Chen et al., 2013). The Topic Aspect Model (TAM) (Paul and Girju, 2010) jointly discovers aspects and topics from documents. The aspect and topic are independent and each aspect affects all topics in similar manner. However, in reviews, the topics discussed are often closely related to an aspect. JTV (Trabelsi and Zaiane, 2014) encodes topic-viewpoint dependency, but assumes that a document contains only one aspect. JST (Lin and He, 2009) assumes that there is a single sentiment polarity for a review and the topics are chosen conditioned on that, while ASUM in (Jo and Oh, 2011) assumes that all words in a sentence are associated with the same topic and sentiment. In contrast, our proposed model handles the more realistic scenario where sentiments may vary depending on the topics discussed in a review.

For incorporating author information, the User-Sentiment topic model (Zhao et al., 2012) considers the topic-sentiment distribution only from the author perspective and ignores the characteristics of the entity. Supervised topic model (Li et al., 2014) uses explicit ratings to infer sentiments. PDA-LDA (Zhang and Wang, 2015) associates its Dirichlet prior distribution with user and item topic factors. The work in (Yang et al., 2015) models aspects and sentiments based on the demography of authors. However, such demographic information are not always available and it cannot model the bias or preference of an individual.

Additionally, most topic models are concerned about the discourse at word level, and ignore the document structure. HTMM (Gruber et al., 2007) models topic coherence by considering topic transition between sentences. HTSM (Rahman and Wang, 2016) extends HTMM by capturing sentiment shifts along with topic coherence. Both models do not capture the non-repetitive discourse of reviews. Progressive topical dependency model (Du et al., 2010, 2015) captures the sequential nature of ideas among segments, especially in movies or books. However, unlike books, the sequence of topics in reviews is not significant. Rather, once a topic has been discussed in a review, it is unlikely to be mentioned again in a later segment. From this perspective, it is similar to labeled LDA (Ramage et al., 2009) where topic distribution of a document is constrained. However, unlike labeled LDA, the possible aspects of a segment are dynamically constrained depending on previously sampled aspects.

3 Author-ATS Model

Author-ATS models an opinion as hierarchical dependent mixtures, where words are generated from a three-level hierarchical structure of aspects, topics and sentiments. We assume there are A distinct aspects for a domain, for each aspect there are Z topics and for each aspect-topic pair S possible sentiments. We treat a segment as the basic semantic unit, discussing a particular aspect. A review r is a collection of D_r segments where each segment is a document d, consisting of N_d words. We now describe the assumptions and detailed construction of the proposed model.

3.1 Constrained Aspect Generation

We explicitly model the behavior that after an author has finished discussing an aspect and has moved on to the next, he or she is unlikely to return to it again. We assume that each document d discusses a single aspect a_d. The aspect distribution σ_r is drawn from a Dirichlet with parameter α. In order to model the linear writing style of authors, we constrain the possible aspects that can be sampled from σ_r. Whenever an author starts writing a segment, he or she can choose to either (a) talk about an aspect not yet discussed, or (b) continue with the aspect of the previous segment. This is captured by imposing the constraint that the aspect of the j^{th} document is dependent on the aspects of the $(j-1)^{th}, (j-2)^{th}, \cdots, 1^{st}$ documents of the same review.

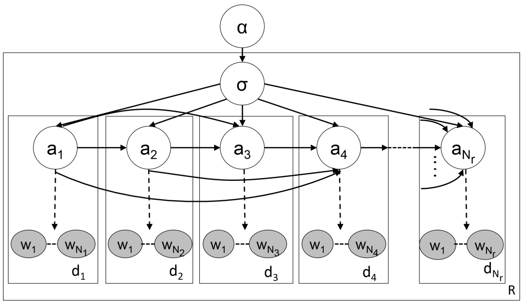


Figure 2: Constrained aspect generation in Author-ATS. Aspects in review form a Markov chain.

With this we relax the *independent mixture* assumption of the standard LDA model for aspects and form a review-specific Markov chain (see Figure 2). Such a higher order Markov chain would normally incur intractable computational complexity due to the exponential size of transition probability matrix. However, in our case, the transition probability can be determined by overall aspect distribution of the review, σ_r and a list of possible aspects for the segment. Since we assume a non-repetitive nature of discourse, the number of possible aspects for a segment is monotonically decreasing for successive segments. This special property enables us to devise a dynamic programming strategy to solve the problem with linear complexity.

Each document is associated with a binary aspect vector Λ. We restrict the sampled aspect of a document to be drawn from only the aspects that are turned on, in Λ of that document. For a document d, $\Lambda_d =< l_1, \cdots, l_A >$ where each $l_a \in \{0, 1\}$ and A is the total number of aspects. Traditionally, for a document d, an aspect a_d is sampled from a multinomial distribution σ_r. Here, we restrict the possible sampled aspects to the list Λ_d. A value of 1 for the entry l_a indicates that the aspect a can be sampled, while 0 indicates that the aspect should not be sampled.

We generate Λ_d by tossing a Bernoulli coin for each aspect a with prior probability Φ_a for value 0. We set Φ_a as the sampling probability for aspects which have been sampled for a previous document. This ensures that an aspect which has been discussed before has lesser probability of coming up again. We set $\Phi_a = 0$ for aspects not sampled in the past, and for the aspect of (immediately) preceding segment. This models aspect coherency in a review document where an author either chooses to discuss a new aspect or continues to talk about the current one.

We define the list of possible aspects for the document d to be $\lambda_d = \{a \mid \Lambda_d[a] = 1\}$. We sample an aspect a_d from σ_r with the constraint that $a_d \in \lambda_d$ i.e. an aspect can be sampled for a document only if it is turned on in the binary aspect vector for the document and thereby exists in the list of possible aspects for the document. Thus, the aspect transition probability among documents becomes dependent on σ_r and the vector λ_d. Unlike regular topic models, Author-ATS is no longer invariant to reshuffling of words and is able to model linear aspect coherency in a review.

3.2 Author-Entity dependent Sentiment Distribution

We account for the dual role of entity and author in a review, by observing that the sentiments expressed are influenced by both the quality of the entity being reviewed and the preferences of the author. We use two Dirichlet distributions to derive sentiment, namely, entity-dependent distribution (ξ) and author-dependent distribution (χ). For each aspect-topic combination, ξ is drawn from a Dirichlet distribution with prior γ^1 and χ is drawn from a Dirichlet distribution with prior γ^0.

Since online reviews describe experiences of people, some words tend to appear frequently (e.g.: 'hotel','trip' or 'mobile', 'phone' for hotel and mobile reviews respectively). We call them *domain stopwords* as they are not specific to any aspect. We use a binary switching variable y_i to determine the type for the i^{th} word. If $y_i = 0$, then the word is aspect neutral (domain stopword); and if $y_i = 1$, it is aspect dependent.

The generative process of the model is as follows:

- Draw a multinomial word distribution ϕ_0 for domain stopwords and ϕ_1 for each aspect, topic and sentiment words from Dir (ω).
- For each author u, draw a multinomial sentiment mixture χ for each aspect and topic from Dir(γ^0)
- For each entity e, draw a multinomial sentiment mixture ξ for each aspect and topic from Dir (γ^1)
- For each review r:
    1. Draw multinomial aspect mixture σ from Dir(α)
    2. For each document $d \in r$:
        (a) Draw Λ_d from Bernoulli (Φ)
        (b) Draw a type mixture ψ from Beta (δ_0, δ_1)
        (c) Sample an aspect a_d from σ s.t. $a_d \in \lambda_d$
        (d) For sampled aspect a_d, draw a topic mixture θ from Dir (β)
        (e) For each word position i where $0 \leq i \leq N_d$
            i. Sample a type y_i from ψ
            ii. Sample a topic z_i from θ
            iii. Sample a sentiment s_i from χ and ξ
            iv. Sample a word w_i from $\begin{cases} \phi_0 & \text{if } y_i = 0, \\ \phi_1 & \text{if } y_i = 1 \end{cases}$

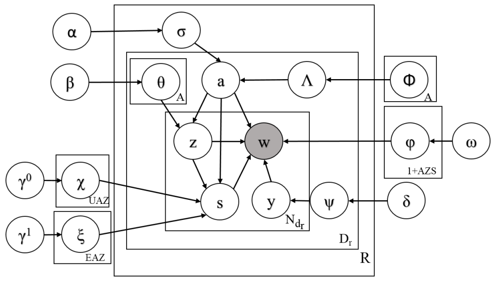


Figure 3: Graphical representation of Author-ATS

Note that for the first document of a review, we set λ_0 to the set of all possible aspects, such that there is no constraint when sampling for the first segment of a review. Figure 3 shows the plate notation for Author-ATS model.

3.3 Bayesian Inference

We employ collapsed Gibbs sampling for inference. Markov chain introduced for aspect coherency makes the aspects non-exchangeable, hence sampling an aspect for a segment will also affect all subsequent segments. Since the exact sampling for this would be computationally expensive, we propose the following approximate posterior considering only the previous segments, which has been shown to work well in similar cases previously (Mimno et al., 2011).

We sample an aspect (a_d) for each document based on the posterior probability of the type, topic and sentiment assignment of each word in the document and the aspects sampled for preceding documents in the review.

$$\begin{aligned} P(a_d|a_{\vec{-d}},\vec{y}_{-d},\vec{z}_{-d},\vec{s}_{-d},\vec{w}) &\propto P(a_d|a_{1:\vec{d}-1}) \\ &\prod_{z=1}^{Z}\prod_{s=1}^{S}\frac{\sum_{w=1}^{W}B(n_w^{a_d,z,s}+\omega)}{\sum_{w=1}^{W}B(n_w^{a_d,z,s,-d}+\omega)} \end{aligned} \tag{1}$$

$$P(a_d|a_{1:\vec{d}-1}) \propto \begin{cases} \frac{n_{a_d}^{r,-d}+\alpha}{\sum_{a\in\lambda_d} n_a^{r,-d}+|\lambda_d|*\alpha} & \text{if } a_d \in \lambda_d \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $B(\vec{x})$ is the multidimensional extension of the Beta function. The notation $n_a^{b,-c}$ refers to the number of times a has been assigned to b excluding current occurrence c, e.g. $n_{a_d}^{r,-d}$ denotes the number of documents in review r that has been assigned aspect a_d excluding current document d.

The target aspect a_d is dependent on the aspects sampled for the 1^{st} to $(d-1)^{th}$ documents of the review, denoted by $a_{1:\vec{d}-1}$. We restrict the target aspect a_d to belong to the set defined by Λ_d of the document d to achieve coherence among aspects respecting the nature of discourse observed in review writing styles. This constrained aspect sampling differentiates Author-ATS from existing topic modeling works on review text by explicitly modeling the topic coherence of opinionated text.

After sampling the aspect for the document, we jointly sample the latent type, topic and sentiment for each word within the document. The posterior for the i^{th} word of document d (written by author u for entity e) is given as:

$$P(y_i, z_i, s_i | a_d, \vec{w}, \vec{y}_{-i}, \vec{z}_{-i}, \vec{s}_{-i}) \propto P(y_i|d) * P(z_i|a_d, d)$$
$$* P(s_i|a_d, z_i, u, e, d) * P(w_i|y_i, a_d, z_i, s_i,) \qquad (3)$$

$$\propto \frac{n_{y_i}^{d,-i} + \delta_{y_i}}{\sum_{y=0}^{1}(n_y^{d,-i} + \delta_y)} * \frac{n_{z_i}^{d,a_d,-i} + \beta}{\sum_{z=1}^{Z} n_z^{d,a_d,-i} + Z\beta} *$$
$$\Big(q_1 \frac{n_{s_i}^{u,a_d,z_i,-i} + \gamma^0}{\sum_{s=1}^{S} n_s^{u,a_d,z_i,-i} + S\gamma^0} + q_2 \frac{n_{s_i}^{e,a_d,z_i,-i} + \gamma^1}{\sum_{s=1}^{S} n_s^{e,a_d,z_i,-i} + S\gamma^1}\Big)$$
$$* \frac{n_{w_i}^{\zeta,-i} + \omega}{\sum_{w=1}^{W} n_w^{\zeta,-i} + W\omega} \qquad (4)$$

$$y_i = 0 \Rightarrow \zeta = y_i$$
$$y_i = 1 \Rightarrow \zeta = a_d, z_i, s_i$$

For sampling sentiment, instead of using a single Dirichlet density we use a Dirichlet mixture as the prior (Sjölander et al., 1996; Smucker et al., 2005). It is a weighted combination of two individual Dirichlet densities χ and ξ. Mixture coefficients q_1, q_2 are set to 0.5, giving equal weights to both author and entity. This ensures that the chosen sentiment reflects both the entity's quality for that topic as well as the author's preferences.

3.4 Non-parametric Author-ATS (DP) Model

While the number of aspects for a domain are limited, the number of topics for each aspect may vary significantly and can be difficult to estimate. For restaurants, the topics for *ambiance* are fewer (e.g. music, crowd etc.) compared to *food*. This motivates us to propose a non-parametric version of the Author-ATS model where the number of topics can be automatically discovered.

In this non-parametric version, topic inference is done through Chinese Restaurant Process (CRP), a popular variant of Dirichlet Process (DP). In a Chinese restaurant with infinite number of tables, each with infinite capacity, CRP determines if a customer chooses to sit at an occupied table (with a probability proportional to the number of customers already sitting at the table), or an unoccupied one. Following the idea of CRP, each observed aspect dependent word can either be assigned to an existing topic or to a new topic. The conditional distributions for the Gibbs sampler are omitted due to space constraints.

4 SURF Framework

Given a target sentence in a review SURF computes its similarity with other review sentences using the distributions learned by Author-ATS and returns a list of supporting reviews.

A sentence supports another sentence if they are either lexically or semantically similar. Two sentences are **lexically similar** if they share keywords that are important for an aspect. Whereas two sentences can be **semantically similar** if they share the same sentiment for an aspect and topic even though they use different words. For example, *"The hotel was quite close to space needle"* and *"Major attractions are just walking distance from the hotel"* have high semantic similarity as they both talk about the same aspect 'location' on the topic 'attractions' with a positive sentiment.

We treat each review sentence as a vector and lexical similarity ($lexical\_sim$) is computed as cosine-similarity between the two vectors. The i^{th} entry of a vector signifies importance of the corresponding word to its assigned aspect computed using the *tf-idf* weighting scheme. We define the tf-idf of a word w w.r.t. an aspect a as:

$$tf(w,a) = \sum_{d=1}^{D} P(w|d,a)$$

$$P(w|d,a) = \begin{cases} P(w) & \text{if } w \text{ assigned to } a \text{ in } d \\ 0 & \text{otherwise} \end{cases}$$

$$idf(w, \mathbf{A}) = log \frac{A}{1 + |a \in \mathbf{A} : \exists d \in \mathbf{D}, P(w|d,a) > 0|}$$

$P(w)$ is the generation probability obtained from Author-ATS model. Since words are important with respect to an aspect, unlike traditional tf-idf, these values are computed across reviews on the whole corpus. Words frequently used for describing an aspect often tend to converge across reviews, even though written by different users.

Two sentences are considered semantically similar if they share the same sentiment for an aspect. Let C be the set of words in a sentence. Aspect-topic probability of a sentence is defined as the ratio of generation probability of words generated from the aspect-topic pair a, z to the summation of generation probabilities of all the words.

$$P(C|a,z) = \frac{\sum_{w \in C} P(w|w \text{ has aspect } a \text{ and topic } z)}{\sum_{w \in C} P(w)}$$

We define sim_0 to measure the similarities between two sentences (C_1 and C_2) having the same aspect, topic and sentiment, and sim_1 to measure the similarities of two sentences with the same aspect and sentiment but discussing different topics.

$$sim_0(C_1, C_2, a) = \sum_{z=1}^{Z} P(C_1|a,z)P(C_2|a,z)$$
$$sim_1(C_1, C_2, a) = \sum_{z_1, z_2 \in [1 \cdots Z] z_1 \neq z_2} P(C_1|a,z_1)P(C_2|a,z_2)$$

The semantic similarity between two sentences is:

$$\begin{aligned} semantic\_sim(C_1, C_2, a) = {} & sim_0(C_1, C_2, a) \\ & + \delta sim_1(C_1, C_2, a) \end{aligned}$$

where δ is a damping factor with value less than 1.

Lexical-semantic similarity (LSS) of two sentences with same sentiment for an aspect is measured as a weighted combination of their $lexical\_sim$ and $semantic\_sim$ as defined above.

Ranking of Reviews. Given a review sentence, we employ kNN search to find the k most similar sentences for each of its aspects according to LSS measure. Since a target sentence C may contain multiple aspects, we determine the importance of an aspect a to C as follows:

$$Imp(C,a) = \frac{\sum_{w \in C} P(w|w \text{ has aspect } a)}{\sum_{w \in C} P(w)}$$

For each aspect a with $Imp(C,a) > 0$, we return the top $k * Imp(C,a)$ sentences from the review corpus. Proportionately allocating supporting sentences from each aspect in the top-k results diversifies the result set and ensures that a user is able to find information about whichever aspect of the target sentence she wished to verify.

5 Experiments

Dataset	# entity	# author	# review	# sentence	# vocab
TripAdvisor	12,773	781,403	1,621,956	20,244,293	980,323
Yelp	578	16,981	25,459	232,107	56,200

Table 2: Statistics of datasets used

We perform two sets of experiments to evaluate our proposed framework. We first compare Author-ATS with state-of-the-art topic models using perplexity on test data. Then we evaluate the performance of SURF, for the task of retrieving supporting opinions using human annotation, against keyword based search engine Lucene and a competent word embedding model Word2Vec. We use two real world datasets: (a) hotel reviews from TripAdvisor (Wang et al., 2010), and (b) restaurant reviews from yelp.com. Table 2 shows the statistics of the two datasets.

We pre-process both datasets by removing domain independent stopwords[1]. We retain some negation stopwords (e.g.: *not, can't, didn't*) and join them with the next word (so that 'not good' is treated as a single unit) to help discover sentiment properly. We use common punctuations like '.', '?', '!' to split into sentences. To further split a sentence into segments we use punctuations used to separate clauses like ',', ';' and conjunctions like 'and', 'however', 'but' as separators. We use a domain independent subjectivity lexicon[2] to initialize sentiment distributions. Since aspect words may consist of highly co-occurring words (e.g. 'front-desk', 'walking distance') we use Pointwise Mutual Information (PMI) (Manning and Schütze, 1999) to find such collocations. Bigrams with PMI greater than a threshold (we use 0.05 in our experiments) are treated as a single word.

To make the discovered aspects understandable and intuitive, we provide a few seed words to the models. The seeds are only used during initialization and subsequent iterations of Gibbs sampling are not dependent on them. Table 3 lists the aspect seed words used for both domains.

Aspects	Seed Words
Value for Money	value, rate, price
Room	room, bed, bathroom, clean
Location	location, walk, minute
Service	staff, reservation, front-desk
Food	restaurant, breakfast, buffet
Amenities	pool, parking, internet, wifi

(a) TripAdvisor Dataset

Aspects	Seed Words
Value for Money	value, rate, portions, price
Service	ambience, wifi, music, service
Food	steak, rice, burger, cocktail

(b) Yelp Dataset

Table 3: Sample Aspect Seed Words

[1] http://www.ranks.nl/stopwords

[2] http://mpqa.cs.pitt.edu/lexicons/subj_lexicon

5.1 Evaluation of Author-ATS Model

In this set of experiments, we examine the ability of Author-ATS to capture the opinions in reviews.

Perplexity is derived from the likelihood of unseen test data and is a standard measure for evaluating topic models. The lower the perplexity, the less confused the model is on seeing new data, implying a better generalization power. We compare with the following state-of-the-art opinion models:

LDA (Blei et al., 2003) : A topic model where words are generated from a latent topic dimension.

TAM (Paul and Girju, 2010): A topic model for opinion mining where words are generated from a two-level hierarchy of aspect and topic.

JTV (Trabelsi and Zaiane, 2014): A topic model especially for contentious documents where each word has a topic and a viewpoint.

We also implement a baseline model **ATS** based on three-level Aspect-Topic-Sentiment hierarchy. We use this model to show the performance gain by just considering a hierarchical dependency between these dimensions while capturing an opinion. For Author-ATS and ATS, we use 6 aspects, 5 topics for each aspect and 3 sentiments. For fair comparison, we keep the total number of dimensions as close as possible across models. We partition our dataset into train (80%) and test (20%) sets and report five fold cross validation results.

Table 4 shows that ATS outperforms other models in both datasets due to its hierarchical modeling of words. Author-ATS further improves the performance by considering author and entity characteristics as well as the thought patterns of the authors. We note that the performance of the non-parametric model is comparable with Author-ATS, making it easier to use the model for any new domain without having much prior knowledge.

Model	**TripAdvisor**	**Yelp**
LDA	5070	5737
TAM	2980	3468
JTV	3430	4370
ATS	2385	3337
Author-ATS	**2212**	**2784**
Author-ATS(DP)	2300	2829

Table 4: Perplexity values for different models.

Table 5 shows the top words extracted by Author-ATS as domain stopwords. Although these words do not convey any aspect information, they are domain dependent and are not found in a general stopword dictionary.

From Table 6, we observe that the majority of the words are correctly clustered in aspects, and further into specific topics. For example, the first topic for aspect *Room* is about in-room experience ('bed','king-size','view'), whereas the second topic seems to be about bathroom ('shower', 'towels', 'tub'). We also observe that the model is able to obtain contextual sentiment terms which are aspect-topic coherent. For example, words such as 'noise', 'night', 'hear' could be assigned negative sentiment labels for topic 0 of *Room* due to the context in which they are used, e.g., when describing a room, these words probably indicate a noisy room bothering their sleep at night.

Dataset	Domain Stopwords
TripAdvisor	hotel, nice, stay, trip, times, day, place, back
Yelp	good, place, food, time, order, bit, make

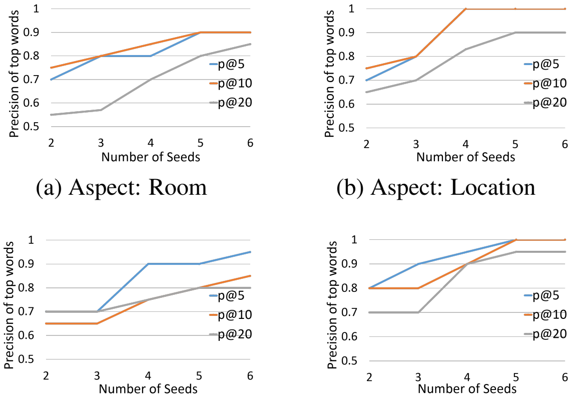
Table 5: Domain stopwords from Author-ATS.

Aspect: **Room**		
Topic 0		
Positive	Negative	Neutral
bed	noise	room
comfortable	night	floor
spacious	sleep	view
king-size	window	size
clean	hear	modern
Topic 1		
Positive	Negative	Neutral
bathroom	small	room
large	door	bathroom
tub	barely	shower
shower	tiny	water
shampoo	kitchen	towels

Aspect: **Service**		
Topic 0		
Positive	Negative	Neutral
staff	night	staff
extremely	greet	call
welcoming	problem	front-desk
care	asked	service
friendly	manager	shuttle
Topic 1		
Positive	Negative	Neutral
card	called	check-in
reservation	upgrade	day
airport	manager	arrived
polite	rude	directions
excellent	questions	time

Table 6: Top words for aspect-topic-sentiments found by Author-ATS for TripAdvisor dataset.

Impact of Seed Words We vary the number of seed words for an aspect and examine its effect on the aspect discovery. We use $p@n$, the fraction of correctly discovered aspect words among the top n words, to evaluate the quality of the results.



(a) Aspect: Room

(b) Aspect: Location

(c) Aspect: Service

(d) Aspect: Food

Figure 4: Impact of varying number of seeds.

The average precision of top-n words for different aspects is obtained by taking the average over all combinations $\binom{6}{m}$ of seed words where m is the number of selected seed words, $2 \leq m \leq 6$. Figure 4 shows the results. We observe that the average precision increases with the number of seeds, and stabilizes when $m \geq 4$. This demonstrates that providing a handful of seed words can go a long way for discovering intended, explainable domain specific aspects.

5.2 Evaluation of SURF

We now evaluate Author-ATS model and LSS measure on retrieving sentences that are *relevant* to a target sentence. A sentence is considered relevant if it expresses similar opinions as the target sentence. A sentence with multiple aspects is relevant if it expresses at least one of the opinions in the target sentence. Precision of the top-k answers are manually determined by three annotators and conflicts are resolved by majority voting.

Recall that LSS considers both lexical and semantic similarity. The computation of semantic similarity requires the aspect-topic-sentiment distribution which is only available in the baseline ATS and Author-ATS models. We define a similarity measure called CJSD that can be used by the various topic models to facilitate comparison. CJSD measures the lexical similarity of two sentences as the cosine similarity of their tf-idf vectors, while the semantic similarity is measured by the similarity of their topic distributions using Jensen-Shannon Divergence(JSD) as follows:

$$CJSD(s_1, s_2) = \lambda\, cosine\_sim(s_1, s_2) + (1-\lambda)\, JSD(s_1, s_2)$$

We randomly select 5 hotels from TripAdvisor and 5 restaurants from Yelp datasets. For each hotel/restaurant, we randomly pick 10 target sentences and retrieve their supporting sentences.The topic distributions of these sentences are obtained using LDA, TAM, JTV, and the proposed models ATS and Author-ATS.

Table 7 shows the average precision for top 5, 10 and 20 results retrieved using various topic models with similarity measure CJSD. We see that Author-ATS model always outperforms other topic models for the task of retrieving supporting sentences. This is consistent with the perplexity results of the models obtained previously.

Table 8 shows the average precision using variants of the proposed model with LSS. Clearly, using LSS always yields a better precision compared to using CJSD, with the best performer being the Author-ATS with LSS combination. SURF framework utilizes this combination for retrieving top-k supporting reviews.

	TripAdvisor			Yelp		
	p@5	p@10	p@20	p@5	p@10	p@20
LDA	0.56	0.48	0.45	0.43	0.42	0.42
TAM	0.58	0.53	0.52	0.49	0.47	0.47
JTV	0.51	0.47	0.53	0.41	0.41	0.43
ATS	0.62	0.60	0.55	0.60	0.57	0.44
Author-ATS	**0.68**	**0.62**	**0.61**	**0.60**	**0.58**	**0.56**

Table 7: Average precision using CJSD

	TripAdvisor			Yelp		
	p@5	p@10	p@20	p@5	p@10	p@20
ATS	0.69	0.62	0.58	0.62	0.59	0.58
Author-ATS	**0.74**	**0.66**	**0.60**	**0.68**	**0.64**	**0.62**
Author-ATS (DP)	0.64	0.63	0.57	0.62	0.56	0.54

Table 8: Average precision using LSS

Next, we compare SURF with the following:

Lucene: A popular keyword based ranking method. We used its default combination of vector space model and boolean model for retrieval.

Word2Vec: (Mikolov et al., 2013) A state-of-the-art algorithm for word embeddings using neural network. Supporting sentences are ranked with Word Mover's distance using the word embeddings. We train on TripAdvisor dataset using CBOW algorithm with context window set to 5 as recommended by the authors. We do not train Word2Vec on the Yelp dataset as it is too small. We set the vector dimension to 500 based on grid search. We also compare with Word2Vec model pre-trained on the large GoogleNews dataset[3].

Table 9 shows the average precision for the top 5, 10 and 20 results retrieved using Lucene, Word2Vec and SURF. Word2Vec performs better when trained on review data, compared to the model trained on general news data. This confirms that domain knowledge is important. It is evident from the results that SURF significantly outperforms existing approaches for opinion search.

	p@5	p@10	p@20
Lucene	0.67	0.58	0.52
Word2Vec (GoggleNews)	0.62	0.48	0.39
Word2Vec (TripAdvisor)	0.70	0.61	0.51
SURF	**0.74**	**0.66**	**0.60**

(a) TripAdvisor

	p@5	p@10	p@20
Lucene	0.61	0.54	0.49
Word2Vec (GoogleNews)	0.52	0.47	0.37
SURF	**0.68**	**0.64**	**0.62**

(b) Yelp

Table 9: Comparison with Lucene and Word2Vec

For evaluating the coherence of retrieved set of supporting reviews for an aspect, we look at their corresponding user given aspect ratings. For each aspect of each review sentence, we retrieve its top-k supporting sentences. Then we compute the

[3] https://code.google.com/archive/p/word2vec/

Target Sentence: bedroom had the most comfortable mattress, feather soft pillows as well as firmer ones, they thought of keeping every guest comfortable		
Supporting Sentences by **SURF**	Supporting Sentences by **Lucene**	Supporting Sentences by **Word2Vec**
Aspect : **Room** **Statement**: bill clinton suite was huge with two baths, a wonderful jacuzzi and a comfortable bed	bed was very comfortable, as were the large pillows	The room had a microwave, coffemaker, hairdryer, bottled water replenished each day **(x)**
Aspect : **Room** **Statement**: the beds are the most comfortable of any hotel I have stayed in	we were recommending it for our out of town wedding guests, and wanted to make sure they were comfortable **(x)**	It really is a shame because the bed and pillows were super comfortable and we could have had a great night sleep on both nights
Aspect : **Room** **Statement**: the beds were comfortable and they had good selection of towels	who would have imagined that somebody actually thought about where a guest would watch tv **(x)**	They took regular sized hotel rooms and divided them into a sitting room with a bedroom with a door, keeping the bathroom to divide the two areas **(x)**

(a) Target Sentence with Single Aspect

Target Sentence: the check in was quick, with friendly polite service, and the room was very big with a very comfortable king size bed		
Supporting Sentences by **SURF**	Supporting Sentences by **Lucene**	Supporting Sentences by **Word2Vec**
Aspect : **Room** **Statement**: bed was extremely comfortable, I'm hard to please in the department because I sleep on a sleep number bed at home	the room was a great size; bed was very comfortable	The first room assigned was very small and dingy with one king sized bed that just fit **(x)**
Aspect : **Room** **Statement**: room size was large and bed was comfortable	king size bed was comfy	bathroom was well furnished with soap, shampoo/ conditioner, very large, soft towels - perfect **(x)**
Aspect : **Service** **Statement**: service is very friendly	our room faced denny park **(x)**	the room was large and the bed very comfortable and our room faced the street and it was very quiet

(b) Target Sentence with Multiple Aspects

Table 10: Sample Supporting Sentences Retrieved by SURF, Lucene and Word2Vec. Aspects shown for SURF are discovered by Author-ATS model.

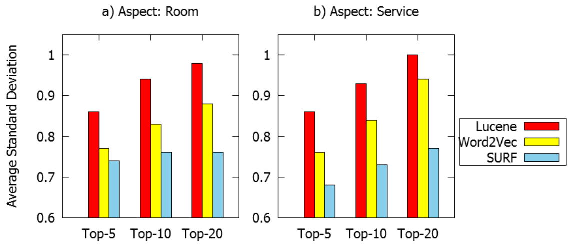


Figure 5: Average standard deviations of aspect ratings for supporting reviews. Smaller deviation implies greater coherence.

standard deviation of the ratings for that aspect in the retrieved supporting reviews. We aggregate the standard deviation values for each aspect over all the reviews and look at the average value. Figure 5 shows results for two aspects from the TripAdvisor dataset. Other aspects also had similar trends.

We rank the retrieved results based on their similarity to the target sentence. Naturally, the longer the retrieved list, the larger is the average standard deviation. We see that SURF has a smaller average standard deviation compared to Word2Vec and Lucene. The gap between the performance of SURF and the other methods also widens as the size of the retrieved results increases. This demonstrates SURF's superiority in retrieving reviews with similar opinions.

Table 10 shows samples of supporting sentences extracted by the different methods. We observe that the sentences retrieved by SURF are semantically similar although the words may be quite different from the target sentence. In contrast, Lucene may retrieve irrelevant sentences matching a keyword used in a totally different context. Word2Vec considers words used in proximity of one another (e.g. *bed, pillow* with *microwave, coffemaker* etc.) to be similar which clearly does not always imply conformity of opinions.

Furthermore, the retrieved results of SURF are categorized according to their aspects making them easy to interpret. Particularly if a target sentence has multiple aspects, then SURF will retrieve results for each aspect. For example, for the second target sentence shown in Table 10, the results contain supporting statements for both *room* and *service*. If a user then wishes to view more results for one of those aspects it will be possible for SURF to fetch more results only for that aspect.

6 Conclusion

We studied the problem of finding supporting sentences to help a user get an idea of consensus about an entity. To this end, we developed a hierarchical topic model to jointly infer aspect-topic-sentiment, and a fine-grained similarity measure. Author-ATS model encodes the coherent writing style of a review by constraining the aspect distributions dynamically. It considers the sentiment distribution of a review to have influence of both the author and the entity. Experimental results on two datasets indicate that the proposed approach is promising compared to existing techniques. With growing amount of user generated content on the web, and more people relying on them to make decisions, we believe that the ability to verify opinions will become increasingly important.

References

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Exploiting domain knowledge in aspect extraction. In *Proc. of EMNLP*.

Lan Du, Wray Buntine, and Huidong Jin. 2010. Sequential latent dirichlet allocation: Discover underlying topic structures within a document. In *Proc. of IEEE ICDM*.

Lan Du, John K Pate, and Mark Johnson. 2015. Topic segmentation with an ordering-based topic model. In *Proc. of AAAI*.

Amit Gruber, Yair Weiss, and Michal Rosen-Zvi. 2007. Hidden topic markov models. In *International Conference on Artificial Intelligence and Statistics*.

Eduard H Hovy. 1993. Automated discourse generation using discourse structure relations. *Artificial intelligence*, 63(1):341–385.

Yohan Jo and Alice H Oh. 2011. Aspect and sentiment unification model for online review analysis. In *ACM International Conference on Web Search and Data Mining*.

Suin Kim, Jianwen Zhang, Zheng Chen, Alice H Oh, and Shixia Liu. 2013. A hierarchical aspect-sentiment model for online reviews. In *Proc. of AAAI*.

Fangtao Li, Sheng Wang, Shenghua Liu, and Ming Zhang. 2014. Suit: A supervised user-item topic model for sentiment analysis. In *Proc. of AAAI*.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proc. of ACM CIKM*.

Christopher D Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT press.

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

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proc. of EMNLP*.

Samaneh Moghaddam and Martin Ester. 2011. Ilda: interdependent lda model for learning latent aspects and their ratings from online product reviews. In *Proc. of SIGIR*.

Arjun Mukherjee and Bing Liu. 2012. Aspect extraction through semi-supervised modeling. In *Proc. of ACL*.

Michael Paul and Roxana Girju. 2010. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *Proc. of AAAI*.

Lahari Poddar, Wynne Hsu, and Mong Li Lee. 2017. Quantifying aspect bias in ordinal ratings using a bayesian approach. In *Proc. of IJCAI*.

Md Mustafizur Rahman and Hongning Wang. 2016. Hidden topic sentiment model. In *Proc. of World Wide Web*.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. 2009. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proc. of EMNLP*.

Kimmen Sjölander, Kevin Karplus, Michael Brown, Richard Hughey, Anders Krogh, I Saira Mian, and David Haussler. 1996. Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology. *Computer applications in the biosciences: CABIOS*.

Mark D Smucker, David Kulp, and James Allan. 2005. Dirichlet mixtures for query estimation in information retrieval. *Center for Intelligent Information Retrieval*.

Ivan Titov and Ryan McDonald. 2008a. Modeling online reviews with multi-grain topic models. In *Proc. of WWW*.

Ivan Titov and Ryan T McDonald. 2008b. A joint model of text and aspect ratings for sentiment summarization. In *Proc. of ACL*.

Amine Trabelsi and Osmar R Zaiane. 2014. Mining contentious documents using an unsupervised topic model based approach. In *Proc. of IEEE ICDM*.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: a rating regression approach. In *Proc. of SIGKDD*.

Zaihan Yang, Alexander Kotov, Aravind Mohan, and Shiyong Lu. 2015. Parametric and non-parametric user-aware sentiment topic models. In *Proc. of SIGIR*.

Wei Zhang and Jianyong Wang. 2015. Prior-based dual additive latent dirichlet allocation for user-item connected documents. In *Proc. of IJCAI*.

Tong Zhao, Chunping Li, Qiang Ding, and Li Li. 2012. User-sentiment topic model: refining user's topics with sentiment information. In *ACM SIGKDD Workshop on Mining Data Semantics*.

Magnets for Sarcasm:
Making Sarcasm Detection Timely, Contextual and Very Personal

Aniruddha Ghosh
University College Dublin, Ireland.
Aniruddha.Ghosh@ucdconnect.ie

Tony Veale
University College Dublin, Ireland.
Tony.Veale@ucd.ie

Abstract

Sarcasm is a pervasive phenomenon in social media, permitting the concise communication of meaning, affect *and* attitude. Concision requires wit to produce and wit to understand, which demands from each party knowledge of norms, context and a speaker's mindset. Insight into a speaker's psychological profile at the time of production is a valuable source of context for sarcasm detection. Using a neural architecture, we show significant gains in detection accuracy when knowledge of the speaker's mood at the time of production can be inferred. Our focus is on sarcasm detection on Twitter, and show that the mood exhibited by a speaker over tweets leading up to a new post is as useful a cue for sarcasm as the topical context of the post itself. The work opens the door to an empirical exploration not just of sarcasm in text but of the sarcastic state of mind.

1 Introduction

Oscar Wilde memorably described sarcasm as "the lowest form of wit but the highest form of intelligence." Though sarcasm lacks the sophistication of irony, and does little to conceal the speaker's disdain for a target, it is a figurative device that requires as much intelligence from its consumers as its producers. The concision with which sarcasm and irony allow speakers to conflate propositional content and affective stance makes it a pervasive mode of communication in the 140-character *tweets* of Twitter. By combining an overtly positive attitude with a meaning that is more deserving of scorn, sarcasm allows speakers to communicate disappointment about a state of affairs that bites (or etymologically "cuts the flesh") of an addressee. It conveys the feeling the speaker would wish to experience ("I love it when ...") with the state of affairs that up-ends this feeling ("... my friends forget my birthday"). It often combines politeness with mockery to disguise the appearance of hostility while heightening its effect on a listener (Brown and Levinson, 1978; Dews and Winner, 1995). It establishes a wry environment (Dews and Winner, 1999) that has its roots in social norms *and* the speaker's state of mind.

Psychological theories of irony, such as *echoic reminder theory* (Kreuz and Glucksberg, 1989) and *implicit display theory* (Utsumi, 2000b) have yet to fully translate into text-analytic methods. Neuropsychology researchers who have sought patterns of brain activity to identify the neural correlates of sarcasm note that an understanding of sarcasm is highly dependent not just on the context of an utterance but on the state-of-mind and personality of the speaker, as well as on facial expressions and prosody (Shamay-Tsoory et al., 2005). Without the latter markers, purely textual detection must depend largely on the content and context of an utterance, though speaker personality and state-of-mind can also be approximated via text-analytic means. Probabilistic classification models that exploit textual cues – such as the juxtaposition of positive sentiment and negative situations (Riloff et al., 2013), discriminative words and punctuation marks (Davidov et al., 2010), and emoticon usage (González-Ibánez et al., 2011) – have achieved good performance across domains, yet these models typically suffer from an absence of psychological insight into a speaker and topical insight into the context of utterance production. Kreuz and Link (2002) argue that the likelihood of sarcasm is proportional to the amount of knowledge shared by speaker and audience, which includes knowledge of the world *and* knowledge of the speaker and audience. Personality is defined

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 482–491
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

by Olver and Mooradian (2003) as the "enduring characteristics of the individual" though *mood* – which is changeable – is perhaps just as useful if sampled in a timely fashion. The difference between personality and mood can be likened to that between climate and weather. Tausczik & Pennebaker (2010) have developed a Twitter-based mood analysis web service at *AnalyzeWords.com* which uses a variety of psycholinguistic criteria and the LIWC (*Linguistic Inquiry and Word Count*) resource[1] to quantify the recent mood – i.e. the *recent weather* – of a user along 11 dimensions ranging from Arrogance/Remoteness to Anger and Analyticity. To exploit the stable personality of an online user, Celli et al. (2016) sought a correlation between *Big Five* personality traits (Costa and McCrae, 2008) and the LIWC-quantifiable dimensions found in re-tweets amongst Twitter users. (Rajadesingan et al., 2015) have also shown how relevant aspects of personality can be acquired from a speaker's past tweets. Since personality and mood can each influence the detection process, they underpin our first research question: To what extent can the quantifiable dimensions of either lead to a better understanding of sarcasm? Reliable detection depends as much on the context of an utterance – which provides the motivation for sarcasm – as its content. Consider e.g.:

Speaker Utterance: @MSNBC of course all of those jobs will be in China
In reply to *@realDonaldTrump*: I will be the greatest jobs-producing president that God ever created.

The speaker's sarcastic intent cannot be grasped without knowledge of the larger context. This issue provides our second research question: How can we usefully incorporate utterance context into a neural network model of sarcasm detection? Sarcasm is ubiquitous but always in flux, relying on a changing swirl of socially relevant viewpoints. The following tweet is sarcastic by virtue of its echoic mockery of a widely ventilated opinion:

Time to get my Sunday dose of #fakenews from the failing @nytimes.

This begs the third research question that we explore in the following sections: How can we train our sarcasm detection model to exploit evolving social norms and public opinions?

2 Related Work and Ideas

Sarcasm has been extensively researched by linguists and psychologists (Gibbs and Clark, 1992; Gibbs and Colston, 2007; Kreuz and Glucksberg, 1989; Utsumi, 2000a), yet due to the limited availability of stimuli, sarcasm detection in text has relied chiefly on the recognition of stock patterns and lexical cues. Sarcasm often highlights failed expectations by engaging in a pragmatic pretense that is designed to be seen through (Campbell and Katz, 2012), so cues such as interjections, intensifiers, punctuation and markers of non-veridicality and hyperbole play a crucial role in recognizing sarcastic intent. Likewise, stock plaudits such as "yay!" or "great!" are common in sarcastic product reviews (Tsur et al., 2010), while hashtags such as *#sarcasm*, as compressed vehicles for user intent, are often used to self-annotate sarcastic texts (Davidov et al., 2010). Liebrecht et al. (2013) used topic-specific information and n-grams as discriminative features, while (Lukin and Walker, 2013) showed that phrases such as "no way", "Oh really?" and "not so much" serve to flag a sarcastic intent when used with specific linguistic patterns.

Capelli et al. (1990); Woodland and Voyer (2011) suggest that contextual awareness is a necessary precursor to identifying sarcasm. Sarcasm is a response to a motivating context that appears to force a rueful incongruity between a text and its context. Exploiting the *principle of inferability* (see Kreuz (1996)), Bamman and Smith (2015) modeled shared common knowledge by extracting features from context, the author, and the audience. Khattri et al. (2015) identified sarcasm by seeking a strong contrast in affect toward named entities in current vs. historical tweets, while Rajadesingan et al. (2015) also exploited a contrast in statistically-derived author traits across current and historical tweets. Zhang et al. (2016) use similar sources of contextual information to show the effectiveness of a neural network over more traditional approaches involving manually-selected, discrete features, claiming that automatic feature induction can uncover more subtle markers of sarcasm. Amir et al. (2016) argue that sarcasm detection hinges on speaker modeling, and exploited *user embedding* to quantify incongruity between utterances and the behavioral traits of their authors. These methods measure the disparity between an utterance and expectations arising from knowledge of context or speaker or both together.

[1] https://liwc.wpengine.com/

We build on this double-grounding for sarcasm to improve detection in a neural network model of sarcasm and thereby address our first two research questions. We model the speaker *at the time of utterance production* using mood indicators derived from the most recent prior tweets, and model context using features derived from the proximate cause of the new utterance, the tweet to which an utterance is a response. For our third research question, we present a novel feedback-based annotation scheme that engages authors of training/test tweets in a process of explicit annotation, feeding new examples back into the model. Section 3 outlines the kind and source of features exploited in the model. Section 4 outlines our methods of data collection and annotation. Section 5 presents the neural network model, while section 6 & 7 present our experimental set-up and analysis of results. Finally, section 8 offers some closing remarks.

3 Psychological dimensions and Sarcasm

We cannot perceive a user's state-of-mind directly on Twitter, but we might infer one's current disposition from an analysis of recent tweets, as linguistic expressions tend to be congruent with an author's state-of-mind (Campbell and Katz, 2012). An informative if *low-res* psychological portrait is sketched by web services such as *AnalyzeWords* (Tausczik and Pennebaker, 2010), which analyzes the most recent 1000-words or so of a Twitter user using LIWC to score the user on 11-dimensions: *Upbeat, Worried, Angry, Depressed, Plugged in, Personable, Arrogant, Spacy, Analytic, Sensory* and *In-the-moment*. Sarcasm is often perceptible in the incongruity between utterance and context (Joshi et al., 2015) but it can also be conveyed by an incongruity between text and recent mood.

To understand the relationship between these 11 dimensions (each scored 0..100) and a propensity for sarcasm, we performed a k-Nearest Neighbors (KNN) clustering of the Twitter users that provide the tweets of our sarcastic data set. The *AnalyzeWords* snapshot of each user was taken at the time of that user's tweet in the dataset. A value of 30 for k was chosen empirically to ensure a decent size for the clusters. By calculating Spearman correlations between each group and the 11 *AnalyzeWords* dimensions, we estimated the affinity for sarcasm of different dimensions. Unsurprisingly, we observed that clusters showing a high correlation with negative dimensions, such as *Angry*, also tend to use positive expressions such as 'funny" and 'wow" to mark sarcasm. Here is an example:

@realDonaldTrump They can all fit in your head? Wow! Have you seen someone about this?

Unless one knows that @*realDonaldTrump* often elicits anger, or that the author scored 83 (of 100) for *Angry*, this tweet might seem quite positive. Valence shifters such as "not" might also suggest literal positivity if not for the implicit anger of the author. At the time of the following tweet *AnalyzeWords* scored its author as *Angry=98*.

@realdonaldtrump funny the founder of the birther movement is saying that he's not racist #trumpbirther

Polarizing figures such as @*realDonaldTrump* are magnets for sarcasm on Twitter. By identifying these magnets, we can better detect the sarcasm of a tweet that offers plaudits for negative qualities. We use *AnalyzeWords* to obtain the popular affective feelings for common addressees by averaging the affective dimensions of the users that tweet at them. The top 5 magnets for sarcasm in our data-set of 18K sarcastic tweets are @*hillaryClinton, @realDonaldTrump, @bernieSanders, @AP* and @*megynKelly*. Of these, @*hillaryClinton* is the biggest target for *Angry* tweets while @*megynKelly* is the biggest target for *Analytic* tweets.

Addresses in the political domain score high for both angry tweets and analytic tweets: people analyze the news *and* shoot the messenger. We see much less analyticity – a tendency to use complex expressions linked with logical connectives – in tweets about popular entertainers. To mock such targets, users tend to use affective words that contrast with overall public opinion. The magnet with the highest mean *Angry* score for the tweets that target him is @*realDonaldTrump*, yet 63% of the affective words in the tweets that target him in the data-set are positive. Knowing that @*realDonaldTrump* is a magnet for anger can help a sarcasm detector overcome this positive bias.

4 Dataset Construction

Tweets with sarcastic intent are often misclassified due to a lack of shared context or knowledge between speaker and annotator. Opposing social beliefs and a dearth of topical or personal knowledge can lead to serious misjudgments. Relevant tweet sets can be harvested by searching sarcasm specific hashtags (e.g. #sarcasm). This approach overlooks tweets that are not explicitly tagged as sarcastic by their authors. Thus we have devised a feedback-based system that contacts tweet authors directly *after-the fact* to ask for their authoritative self-annotations for a potentially sarcastic tweet.

4.1 Data collection

To collect annotations from authors for *their own* tweets, we used a Twitterbot named *@onlinesarcasm* to exploit the "retweet with comment" function in Twitter. The bot chooses randomly from tweets that are addressed to any of 700 top Twitter users (as listed by *TwitterCounter.com*), as we expect high-profile figures to be magnets for sarcasm from others. The bot retweets a chosen tweet (s_i) to its author, appending a yes/no question (q_i) as a comment to elicit a reply.

At the time of retweeting (s_i), the 11 *AnalyzeWords.com* dimensions (aw_i) of the tweet's author (u_i) are saved, along with the context tweet (s_j) by author (u_j) that provoked (s_i). Authors respond to the bot by favoriting/retweeting the bot's request or via a reply (re_i) containing #Yes or #No. Author responses often contain more than a simple #Yes or #No response, and so, after observing a series of responses the following linguistic rules were used to extract the training annotations:

- If the number of *retweets* (r_i) or *likes* (l_i) for q_i is non-zero or re_i contains #Yes, then s_i is deemed positive for sarcasm.
- If re_i contains #No or an explicit mention of 'not sarcastic' or 'no sarcasm' or 'truth', then s_i is deemed negative for sarcasm.

We discarded any s_i lacking a context tweet s_j. Using author feedback, a data set of 40K tweets was collected, comprising 18K tweets acknowledged as sarcastic and 22K deemed non-sarcastic. For another test set, we collected 1200 tweets: 550 tweets acknowledged as sarcastic by their authors and 650 acknowledged to be non-sarcastic.

4.2 External datasets

In addition to our own training and test sets, whose annotations come directly from tweet authors, we also used 5 Twitter datasets where tweet information, fetched by tweet identifier, contains identifier of context tweet (Ptáček et al., 2014; Bamman and Smith, 2015; Rajadesingan et al., 2015; Cliche, 2014), from which motivating contexts can be discerned for each. (This contextual requirement prevents us from considering even more of the available sarcasm datasets.) For the context tweets s_j for each s_i in these sets we collected the most recent linked tweets of s_i. To obtain the 11 *AnalyzeWords.com* dimensions for tweet authors, we collect the 50 tweets of u_i posted just prior to s_i, and use the LIWC to estimate the 11 dimensions (Anger, Arrogance, etc.) from those tweets. As *AnalyzeWords.com* does not provide retrospective analyses, and as its code is not public, we reverse-engineered a substitute using the LIWC by following the creators' guidelines in (Tausczik and Pennebaker, 2010). For subsequent evaluations, the 5 external datasets were split into 3 parts each: 80% for training, 10% for development/tuning, and 10% for testing.

5 The Neural Network Model

Ghosh and Veale (2016) described an Artificial Neural Network (ANN) model built around layers of CNNs (Convolutional Neural Networks) and LSTMs (Long Short Term Memory) for sarcasm detection to efficiently capture contrasting text signals of sarcasm within a tweet. We build here on this model as shown in Fig.1, adding input features for the psychological profile of the author and the context of the tweet to those for the tweet itself. The LSTM layer (Hochreiter and Schmidhuber, 1997) captures dependencies amongst non-adjacent contrasting signals for sarcasm within each s_i. We extend this architecture to include a context tweet s_j for each s_i, but instead of concatenating s_j and s_i at the input layer, we stitch them together after the LSTM layer. The text input layer is initialized with embeddings from Google's *Word2Vec* model (Mikolov et al., 2013) with a dimension setting of 300. To further integrate features reflecting the state of mind of the speaker at utterance-time, the values aw_i ($i = 1...11$) for each s_i are concatenated with the feature vector of s_j & s_i in the merge layer. We use a bi-directional LSTM (BLSTM) and forego a *maxpooling* layer to increase throughput to the BLSTM. We prevent overfitting using a dropout layer with a dropout rate of 0.25 after the BLSTM layers. The concatenation layer combines the feature maps of the source and context tweets (s_i & s_j) along with a vector of $\mathrm{aw}_{1...11}$ for the author u_i. The concatenation yields a merge layer of size $\Re^{f(2(|s|m+1)+l)}$ where f, s, m and l are, respectively, the number of BLSTM units, the length of the input se-

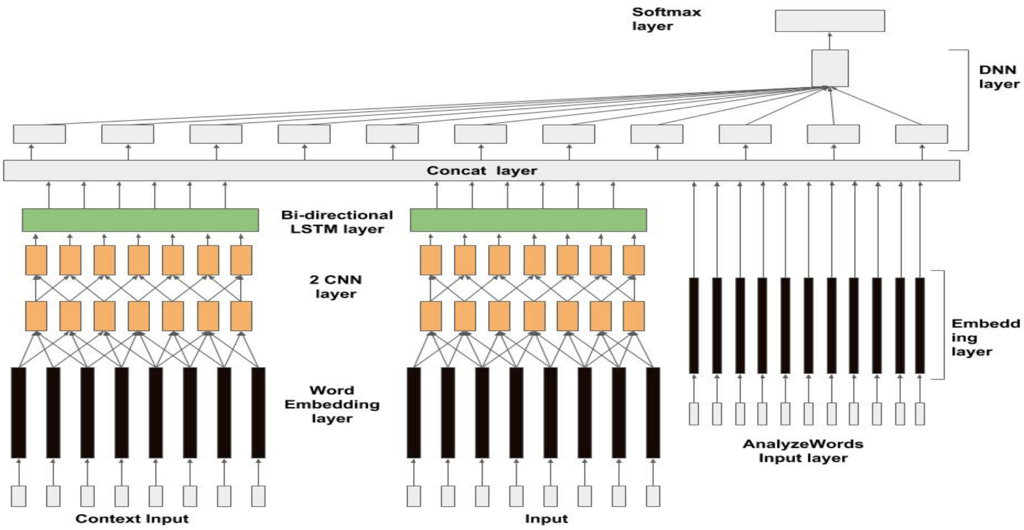


Figure 1: A Neural Architecture for Detecting Sarcasm in Contextualized Utterances

quence, the width of the CNN filter and the length of aw. Notice that the features for a tweet s_i and its immediate context s_j – which we consider the proximate cause of the sarcasm (if any) in s_i – are concatenated only after they have passed through separate sets of CNN and LSTM layers (CNN1 + BLSTM1 and CNN2 + BLSTM2). It is important to keep a tweet and its context separate for as long as possible, as the model is designed to recognize an inherent incongruity between each. This incongruity becomes diffuse if the inputs are combined too soon. EAW is the embedding layer for the 11 *AnalyzeWords* dimensions; it combines the vectors of s_j, s_i and aw, and passes the concatenated features to a Deep Neural Network (DNN) to discriminate both classes (sarcasm vs. non-sarcasm). The code[2] is developed using Keras[3].

6 Evaluation and Experimental Setup

Success with a neural architecture requires apt input features and an equally apt selection of hyper-parameters. After performing a grid search over hyper-parameters, the best configuration of the CNN, LSTM and DNN layers places 1280 hidden memory units into each layer and uses a CNN filter width of 3. A simple baseline will use only the textual content of a tweet s_i without a context s_j or an affective profile aw of the author u_i. To appreciate the contribution of different input sources of information we trained the network on different combinations of these sources.

6.1 Addressee information

If s_i is addressed to u_j, this information can provide additional insights into s_i's tone. In the TTIA *(Target Tweet Including Addressee)* setting the name of the addressee (but not an estimation of the public opinion of the addressee, as so few addresses are actually famous) is added to the baseline along with s_i. If the addressee is a magnet for sarcasm, aspects of this magnetism should still impress themselves on the network during training.

6.2 Contextual Information

In a variant of the baseline called CT *(Context Tweet)*, the features of the tweet s_j to which the target s_i is a response are also added as inputs to the model, to be stitched together with the features of s_i at the concatenation layer. Changes in performance with and without CT will allow us to estimate the value of context in sarcasm detection.

6.3 Author Profile Information

The 11-dimensional *AnalyzeWords* snapshot aw for author u_i at the time s_i is posted offers valuable insights into the intent of u_i. In the PD *(Psychological Dimensions)* configuration, the 11 affective dimensions aw_i are added to the model. They pass through an embeddings layer to be combined with utterance (and possibly context) features at the concatenation layer. To determine the relative contribution of each dimension aw_i to detection competence, we trained the model in two extra modes. In the first, we fed the model with false values for each aw_i, varying values from 0 to

[2] https://github.com/AniSkywalker/SarcasmDetection

[3] https://keras.io/

100, to observe the effects on accuracy when e.g. *Angry* is over- or under-estimated for u_i. In the second extra mode, we excised each aw_i, one at a time in different training runs, to quantify its lack on the model.

6.4 Automatic adaptation

Online sarcasm is often used to comment on the vagaries of politics and current affairs. As topicality is of the essence, we expect a model that regularly acquires new author-annotated training data to bootstrap itself will adapt better to the times and yield better results. To estimate the benefits of bootstrapping we tested the model on an evolving version of the data-set that acquired new training data each week for a month in August 2016.

7 Results & Analysis

Table 3 shows the recall (R), precision (P) and f-score (F1) for our model, called *Sarcasm Magnet*, with alternate configurations on different datasets. The configuration for each setup is given in the second row (e.g. the addition of context tweets requires the use of two LSTMs and two CNNs). Setup TTEA is the baseline which uses only the text of a target tweet; it excludes addressee handles, context tweets (CT) and the psychological dimensions (PD) of authors. Setup TTIA adds addressee handles to the baseline to give our model a small boost, mostly in recall. Setup TTEA+CT adds context tweets to the baseline, yielding a significant boost since a good deal of sarcasm is conversational in nature. In this setup the most significant improvement in recall was observed with the Bamman dataset (Bamman and Smith, 2015). In setup TTIA+CT, which uses context *and* addressee handles, no significant improvement over TTEA+CT is observed, except for precision on Bamman's dataset. In setup TTEA+PD, the affective profile of each author at tweet-time is added to the baseline to yield a significant boost in performance almost as large as that for TTIA+CT. Setup TTIA+CT+PD includes all available information sources (addressee, content, and psychological profile). This column reports (in parentheses) the performance on each dataset by the dataset creator's own system, which is either publicly available or re-implemented from their paper. The results show that *Sarcasm Magnet* beats the state of the art for these data-sets.

Table 1 shows the effect on the model's performance in the absence of specific aw_i values. A boost in recall and a drop in precision shows the bias of the model shifting towards sarcasm when the space of non-sarcastic tweets overlaps with that of sarcastic tweets in the absence of an aw_i that confirms literal intent. So political tweets may be mis-classified as sarcasm in the absence of values for *Angry*, *Depressed*, and *Worried*, suggesting that sarcastic authors often seem less angry, depressed or worried. A drop in precision and recall when *Arrogant/Remote*, *Analytic*, *Plugged in* and *In-the-moment* dimensions are absent suggests sarcastic people to be more socially active and aware, and smarter but more arrogant.

Dimension omitted	**Precision**	**Recall**	**F-score**
None omitted	0.9	0.89	0.9
Sensory	0.85	0.93	0.89
Plugged In	0.84	0.84	0.84
Depressed	0.78	0.96	0.86
Angry	0.81	0.95	0.87
Spacy/Valley girl	0.78	0.97	0.87
Worried	0.79	0.97	0.87
Arrogant/Distant	0.84	0.85	0.84
Analytic	0.86	0.83	0.84
In-the-moment	0.84	0.86	0.85
Upbeat	0.86	0.91	0.88
Personable	0.87	0.88	0.88

Table 1: Performance of the model when a specific dimension aw_i is omitted from training.

7.1 A Tale of Two Contexts

The CT and PD additions each bring significant improvements in F-score, yet when added jointly they bring no significant increases over either used individually. For each is a form of context drawn from different sources that reflects different intuitions but which ultimately offers much the same insights. The impact of the 11 aw dimensions is lower on the 5 external datasets than for the new feedback-based dataset, no doubt because the *AnalyzeWords.com* snapshot of authors in the latter could be taken directly at tweet-time, whilst for the former it was retrospectively approximated using our own jerry-rigged version based on the LIWC. If the official web service were to allow retrospective analyses of Twitter users at specific times we are confident the improvements on the external datasets would mirror those on our own dataset. For now it is interesting to note the effectiveness of the *AnalyzeWords.com* service at affectively profiling Twitter users at specific times, which is to say, at specific contexts in their Twitter time-lines.The service boils down the most recent tweets (approx. 1000 words in total) to 11 dimensions that are more than simple functions of the lexical scores in the LIWC. Rather, it analyzes the selected text as a coherent product of a coherent mind-set, to measure a local propensity for hostility, optimism, depression, emotional detachment and preference for reason. To use our earlier analogy, *AnalyzeWords.com* forecasts the psychological weather around an author, not the user's stable climate. Though we may often speak of a "sarcastic personality" as a stable aspect of some speakers, most users of sarcasm will not fall into this category. As such, insight into the recent mind-set of an author is more valuable to a detector than knowledge of one's personality overall.

7.2 Rolling With The Punches

Our feedback-annotated dataset was collected during a fertile period for sarcasm online: the heights of the 2016 US presidential campaign. The main body of the new dataset was collected and annotated (as described earlier) in the early summer of 2016. During the month of August we acquired additional annotated training data in four weekly tranches, to incrementally retrain the model to an evolving political and social context. As shown in Table 2, each weekly tranche of extra training data yielded increased dividends in terms of F-score and precision or recall when evaluating the model on the same test set (of 1,200 tweets, 550 sarcastic and 650 non-sarcastic). The new annotated data harvested in the final week of August yielded the biggest dividends, especially in Recall, perhaps in a reflection of the growing bitterness of the campaign and of frantic campaigning in (and online commentary about) the pivotal *swing states*. As the candidates were revealing more of themselves to voters, the voters were revealing more of themselves to our model. Specifically, in week 1, 4719 sarcastic and 5361 non-sarcastic tweets were added for training; in week 2, an additional 3179 and 6901 were added; in week 3, 3571 and 6509; and in a week 4 reversal, 6504 and 3574 tweets.

Week	Precision	Recall	F-score
Week 1	0.751	0.752	0.752
week 2	0.790	0.752	0.771
Week 3	0.798	0.775	0.786
Week 4	0.839	0.869	0.85

Table 2: Bootstrapping gains (August, 2016)

8 Conclusions & Future work

Context is vital to the understanding of the fruits of any figurative device, whether metaphor, irony or sarcasm. We have explored two sources of contextual information in this work: the linguistic context of the utterance itself – which we take to be another utterance that is the proximate cause of the text under consideration – and the psychological context of the utterance's author – which we take to be the mind-set that is apparent in the author's most recent writings on Twitter. Each source of context is ultimately grounded in a text and understood in text-analytic terms. It is perhaps not so surprising then that each kind of context yields similarly large improvements to a neural model of sarcasm detection when added in isolation, but no large improvements over either alone when both are combined in a single model.

This work makes three principle contributions to the computational analysis of sarcasm. First, as outlined above, it shows how different kinds of context – from the linguistic to the psychological – can be usefully incorporated to yield improved detection. Second, it shows how accurate annotation of training data can be automated on Twitter by going directly to the source of each training text, to obtain a definitive answer as to its figurative status. So the resulting neural model does not learn to approximate the reasoning of independent human annotators but the mind-set and intent of the authors themselves. Thirdly, and perhaps most usefully for future work by others, this feedback-based dataset will be made available for use by other researchers and in other evaluations. Importantly, this dataset is not merely a collection of yes/no annotated texts, even if the *yes*es and *no*s come from authoritative sources. For each text in the dataset, we can provide the linguistic context to which it is a response, and furthermore, we can provide a psychological snapshot of the author *at the time* the tweet was posted on Twitter. In the end we believe this is the most valuable contribution of the work, as it will allow others to incorporate an understanding of personality and mind-set into their own models of that most personal and moody of figurative devices, sarcasm.

Acknowledgements: This work was funded by Science Foundation Ireland (SFI) via the ADAPT centre for Digital Content Technology.

Dataset		Alternate Configurations of the *Sarcasm Magnet** model						
		TTEA	TTIA	TTEA + CT	TTIA + CT	TTEA + PD	TTIA + PD	TTIA + CT + PD
		CNN1 + LSTM1 + DNN	CNN1 + LSTM1 + DNN	CNN1 + CNN2 + LSTM1 + LSTM2 + CL + DNN	CNN1 + CNN2 + LSTM1 + LSTM2 + CL + DNN	CNN1 + LSTM1 + EAW + CL + DNN	CNN1 + LSTM1 + EAW + CL + DNN	CNN1 + CNN2 + LSTM1 + LSTM2 + EAW + CL + DNN
(Ptáček et al., 2014) (balanced dataset) (S:15K, NS:17K)	P	0.821	0.832	0.908	0.92	0.857	0.86	0.947
	R	0.821	0.832	0.908	0.92	0.857	0.86	0.947
	F1	0.821	0.832	0.908	0.92	0.857	0.86	**0.9472** (0.9466)
(Ptáček et al., 2014) (unbalanced dataset) (S:15K, NS:39K)	P	0.814	0.813	0.926	0.937	0.851	0.843	0.946
	R	0.832	0.833	0.93	0.93	0.833	0.838	0.933
	F1	0.823	0.823	0.928	0.933	0.842	0.84	**0.94** (0.924)
(Bamman and Smith, 2015) (S:8K, NS:7K)	P	0.896	0.90	0.886	0.919	0.825	0.835	**0.9** (0.857)
	R	0.651	0.672	0.819	0.817	0.803	0.827	0.858 (**0.872**)
	F1	0.754	0.77	0.851	0.865	0.814	0.831	**0.878** (0.864)
(Cliche, 2014) (S:50K, NS:100K)	P	0.788	0.8	0.874	0.893	0.884	0.883	0.896
	R	0.751	0.769	0.842	0.843	0.812	0.817	0.862
	F1	0.769	0.784	0.858	0.867	0.846	0.849	**0.879** (0.6)
(Rajadesingan et al., 2015) (S:6K, NS:6K)	P	0.957	0.957	0.957	0.957	0.958	0.958	0.956
	R	0.807	0.807	0.807	0.807	0.861	0.861	0.905
	F1	0.875	0.875	0.875	0.875	0.907	0.907	**0.93** (0.903)
Sarcasm Magnet* (this paper)	P	0.733	0.731	0.84	0.841	0.846	0.853	0.90
	R	0.717	0.732	0.833	0.858	0.852	0.861	0.89
	F1	0.725	0.732	0.836	0.849	0.848	0.856	0.90

Table 3: Evaluation of Sarcasm Magnet (P - precision, R - recall, F1 -f-score, TTEA - target tweet excluding addressee; TTIA - target tweet including addressee; CT - Context Tweet; PD - Psychological dimensions; S - sarcastic; NS - non-sarcastic). All results are for the Sarcasm Magnet model; when available, results obtained by other authors on their own datasets are in parentheses. **Sarcasm Magnet* is the name of the current system and its associated dataset.

References

Silvio Amir, Byron C Wallace, Hao Lyu, and Paula Carvalho Mário J Silva. 2016. Modelling context with user embeddings for sarcasm detection in social media. *arXiv preprint arXiv:1607.00976* .

David Bamman and Noah A Smith. 2015. Contextualized sarcasm detection on twitter. In *Ninth International AAAI Conference on Web and Social Media*. http://homes.cs.washington.edu/ nasmith/papers/bamman+smith.icwsm15.pdf.

Penelope Brown and Stephen C Levinson. 1978. *Universals in language usage: Politeness phenomena*. Cambridge University Press. http://www.mpi.nl/publications/escidoc-66660/.

John D Campbell and Albert N Katz. 2012. Are there necessary conditions for inducing a sense of sarcastic irony? *Discourse Processes* 49(6):459–480. http://dx.doi.org/10.1080/0163853X.2012.687863.

Carol A Capelli, Noreen Nakagawa, and Cary M Madden. 1990. How children understand sarcasm: The role of context and intonation. *Child Development* 61(6):1824–1841. https://www.jstor.org/stable/1130840.

Fabio Celli, Arindam Ghosh, Firoj Alam, and Giuseppe Riccardi. 2016. In the mood for sharing contents: Emotions, personality and interaction styles in the diffusion of news. *Information Processing & Management* 52(1):93–98.

Mathieu Cliche. 2014. The sarcasm detector. `http://www.thesarcasmdetector.com/`.

Paul T Costa and Robert R McCrae. 2008. The revised neo personality inventory (neo-pi-r). *The SAGE handbook of personality theory and assessment* 2:179–198.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcasm in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Uppsala, Sweden, pages 107–116. http://www.aclweb.org/anthology/W10-2914.

Shelly Dews and Ellen Winner. 1995. Muting the meaning a social function of irony. *Metaphor and Symbol* 10(1):3–19. http://dx.doi.org/10.1207/s15327868ms1001_2.

Shelly Dews and Ellen Winner. 1999. Obligatory processing of literal and nonliteral meanings in verbal irony. *Journal of pragmatics* 31(12):1579–1599. http://psycnet.apa.org/psycinfo/1999-01341-003.

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *Proceedings of NAACL-HLT*. pages 161–169.

Deanna W. Gibbs and Herbert H. Clark. 1992. Coordinating beliefs in conversation. *Journal of Memory and Language* 31(2):183–194. doi:10.1016/0749-596X(92)90010-U.

Raymond W. Gibbs and Herbert L. Colston. 2007. *Irony in language and thought: A cognitive science reader*. Psychology Press. https://books.google.ie/books?isbn=0805860622.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, pages 581–586. http://www.aclweb.org/anthology/P11-2102.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780. http://dl.acm.org/citation.cfm?id=1246450.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. In *Association for Computational Linguistics Volume 2: Short Papers*. pages 757–762. https://www.aclweb.org/anthology/P/P15/P15-2124.pdf.

Anupam Khattri, Aditya Joshi, Pushpak Bhattacharyya, and Mark Carman. 2015. Your sentiment precedes you: Using an author's historical tweets to predict sarcasm. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Association for Computational Linguistics, Lisboa, Portugal, pages 25–30. http://aclweb.org/anthology/W15-2905.

Roger J Kreuz. 1996. The use of verbal irony: Cues and constraints. *Metaphor: Implications and applications* pages 23–38.

Roger J Kreuz and Sam Glucksberg. 1989. How to be sarcastic: The echoic reminder theory of verbal irony. *Journal of Experimental Psychology: General* 118(4):374. http://dx.doi.org/10.1037/0096-3445.118.4.374.

Roger J Kreuz and Kristen E Link. 2002. Asymmetries in the use of verbal irony. *Journal of Language and Social Psychology* 21(2):127–143. http://dx.doi.org/10.1177/02627X02021002002.

Christine Liebrecht, Florian Kunneman, and Antal Van den Bosch. 2013. The perfect solution for detecting sarcasm in tweets #not. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Association for Computational Linguistics, Atlanta, Georgia, pages 29–37. http://www.aclweb.org/anthology/W13-1605.

Stephanie Lukin and Marilyn Walker. 2013. Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue. In *Proceedings of the Workshop on Language Analysis in Social Media*. Association for Computational Linguistics, Atlanta, Georgia, pages 30–40. http://www.aclweb.org/anthology/W13-1104.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119. https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

James M. Olver and Todd A. Mooradian. 2003. Personality traits and personal values: a conceptual and empirical integration. *Personality and Individual Differences* 35(1):109 – 125. https://doi.org/http://dx.doi.org/10.1016/S0191-8869(02)00145-9.

Tomáš Ptáček, Ivan Habernal, and Jun Hong. 2014. Sarcasm detection on czech and english twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, Dublin, Ireland, pages 213–223. http://www.aclweb.org/anthology/C14-1022.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, pages 97–106. http://dl.acm.org/citation.cfm?id=2685316.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Empirical Methods on Natural Language Processing*. volume 13, pages 704–714. http://www.anthology.aclweb.org/D/D13/D13-1066.pdf.

SG Shamay-Tsoory, Rachel Tomer, and Judith Aharon-Peretz. 2005. The neuroanatomical basis of understanding sarcasm and its relationship to social cognition. *Neuropsychology* 19(3):288. http://www.apa.org/pubs/journals/releases/neu-193288.pdf.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology* 29(1):24–54. http://dx.doi.org/10.1177/0261927X09351676.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *Proceedings of the Fourth International Conference on Weblogs and Social Media*.

A. Utsumi. 2000a. Verbal irony as implicit display of ironic environment: Distinguishing ironic utterances from nonirony. *Journal of Pragmatics* http://www.utm.se.uec.ac.jp/ utsumi/paper/jop2000-utsumi.pdf.

Akira Utsumi. 2000b. Verbal irony as implicit display of ironic environment: Distinguishing ironic utterances from nonirony. *Journal of Pragmatics* 32(12):1777–1806.

Jennifer Woodland and Daniel Voyer. 2011. Context and intonation in the perception of sarcasm. *Metaphor and Symbol* 26(3):227–239. http://dx.doi.org/10.1080/10926488.2011.583197.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Tweet sarcasm detection using deep neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, pages 2449–2460. http://aclweb.org/anthology/C16-1231.

Identifying Humor in Reviews using Background Text Sources

Alex Morales and **ChengXiang Zhai**
Department of Computer Science
University of Illinois, Urbana-Champaign
amorale4@illinois.edu
czhai@illinois.edu

Abstract

We study the problem of automatically identifying humorous text from a new kind of text data, i.e., online reviews. We propose a generative language model, based on the theory of incongruity, to model humorous text, which allows us to leverage background text sources, such as Wikipedia entry descriptions, and enables construction of multiple features for identifying humorous reviews. Evaluation of these features using supervised learning for classifying reviews into humorous and non-humorous reviews shows that the features constructed based on the proposed generative model are much more effective than the major features proposed in the existing literature, allowing us to achieve almost 86% accuracy. These humorous review predictions can also supply good indicators for identifying helpful reviews.

1 Introduction

The growth of online feedback systems, such as online reviews in which users can write about their preferences and opinions, has allowed for creativity in the written communication of user ideas. As such, these feedback systems have become ubiquitous, and it's not difficult to imagine a future with smart systems reacting to user's behaviour in a human-like manner (Nijholt, 2014). An essential component for personal communication is the expression of humor. Although many people have studied the theory of humor, it still remains loosely defined (Ritchie, 2009), this leads to difficulties in modelling humor. While the task for identifying humor in text has been previously studied, most approaches have focused on shorter text such as Twitter data (Mihalcea and Strapparava, 2006; Reyes et al., 2012, 2010) (see Section 6 for a more complete review of related work). In this paper, we study the problem of automatically identifying humorous text from a new kind of text data, i.e., online reviews. In order to quantitatively test whether the review is humorous, we devise a novel approach, using the theory of incongruity, to model the reviewer's humorous intent when writing the review. The theory of incongruity states that we laugh because there is something incongruous (Attardo, 1994), in other words, there is a change from our expectation.

Specifically, we propose a general generative language model to model the generation of humorous text. The proposed model is a mixture model with multinomial distributions as component models (i.e., models of topics), similar to Probabilistic Latent Semantic Analysis (Hofmann, 1999). However, the main difference is that the component word distributions (i.e., component language models) are all assumed to be known in our model, and they are designed to model the two types of language used in a humorous text, including 1) the *general background model* estimated using all the reviews, and 2) the reference language models of all the *topical aspects* covered in the review that capture the typical words used when each of the covered aspects is discussed. Thus the model only has the parameters indicating the relative coverage of these component language models. The idea here is to use these parameters to assess how well a review can be explained by collectively by the reference language models corresponding to all the topical aspects covered in the review, which are estimated using an external text source (e.g., Wikipedia).

We construct multiple features based on the generative model and evaluate them using supervised learning for classifying reviews into humorous and non-humorous reviews. Experiment re-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 492–501
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

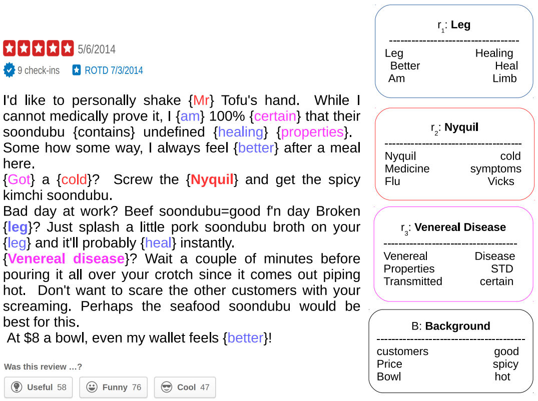
sults on a Yelp[1] review data set show that the features constructed based on the proposed generative model are much more effective than the major features proposed in the existing literature, allowing us to achieve almost 86% accuracy. We also experimented with using the results of humorous review prediction to further predict helpful reviews, and the results show that humorous review prediction can supply good indicators for identifying helpful reviews for consumers.

2 Referential Humor and Incongruity

In this section we describe some observations in our data that have motivated our approach to solving the problem. In particular, we show that humorous reviews tend to reference aspects which deviate from what is expected. That is, in funny reviews, the authors tend to use referential humor, in which specific concepts or entities are referenced to produce comedic effects, which we call *aspects*. Here we define *referential humor* to be a humorous piece of text which references aspects outside of the typical context, in our case restaurant reviews. For the rest of the paper we use humorous and funny interchangeably.

Our study uses review data from Yelp. Yelp has become a popular resource for identifying high quality restaurants. A Yelp user is able to submit reviews rating the overall experience of the restaurants. The reviews submitted to Yelp tend to have similar context, in particular they mention several aspects rating the quality of the restaurant such as food, price, service and so on. This information is expected from the reviewer in their review, however it is not always the case since there is no requirement for writing the review. Yelp users are able to vote for a review in several criterion, such as funny, cool, and useful. This gives the users an incentive for not only creating informative reviews but possibly entertaining reviews.

In Figure 1, we show a humorous review, randomly sampled by using our classifier with a high probability of being funny, where the reviewer asserts that the food has extreme medicinal properties. The reviewer refers to "Nyquil" a common cold medicine to express the food's incredible ability to cure ailments. This appears almost surprising since it would not normally be mentioned in restaurants reviews. To identify the intended humor, we can use the references the reviewer makes, e.g. Nyquil, as clues to what she is emphasising, e.g. the savory soondubu, by making such comparisons, e.g. the heavenly taste and amazing price. Yelp users seem to consider funny reviews which tended to deviate from what was expected into things which would seem out of place.



I'd like to personally shake {Mr} Tofu's hand. While I cannot medically prove it, I {am} 100% {certain} that their soondubu {contains} undefined {healing} {properties}. Some how some way, I always feel {better} after a meal here.
{Got} a {cold}? Screw the {**Nyquil**} and get the spicy kimchi soondubu.
Bad day at work? Beef soondubu=good f'n day Broken {**leg**}? Just splash a little pork soondubu broth on your {leg} and it'll probably {heal} instantly.
{**Venereal disease**}? Wait a couple of minutes before pouring it all over your crotch since it comes out piping hot. Don't want to scare the other customers with your screaming. Perhaps the seafood soondubu would be best for this.
At $8 a bowl, even my wallet feels {better}!

Figure 1: A funny review (left), with $K_d = 3$, aspect topics (right) contain words in their corresponding language model, probabilities removed for clarity, the colored (bracketed) word correspond to a different aspect assignment.

3 Language Models as a Proxy for Incongruity

Motivated by the observations discussed in the previous section (i.e., reviewers tend to reference some entities which seem unexpected in the context of the topic of the review), we propose a generative language model based on the theory of incongruity to model the generation of potentially humorous reviews. Following previous work on humor, we use the definition of incongruity in humor as "what people find unexpected" (Mihalcea and Strapparava, 2006), where "unexpected" concepts are those concepts which people do not consider to be the norm in some domain, later we formalize unexpectedness using our model.

We now describe the proposed model in more detail. Suppose we observe the following references to K_d topical aspects $A_d = \{r_1, r_2, ..., r_{K_d}\}$ in a review $R_d = [w_1, w_2, ..., w_{N_d}]$, where each r_i corresponds to an aspect reference (i.e. NyQuil in our running example), and $w_i \in V$, where V is the vocabulary set. The model generates a word, for some review, at a time, which talks about a specific aspect or is related to the language used in Yelp more broadly; we call the latter the background language model. Thus a word is generated from a mixture model, and its probability is an interpola-

[1] www.yelp.com

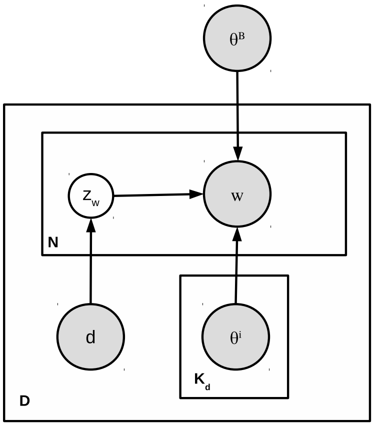


Figure 2: Generation model for reviews, where the dth review has K_d aspects in the review. The shaded nodes here are the observed data and the light node z are the latent variables corresponding to aspect assignments.

tion of the background language and the language of the references as shown in Figure 2.

These aspects provide some context to the underling meaning of a review; the reviewers use these aspects for creative writing when describing their dining experience. These aspects allow us to use external information as the context, thus we develop measures for incongruity addressing the juxtaposition of the aspect's context and the review. The review construction process is represented in a generative model, see Figure 2, where the shaded nodes represent our observations, we have observed the words as well as the referenced aspects which the reviewer has mentioned in their review. The light nodes are the labels for the aspect which has generated the corresponding word. Since the background language model, denoted by θ^B, is review independent, we can simplify the generative model by copying the background language model for each review, thus we can focus on the parameter estimation for each review in parallel.

A key component to the success of our features is the mesh of background text from external sources, or *background text sources*, and the reviews. In our example, Figure 1, Nyquil is a critical component for understanding the humor. However it is difficult to understand some references a reviewer makes without any prior knowledge. To do so, we incorporate external background knowledge in the form of language models for the referenced aspect present in the reviews. If the reviewer has made K_d references to different aspects A_d in review R_d, then for each r_i there is a corresponding language model $\theta_w^{r_i} = P(w|\theta^{r_i})$ over the vocabulary $w \in V$. For simplicity, we describe the model for each document, and use the notation θ_w^i and θ^i for the corresponding language model of r_i.

3.1 Incorporating Background Text Sources

As described before, some features we will use to describe incongruity correspond to the weights of the mixture model used to generate the words in the review, which take into account the language of the references she will make or allude, as shown in Figure 2. The probability that an author will generate a word w, for the dth review given corresponding aspects $\Theta = \{\theta^B, \theta^1, ..., \theta^{K_d}\}$, is

$$P(w, d, \Theta) = \sum_{z=0}^{K_d} P(w, z, d, \Theta) =$$

$$\sum_{z=0}^{K_d} P(w|z, \Theta)P(z|d) = \lambda\theta_w^B + (1-\lambda)\sum_{i=1}^{K_d} \pi_i\theta_w^i$$

Note K_d indicates the different aspects the reviewer will mention in a review, R_d, and hence it can vary between reviews. $\theta_w^B = P(w|z = 0, \Theta)$ is the probability that the word will appear when writing a review (e.g. background language model) and θ_w^i can be interpreted as word distributions over aspect i. Here $\lambda = P(z = 0|d)$ is the weight for the background language model and $\pi_i = \dfrac{P(z = i|d)}{1 - P(z = 0|d)}$ denotes the relative weights of the referenced aspect's language models used in the review. We denote our parameters for review R_d as $\Lambda_{R_d} = \{\pi_1, ..., \pi_{K_d}, \lambda\}$. Note that the parameter set varies depending on how many references the review makes. In order to estimate $P(w|\theta^i)$, we first need to find the aspects that the user is mentioning in their reviews. In general aspects can be defined as any topics explicitly defined in external background text data; in our experiments we define aspects as Wikipedia entities. In subsection 5.1, we describe one way of obtaining these aspects, but first we describe the estimation methodology.

3.2 Parameter Estimation

To estimate our parameters Λ_{R_d}, we would like to maximize the likelihood of $P(R_d)$, which is the same as maximizing the log-likelihood of $P(R_d)$.

That is

$$\begin{aligned}\hat{\Lambda} &= \text{argmax}_{\Lambda} \log P(R_d|\Lambda) \\ &= \text{argmax}_{\Lambda} \sum_{w\in V} c(w, R_d) \log \left(P(w, d, \Theta)\right)\end{aligned}$$

Here $c(w, R_d)$ represents the number of occurrences of the word w in R_d. In order to maximize the log-likelihood we use the EM algorithm (Dempster et al., 1977), to compute the update rules for the parameters λ and $\pi_1, ...\pi_{K_d}$. For the E-Step, at the $n+1$th iteration we have

$$P(z_w = 0) = \frac{\theta_w^B \lambda^{(n)}}{\left(\sum_{l=1}^{K_d} \theta_w^l \pi_l^{(n)}\right)(1-\lambda^{(n)}) + \theta_w^B \lambda^{(n)}}$$

$$P(z_w = j) = \frac{\theta_w^j \pi_j^{(n)}}{\sum_{l=1}^{K_d} \theta_w^l \pi_l^{(n)}}$$

Where z_w is a hidden variable indicating whether we have selected any of the aspect language models, or the background language model, when generating the word w. The update rules for the M-Step are as follows:

$$\lambda^{(n)} = \frac{\sum_{w\in V} c(w, R_d) P(z_w = 0)}{n}, \pi_j^{(n)} = \frac{\sum_{w\in V} c(w, R_d) P(z_w = j)(1 - P(z_w = 0))}{\sum_{l=1}^{K_d} \sum_{w\in V} c(w, R_d) P(z_w = l)(1 - P(z_w = 0))}$$

We ran EM until the parameters converged or a small threshold was reached. Note there is some similarity to other topic modelling approaches like PLSA (Hofmann, 1999). PLSA is a way to soft cluster the documents into several topics, in doing so a word distribution for each topic is learned. In our work we make the assumption that the "topics" are fixed, namely they are the aspects which the reviewer mentions in their review. Note that, we can similarly derive update rules for an different topic model such as LDA (Blei et al., 2003), however prior work, (Lu et al., 2011), shows that LDA does not show superior performance over PLSA empirically for a number of tasks.

4 Features construction

Since we are interested in studying discriminative features for humorous and non-humorous reviews, we set up a classification problem to classify a review into either humorous or non-humorous. In classification problems the data plays a critical role; here the labels are obtained from the funny votes in our Yelp dataset, and we describe how we created the ground-truth in Section 5. Here in this section, we discuss the new features we can construct based on the proposed language model and estimated parameter values.

4.1 Incongruity features

A natural feature in our incongruity model is the estimated background weight, λ, since it indicates how much emphasis the reviewer puts in their review to describe the referenced aspects, we denote this feature by **A1**. Another feature is based on the relative weights for the referenced aspect's language models. There tends to be more 'surprise' in a review when the reviewer talks about multiple aspects equally, this is because the more topics the reviewer writes about the more intricate the review becomes. We use the entropy of the weights $H(R_d) = -\sum_{i=1}^{K_d} \pi_i \log \pi_i$ as another incongruity score and label this feature as **A2**.

4.2 Unexpectedness features

Humor often relies on introducing concepts which seem out of place to produce a comedic effect. Thus we want to measure this divergence from the references and the language expected in the reviews. Hence a natural measure is the KL-divergence measure the distance between the background language model and the aspect language models. We use the largest deviation, $\max_i\{D_{KL}(\theta^i||\theta^B)\}$ as feature **D2**. For this feature we tried different combinations such as a weighted average, but both features seemed to perform equally so we only describe one of them.

By considering the context of the references in the reviews we can distinguish which statements should be considered as humorous, thus we also use the relative weight for each aspect to measure unexpectedness. Formally we have $U_j = \pi_j D_{KL}(\theta^j||\theta^B)$, lastly we will denote $\max_i\{U_i\}$ these set of features as **U2**.

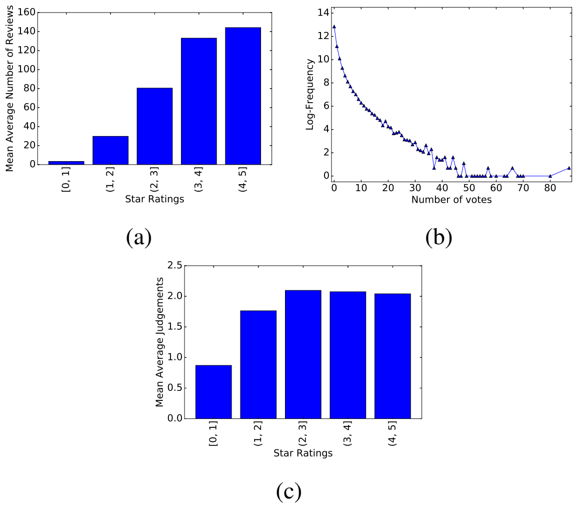
4.3 Baseline features from previous work

For completeness, we also include a description of all the baseline features used in our experiments; they represent the state of the art in defining features for this task. These features described below do not use any external text sources (leveraging external text sources is a novel aspect of our work), and they are more contextual and syntactical based features. We describe some of the most promising features, which have previously shown to be useful in identifying humor in text.

Context features: Due to the popular success of context features by Mihalcea and Pulman (2007) we tried the following features content related features: **C1**: the uni-grams in the review.[2] **C2**: length of the review. **C3**: average word length. **C4**: the ratio of uppercase and lowercase characters to other characters in the review text.

Alliteration: Inspired by the success that Mihalcea and Strapparava (2006) had using the presence and absence of alliteration in jokes, we developed a similar feature for identifying funny reviews. We used CMU's pronunciation dictionary [3] to extract the pronunciation to identify alliteration chains, and rhyme chains in sentences. A chain is a consecutive set of words which have similar pronunciation, for example if the words words "scenery" and "greenery" are consecutive they would form a rhyme chain. Similarly, "vini, vidi, visa" also forms another chain this time an alliteration chain. We used the review's total number of alliteration chains and rhyme chains and denote it by **E1**. Note that there could be different lengths of chains, we experimented with some variations but they performed roughly the same, for simplicity we did not describe them here.

Ambiguity: Ambiguity in word interpretation has also been found to be useful in finding jokes. The reasoning is that if a word has multiple interpretation it is possible that the author intended another interpretation of the word instead of the more common one. We restricted the words in the reviews to only nouns and used Wordnet [4] to extract the synsets for these words. Then we counted the average number of synsets for each of these words, finally we took the mean score for all the words in the reviews. We call these features lexical ambiguity and denote it by **E2**.



(a) (b)

(c)

Figure 3: (a) Mean average number of reviews for restaurants falling in five different star rating ranges. (b) Log occurrences of funny votes per review. (c) Mean average voting judgements for restaurants in different star ratings.

5 Experimental Results

For our experiments we obtained the reviews from the Yelp Dataset Challenge[5], this dataset contains over 1.6 million reviews from 10 different cities. We also crawled reviews from Yelp in the Los Angeles area which is not included in the Yelp Dataset Challenge. This dataset was particularly interesting since the readers are able to vote whether a review is considered *cool*, *funny*, and/or *helpful*. It also allows the flexibility for the reviewers to write longer pieces of text to express their overall rating of a restaurant.

5.1 Identifying Aspects in Reviews

We use recent advancements in Wikification, which aims to connect important entities and concepts in text to Wikipedia, it is also known as disambiguation to Wikipedia. In particular we use the work of Ratinov et al. (2011), in order to obtain the Wikipedia pages of the entities in the reviews, we call these *aspects* of the review. Using the Wikipedia description of the aspects we can compute the language models for each aspect. Using *mitlm*, the MIT language modeling toolkit by Hsu and Glass (2008), we apply Modified Kneser-Ney smoothing to obtain the language models from the Wikipedia pages obtained from review's aspects.

5.2 Preliminaries and Groundtruth Construction

In Figure 3 we give an account of data statistics based on a random sample of 500,000 reviews, focusing on the funny voting judgements and the star rating distributions. In Figure 3a, we notice that on average the highly rated restaurants tend to have more reviews. Since users would

[2] We also considered content-based features derived from PLSA topic weights, however the unigram features outperform these features, thus we exclude them for lack of space.

[3] `www.speech.cs.cmu.edu/cgi-bin/cmudict`

[4] `http://wordnet.princeton.edu/`

[5] `http://www.yelp.com/dataset_challenge`

Features		**Classifiers**		
		Naive Bayes	Perceptron	AdaBoost
Content Related Features	**C1**	69.92 (0.545)	57.62 (1.084)	69.44 (0.485)
	C2	51.33 (1.250)	50.35 (0.763)	50.56 (1.155)
	C3	50.86 (0.812)	50.00 (0.012)	50.59 (1.122)
	C4	53.85 (0.486)	50.03 (0.172)	51.41 (1.205)
Alliteration	**E1**	50.81 (0.408)	50.11 (0.301)	50.28 (1.195)
Ambiguity	**E2**	51.53 (0.677)	50.39 (0.857)	51.78 (1.533)
Incongruity	**A1**	81.32 (0.974)	81.32 (0.974)	81.32 (0.974)
	A2	83.68 (0.623)	83.68 (0.623)	83.68 (0.623)
Divergence Features	**D2**	84.55 (0.550)	83.68 (0.627)	84.23 (0.561)
Unexpectedness	**U2**	83.68 (0.627)	83.68 (0.627)	83.68 (0.627)
Combination features	**A1 + D2**	84.55 (0.549)	83.68 (0.627)	84.35 (0.548)
	A2 + D2	84.55 (0.549)	84.00 (0.579)	84.41 (0.496)
	D2 + U2	84.55 (0.549)	84.00 (0.579)	84.40 (0.549)
	A2 + D2 + U2	84.55 (0.550)	83.89 (0.593)	84.35 (0.590)
	D2 + U2 + C1	78.28 (0.545)	79.63 (0.534)	83.18 (1.109)
	A2 + D2 + C1	78.87 (0.546)	82.68 (0.353)	85.61 (0.900)
	A1 + D2+U2+C1	78.62 (0.671)	79.63 (0.528)	85.77 (0.843)
	A2 + D2+U2+C1	78.87 (0.546)	81.60 (0.703)	85.60 (0.968)

Table 1: Classification accuracies, using 5-fold cross validation, the 95% confidence is given inside the parenthesis.

prefer to dine in a restaurant expecting to get a better overall experience, they create a feedback on the reviews for those highly rated restaurants. This "rich-get-richer" effect has been also been recently observed in other social networks (Su et al., 2016) and a more detailed analysis is out of scope of this paper. We observe that most of the reviews receive a low number of funny votes in Figure 3b, with $\mu = 0.55$, where μ is the average funny rating. Computing the restaurant's average funny votes, then taking the mean by the star ratings for each category range, see Figure 3c, which seems to be consistently increasing across the different star ratings. Note that this also includes the restaurants with zero funny votes, by excluding these we found that the ratings were more consistently stable on about 2.1 votes. Thus regardless of restaurant rating, the funny reviews distribution are stable on average. Considering the prevalence of noise in the voting process, we also analysed those reviews with more than one funny vote ($\mu = 3.90$), and with more than two votes ($\mu = 5.54$).

To construct our ground-truth data, we took all of the reviews at least five funny votes, which indicates the review was collectively funny, and considered those as humorous reviews, we considered all the reviews with zero funny votes as non-humorous reviews. We obtained 17,769 humorous reviews and 856,202 non-humorous, from which we sampled 12,000 reviews from each category, and another 5,000 reviews was left for a development dataset, to obtain a corpus with 34,000 reviews total. In total we collected 2,747 wikipedia pages with an average of about 247 sentences per page. In our work we focused on identifying distinguishing features and relative improvement in a balanced dataset and while the true distribution may be skewed, we leave the unbalance distribution study for future work.

Finally we use five-fold cross validation to evaluate all the methods. Due to the success of linear classifiers in text classification tasks we were interested in studying the Perceptron and Adaboost algorithms, we also used a Naive Bayes classifier which has been shown to perform relatively well in humor recognition tasks (Mihalcea and Strapparava, 2006). We used the Learning Based Java (LBJava) toolkit by Rizzolo and Roth (2010) for the implementation of all the classifiers and used their recommended parameter settings. For the Averaged Perceptron implementation, we used a learning rate of 0.05 and thickness of 5. In Adaboost, we choose BinaryMIRA as our weak learner to do our boosting on. We also considered SparseWinnow and SparseConfidenceWeighted to be our weak learner as well, but the boosting performance for those two learners is marginal on the development set.[6] All experiments were run on an Intel Core i5-4200U CPU with 1.60GHz running Ubuntu.

5.3 Predicting Funny Reviews

We report the results of the features in Table 1. First we can compare the accuracies of the individual features. For the content related features we see that the best features is **C1**, which is consistent to what others have found in humor recognition research (Mihalcea and Pulman, 2007). The other content related features are based on some popular features for detecting useful reviews, however we notice that in the humor context it is not very effective. The performance of the contextual features could indicate that humor is not specific to a particular context and thus comparing different context between humorous and non-humorous text will not always work.

For the alliteration and ambiguity features which were reported to be very useful in short text, such as one-liners and on Twitter, are not as useful in detecting humours reviews. The reason is pretty clear since when writing a funny review, the reviewer does not worry about the limitation of text and thus their humor does not rush to a punch-line. Instead the reviewer is able to write a longer more creative piece, adhering to less structure. The features based on incongruity and unexpectedness, do really well in distinguishing the funny and non-funny reviews. For incongruity the best feature is **A2**, achieving about the same accuracy as unexpectedness features of about 83% accuracy.

The best feature was **D2** achieving an accuracy of around 84% accuracy. The features seem to be consistent over all of our classifiers. This indicates that incorporating background text sources to identify humor in reviews is crucial, and our features we can indirectly capture some common knowledge, e.g. prior knowledge. In particular it provides evidence that humor in online reviews can be better categorized as referential humor (Ritchie, 2009) rather then shorter jokes. The results also suggest that we can use these features to help predict the style of humorous text.

Exploring this would be an interesting venue for future work. When we combine our features for the classification task and find that the best combination is the incongruity features with the divergence features. We do not report the results for features **E1, E2** and other context features, **C2, C3, C4**, since their performance when combined with other features did not add to the accuracy of the more discriminant feature. The divergence feature **D2** plays a big role in the accuracy performance. This is in line with our hypothesis that the more uncommon language used the more it is possible to be for a humorous purpose.

It is interesting to see that AdaBoost performed the best out of all three classifiers achieving about 86% accuracy, especially when more features were added, the classifier was able to use this information for improvement. While Naive Bayes and the Perceptron algorithm did not make such improvement achieving about 85% accuracy.

5.4 Ranking Funny Reviews

From the data we noticed that funny reviews tend to be voted highly useful,in particular we noticed a correlation coefficient of 0.77. Although it would have been easy to use the useful votes as a feature to determine whether the review is funny/not funny, these scores are only available after people have been exposed to these reviews. To test how well the features worked when identifying helpful reviews, in a more realistic setting, we formulated a retrieval problem. Given a set of reviews, $\mathcal{D} = \{R_1, R_2, ..., R_m\}$ and relevant scores based on usefulness, $U = \{u_1, u_2, ..., u_m\}$, is it possible to develop a scoring function such that we rank the useful reviews higher? For this task we used the classification output of Naive Bayes, $P(\text{funny}|R_i)$ where i is the current example under consideration, for our scoring function and trained with the best performing features in the original dataset. We used a with-held dataset crawled from restaurants in Yelp in the Los Angeles area containing about 1,360 reviews with 260 reviews labelled as helpful and the other reviews labelled as not helpful. To obtain the ground truth we used the useful votes in Yelp similar to how we constructed the funny labels, using a threshold of 5 votes minimum to be considered helpful. This experiment reveals two things about our features for detecting humorous reviews. First we see that the preci-

[6] Since our main goal is to understand the effectiveness of various features we did not further tune these parameters since they are presumably orthogonal to the question we study.

K	Precision @ K
1	1.00
10	0.50
25	0.48
50	0.44
100	0.45
200	0.54

Table 2: Precision of useful reviews.

sion is around 50%, see Table 2, this is more than two times better than random guess which is about 19% and second that our features can be used to filter out some useful reviews.

6 Related Work

Although there has been much work in the theory of humor by many linguists, philosophers and mathematicians (Paulos, 2008), the definition of humor is still a debated topic of research (Attardo, 1994). There have been many applications from computational humor research; for instance, creating embodied agents using humor, such as chat bots, which could allow for more engaging interactions and can impact many domains in education (Binsted et al., 2006). Existing work on computational humor research can typically be divided into humor recognition and humor generation.

In humor generation, some systems have successfully generated jokes and puns by exploiting some lexical structure in the pun/joke (Lessard and Levison, 1992; Manurung et al., 2008; McKay, 2002). The HAHAcronym project was able to take user inputs and output humorous acronyms and it achieves comical effects by exploiting incongruity (Stock and Strapparava, 2002). Work in automatic generation of humor is limited to particular domains, usually only generating short funny texts.

One of the earliest work on humor recognition in text data is the work of Mihalcea and Strapparave (2006), trying to identify *one-liners*, short sentences with a humorous effect. They frame the problems as a classification problem and develop surface features (alliteration, antonym, and adult slang) as well as context related features. They ultimately proposed that additional knowledge such as, irony, ambiguity, incongruity, and common sense knowledge among other things would be beneficial in humor recognition, but they do not further pursue these avenues. Although they are able to distinguish between humorous and non-humorous one liners, in longer of texts such as reviews it is not so clear that these features suffice. Instead we make use of the creative writing structure of the reviewers by looking at the referenced entities in their reviews.

Although verbal irony can be humorous, and an active topic of research (Wallace, 2013), it is often defined as the "opposite to what the speaker means", and combining features for identifying both humor and irony has been studied (see, e.g., Reyes et al. (2012)). In the work by Reyes et al. (2012), the authors defined the unexpectedness feature as semantic relatedness of concepts in Wordnet and assuming that the less the semantic relatedness of concepts the funnier the text. In our work we use a similar definition but applying it to the "topical" relatedness of the referenced aspects and the background language model. The authors demonstrate that irony and humor share some similar characteristics and thus we can potentially use similar features to discriminate them. There has been some early work in identifying humor features in web comments (Reyes et al., 2010), in these comments the users are able to create humor through dialogue thus making the problem more complex. More recently there was a workshop in SemEval-2017 [7], which focus is on identifying humorous tweets which are related, typically as a punchline, to a particular hashtag.

Kiddon and Brun (2011) aimed to understand "That's what she said" (TWSS) jokes, which they classify as double entendres. They frame the problem as metaphor identification and notice that the source nouns are euphemisms for sexually explicit nouns. They also make use of the common structure of the TWSS jokes to the erotic domains to improve 12% in precision over word-based features. In our work we try to explicitly model the incongruity of the reviewer, by doing so we are able to distinguish the separate language used by the user when introducing humorous concepts. Recently there has been work in consumer research, to identify the prevalence of humor in social media (McGraw et al., 2015). The main focus was to examine the benign violation theory, which "suggest that things are humorous when people perceive something as wrong yet okay". One of their finding suggests that humor is more prevalent in complaints than in praise, thus motivating

[7] http://alt.qcri.org/semeval2017/task6/

the usage of automatic humor identification methods for restaurants regardless of its popularity.

While there is a breadth of work in identifying helpful reviews and opinion spam in reviews (Jindal and Liu, 2008) as well as deceptive opinion spam (Ott et al., 2011), and synthetic opinion spam (Sun et al., 2013); we show that humour can also be used to identify helpful reviews.

7 Conclusion

We have studied humorous text identification in a novel setting involving online reviews. This task has not been studied in the previous work and is different than detecting humorous jokes or one-liners, this allows for creative and expressive writing since the reviewer is not limited in text. In this problem we cannot directly apply the ideas that others have developed in order to identify the humorous reviews. Instead features that are based on the theory of incongruity are shown to outperform previous features and are effective in the classification task. Our model introduces a novel and way to incorporate external text sources for humor identification task, and which can be applied to any natural language provided there is a reference database, i.e. news articles or Wikipedia pages, in that language. We also show that the features developed can also be used to identify helpful reviews. This is very useful in the online review setting since there tends to be a cumulative advantage, that is the "rich get richer" effect which limits the exposure that the users get to other helpful reviews. Thus identifying these types of review early can potentially diversify the types of reviews that the users read.

Although we used a background language model on the entire corpus to capture a sense of expectation, there could be other ways to do this. For example, we could develop neural network embeddings to capture the entities descriptions in the reviews. Another direction would be to use topic models and see whether reviewers are more inclined to compare different types of references when talking about certain aspects of restaurants or other products. A different approach to identifying helpful reviews would be to create entertaining and informative summaries.

Acknowledgments

The first author was supported by the University of Illinois, Urbana-Champaign College of Engineering's Support for Underrepresented Groups in Engineering (SURGE) Fellowship and the Graduate College's Graduate Distinguished Fellowship.

References

Salvatore Attardo. 1994. *Linguistic theories of humor*, volume 1. Walter de Gruyter.

Kim Binsted, Anton Nijholt, Oliviero Stock, Carlo Strapparava, G Ritchie, R Manurung, H Pain, Annalu Waller, and D O'Mara. 2006. Computational humor. *Intelligent Systems, IEEE*, 21(2):59–69.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc.

Bo-June Hsu and James Glass. 2008. Iterative language model estimation: efficient data structure & algorithms. In *Proceedings of Interspeech*, volume 8, pages 1–4.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230. ACM.

Chloe Kiddon and Yuriy Brun. 2011. That's what she said: double entendre identification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 89–94. Association for Computational Linguistics.

Greg Lessard and Michael Levison. 1992. Computational modelling of linguistic humour: Tom swifties. In *ALLC/ACH Joint Annual Conference, Oxford*, pages 175–178.

Yue Lu, Qiaozhu Mei, and ChengXiang Zhai. 2011. Investigating task performance of probabilistic topic models: an empirical study of plsa and lda. *Information Retrieval*, 14(2):178–203.

Ruli Manurung, Graeme Ritchie, Helen Pain, Annalu Waller, Dave O'Mara, and Rolf Black. 2008. The construction of a pun generator for language skills development. *Applied Artificial Intelligence*, 22(9):841–869.

A Peter McGraw, Caleb Warren, and Christina Kan. 2015. Humorous complaining. *Journal of Consumer Research*, 41(5):1153–1171.

Justin McKay. 2002. Generation of idiom-based witticisms to aid second language learning. *Stock et al.(2002)*, pages 77–87.

Rada Mihalcea and Stephen Pulman. 2007. Characterizing humour: An exploration of features in humorous texts. In *Computational Linguistics and Intelligent Text Processing*, pages 337–347. Springer.

Rada Mihalcea and Carlo Strapparava. 2006. Learning to laugh (automatically): Computational models for humor recognition. *Computational Intelligence*, 22(2):126–142.

Anton Nijholt. 2014. Towards humor modelling and facilitation in smart environments. *Advances in Affective and Pleasurable Design*, pages 260–269.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics.

John Allen Paulos. 2008. *Mathematics and humor: A study of the logic of humor*. University of Chicago Press.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics.

Antonio Reyes, Martin Potthast, Paolo Rosso, and Benno Stein. 2010. Evaluating humour features on web comments. In *LREC*.

Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering*, 74:1–12.

Graeme Ritchie. 2009. Can computers create humor? *AI Magazine*, 30(3):71.

Nick Rizzolo and Dan Roth. 2010. Learning based java for rapid development of nlp systems. In *LREC*.

Oliviero Stock and Carlo Strapparava. 2002. Hahacronym: Humorous agents for humorous acronyms. *Stock, Oliviero, Carlo Strapparava, and Anton Nijholt. Eds*, pages 125–135.

Jessica Su, Aneesh Sharma, and Sharad Goel. 2016. The effect of recommendations on network structure. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1157–1167. International World Wide Web Conferences Steering Committee.

Huan Sun, Alex Morales, and Xifeng Yan. 2013. Synthetic review spamming and defense. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1088–1096. ACM.

Byron C Wallace. 2013. Computational irony: A survey and new perspectives. *Artificial Intelligence Review*, pages 1–17.

Sentiment Lexicon Construction with Representation Learning Based on Hierarchical Sentiment Supervision

Leyi Wang and **Rui Xia**[*]
School of Computer Science & Engineering
Nanjing University of Science & Technology, China
leyiwang.cn@gmail.com, rxia@njust.edu.cn

Abstract

Sentiment lexicon is an important tool for identifying the sentiment polarity of words and texts. How to automatically construct sentiment lexicons has become a research topic in the field of sentiment analysis and opinion mining. Recently there were some attempts to employ representation learning algorithms to construct a sentiment lexicon with sentiment-aware word embedding. However, these methods were normally trained under document-level sentiment supervision. In this paper, we develop a neural architecture to train a sentiment-aware word embedding by integrating the sentiment supervision at both document and word levels, to enhance the quality of word embedding as well as the sentiment lexicon. Experiments on the SemEval 2013-2016 datasets indicate that the sentiment lexicon generated by our approach achieves the state-of-the-art performance in both supervised and unsupervised sentiment classification, in comparison with several strong sentiment lexicon construction methods.

1 Introduction

Sentiment lexicon is a set of words (or phrases) each of which is assigned with a sentiment polarity score. Sentiment lexicon plays an important role in many practical sentiment analysis and opinion mining tasks. There were some manually annotated universal sentiment lexicons such as General Inquireer (GI) and HowNet. However, due to the ubiquitous domain diversity and absence of domain prior knowledge, the automatic construction technique for domain-specific sentiment lexicons has become a challenging research topic in the field of sentiment analysis and opinion mining (Wang and Xia, 2016).

The early work employed unsupervised learning for sentiment lexicon construction. They normally labelled a set of seed words at first, and then learned the polarity of each candidate word, based on either word conjunction relations (e.g., constellation and transition in texts) (Hatzivassiloglou and McKeown, 1997), or the word co-occurrence information (such as pointwise mutual information, PMI) (Turney, 2002), between the candidate word and the seed words. However, the unsupervised manner showed limited effect in sentiment prediction, and the performance greatly depends on the quality of the seed words.

To fully exploit the sentiment labeling information in texts, a series of supervised learning methods was further proposed to learn the sentiment lexicons. For example, Mohammad et al. (2013) proposed to construct sentiment lexicons by calculating PMI between the word and the distantly supervised sentiment labels (such as emoticons) in tweets and the word's sentiment orientation (SO). The resulting lexicons obtained the best results in SemEval 2013. More advanced representation learning models were also utilized, with the aim to construct the sentiment lexicons with efficient word embeddings (Tang et al., 2014a; Hamilton et al., 2016; Vo and Zhang, 2016). The traditional representation learning framework such as Word2Vec only captures the syntactic information in the texts, but ignores the sentiment relations between words. Therefore, some researchers attempted to add sentiment supervision into the network structure, in order to train a sentiment-aware word embedding. For example, Tang et al. (2014a) exploited a dedicated neural architecture to integrate document-level sentiment supervision and the syntactic knowledge for representation

[*] The corresponding author of this paper.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 502–510
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

learning. The sentiment-aware word embedding is then used to construct a sentiment lexicon. Vo and Zhang (2016) proposed to learn a two-dimensional sentiment representation based on a simple neural network. The sentiment lexicons generated by their approach obtained better performance to predict the tweet sentiment labels, in comparison with the PMI-based method (Mohammad et al., 2013).

Although these supervised learning methods can to some extent exploit the sentiment labeling information in the texts and can learn a sentiment-aware word embedding, the manner of using document-level sentiment supervision suffers from some complex linguistic phenomena such as negation, transition and comparative degree, and hence unable to capture the fine-grained sentiment information in the text. For example, in the following tweet

> *"Four more fake people added me. Is this why people don't like Twitter? :(",*

the document-level sentiment label is negative, but there is a positive word *"like"* in the text. In representation learning, the embeddings of words are summed up to represent the document, and the word *"like"* will be falsely associated with the negative sentiment label. Such linguistic phenomena occur frequently in review texts, and makes sentiment-aware word representation learning less effective.

To address this problem, in this paper, we propose a new representation learning framework called HSSWE, to learn sentiment-aware word embeddings based on hierarchical sentiment supervision. In HSSWE, the learning algorithm is supervised under both document-level sentiment labels and word-level sentiment annotations (e.g., labeling *"like"* as a positive word). By leveraging the sentiment supervision at both document and word level, our approach can avoid the sentiment learning flaws caused by coarse-grained document-level supervision by incorporating fine-grained word-level supervision, and improve the quality of sentiment-aware word embedding. Finally, following Tang et al. (2014a), a simple classifier was constructed to obtain the domain-specific sentiment lexicon by using word embeddings as inputs.

The main contributions of this work are as follows:

1. To the best of our knowledge, this is the first work that learns the sentiment-aware word representation under supervision at both document and word levels.

2. Our approach supports several kinds of word-level sentiment annotations such as 1) predefined sentiment lexicon; 2) PMI-SO lexicon with hard sentiment annotation; 3) PMI-SO lexicon with soft sentiment annotation. By using PMI-SO dictionary as word-level sentiment annotation, our approach is totally corpus-based, without any external resource.

3. Our approach obtains the state-of-the-art performance in comparison with several strong sentiment lexicon construction methods, on the benchmark SemEval 2013-2016 datasets for twitter sentiment classification.

2 Related Work

In general, sentiment lexicons construction can be classified into two categories, dictionary-based methods and corpus-based methods.

Dictionary-based methods generally integrate predefined resources, such as WordNet, to construct sentiment lexicons. Hu and Liu (2004) exploited WordNet for sentiment lexicon construction. They first labelled two sets of seed words by polarities, then extended the sets by adding the synonyms for each word to the same set and antonyms to the other. For a given new word, Kim and Hovy (2004) introduced a Naive Bayes model to predict the polarities with .the synonym set obtained from WordNet as features. Kamps et al. (2004) investigated a graph-theoretic model of WordNet's synonymy relation and measured the sentiment orientation by distance between each candidate word and the seed words with different polarities. Heerschop et al. (2011) proposed a method to propagate the sentiment of seed set words through semantic relations of WordNet.

Corpus-based approaches originate from the latent relation hypothesis: "*Pairs of words that cooccur in similar patterns tend to have similar semantic and sentiment relations*" (Turney, 2008).

The primary corpus-based method made the use of PMI. Turney (2002) built a sentiment lexicon by calculating PMI between the candidate word and seed words. The difference of the PMI score between positive and negative seed words is finally used as the sentiment orientation (SO) of each candidate word (Turney, 2002). Many variants of

PMI were proposed afterwards, for example, positive pointwise mutual information (PPMI), second order co-occurrence PMI (SOC-PMI), etc. Hamilton et al. (2016) proposed to build a sentiment lexicon by a propagation method. The key of this method is to build a lexical graph by calculating the PPMI between words. Instead of calculating the PMI between words, Mohammad et al. (2013) proposed to use emoticons as distant supervision and calculate the PMI between words and the distant class labels, and obtained sound performance for tweet sentiment classification.

The latest corpus-based approaches normally utilize the up-to-date machine learning models (e.g. neural networks) to first learn a sentiment-aware distributed representation of words, based on which the sentiment lexicon is then constructed. There were many word representation learning methods such as NNLM (Bengio et al., 2003) and Word2Vec (Mikolov et al., 2013). However, they mainly consider the syntactic relation of words in the context but ignore the sentiment information. Some work were later proposed to deal with this problem by incorporating the sentiment information during representation learning. For example, Tang et al. (2014a) adapted a variant of skip-gram model, which can learn the sentiment information based on distant supervision. Furthermore, Tang et al. (2014b) proposed a new neural network approach called SSWE to train sentiment-aware word representation. Vo and Zhang (2016) exploited a simple and fast neural network to train a 2-dimensional representation. Each dimension is explicitly associated with a sentiment polarity.

The sentiment-aware word representation in these methods was normally trained based on only document-level sentiment supervision. In contrast, the learning algorithm in our approach is supervised under both document-level and word-level sentiment supervision.

3 Our Approach

Our approach is comprised of three base modules: (1) Word-level sentiment learning and annotation; (2) Sentiment-aware word embedding learning; (3) Sentiment lexicon construction.

Our approach depends on document-level sentiment labels. The tweet corpus provides a cheap way to get document-level sentiment annotation, owing to the distant sentiment supervision. But it should be noted that our approach is feasible for any corpus provided with document-level sentiment labels (not merely tweets).

The first module of our method aims to learn the pseudo sentiment distribution for each word and use it as word-level sentiment annotations to supervise word embedding learning.

In the second module, we learn the sentiment-aware embeddings for each word in corpus, based on hierarchical sentiment supervision.

In the last module, we construct a sentiment lexicon by using the sentiment-aware word embeddings as the basis.

3.1 Learning Word-Level Sentiment Supervision

In addition to use a pre-defined sentiment lexicon for word-level annotations, we also propose to learn the word-level sentiment supervision, based on PMI and SO.

(1) PMI and SO

Given a corpus with document-level class labels. We first compute the PMI score between each word t and two class labels

$$PMI(t,+) = \log \frac{p(+|t)}{p(+)}, \tag{1}$$

$$PMI(t,-) = \log \frac{p(-|t)}{p(-)}, \tag{2}$$

where $+$ and $-$ denote the positive and negative document-level class labels, respectively.

Second, we compute the SO score for each word t:

$$SO(t) = PMI(t,+) - PMI(t,-). \tag{3}$$

We call $\{t, SO(t)\}$ as PMI-SO dictionary. The PMI-SO dictionary was widely used as a corpus-based sentiment lexicon for sentiment classification. By contrast, in our approach, it is the first step to learn the sentiment-aware word representation. Our approach supports two kinds of word-level sentiment annotations: 1) PMI-SO dictionary with hard sentiment annotation; 2) PMI-SO dictionary with soft sentiment annotation.

The word-level sentiment annotation is represented as $[\hat{p}(-|t), \hat{p}(+|t)]$. We employ the following two ways to obtain $[\hat{p}(-|t), \hat{p}(+|t)]$.

(2) PMI-SO lexicon with hard sentiment annotation

Notations	Description
e_t	The embedding of word t
de	The document representation of d
b_t	The bias of word-level softmax layer
b_d	The bias of document-level softmax layer
θ_t	Weight of word-level softmax layer
θ_d	Weight of document-level softmax layer
$p(c\|e_t)$	The sentiment distribution of word t predicted by our model
$p(c\|de)$	The sentiment distribution of document d predicted by our model
$\hat{p}(c\|t)$	The word-level sentiment annotation of word t with respect to class c
$\hat{p}(c\|d)$	The document-level sentiment annotation of document d with respect to class c

Table 1: The parameters used in our neural network.

"Hard sentiment annotation" indicates that $[\hat{p}(-|t), \hat{p}(+|t)]$ is a two-dimensional one-hot representation, where the annotation of words is given by the class labels:

$$[\hat{p}(-|t), \hat{p}(+|t)] \\ = \begin{cases} [0,1], & \text{if } SO(t) > 0 \\ [1,0], & \text{if } SO(t) < 0 \\ random\{[1,0] or [0,1]\}, & \text{otherwise} \end{cases}. \tag{4}$$

(3) PMI-SO lexicon with soft sentiment annotation

"Soft sentiment annotation" means that the annotation is given by the probability of two sentiment polarities, rather than the class label. We first use the sigmoid function to map the SO score to the range of a probability, and then define

$$[\hat{p}(-|t), \hat{p}(+|t)] = [1 - \sigma(SO(t)), \sigma(SO(t))] \tag{5}$$

as the PMI-SO soft sentiment distribution of the word t.

3.2 Learning Sentiment-aware Word Representation under Hierarchical Sentiment Supervision

Till now we have obtained both document and word-level sentiment annotations, in the next step, we propose a neural network framework to learn the sentiment-aware word representation by integrating the sentiment supervision at both word and document granularities. We call it "hierarchical sentiment supervision". The architecture of our model is shown in Figure 1. We denote the corpus as $D = \{d_1, d_2, ..., d_N\}$ where N is the size of the corpus. Suppose d_k is k-th document in D, and t_i represents the i-th word in a document d. The parameters used in our neural network are described in Table 1.

We construct a embedding matrix $C \in R^{V \times M}$, of which each row represents the embedding of a word in the vocabulary, where V is the size of the vocabulary and M is the dimension of word embedding. We randomly initialize each element of matrix C with a normal distribution.

(1) Word-Level Sentiment Supervision

We use the word-level sentiment annotation $[\hat{p}(-|t), \hat{p}(+|t)]$ provided in Section 3.1 to supervise word representation learning at the word level.

For each word in document d, we map it to a continuous representation as $e \in C$ and feed e into our model to predict the sentiment distribution of the input word:

$$p(c|e) = softmax(\theta_t \cdot e + b_t). \tag{6}$$

The cost function is defined as the average cross entropy that measures the difference between the sentiment distribution predicted in our model and the sentiment annotations at the word level:

$$f_{word} = -\frac{1}{T} \sum_{k=1}^{N} \sum_{t \in d_k} \sum_{c \in \{+,-\}} \hat{p}(c|t) \log p(c|e_t) \tag{7}$$

where T is the number of words in corpus.

(2) Document-Level Sentiment Supervision

We use the document-level sentiment annotations to supervise word representation learning at the document level.

In order to obtain a continuous representation of a document d, we simply use the average embedding of words in d as de:

$$de = \frac{1}{|d|} \sum_{t \in d} e_t. \tag{8}$$

We feed de into our model to predict the sentiment probability:

$$p(c|de) = softmax(\theta_d \cdot de + b_d). \tag{9}$$

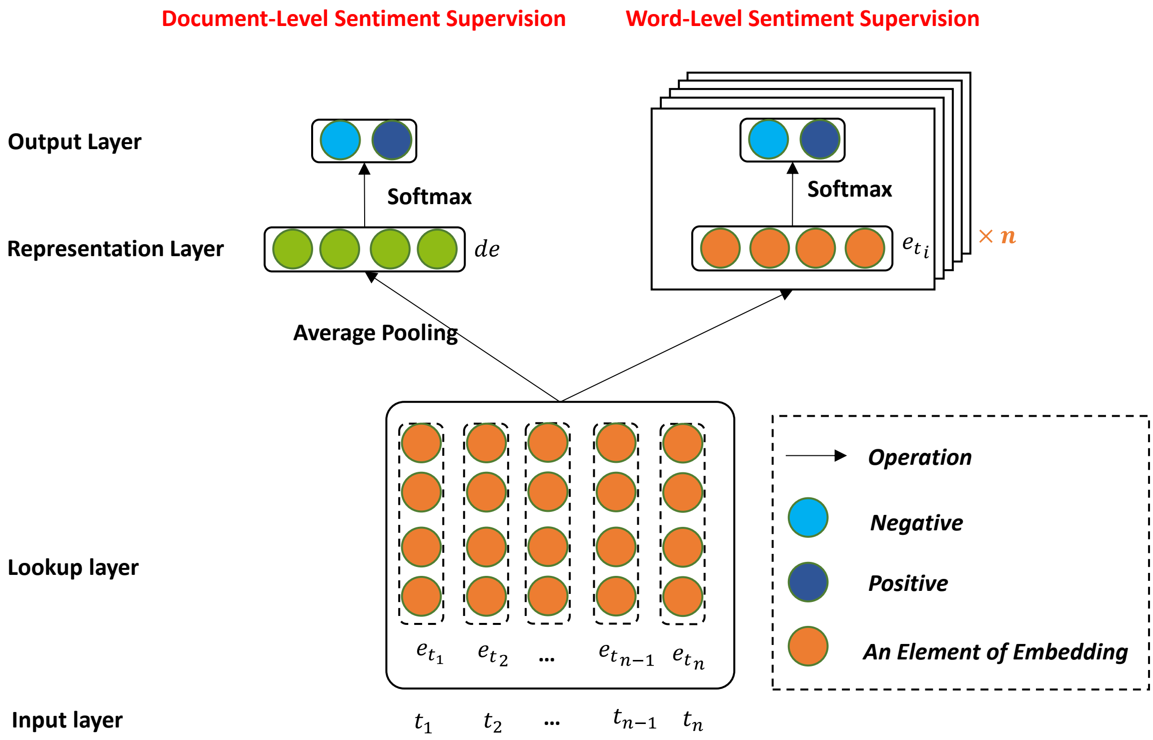

Document-Level Sentiment Supervision
Word-Level Sentiment Supervision
Output Layer
Softmax
Softmax
Representation Layer
de
e_{t_i}
× n
Average Pooling
Lookup layer
e_{t_1} e_{t_2} ... $e_{t_{n-1}}$ e_{t_n}
Input layer
t_1 t_2 ... t_{n-1} t_n
Operation
Negative
Positive
An Element of Embedding

Figure 1: The Architecture of our Neural Network. Given a document d, represented as $[t_1, t_2, \ldots, t_n]$. t_i is the i-th word in d. And e_{t_i} represents the embedding of the word t_i. We take de, the average embedding of $[e_{t_1}, e_{t_2}, \ldots, e_{t_n}]$, as the representation of document d. We get each embedding of words in d as input to predict its sentiment polarities. We also take de as input to predict the sentiment for document d one time per epoch.

Similarly, the cost function is defined as average cross entropy that measures the difference between the sentiment distribution predicted in our model and the sentiment annotation at the document level:

$$f_{doc} = -\frac{1}{N}\sum_{k=1}^{N}\sum_{c\in\{+,-\}}\hat{p}(c|d_k)\log p(c|de_k) \tag{10}$$

where $\hat{p}(c|d_k)$ is the sentiment annotation of document d_k. $\hat{p}(c|d_k) = 1$ denotes the class label of d_k is positive, otherwise $\hat{p}(c|d_k) = 0$.

(3) Word and Document-Level Joint Learning

In order to learn the sentiment-aware word representation at both word and document levels, we integrate the cost function of two levels in a weighted combination way. The final cost function is defined as follows:

$$f = \alpha f_{word} + (1-\alpha) f_{doc} \tag{11}$$

where α is a tradeoff parameter$(0 \leq \alpha \leq 1)$. The weight of f_{word} can be increased by choosing a lager value of α.

We train our neural model with stochastic gradient descent and use AdaGrad (Duchi et al., 2011) to update the parameters.

3.3 From Sentiment Representation to Sentiment Lexicon

In this part, we follow the method proposed by Tang et al. (2014a) to build a classifier to convert the sentiment-aware word representation learned in Section 3.2 to a sentiment lexicon. The word representation is the input of the classifier and word sentiment polarity is the output.

Firstly, we utilize the embedding of 125 positive and 109 negative seed words manually labelled by Tang et al. (2014a) as training data[1].

Secondly, a variant-KNN classifier is also applied to extending the seed words on a web dictionary called Urban Dictionary. Unlike (Tang et al.,

[1] http://ir.hit.edu.cn/ dytang/paper/14coling/data.zip

Dataset	$\#pos$	$\#neg$	$Total$
SemEval2013-train	3632	1449	5081
SemEval2013-dev	482	282	764
SemEval2013-test	1474	559	2033
SemEval2014-test	982	202	1184
SemEval2015-test	1038	365	1403
SemEval2016-test	7059	3231	10290

Table 2: Statistics of Evaluation Datasets

2014a), we did not extend the neutral words.

Thirdly, a traditional logistic regression classifier is trained by using the embeddings of extended sentiment words as the inputs. The sentiment score of a word is the difference between its positive and negative probabilities.

Finally, the sentiment lexicon can be collected by using the classifier to predict the other words' sentiment score.

4 Experiment Study

4.1 Datasets and Settings

We utilize the public distant-supervision corpus[2] (Go et al., 2009) to learn our lexicons. We set M, the dimension of embedding, as 50. The learning rate is 0.3 for stochastic gradient descent optimizer. We tune the hyper-parameter α in the training process.

We evaluate the sentiment lexicons in both supervised and unsupervised sentiment classification tasks, on the SemEval 2013-2016 datasets. The statistics of evaluation datasets are shown in Table 2.

Supervised Sentiment Classification Evaluation: To evaluate the effect of the sentiment lexicon in supervised sentiment classification, we report the supervised sentiment classification performance by using some pre-defined lexicon features. We follow (Mohammad et al., 2013) to extract the lexicon features as follows:

- Total count of words in the tweet score of which is greater than 0;
- Total count of words in the tweet score of which is less than 0;
- The sum of scores for all word great than 0;
- The sum of scores for all word less than 0;
- The max score greater than 0;
- The min score less than 0;
- Non-zero score of the last positive word in the tweet;
- Non-zero score of the last negative word in the tweet.

We report the performance of SVM by using these lexicon features. The LIBSVM [3] toolkit is used with a linear kernel and the penalty parameter is set as the default value. The metric is F_1 score.

Unsupervised Sentiment Classification Evaluation: For unsupervised sentiment classification, we sum up the scores of all sentiment words in the document, according to the sentiment lexicon. If the sum is greater than 0, the document will be considered as positive, otherwise negative. The unsupervised learning evaluation metric is accuracy.

4.2 (External) Comparison with Public Lexicons

We compare our HSSWE method with four sentiment lexicons generated by the related work proposed in recent years:

- **Sentiment140** was constructed by Mohammad et al. (2013) on tweet corpus based on PMI between each word and the emoticons.
- **HIT** was constructed by Tang et al. (2014a) with a representation learning approach.
- **NN** was constructed by Vo and Zhang (2016) with a neural network method.
- **ETSL** refers to SemEval-2015 English Twitter Sentiment Lexicon[4] (Rosenthal et al., 2015; Kiritchenko et al., 2014), which is done using Best-Worst Scaling.

Note that Tang et al. (2014a), Vo and Zhang (2016) used incomplete dataset of SemEval2013 in their papers. For fair comparison, we conduct

[2] http://help.sentiment140.com/for-students

[3] http://www.csie.ntu.edu.tw/ cjlin/libsvm

[4] http://www.saifmohammad.com/WebDocs/lexiconstoreleaseonsclpage/SemEval2015-English-Twitter-Lexicon.zip

Lexicon	Semeval2013	Semeval2014	Semeval2015	Semeval2016	Average
Sentiment140	0.7317	0.7271	0.6917	0.6809	0.7079
HIT	0.7181	0.6947	0.6797	0.6928	0.6963
NN	0.7225	0.7115	***0.6970***	0.6887	0.7049
ETSL	0.7104	0.7090	0.6650	0.6862	0.6926
HSSWE	***0.7550***	***0.7424***	0.6921	***0.7097***	***0.7248***

Table 3: Supervised Evaluation for External Comparison (F_1 Score)

Lexicon	Semeval2013	Semeval2014	Semeval2015	Semeval2016	Average
Sentiment140	0.7208	0.7416	0.6935	0.6928	0.7122
HIT	0.7566	0.7922	0.7128	0.7282	0.7474
NN	0.6903	0.7280	0.6507	0.6585	0.6819
ETSL	0.7675	0.8226	0.7505	***0.7365***	0.7693
HSSWE	***0.7734***	***0.8539***	***0.7669***	0.7206	***0.7787***

Table 4: Unsupervised Evaluation for External Comparison (Accuracy)

all the comparison experiments on the complete benchmark datasets.

Supervised Sentiment Classification: We first report the supervised sentiment classification F_1 score of five compared methods on the Semeval 2013-2016 datasets in Table 3. It can be seen that our HSSWE method gets the best result on all four datasets. It outperforms Sentiment140, HIT, NN and ETSL 1.7, 2.8, 1.9, and 3.2 percentages on the average of four datasets. The improvements are significant according to the paired t-test.

Unsupervised Sentiment Classification: We then report the unsupervised sentiment classification accuracy of five methods on the Semeval 2013-2016 datasets in Table 4. In can be seen that HSSWE obtains the best performance on Semeval 2013-2015. On the Semeval 2016 dataset, it is slightly lower than ETSL. Across four datasets, the average accuracy of HSSWE is 6.6, 3.1, 9.6 and 0.94 higher than Sentiment140, HIT, NN and ETSL, respectively.

4.3 (Internal) Comparison within the Model

In order to further verify the effectiveness of our method and analyze which part of our model contributes the most, we carried out the internal comparison within our model. We design the following two simplified versions of our model for comparison:

- **PMI-SO** denotes a PMI-SO based sentiment lexicon with soft sentiment annotation learned in Section 3.1.
- **Doc-Sup** denotes the neural network system with only document-level sentiment supervision. It equals to HSSWE when $\alpha = 0$.

Actually, HSSWE can be viewed as a "combination" of PMI-SO and Doc-Sup. In Tables 5 and 6, we report the comparison results on supervised and unsupervised sentiment classification respectively.

Supervised Sentiment Classification: As is shown in Table 5, two basic models PMI-SO and Doc-Sup show similar performance. They have distinct superiority across different datasets. But both are significantly lower than HSSWE. It shows that by combing the supervision at both document and word levels, it can indeed improve the quality of sentiment-aware word embedding and the subsequent sentiment lexicon.

Unsupervised Sentiment Classification: As is shown in Table 6, the conclusions are similar with that in supervised sentiment classification: HSSWE achieves the significantly better performance.

4.4 Word-level Sentimnt Annotation: Hard vs. Soft

In Section 3.1, we introduce two kinds of word-level sentiment annotation, i.e., soft and hard sentiment annotation. We now compare two methods. The results are reported in Tables 5 and 6. It can be seen that for supervised evaluation, HSSWE (soft) and HSSWE (hard) yield comparative performance. HSSWE (hard) has slight superiority over HSSWE (hard) in Semeval 2013, 2014 and 2016, but HSSWE (hard) is better on Semeval2015. In contrast, for unsupervised evaluation, HSSWE (soft) is significantly better than

Lexicon	Semeval2013	Semeval2014	Semeval2015	Semeval2016	Average
PMI-SO	0.7265	0.7333	0.7008	0.6858	0.7116
Doc-Sup	0.7326	0.7302	0.6814	0.6986	0.7107
HSSWE (soft)	***0.7550***	***0.7424***	0.6921	***0.7097***	***0.7248***
HSSWE (hard)	0.7503	0.7383	***0.7020***	0.7061	0.7242

Table 5: Supervised Evaluation for Internal Comparison (F_1 Score), where HSSWE (hard) and HSSWE (soft) utilize the PMI-SO lexicon with hard sentiment annotation and soft sentiment annotation at the word level, respectively.

Lexicon	Semeval2013	Semeval2014	Semeval2015	Semeval2016	Average
Doc-Sup	0.7252	0.8294	0.7391	0.6859	0.7449
HSSWE (soft)	***0.7734***	***0.8539***	***0.7669***	***0.7206***	***0.7787***
HSSWE (hard)	0.7418	0.8395	0.7633	0.7011	0.7614

Table 6: Unsupervised Evaluation for Internal Comparison (Accuracy)

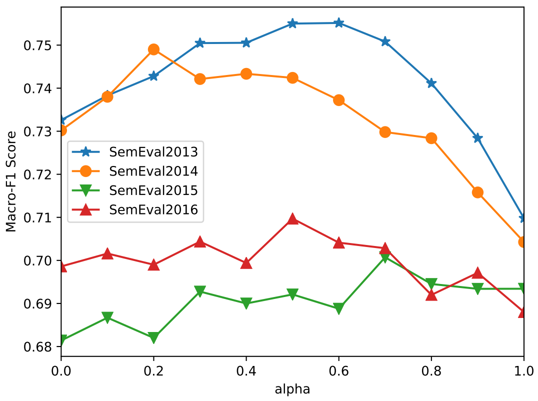


Figure 2: HSSWE (soft) with different α

HSSWE (hard). The average improvement is 1.7 percentage.

4.5 Tuning the Parameter α

In this section, we discuss the tradeoff between two parts of supervisions by turning the tradeoff parameter α. When α is 0, HSSWE only benefits from the document-level sentiment supervision and when α is 1, HSSWE benefits from only word-level sentiment supervision. We observe that HSSWE performs better when α is in the range of [0.45,0.55]. By integrating two component parts of sentiment supervision, HSSWE has significant superiority over that learned from either one.

4.6 Lexicon Analysis

In order to gain more insight of our model and observe the effectiveness of the sentiment lexicon, in Table 7 we extract the positive sentiment score of some representative words learned by different methods. The positive scores are supposed to be: *best*>*better*>*well*. HSSWE captures such comparative sentiment strength but PMI-SO does not. We further observe that in many cases where the results of PMI-SO and Doc-Sup are inconsistent (e.g., Doc-Sup incorrectly predicts *“unreasonable”*, *“boreddddd”* and *“sickkk”* as positive words, but PMI-SO predicts them correctly; PMI-SO incorrectly predicts *“fit”* but Doc-Sup predicts it correctly.), HSSWE often yield the correct results. It shows the advantages of hierarchical sentiment supervision. HSSWE can also correct the sentiment prediction where both PMI-SO and Doc-Sup are inefficient (e.g., *“overplayed”*).

Words	HSSWE	PMI-SO	Doc-Sup
well	0.5740	0.5430	0.7898
better	0.5837	*0.5358 (F)*	0.8440
best	0.9455	0.6823	0.9639
fit	0.2894	*-0.5594 (F)*	0.6076
unreasonable	-0.2441	-0.8421	*0.5275 (F)*
boreddddd	-0.1137	-0.7142	*0.4843 (F)*
sickkkk	-0.3892	-0.7692	*0.1323 (F)*
overplayed	-0.1390	*0.5000 (F)*	*0.8448 (F)*

Table 7: Sentiment Lexicon Analysis, where score with *(F)* means falsely predicted polarity or strength.

5 Conclusion

In this paper, we proposed to construct sentiment lexicons based on a sentiment-aware word representation learning approach. In contrast to traditional methods normally learned based on only the document-level sentiment supervision. We proposed word representation learning via hierarchical sentiment supervision, i.e., under the supervision at both word and document levels. The word-level supervision can be provided based on either predefined sentiment lexicons or the learned PMI-SO based sentiment annotation of words. A wide range of experiments were conducted on several benchmark sentiment classification datasets. The results indicate that our method is quite effective for sentiment-aware word representation, and the sentiment lexicon generated by our approach beats the state-of-the-art sentiment lexicon construction approaches.

Acknowledgements

The work was supported by the Natural Science Foundation of China (No. 61672288), and the Natural Science Foundation of Jiangsu Province for Excellent Young Scholars (No. BK20160085).

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(2009):12.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 595–605.

Vasileios Hatzivassiloglou and Kathleen R McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 174–181.

Bas Heerschop, Alexander Hogenboom, and Flavius Frasincar. 2011. Sentiment lexicon creation from lexical resources. In *International Conference on Business Information Systems*, pages 185–196.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Jaap Kamps, Maarten Marx, Robert J Mokken, Maarten De Rijke, et al. 2004. Using wordnet to measure semantic orientations of adjectives. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, volume 4, pages 1115–1118.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1367.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at 1st International Conference on Learning Representations (ICLR2013)*.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)*, pages 321–327.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 task 10: Sentiment analysis in twitter. In *SemEval@ NAACL-HLT*, pages 451–463.

Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and T-ing Liu. 2014a. Building large-scale twitter-specific sentiment lexicon : A representation learning approach. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 172–182.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014b. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistic-s*, pages 1555–1565.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424.

Peter D Turney. 2008. The latent relation mapping engine: Algorithm and experiments. *Journal of Artificial Intelligence Research*, 33:615–655.

Duy Tin Vo and Yue Zhang. 2016. Dont count, predict! an automatic approach to learning sentiment lexicons for short text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 219–224.

Ke Wang and Rui Xia. 2016. A survey on automatical construction methods of sentiment lexicons. *Acta Automatica Sinica*, 42(4):495–511.

Towards a Universal Sentiment Classifier in Multiple languages

Kui Xu and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{kuixu, wanxiaojun}@pku.edu.cn

Abstract

Existing sentiment classifiers usually work for only one specific language, and different classification models are used in different languages. In this paper we aim to build a universal sentiment classifier with a single classification model in multiple different languages. In order to achieve this goal, we propose to learn multilingual sentiment-aware word embeddings simultaneously based only on the labeled reviews in English and unlabeled parallel data available in a few language pairs. It is not required that the parallel data exist between English and any other language, because the sentiment information can be transferred into any language via a pivot languages. We present the evaluation results of our universal sentiment classifier in five languages, and the results are very promising even when the parallel data between English and the target languages are not used. Furthermore, the universal single classifier is compared with a few cross-language sentiment classifiers relying on direct parallel data between the source and target languages, and the results show that the performance of our universal sentiment classifier is very promising compared to that of different cross-language classifiers in multiple target languages.

1 Introduction

Nowadays, a large amount of user-generated content (UGC) appears online everyday, such as tweets, comments and product reviews. Sentiment classification on these data has become a popular research topic over the past few years (Pang et al., 2002; Blitzer et al., 2007; Agarwal et al., 2011; Liu, 2012). Distributed representations of words or word embeddings have been widely explored, and have proved its great usability for the sentiment classification task (Tang et al., 2014; Zhou et al., 2015; Xu et al., 2015; Bollegala et al., 2016; Ferreira et al., 2016).

Most existing sentiment classifiers rely on labeled training data and the data are usually language-dependent. In other words, a sentiment classifier is learned from a labeled dataset in a specific language and this sentiment classifier can be used for sentiment classification in this language. However, labeled training data for sentiment classification are not available or not easy to obtain in many languages in the world (e.g., Malaysian, Mongolian, Uighur). Without reliable labeled data, it is hard to build a sentiment classifier in these resource-poor languages.

Fortunately, there are a few studies investigating the task of cross-language sentiment classification (Banea et al., 2008; Wan, 2009; Meng et al., 2012; Xiao and Guo, 2013; Gao et al., 2015; Chen et al., 2015; Zhou et al., 2015; Li et al., 2017; Zhou et al., 2016a,b), which aims to make use of the labeled data in a source language (English in most cases) to build a sentiment classifier in a target language. However, cross-language sentiment classification methods rely on parallel data between the source and target languages[1] In a resource-poor language, the parallel data between this language and the source language may not be available or is not easy to obtain. In this circumstance, previous cross-language sentiment classification meth-

[1] Note that a few methods rely on a machine translation system to produce parallel data between the two languages, while the machine translation system is built on a large amount of parallel data between the two languages. In this sense, the methods rely on both the parallel data for machine translation and the pseudo parallel data produced by machine translation systems.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 511–520
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

ods will fail to work.

Another shortcoming of previous cross-language sentiment classification researches is that we have to build an individual cross-language sentiment classifier for each target language, even when we want to perform sentiment classification in a couple of languages at the same time.

In this study, instead of building a sentiment classifier for each target language, we aim to build a universal sentiment classifier in multiple languages and this universal sentiment classifier only learns one single sentiment classification model and it can be applied for sentiment classification in many languages.

In order to achieve this goal, we propose an approach to learn multilingual sentiment-aware word embeddings simultaneously based only on the labeled reviews in English and unlabeled parallel data available in a few language pairs. As mentioned earlier, in some resource-poor languages, there do not exist direct parallel data between these languages and the source English language. In order to address this problem, we propose a pivot-based model to transfer the sentiment information from the source language to any resource-poor language via pivot languages. Finally, a universal sentiment classifier can be built because the multilingual word embeddings are in the same semantic space.

We build three different models (Bilingual Model, Pivot-Driven Bilingual Model and Universal Multilingual Model) and compare them empirically in order to answer two questions in this paper: 1) Can pivot-based models learn bilingual sentiment-aware word embeddings effectively? 2) Can an effective universal sentiment classifier be built for multiple languages?

Without loss of generality, we present and compare the evaluation results of the models in five languages. Evaluation results show that pivot-driven bilingual models perform as well as the bilingual model using direct parallel data, which lays the solid foundation of our universal model. Moreover, it is very promising that our universal sentiment classifier can work well in five languages, and it can achieve very promising classification results as compared to several typical cross-language sentiment classification models.

The main contributions of our study in this paper are summarized as follows:

- We are the first to build a universal sentiment classifier in multiple languages by learning multilingual sentiment-aware word embeddings, which can not be addressed by previous researches on cross-language sentiment classification.
- We propose pivot-based models to bridge two languages in which there do not exist parallel data, and thus the sentiment information can be transferred to any target language.
- Evaluation results on five languages demonstrate the efficacy of our proposed pivot-based models and the universal sentiment classifier.

2 Our Approach

In order to build a universal sentiment classifier, we propose an approach to learn multilingual sentiment-aware word embeddings simultaneously, and then train a universal sentiment classification model in the embedding space by averaging the word embeddings in a document as the document representation. Note that in this study, we focus on only using the labeled data in English and do not make use of any labeled data in other languages, which makes the task more challenging[2]. Formally, we aims to build a single sentiment classifier which can perform sentiment classification in many languages $\{S, T_1, T_2, ..., T_N\}$, where S refers to English language, and T_1 to T_N refer to other N languages.

In our approach, the multilingual sentiment-aware word embeddings play the key role in building the universal sentiment classifier, and now the question is how to learn the multilingual sentiment-aware word embeddings? Inspired by previous studies on cross-lingual sentiment classification and bilingual word embedding learning, we can leverage the labeled data in S (i.e., English) and unlabeled parallel data between S and language T to learn bilingual sentiment-aware word embeddings in both English and T languages with a bilingual model. However, such unlabeled parallel data are not always easy to obtain for all other languages. For a specific language T, if the unlabeled parallel data between T and S do not exist, the bilingual model cannot be applied. In order to address this problem, we propose a pivot-driven bilingual model to leverage pivot languages

[2] Note that the labeled data in other languages can be easily used by our approach in the same way as the English labeled data, and we believe more labeled data will eventually improve the performance of the sentiment classifier.

to bridge T and S. We choose a pivot language P where the parallel data between P and S, and the parallel data between P and T are easy to obtain, and then leverage them to learn the multilingual sentiment-aware word embeddings in the three languages P, T and S. Furthermore, we can leverage more parallel data between multiple languages, some of which are parallel data between S and other languages, and some of which are parallel data within other languages, to build an universal multilingual model. The sentiment information will be directly or indirectly transfered to each language as well, and thus we obtain multilingual sentiment-aware word embeddings in many languages.

The bilingual model, pivot-driven bilingual model and universal multilingual model will be described in next sections, respectively.

2.1 Bilingual Model

The bilingual model tries to induce bilingual word embeddings from a parallel corpus, and in the meantime make similar words from the two languages share adjacent vector representations in the same vector space.

Formally, we assume a source language S with $|S|$ words and a target language T with $|T|$ words. We use s and t to represent a word from S and T, respectively. Given the bilingual parallel corpus $\mathcal{C}$ between language S and T, it can be divided into a corpus $\mathcal{C_S}$ in language S and a corpus $\mathcal{C_T}$ in language T. And we use a notation $S-T$ to indicate a parallel corpus between languages S and T.

Previous studies have proposed some bilingual models for learning bilingual word embeddings, so we extend the well-behaved BiSkip model (Luong et al., 2015) to Bilingual Model (**BM**). This model requires word alignment information, and in this study word alignment is automatically obtained from parallel sentences by using a word alignment tool.

In our bilingual model, every word s in language S is required to predict the adjacent words of itself and the aligned word t in the target language T. For corpus $\mathcal{C_S}$, the monolingual constraint on itself ($\mathcal{C_S} \rightarrow \mathcal{C_S}$) is:

$$Obj(\mathcal{C_S}|\mathcal{C_S}) = \sum_{s\in\mathcal{C_S}} \sum_{w\in adj(s)} \log p(w|s), \quad (1)$$

and the cross-lingual constraint on $\mathcal{C_T}$ ($\mathcal{C_S} \rightarrow \mathcal{C_T}$) is:

$$Obj(\mathcal{C_T}|\mathcal{C_S}) = \sum_{s\in\mathcal{C_S}} \sum_{w\in adj(t), s\leftrightarrow t} \log p(w|s) \quad (2)$$

where $s\leftrightarrow t$ means word $s(\in \mathcal{C_S})$ is aligned to word $t(\in \mathcal{C_T})$ and $adj(s)$ or $adj(t)$ mean the adjacent words of word s or t.

Similarly, for corpus $\mathcal{C_T}$ we can obtain:

$$Obj(\mathcal{C_T}|\mathcal{C_T}) = \sum_{t\in\mathcal{C_T}} \sum_{w\in adj(t)} \log p(w|t), \quad (3)$$

and

$$Obj(\mathcal{C_S}|\mathcal{C_T}) = \sum_{t\in\mathcal{C_T}} \sum_{w\in adj(s), t\leftrightarrow s} \log p(w|t) \quad (4)$$

Combining equations 1, 2, 3 and 4, we get the objective for obtaining bilingual word embeddings from parallel corpus:

$$Obj(\mathcal{C}) = \alpha_1 Obj(\mathcal{C_S}|\mathcal{C_S}) + \alpha_2 Obj(\mathcal{C_T}|\mathcal{C_S})$$
$$+\alpha_3 Obj(\mathcal{C_T}|\mathcal{C_T}) + \alpha_4 Obj(\mathcal{C_S}|\mathcal{C_T})$$

where $\alpha_1, \alpha_2, \alpha_3$ and α_4 are scalar parameters.

We still have to incorporate the sentiment information into the bilingual word embeddings. Similar to previous studies (Zhou et al., 2015), we make use of the sentiment polarity of texts as supervision in the learning process. Given a labeled sentimental corpus $\mathcal{C_L}$[3], we use S^* to represent a sentence in $\mathcal{C_L}$ and w as a word in S^*. And x^T is a sum of word embeddings in S^*. We simply adopt the logistic regression classifier to enforce the sentiment constraint, and thus make the bilingual word embeddings absorb the corresponding sentiment information. The objective function is:

$$L(\mathcal{C_L}) = \sum_{S^*\in\mathcal{C_L}} y \log \sigma(Wx^T + b)$$
$$+ (1-y) \log \sigma(1 - (Wx^T + b)) \quad (5)$$

where y is the label of the sentence S^*, W is a weight vector and b is a bias.

The overall objective function for inducing bilingual sentiment-aware word embeddings is to maximize:

$$Obj(\mathcal{C}) + L(\mathcal{C_L})$$

[3] Note that the labeled corpus is usually provided in the source language S, which means L is S. but the labeled corpus usually does not overlap with the parallel corpus.

2.2 Pivot-Driven Bilingual Model

For some resource-poor target language T, it is quite expensive to get direct parallel corpus between T and the source language S. Without such parallel corpus, it is not possible to apply the above bilingual model to learn bilingual sentiment-aware word embeddings. In order to address this problem we propose our Pivot-Driven Bilingual Model (**PDBM**) by using a pivot language to bridge T and S. The model is inspired by (Wu and Wang, 2007), in which pivot languages are used for phrase-based SMT. A pivot language P is chosen if the parallel corpus between P and S, and the parallel corpus between P and T are easy to obtain. Given two parallel corpora: S-P and P-T, our PDBM model tries to get the trilingual word embeddings by putting constrains on the two corpora. Under the well-designed constraint, the pivot language P can pass the sentiment information from the source language S to the target language T. Similarly, we further assume the pivot language P with $|P|$ words, and use $\mathcal{C_P}$ to denote the corpus in language P.

We design constraints on the two parallel corpora S-P and P-T, instead of direct constraints on S and T. Derived from the BM model, we can get three monolingual constraints $\mathcal{C_S}\rightarrow \mathcal{C_S}$, $\mathcal{C_T}\rightarrow\mathcal{C_T}$, $\mathcal{C_P}\rightarrow\mathcal{C_P}$ and four bilingual constraints $\mathcal{C_S}\rightarrow\mathcal{C_P}$, $\mathcal{C_T}\rightarrow\mathcal{C_P}$, $\mathcal{C_P}\rightarrow\mathcal{C_S}$ and $\mathcal{C_P}\rightarrow\mathcal{C_T}$. The final objective function for learning the trilingual word embeddings can be summarized as:

$$\begin{aligned} Obj_p(\mathcal{C}) = \beta_1 Obj(\mathcal{C_S}|\mathcal{C_S}) + \beta_2 Obj(\mathcal{C_S}|\mathcal{C_P}) \\ +\beta_3 Obj(\mathcal{C_T}|\mathcal{C_T}) + \beta_4 Obj(\mathcal{C_T}|\mathcal{C_P}) \\ +\beta_5 Obj(\mathcal{C_P}|\mathcal{C_S}) + \beta_6 Obj(\mathcal{C_P}|\mathcal{C_T}) \\ +\beta_7 Obj(\mathcal{C_P}|\mathcal{C_P}) \end{aligned}$$

where $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7$ are scalar parameters. Similarly, the objective for enforcing the sentiment constraint is the same as equation 5, so we combine them together to get the overall objective function:

$$Obj_p(\mathcal{C}) + L(\mathcal{C_L})$$

Through the pivot language, the sentiment information can be passed from a source language to a target language by maximizing the above objective function.

2.3 Universal Multilingual Model

The bilingual model and the pivot-driven bilingual model lay the foundations of build a universal multilingual model for sentiment classification in many languages. Given a source language S and a few other languages $\{T_1, T_2, ..., T_N\}$. If there exist parallel data between a language T_i and S, then the bilingual sentiment-aware word embeddings can be learned by the bilingual model. If the parallel data between languages T_i and S are not available, a pivot language can be selected and the pivot-driven model can be applied to learn the trilingual sentiment-aware word embeddings. Even when a single pivot language cannot be found for languages T_i and S, we still can find two or more pivot languages $\{P_1, P_2, ..., P_M\}$ to form a pivot chain and the sentiment information in the source language can be passed through the pivot chain ($S - P_1 - ... - P_M - T_i$) to the target language.

Therefore, in this model, we will make use of all parallel corpora between any pair of languages (including parallel corpora between the source language and any other language, and parallel corpora between other languages) and learn the sentiment-aware word embeddings in all the languages simultaneously. The monolingual objective in each language and the cross-lingual objective for any available parallel corpus are defined in the same way as in the above models, and we sum all the objectives and denote it as $Obj_{universal}(C)$, and this objective is then combined with the sentiment constraint as follows:

$$Obj_{universal}(\mathcal{C}) + L(\mathcal{C_L})$$

By maximizing the above objective function, the sentiment-aware word embeddings in all the languages will be learned.

3 Evaluations

3.1 Dataset

Without loss of generality, we evaluate our models in five languages (including three western languages and two Asian languages): English (en), German (de), French (fr), Japanese (jp) and Chinese (en/zh). Among these languages, the English language is the source language with labeled training data, and we do no use any labeled data in the other languages.

Particularly, we use the multilingual multi-domain Amazon review dataset [4] provided by (Prettenhofer and Stein, 2010) and the NLPCC2013 dataset [5]. The review dataset provided by (Prettenhofer and Stein, 2010) contains labeled data in four languages: English, German, French and Japanese, and the NLPCC2013 dataset further provides labeled data in Chinese. The reviews in each language are divided into three domains: *dvd*, *music* and *books*. Each domain of product reviews contains a balanced training set and test set, each of which consists of 1000 positive and 1000 negative reviews for each language except for Chinese. While for Chinese language, the test set consists of 2000 positive and 2000 negative reviews. We only use English training data as the labeled data in the experiments.

We further obtain unlabeled parallel data from Europarl v7 [6] (Koehn, 2004) (*Eu v7*) and The United Nations Parallel Corpus v1.0 [7] (Ziemski et al., 2016) (*UN v1.0*). The Europarl corpus contains bilingual parallel corpus between English and other 20 Europe languages. The United Nations Parallel Corpus is composed of official records and other parliamentary documents of the United Nations that are in the public domain. These documents are mostly available in the six official languages of the United Nations. Besides, we use the *cldc-2009-004* [8] Chinese-English (*CN-EN*) news parallel corpus and *Japanese-English Bilingual Corpus of Wikipedia's Kyoto Articles Version 2.01* [9] (*JP-EN*), which is created manually by translating Japanese Wikipedia articles (related to Kyoto) into English. In addition, *CJWikiCorpus* (*CN-JP*) is a Chinese-Japanese Parallel Corpus Constructed from Wikipedia [10]

For the the BM model, we use *en-de* ($\in$ *Eu v7*) and *en-fr* ($\in$ *Eu v7*), *en-zh* ($\in$ *CN-EN*), and *en-jp* ($\in$ *JP-EN*).

For the PDBM model, we use *en-fr* ($\in$ *UN v1.0*) with *fr-de* ($\in$ *Eu v7*) to get the case *en-fr-de* (*fr* acts as a pivot), *en-zh* ($\in$ *CN-EN*) with *zh-jp* ($\in$ *CN-JP*) to build the case *en-zh-jp* (*zh* acts as a pivot), *en-zh* ($\in$ *CN-EN*) with *zh-fr* ($\in$ *UN v1.0*) to build *en-zh-fr* (*zh* acts as a pivot), and *en-fr* ($\in$ *Eu v7*) with *zh-fr* ($\in$ *UN v1.0*) to build *en-fr-zh* (*fr* acts as a pivot). Note that any pivot language can be selected if the parallel corpora between the pivot language and other languages can be obtained, but in our experiments, we only use one pivot language in each test case to validate the feasibility of our proposed model. In practice, a popular language (such as English, Chinese) can be used as the pivot because it can act as a link between two unpopular languages.

While for the UMM model, we use all the corpora used in PDBM to build a universal model. All the details can be found in Table 1.

3.2 Comparison Methods

In addition to the comparison between our models, we further compare them with popular cross-lingual (CL) sentiment classification methods.

For comparison in German, French and Japanese, we adopt a few typical CL classification methods, and the results are directly borrowed from the corresponding published papers:

MT-BOW: It is a simple model to train a linear classifier based on the bag-of-words features, and it uses a machine translator to translate the test data into the source language (Prettenhofer and Stein, 2010) .

CL-SCL: It is the cross-lingual structural correspondence learning algorithm proposed by (Prettenhofer and Stein, 2010) and the features in the two languages are mapped to a unified space.

BSE: It is introduced in (Tang and Wan, 2014) by forcing the representations of words from both the source and target languages to share the same feature space. In this way, bilingual word embeddings are learned for cross-lingual sentiment classification.

CR-RL: It is the bilingual word representation learning method of (Xiao and Guo, 2013). It learns different representations for words in different languages. Part of the word vector is shared among different languages and the rest is language dependent. The document representation is calculated by taking average over all words in the document.

Bi-PV: It extends the paragraph vector model into bilingual setting by sharing the document representation of a pair of parallel documents (Pham et al., 2015).

For comparison in Chinese, we adopt several typical CL classification methods:

MT-LR and **MT-SVM**: We use logistic regres-

[4] https://www.uni-weimar.de/medien/webis/corpora/corpus-webis-cls-10/

[5] http://tcci.ccf.org.cn/conference/2013/pages/page04_evares.html

[6] http://www.statmt.org/europarl/v7/

[7] https://conferences.unite.un.org/UNCorpus/

[8] http://www.chineseldc.org/resource_info.php?rid=141

[9] http://alaginrc.nict.go.jp/WikiCorpus/index_E.html

[10] http://lotus.kuee.kyoto-u.ac.jp/ chu/resource/wiki_zh_ja.tgz

Model	Parallel corpora with size	Test case
BM	*en-de* (∈ *Eu v7*, 1.92M)	*en-de*
	en-fr (∈ *Eu v7*, 2.0M)	*en-fr*
	en-zh (∈ *CN-EN*, 1.0M)	*en-zh*
	en-jp (∈ *JP-EN*, 0.5M)	*en-jp*
PDBM	*en-fr* (∈ *UN v1.0*, 2.0M) + *fr-de* (∈ *Eu v7*, 1.5M)	*en-fr-de*
	en-zh (∈ *CN-EN*, 1.0M) + *zh-jp* (∈ *CN-JP*, 0.12M)	*en-zh-jp*
	en-zh (∈ *CN-EN*, 1.0M) + *zh-fr* (∈ *UN v1.0*, 2.0M)	*en-zh-fr*
	en-fr (∈ *Eu v7*, 2.0M) + *zh-fr* (∈ *UN v1.0*, 2.0M)	*en-fr-zh*
UMM	*all the corpora used in PDBM*	*en,de,fr,zh,jp*

Table 1: Parallel corpora used in our models.

sion and SVM to learn different classifiers based on the translated Chinese training data. Bag of words features are used for classification.

Bi-PV: The same as that described above.

BSWE: It uses the bilingual sentiment word embedding algorithm based on denoising autoencoders (Zhou et al., 2015) to learns word representations. Each document is then represented by the sentiment words and the corresponding negation words.

3.3 Settings and Preprocessing

We utilize *cdec* (Dyer et al., 2010) as an alignment tool to get word-level alignment, and we also use it to lowercase the characters in western languages. We use the *stanford-segmenter* [11] to segment Chinese words, and use *Mecab* [12] to segment Japanese words. The *SnowNLP* [13] is used to convert traditional words to simplified ones. Besides, we remove all the irregular characters (e.g., ©, £, ♡) in the texts.

For all the three models, we use stochastic gradient descent (SGD) for learning, with a default learning rate of 0.025, negative sampling with 30 samples, skip-gram with context window of size 5, and a subsampling rate of value 1e-4. The embedding size is set to 400. The training epochs are all set to 10. All the parameters of α and β used in the three models are simply set to 1. The word embeddings in a document are averaged to get the document representation, and then the logistic regression classier is adopted for sentiment classification.

3.4 Results

The sentiment classification results of our three models and the CL classification methods in the three domains and in the German, French and Japanese languages are presented in Table 2. The results in the Chinese language are presented in Table 3. Note that the results of the CL methods are not reported on English test sets, and we only compare our three models on English test sets in Table 3.

First and most importantly, we compare our three models. The BM model relies on the direct parallel data between the source and target languages, and it generally works slightly better than the other models, including the PMDB model and the UMM model. The reason is that direct parallel data can be used for transferring the sentiment information from the source language to the target language directly. However, the performance achieved by the PDBM model is very close to the BM model in most test cases. In some cases (DE-DVD, JP-book and EN-music), the PDBM model can even outperform the BM model. Note that the PDBM model does not leverage the direct parallel data between the source and target languages, but uses a pivot language as a bridge. The results demonstrate that the pivot-driven model is very effective for learning bilingual / trilingual sentiment-aware word embeddings. The results also verify the feasibility of using pivot languages to address the problem of sentiment classification in resource-poor languages, which lays a good foundation for building a universal sentiment classifier in multiple languages. When comparing the UMM model with BM and PDBM, the results of UMM are very close to that of BM and PDBM in most cases, Note that the UMM model does not use the direct parallel corpora of *en-de*

[11] http://nlp.stanford.edu/software/segmenter.shtml

[12] http://taku910.github.io/mecab/

[13] https://github.com/isnowfy/snownlp

TL	Domain	BM	PDBM	UMM	MT-BOW	CL-SCL	BSE	CR-RL	Bi-PV
DE	book	82.46	81.97	81.65	79.68	79.50	80.27	79.89	79.51
	DVD	81.47	82.67	81.27	77.92	76.92	77.16	77.14	78.60
	music	82.95	81.93	81.32	77.22	77.79	77.98	77.27	82.45
FR	book	82.47	81.01	80.27	80.76	78.49	-	78.25	84.25
	DVD	81.86	81.68	80.27	78.83	78.80	-	74.83	79.60
	music	81.51	80.03	79.41	75.78	77.92	-	78.71	80.09
JP	book	70.93	71.59	71.23	70.22	73.09	70.75	71.11	71.75
	DVD	74.62	72.82	72.55	71.30	71.07	74.96	73.12	75.40
	music	76.48	76.26	75.38	72.02	75.11	77.06	74.38	75.45

Table 2: Comparison results (accuracy) on DE (German), FR (French) and JP (Japanese).

TL	Domain	BM	PDBM	UMM	MT-LR	MT-SVM	Bi-PV	BSWE
CN	book	79.7	77.8	78.4	76.5	77.9	78.5	81.1
	DVD	81.7	80.9	79.8	79.6	81.4	82.0	81.6
	music	79.2	77.3	75.8	74.1	70.7	75.3	79.4
EN	book	81.6	80.5	80.2	-	-	-	-
	DVD	81.7	80.8	79.5	-	-	-	-
	music	76.8	78.8	77.9	-	-	-	-

Table 3: Comparison results (accuracy) on CN (Chinese) and EN (English).

and *en-jp*, but relies on pivot-based methods for bridging language gaps. We also find that the different parallel corpora used by the UMM model are of different quality and genres, and if they are used at the same time, they may have some negative influence on each other and thus the learned word embeddings are not always better than the BM and PDBM models using only one or two parallel corpora. What's more, the available parallel data in different language pairs are of various sizes (0.12M $\sim$ 2.0M). Considering all these issues, the results of UMM are promising because the learned single sentiment classifier can work generally well in multiple languages. We believe that if more high-quality and balanced parallel data are used, the performance of the universal sentiment classifier will be improved.

Second, we compare our models with typical CL classification methods. In Table 2, we can see our models can outperform MT-BOW, CL-SCL, and CR-RL in most test cases, and outperform BSE in the German language. Our models can achieve very close results with the other sophisticated CL methods, including Bi-PV. In Table 3, we can see our models can generally outperform MT-LR and MT-SVM, and achieve very competitive results with other strong CL methods, including Bi-PV and BSWE. Most CL classification methods rely on commercial machine translation systems (e.g. Google Translate) for translating the reviews (including the training reviews, the test reviews and additional unlabeled reviews) to get parallel data. Compared with the large amount of parallel data used by commercial machine translation systems, the parallel data used by our models are of a very small size. Though our models are simply based on word embeddings, and the parallel data used by our models are in a small scale, the performance achieved by our models are very competitive.

In Figure 1, we show the visualization of word embeddings learned by the UMM model for some example words. We can see that similar sentiment words in different languages appear nearby with each other. The figure demonstrate that the UMM model are successful in learning sentiment-aware word embeddings in multiple languages.

4 Related Work

The most closely related work is cross-lingual sentiment classification, which aims to leverage the labeled sentiment data from a language with rich sentiment resources (e.g., English) to perform sentiment classification in a target language lacking sentiment resources (e.g., Japanese). Some studies tried to transfer labeled data from the source

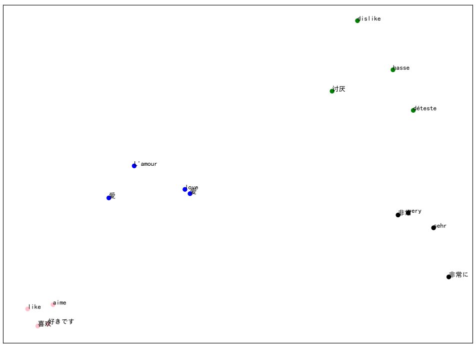


Figure 1: Visualization of word embeddings in UMM (Chinese, Japanese, English, French, German). The similar words are marked in the same color.

language to the target language (Banea et al., 2008; Wan, 2009; Gao et al., 2015; Chen et al., 2015), and some other studies tried to build a unified feature/semantic space in both two languages(Prettenhofer and Stein, 2010; Xiao and Guo, 2013; Zhou et al., 2015, 2016b,a; Li et al., 2017). In the latter case, the sentiment classifier learned in the source language can be used for sentiment classification in both languages. Particularly, Wan (2009) used machine translation to translate the source language to the target language to bridge the gap and applied the co-training approach. Prettenhofer and Stein (2010) provided a CL-SCL model based on structural correspondence learning (SCL) for sentiment classification. Lu et al. (2011) explored to increase the labeled data in both the source and target languages by applying an extra unlabeled parallel data. Xiao and Guo (2013) expected to get cross-lingual discriminative word embeddings to perform the multiple document classification tasks. Their intuitive thought is based on a delicate log-losses function, which aims to increase the probability of the documents with their labels. Like Lu et al. (2011), Meng et al. (2012) also proposed their cross-lingual mixture model to leverage an unlabeled parallel dataset. They intended to learn the previously unseen sentimental words from the big parallel corpus. Some studies have attempted to address multi-lingual sentiment classification (Deriu et al., 2017), but different from our study, they directly leverage training data in multiple languages, by assuming the training data can be obtained directly or in a distant supervision way in each language, and they did not consider the resource or data transfer problem at all.

Word embeddings have shown its great practicable usability in plenty of natural language processing tasks, such as information retrieval (Diaz et al., 2016; Zuccon et al., 2015), machine translation (Shi et al., 2016; Zhang et al., 2014), sentiment analysis (Ren et al., 2016; Xu et al., 2015; Tang et al., 2014) and so on. Bilingual word embeddings have been induced for cross-lingual NLP tasks (Vulić and Moens, 2015; Guo et al., 2014; Zou et al., 2013; Tang et al., 2014; Luong et al., 2015; Zhou et al., 2015). In particular, Luong et al. (2015) proposed the BiSkip model to induce bilingual word embeddings, which is extended from the monolingual skip-gram model in *word2vec* to a bilingual model. They added constraint mutually on both the source language and the target language, while the monolingual model only has constraint on a single language. Zhou et al. (2015) proposed an approach to learning bilingual sentiment word embeddings by using sentiment information of text as supervision, based on labeled corpora and their translations. Ferreira et al. (2016) used a single optimization problem by combining a co-regularizer for the bilingual embeddings with a task-specific loss. However, these methods for inducing bilingual word embeddings usually rely on directly parallel corpus.

5 Conclusion and Future Work

In this paper, we proposed an approach to build a universal sentiment classifier in multiple languages. Particularly we proposed a pivot-based model to transfer the sentiment information from the source language to any resource-poor language via pivot languages. Evaluation results show that the pivot-based model can learn bilingual sentiment-aware word embeddings as well as the bilingual model using direct parallel data. Moreover, the universal sentiment classifier built in the five languages can achieve promising results.

In future work, we will investigate using more advanced document embedding techniques (e.g., CNN, RNN) to directly model document-level sentiment information. We will also extend our model to other languages.

Acknowledgments

This work was supported by NSFC (61331011), 863 Program of China (2015AA015403) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the anonymous reviewers for helpful comments. Xiaojun Wan is the corresponding author.

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the workshop on languages in social media*, pages 30–38. Association for Computational Linguistics.

Carmen Banea, Rada Mihalcea, Janyce Wiebe, and Samer Hassan. 2008. Multilingual subjectivity analysis using machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 127–135. Association for Computational Linguistics.

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Association for Computational Linguistics*, volume 7, pages 440–447.

Danushka Bollegala, Tingting Mu, and J Y Goulermas. 2016. Cross-domain sentiment classification using sentiment sensitive embeddings. *IEEE Transactions on Knowledge and Data Engineering*, 28(2):398–410.

Qiang Chen, Wenjie Li, Yu Lei, Xule Liu, and Yanxiang He. 2015. Learning to adapt credible knowledge in cross-lingual sentiment analysis. In *Association for Computational Linguistics*, pages 419–429.

Jan Deriu, Aurelien Lucchi, Valeria De Luca, Aliaksei Severyn, Simon Müller, Mark Cieliebak, Thomas Hofmann, and Martin Jaggi. 2017. Leveraging large amounts of weakly supervised data for multi-language sentiment classification. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1045–1052. International World Wide Web Conferences Steering Committee.

Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query expansion with locally-trained word embeddings. *Association for Computational Linguistics*, pages 367–377.

Chris Dyer, Jonathan Weese, Hendra Setiawan, Adam Lopez, Ferhan Ture, Vladimir Eidelman, Juri Ganitkevitch, Phil Blunsom, and Philip Resnik. 2010. cdec: a decoder, alignment, and learning framework for finite-state and context-free translation models. In *ACL 2010, Proceedings of the Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, System Demonstrations*, pages 7–12.

Daniel C. Ferreira, Andr F. T. Martins, and Mariana S. C. Almeida. 2016. Jointly learning to embed and predict with multiple languages. In *Meeting of the Association for Computational Linguistics*, pages 2019–2028.

Dehong Gao, Furu Wei, Wenjie Li, Xiaohua Liu, and Ming Zhou. 2015. Cross-lingual sentiment lexicon learning with bilingual word graph label propagation. *Computational Linguistics*.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning sense-specific word embeddings by exploiting bilingual resources. In *International Conference on Computational LinguisticsCOLING*, pages 497–507.

Philipp Koehn. 2004. A parallel corpus for statistical machine translation. *Proceedings of the Third Workshop on Statistical Machine Translation*, (1):3–4.

Nana Li, Shuangfei Zhai, Zhongfei Zhang, and Boying Liu. 2017. Structural correspondence learning for cross-lingual sentiment classification with one-to-many mappings. pages 3490–3496.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Bin Lu, Chenhao Tan, Claire Cardie, and Benjamin K Tsou. 2011. Joint bilingual sentiment classification with unlabeled parallel corpora. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 320–330. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *The Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159.

Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Ge Xu, and Houfeng Wang. 2012. Cross-lingual mixture model for sentiment classification. In *Meeting of the Association for Computational Linguistics: Long Papers*, pages 572–581.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Hieu Pham, Thang Luong, and Christopher Manning. 2015. Learning distributed representations for multilingual text sequences. In *The Workshop on Vector Space Modeling for Natural Language Processing*, pages 88–94.

Peter Prettenhofer and Benno Stein. 2010. Cross-language text classification using structural correspondence learning. In *ACL 2010, Proceedings of the Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 1118–1127.

Yafeng Ren, Ruimin Wang, and Donghong Ji. 2016. A topic-enhanced word embedding for twitter sentiment classification. *Information Sciences*, 369:188–198.

Chen Shi, Shujie Liu, Shuo Ren, Shi Feng, Mu Li, Ming Zhou, Xu Sun, and Houfeng Wang. 2016. Knowledge-based semantic embedding for machine translation. pages 2245–2254.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. pages 1555–1565.

Xuewei Tang and Xiaojun Wan. 2014. Learning bilingual embedding model for cross-language sentiment classification. In *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 02*, pages 134–141. IEEE Computer Society.

Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 363–372. ACM.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *ACL 2009, Proceedings of the Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the Afnlp, 2-7 August 2009, Singapore*, pages 235–243.

Hua Wu and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. *Machine Translation*, 21(3):165–181.

Min Xiao and Yuhong Guo. 2013. Semi-supervised representation learning for cross-lingual text classification. In *Conference on Empirical Methods in Natural Language Processing*, pages 1465–1475.

Ruifeng Xu, Tao Chen, Yunqing Xia, Qin Lu, Bin Liu, and Xuan Wang. 2015. Word embedding composition for data imbalances in sentiment and emotion classification. *Cognitive Computation*, 7(2):226–240.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, Chengqing Zong, et al. 2014. Bilingually-constrained phrase embeddings for machine translation. pages 111–121.

Huiwei Zhou, Long Chen, Fulin Shi, and Degen Huang. 2015. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Association for Computational Linguistics*, pages 430–440.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016a. Attention-based lstm network for cross-lingual sentiment classification.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016b. Cross-lingual sentiment classification with bilingual document representation learning. In *Meeting of the Association for Computational Linguistics*, pages 1403–1412.

Micha Ziemski, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. The united nations parallel corpus v1.0. In *Lrec*.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. pages 1393–1398.

Guido Zuccon, Bevan Koopman, Peter Bruza, and Leif Azzopardi. 2015. Integrating and evaluating neural word embeddings in information retrieval. page 12.

Capturing User and Product Information for Document Level Sentiment Analysis with Deep Memory Network

Zi-Yi Dou
National Key Laboratory for Novel Software Technology, Nanjing University,
Nanjing, 210023, China
141242042@smail.nju.edu.cn

Abstract

Document-level sentiment classification is a fundamental problem which aims to predict a user's overall sentiment about a product in a document. Several methods have been proposed to tackle the problem whereas most of them fail to consider the influence of users who express the sentiment and products which are evaluated. To address the issue, we propose a deep memory network for document-level sentiment classification which could capture the user and product information at the same time. To prove the effectiveness of our algorithm, we conduct experiments on IMDB and Yelp datasets and the results indicate that our model can achieve better performance than several existing methods.

1 Introduction

Sentiment analysis, sometimes known as opinion mining, is the field of study that analyzes people's opinions, sentiments, evaluations, attitudes and emotions from written language. It is one of the most active and critical research areas in natural language processing (Liu, 2012). On the one hand, from the industry point of view, knowing the feelings among consumers based on their comments is beneficial and may support strategic market decisions. On the other hand, potential customers are often interested in other people's opinion in order to find out the choices that best fits their preferences (Moraes et al., 2013).

Previous studies tackled the sentiment analysis problem at various levels of granularity, from document level to sentence level due to different objectives of applications (Zhang et al., 2009). In this work, we mainly focus on document-level sentiment classification Basically, the task is to predict user's overall sentiment or polarity in a document about a product (Pang and Lee, 2008).

Most existing methods mainly utilize local text information whereas ignoring the influences of users and products (Tang et al., 2015). As is often the case, there are certain consistencies for both users and products. To illustrate, lenient users may always give higher ratings than fastidious ones even if they post the same review. Also, it is not surprising that some products may always receive low ratings because of their poor quality and vice versa. Therefore, it is necessary to leverage individual preferences of users and overall qualities of products in order to achieve better performance.

Tang *et al.* (2015) proposed a novel method dubbed User Product Neural Network (UPNN) which capture user- and product-level information for sentiment classification. Their approach has shown great promise but one major drawback of their work is that for users and products with limited information, it is hard to train the representation vector and matrix for them.

Inspired by the recent success of computational models with attention mechanism and explicit memory (Graves et al., 2014; Sukhbaatar et al., 2015), we addressed the aforementioned issue by proposing a method based on deep memory network and Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). The model can be divided into two separate parts. In the first fart, we utilize LSTM to represent each document. Afterwards, we apply deep memory network consists of multiple computational layers to predict the ratings for each document and each layer is a content-based attention model.

To prove the effectiveness of our algorithm, we have conducted experiments on three datasets derived from IMDB and Yelp Dataset Challenge and compare to several other algorithms. Experimental results show that our algorithm can outperform

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 521–526
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

baseline methods for sentiment classification of documents by leveraging users and products for document-level sentiment classification.

2 Related Work

2.1 Memory Network

In 2014, Weston *et al.* (2014) introduced a new class of learning models called memory networks. Memory networks reason with inference components combined with a long-term memory component. The long-term memory can be read and written to and then it can be used for prediction. Generally, a memory network consists of an array of objects called memory m and four components I, G, O and R, where I converts input to internal feature representation, G updates old memories, O generates an output representation and R outputs a response.

Based on their work, Sukhbaatar *et al.* (2015) proposed a neural network with a recurrent attention model over a possibly large external memory. Unlike previous model, their model is trained end-to-end and hence requires significantly less supervision during training. They have shown that their model yields improved results in language model and question answering.

Inspired by the success of memory network, Tang *et al.* (2016) introduce a deep memory network for aspect-level sentiment classification. The architecture of their model is similar to the previous model and experimental results demonstrate that their approach performs comparable to other state-of-the-art systems. Also, Li *et al.* (2017) decompose the task of attitude identification into two separate subtasks: target detection and polarity classification; and then solve the problem by applying deep memory network so that signals produced in target detection provide clues for polarity classification and the predicted polarity provides feedback to the identification of targets.

2.2 Sentiment Classification

Most existing work tackle the problem of sentiment classification by manually design effective features. such as text topic (Ganu et al., 2009) and bag-of-opinion (Qu et al., 2010) . Some work take user information into consideration. For example, in 2013, Gao *et al.* (2013) design user-specific features to capture user leniency. Also, Li *et al.* (2014) incorporate textual topic and user-word factors with supervised topic modeling.

Tang *et al.* (2015) points out that it is critical to leverage users and products for document-level sentiment classification. They assume there are four types of consistencies for sentiment classification and validate the influences of users and products in terms of sentiment and text on massive IMDB and Yelp reviews. Their model represent each user and product as both vector and matrix in order to capture the consistencies and then apply convolutional neural network to solve the task.

To the best of our knowledge, no one has ever applied deep memory network to capture the user and product information and solve the tasks in sentiment classification at document-level.

3 Proposed Methods

In this section, we present the details of User Product Deep Memory Network (UPDMN) for sentiment classification at document level.

3.1 Basic Symbol and Definition

First we suppose U, P, D is the set of users, products and documents respectively. If user $u \in U$ writes a document $d \in D$ about a product $p \in P$ and give the rating, we denote $U(d) = \{ud | ud \text{ is written by } u, ud \neq d\}$ and $P(d) = \{pd | pd \text{ is written about } p, pd \neq d\}$. Then, our task can be formalized as follows: suppose u write a document d about a product p , we should output the predicted score y for the document d based on the input $< d, U(d), P(d) >$. The detail of these symbols would be illustrated in the following part.

3.2 General Framework of UPDMN

Figure 1 illustrates the general framework of our approach. Basically, inspired by the use of memory network in question answering and aspect-level sentiment analysis (Sukhbaatar et al., 2015; Tang et al., 2016), our model consists of multiple computational layers (hops), each of which contains an attention layer and a linear layer.

For every document in $U(d)$ and $P(d)$, we embed it into a continuous vector d_i and store it in the memory. The model writes all document to the memory up to a fixed buffer size. Suppose we are given $\{d_i\} = \{d_1, ..., d_n\}$ to be stored in memory, for each layer we can convert them into memory vectors $\{m_i\}$ using an embedding matrix. The document d should also be embedded into q. Then, we compute the match $\{p_i\}$ between q and each memory m_i. Afterwards, we embed $\{d_i\}$ into

Dataset	#users	#products	#reviews	#docs/user	#docs/product	#sents/doc	#words/doc
IMDB	1,310	1,635	84,919	64.82	51.94	16.08	394.6
Yelp 2014	4,818	4,194	231,163	47.97	55.11	11.41	196.9
Yelp 2013	1,631	1,633	78, 966	48.42	48.36	10.89	189.3

Table 1: Statistical information of datasets.

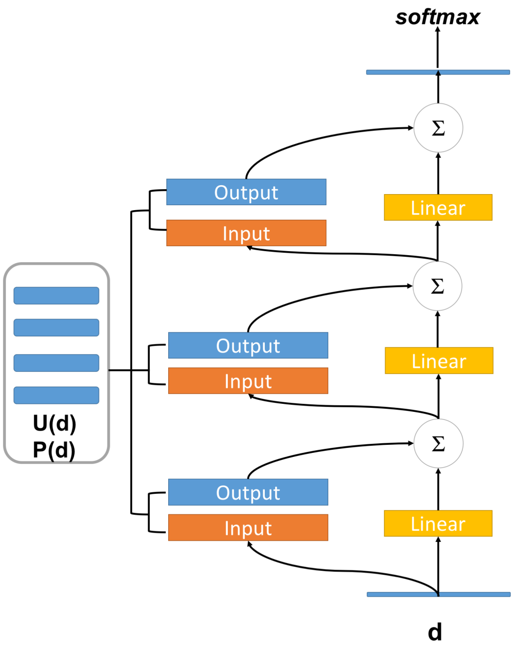


Figure 1: General Framework

output vector $\{c_i\}$ using another embedding matrix and generate the output of attention layer. The output is further summed with the linear transformation of q and considered as the input of next hop. The output vector at last hop is fed into a *softmax* layer and then generates the final prediction y for document-level sentiment classification.

3.3 Embedding Documents

Although there are several state-of-the-art techniques to embed word into vectors (Mikolov et al., 2013a), for document-level sentiment classification, the document we need to classify is usually too long to be represented as a vector. People have tried different ways to solve the task. For example, Kalchbrenner *et al.* (2014) apply convolutional neural network for modeling sentences and Li *et al.* (2015) introduce an LSTM model that hierarchically builds an embedding for a paragraph from embeddings for sentences and words.Some of these work can be incorporated into our methods. However, here we only use the LSTM model to embed each document, *i.e.* every word in the document is fed into LSTM and the final representation is obtained by averaging the hidden state of each word, and the experimental results shows that this simple embedding method can actually obtain satisfactory results.

3.4 Attention Model

After obtaining the embedding vector q for document d the memory vectors $\{m_i\}$ for each memory, we calculate the match between q and m_i using the following equation:

$$p_i = softmax(W_{att}[m_i; q] + b_{att}) \qquad (1)$$

where $softmax(z_i) = e^{z_i} / \sum_j e^{z_j}$.

Afterwards, we compute the corresponding output o for each hop by summing over the c_i, weighted by the probability vector from the input:

$$o = \sum_i p_i c_i \qquad (2)$$

3.5 Final Prediction and Training Strategy

At last hop, the output vector is fed into a *softmax* layer and thus generates a probability distribution $\{y_i\}$ over ratings. The score with the highest probability would be considered as our final prediction py. During training, we try to minimize the cross entropy error of sentiment classification in a supervised manner. The specific equation is shown as follows:

$$Loss = -\sum_{d \in D} \sum_{y_i \in Y} \mathbb{I}(y = y_i|d) log(P(y = y_i|d)) \qquad (3)$$

where Y is the collection of sentiment categories, $\mathbb{I}(y = y_i|d)$ is 1 or 0, indicating whether the correct category for d is y_i, and $P(y = y_i|d)$ represents the probability of classifying document d as category y_i.

4 Experiment

In this section, we will first discuss the experimental setting and then display the results.

	IMDB			Yelp 2014			Yelp 2013		
	Acc	MAE	RMSE	Acc	MAE	RMSE	Acc	MAE	RMSE
Majority	0.196	1.838	2.495	0.392	0.779	1.097	0.411	0.744	1.060
Trigram	0.399	1.147	1.783	0.577	0.487	0.804	0.569	0.513	0.814
TextFeature	0.402	1.134	1.793	0.572	0.490	0.800	0.556	0.520	0.845
AvgWordvec + SVM	0.304	1.361	1.985	0.530	0.562	0.893	0.526	0.568	0.898
SSWE + SVM	0.312	1.347	1.973	0.557	0.523	0.851	0.549	0.529	0.849
Paragraph Vector	0.341	1.211	1.814	0.564	0.496	0.802	0.554	0.515	0.832
RNTN + Recurrent	0.400	1.133	1.764	0.582	0.478	0.821	0.574	0.489	0.804
Trigram + UPF	0.404	1.132	1.764	0.576	0.471	0.789	0.570	0.491	0.803
TextFeature +UPF	0.402	1.129	1.774	0.579	0.476	0.791	0.561	0.509	0.822
JMARS	N/A	1.285	1.773	N/A	0.710	0.999	N/A	0.699	0.985
UPNN	0.435	0.979	1.602	0.608	0.447	0.764	0.596	0.464	0.784
UPDMN(1)	0.428	0.936	1.443	0.588	0.457	0.757	0.596	0.454	0.747
UPDMN(2)	0.446	0.944	1.495	0.592	0.451	0.748	0.602	0.437	0.732
UPDMN(3)	0.459	0.883	1.397	0.599	0.444	0.742	0.627	0.386	0.681
UPDMN(4)	**0.465**	**0.853**	**1.351**	0.609	0.432	0.731	**0.639**	**0.369**	**0.662**
UPDMN(5)	0.456	0.928	1.471	**0.613**	**0.425**	**0.720**	0.611	0.405	0.704

Table 2: Experimental results.

4.1 Experimental Settings

We use the same datasets as Tang *et al.* (2015), which are derived from IMDB (Diao et al., 2014) and Yelp Dataset Challenge in 2013 and 2014 [1]. Statistical information of the datasets are given in Table 1.

In order to measure the performance of our model, here we use three metrics. Specifically, we use $accuracy$ to measure the overall sentiment classification performance, MAE and $RMSE$ to measure the divergences between prediction py and ground truth gy. The formulas for these three metrics are listed as follows:

$$accuracy = \frac{T}{N} \tag{4}$$

$$MAE = \frac{\sum_i |py_i - gy_i|}{N} \tag{5}$$

$$accuracy = \sqrt{\frac{\sum_i (py_i - gy_i)^2}{N}} \tag{6}$$

4.2 Baseline Models

We compare UPDMN with the following models:

(1) **Majority** : it assigns each review in the test dataset with the majority sentiment category in training set.

(2) **Trigram** : it first takes unigrams, bigrams and trigrams as features and then trains a classifier with SVM (Fan et al., 2008).

(3) **TextFeature** : it takes hard-crafted text features such as word/character n-grams, negation features and then trains a classifier with SVM.

(4) **UPF**: it extracts user-leniency features and corresponding product features from training data and then concatenates them with features in model (2) and (3) (Gao et al., 2013).

(5) **AvgWordvec+SVM** : it learns word embeddings from training and development sets with $word2vec$, averages word embeddings and then trains an SVM classifier (Mikolov et al., 2013b).

(6) **SSWE+SVM** : it learns sentiment-specific word embeddings (SSWE), uses max/min/average pooling to generate document representation and then trains an SVM classifier (Tang et al., 2014).

(7) **RNTN+RNN** : it represents each sentence with RNTN, composes document with recurrent neural network , and then averages hidden vectors of recurrent neural network as the features (Socher et al., 2013).

(8) **Paragraph Vector**: it implements the PVDM for document-level sentiment classification (Le and Mikolov, 2014).

(9) **JMARS**: it is the recommendation algorithm which leverages user and aspects of a review with collaborative filtering and topic modeling (Diao et al., 2014).

(10) **UPNN** : as has been stated above, it also leverages user and product information for sentiment classification at document level (Tang et al., 2015).

4.3 Experimental Results and Discussion

The experimental results are given in Table 2. The results of baseline models are reported in (Tang et al., 2015). Our model is abbreviated to UPDMN(k), where k is the number of hops. With the increase of the number of hops, the performance of UPDMN will get better intially, which indicates that multiple hops can indeed capture

[1] http://www.yelp.com/dataset_challenge

more information to improve the performance. However, if there are too many hops, the performance would be not as well as before, which may be caused by over-fitting.

Compared with other models, we can see that with proper setting, our model achieve superior results. All these results prove the effectiveness of UPDMN and the necessity to utilizing user and product information at document level.

It should be noticed that there are still several improvements can be made, such as better representation of documents or more sophisticated attention mechanism. We believe that our model has great potential and can be improved in many ways.

Acknowledgments

We thank anonymous reviewers and Hao Zhou for helpful advice.

References

Qiming Diao, Minghui Qiu, Chao Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, pages 193–202.

Rong En Fan, Kai Wei Chang, Cho Jui Hsieh, Xiang Rui Wang, and Chih Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9(9):1871–1874.

Gayatree Ganu, Noemie Elhadad, and Amlie Marian. 2009. Beyond the stars: Improving rating predictions using review text content. In *International Workshop on the Web and Databases, WEBDB 2009, Providence, Rhode Island, Usa, June*.

Wenliang Gao, Naoki Yoshinaga, Nobuhiro Kaji, and Masaru Kitsuregawa. 2013. Modeling user leniency and product popularity for sentiment classification. In *IJCNLP*, pages 1107–1111.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *Computer Science*.

Seppu Hochreiter and Jrgen Schmidhuber. 1997. *Long short-term memory*. Springer Berlin Heidelberg.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *Eprint Arxiv*, 1.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *Computer Science*, 4:1188–1196.

Fangtao Li, Sheng Wang, Shenghua Liu, and Ming Zhang. 2014. Suit: a supervised user-item based topic model for sentiment analysis. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1636–1642.

Jiwei Li, Minh Thang Luong, and Jurafsky Dan. 2015. A hierarchical neural autoencoder for paragraphs and documents. *Computer Science*.

Bing Liu. 2012. Sentiment analysis and opinion mining. In *Synthesis Lectures on Human Language Technologies*, page 167.

Qiaozhu Mei, Qiaozhu Mei, and Qiaozhu Mei. 2017. Deep memory networks for attitude identification. pages 671–680.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119.

Rodrigo Moraes, Jo Valiati, and Wilson P O Neto. 2013. Document-level sentiment classification: An empirical comparison between svm and ann. *Expert Systems with Applications*, 40(2):621–633.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(12):1–135.

Lizhen Qu, Georgiana Ifrim, and Gerhard Weikum. 2010. The bag-of-opinions method for review rating prediction from sparse text patterns. In *COLING 2010, International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China*, pages 913–921.

R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. *Computer Science*.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, pages 1014–1023.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Conference on Empirical Methods in Natural Language Processing*, pages 214–224.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Meeting of the Association for Computational Linguistics*, pages 1555–1565.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *Eprint Arxiv*.

Changli Zhang, Daniel Zeng, Jiexun Li, Fei Yue Wang, and Wanli Zuo. 2009. Sentiment analysis of chinese documents: From sentence to document level. *Journal of the Association for Information Science and Technology*, 60(12):2474–2487.

Identifying and Tracking Sentiments and Topics from Social Media Texts during Natural Disasters

Min Yang[1,4], **Jincheng Mei**[2], **Heng Ji**[3], **Wei Zhao**[4], **Zhou Zhao**[5], **Xiaojun Chen**[1*]
[1] Shenzhen University [2] University of Alberta
[3] Rensselaer Polytechnic Institute [4] Tencent AI Lab
[5] Zhejiang University

Abstract

We study the problem of identifying the topics and sentiments and tracking their shifts from social media texts in different geographical regions during emergencies and disasters. We propose a **l**ocation-based **d**ynamic **s**entiment-**t**opic model (LDST) which can jointly model topic, sentiment, time and Geolocation information. The experimental results demonstrate that LDST performs very well at discovering topics and sentiments from social media and tracking their shifts in different geographical regions during emergencies and disasters[1].

1 Introduction

Social media has become pervasive in our daily life, and it is a great way to spread important information efficiently. Using social media (e.g., Twitter, Facebook, Pinterest), people can conveniently inform others and express support during emergencies and disasters.

Nowadays, the social media is keeping producing a huge amount of information. Unlike before, people are not only interested in identifying the static topics and sentiments from given texts, but also, perhaps more concerned with tracking the evolution of topics and sentiments among different geographical regions. On the one hand, this new requirement can be very helpful, especially in the case of emergencies, such as pre-disaster preparation and post-disaster relief in local natural disasters (Beigi et al., 2016). For example, people's sentiments on the topics related to *medical* and *rescue* may guide the management and distribution of emergency supplies. On the other hand, the existing models do not take the temporal evolution and the impact of location over topics and sentiments into consideration, which makes them unable to fulfill the new requirement of tracking the evolution of topics and sentiments in different geographical places.

In this paper, we aim to identify the topics and sentiments and track their shifts in different geographical regions during emergencies and disasters. We are inspired by several observations. First, people are interested in not only the overall sentiment or topic distribution of the documents but also the sentiments towards specific topics. For example, a person may be happy with that the disaster passed away, but at the meanwhile he/she may be unsatisfied with the post-disaster relief. Second, most existing sentiment-topic models ignore the temporal evolution of topics and sentiment in a time-variant data corpus such as the Twitter stream. There are strong evidences which indicate that people's attitudes toward a disaster will gradually change over time with the distribution of emergency supplies (Beigi et al., 2016; Caragea et al., 2014; Mandel et al., 2012). Third, people in different places tend to have different opinions towards particular topics. This motivated us to find the influence of specific topics and the relationship between different topics in different regions, which may improve people's awareness to help themselves during disasters.

We propose a **l**ocation-based **d**ynamic **s**entiment-**t**opic model (LDST) which generalizes latent Dirichlet allocation (LDA) (Blei et al., 2003), by jointly modeling topic, sentiment, time and geolocation information. After learning the LDST model, we can identify the topics and sentiments held by people in different locations over time. Our model works in an unsupervised way, and we learn the model according to the frequency of terms co-occurring in different

[*] Corresponding author (xjchen@szu.edu.cn)

[1] Data and codes are available at https://goo.gl/uee3QK

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 527–533
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

contexts. To leverage the prior knowledge, we construct a small set of seed words for each topic of interest to enable the model to group semantically related terms into the same topic. Consequently, the topic words will be more related to the seed words of the same topic.

We conduct experiments using a Hurricane Sandy Twitter corpus which consists of 159,880 geotagged Twitter posts from the geographic area and time period of the 2012 Hurricane Sandy. We show the evolution of people's topics and sentiments, which change according to not only the time the disaster happens, but also people's locations during the hurricane Sandy.

2 Related work

Sentiment analysis is widely applied in many fields, such as business intelligence, politics, sociology. The papers by Pang and Lee (Pang and Lee, 2008) and Liu (Liu, 2012) described most of the existing techniques for sentiment analysis and opinion mining, which could be categorized into lexicon-based approaches (Kennedy and Inkpen, 2006; Turney, 2002; Yang et al., 2014a,b) and corpus-based approaches (Pang et al., 2002; Yang et al., 2015; Wan, 2009).

Recently, researchers have turned their attention to exploring sentiment analysis on the social media posts of individuals during natural disasters and emergencies (Beigi et al., 2016; Buscaldi and Hernandez-Farias, 2015; Caragea et al., 2014; Kryvasheyeu et al., 2015; Mandel et al., 2012; Shalunts et al., 2014). For example, a sentiment analysis system is applied for Italian to a set of tweets during the Genoa flooding (Buscaldi and Hernandez-Farias, 2015). They attempted to identify trending topics, toponym and sentiments that might be relevant from a disaster management perspective. However, the existing studies only focused on sentiment analysis on the document level, without considering the specific topics in the document.

Meanwhile, topic models, such LDA (Blei et al., 2003), have become popular in extracting interesting topics. Some recent work incorporates context information into LDA, such as time (Wang and McCallum, 2006; Zhao et al., 2014) and authorship (Steyvers et al., 2004; Yang et al., 2016) to make topic models fit expectations better. Some studies also attempt to detect sentiment and topic simultaneously from documents (Dermouche et al., 2015; Lin et al., 2012; Mukherjee et al., 2014; He et al., 2012). Nevertheless, none of existing methods takes advantage of temporal and geographical information to identify and track people's topics and sentiment during emergencies and disasters.

3 Model

In this section, we firstly introduce the generative process of the LDST model. Then we present the inference algorithm for estimating the model parameters.

3.1 Model Description

We assume the corpus consists a set of authors, a set of locations, and a collection of documents with specific timestamps. Formally, we use V and U to denote the sets of locations and authors, respectively. A document $d \in D$ is a short text written by an author $u \in U$ in location $v \in V$ at time t. Also, let S be the number of distinct sentiment labels, and T be the total number of topics, where S and T are predefined constant values. Since each tweet is a short text, studying them individually is not very informative. We thus use pooling methods to construct aggregated documents for each location or each author. For a venue v, we use A_v to define the set of all authors that have written documents in location v, and d_v (location document) to refer to the union of all the documents written in location v. N_{d_v} is the number of words in location document d_v.

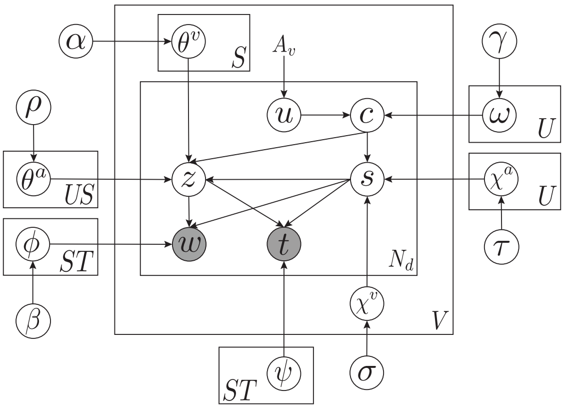


Figure 1: Graphical representation of our algorithm

Our model generalizes LDA by jointly modeling topic, sentiment, time and geolocation information. Figure 1 shows the graphical model of

LDST. The formal definition of the generative process of LDST is as follows:

1. For each sentiment label s,
   a. For each topic z under sentiment s,
      – Draw a word distribution $\phi_{s,z}$ ~ $Dir(\beta_{s,z})$.
2. For each author u,
   a. Draw $\omega_u \sim Beta(\gamma)$
   b. Draw a sentiment distribution $\chi_u^{(a)} \sim Dir(\tau)$
   c. For each sentiment label s,
      – Draw a topic distribution $\theta_{u,s}^{(a)} \sim Dir(\rho)$
3. For each location document d_v,
   a. Draw a sentiment distribution $\chi_{d_v}^{(v)} \sim Dir(\sigma)$
   b. For each sentiment label s,
      – Draw a topic distribution $\theta_{d_v,s}^{(v)}$ ~ $Dir(\alpha)$
   c. For each term $w_{d_v i}$ in the location document d_v,
      – Draw a author $u \sim A_v$ uniformly,
      – Draw a switch $c \sim Bernoulli(\omega_u)$
      – if $c = 0$,
        * Draw a sentiment $s_{d_v i} \sim Mult(\chi_u^{(a)})$
        * Draw a topic $z_{d_v i}$ from $Mult(\theta_{u,s_{d_v i}}^{(a)})$
      – if $c = 1$,
        * Draw a sentiment $s_{d_v i} \sim Mult(\chi_{d_v}^{(v)})$
        * Draw a topic $z_{d_v i} \sim Mult(\theta_{d_v,s_{d_v i}}^{(v)})$
      – Draw a word $w_{d_v i} \sim Mult(\phi_{s_{d_v i}, z_{d_v i}})$
      – Draw a timestamp $t_{d_v i}$ from $Beta(\psi_{s_{d_v i}, z_{d_v i}})$

In the model, α, β, ρ, σ, τ and γ are hyperparameters. The latent sentiments and topics depend on the document venues and personalities of the author. We use ω to control the influence from the venue and the author. In particular, ω is the parameter of a Bernoulli distribution, from which a binary variable c is generated to determine whether the document is influenced by venue or user.

3.2 Inference Algorithm

We use Collapsed Gibbs sampling (Porteous et al., 2008) to estimate the unknown latent variables $\{\phi, \omega, \theta^a, \theta^v, \chi^a, \chi^v, \psi\}$. The posterior distribution of the hidden variables for each word $w_{d_v,i}$ (i-th word in venue document d_v) is calculated as follows (to simplify, we use Θ to refer to <$\mathbf{u}^{-d_v,i}, \mathbf{c}^{-d_v,i}, \mathbf{s}^{-d_v,i}, \mathbf{z}^{-d_v,i}, \mathbf{w}, \mathbf{t}, A_v$>):

$$\begin{aligned}
& P\left(u_{d_v,i} = u, c_{d_v,i} = 0, s_{d_v,i} = s, z_{d_v,i} = z \mid \Theta\right) \\
\propto = & \frac{n_{u,c}^{-d_v,i}(0) + \gamma'}{n_{u,c}^{-d_v,i} + \gamma + \gamma'} \cdot \frac{n_{u,s}^{-d_v,i} + \tau_s}{\sum_{s'} (n_{u,s'}^{-d_v,i} + \tau_{s'})} \\
& \cdot \frac{n_{u,s,z}^{-d_v,i} + \rho_z}{\sum_{z'} (n_{u,s,z'}^{-d_v,i} + \rho_{z'})} \cdot \frac{n_{s,z,w}^{-d_v,i} + \beta_w}{\sum_{w'} (n_{s,z,w'}^{-d_v,i} + \beta_{w'})} \\
& \cdot \frac{(1-t)^{\psi_{s,z}^{(1)}-1} \cdot t^{\psi_{s,z}^{(2)}-1}}{\mathrm{Beta}(\psi_{s,z}^{(1)}, \psi_{s,z}^{(2)})} \qquad (1)
\end{aligned}$$

$$\begin{aligned}
& P\left(u_{d_v,i} = u, c_{d_v,i} = 1, s_{d_v,i} = s, z_{d_v,i} = z \mid \Theta\right) \\
\propto = & \frac{n_{u,c}^{-d_v,i}(1) + \gamma}{n_{u,c}^{-d_v,i} + \gamma + \gamma'} \cdot \frac{n_{v,s}^{-d_v,i} + \sigma_s}{\sum_{s'} (n_{v,s'}^{-d_v,i} + \sigma_{s'})} \\
& \cdot \frac{n_{v,s,z}^{-d_v,i} + \alpha_z}{\sum_{z'} (n_{v,s,z'}^{-d_v,i} + \alpha_{z'})} \cdot \frac{n_{s,z,w}^{-d_v,i} + \beta_w}{\sum_{w'} (n_{s,z,w'}^{-d_v,i} + \beta_{w'})} \\
& \cdot \frac{(1-t)^{\psi_{s,z}^{(1)}-1} \cdot t^{\psi_{s,z}^{(2)}-1}}{\mathrm{Beta}(\psi_{s,z}^{(1)}, \psi_{s,z}^{(2)})} \qquad (2)
\end{aligned}$$

where $n_{u,c}(0)$ and $n_{u,c}(1)$ are the numbers of times that $c = 0$ and $c = 1$ are sampled for user u, respectively, and we have $n_{u,c} \triangleq n_{u,c}(0)$ +$n_{u,c}(1)$. $n_{u,s}$ is the number of times that sentiment s is sampled from the distribution χ_u^a specific to user u, and $n_{v,s}$ is the number of times that sentiment s is sampled from the distribution $\chi_{d_v}^v$ specific to document venue v. $n_{u,s,z}$ is the number of times that topic z is sampled from the distribution $\theta_{u,s}^a$ specific to user u and sentiment s, and $n_{v,s,z}$ is the number of times that topic z is sampled from the distribution$\theta_{d_v s}^v$ specific to document venue v and sentiment s. $n_{s,z,w}$ is the number of times that word w is sampled from the distribution $\phi_{s,z}$ specific to sentiment s and topic z. The superscript $-d_v, i$ denotes a quantity excluding the current word $-d_v, i$.

After Gibbs sampling, $\{\phi, \omega, \theta^a, \theta^v, \chi^a, \chi^v, \psi\}$ can be estimated as follows:

$$\hat{\phi}_{s,z,w} = \frac{n_{s,z,w} + \beta_w}{\sum_{w'} (n_{s,z,w'} + \beta_{w'})},$$

$$\hat{\omega}_u = \frac{n_{uc}(1) + \gamma}{n_{u,c} + \gamma + \gamma'},$$

$$\hat{\theta}_{v,s,z}^v = \frac{n_{v,s,z} + \alpha_z}{\sum_{z'} (n_{v,s,z'} + \alpha_{z'})},$$

$$\hat{\theta}_{u,s,z}^a = \frac{n_{u,s,z} + \rho_z}{\sum_{z'} (n_{u,s,z'} + \rho_{z'})},$$

$$\hat{\chi}_{u,s}^a = \frac{n_{u,s} + \tau_s}{\sum_{s'} (n_{u,s'} + \tau_{s'})},$$

$$\hat{\chi}_{d_v,s}^v = \frac{n_{v,s} + \sigma_s}{\sum_{s'} (n_{v,s'} + \sigma_{s'})},$$

$$\hat{\psi}_{s,z}^{(1)} = \bar{m}_{s,z} \cdot \left(\frac{\bar{m}_{s,z} \cdot (1 - \bar{m}_{s,z})}{\xi_{s,z}^2} - 1 \right),$$

$$\psi_{s,z}^{(2)} = (1 - \bar{m}_{s,z}) \cdot \left(\frac{\bar{m}_{s,z} \cdot (1 - \bar{m}_{s,z})}{\xi_{s,z}^2} - 1 \right).$$

where $\overline{m}_{sz}$ and ξ_{sz}^2 indicate the sample mean and the biased sample variance of the timestamps belonging to sentiment s and topic z, respectively.

3.3 Defining the Prior Knowledge

In our model, the prior knowledge is employed to guide the generative process of topics. The prior knowledge can be obtained from natural disaster experts. We collect a small set of seed words for each topic of interest during emergency and disaster. For each topic, the LDST model draws the word distribution from a biased Dirichlet prior $Dir(\beta)$. Each vector $\beta_{.,z} \in \mathbb{R}^V$ is constructed from the sets of seed words, where

$$\beta_{.,z} := \lambda_1(1^V - \Lambda_{.,z}) + \lambda_2\Lambda_{.,z} \qquad (3)$$

Here, $\Lambda_{.,z,w} = 1$ if and only if word w is a seed word for topic z, otherwise $\Lambda_{.,z,w} = 0$. The scalars λ_1 and λ_2 are hyperparameters. Intuitively, when $\lambda_1 < \lambda_2$, the biased prior ensures that the seed words are more probably drawn from the associated topic.

4 Experiments

4.1 Hurricane Sandy Twitter Datasets

This dataset contains nearly 15 million tweets posted on Twitter while Hurricane Sandy was hitting the United States. Tweets were collected from October 25, 2012 to November 4, using the keywords 'hurricane' and 'sandy' (Zubiaga and Ji, 2014). In this paper, we only keep the geotagged tweets. The final experimental dataset consists of 159,880 geotagged tweets. The original geographical information is expressed by using longitude and latitude in decimal degree. We set the granularity of location as a state via Google Maps Geocoding API[2] and analyze the tweets within the United States.

4.2 Baseline Methods

We evaluate and compare our model with several baseline methods as follows:

LDA: We use gensim toolkit to do inference for LDA model (Blei et al., 2003).

ToT: Topics over Time, a non-Markov continuous time model proposed in (Wang and McCallum, 2006).

JST: The first Joint Sentiment-Topic model to identify the sentiment-topic pairs (Lin and He, 2009) .

TS: Topic-Sentiment model proposed in (Dermouche et al., 2015).

LDST-w/oS: This is the LDST model without employing prior knowledge (seed words). We use this method to evaluate the influence of seed words.

4.3 Implementation details

In our implementation, we set topic number $T = 50$, and the prior hyperparameters $\gamma = 0.5$, $\tau = \sigma = 0.1$, $\rho = \alpha = 50/T$. $\beta_{s,z}$ is calculated using the set of seed words with $\lambda_1 = 0.1$ and $\lambda_2 = 0.8$.

As described in Section 3.3, we use a small set of seed words as our topic prior knowledge. Specifically, the seed words list contains 5 to 10 seed words for each of the five topics of interest[3] about *Hurricane impact, public utility, food, shelter, medical*, respectively. We choose these five topics based on the actual requirements of our project. However, it is important to note that the model is flexible and do not need to have seed words for every topic.

4.4 Experiment results

4.4.1 Quantitative evaluation

We first compare our model with the baseline models in terms of perplexity which is a widely used measurement of how well a probability model predicts a sample. The lower the perplexity, the better the model. We calculate the average perplexity (log-likelihood) using 1000 held-out documents which are randomly selected from the test data. The average test perplexity of each word is calculated as $\exp\{-\frac{1}{N}\sum_w \log p(w)\}$, where N is the total number of words in the held-out test documents. Table 1 shows the perplexity results for Hurricane Sandy dataset. Our model outperforms the baseline models. In particular, the perplexity of our model is 1122, which is 40 and 116 lower than that of JST and TS. The perplexity of LDST is 35 lower than that of LDST-w/oS, which indicates that the seed words can further improve the performance of our model.

LDA	ToT	JST	TS	LDST-w/oS	LDST
1320	1244	1162	1238	1157	1122

Table 1: Comparison of test perplexity per word

Following the same evaluation as in (Lin et al., 2012), we also present and discuss the experimental results of sentiment classification. The docu-

[2] https://developers.google.com/maps/

[3] Details available at https://goo.gl/ykbrXZ

ment sentiment is classified based on the probability of sentiment label given document, which can be approximated by $\hat{\chi}_u^{(a)}$ and $\hat{\chi}_{d_v}^v$. Similar to (Lin et al., 2012), we only consider the probability of positive and negative label given document, with the neutral label probability being ignored. We define that a document d is classified as a positive-sentiment document if its probability of positive sentiment label given document is greater than its probability of negative sentiment label given document, and vice versa. The ground truth of sentiment classification labels of tweets are set by using human annotation. Specifically, we randomly select 1000 documents from the dataset, and label each document as *positive*, *negative* or *neutral* manually. We measure the performance of our model using the tweets with *positive* or *negative* labels. The classification accuracies are summarization in Table 2. LDST significantly outperforms other methods on test data. This verifies the effectiveness of the proposed approach.

LDA	ToT	JST	TS	LDST-w/oS	LDST
0.574	0.592	0.622	0.597	0.635	0.646

Table 2: Sentiment classification accuracy.

4.4.2 Qualitative evaluation

We present the sentiments and topics discovered by LDST to see whether LDST captured meaningful semantics. We analyze the extracted topics under positive and negative sentiment labels. Due to the limited space, we only report *Hurricane impact* and *public utility* topics under positive and negative sentiments for New York and Florida states on Oct. 27, 2012 and Oct. 30, 2012. For each topic, we visualize it using the top 5 words which are most likely generated from the topic. As shown in Table 3, the extracted topics are quite informative and coherent. For example, the first topic (left) is closely related to *Hurricane impact*, and the other one (right) is related to *public utility*. The results show that our model can extract topic and sentiment simultaneously under different time and location. First, at the same period, people in different location had different topics of interest and sentiments. Taking the *public utility* topic as an example, on Oct. 30, people from Florida were concerned with post-disaster relief, while people from New York focused on the outage of the *public utility*. Second, for the people in the same location, they were interested in different topics, and had different sentiments towards the topics. For example, the people in Florida paid close attention to the damage of the *public utility* on Oct. 27, while they changed their attention to post-disaster relief of the *public utility* on Oct. 30.

	New York		Florida	
pos	pray response volunteer usa florida	island careful volunteer preparation state	florida prepare boat protect power	work service island customer system
neg	destroy homeless flood eastcoast damage	rising moves arrival shortages cause	sandy storm broken landfall disaster	power outage damage energy utilities
	New York		Florida	
pos	aid alert rescue york shelter	nation people delivered safety companies	volunteer claimed crew power relief	restore system resources good rebuild
neg	hirricane weatherd death halloween flood	power metro destroyed outage destructive	east criticize eastern election tourism	damage effect weeks millions delay

Table 3: Hurricane impact (left) and public utility (right) topics under positive and negative sentiment labels for New York and Florida states on Oct. 27, 2012 (above) and Oct. 30, 2012 (bottom).

5 Conclusion

In this paper, we propose a location-based dynamic sentiment-topic (LDST) model, which can jointly model sentiment, topic, temporal, and geolocation information.

Acknowledgement

Heng Ji's work was supported by U.S. DARPA LORELEI Program No. HR0011-15-C-0115. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Ghazaleh Beigi, Xia Hu, Ross Maciejewski, and Huan Liu. 2016. An overview of sentiment analysis in social media and its applications in disaster relief. In *Sentiment Analysis and Ontology Engineering*, pages 313–340. Springer.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *The Journal of machine Learning research*, 3:993–1022.

Davide Buscaldi and Irazú Hernandez-Farias. 2015. Sentiment analysis on microblogs for natural disasters management: a study on the 2014 genoa floodings. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1185–1188. ACM.

Cornelia Caragea, Anna Squicciarini, Sam Stehle, Kishore Neppalli, and Andrea Tapia. 2014. Mapping moods: geo-mapped sentiment analysis during hurricane sandy. In *Annual Conference for Information Systems for Crisis Response and Management*.

Mohamed Dermouche, Leila Kouas, Julien Velcin, and Sabine Loudcher. 2015. A joint model for topic-sentiment modeling from text. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 819–824. ACM.

Yulan He, Chenghua Lin, Wei Gao, and Kam-Fai Wong. 2012. Tracking sentiment and topic dynamics from social media. In *Sixth International AAAI Conference on Weblogs and Social Media*.

Alistair Kennedy and Diana Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational intelligence*, 22(2):110–125.

Yury Kryvasheyeu, Haohui Chen, Esteban Moro, Pascal Van Hentenryck, and Manuel Cebrian. 2015. Performance of social network sensors during hurricane sandy. *PLoS one*, 10(2):e0117288.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 375–384. ACM.

Chenghua Lin, Yulan He, Richard Everson, and Stefan Ruger. 2012. Weakly supervised joint sentiment-topic detection from text. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 1134–1145. IEEE.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Benjamin Mandel, Aron Culotta, John Boulahanis, Danielle Stark, Bonnie Lewis, and Jeremy Rodrigue. 2012. A demographic analysis of online sentiment during hurricane irene. In *Proceedings of the Second Workshop on Language in Social Media*, pages 27–36. Association for Computational Linguistics.

Subhabrata Mukherjee, Gaurab Basu, and Sachindra Joshi. 2014. Joint author sentiment topic model. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 370–378. SIAM.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 79–86. Association for Computational Linguistics.

Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2008. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577. ACM.

Gayane Shalunts, Gerhard Backfried, and Prinz Prinz. 2014. Sentiment analysis of german social media data for natural disasters. In *Annual Conference for Information Systems for Crisis Response and Management*.

Mark Steyvers, Padhraic Smyth, Michal Rosen-Zvi, and Thomas Griffiths. 2004. Probabilistic author-topic models for information discovery. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 306–315. ACM.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 417–424. Association for Computational Linguistics.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*, pages 235–243. Association for Computational Linguistics.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM.

Min Yang, Jincheng Mei, Fei Xu, Wenting Tu, and Ziyu Lu. 2016. Discovering author interest evolution in topic modeling. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 801–804. ACM.

Min Yang, Baolin Peng, Zheng Chen, Dingju Zhu, and Kam-Pui Chow. 2014a. A topic model for building fine-grained domain-specific emotion lexicon. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 421–426.

Min Yang, Wenting Tu, Ziyu Lu, Wenpeng Yin, and Kam-Pui Chow. 2015. Lcct: a semisupervised model for sentiment classification. In *The 2015 Annual Conference of the North American Chapter of the ACL*. Association for Computational Linguistics.

Min Yang, Dingju Zhu, Rashed Mustafa, and Kam-Pui Chow. 2014b. Learning domain-specific sentiment lexicon with supervised sentiment-aware lda. *The 21st Eruopean Conference on Artificial Intelligence*, pages 927–932.

Zhou Zhao, James Cheng, and Wilfred Ng. 2014. Truth discovery in data streams: A single-pass probabilistic approach. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1589–1598. ACM.

Arkaitz Zubiaga and Heng Ji. 2014. Tweet, but verify: epistemic study of information verification on twitter. *Social Network Analysis and Mining*, 4(1):163.

Refining Word Embeddings for Sentiment Analysis

Liang-Chih Yu[1,3], Jin Wang[2,3,4], K. Robert Lai[2,3] and Xuejie Zhang[4]
[1]Department of Information Management, Yuan Ze University, Taiwan
[2]Department of Computer Science & Engineering, Yuan Ze University, Taiwan
[3]Innovation Center for Big Data and Digital Convergence Yuan Ze University, Taiwan
[4]School of Information Science and Engineering, Yunnan University, Yunnan, P.R. China
Contact: lcyu@saturn.yzu.edu.tw

Abstract

Word embeddings that can capture semantic and syntactic information from contexts have been extensively used for various natural language processing tasks. However, existing methods for learning context-based word embeddings typically fail to capture sufficient sentiment information. This may result in words with similar vector representations having an opposite sentiment polarity (e.g., *good* and *bad*), thus degrading sentiment analysis performance. Therefore, this study proposes a word vector refinement model that can be applied to any pre-trained word vectors (e.g., Word2vec and GloVe). The refinement model is based on adjusting the vector representations of words such that they can be closer to both semantically and sentimentally similar words and further away from sentimentally dissimilar words. Experimental results show that the proposed method can improve conventional word embeddings and outperform previously proposed sentiment embeddings for both binary and fine-grained classification on Stanford Sentiment Treebank (SST).

1 Introduction

Word embeddings are a technique to learn continuous low-dimensional vector space representations of words by leveraging the contextual information from large corpora. Examples include C&W (Collobert and Weston, 2008; Collobert et al., 2011), Word2vec (Mikolov et al., 2013a; 2013b) and GloVe (Pennington et al., 2014). In addition to the contextual information, character-level subwords (Bojanowski et al., 2016) and semantic knowledge resources (Faruqui et al., 2015; Kiela et al., 2015) such as WordNet (Miller, 1995) are also useful information for learning word embeddings. These embeddings have been successfully used for various natural language processing tasks.

In general, existing word embeddings are semantically oriented. They can capture semantic and syntactic information from unlabeled data in an unsupervised manner but fail to capture sufficient sentiment information. This makes it difficult to directly apply existing word embeddings to sentiment analysis. Prior studies have reported that words with similar vector representations (similar contexts) may have opposite sentiment polarities, as in the example of *happy*-*sad* mentioned in (Mohammad et al., 2013) and *good*-*bad* in (Tang et al., 2016). Composing these word vectors may produce sentence vectors with similar vector representations but opposite sentiment polarities (e.g., a sentence containing *happy* and a sentence containing *sad* may have similar vector representations). Building on such ambiguous vectors will affect sentiment classification performance.

To enhance the performance of distinguishing words with similar vector representations but opposite sentiment polarities, recent studies have suggested learning sentiment embeddings from labeled data in a supervised manner (Maas et al., 2011; Labutov and Lipson, 2013; Lan et al., 2016; Ren et al., 2016; Tang et al., 2016). The common goal of these methods is to capture both semantic/syntactic and sentiment information such that sentimentally similar words have similar vector representations. They typically apply an objective function to optimize word vectors based on the sentiment polarity labels (e.g., positive and negative) given by the training instances. The use of such sentiment embeddings has improved the performance of binary sentiment classification.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 534–539
Copenhagen, Denmark, September 7–11, 2017.

This study adopts another strategy to obtain both semantic and sentiment word vectors. Instead of building a new word embedding model from labeled data, we propose a word vector refinement model to refine existing semantically oriented word vectors using sentiment lexicons. That is, the proposed model can be applied to the pre-trained vectors obtained by any word representation learning models (e.g., Word2vec and GloVe) as a post-processing step to adapt the pre-trained vectors to sentiment applications. The refinement model is based on adjusting the pre-trained vector of each affective word in a given sentiment lexicon such that it can be closer to a set of both semantically and sentimentally similar nearest neighbors (i.e., those with the same polarity) and further away from sentimentally dissimilar neighbors (i.e., those with an opposite polarity).

The proposed refinement model is evaluated by examining whether our refined embeddings can improve conventional word embeddings and outperform previously proposed sentiment embeddings. To this end, several deep neural network classifiers that performed well on the Stanford Sentiment Treebank (SST) (Socher et al., 2013) are selected, including convolutional neural networks (CNN) (Kim, 2014), deep averaging network (DAN) (Iyyer et al., 2015) and long-short term memory (LSTM) (Tai et al., 2015; Looks et al., 2017). The conventional word embeddings used in these classifiers are then replaced by our refined versions and previously proposed sentiment embeddings to re-run the classification for performance comparison. The SST is chosen because it can show the effect of using different word embeddings on fine-grained sentiment classification, whereas prior studies only reported binary classification results.

The rest of this paper is organized as follows. Section 2 describes the proposed word vector refinement model. Section 3 presents the evaluation results. Conclusions are drawn in Section 4.

2 Word Vector Refinement

The refinement procedure begins by giving a set of pre-trained word vectors and a sentiment lexicon annotated with real-valued sentiment scores. Our goal is to refine the pre-trained vectors of the affective words in the lexicon such that they can capture both semantic and sentiment information. To accomplish this goal, we first calculate the semantic similarity between each affective word (target word) and the other words in the lexicon based on the cosine distance of their pre-trained vectors, and then select top-*k* most similar words as the nearest neighbors. These semantically similar nearest neighbors are then re-ranked according to their sentiment scores provided by the lexicon such that the sentimentally similar neighbors can be ranked higher and dissimilar neighbors lower. Finally, the pre-trained vector of the target word is refined to be closer to its semantically and sentimentally similar nearest neighbors and further away from sentimentally dissimilar neighbors. The following subsections provide a detailed description of the nearest neighbor ranking and refinement model.

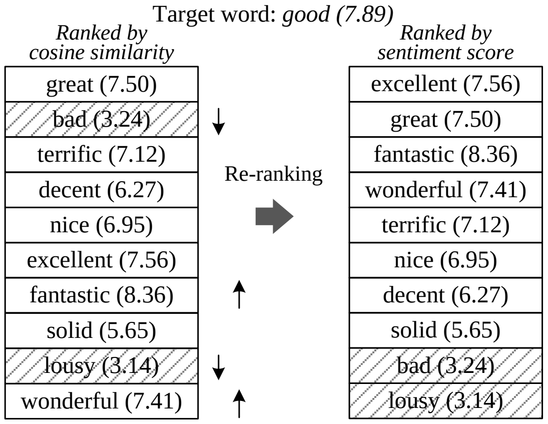


Figure 1: Example of nearest neighbor ranking.

2.1 Nearest Neighbor Ranking

The sentiment lexicon used in this study is the extended version of Affective Norms of English Words (E-ANEW) (Warriner et al., 2013). It contains 13,915 words and each word is associated with a real-valued score in [1, 9] for the dimensions of valence, arousal and dominance. The valence represents the degree of positive and negative sentiment, where values of 1, 5 and 9 respectively denote most negative, neutral and most positive sentiment. In Fig. 1, *good* has a valence score of 7.89, which is greater than 5, and thus can be considered positive. Conversely, *bad* has a valence score of 3.24 and is thus negative. In addition to the E-ANEW, other lexicons such as SentiWordNet (Esuli and Fabrizio, 2006), SoCal (Taboada et al., 2011), SentiStrength (Thelwall et al., 2012), Vader (Hutto et al., 2014), ANTUSD (Wang and Ku, 2016) and SCL-NMA (Kiritchenko and Mohammad, 2016) also provide real-valued sentiment intensity or strength scores like the valence scores.

For each target word to be refined, the top-k semantically similar nearest neighbors are first selected and ranked in descending order of their cosine similarities. In Fig. 1, the left ranked list shows the top 10 nearest neighbors for the target word *good*. The semantically ranked list is then sentimentally re-ranked based on the absolute difference of the valence scores between the target word and the words in the list. A smaller difference indicates that the word is more sentimentally similar to the target word, and thus will be ranked higher. As shown in the right ranked list in Fig. 1, the re-ranking step can rank the sentimentally similar neighbors higher and the dissimilar neighbors lower. In the refinement model, the higher ranked sentimentally similar neighbors will receive a higher weight to refine the pre-trained vector of the target word.

2.2 Refinement Model

Once the word list ranked by both cosine similarity and valence scores for each target word is obtained, its pre-trained vector will be refined to be (1) closer to its sentimentally similar neighbors, (2) further away from its dissimilar neighbors, and (3) not too far away from the original vector.

Let $V = \{v_1, v_2, \ldots, v_n\}$ be a set of the pre-trained vectors corresponding to the affective words in the sentiment lexicon. For each target to be refined, the refinement model iteratively minimizes the distance between the target word and its top-k nearest neighbors. The objective function $\Phi(V)$ can thus be defined as

$$\Phi(V) = \sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij}\,dist(v_i, v_j) \qquad (1)$$

where n denotes the total number of vectors in V to be refined, v_i denotes the vector of a target word, v_j denotes the vector of one of its nearest neighbors in the ranked list, $dist(v_i, v_j)$ denotes the distance between v_i and v_j, and w_{ij} denotes the weight of the target word's nearest neighbor, defined as the reciprocal rank of a ranked list. For example, *excellent* in Fig. 1 will receive a weight of 1, *great* will receive a weight of 1/2, and so on. A word ranked higher will receive a higher weight. This weight is used to control the movement direction of the target word towards to its nearest neighbors. That is, the target word will be moved closer to the higher-ranked sentimentally similar neighbors and further away from lower-ranked dissimilar neighbors, as shown in Fig. 2.

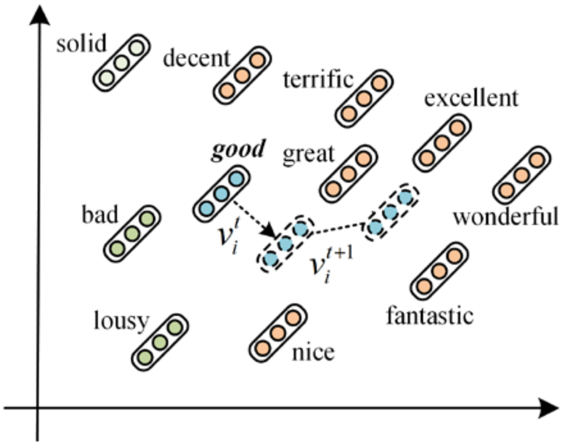


Figure 2: Conceptual diagram of word vector refinement.

To prevent too many words being moved to the same location and thereby producing too many similar vectors, we add a constraint to keep each pre-trained vector within a certain range from its original vector. The objective function is thus divided as two parts:

$$\arg\min \Phi(V) = \arg\min \sum_{i=1}^{n}\left[\alpha\,dist(v_i^{t+1}, v_i^{t}) + \beta\sum_{j=1}^{k} w_{ij}\,dist(v_i^{t+1}, v_j^{t})\right] \qquad (2)$$

where $dist(v_i^{t+1}, v_i^{t})$ denotes the distance between the vector of the target word in step t and t+1, i.e., the distance between the refined vector and its original vector. The later one represents the distance between the vector of the target word and that of its neighbors (similar to Eq. (1)). The parameters α and β together are used as a ratio to control how far the refined vector can be moved away from its original vector and toward its nearest neighbors. A greater ratio indicates a stronger constraint on keeping the refined vector closer to its original vector. For the extreme case of α=1 and β=0, the target word will not be moved (refined). As the ratio decreases, the constraint decreases accordingly and the refined vector can be moved closer to its nearest neighbors. The setting of α=0 and β=1 means that the constraint is disabled.

To facilitate the calculation of the partial derivative of $\Phi(V)$, $dist(v_i, v_j)$ in the above equations is measured by the squared Euclidean distance, defined as

$$dist(v_i, v_j) = \sum_{d=1}^{D} (v_i^d - v_j^d)^2 \qquad (3)$$

where D is the dimensionality of the word vectors. The global optimal solution of $\Phi(V)$ can be found by using an iterative update method. To do so, we solve the partial derivation of Eq. (2) in step t with respect to word vector v_i^t, and by setting $\frac{\partial \Phi(V)}{\partial v_i^t} = 0$ to obtain a new vector v_i^{t+1} in step t+1. The iterative update procedure is defined as

$$v_i^{t+1} = \frac{\gamma v_i^t + \beta \sum_{j=1}^{k} w_{ij} v_j^t}{\gamma + \beta \sum_{j=1}^{k} w_{ij}} \qquad (4)$$

Through the iterative procedure, the vector representation of each target word will be iteratively updated until the change of the location of the target word's vector is converged. The refinement process will be terminated when all target words are refined.

3 Experimental Results

This section evaluates the proposed refinement model, conventional word embeddings and previously proposed sentiment embeddings using several deep neural network models for binary and fine-grained sentiment classification.

Dataset. SST was adopted as the evaluation corpus (Socher et al., 2013). The binary classification subtask (positive and negative) contains 6920/872/1821 samples for the train/dev/test sets, while the fine-grained ordinal classification subtask (very negative, negative, neutral, positive, and very positive) contains 8544/1101/2210 samples of the train/dev/test sets.

Word Embeddings. The word embeddings used for comparison included two conventional word embeddings (GloVe and Word2vec), our refined versions (Re(GloVe) and Re(Word2vec)), and previously proposed sentiment embeddings (HyRank) (Tang et al., 2016). We used the same dimensionality of 300 for all word embeddings.

- GloVe and Word2vec: The respective GloVe and Word2vec (skip-gram) were pre-trained on Common Crawl 840B [1] and GoogleNews[2].
- Re(Glove) and Re(Word2vec): Both the pre-trained GloVe and Word2vec were refined using E-ANEW (Warriner et al., 2013). Each affective word was refined by its top k=10 nearest neighbors with parameters of α:β=0.1 (1:10) (see Eq. (2)).
- HyRank: It was trained using SST, NRC Sentiment140 and IMDB datasets. We compared this method because its code is publicly accessible[3].

Classifiers. The above word embeddings were used by CNN (Kim, 2014) [4], DAN (Iyyer et al., 2015)[5], , bi-directional LSTM (Bi-LSTM) (Tai et al., 2015)[6] and Tree-LSTM (Looks et al., 2017)[7] with default parameter values.

Comparative Results. Table 1 presents the evaluation results of using different word embeddings for different classifiers. For the pre-trained word embeddings, GloVe outperformed Word2vec for DAN, Bi-LSTM and Tree-LSTM, whereas Word2vec yielded better performance for CNN. After the proposed refinement model was applied, both the pre-trained Word2vec and GloVe were improved. The Re(Word2vec) and Re(GloVe) respectively improved Word2vec and GloVe by 1.7% and 1.5% averaged over all classifiers for binary classification, and both 1.6% for fine-grained classification. In addition, both Re(GloVe) and Re(Word2vec) outperformed the sentiment embeddings HyRank for all classifiers on both binary and fine-grained classification, indicating that the real-valued intensity scores used by the proposed refinement model are more effective than the binary polarity labels used by the previously proposed sentiment embedings.

The proposed method yielded better performance because it can remove semantically similar but sentimentally dissimilar nearest neighbors for the target words by refining their vector representations. To demonstrate the effect, we define a measure noise@k to calculate the percentage of top k nearest neighbors with an opposite polarity (i.e., noise) to each word in E-ANEW. For instance, in Fig. 1, the noise@10 for *good* is 20% because there are two words with an opposite polarity to *good* among its top 10 nearest neighbors. Table 2 shows the average noise@10 for different

[1] http://nlp.stanford.edu/projects/glove/

[2] https://code.google.com/archive/p/word2vec/

[3] http://ir.hit.edu.cn/~dytang/

[4] https://github.com/yoonkim/CNN_sentence

[5] https://github.com/miyyer/dan

[6] https://github.com/stanfordnlp/treelstm

[7] https://github.com/tensorflow/fold

Method	Fine-grained	Binary
DAN		
- Word2vec	46.2	84.5
- GloVe	46.9	85.7
- Re(Word2vec)	48.1	87.0
- Re(GloVe)	**48.3**	**87.3**
- HyRank	47.2	86.6
CNN		
- Word2vec	48.0	87.2
- GloVe	46.4	85.7
- Re(Word2vec)	**48.8**	**87.9**
- Re(GloVe)	47.7	87.5
- HyRank	47.3	87.6
Bi-LSTM		
- Word2vec	48.8	86.3
- GloVe	49.1	87.5
- Re(Word2vec)	49.6	88.2
- Re(GloVe)	**49.7**	**88.6**
- HyRank	49.0	87.3
Tree-LSTM		
- Word2vec	48.8	86.7
- GloVe	51.8	89.1
- Re(Word2vec)	50.1	88.3
- Re(GloVe)	**54.0**	**90.3**
- HyRank	49.2	88.2

Table 1: Accuracy of different classifiers with different word embeddings for binary and fine-grained classification.

Word Embeddings	Noise@10 (%)
Word2vec	24.3
GloVe	24.0
HyRank	18.5
Re(Word2vec)	14.4
Re(GloVe)	13.8

Table 2: Average percentages of noisy words in the top 10 nearest neighbors for different word embeddings.

word embeddings. For the two semantic-oriented word vectors, GloVe and Word2vec, on average around 24% of the top 10 nearest neighbors for each word are noisy words. After refinement, both Re(GloVe) and Re(Word2vec) can reduce noise@10 to around 14%. The HyRank also yielded better performance than both GloVe and Word2vec.

4 Conclusion

This study presents a word vector refinement model that requires no labeled corpus and can be applied to any pre-trained word vectors. The proposed method selects a set of semantically similar nearest neighbors and then ranks the sentimentally similar neighbors higher and dissimilar neighbors lower based on a sentiment lexicon. This ranked list can guide the refinement procedure to iteratively improve the word vector representations. Experiments on SST show that the proposed method yielded better performance than both conventional word embeddings and sentiment embeddings for both binary and fine-grained sentiment classification. In addition, the performances of various deep neural network models have also been improved. Future work will evaluate the proposed method on another datasets. More experiments will also be conducted to provide more in-depth analysis.

Acknowledgments

This work was supported by the Ministry of Science and Technology, Taiwan, ROC, under Grant No. MOST 105-2221-E-155-059-MY2 and MOST 105-2218-E-006-028. The authors would like to thank the anonymous reviewers and the area chairs for their constructive comments.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv: 1607.04606*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML-08)*, pages 160–167.

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

Andrea Esuli, and Fabrizio Sebastiani. 2006. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-06)*, pages 417-422.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard H. Hovy and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In

Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL/HLT-15), pages 1606-1615.

Clayton J. Hutto and Eric Gilbert. 2014. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of 8th International AAAI Conference on Weblogs and Social Media*, pages 216-225.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15)*, pages 1681-1691.

Douwe Kiela, Felix Hill and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP-15)*, pages 2044-2048.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*, pages 1746-1751.

Svetlana Kiritchenko and Saif M. Mohammad. 2016. The effect of negators, modals, and degree adverbs on sentiment composition. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis at NAACL-HLT 2016*, pages 43-52.

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-13)*, pages 489-493.

Man Lan, Zhihua Zhang, Yue Lu, and Ju Wu. 2016. Three convolutional neural network-based models for learning sentiment word vectors towards sentiment analysis. In *Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN-16)*, pages 3172-3179.

Moshe Looks, Marcello Herreshoff, DeLesley Hutchins, and Peter Norvig. 2017. Deep learning with dynamic computation graphs. In *Proceedings of the 5th International Conference on Learning Representations (ICLR-17)*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-11)*, pages 142-150.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Annual Conference on Advances in Neural Information Processing Systems (NIPS-13)*, pages 1-9.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013b. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR-2013)*, pages 1-12.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39-41.

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

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*, pages 1532-1543.

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Improving twitter sentiment classification using topic-enriched multi-prototype word embeddings. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 3038-3044.

Richard Socher, Alex Perelygin, and Jy Wu. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*, pages 1631-1642.

Maite Taboada, Julian Brooke, and Kimberly Voll. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15)*, pages 1556–1566.

Duyu Tang, Furu Wei, Bing Qin, Nan Yang, Ting Liu, and Ming Zhou. 2016. Sentiment embeddings with applications to sentiment analysis. *IEEE Trans. Knowledge abd Data Engineering*, 28(2):496-509.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2012. Sentiment strength detection for the social web. *Journal of the Association for Information Science and Technology*, 63(1):163-173.

Shih-Ming Wang and Lun-Wei Ku. 2016. ANTUSD: A large Chinese sentiment dictionary. In *Proc. of the 10th International Conference on Language Resources and Evaluation (LREC-16)*, pages 2697-2702.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 English lemmas. *Behavior Research Methods*, 45(4):1191-1207.

A Multilayer Perceptron based Ensemble Technique for Fine-grained Financial Sentiment Analysis

Md Shad Akhtar, Abhishek Kumar, Deepanway Ghosal
Asif Ekbal, Pushpak Bhattacharyya
Department of Computer Science and Engineering
Indian Institute of Technology Patna, India
{shad.pcs15,abhishek.ee14,deepanway.me14,asif,pb}@iitp.ac.in

Abstract

In this paper, we propose a novel method for combining deep learning and classical feature based models using a Multi-Layer Perceptron (MLP) network for financial sentiment analysis. We develop various deep learning models based on Convolutional Neural Network (CNN), Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). These are trained on top of pre-trained, autoencoder-based, financial word embeddings and lexicon features. An ensemble is constructed by combining these deep learning models and a classical supervised model based on Support Vector Regression (SVR). We evaluate our proposed technique on a benchmark dataset of SemEval-2017 shared task on financial sentiment analysis. The propose model shows impressive results on two datasets, i.e. microblogs and news headlines datasets. Comparisons show that our proposed model performs better than the existing state-of-the-art systems for the above two datasets by 2.0 and 4.1 cosine points, respectively.

1 Introduction

Microblog messages and news headlines are freely available on Internet in vast amount. Dynamic nature of these texts can be utilized effectively to analyze the shift in the stock prices of any company (Goonatilake and Herath, 2007). By keeping a track of microblog messages and news headlines for financial domain one can observe the trend in stock prices, which in turn, allows an individual to predict the future stock prices. An increase in positive opinions towards a particular company would indicate that the company is doing well and this would be reflected in the increase in company stock prices and vice-versa. Benefits of such analysis are two-fold: (i). an individual can take informed decision before buying/selling his/her share; and (ii). an organization can utilize this information to forecast its economic situation.

Sentiment prediction is a core component of an end-to-end stock market forecasting business model. Thus, an efficient sentiment analysis system is required for real-time analysis of financial text originating from the web. Sentiment analysis in financial domain offers more challenges (as compared to product reviews domains etc.) due to the presence of various financial and technical terms along with numerous statistics. Coarse-level sentiment analysis in financial texts usually ignores critical information towards a particular company, therefore making it unreliable for the stock prediction. In fine-grained sentiment analysis, we can emphasize on a given company without losing any critical information. For example, in the following tweet sentiment towards $APPL (APPLE Inc.) is *positive* while *negative* towards $FB (Facebook Inc.).

'$APPL going strong; $FB not so.'

In literature, many methods for sentiment analysis from financial news have been described. O'Hare et al. (2009) used word-based approach on financial blogs to train a sentiment classifier for automatically determining the sentiment towards companies and their stocks. Authors in (Schumaker and Chen, 2009) use the bag-of-words and named entities for predicting stock market. They successfully showed that the stock market behavior is based on the opinions. A fine-grained sentiment annotation scheme was incorporated by (de Kauter et al., 2015) for predicting the explicit

First three student authors have equally contributed to this work

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 540–546
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

and implicit sentiment in the financial text. An application of multiple regression model was developed by (Oliveira et al., 2013).

In this paper, we propose a novel Multi-Layer Perceptron (MLP) based ensemble technique for fine-grained sentiment analysis. It combines the outputs of four systems, one is feature-driven supervised model and the rest three are deep learning based.

We further propose to develop an enhanced word representation by learning through a stacked denoising autoencoder network (Vincent et al., 2010) using word embeddings of Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) models.

For evaluation purpose we use datasets of SemEval-2017 'Fine-Grained Sentiment Analysis on Financial Microblogs and News' shared task (Keith Cortis and Davis, 2017). The dataset comprises of financial short texts for two domains i.e. microblog messages and news headlines. Comparisons with the state-of-the-art models show that our system produces better performance.

The main contributions of the current work are summarized as follows: a) we effectively combine competing systems to work as a team via MLP based ensemble learning; b) develop an enhanced word representation by leveraging the syntactic and semantic richness of the two distributed word representation through a stacked denoising autoencoder; and c) build a state-of-the-art model for sentiment analysis in financial domain.

2 Proposed Methodology

We propose a Multi-Layer Perceptron based ensemble approach to leverage the goodness of various supervised systems. We develop three deep neural network architecture based models, *viz.* Convolution Neural Network (CNN) (Kim, 2014), Long Short Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) network (Cho et al., 2014)). The other model is based on Support Vector Regression (SVR) (Smola and Schölkopf, 2004) based feature-driven system.

The classical feature based system utilizes a diverse set of features (c.f. Section 2.D). On the other hand we train a CNN, a LSTM and a GRU network on top of distributed word representations. We utilize Word2Vec and GloVe word representation techniques to learn our financial word embeddings. Since Word2Vec and GloVe models capture syntactic and semantic relations among words using different techniques (Word2Vec: given a context, a word is predicted or vice-versa; GloVe: count-based model utilizing word co-occurrence matrix), some applications adapt well to Word2Vec while others perform well on GloVe model. We, therefore, attempt to leverage the richness of both the models by using a stacked denoising autoencoder.Finally, we combine predictions of these models using the MLP network in order to obtain the final sentiment scores. An overview of the proposed method is depicted in Figure 1.

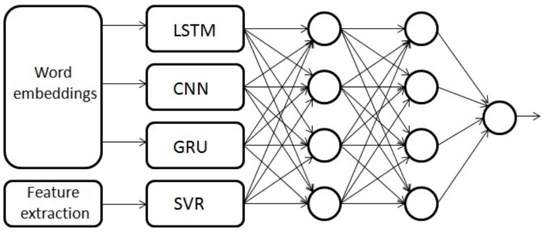


Figure 1: MLP based ensemble architecture.

A. Convolution Neural Network (CNN): Literature suggests that CNN architecture had been successfully applied for sentiment analysis at various level (Kim, 2014; Akhtar et al., 2016; Singhal and Bhattacharyya, 2016). Most of these works involve classification tasks, however, we adopt CNN architecture for solving the regression problem. Our proposed system employs a convolution layer followed by a max pool layer, 2 fully connected layers and an output layer. We use 100 different filters while sliding over 2, 3 and 4 words at a time. We employ all these filters in parallel.

B. Long Short Term Memory Network (LSTM): LSTMs (Hochreiter and Schmidhuber, 1997) are a special kind of recurrent neural network (RNN) capable of learning long-term dependencies by effectively handling the vanishing or exploding gradient problem. We use two LSTM layers on top of each other having 100 neurons in each. This was followed by 2 fully connected layers and an output layer.

C. Gated Recurrent Unit (GRU): GRUs (Cho et al., 2014) are also a special kind of RNN which can efficiently learn long-term dependencies. A key difference of GRU with LSTM is that, GRU's recurrent state is completely exposed at each time step in contrast to LSTM's recurrent state which controls its recurrent state. Thus, comparably GRUs have lesser parameters to learn and training is computationally efficient. We use two GRU layers on top of each other having 100 neurons in each. This was followed by 2 fully connected layers and an output layer.

D. Feature based Model (SVR): We extract and implement following set of features to train a Support Vector Regression (SVR) (Smola and Schölkopf, 2004) for predicting the sentiment score in the continuous range of -1 to +1.

- Word Tf-Idf: Term frequency-inverse document frequency (tf-idf) is a numerical statistic that is intended to reflect how important a word is to a document in a corpus. We consider tf-idf weighted counts of continuous sequences of n-grams (n=2,3,4,5) at a time.

- Lexicon Features: Sentiment lexicons are widely utilized resources in the field of sentiment analysis. Its application and effectiveness in sentiment prediction task had been widely studied. We employ two lexicons i.e. Bing Liu opinion lexicon (Ding et al., 2008) and MPQA (Wilson et al., 2005) subjectivity lexicon for news headlines domain. First we compile a comprehensive list of positive and negative words form these lexicons and then extract the following lexicon driven features.

Agreement Score : This score indicates the polarity of the sentence i.e. whether the sentence takes a polar or neutral stance. If the agreement score is 1 then it implies that the instance is of having either high positive or negative sentiment whereas, a 0 agreement score indicates a mixed or disharmony in the positive and negative sentiment implying the sentence is not polar (Rao and Srivastava, 2012).

$$A = 1 - \sqrt{1 - \left| \frac{T_{pos} - T_{neg}}{T_{pos} + T_{neg}} \right|}$$

Class score : Each text is assigned a class score indicating an overall sentiment value. The class score is -1, 0 or +1 depending on whether T_{pos} is less than, equal to or greater than T_{neg}. This helps in detecting the correct class of the sentence.

We also use four Twitter specific sentiment lexicons. These are NRC (Hashtag Context, Hashtag Sentiment, Sentiment140, Sentiment140 Context) lexicons (Kiritchenko et al., 2014; Mohammad et al., 2013) which associate a positive or negative score to a token. Following features are extracted for each of these: i) *positive*, *negative* and *net* count. ii) maximum of *positive* and *negative* scores. iii) sum of *positive*, *negative* and *net* scores.

- Vader Sentiment: Vader sentiment (Gilbert, 2014) score is a rule-based method that generates a compound sentiment score for each sentence between -1 (extreme negative) and +1 (extreme positive). It also produces ratio of positive, negative and neutral tokens in the sentence. We obtain score and ratio of each instance in the datasets and use as feature for training.

Network parameters for CNN, LSTM & GRU: In the fully connected layers we use 50 and 10 neurons , respectively for the two hidden layers. We use *Relu* activations (Glorot et al., 2011) for intermediate layers and *tanh* activation in the final layer. We employ 20% *Dropout* (Srivastava et al., 2014) in the fully connected layers as a measure of regularization and *Adam* optimizer (Kingma and Ba, 2014) for optimization.

E. MultiLayer Perceptron (MLP) based Ensemble: Ensemble of models improves the prediction accuracy by combining the outputs of all the individual models. It exploits the strengths of all the participating models. Some of these exiting ensemble techniques cover a wide variety of traditional approaches such as bagging, boosting, majority voting, weighted voting (Xiao et al., 2013; Remya and Ramya, 2014; Ekbal and Saha, 2011) etc. In recent times Particle Swarm Optimization based ensemble technique (Akhtar et al., 2017) has been proposed for aspect based sentiment analysis. However, our current work differs significantly *w.r.t.* the methodology that we adapt as well as the problem domain that we focus on.

In this work we propose a new ensemble technique based on MLP which learns on top of the predictions of candidate models. We use a small MLP network consisting of 2 hidden layers (4 neurons in each) and an output layer. We use *Relu* activations for hidden layers and *tanh* activation in the final layer. We employ 25% dropout in the intermediate layers and use Adam optimizer during backpropagation. The output of this network serves as the final prediction value.

We separately train and tune all the four models (Section 2.A - 2.D) for both the domains. Evaluation shows that results of these individual models are encouraging, and an effective combination of these through the proposed ensemble further increases the performance.

Word Embeddings: Distributed representation models such as Word2Vec and Glove are generally very effective in a multitude of natural language processing tasks. Quality of any word embedding directly depends upon two entities: a) in-domain corpus and b) size of the corpus. Pre-trained word embeddings of Word2Vec (PWE-W2V) and GloVe (PWE-GLV) serve general purpose rather than focusing on a specific domain. Since we are addressing the problem in financial text we train and use our own embedding for financial domain corpus (FWE-W2V & FWE-GLV). We collected 126,000 financial news articles from Google News having a total of 92 million tokens. Although the corpus size is not as large as pre-trained word embedding corpus, it works reasonably well (c.f Table 1).

We observe that Word2Vec and GloVe word embeddings are quite competitive. In some cases GloVe has the advantage while in others Word2Vec performs better. Therefore, we derive a new hybrid word embedding model using a stacked denoising autoencoders (DAWE). A denoising autoencoder (Vincent et al., 2010) is a neural network which is trained to reconstruct a clean repaired input from a noisy version of the input. Following (AP et al., 2014), we combine pretrained Word2Vec and GloVe representation of a word and fed it to the network in order to capture the richness of both representations. The input to the network is a combined 600 dimensional vector (300 W2V + 300 GLV) with statistically added *salt-and-pepper* noise.

In total we employ five different word embedding models for both the domains. Dimension of all these word embeddings are set to 300. While training our deep learning models, we keep the word embeddings dynamic so that they can be fine-tuned during the process.

3 Experiments, Results and Analysis

Dataset: We evaluate our proposed approach on the benchmark datasets of SemEval-2017 shared task 5 on 'fine-grained sentiment analysis on financial microblogs and news' (Keith Cortis and Davis, 2017). The two datasets consist of financial texts from Microblogs (Twitter and StockTwits) and News, respectively. There are 1,700 and 1,142 instances of microblog messages and news headlines in the training data. The test dataset comprises of 800 microblog messages and 491 news headlines.

Experiments: We use Python based libraries Keras and Scikit-learn for the implementation. Following the shared task guideline we use cosine similarity as the metric for evaluation. Cosine score represents the degree of agreement between predicted and actual values.

Table 1 shows evaluation of our various models. In microblog dataset we obtain the best cosine similarities of 0.724, 0.727, 0.721 and 0.765 for CNN, LSTM, GRU and feature based systems, respectively. Similarly, for news datasets we obtain the best cosine similarities of 0.722, 0.720, 0.721 and 0.760. It can be observed that results for all the models are numerically comparable, however, on a qualitative side they are quite contrasting in nature. Figure 2 shows the contrasting nature of different models for microblog messages. The predicted output of different models (i.e. CNN, LSTM, GRU and SVR) are often complimentary in nature. In some case, one model predicts correctly (or, closer to the gold output), while other models make incorrect predictions and the *vice-versa*. We also observe the same phenomena for news headline. Motivated by this contrasting behavior we choose to combine the predictions of these models.

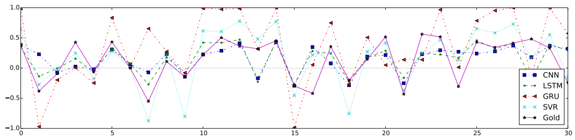


Figure 2: Contrasting nature of different models *w.r.t.* to the gold standard values; Sample size:30.

Consequently, we construct an ensemble by taking the best performing deep learning (CNN, LSTM and GRU each) and classical feature based (SVR) models using a MLP network. The proposed ensemble yields enhanced cosine scores of 0.797 and 0.786 for the microblog messages and news headline, respectively.

For comparison we choose two state-of-the-art systems (ECNU (Lan et al., 2017) and Fortia-FBK (Mansar et al., 2017)) which were the best performing systems at SemEval-2017 shared task 5. ECNU reported to have obtained cosine similarity of 0.777 in microblog as compared to 0.797 cosine similarity of our proposed system, whereas, for news headlines Fortia-FBK reported cosine similarity of 0.745. ECNU employed several regressors on top of optimized feature set obtained through *hill climbing* algorithm. For the final prediction, authors averaged the predictions of different regressors. Fortia-FBK trained a CNN with the assistance of sentiment lexicons for predicting the sentiment score. It should be noted that both the systems (ECNU and Fortia-FBK) utilize different approaches to achieve the stated cosine similarities on two different domains. The proposed approach of ECNU does not perform well for the news headlines, and Fortia-FBK reported results only for the news headlines. Our proposed system performs better compared to these existing systems for both the domains.This shows that our system is more robust and generic in nature in predicting the sentiment scores. We also perform statistical significance test on the obtained results and observe that the performance gain is significant with p-value = 0.00747. Table 2 depicts the comparative results on the test datasets.

3.1 Error Analysis

We also perform qualitative error analysis on the obtained results and observe that the proposed system faces problems in the following scenarios:

• Presence of implicit negation makes it a non-trivial task for the proposed system to predict the sentiment and intensity correctly. For the example given below, the overall negative sentiment is altered because of the presence of the word ‘*breaks*’.

Example : ***Tesco*** *breaks its downward slide by cutting sales decline in half*

Predicted: -0.694 **Actual:** 0.172

• The proposed system often confuses when the input text contains a question (?) mark.

Example : *Is* ***$FB*** *a BUY? Topeka Capital Markets thinks so*

Predicted: 0.363 **Actual:** -0.373.

4 Conclusion

In this paper, we have presented an ensemble network of deep learning and classical feature driven models. Evaluation on the benchmark datasets show that it performs remarkably well to identify bullish and bearish sentiments associated with companies in financial texts. We have implemented a variety of linguistic and semantic features for our analysis of the noisy text in Tweets and news headlines. Our proposed approach achieves state-of-the-art performance with the increments of 2.0 and 4.1 points over the existing systems for the tasks of sentiment prediction of financial microblog and news data. In future, we would like to extend our work by creating an end to end stock prediction system where the system would predict the future stock prices based on the sentiment score and stock value of the company.

	Models	**Microblog**	**News**
Convolutional neural network (CNN)			
C1	PWE-W2V CNN	0.705	**0.722**
C2	PWE-GLV CNN	0.721	0.697
C3	FWE-W2V CNN	0.710	0.705
C4	FWE-GLV CNN	**0.724**	0.714
C5	DAWE CNN	0.697	0.698
Long short term memory (LSTM)			
L1	PWE-W2V LSTM	0.700	0.704
L2	PWE-GLV LSTM	0.715	0.683
L3	FWE-W2V LSTM	**0.727**	0.680
L4	FWE-GLV LSTM	0.717	0.691
L5	DAWE LSTM	0.722	**0.720**
Gated Recurrent Unit (GRU)			
G1	PWE-W2V GRU	0.689	**0.721**
G2	PWE-GLV GRU	0.713	0.705
G3	FWE-W2V GRU	0.715	0.687
G4	FWE-GLV GRU	0.713	0.703
G5	DAWE GRU	**0.721**	0.712
Feature - SVR			
F1	Tf-idf + Lexicon + Vader	0.752	0.749
F2	Tf-idf + Lexicon + Vader + PWE-W2V	0.740	0.731
F3	Tf-idf + Lexicon + Vader + PWE-GLV	0.758	0.745
F4	Tf-idf + Lexicon + Vader + FWE-W2V	0.709	0.702
F5	Tf-idf + Lexicon + Vader + FWE-GLV	0.732	0.725
F6	Tf-idf + Lexicon + Vader + DAWE	**0.765**	**0.760**
Ensemble			
E1	C4 + L3 + G5 + F6 (MLP)	**0.797**	0.765
E2	C1 + L5 + G1 + F6 (MLP)	0.779	**0.786**

Table 1: Cosine similarity score of various models on test dataset.

Systems	**Microblogs**	**News**
ECNU (Lan et al., 2017)	0.777	0.710
Fortia-FBK (Mansar et al., 2017)	-	0.745
Proposed System	**0.797**	**0.786**

Table 2: Comparison with the state-of-the-art systems.

References

Md Shad Akhtar, Deepak Gupta, Asif Ekbal, and Pushpak Bhattacharyya. 2017. Feature selection and ensemble construction: A two-step method for aspect based sentiment analysis. *Knowledge-Based Systems* 125:116 – 135.

Md Shad Akhtar, Ayush Kumar, Asif Ekbal, and Pushpak Bhattacharyya. 2016. A Hybrid Deep Learning Architecture for Sentiment Analysis. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016): Technical Papers, December 11-16, 2016*. Osaka, Japan, pages 482–493.

Sarath Chandar AP, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An Autoencoder Approach to Learning Bilingual Word Representations. In *Advances in Neural Information Processing Systems*. pages 1853–1861.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *CoRR* abs/1409.1259. http://arxiv.org/abs/1409.1259.

Marjan Van de Kauter, Diane Breesch, and Vronique Hoste. 2015. Fine-grained analysis of explicit and implicit sentiment in financial news articles. *Expert Systems with Applications* 42(11):4999 – 5010.

Xiaowen Ding, Bing Liu, and Philip S Yu. 2008. A Holistic Lexicon-Based Approach to Opinion Mining. In *Proceedings of the 2008 international conference on web search and data mining*. ACM, pages 231–240.

Asif Ekbal and Sriparna Saha. 2011. Weighted Vote-Based Classifier Ensemble for Named Entity Recognition: A Genetic Algorithm-Based Approach. *ACM Transactions on Asian Language Information Processing* 10:9:1–9:37.

CJ Hutto Eric Gilbert. 2014. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In Geoffrey J. Gordon and David B. Dunson, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*. Journal of Machine Learning Research - Workshop and Conference Proceedings, volume 15, pages 315–323.

Rohitha Goonatilake and Susantha Herath. 2007. The Volatility of the Stock Market and News. *International Research Journal of Finance and Economics* 3(11):53–65.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation* 9(8):1735–1780.

Andre Freitas Tobias Daudert Manuela Huerlimann Manel Zarrouk Keith Cortis and Brian Davis. 2017. SemEval-2017 Task 5: Fine-Grained Sentiment Analysis on Financial Microblogs and News. In *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*. ACL, Vancouver, Canada, pages 519–535.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980. http://dblp.uni-trier.de/db/journals/corr/corr1412.html.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M. Mohammad. 2014. Sentiment Analysis of Short Informal Texts. *Journal of Artificial Intelligence Research (JAIR)* 50:723–762.

Man Lan, Mengxiao Jiang, and Yuanbin Wu. 2017. ECNU at SemEval-2017 Task 5: An Ensemble of Regression Algorithms with Effective Features for Fine-grained Sentiment Analysis in Financial Domain. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. ACL, Vancouver, Canada, pages 888–893.

Youness Mansar, Lorenzo Gatti, Sira Ferradans, Marco Guerini, and Jacopo Staiano. 2017. Fortia-FBK at SemEval-2017 Task 5: Bullish or Bearish? Inferring Sentiment towards Brands from Financial News Headlines. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. ACL, Vancouver, Canada, pages 817–822.

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

Saif Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Atlanta, Georgia, USA, pages 321–327.

Neil O'Hare, Michael Davy, Adam Bermingham, Paul Ferguson, Páraic Sheridan, Cathal Gurrin, and Alan F. Smeaton. 2009. Topic-dependent Sentiment Analysis of Financial Blogs. In *Proceedings of the 1st International CIKM Workshop on Topic-sentiment Analysis for Mass Opinion*. ACM, New York, NY, USA, TSA '09, pages 9–16.

Nuno Oliveira, Paulo Cortez, and Nelson Areal. 2013. On the Predictability of Stock Market Behavior Using StockTwits Sentiment and Posting Volume. In *EPIA*. Springer, volume 8154 of *Lecture Notes in Computer Science*, pages 355–365.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-2014)*. ACL, Doha, Qatar, pages 1532–1543.

Tushar Rao and Saket Srivastava. 2012. Analyzing stock market movements using twitter sentiment analysis. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*. IEEE Computer Society, pages 119–123.

K. R. Remya and J. S. Ramya. 2014. Using weighted majority voting classifier combination for relation classification in biomedical texts. In *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. pages 1205–1209.

Robert P. Schumaker and Hsinchun Chen. 2009. Textual Analysis of Stock Market Prediction Using Breaking Financial News: The AZFin Text System. *ACM Transactions on Information Systems* 27(2).

Prerana Singhal and Pushpak Bhattacharyya. 2016. Borrow a Little from your Rich Cousin: Using Embeddings and Polarities of English Words for Multilingual Sentiment Classification. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016): Technical Papers, December 11-16, 2016*. Osaka, Japan, pages 3053–3062.

Alex J. Smola and Bernhard Schölkopf. 2004. A Tutorial on Support Vector Regression. *Statistics and Computing* 14(3):199–222.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research* 11(Dec):3371–3408.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-level Sentiment Analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 347–354.

Tong Xiao, Jingbo Zhu, and Tongran Liu. 2013. Bagging and Boosting Statistical Machine Translation Systems. *Artif. Intell.* 195:496–527.

Sentiment Intensity Ranking among Adjectives Using Sentiment Bearing Word Embeddings

Raksha Sharma, Arpan Somani, Lakshya Kumar, Pushpak Bhattacharyya
Dept. of Computer Science and Engineering
IIT Bombay, India
raksha,somani,lakshya,pb@cse.iitb.ac.in

Abstract

Identification of intensity ordering among polar (positive or negative) words which have the same semantics can lead to a fine-grained sentiment analysis. For example, *master, seasoned* and *familiar* point to different intensity levels, though they all convey the same meaning (semantics), *i.e.*, *expertise: having a good knowledge of*. In this paper, we propose a semi-supervised technique that uses sentiment bearing word embeddings to produce a continuous ranking among adjectives that share common semantics. Our system demonstrates a strong Spearman's rank correlation of 0.83 with the gold standard ranking. We show that sentiment bearing word embeddings facilitate a more accurate intensity ranking system than other standard word embeddings (word2vec and GloVe). Word2vec is the state-of-the-art for intensity ordering task.

1 Introduction

The interchangeable use of semantically similar words stimulates sentiment intensity variation among sentences. To understand the phenomenon, let us consider the following example:

1. (a) We were *pleased* by the beauty of the island. (Positively low intense)
   (b) We were *delighted* by the beauty of the island. (Positively medium intense)
   (c) We were *exhilarated* by the beauty of the island. (Positively high intense)

Pleased, Exhilarated and *delighted* are the positive words bearing the same semantics, *i.e.*, *directing the emotion*, but their use intensifies the positive sentiment in the sentences 1(a), 1(b) and 1(c) respectively. Identification of intensity ranking among the words which have the same semantics can facilitate such a fine-grained sentiment analysis as exemplified in 1(a), 1(b) and 1(c).[1]

In this paper, we present a semi-supervised approach to establish a continuous intensity ranking among polar adjectives having the same semantics. Essentially, our approach is a refinement of the work done by Sharma et al., (2015). They also built a system that generates intensity of the words that bear the same semantics; however, their system considers only three discrete intensity levels, *viz., low*, *medium* and *high*. The important feature of our approach is that it uses Sentiment Specific Word Embeddings (SSWE). SSWE are an enhancement to the normal word embeddings with respect to the sentiment analysis task (Tang et al., 2014). SSWE capture syntactic, semantic as well as sentiment information, unlike normal word embeddings (word2vec and GloVe), which capture only syntactic and semantic information.

Our Contribution: We propose an approach that generates a continuous (finer) intensity ranking among polar words, which belong to the same semantic category. In addition, we show that SSWE produce a significantly better intensity ranking scale than word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), which do not capture sentiment information of the words.

The remaining paper is organized as follows. Section 2 describes the previous work related to intensity ranking task. Section 3 describes the different word embeddings explored in the paper. Section 4 gives the description of the data and the resources. Section 5 provides details of the gold

[1] Words which have the different semantic concepts cannot be used interchangeably. For example, *master (expertise)* and *delighted (directing the emotion)* cannot be a replacement of each other. Hence, our understanding is that a comparison between words belonging to different semantic categories does not give any meaningful information.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 547–552
Copenhagen, Denmark, September 7–11, 2017.

standard data. Section 6 elaborates the proposed intensity ranking approach. Section 7 presents the results and experimental setup. Section 8 concludes the paper.

2 Related Work

Sentiment analysis on adjectives has been extensively explored in NLP literature. However, most of the work addressed the problem of finding polarity orientation of the adjectives (Hatzivassiloglou and McKeown, 1997; Wiebe, 2000; Wilson et al., 2005; Fahrni and Klenner, 2008; Dragut et al., 2010; Taboada and Grieve, 2004; Baccianella et al., 2010).

The task of ranking polar words has received much attention recently due to the vital role of word's intensity in several real world applications. Most of the literature on intensity ranking consists of manual approaches or corpus-based approaches. Affective Norms (Warriner et al., 2013), SentiStrength (Thelwall et al., 2010), SoCAL (Taboada et al., 2011), and LABMT (Dodds et al., 2011), Best–Worst Scaling (Kiritchenko and Mohammad, 2016) are a few such publicly available sentiment intensity lexicons which are manually created.

Corpus-based approaches follow the assumption that the polarity of a new word can be inferred from the corpus (Hatzivassiloglou and McKeown, 1993; Kiritchenko et al., 2014; De Melo and Bansal, 2013). Corpus-based approaches require a huge amount of data, otherwise they suffer from the data sparsity problem. None of the these approaches considers the concept of semantics of adjectives, assuming one single intensity scale for all adjectives. Ruppenhofer et al., (2014) made the first attempt in this direction. They provided ordering among polar adjectives that bear the same semantics using a corpus-based approach. On the contrary, Sharma et al., (2015) used publicly available embeddings (word2vec) of words to assign intensity to words. Learning of word embeddings does not require annotated (labeled) corpus.

The embeddings used in our work are sentiment specific word embeddings. Integration of sentiment information of a word with syntactic and semantic information makes our approach more accurate for fine-grained sentiment intensity ranking of words.

3 Word Embeddings

In recent years, several models have been proposed to learn word embeddings from large corpora. In this paper, we have explored three types of word embeddings, *viz.*, word2vec (Mikolov et al., 2013), Glove (Pennington et al., 2014) and SSWE (Tang et al., 2014). The word embeddings given by word2vec are the distributed vector representation of the words that capture both the syntactic and semantic relationships among words. The Global Vector model, referred as GloVe, combines word2vec with ideas drawn from matrix factorization methods, such as LSA (Deerwester et al., 1990). Word2vec and GloVe model the syntactic context of the words but ignore their sentiment information. For sentiment analysis task, this is problematic as these word embeddings map words with similar syntactic context but opposite polarity, such as *love* and *hate* closer to each other in the vector space.

Sentiment Specific Word Embeddings (SSWE) encode sentiment information along with the syntactic and semantic information in word vector space. These word embeddings are able to separate the words like *love* and *hate* to the opposite ends of the spectrum. Tang et al., (2014) proposed a method to learn sentiment specific word embeddings from tweets with *emoticons* as distant-supervised corpora without any manual annotation. Specifically, they developed three neural networks to effectively incorporate the supervision from sentiment polarity of text in their loss functions.

4 Data and Resources

In this work, we have used the 52 polar semantic categories from the FrameNet data.[2] FrameNet-1.5 (Baker et al., 1998) is a lexical resource which groups words based on their semantics.[3] We also used a star-rated movie review corpus of 5006 files (Pang and Lee, 2005) to extract the pivot for each semantic category.[4] Though our approach uses a corpus, its use is limited to identification of pivot. Intensity ranking of other words of the semantic category is derived by exploiting the cosine-

[2] Sharma et al., (2015) also have presented their results using the same 52 polar semantic categories of the FrameNet data.

[3] Available at: `https://framenet.icsi.berkeley.edu/fndrupal/about`.

[4] Available at: `http://www.cs.cornell.edu/people/pabo/movie-review-data/`.

similarity between word embeddings of the pivot and the other words of the semantic category. For all three types of word embeddings, we have used precomputed 300 dimensional vectors of words.[5][6]

5 Gold Standard Data Preparation

The objective of our work is to obtain a continuous ranking among words having the same semantics as per FrameNet data. We asked 5 annotators[7] to rank words in each semantic category on a scale of -50 to $+50$. Here, -50 represents the most negatively intense point and $+50$ represents the most positively intense point on the scale. 0 represents a neutral (neither positive nor negative) point on the scale. It is hard to get any neutral word in the data as we have used only polar semantic categories of the FrameNet. The final ranking scale in a category is obtained by averaging the score assigned by all 5 annotators. For example, for a word, if annotator-1 gave ranking r1, annotator-2 gave ranking r2, annotator-3 gave ranking r3, annotator-4 gave ranking r4 and annotator-5 gave ranking r5, then final ranking is ((r1+r2+r3+r4+r5)/5).

To check the agreement among 5 annotators, we computed Fleiss' kappa. It is a statistical measure of inter-rater reliability. Fleiss' kappa is chosen over Scott's pi and Cohen's kappa, because these measures work for two raters, whereas Fleiss' kappa works for any number of raters giving categorical ratings to a fixed number of items (Fleiss, 1971). We obtained a Fleiss' kappa score of **0.64** by dividing words of the semantic category into six levels (high-positive, medium-positive, low-positive, low-negative, medium-negative, high-negative).

6 Approach: Derive Intensity Ordering Among Words

Our approach establishes a continuous intensity ranking among words based on the following hypotheses:

Hypothesis-1 The classic *semantic bleaching theory* states that a word which has fewer number of senses (possibly one) tends to have a higher intensity in comparison to words having more senses.

Hypothesis-2 Semantically similar words that have fewer number of senses exhibit higher cosine-similarity with each other in comparison to words having many senses. Essentially, fewer number of senses cause fewer number of context words or vice versa.

Considering hypothesis-1 and 2 as a base, Sharma et al., (2015) claimed that the word embeddings (context vectors) of high intensity words depict higher cosine-similarity with each other than with low or medium intensity words. However, they used word embeddings which capture only syntactic and semantic similarity among words (Mikolov et al., 2013). Our approach uses SSWE, which integrate sentiment information with the normal word embeddings. Use of SSWE in place of normal word embeddings provides a more accurate cosine-similarity scores, which in turn leads to a more accurate continuous intensity scale. Section 6.1 describes how a high intensity word (pivot) for each semantic category is extracted from an intensity annotated corpus. Section 6.2 presents the algorithm that assigns intensity ordering to words of a semantic category using the pivot (high intensity) word.

6.1 Pivot Selection Method

An amalgamation of χ^2 test and Weighted Normalized Polarity Intensity (WNPI) formula extracts a high intensity word as pivot for each semantic category from the 5 star-rated review corpus. χ^2 test assures that no biased word should be selected as the pivot (Oakes and Farrow, 2007).[8] By biased word we mean that a word which has very few occurrences in the corpus, but these occurrences are in the high star-rated reviews. For example, in our corpus, the word *lame* occurs only 3 times in the corpus, and these occurrences happen to be in 1-star (negatively high intense) reviews only. In addition, χ^2 test derives polarity orientation of the pivot from the corpus as it associates a class (positive or negative) label with the word (Sharma and Bhattacharyya, 2013).

The WNPI formula assigns a intensity score to

[5] In this work, we have opted for precomputed word embeddings, because they are trained on sufficiently large corpora and widely tested for NLP applications.

[6] Embeddings are available here: word2vec (trained on news corpus of 320M words): https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit, GloVe (trained on Wikipedia 2014): http://nlp.stanford.edu/projects/glove/, SSWE (trained on 91M tweets): http://ir.hit.edu.cn/~dytang/.

[7] Two of the annotators are professional linguists of English language and other three are post graduate students.

[8] The details about the *goodness of fit chi*^{2} test: http://stattrek.com/chi-square-test/goodness-of-fit.aspx?Tutorial=AP.

words based on their frequency count in different star ratings. It is defined based on the concept that a high intensity word would occur more frequently in high star-rated reviews, for example, *outstanding* would occur more frequently in 5-star or 4-star reviews in comparison to 1,2,3-star reviews. In the WNPI formula (Algorithm 1), the value of i ranges from 1 to 5, here star rating is used as intensity of the review. The algorithm extracts two pivots for each category, one positive pivot for positive words and one negative pivot for negative words. For the sake of simplicity, we have used the term 'pivot' only in the Algorithm 1. For a positive word '5-star' is treated as 'i=5' (highest positive intensity) and for a negative word '1-star' is treated as 'i=5' (highest negative intensity) in the WNPI formula. A word which gets the highest score by the χ^2 test and the WNPI formula is set as positive (or negative) pivot.

6.2 Algorithm

Algorithm 1 illustrates the sequence of steps carried out to obtain the intensity ordering of words within a semantic category. c_p^w and c_n^w are the counts of a word w in the positive and negative documents respectively. μ^w is an average of c_p^w and c_n^w. To obtain the values of c_p^w and c_n^w, we divided the 5 star-rated review corpus in two equal parts as the positive corpus and the negative corpus. C_i is the count of a word in i intensity documents. Polarity of the words other than the pivot words is inferred by computing the cosine-similarity between SSWE of other words with the SSWE of the pivot word. Since SSWE have sentiment information, a positive pivot gives *positive* cosine-similarity with the positive words and *negative* cosine-similarity with the negative words.[9] Cosine-similarity order between SSWE of the pivot and other words establishes intensity ranking among words of a semantic category.

7 Results and Experimental Setup

To evaluate the efficacy of our SSWE-based approach over word2vec-based system (state-of-the-art) (Sharma et al., 2015) and GloVe-based system, we compute rank correlation and Macro-F1 between the intensity ranking produced by the embeddings and the gold standard intensity ranking.

Algorithm 1: Generating an Intensity ordering of words within a semantic category

```
Input: Set of words within a semantic
       category $W_{sc}$ ;
       Intensity ($i$) annotated corpus $C$ ;
       Pre-trained Sentiment embeddings
       $SSWE$.
Output: Ranking of words based on intensity.
for each word $w_i \in W_{sc}$ do
    $\chi^2(w_i) = ((c_p^w - \mu^w)^2 + (c_n^w - \mu^w)^2)/\mu^w$
    $WNPI(w_i) = \frac{\sum_{i=1}^{5} i*C_i}{5*\sum_{i=1}^{5} C_i}$
    Store in dictionary
      $(w_i, \chi^2(w_i), WNPI(w_i))$
Select word from the dictionary with the
  highest $\chi^2$ and WNPI score as pivot.
for each word $w_i$ in $W_{sc}$ do
    Calculate Cosine-Similarity between
    $(SSWE_{(w_i)}, SSWE_{(Pivot)})$
Words arranged in increasing order of their
  cosine-similarity is the Intensity Ordering.
```

7.1 Rank Correlation

Table 1 shows the average rank correlation coefficient obtained for 52 polar semantic categories of the FrameNet data using Spearman's ρ and Kendall's τ. Spearman's ρ measures the degree of association between the two rankings. Kendall's τ finds the number of concordant and discordant pairs in the rankings to measure association. We observed that SSWE-based system results in a significantly better ρ and τ as per t-test.

Embeddings	**Spearman's ρ**	**Kendall's τ**
Glove	0.525	0.425
Word2vec	0.71	0.565
SSWE	**0.82**	**0.70**

Table 1: Spearman's ρ and Kendall's τ rank correlations.

7.2 F1 Measure

In order to compare our work with the state-of-the-art (Sharma et al., 2015), the intensity ordering of words within a semantic category is divided into 3 levels, i.e, *low*, *medium* and *high* for both the positive and negative words respectively. In order to create three levels, we placed 2 break points in

[9] Sharma et al., (Sharma et al., 2015) used Bing Liu's lexicon in their approach to identify polarity orientation of words. The use of SSWE in our approach helped us to remove the need of a sentiment lexicon to identify polarity of words.

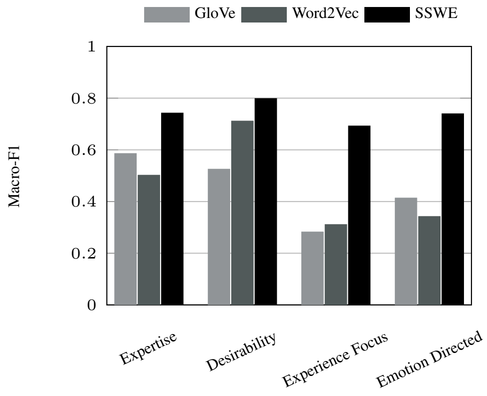


Figure 1: Individual Macro-F1 score for the 4 semantic categories.

the intensity ordering sequence where consecutive similarity scores differ the most.[10] Comparison of Macro-F1 scores for 4 different categories is shown in Figure 1. SSWE outperforms word2vec and GloVe by a big margin in all 4 cases. In addition, we obtain an average Macro-F1 score of 74.32% with SSWE, 54.38% with word2vec and 45.10% with GloVe for the 52 semantic categories.

7.3 Error Analysis

In a few semantic categories of the FrameNet data, words are not confined to any one sentiment and to say that one kind of sentiment has a higher intensity than the other is difficult at times. For example, it is difficult to compare *sadness* and *embarrassment* relatively in terms of intensity, whereas both the words belong to the same semantic category, that is, *emotion directed* as per FrameNet data. In addition, annotators mutually agreed on the fact that when there are limited number of words then it is easier and logical to scale them. More separation based on the finer semantic property within the existing semantic category of the FrameNet data may bring on improvement in the performance of automatic intensity ranking systems.

8 Conclusion

In this paper, we have given a technique that uses Sentiment Specific Word Embeddings (SSWE) to produce a fine-grained intensity ordering among polar words which bear the same semantics. In addition, the use of sentiment embeddings reduces the need of sentiment lexicon for identification of polarity orientation of words. Results show that SSWE are significantly better than word2vec and GloVe, which do not capture sentiment information of words for intensity ranking task. Sentiment intensity information of words can be used in various NLP applications, for example, star-rating prediction, normalization of over-expressed or under-expressed texts, *etc*.

Acknowledgments

We heartily thank English linguists Rajita Shukla and Jaya Saraswati from CFILT Lab, IIT Bombay for giving their valuable contribution in the gold standard data creation.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Gerard De Melo and Mohit Bansal. 2013. Good, great, excellent: Global inference of semantic intensities. *Transactions of the Association for Computational Linguistics*, 1:279–290.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Peter Sheridan Dodds, Kameron Decker Harris, Isabel M Kloumann, Catherine A Bliss, and Christopher M Danforth. 2011. Temporal patterns of happiness and information in a global social network: Hedonometrics and twitter. *PloS one*, 6(12):e26752.

Eduard C Dragut, Clement Yu, Prasad Sistla, and Weiyi Meng. 2010. Construction of a sentimental word dictionary. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1761–1764. ACM.

Angela Fahrni and Manfred Klenner. 2008. Old wine or warm beer: Target-specific sentiment analysis of adjectives. In *Proc. of the Symposium on Affective Language in Human and Machine, AISB*, pages 60–63.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

[10] The same break point convention is followed by Sharma et al., (2015) to assign intensity levels to words.

Vasileios Hatzivassiloglou and Kathleen R McKeown. 1993. Towards the automatic identification of adjectival scales: Clustering adjectives according to meaning. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 172–182. Association for Computational Linguistics.

Vasileios Hatzivassiloglou and Kathleen R McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics*, pages 174–181. Association for Computational Linguistics.

Svetlana Kiritchenko and Saif M. Mohammad. 2016. Capturing reliable fine-grained sentiment associations by crowdsourcing and best–worst scaling. In *Proceedings of The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, San Diego, California.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Michael P Oakes and Malcolm Farrow. 2007. Use of the chi-squared test to examine vocabulary differences in english language corpora representing seven different countries. *Literary and Linguistic Computing*, 22(1):85–99.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

Josef Ruppenhofer, Michael Wiegand, and Jasper Brandes. 2014. Comparing methods for deriving intensity scores for adjectives. *EACL 2014*, 117.

Raksha Sharma and Pushpak Bhattacharyya. 2013. Detecting domain dedicated polar words. In *IJCNLP*, pages 661–666.

Raksha Sharma, Mohit Gupta, Astha Agarwal, and Pushpak Bhattacharyya. 2015. Adjective intensity and sentiment analysis. *EMNLP2015, Lisbon, Portugal*.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.

Maite Taboada and Jack Grieve. 2004. Analyzing appraisal automatically. In *Proceedings of AAAI Spring Symposium on Exploring Attitude and Affect in Text (AAAI Technical Re# port SS# 04# 07), Stanford University, CA, pp. 158q161. AAAI Press*.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565, Baltimore, Maryland. Association for Computational Linguistics.

Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior research methods*, 45(4):1191–1207.

Janyce Wiebe. 2000. Learning subjective adjectives from corpora. In *AAAI/IAAI*, pages 735–740.

Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Opinionfinder: A system for subjectivity analysis. In *Proceedings of hlt/emnlp on interactive demonstrations*, pages 34–35. Association for Computational Linguistics.

Sentiment Lexicon Expansion Based on Neural PU Learning, Double Dictionary Lookup, and Polarity Association

Yasheng Wang[1], **Yang Zhang**[1], **Bing Liu**[2]
[1]Noah's Ark Lab, Huawei Technologies
[1]{wangyasheng, zhangyang86}@huawei.com
[2]Department of Computer Science, University of Illinois at Chicago, USA
[2]liub@uic.com

Abstract

Although many sentiment lexicons in different languages exist, most are not comprehensive. In a recent sentiment analysis application, we used a large Chinese sentiment lexicon and found that it missed a large number of sentiment words used in social media. This prompted us to make a new attempt to study sentiment lexicon expansion. This paper first formulates the problem as a *PU learning* problem. It then proposes a new PU learning method suitable for the problem based on a neural network. The results are further enhanced with a new dictionary lookup technique and a novel polarity classification algorithm. Experimental results show that the proposed approach greatly outperforms baseline methods.

1 Introduction

Sentiment lexicons contain words (such as *good*, *beautiful*, *bad*, and *awful*) that convey positive or negative sentiments. They are instrumental for many sentiment analysis applications. So far many algorithms have been proposed to generate such lexicons (Liu, 2012). These algorithms are either dictionary-based or corpus-based. In the dictionary-based approach, one exploits synonym and antonym relations in the dictionary to bootstrap a given seed set of sentiment words (Hu and Liu, 2004; Kim and Hovy, 2004; Kamps et al., 2004), or learns a classifier to classify the gloss of each word in the dictionary (Esuli and Sebastiani, 2005). The corpus-based approach uses various linguistic rules or patterns (Hatzivassiloglou and McKeown, 1997; Kanayama and Nasukawa, 2006; Qiu et al., 2011; Tang et al., 2014). We will discuss these and other existing methods in the related work section.

Despite many existing studies, the problem is far from being solved. In a recent application, we used a popular Chinese sentiment lexicon for sentiment classification of Weibo posts (similar to Twitter), and found that it missed a large number of sentiment words. As the lexicon was compiled using formal text such as news, it misses a large number of sentiment words used in social media. Due to the informal nature, many "low class" words are used in social media but seldom used in formal media. New words are also created constantly. Note that new words in Chinese are easier to create from individual characters than in other languages. Thus many of these words are not in the dictionary. All these prompted us to make a new attempt to study *sentiment lexicon expansion*.

In this paper, we solve the problem in two steps: (1) identify sentiment words from a given corpus, and (2) classify their polarity. We formulate Step 1 as a *PU learning* problem (*learning from positive unlabeled examples*). To our knowledge, this is the first such formulation. This is important because it gives us a formal model to tackle the problem. PU learning is stated as follows (Liu et al., 2002): given a set P of examples of a particular class (we also use P to denote the class) and a set U of unlabeled examples which contains hidden instances from both classes P and *not-P* (called N), we want to build a classifier using P and U to classify the data in U or future test data into the two classes, i.e., P and N (or *not-P*). In our case, P is the existing sentiment lexicon, and U is a set of candidate words from a social media corpus. We identify those words in U that are also sentiment words.

A typical PU learning algorithm works by first identifying a small set of *reliable* N class examples (*RN*) from the unlabeled set U and then running a supervised learning method (e.g., SVM) it-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 553–563
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

eratively to add more and more data to the *RN* set to finally build a classifier (Liu, 2011).

In this work, we first adapt a popular such approach to an *augmented multilayer perceptron* (AMP) method and use it to replace SVM, and show that using SVM as the learning method is inferior to using AMP. However, we can do much better. We then propose a new PU learning method, called SE-AMP (*Spy-based Elimination of P class instances using AMP*), which can better exploit the specific nature of our problem. SE-AMP goes in the opposite direction to the existing approach. It starts by treating U as the class N (*not-P*) data, and then runs the AMP learning method on P and U iteratively to gradually remove potential P class instances from U to purify U so that as iterations progress, fewer and fewer P class instances are still in U. We detail the method in Sec. 3.3. Note that SE-AMP is general and not limited to our task of sentiment lexicon expansion.

We also propose a new method based on dictionary lookup, called *Double dictionary Lookup* (DL), to enhance the results from the proposed PU learning method. The DL method is also in the framework of PU learning. Our final proposed method for Step 1 is called SE-AMP-DL.

For polarity classification (Step 2, after sentiment words are found), we propose a novel method that is based on polarity association of individual (Chinese) characters in each word.

In summary, this paper has several innovations:

1. It formulates Step 1 of sentiment lexicon expansion as a PU learning problem. To the best of our knowledge, this is the first such formulation.

2. It proposes a new neural learning method AMP and shows that AMP outperforms the traditional SVM based PU learning approach.

3. It further proposes a new and general PU learning strategy that works in the opposite direction to the popular existing approach to suit our task.

4. It also proposes a double dictionary lookup technique to improve the result further.

5. It proposes a novel polarity classification method to classify the polarity of each word.

Experimental results show that the proposed approach makes considerable improvement over existing baseline methods.

2 Related Work

There are two main approaches for sentiment lexicon generation (Liu, 2012): the *dictionary-based approach* and the *corpus-based approach*. Under the dictionary-based approach, one method is to use synonym and antonym relations and WordNet graph in the dictionary to bootstrap a set of given seed sentiment words. There are numerous variations of and enhancements to this approach (Hu and Liu, 2004; Valitutti et al., 2004; Kim and Hovy, 2004; Takamura et al., 2007; Andreevskaia and Bergler, 2006; Kaji and Kitsuregawa, 2007; Blair-Goldensohn et al., 2008; Cambria et al., 2016; Rao and Ravichandran, 2009; Perez-Rosas et al., 2012). For example, (Valitutti et al., 2004; Kim and Hovy, 2004) tried to remove error words and assign a sentiment strength to each word. Mohammad et al. (2009) exploited many antonym-generating affix patterns, Kamps et al. (2004) used a WordNet distance, and Hassan and Radev (2010) used a Markov random walk model over a word relatedness graph. Dragut et al. (2010) used a set of inference rules to determine word sentiment polarity through a deductive process, and Schneider and Dragut (2015) employed a linear programming approach. Another method is to build a supervised sentiment classifier to classify the gloss text of each word in the dictionary (Esuli and Sebastiani, 2005, 2006). Xu et al. (2010a) integrated both dictionaries and corpora to find emotion words based on label-propagation. Perez-Rosas et al. (2012) also worked on cross lingual lexicon construction.

In the corpus-based approach, one key idea is to exploit some linguistic conventions on connectives such as AND and OR Hatzivassiloglou and McKeown (1997). For example, in the sentence “This car is beautiful and spacious,” if “beautiful” is known to be positive, it can be inferred that “spacious” is also positive. Kanayama and Nasukawa (2006) extended the idea to the sentence level by exploiting adversative expressions such as “but” and “however.” Qiu et al. (2011) proposed a double propagation (DP) method that uses both sentiment and target relation and various connectives to extract sentiment words. (Wang and Wang, 2008) did similar works. Huang et al. (2014) detected new sentiment words using lexical patterns. Wilson et al. (2005), Ding et al. (2008), Choi and Cardie (2008) and Zhang and Liu (2011) studied contextual sentiments at the phrase or expression level. We do not study contextual sentiments.

Another idea for the corpus-based approach is to use word co-occurrences. Turney (2002) computed the Pointwise Mutual Information (PMI) between the target word and seed words to decide its sentiment polarity. This method was extended in (Mohammad et al., 2013; Yang et al., 2014). (Hamilton et al., 2016; Xu et al., 2010b) constructed domain lexicons using lexical graphs.

Recent works also exploited neural networks and word embedding, and treated lexicon generation as a classification problem like us. Tang et al. (2014) expanded a sentiment lexicon using a softmax classifier with the proposed sentiment-specific embedding. Vo and Zhang (2016) obtained sentiment attribute scores of each word through a neural network model to predict tweets emoticons. Bravo-Marquez et al. (2015) classified words using manual features and emoticon-annotated tweets. However, all these works require different kinds of labeled data. We do not rely on emoticons or other forms of annotations.

The problem of adapting a general lexicon to a domain specific one was studied in (Choi and Cardie, 2009; Jijkoun et al., 2010; Du et al., 2010). Feng et al. (2011) also generated a connotation lexicon. These are clearly different from our work.

3 The Proposed Approach

As mentioned in Sec. 1, our problem is solved in two steps: (1) identify sentiment words and (2) classify their polarity. In the following, we first introduce a traditional PU learning method using SVM (Sec. 3.1), and the augmented multilayer perceptron (AMP) method (Sec. 3.2) to set the background for the proposed technique. The proposed PU learning algorithm is presented in Sec. 3.3 for performing the task of step 1, which uses AMP. After that, a dictionary based method called Double Dictionary Lookup (Sec. 3.4) is presented to further improve the result of the first step. The proposed polarity classification method for the second step is discussed in (Sec. 3.5).

3.1 Traditional PU Learning

PU learning has been investigated by several researchers (Liu et al., 2002; Denis et al., 2002; Li and Liu, 2003; Yu et al., 2002; Elkan and Noto, 2008; Hsieh et al., 2015). A popular approach follows a two-stage strategy: (i) identifying a set of reliable N class instances RN from the unlabeled set U; and (ii) building a classifier using P (P class) and RN (N class) and $Q = U - RN$ (unlabeled) by applying a learning algorithm (e.g., SVM) iteratively.

To understand the difference between the proposed algorithm and the above two-stage approach, we give more details to an existing algorithm (Liu, 2011). In the first stage, a *Spy* technique is used to identify the set of reliable N class instances (or examples) RN from U. It works as follows: 10% of P class instances (in our cases, they are words) is first randomly selected as a *spy* set S and put in the unlabeled set U. Then SVM is run using the set $P - S$ as the P class training data and $U \cup S$ as the N class training data. After training, the resulting classifier assigns a probability to each instance in S to decide a probability threshold th. Instances in U that has a lower probability than th are selected as RN. As suggested in (Liu, 2011), th is set to the probability that separates the last 15% instances in S. We also experimented with 10% and 20%, but the results are similar.

In the second stage, we run SVM iteratively. In each iteration, a classifier trained using P and RN is used to classify the instances in $Q = U - RN$. Those instances assigned the N class in Q are removed and added to RN. Iterations stop when no instance in Q is classified to the N class.

3.2 Augmented Multilayer Perceptron

We now present the proposed AMP (*Augmented Multilayer Perceptron*) method, which we will use to replace SVM in PU learning as AMP produces better results. AMP has three layers (Figure 1). The first layer takes the *word vector* of each word as input with an output of 50 dimensions. The second layer takes the output of the first layer as input to produce an output of 2 dimensions. Both the first and second layers use the RELU activation function. The 2 dimension output of the second layer concatenates with 5 POS features (see Sec. 4.1.3) to form a 7 dimension feature vector as input of the third layer, which is the final layer with the activation function of Sigmoid.

We note that instead of putting POS features in the first layer, we first reduce the dimension of word vector from 200 to 2 with two layers, then compose a vector with POS features as the input to the third layer. This enables POS features to play a more important role. This method gives better results than combining the word vector and POS

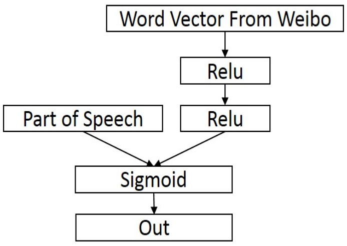


Figure 1: Aumented Multilayer Perceptron

tag features as the input in the first layer. Our experimental results also show that using AMP to replace SVM above in PU learning produces much better results.

3.3 Proposed New PU Learning Algorithm

We now present the proposed new PU learning strategy, which has only one stage that runs a supervised learning algorithm iteratively using P and U by treating U as the N class data. This strategy is more suitable for our task as we will explain in Sec. 4.1.4. The general idea is to remove words from U that are likely to belong to the P class until some stopping criterion is satisfied. The proposed algorithm is called SE-AMP (*Spy-based Elimination of P class instances using AMP*), which is given in Algorithm 1. Note that in the algorithm, we use $+1$ to denote class P (line 1) and -1 to denote class N (line 5). We don't use SVM any more as AMP performs much better. We now detail the working of the algorithm.

The algorithm still uses *Spies* and also works iteratively, but in a very different way. It first randomly sample a small proportion γ of words from set P to form the spy set S (line 2), which is added to the current U set to form U_s (line 4). The U set is updated in each iteration. An AMP classifier is trained using set P (with the class label $+1$) and set U_s (regarded as class N with the class label -1) (line 6). The resulting classifier or model M is used to score or assign a probability to each instance in U and in S (line 7).

Now we come to the crucial steps of the proposed algorithm. It tries to remove likely P class instances from U. U is essentially regarded as an noisy N class data, i.e., it has a lot of errors (which are hidden P class instances). Thus this step effectively aims to purify the N class set. In line 8, the algorithm determines a threshold θ to remove some likely P class instances from U. We will discuss the function for determining θ below.

Based on the probability threshold θ, the algorithm removes those instances in U with greater probability than θ (lines 9-13) and those instances in the spy set S (line 14-18).

The algorithm stops when the stopping criteria is met (lines 19 and 20); otherwise, it goes to the next iteration with the updated U and S. We determine the threshold δ using a validation set (Sec. 4.1.1).

Determine θ: In this new algorithm, each iteration removes instances that are likely to be of P class from the unlabeled set U. One simple way is to remove the top $k\%$ of U based on the classifier result of each iteration. However, this method is undesirable because we cannot control the probabilities of the eliminated instances. We propose to use Gaussian fitting to determine the threshold θ.

After each iteration, every spy word $w_i \in S$ is assigned a probability x_i $(= Pr(P|w_i))$ to be in class P by the classifier M, a Gaussian fitting is done on these probabilities. Parameters of Gaussian distribution are obtained using Maximum Likelihood Estimate (MLE) as follows: $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$, and $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$. We set the threshold $\theta = \mu + \sigma$. Thus those words have probabilities higher than θ are considered very likely to belong to the P class. This threshold is very conservative and only allows those very likely P class instances to be removed from U.

We will discus why the proposed SE-AMP is better than S-AMP in Sec. 4.1.4 after we see the experiment results. Note that S-AMP uses the traditional PU-learning strategy discussed in Sec. 3.1 but it replaces SVM with AMP.

3.4 Double Dictionary Lookup

To improve PU learning of SE-AMP further, we propose to employ a dictionary. Using a dictionary is natural because human beings always use dictionaries to understand a word. The proposed *Double Lookup* (DL) technique is given in Algorithm 2. Why *double lookup* is needed will be clear shortly.

The algorithm works as follows. Let the set of given sentiment lexicon be P, the given dictionary be D, and the set of candidate words be U (in this case, it is the test set). For each candidate word $w \in U$, we first look w up in the dictionary D (lines 1-2). If w can be found in D (line 2), we use a lexicon-based sentiment classifier (C) (see below) to classify the gloss or explanation note (G_w) of w (lines 3-4). The function *classify* returns a set of sentiment words O_1 from G_w (line 4), which are also in P. If O_1 is not empty, it means that G_w

Algorithm 1 SE-AMP(P, U)

```
1:  Every instance in $P$ is assigned the class label $+1$
2:  $S = Sample(P, \gamma)$;   // $\gamma = 0.1$ in this experiment; instances in $S$ are spies
3:  **loop**
4:      $U_s = U \cup S$
5:      Every instance $u$ in $U_s$ is assigned the class label $-1$   // $-1$ denotes the $N$ class
6:      $M$ = AMP($P$, $U_s$)   // Build a binary AMP classifier $M$
7:      score($M, U, S$)   // Score each instance *i* in $U$ and $S$ using $M$ to give each *i* a probability score
8:      $\theta$ = DetermineThreshold($S$)   // decide a probability threshold $\theta$ using $S$;
9:      **for** each instance $u \in U$ **do**
10:         **if** its probability $Pr(+1|u) > \theta$ **then**
11:             $U = U - \{u\}$
12:         **end if**
13:     **end for**
14:     **for** each instance $s \in S$ **do**
15:         **if** its probability $Pr(+1|s) > \theta$ **then**
16:             $S = S - \{s\}$
17:         **end if**
18:     **end for**
19:     **if** $|S| \leq \delta(\gamma|P|)$ **then**   // Stopping criterion; $\gamma|P|$ is actually the original size of $S$ in line 2
20:         exit-loop
21:     **end if**
22: **end loop**
```

is classified to the sentiment class. If it is empty, it is classified to the non-sentiment class.

This one dictionary lookup is not safe to determine whether word w is a sentiment word or not because some sentiment words in the lexicon P are noisy and don't have clear sentiments. This gives us too low precision. That is why we perform the second dictionary lookup, which is on the words in O_1 to ensure that at least one word in O_1 is very likely to be a true sentiment word because a noisy sentiment word in P is unlikely to be explained by another word in P. But a true sentiment word in P is very likely to be explained by another sentiment word in P.

Line 7 looks up each word $o \in O_1$ in D and finds its gloss or explanation note G_o. G_o is then classified in line 8, which returns a list of sentiment words O_2 from G_o, also in P. If O_2 is not empty, meaning that G_o is a sentiment sentence (line 9), we return w as a sentiment word (line 10).

Lexicon-based sentiment classifier (*C*): C is a binary classifier with two classes *sentiment* or *non-sentiment*. Given a sentence s (e.g., the explanation note of a word in the dictionary), it simply finds sentiment words in s that are also in the given sentiment lexicon P. If some sentiment words are found, it returns them in a set indicating the sentence s is a sentiment sentence. Although simple, this works quite well because the explanation note of a word in a dictionary is usually quite simple.

Integrating SE-AMP and DL: Our final proposed method (SE-AMP-DL) combines SE-AMP and DL. As we will see that DL has high precision but low recall because most of the new words cannot be found in the dictionary, we use the results of DL to correct the results of the SE-AMP algorithm. Words that are classified as belong to the N class by SE-AMP are moved to the P class if the DL method believes them to be sentiment words.

3.5 Polarity Classification

After sentiment words are discovered, this step determines the polarity (positive or negative) of each discovered sentiment word. We propose a new classification method exploiting the fact that new Chinese words are created with 2 or more Chinese characters. The meaning of a Chinese word is closely related to the meaning of each individual character of the word. So the polarity of a Chinese word is strongly related to the polarity of the characters that form the word, which is the motivation of the new classification method. We use the polarity association of the characters in each word to predict the polarity of the word based on super-

Algorithm 2 DL(P, D, U, C)

```
for each candidate word $w \in U$ do
    if $w$ can be found in $D$ then    // First dictionary lookup; $D$ is the dictionary
        Let $G_w$ be the gloss or explanation of word $w$ in $D$
        $O_1$ = classify($G_w$, $C$)    // $O_1$ is the set of sentiment words in $G_w$ and $O_1 \subseteq P$ (lexicon)
        if $O_1 \neq \emptyset$ then    // $G_w$ is classified to the sentiment class
            for each word $o \in O_1 (\subseteq P)$ do
                Let $G_o$ be the gloss or explanation of word $o$ in $D$    // Second dictionary lookup
                $O_2$ = classify($G_o$, $C$)    // $O_2 \subseteq P$ (lexicon) and $C$ is the sentiment classifier
                if $O_2 \neq \emptyset$ then    // $G_o$ is classified as a sentiment sentence
                    $w$ is a new sentiment word
                end if
            end for
        end if
    end if
end for
```

vised learning. We call the method CPA (*Character Polarity Association*). This method is very useful for languages whose words are constructed by characters such as Chinese and Japanese.

Feature Vector: For a character, we calculate the percentages of it appearing in the positive words and negative words in the existing lexicon P to form a 2-dimensional polarity vector. For example, the polarity vector (0.9, 0.1) means that 90% of the words in the existing lexicon that contains the character are positive and the other 10% are negative. Thus, for a word with 2 characters, which covers most cases in Chinese, a 4-dimensional vector is formed. For those words with more than 2 characters, we choose 2 characters with the strongest polarity to form a 4-dimensional vector.

Classifier Building: Using all positive and negative words in the existing lexicon P as the training sample, each word represented as a vector of four features, we apply a Naive Bayesian Classifier to build a polarity classifier. For testing, a word is represented in the same way. If a test word contains charters that don't exist in the lexicon, we give each character (0.5, 0.5) as the polarity vector.

4 Experimental Evaluation

We now evaluate the proposed technique SE-AMP-DL to expand an existing Chinese sentiment lexicon, DUTIR (Dalian University of Technology, Information Retrieve Lab) Affective Lexicon Ontology (Xu et al., 2008). DUTIR lexicon is perhaps the largest Chinese sentiment lexicon with 27466 words. Although large, since it was built based on formal text such as news, essays, and novels, it does not contain many sentiment words often used in social media as we will see later. We will also see that many new sentiment words that we discovered are not even in an authoritative Chinese language dictionary. Thus, compiling a comprehensive sentiment lexicon is needed. Below, we first evaluate sentiment words discovery (Step 1) and then polarity classification (Step 2).

4.1 Sentiment Words Discovery

4.1.1 Data and Parameter Settings

We use a large Chinese Weibo corpus (Chinese version of Twitter) from (Wang et al., 2013) for our lexicon expansion, which has about 4.4 million pairs of post and response messages. Although it was originally used to study natural language conversations, it is quite suitable for our purpose as online conversations are sentiment rich.

Word embedding: We first used the Stanford Chinese word segmenter to split sentences into sequences of words (the POS-tag of each word is also obtained in the process). For word embedding, we used word2vec (Mikolov et al., 2013). Each word vector has 200 dimensions.

P **set,** U **set, validation set, and test set**: We randomly sampled 200K messages, and used all 54303 words contained in them as our experiment dataset. Out of the 54303 words (stopwords have been removed), 4957 of them also appear in the DUTIR lexicon and are thus sentiment words. The remaining 49346 words are unlabeled.

Validation set: The validation set consists of randomly selected 300 words from the 4957 sentiment words and 700 words from the 46742 unlabeled words. Although the 700 words are unlabeled, they are treated as N class words.

Test set: 1000 words are randomly sampled from 48646 ($= 49346 - 700$) unlabeled words as the test set, which is labeled manually. Two native speakers labeled the 1000 test words. The Kappa agreement score is 0.695. For any word with disagreement, the annotators discussed to come to a final decision. The annotated test set has 199 sentiment words, 78 positive and 121 negative words.

P set and U set: The remaining 4657 ($= 4957 - 300$ sentiment words serve as the P set and the remaining 45042 ($= 46742 - 700 - 1000$) words serve as the U set for learning.

Parameter settings: As indicated earlier, 10% (γ) of the P class examples are randomly selected as spies S (465). The probability threshold θ is set using the Gaussian fitting (Sec. 3.3). The iteration stopping criterion of SE-AMP (Sec. 3.3) is decided using the validation set ($\delta = 30\%$).

Evaluation measures: We use the classic $precision$, $recall$, and F score for evaluation.

4.1.2 Compared Systems

We compare the following seven (7) techniques:

DP: The Double Propagation (DP) Method in (Qiu et al., 2011). This method uses dependency patterns for extraction.

PMI: The classic PMI method (Turney, 2002) using the full Weibo corpus and 100 positive and 100 negative words in the DUTIR sentiment lexicon as the reference words. These words appear most frequently in the Weibo corpus. In (Turney, 2002), only 1 positive and 1 negative reference words are used. We also tried to use 1, 50, 150, 200, and all words in the positive and negative classes as reference words, respectively. However, they give poorer results. Since the PMI method can only determine the polarity, but cannot decide whether a test word is a sentiment word or not. We make that decision by using the mean score the PMI method of all positive words in the lexicon as the positive threshold and the mean score of all negative words as the negative threshold.

S-SVM: The traditional PU learning method described in Sec. 3.1 using SVM.

S-AMP: Same as S-SVM, but SVM is replaced with AMP.

SE-AMP: The proposed PU learning method without DL.

DL: Only the double dictionary lookup method, using the Contemporary Chinese Dictionary.

SE-AMP-DL: This is our final proposed method, which combines SE-AMP and DL.

We could not compare with recent approaches Tang et al. (2014); Vo and Zhang (2016); Bravo-Marquez et al. (2015) as they all require some supervised information on the data. Our method is unsupervised except the use of the lexicon. These methods were also evaluated indirectly based on the social media post sentiment or emotion classification results. None reported precision, recall, or F score of the discovered sentiment words.

We do not compare with a dictionary-based approach because most of the test words are not even in the dictionary. Only 87 of 199 sentiment words in the test set can be found in the Contemporary Chinese dictionary. Note that in Chinese, one can form a word using characters fairly easily.

4.1.3 Features

For both SVM and AMP, the feature vector for each word is the word vector and POS tags. POS tags are divided into 5 classes (noun, verb, adjective, adverb, others) and form a 5 dimension binary vector (e.g. $[1, 0, 0, 0, 0]$ for noun). In the SVM approach, POS tags are concatenated to the word embedding features to form a 205 dimension feature vector (5 POS tags and 200 word embedding features). We used the RBF kernel, which gives the best result as compared to other kernels.

4.1.4 Experiment Results

We now present and discuss the results. The syntactic pattern based DP approach performed very poorly because social media posts are brief and the use of patterns to link sentiment words are quite infrequent. Thus, we will not include its results. Below, we first compare PMI, S-SVM and S-AMP, and then the results of the proposed PU learning method SE-AMP (without using DL). The results of incorporating DL are discussed last.

Existing Approach - PMI vs. S-SVM vs. S-AMP: S-SVM and S-AMP use the traditional PU learning approach described in Sec. 3.1 for model building. 10% of P class examples are sampled as spies to produce the RN set.

Table 1 shows the results of S-SVM and S-AMP iterations. We observe that the AMP version performs much better than the SVM version. The first three iterations improve the results. But after that, the results deteriorate for both systems.

Thus the table only shows the first few iterations since the results keep deteriorating after iteration 2. The best results are obtained by S-AMP, which is 0.548 in F score. S-SVM's best F score is only 0.509, which is poorer. We also see that the precision and recall of S-AMP are both consistently better than those of S-SVM.

We note that these two algorithms cannot catch the best results when they stop following the algorithm in (Liu, 2011). We did not use the validation set here to find the best stopping criterion because even their best results are poorer than those of SE-AMP. PMI does similarly to S-AMP.

Proposed Approach - SE-AMP: Table 2 shows the results of the proposed PU learning approach (SE-AMP). As noted above, the iteration stopping criterion δ is determined using the validation set. Iteration 0 means the classifier uses all unlabeled examples in U as the N set. Compared with iteration 0 of the traditional strategy (S-AMP), the F score of SE-AMP improves slightly (from 52.0% to 54.8%), but both are low. This is because the reliable N set RN for the traditional approach is too small (not representative of all N class examples) while for the proposed approach, the N set has too many hidden P class instances. The traditional PU learning tries to include more and more likely N examples iteratively to move to the P direction while the proposed approach doing the opposite, eliminating likely P instances from the unlabeled set U. As the table shows, the results of SE-AMP gets better and better after each iteration (it stops when the stopping condition is met). Precision, recall and F score all improve consistently, which result in improvements of 12.0%, 8.5% and 9.6% respectively. Compared with the best result of S-AMP, the precision of SE-AMP improves from 55.4% to 67.4%, recall improves from 51.8% to 59.3% and the F score improves from 53.5% to 63.1%.

We now explain why SE-AMP is better than S-AMP. We believe that the main reason is the high level of noise in P, i.e., many words in P don't have clear sentiments. The traditional PU learning (S-AMP) tries to add classified N class instances into the RN set in each iteration. This works fine for the first few iterations but then goes wrong because the noise in P resulted in a lot of hidden P instances added into the RN set. Then the results deteriorate as more and more wrong instances are added as iterations progress. In contrast, the proposed SE-AMP removes likely P instances (including those noisy ones) from the U set to obtain a purer and purer N set. Due to the very conservative setting of the θ parameter (see Sec. 3.3), the number of words removed from U in each iteration is small, so is the number of spy words removed from S as we can see in Table 2. Thus, U becomes purer and purer slowly, and the validation set helps find a good stopping iteration.

	Iteration	Precision	Recall	F-score
S-SVM	0	46.1	41.7	43.8
	1	49.7	45.7	47.6
	2	52.7	49.2	50.9
	3	48.9	46.2	47.5
S-AMP	0	52.8	51.3	52.0
	1	54.4	52.8	53.6
	2	56.4	53.3	54.8
	3	54.2	52.3	53.2
PMI		56.7	50.8	53.6

Table 1: Results of PMI and the traditional PU learning approach: S-SVM and S-AMP.

Iteration	Precision	Recall	F-score	Spy Words
0	55.4	51.8	53.5	456
1	57.1	52.8	54.8	372
2	59.8	53.8	56.6	268
...	...	...	...	...
6	67.4	59.3	63.1	143

Table 2: Results of the proposed SE-AMP.

Incorporating Double Dictionary Lookup (DL) - SE-AMP-DL: The double dictionary lookup (DL) method improves the results further. DL uses the most authoritative Chinese dictionary: The Contemporary Chinese Dictionary. However, only 379 out of the 1000 test words are in the dictionary, among which 87 are sentiment words. As Table 3 shows, the DL method alone has a high precision but low recall as a lot of sentiment words are not in the dictionary. After correction of the results from SE-AMP by DL, the F score improves from 63.1 of SE-AMP to 65.6 of SE-AMP-DL.

Note: We also tried to clean up the lexicon first using DL to reduce the noise in the P set

	Precision	Recall	F-score
DL	74.1	20.1	31.6
SE-AMP	67.4	59.3	63.1
SE-AMP-DL	69.3	62.3	65.6

Table 3: Results with double dictionary lookup.

			Prec.	Rec.	F-score
PMI	199 set	pos.	60.7	65.4	63.0
		neg.	76.5	72.7	74.6
	identified set	pos.	43.8	35.9	39.2
		neg.	45.1	42.1	43.6
CPA	199 set	pos.	83.8	79.5	81.6
		neg.	87.2	90.1	88.6
	identified set	pos.	60.9	50.0	54.9
		neg.	65.2	60.3	62.6

Table 4: PMI and CPA classification results.

before performing the proposed algorithms. But after cleaning, only 1968 sentiment words out of 4957 remained. We inspected the result and found that the cleaning removed a lot of true sentiment words, making the P set too small for our algorithms and thus produced much poorer results.

4.2 Polarity Classification

Here we use the well-known PMI method in (Turney, 2002) as the baseline, which was designed for polarity classification. Again, for PMI computation, we use 100 positive and 100 negative words in our lexicon that appear most frequently in the Weibo corpus as the reference words, and compute the PMI scores between the candidate words and the references. Using many other numbers of sentiment words in the lexicon as the reference words give poorer results (see also Sec. 4.1.2). A word is considered as a positive sentiment word if its score is positive, or negative if the score is negative.

We apply the proposed CPA method and PMI to all 199 true sentiment words in the test set (199 set), and the sentiment words identified by SE-AMP-DL (identified set), which has many errors, i.e., non-sentiment words, Experimental results in Table 4 show that CPA outperforms PMI greatly in both cases. Those non-sentiment words are considered wrong in the "identified set" case.

5 Conclusion

This paper made a new attempt to study sentiment lexicon expansion based on a social media corpus. It first showed that the problem can be formulated as PU learning. It then proposed an *augmented multilayer perceptron* method to give PU learning an neural network solution. It then proposed a new PU learning method, which outperforms a classic existing PU learning method. To improve the results further, it proposed a double dictionary lookup technique. Additionally, a novel polarity classification method was also designed. Experimental results demonstrated the effectiveness of these proposed methods.

Acknowledgments

Bing Liu's work was supported in part by National Science Foundation (NSF) under grant no. IIS-1407927 and IIS-1650900. We would like to express our gratitude to Feng Liu, Jiansheng Wei and Youliang Yan from Noah's Ark Lab, Huawei Technologies for their help and excellent advice in this project.

References

Alina Andreevskaia and Sabine Bergler. 2006. Mining wordnet for a fuzzy sentiment: Sentiment tag extraction from wordnet glosses. In *EACL*. volume 6, pages 209–216.

Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George A Reis, and Jeff Reynar. 2008. Building a sentiment summarizer for local service reviews. In *WWW workshop on NLP in the information explosion era*. volume 14, pages 339–348.

Felipe Bravo-Marquez, Eibe Frank, and Bernhard Pfahringer. 2015. Positive, negative, or neutral: Learning an expanded opinion lexicon from emoticon-annotated tweets. In *IJCAI 2015*. AAAI Press, volume 2015, pages 1229–1235.

Erik Cambria, Soujanya Poria, Rajiv Bajpai, and Björn Schuller. 2016. Senticnet 4: A semantic resource for sentiment analysis based on conceptual primitives. In *the 26th International Conference on Computational Linguistics (COLING), Osaka*.

Yejin Choi and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 793–801.

Yejin Choi and Claire Cardie. 2009. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*. Association for Computational Linguistics, pages 590–598.

François Denis, Remi Gilleron, and Marc Tommasi. 2002. Text classification from positive and unlabeled examples. In *Proceedings of IPMU*.

Xiaowen Ding, Bing Liu, and Philip S Yu. 2008. A holistic lexicon-based approach to opinion mining.

In *Proceedings of the 2008 international conference on web search and data mining*. pages 231–240.

Eduard C Dragut, Clement Yu, Prasad Sistla, and Weiyi Meng. 2010. Construction of a sentimental word dictionary. In *Proceedings of the 19th ACM international conference on Information and knowledge management*. pages 1761–1764.

Weifu Du, Songbo Tan, Xueqi Cheng, and Xiaochun Yun. 2010. Adapting information bottleneck method for automatic construction of domain-oriented sentiment lexicon. In *Proceedings of the third ACM international conference on Web search and data mining*. pages 111–120.

Charles Elkan and Keith Noto. 2008. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. pages 213–220.

Andrea Esuli and Fabrizio Sebastiani. 2005. Determining the semantic orientation of terms through gloss classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. pages 617–624.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC06*. pages 417–422.

Song Feng, Ritwik Bose, and Yejin Choi. 2011. Learning general connotation of words using graph-based algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1092–1103.

William L Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ahmed Hassan and Dragomir Radev. 2010. Identifying text polarity using random walks. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 395–403.

Vasileios Hatzivassiloglou and Kathleen R McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 174–181.

Cho-Jui Hsieh, Nagarajan Natarajan, and Inderjit S Dhillon. 2015. Pu learning for matrix completion. In *ICML*. pages 2445–2453.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. pages 168–177.

Minlie Huang, Borui Ye, Yichen Wang, Haiqiang Chen, Junjun Cheng, and Xiaoyan Zhu. 2014. New word detection for sentiment analysis. In *ACL (1)*. pages 531–541.

Valentin Jijkoun, Maarten de Rijke, and Wouter Weerkamp. 2010. Generating focused topic-specific sentiment lexicons. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 585–594.

Nobuhiro Kaji and Masaru Kitsuregawa. 2007. Building lexicon for sentiment analysis from massive collection of html documents. In *EMNLP-CoNLL*. pages 1075–1083.

Jaap Kamps, Maarten Marx, Robert J Mokken, Maarten De Rijke, et al. 2004. Using wordnet to measure semantic orientations of adjectives. In *LREC*. Citeseer, volume 4, pages 1115–1118.

Hiroshi Kanayama and Tetsuya Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the 2006 conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 355–363.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics, page 1367.

Xiaoli Li and Bing Liu. 2003. Learning to classify texts using positive and unlabeled data. In *IJCAI*. volume 3, pages 587–592.

Bing Liu. 2011. *Web data mining: exploring hyperlinks, contents, and usage data. Second Edition*. Springer.

Bing Liu. 2012. *Sentiment analysis and data mining*. Morgan Claypool Publishers.

Bing Liu, Wee Sun Lee, Philip S. Yu, and Xiaoli Li. 2002. Partially supervised classification of text documents. In *Proceedings of the Nineteenth International Conference on Mach ine Learning (ICML-2002)*.

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

Saif Mohammad, Cody Dunne, and Bonnie Dorr. 2009. Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus. In

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2. Association for Computational Linguistics, pages 599–608.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242* .

Veronica Perez-Rosas, Carmen Banea, and Rada Mihalcea. 2012. Learning sentiment lexicons in spanish. In *LREC*. volume 12, page 73.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational linguistics* 37(1):9–27.

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 675–682.

Andrew Schneider and Eduard C Dragut. 2015. Towards debugging sentiment lexicons. In *ACL (1)*. pages 1024–1034.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2007. Extracting semantic orientations of phrases from dictionary. In *HLT-NAACL*. volume 2007, pages 292–299.

Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and Ting Liu. 2014. Building large-scale twitter-specific sentiment lexicon: A representation learning approach. In *COLING*. pages 172–182.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 417–424.

Alessandro Valitutti, Carlo Strapparava, and Oliviero Stock. 2004. Developing affective lexical resources. *PsychNology Journal* 2(1):61–83.

Duy Tin Vo and Yue Zhang. 2016. Dont count, predict! an automatic approach to learning sentiment lexicons for short text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. volume 2, pages 219–224.

Bo Wang and Houfeng Wang. 2008. Bootstrapping both product features and opinion words from chinese customer reviews with cross-inducing. In *IJCNLP*. volume 8, pages 289–295.

Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *EMNLP*. pages 935–945.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*. Association for Computational Linguistics, pages 347–354.

Ge Xu, Xinfan Meng, and Houfeng Wang. 2010a. Build chinese emotion lexicons using a graph-based algorithm and multiple resources. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 1209–1217.

Hongzhi Xu, Kai Zhao, Likun Qiu, and Changjian Hu. 2010b. Expanding chinese sentiment dictionaries from large scale unlabeled corpus. In *PACLIC*. pages 301–310.

Linhong Xu, Hongfei Lin, Yu Pan, Hui Ren, and Jianmei Chen. 2008. Constructing the affective lexicon ontology. *Journal of the China Society for Scientific and Technical Information* 2:180–185.

Min Yang, Baolin Peng, Zheng Chen, Dingju Zhu, and Kam-Pui Chow. 2014. A topic model for building fine-grained domain-specific emotion lexicon. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*. Association for Computational Linguistics (ACL).

Hwanjo Yu, Jiawei Han, and Kevin Chen-Chuan Chang. 2002. Pebl: positive example based learning for web page classification using svm. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. pages 239–248.

Lei Zhang and Bing Liu. 2011. Identifying noun product features that imply opinions. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, pages 575–580.

DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning

Wenhan Xiong and **Thien Hoang** and **William Yang Wang**
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
{xwhan,william}@cs.ucsb.edu, thienhoang@umail.ucsb.edu

Abstract

We study the problem of learning to reason in large scale knowledge graphs (KGs). More specifically, we describe a novel reinforcement learning framework for learning multi-hop relational paths: we use a policy-based agent with continuous states based on knowledge graph embeddings, which reasons in a KG vector space by sampling the most promising relation to extend its path. In contrast to prior work, our approach includes a reward function that takes the **accuracy**, **diversity**, and **efficiency** into consideration. Experimentally, we show that our proposed method outperforms a path-ranking based algorithm and knowledge graph embedding methods on Freebase and Never-Ending Language Learning datasets.[1]

1 Introduction

In recent years, deep learning techniques have obtained many state-of-the-art results in various classification and recognition problems (Krizhevsky et al., 2012; Hinton et al., 2012; Kim, 2014). However, complex natural language processing problems often require multiple inter-related decisions, and empowering deep learning models with the ability of learning to reason is still a challenging issue. To handle complex queries where there are no obvious answers, intelligent machines must be able to reason with existing resources, and learn to infer an unknown answer.

More specifically, we situate our study in the context of multi-hop reasoning, which is the task of learning explicit inference formulas, given a large KG. For example, if the KG includes the beliefs such as *Neymar* plays for *Barcelona*, and *Barcelona* are in the *La Liga* league, then machines should be able to learn the following formula: *playerPlaysForTeam(P,T)* $\wedge$ *teamPlaysInLeague(T,L)* $\Rightarrow$ *playerPlaysInLeague(P,L)*. In the testing time, by plugging in the learned formulas, the system should be able to automatically infer the missing link between a pair of entities. This kind of reasoning machine will potentially serve as an essential components of complex QA systems.

In recent years, the Path-Ranking Algorithm (PRA) (Lao et al., 2010, 2011a) emerges as a promising method for learning inference paths in large KGs. PRA uses a random-walk with restarts based inference mechanism to perform multiple bounded depth-first search processes to find relational paths. Coupled with elastic-net based learning, PRA then picks more plausible paths using supervised learning. However, PRA operates in a fully discrete space, which makes it difficult to evaluate and compare similar entities and relations in a KG.

In this work, we propose a novel approach for controllable multi-hop reasoning: we frame the path learning process as reinforcement learning (RL). In contrast to PRA, we use translation-based knowledge based embedding method (Bordes et al., 2013) to encode the continuous state of our RL agent, which reasons in the vector space environment of the knowledge graph. The agent takes incremental steps by sampling a relation to extend its path. To better guide the RL agent for learning relational paths, we use policy gradient training (Mnih et al., 2015) with a novel reward function that jointly encourages accuracy, diversity, and efficiency. Empirically, we show that our method outperforms PRA and embedding based methods on a Freebase and a Never-Ending Language Learning (Carlson et al., 2010a) dataset.

[1] Code and the NELL dataset are available at https://github.com/xwhan/DeepPath.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 564–573
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Our contributions are three-fold:

- We are the first to consider reinforcement learning (RL) methods for learning relational paths in knowledge graphs;
- Our learning method uses a complex reward function that considers accuracy, efficiency, and path diversity simultaneously, offering better control and more flexibility in the path-finding process;
- We show that our method can scale up to large scale knowledge graphs, outperforming PRA and KG embedding methods in two tasks.

In the next section, we outline related work in path-finding and embedding methods in KGs. We describe the proposed method in Section 3. We show experimental results in Section 4. Finally, we conclude in Section 5.

2 Related Work

The Path-Ranking Algorithm (PRA) method (Lao et al., 2011b) is a primary path-finding approach that uses random walk with restart strategies for multi-hop reasoning. Gardner et al. (2013; 2014) propose a modification to PRA that computes feature similarity in the vector space. Wang and Cohen (2015) introduce a recursive random walk approach for integrating the background KG and text—the method performs structure learning of logic programs and information extraction from text at the same time. A potential bottleneck for random walk inference is that supernodes connecting to large amount of formulas will create huge fan-out areas that significantly slow down the inference and affect the accuracy.

Toutanova et al. (2015) provide a convolutional neural network solution to multi-hop reasoning. They build a CNN model based on lexicalized dependency paths, which suffers from the error propagation issue due to parse errors. Guu et al. (2015) uses KG embeddings to answer path queries. Zeng et al. (2014) described a CNN model for relational extraction, but it does not explicitly model the relational paths. Neelakantan et al. (2015) propose a recurrent neural networks model for modeling relational paths in knowledge base completion (KBC), but it trains too many separate models, and therefore it does not scale. Note that many of the recent KG reasoning methods (Neelakantan et al., 2015; Das et al., 2017) still rely on first learning the PRA paths, which only operates in a discrete space. Comparing to PRA, our method reasons in a continuous space, and by incorporating various criteria in the reward function, our reinforcement learning (RL) framework has better control and more flexibility over the path-finding process.

Neural symbolic machine (Liang et al., 2016) is a more recent work on KG reasoning, which also applies reinforcement learning but has a different flavor from our work. NSM learns to compose programs that can find answers to natural language questions, while our RL model tries to add new facts to knowledge graph (KG) by reasoning on existing KG triples. In order to get answers, NSM learns to generate a sequence of actions that can be combined as a executable program. The action space in NSM is a set of predefined tokens. In our framework, the goal is to find reasoning paths, thus the action space is relation space in the KG. A similar framework (Johnson et al., 2017) has also been applied to visual reasoning tasks.

3 Methodology

In this section, we describe in detail our RL-based framework for multi-hop relation reasoning. The specific task of relation reasoning is to find reliable predictive paths between entity pairs. We formulate the path finding problem as a sequential decision making problem which can be solved with a RL agent. We first describe the environment and the policy-based RL agent. By interacting with the environment designed around the KG, the agent learns to pick the promising reasoning paths. Then we describe the training procedure of our RL model. After that, we describe an efficient path-constrained search algorithm for relation reasoning with the paths found by the RL agent.

3.1 Reinforcement Learning for Relation Reasoning

The RL system consists of two parts (see Figure 1). The first part is the external environment $\mathcal{E}$ which specifies the dynamics of the interaction between the agent and the KG. This environment is modeled as a Markov decision process (MDP). A tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$ is defined to represent the MDP, where $\mathcal{S}$ is the continuous state space, $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ is the set of all available actions, $\mathcal{P}(S_{t+1} = s^{'}|S_t = s, A_t = a)$ is the transition probability matrix, and $\mathcal{R}(s, a)$ is the reward

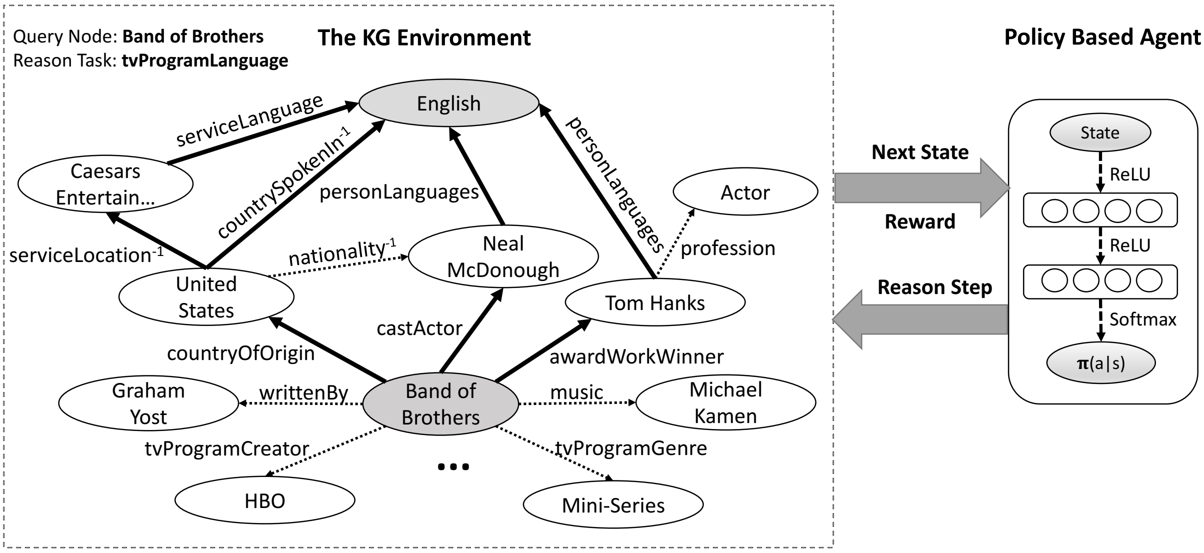


Figure 1: Overview of our RL model. **Left:** The KG environment $\mathcal{E}$ modeled by a MDP. The dotted arrows (partially) show the existing relation links in the KG and the bold arrows show the reasoning paths found by the RL agent. $^{-1}$ denotes the inverse of an relation. **Right:** The structure of the policy network agent. At each step, by interacting with the environment, the agent learns to pick a relation link to extend the reasoning paths.

function of every (s, a) pairs.

The second part of the system, the RL agent, is represented as a policy network $\pi_\theta(s, a) = p(a|s; \theta)$ which maps the state vector to a stochastic policy. The neural network parameters θ are updated using stochastic gradient descent. Compared to Deep Q Network (DQN) (Mnih et al., 2013), policy-based RL methods turn out to be more appropriate for our knowledge graph scenario. One reason is that for the path finding problem in KG, the action space can be very large due to complexity of the relation graph. This can lead to poor convergence properties for DQN. Besides, instead of learning a greedy policy which is common in value-based methods like DQN, the policy network is able to learn a stochastic policy which prevent the agent from getting stuck at an intermediate state. Before we describe the structure of our policy network, we first describe the components (actions, states, rewards) of the RL environment.

Actions Given the entity pairs (e_s, e_t) with relation r, we want the agent to find the most informative paths linking these entity pairs. Beginning with the source entity e_s, the agent use the policy network to pick the most promising relation to extend its path at each step until it reaches the target entity e_t. To keep the output dimension of the policy network consistent, the action space is defined as all the relations in the KG.

States The entities and relations in a KG are naturally discrete atomic symbols. Since existing practical KGs like Freebase (Bollacker et al., 2008) and NELL (Carlson et al., 2010b) often have huge amounts of triples. It is impossible to directly model all the symbolic atoms in states. To capture the semantic information of these symbols, we use translation-based embeddings such as TransE (Bordes et al., 2013) and TransH (Wang et al., 2014) to represent the entities and relations. These embeddings map all the symbols to a low-dimensional vector space. In our framework, each state captures the agent's position in the KG. After taking an action, the agent will move from one entity to another. These two are linked by the action (relation) just taken by the agent. The state vector at step t is given as follows:

$$\mathbf{s}_t = (\mathbf{e}_t, \mathbf{e}_{target} - \mathbf{e}_t)$$

where $\mathbf{e}_t$ denotes the embeddings of the current entity node and $\mathbf{e}_{target}$ denotes the embeddings of the target entity. At the initial state, $\mathbf{e}_t = \mathbf{e}_{source}$. We do not incorporate the reasoning relation in the state, because the embedding of the reasoning relation remain constant during path finding, which is not helpful in training. However, we find out that by training the RL agent using a set of positive samples for one particular relation, the agent can successfully discover the relation semantics.

Rewards There are a few factors that contribute to the quality of the paths found by the RL agent. To encourage the agent to find predictive paths, our reward functions include the following scoring criteria:

Global accuracy: For our environment settings, the number of actions that can be taken by the agent can be very large. In other words, there are much more incorrect sequential decisions than the correct ones. The number of these incorrect decision sequences can increase exponentially with the length of the path. In view of this challenge, the first reward function we add to the RL model is defined as follows:

$$r_{\text{GLOBAL}} = \begin{cases} +1, & \text{if the path reaches } e_{target} \\ -1, & \text{otherwise} \end{cases}$$

the agent is given an offline positive reward $+1$ if it reaches the target after a sequence of actions.

Path efficiency: For the relation reasoning task, we observe that short paths tend to provide more reliable reasoning evidence than longer paths. Shorter chains of relations can also improve the efficiency of the reasoning by limiting the length of the RL's interactions with the environment. The efficiency reward is defined as follows:

$$r_{\text{EFFICIENCY}} = \frac{1}{length(p)}$$

where path p is defined as a sequence of relations $r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_n$.

Path diversity: We train the agent to find paths using positive samples for each relation. These training sample (e_{source}, e_{target}) have similar state representations in the vector space. The agent tends to find paths with similar syntax and semantics. These paths often contains redundant information since some of them may be correlated. To encourage the agent to find diverse paths, we define a diversity reward function using the cosine similarity between the current path and the existing ones:

$$r_{\text{DIVERSITY}} = -\frac{1}{|F|}\sum_{i=1}^{|F|} cos(\mathbf{p}, \mathbf{p}_i)$$

where $\mathbf{p} = \sum_{i=1}^{n} \mathbf{r}_i$ represents the path embedding for the relation chain $r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_n$.

Policy Network We use a fully-connected neural network to parameterize the policy function $\pi(s;\theta)$ that maps the state vector $\mathbf{s}$ to a probability distribution over all possible actions. The neural network consists of two hidden layers, each followed by a rectifier nonlinearity layer (ReLU). The output layer is normalized using a softmax function (see Figure 1).

3.2 Training Pipeline

In practice, one big challenge of KG reasoning is that the relation set can be quite large. For a typical KG, the RL agent is often faced with hundreds (thousands) of possible actions. In other words, the output layer of the policy network often has a large dimension. Due to the complexity of the relation graph and the large action space, if we directly train the RL model by trial and errors, which is typical for RL algorithms, the RL model will show very poor convergence properties. After a long-time training, the agents fails to find any valuable path. To tackle this problem, we start our training with a supervised policy which is inspired by the imitation learning pipeline used by *AlphaGo* (Silver et al., 2016). In the Go game, the player is facing nearly 250 possible legal moves at each step. Directly training the agent to pick actions from the original action space can be a difficult task. *AlphaGo* first train a supervised policy network using experts moves. In our case, the supervised policy is trained with a randomized breadth-first search (BFS).

Supervised Policy Learning For each relation, we use a subset of all the positive samples (entity pairs) to learn the supervised policy. For each positive sample (e_{source}, e_{target}), a two-side BFS is conducted to find same correct paths between the entities. For each path p with a sequence of relations $r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_n$, we update the parameters θ to maximize the expected cumulative reward using Monte-Carlo Policy Gradient (RE-

INFORCE) (Williams, 1992):

$$\begin{aligned} J(\theta) &= \mathbb{E}_{a\sim\pi(a|s;\theta)}(\sum_t R_{s_t,a_t}) \\ &= \sum_t \sum_{a\in\mathcal{A}} \pi(a|s_t;\theta) R_{s_t,a_t} \end{aligned} \qquad (1)$$

where $J(\theta)$ is the expected total rewards for one episode. For supervised learning, we give a reward of $+1$ for each step of a successful episode. By plugging in the paths found by the BFS, the approximated gradient used to update the policy network is shown below:

$$\begin{aligned} \nabla_\theta J(\theta) &= \sum_t \sum_{a\in\mathcal{A}} \pi(a|s_t;\theta) \nabla_\theta \log \pi(a|s_t;\theta) \\ &\approx \nabla_\theta \sum_t \log \pi(a = r_t|s_t;\theta) \end{aligned} \qquad (2)$$

where r_t belongs to the path p.

However, the vanilla BFS is a biased search algorithm which prefers short paths. When plugging in these biased paths, it becomes difficult for the agent to find longer paths which may potentially be useful. We want the paths to be controlled only by the defined reward functions. To prevent the biased search, we adopt a simple trick to add some random mechanisms to the BFS. Instead of directly searching the path between e_{source} and e_{target}, we randomly pick a intermediate node e_{inter} and then conduct two BFS between (e_{source}, e_{inter}) and (e_{inter}, e_{target}). The concatenated paths are used to train the agent. The supervised learning saves the agent great efforts learning from failed actions. With the learned experience, we then train the agent to find desirable paths.

Retraining with Rewards To find the reasoning paths controlled by the reward functions, we use reward functions to retrain the supervised policy network. For each relation, the reasoning with one entity pair is treated as one episode. Starting with the source node e_{source}, the agent picks a relation according to the stochastic policy $\pi(a|s)$, which is a probability distribution over all relations, to extend its reasoning path. This relation link may lead to a new entity, or it may lead to nothing. These failed steps will cause the agent to receive negative rewards. The agent will stay at the same state after these failed steps. Since the agent is following a stochastic policy, the agent will not get stuck by repeating a wrong step. To improve the training efficiency, we limit the episode length with an upper bound max_length. The episode ends if the agent fails to reach the target entity within max_length steps. After each episode, the policy network is updated using the following gradient:

Algorithm 1: Retraining Procedure with reward functions

```
Restore parameters $\theta$ from supervised policy;
for episode ← 1 to N do
    Initialize state vector $s_t \leftarrow s_0$
    Initialize episode length $steps \leftarrow 0$
    while $num\_steps < max\_length$ do
        Randomly sample action $a \sim \pi(a|s_t)$
        Observe reward $\mathcal{R}_t$, next state $s_{t+1}$
        // if the step fails
        if $\mathcal{R}_t = -1$ then
            Save $< s_t, a >$ to $\mathcal{M}_{neg}$
        if success or $steps = max\_length$
          then
            break
        Increment $num\_steps$
    // penalize failed steps
    Update $\theta$ using
      $g \propto \nabla_\theta \sum_{\mathcal{M}_{neg}} \log \pi(a = r_t|s_t;\theta)(-1)$
    if success then
        $R_{total} \leftarrow \lambda_1 r_{\text{GLOBAL}} + \lambda_2 r_{\text{EFFICIENCY}} + \lambda_3 r_{\text{DIVERSITY}}$
        Update $\theta$ using
          $g \propto \nabla_\theta \sum_t \log \pi(a = r_t|s_t;\theta) R_{total}$
```

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_t \log \pi(a = r_t|s_t;\theta) R_{total} \qquad (3)$$

where R_{total} is the linear combination of the defined reward functions. The detail of the retrain process is shown in Algorithm 1. In practice, θ is updated using the Adam Optimizer (Kingma and Ba, 2014) with L_2 regularization.

3.3 Bi-directional Path-constrained Search

Given an entity pair, the reasoning paths learned by the RL agent can be used as logical formulas to predict the relation link. Each formula is verified using a bi-directional search. In a typical KG, one entity node can be linked to a large number of neighbors with the same relation link. A simple example is the relation *personNationality*$^{-1}$, which denotes the inverse of *personNationality*. Following this link, the entity *United States* can reach numerous neighboring entities. If the for-

Algorithm 2: Bi-directional search for path verification

```
Given a reasoning path
  $p : r_1 \rightarrow r_2 \rightarrow ... \rightarrow r_n$
for $(e_i, e_j)$ in test set $\mathcal{D}$ do
    start ← 0; end ← n
    $left \leftarrow \emptyset; right \leftarrow \emptyset$
    while start < end do
        $leftEx \leftarrow \emptyset; rightEx \leftarrow \emptyset$
        if len(left) < len(right) then
            Extend path on the left side
            Add connected nodes to $leftEx$
            $left \leftarrow leftEx$
        else
            Extend path on the right side
            Add connected nodes to $rightEx$
            $right \leftarrow rightEx$
    if $left \cap right \neq \emptyset$ then
        return True
    else
        return False
```

mula consists of such links, the number of intermediate entities can exponentially increase as we follow the reasoning formula. However, we observe that for these formulas, if we verify the formula from the inverse direction. The number of intermediate nodes can be tremendously decreased. Algorithm 2 shows a detailed description of the proposed bi-directional search.

4 Experiments

To evaluate the reasoning formulas found by our RL agent, we explore two standard KG reasoning tasks: link prediction (predicting target entities) and fact prediction (predicting whether an unknown fact holds or not). We compare our method with both path-based methods and embedding based methods. After that, we further analyze the reasoning paths found by our RL agent. These highly predictive paths validate the effectiveness of the reward functions. Finally, we conduct a experiment to investigate the effect of the supervised learning procedure.

4.1 Dataset and Settings

Table 1 shows the statistics of the two datasets we conduct our experiments on. Both of them are subsets of larger datasets. The triples in FB15K-237 (Toutanova et al., 2015) are sampled from FB15K (Bordes et al., 2013) with redundant relations removed. We perform the reasoning tasks on 20 relations which have enough reasoning paths. These tasks consists of relations from different domains like *Sports*, *People*, *Locations*, *Film*, etc. Besides, we present a new NELL subset that is suitable for multi-hop reasoning from the 995th iteration of the NELL system. We first remove the triples with relation *generalizations* or *haswikipediaurl*. These two relations appear more than 2M times in the NELL dataset, but they have no reasoning values. After this step, we only select the triples with Top-200 relations. To facilitate path finding, we also add the inverse triples. For each triple (h, r, t), we append (t, r^{-1}, h) to the datasets. With these inverse triples, the agent is able to step backward in the KG.

Dataset	# Ent.	# R.	# Triples	# Tasks
FB15K-237	14,505	237	310,116	20
NELL-995	75,492	200	154.213	12

Table 1: Statistics of the Datasets. # Ent. denotes the number of unique entities and # R. denotes the number of relations

For each reasoning task r_i, we remove all the triples with r_i or r_i^{-1} from the KG. These removed triples are split into train and test samples. For the link prediction task, each h in the test triples $\{(h, r, t)\}$ is considered as one query. A set of candidate target entities are ranked using different methods. For fact prediction, the true test triples are ranked with some generated false triples.

4.2 Baselines and Implementation Details

Most KG reasoning methods are based on either path formulas or KG embeddings. we explore methods from both of these two classes in our experiments. For path based methods, we compare our RL model with the PRA (Lao et al., 2011a) algorithm, which has been used in a couple of reasoning methods (Gardner et al., 2013; Neelakantan et al., 2015). PRA is a data-driven algorithm using random walks (RW) to find paths and obtain path features. For embedding based methods, we evaluate several state-of-the-art embeddings designed for knowledge base completion, such as TransE (Bordes et al., 2013), TransH (Wang et al., 2014), TransR (Lin et al., 2015) and TransD (Ji et al., 2015) .

The implementation of PRA is based on the

	FB15K-237					NELL-995			
Tasks	PRA	RL	TransE	TransR	Tasks	PRA	RL	TransE	TransR
teamSports	**0.987**	0.955	0.896	0.784	athletePlaysForTeam	0.547	**0.750**	0.627	0.673
birthPlace	0.441	**0.531**	0.403	0.417	athletePlaysInLeague	0.841	**0.960**	0.773	0.912
personNationality	**0.846**	0.823	0.641	0.720	athleteHomeStadium	0.859	**0.890**	0.718	0.722
filmDirector	0.349	**0.441**	0.386	0.399	athletePlaysSport	0.474	0.957	0.876	**0.963**
filmWrittenBy	0.601	0.457	0.563	**0.605**	teamPlaySports	0.791	0.738	0.761	**0.814**
filmLanguage	0.663	**0.670**	0.642	0.641	orgHeadquaterCity	**0.811**	0.790	0.620	0.657
tvLanguage	0.960	**0.969**	0.804	0.906	worksFor	0.681	**0.711**	0.677	0.692
capitalOf	**0.829**	0.783	0.554	0.493	bornLocation	0.668	0.757	0.712	**0.812**
organizationFounded	0.281	0.309	**0.390**	0.339	personLeadsOrg	0.700	**0.795**	0.751	0.772
musicianOrigin	0.426	**0.514**	0.361	0.379	orgHiredPerson	0.599	**0.742**	0.719	0.737
...					...				
Overall	0.541	**0.572**	0.532	0.540		0.675	**0.796**	0.737	0.789

Table 2: Link prediction results (MAP) on two datasets.

code released by (Lao et al., 2011a). We use the TopK negative mode to generate negative samples for both train and test samples. For each positive samples, there are approximately 10 corresponding negative samples. Each negative sample is generated by replacing the true target entity t with a faked one $t^{'}$ in each triple (h, r, t). These positive and negative test pairs generated by PRA make up the test set for all methods evaluated in this paper. For TransE,R,H,D, we learn a separate embedding matrix for each reasoning task using the positive training entity pairs. All these embeddings are trained for 1,000 epochs. [2]

Our RL model make use of TransE to get the continuous representation of the entities and relations. We use the same dimension as TransE, R to embed the entities. Specifically, the state vector we use has a dimension of 200, which is also the input size of the policy network. To reason using the path formulas, we adopt a similar linear regression approach as in PRA to re-rank the paths. However, instead of using the random walk probabilities as path features, which can be computationally expensive, we simply use binary path features obtained by the bi-directional search. We observe that with only a few mined path formulas, our method can achieve better results than PRA's data-driven approach.

	Fact Prediction Results	
Methods	FB15K-237	NELL-995
RL	**0.311**	**0.493**
TransE	0.277	0.383
TransH	0.309	0.389
TransR	0.302	0.406
TransD	0.303	0.413

Table 3: Fact prediction results (MAP) on two datasets.

	# of Reasoning Paths	
Tasks	PRA	RL
worksFor	247	25
teamPlaySports	113	27
teamPlaysInLeague	69	21
athletehomestadium	37	11
organizationHiredPerson	244	9
...		
Average #	137.2	20.3

Table 4: Number of reasoning paths used by PRA and our RL model. *RL achieved better MAP with a more compact set of learned paths.*

4.3 Results

4.3.1 Quantitative Results

Link Prediction This task is to rank the target entities given a query entity. Table 2 shows the mean average precision (MAP) results on two datasets. Since path-based methods generally work better than embedding methods for this task, we do not include the other two embedding baselines in this table. Instead, we spare the room to show the detailed results on each relation reasoning task.

For the overall MAP shown in the last row of the table, our approach significantly outperforms both the path-based method and embedding methods on two datasets, which validates the strong reasoning ability of our RL model. For most relations, since the embedding methods fail to use the path infor-

[2] The implementation we used can be found at https://github.com/thunlp/Fast-TransX

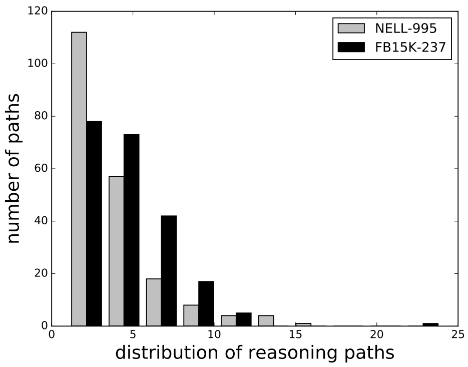


Figure 2: The distribution of paths lengths on two datasets

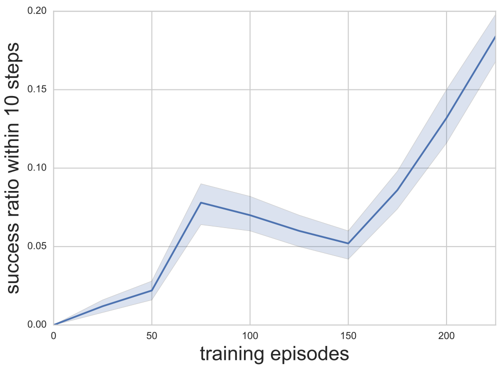


Figure 3: The success ratio ($succ_{10}$) during training. Task: athletePlaysForTeam.[3]

mation in the KG, they generally perform worse than our RL model or PRA. However, when there are not enough paths between entities, our model and PRA can give poor results. For example, for the relation *filmWrittenBy*, our RL model only finds 4 unique reasoning paths, which means there is actually not enough reasoning evidence existing in the KG. Another observation is that we always get better performance on the NELL dataset. By analyzing the paths found from the KGs, we believe the potential reason is that the NELL dataset has more short paths than FB15K-237 and some of them are simply synonyms of the reasoning relations.

Fact Prediction Instead of ranking the target entities, this task directly ranks all the positive and negative samples for a particular relation. The PRA is not included as a baseline here, since the PRA code only gives a target entity ranking for each query node instead of a ranking of all triples. Table 3 shows the overall results of all the methods. Our RL model gets even better results on this task. We also observe that the RL model beats all the embedding baselines on most reasoning tasks.

4.3.2 Qualitative Analysis of Reasoning Paths

To analyze the properties of reasoning paths, we show a few reasoning paths found by the agent in Table 5. To illustrate the effect of the efficiency reward function, we show the path length distributions in Figure 2. To interpret these paths, take the *personNationality* relation for example, the first reasoning path indicates that if we know facts *placeOfBirth(x,y)* and *locationContains(z,y)* then it is highly possible that person x has nationality z. These short but predictive paths indicate the effectiveness of the RL model. Another important observation is that our model use much fewer reasoning paths than PRA, which indicates that our model can actually extract the most reliable reasoning evidence from KG. Table 4 shows some comparisons about the number of reasoning paths. We can see that, with the pre-defined reward functions, the RL agent is capable of picking the strong ones and filter out similar or irrelevant ones.

4.3.3 Effect of Supervised Learning

As mentioned in Section 3.2, one major challenge for applying RL to KG reasoning is the large action space. We address this issue by applying supervised learning before the reward retraining step. To show the effect of the supervised training, we evaluate the agent's success ratio of reaching the target within 10 steps ($succ_{10}$) after different number of training episodes. For each training episode, one pair of entities (e_{source}, e_{target}) in the train set is used to find paths. All the correct paths linking the entities will get a $+1$ global reward. We then plug in some true paths for training. The $succ_{10}$ is calculated on a held-out test set that consists of 100 entity pairs. For the NELL-995 dataset, since we have 200 unique relations, the dimension of the action space will be 400 after we add the backward actions. This means that random walks will get very low $succ_{10}$ since there may be nearly 400^{10} invalid paths. Figure 3 shows the $succ_{10}$ during training. We see that even the agent has not seen the entity before, it can actually pick the promising relation to extend its path. This also validates the effectiveness of our state representations.

[3] The confidence band is generated using 50 different runs.

Relation	Reasoning Path
filmCountry	filmReleaseRegion featureFilmLocation → locationContains^{-1} actorFilm^{-1} → personNationality
personNationality	placeOfBirth→ locationContains^{-1} peoplePlaceLived → locationContains^{-1} peopleMarriage → locationOfCeremony → locationContains^{-1}
tvProgramLanguage	tvCountryOfOrigin → countryOfficialLanguage tvCountryOfOrigin → filmReleaseRegion^{-1} → filmLanguage tvCastActor → filmLanguage
personBornInLocation	personBornInCity graduatedUniversity → graduatedSchool^{-1} → personBornInCity personBornInCity → atLocation^{-1} → atLocation
athletePlaysForTeam	athleteHomeStadium → teamHomeStadium^{-1} athletePlaysSport → teamPlaysSport^{-1} athleteLedSportsTeam
personLeadsOrganization	worksFor organizationTerminatedPerson^{-1} mutualProxyFor^{-1}

Table 5: Example reasoning paths found by our RL model. The first three relations come from the FB15K-237 dataset. The others are from NELL-995. Inverses of existing relations are denoted by $^{-1}$.

5 Conclusion and Future Work

In this paper, we propose a reinforcement learning framework to improve the performance of relation reasoning in KGs. Specifically, we train a RL agent to find reasoning paths in the knowledge base. Unlike previous path finding models that are based on random walks, the RL model allows us to control the properties of the found paths. These effective paths can also be used as an alternative to PRA in many path-based reasoning methods. For two standard reasoning tasks, using the RL paths as reasoning formulas, our approach generally outperforms two classes of baselines.

For future studies, we plan to investigate the possibility of incorporating adversarial learning (Goodfellow et al., 2014) to give better rewards than the human-defined reward functions used in this work. Instead of designing rewards according to path characteristics, a discriminative model can be trained to give rewards. Also, to address the problematic scenario when the KG does not have enough reasoning paths, we are interested in applying our RL framework to joint reasoning with KG triples and text mentions.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

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

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010b. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. *EACL*.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom M Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *EMNLP*, pages 833–838.

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *EMNLP*.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL (1)*, pages 687–696.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Inferring and executing programs for visual reasoning. *arXiv preprint arXiv:1705.03633*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Ni Lao, Tom Mitchell, and William W Cohen. 2011a. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539. Association for Computational Linguistics.

Ni Lao, Tom M. Mitchell, and William W. Cohen. 2011b. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, pages 529–539. ACL.

Ni Lao, Jun Zhu, Xinwang Liu, Yandong Liu, and William W Cohen. 2010. Efficient relational learning with hidden variable detection. In *NIPS*, pages 1234–1242.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. *arXiv preprint arXiv:1504.06662*.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, volume 15, pages 1499–1509. Citeseer.

William Yang Wang and William W Cohen. 2015. Joint information extraction and reasoning: A scalable statistical relational learning approach. In *ACL*.

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

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

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

Task-Oriented Query Reformulation with Reinforcement Learning

Rodrigo Nogueira
Tandon School of Engineering
New York University
rodrigonogueira@nyu.edu

Kyunghyun Cho
Courant Institute of Mathematical Sciences
Center for Data Science
New York University
kyunghyun.cho@nyu.edu

Abstract

Search engines play an important role in our everyday lives by assisting us in finding the information we need. When we input a complex query, however, results are often far from satisfactory. In this work, we introduce a query reformulation system based on a neural network that rewrites a query to maximize the number of relevant documents returned. We train this neural network with reinforcement learning. The actions correspond to selecting terms to build a reformulated query, and the reward is the document recall. We evaluate our approach on three datasets against strong baselines and show a relative improvement of 5-20% in terms of recall. Furthermore, we present a simple method to estimate a conservative upper-bound performance of a model in a particular environment and verify that there is still large room for improvements.

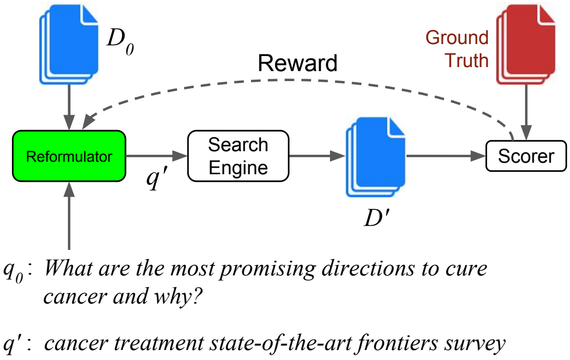

D_0
Reward
Ground Truth
Reformulator
Search Engine
Scorer
q'
D'
q_0: What are the most promising directions to cure cancer and why?
q': cancer treatment state-of-the-art frontiers survey

Figure 1: A graphical illustration of the proposed framework for query reformulation. A set of documents D_0 is retrieved from a search engine using the initial query q_0. Our reformulator selects terms from q_0 and D_0 to produce a reformulated query q' which is then sent to the search engine. Documents D' are returned, and a reward is computed against the set of ground-truth documents. The reformulator is trained with reinforcement learning to produce a query, or a series of queries, to maximize the expected return.

1 Introduction

Search engines help us find what we need among the vast array of available data. When we request some information using a long or inexact description of it, these systems, however, often fail to deliver relevant items. In this case, what typically follows is an iterative process in which we try to express our need differently in the hope that the system will return what we want. This is a major issue in information retrieval. For instance, Huang and Efthimiadis (2009) estimate that 28-52% of all the web queries are modifications of previous ones.

To a certain extent, this problem occurs because search engines rely on matching words in the query with words in relevant documents, to perform retrieval. If there is a mismatch between them, a relevant document may be missed.

One way to address this problem is to automatically rewrite a query so that it becomes more likely to retrieve relevant documents. This technique is known as *automatic query reformulation*. It typically expands the original query by adding terms from, for instance, dictionaries of synonyms such as WordNet (Miller, 1995), or from the initial set of retrieved documents (Xu and Croft, 1996). This latter type of reformulation is known as pseudo (or blind) relevance feedback (PRF), in which the relevance of each term of the retrieved documents is automatically inferred.

The proposed method is built on top of PRF but differs from previous works as we frame the query

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 574–583
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

reformulation problem as a reinforcement learning (RL) problem. An initial query is the natural language expression of the desired goal, and an agent (i.e. reformulator) *learns* to reformulate an initial query to maximize the expected return (i.e. retrieval performance) through actions (i.e. selecting terms for a new query). The environment is a search engine which produces a new state (i.e. retrieved documents). Our framework is illustrated in Fig. 1.

The most important implication of this framework is that a search engine is treated as a *black box* that an agent learns to use in order to retrieve more relevant items. This opens the possibility of training an agent to use a search engine for a task other than the one it was originally intended for. To support this claim, we evaluate our agent on the task of question answering (Q&A), citation recommendation, and passage/snippet retrieval.

As for training data, we use two publicly available datasets (TREC-CAR and Jeopardy) and introduce a new one (MS Academic) with hundreds of thousands of *query/relevant document* pairs from the academic domain.

Furthermore, we present a method to estimate the upper bound performance of our RL-based model. Based on the estimated upper bound, we claim that this framework has a strong potential for future improvements.

Here we summarize our main contributions:

- A reinforcement learning framework for automatic query reformulation.
- A simple method to estimate the upper-bound performance of an RL-based model in a given environment.
- A new large dataset with hundreds of thousands of *query/relevant document* pairs.[1]

2 A Reinforcement Learning Approach

2.1 Model Description

In this section we describe the proposed method, illustrated in Fig. 2.

The inputs are a query q_0 consisting of a sequence of words $(w_1, ..., w_n)$ and a candidate term t_i with some context words $(t_{i-k}, ..., t_{i+k})$, where $k \geq 0$ is the context window size. Candidate terms are from $q_0 \cup D_0$, the union of the terms in the original query and those from the documents D_0 retrieved using q_0.

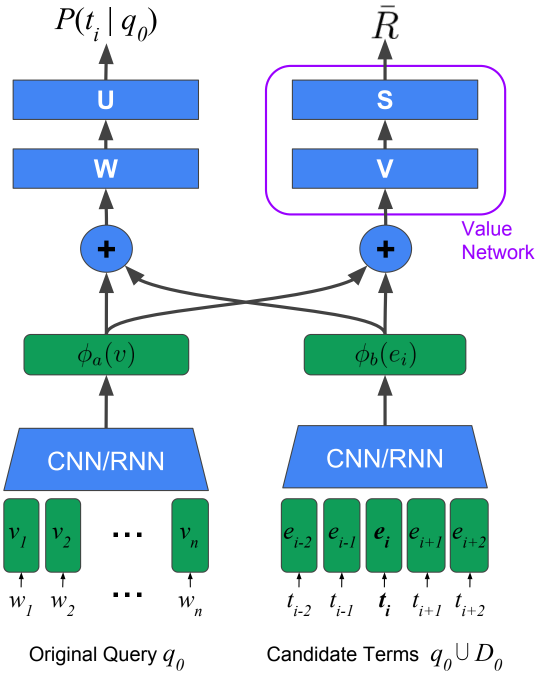


Figure 2: An illustration of our neural network-based reformulator.

We use a dictionary of pretrained word embeddings (Mikolov et al., 2013) to convert the symbolic terms w_j and t_i to their vector representations v_j and $e_i \in \mathbb{R}^d$, respectively. We map out-of-vocabulary terms to an additional vector that is learned during training.

We convert the sequence $\{v_j\}$ to a fixed-size vector $\phi_a(v)$ by using either a Convolutional Neural Network (CNN) followed by a max pooling operation over the entire sequence (Kim, 2014) or by using the last hidden state of a Recurrent Neural Network (RNN).[2]

Similarly, we fed the candidate term vectors e_i to a CNN or RNN to obtain a vector representation $\phi_b(e_i)$ for each term t_i. The convolutional/recurrent layers serve an important role of capturing context information, especially for out-of-vocabulary and rare terms. CNNs can process candidate terms in parallel, and, therefore, are faster for our application than RNNs. RNNs, on the other hand, can encode longer contexts.

Finally, we compute the probability of selecting

[1] The dataset and code to run the experiments are available at `https://github.com/nyu-dl/QueryReformulator`.

[2] To deal with variable-length inputs in a mini-batch, we pad smaller ones with zeros on both ends so they end up as long as the largest sample in the mini-batch.

t_i as:

$$P(t_i|q_0) = \sigma(U^\mathsf{T} \tanh(W(\phi_a(v) \| \phi_b(e_i)) + b)), \tag{1}$$

where σ is the sigmoid function, $\|$ is the vector concatenation operation, $W \in \mathbb{R}^{d \times 2d}$ and $U \in \mathbb{R}^d$ are weights, and $b \in \mathbb{R}$ is a bias.

At test time, we define the set of terms used in the reformulated query as $T = \{t_i \mid P(t_i|q_0) > \epsilon\}$, where ϵ is a hyper-parameter. At training time, we sample the terms according to their probability distribution, $T = \{t_i \mid \alpha = 1 \wedge \alpha \sim P(t_i|q_0)\}$. We concatenate the terms in T to form a reformulated query q', which will then be used to retrieve a new set of documents D'.

2.2 Sequence Generation

One problem with the method previously described is that terms are selected independently. This may result in a reformulated query that contains duplicated terms since the same term can appear multiple times in the feedback documents. Another problem is that the reformulated query can be very long, resulting in a slow retrieval.

To solve these problems, we extend the model to sequentially generate a reformulated query, as proposed by Buck et al. (2017). We use a Recurrent Neural Network (RNN) that selects one term at a time from the pool of candidate terms and stops when a special token is selected. The advantage of this approach is that the model can remember the terms previously selected through its hidden state. It can, therefore, produce more concise queries.

We define the probability of selecting t_i as the k-th term of a reformulated query as:

$$P(t_i^k|q_0) \propto \exp(\phi_b(e_i)^\mathsf{T} h_k), \tag{2}$$

where h_k is the hidden state vector at the k-th step, computed as:

$$h_k = \tanh(W_a \phi_a(v) + W_b \phi_b(t^{k-1}) + W_h h_{k-1}), \tag{3}$$

where t^{k-1} is the term selected in the previous step and $W_a \in \mathbb{R}^{d \times d}$, $W_b \in \mathbb{R}^{d \times d}$, and $W_h \in \mathbb{R}^{d \times d}$ are weight matrices. In practice, we use an LSTM (Hochreiter and Schmidhuber, 1997) to encode the hidden state as this variant is known to perform better than a vanilla RNN.

We avoid normalizing over a large vocabulary by using only terms from the retrieved documents. This makes inference faster and training practical since learning to select words from the whole vocabulary might be too slow with reinforcement learning, although we leave this experiment for a future work.

2.3 Training

We train the proposed model using REINFORCE (Williams, 1992) algorithm. The per-example stochastic objective is defined as

$$C_a = (R - \bar{R}) \sum_{t \in T} - \log P(t|q_0), \tag{4}$$

where R is the reward and $\bar{R}$ is the baseline, computed by the value network as:

$$\bar{R} = \sigma(S^\mathsf{T} \tanh(V(\phi_a(v) \| \bar{e}) + b)), \tag{5}$$

where $\bar{e} = \frac{1}{N} \sum_{i=1}^{N} \phi_b(e_i)$, $N = |q_0 \cup D_0|$, $V \in \mathbb{R}^{d \times 2d}$ and $S \in \mathbb{R}^d$ are weights and $b \in \mathbb{R}$ is a bias. We train the value network to minimize

$$C_b = \alpha ||R - \bar{R}||^2, \tag{6}$$

where α is a small constant (e.g. 0.1) multiplied to the loss in order to stabilize learning. We conjecture that the stability is due to the slowly evolving value network which directly affects the learning of the policy. This effectively prevents the value network to fit extreme cases (unexpectedly high or low reward.)

We minimize C_a and C_b using stochastic gradient descent (SGD) with the gradient computed by backpropagation (Rumelhart et al., 1988). This allows the entire model to be trained end-to-end directly to optimize the retrieval performance.

Entropy Regularization We observed that the probability distribution in Eq.(1) became highly peaked in preliminary experiments. This phenomenon led to the trained model not being able to explore new terms that could lead to a better-reformulated query. We address this issue by regularizing the negative entropy of the probability distribution. We add the following regularization term to the original cost function in Eq. (4):

$$C_H = -\lambda \sum_{t \in q_0 \cup D_0} P(t|q_0) \log P(t|q_0), \tag{7}$$

where λ is a regularization coefficient.

3 Related Work

Query reformulation techniques are either based on a global method, which ignores a set of documents returned by the original query, or a local method, which adjusts a query relative to the documents that initially appear to match the query. In this work, we focus on local methods.

A popular instantiation of a local method is the *relevance model*, which incorporates pseudo-relevance feedback into a language model form (Lavrenko and Croft, 2001). The probability of adding a term to an expanded query is proportional to its probability of being generated by the language models obtained from the original query and the document the term occurs in. This framework has the advantage of not requiring *query/relevant documents* pairs as training data since inference is based on word co-occurrence statistics.

Unlike the relevance model, algorithms can be trained with supervised learning, as proposed by Cao et al. (2008). A training dataset is automatically created by labeling each candidate term as relevant or not based on their individual contribution to the retrieval performance. Then a binary classifier is trained to select expansion terms. In Section 4, we present a neural network-based implementation of this supervised approach.

A generalization of this supervised framework is to *iteratively* reformulate the query by selecting one candidate term at each retrieval step. This can be viewed as navigating a graph where the nodes represent queries and associated retrieved results and edges exist between nodes whose queries are simple reformulations of each other (Diaz, 2016). However, it can be slow to reformulate a query this way as the search engine must be queried for each newly added term. In our method, on the contrary, the search engine is queried with multiple new terms at once.

An alternative technique based on supervised learning is to learn a common latent representation of queries and relevant documents terms by using a *click-through* dataset (Sordoni et al., 2014). Neighboring document terms of a query in the latent space are selected to form an expanded query. Instead of using a *click-through* dataset, which is often proprietary, it is possible to use an alternative dataset consisting of anchor text/title pairs. In contrast, our approach does not require a dataset of paired queries as it learns term selection strategies via reinforcement learning.

Perhaps the closest work to ours is that by Narasimhan et al. (2016), in which a reinforcement learning based approach is used to reformulate queries iteratively. A key difference is that in their work the reformulation component uses domain-specific template queries. Our method, on the other hand, assumes open-domain queries.

4 Experiments

In this section we describe our experimental setup, including baselines against which we compare the proposed method, metrics, reward for RL-based models, datasets and implementation details.

4.1 Baseline Methods

Raw: The original query is given to a search engine without any modification. We evaluate two search engines in their default configuration: Lucene[3] (Raw-Lucene) and Google Search[4] (Raw-Google).

Pseudo Relevance Feedback (PRF-TFIDF): A query is expanded with terms from the documents retrieved by a search engine using the original query. In this work, the top-N TF-IDF terms from each of the top-K retrieved documents are added to the original query, where N and K are selected by a grid search on the validation data.

PRF-Relevance Model (PRF-RM): This is a popular relevance model for query expansion by Lavrenko and Croft (2001). The probability of using a term t in an expanded query is given by:

$$\begin{aligned} P(t|q_0) = (1-\lambda)P'(t|q_0) \\ + \lambda \sum_{d \in D_0} P(d)P(t|d)P(q_0|d), \end{aligned} \quad (8)$$

where $P(d)$ is the probability of retrieving the document d, assumed uniform over the set, $P(t|d)$ and $P(q_0|d)$ are the probabilities assigned by the language model obtained from d to t and q_0, respectively. $P'(t|q_0) = \frac{\text{tf}(t \in q)}{|q|}$, where $\text{tf}(t,d)$ is the term frequency of t in d. We set the interpolation parameter λ to 0.5, following Zhai and Lafferty (2001).

We use a Dirichlet smoothed language model (Zhai and Lafferty, 2001) to compute a language model from a document $d \in D_0$:

$$P(t|d) = \frac{\text{tf}(t,d) + uP(t|C)}{|d| + u}, \quad (9)$$

[3] https://lucene.apache.org/

[4] https://cse.google.com/cse/

where u is a scalar constant ($u = 1500$ in our experiments), and $P(t|C)$ is the probability of t occurring in the entire corpus C.

We use the N terms with the highest $P(t|q_0)$ in an expanded query, where N is a hyper-parameter.

Embeddings Similarity: Inspired by the methods proposed by Roy et al. (2016) and Kuzi et al. (2016), the top-N terms are selected based on the cosine similarity of their embeddings against the original query embedding. Candidate terms come from documents retrieved using the original query (PRF-Emb), or from a fixed vocabulary (Vocab-Emb). We use pretrained embeddings from Mikolov et al. (2013), and it contains 374,000 words.

4.2 Proposed Methods

Supervised Learning (SL): Here we detail a deep learning-based variant of the method proposed by Cao et al. (2008). It assumes that query terms contribute independently to the retrieval performance. We thus train a binary classifier to select a term if the retrieval performance increases beyond a preset threshold when that term is added to the original query. More specifically, we mark a term as relevant if $(R' - R)/R > 0.005$, where R and R' are the retrieval performances of the original query and the query expanded with the term, respectively.

We experiment with two variants of this method: one in which we use a convolutional network for both original query and candidate terms (SL-CNN), and the other in which we replace the convolutional network with a single hidden layer feed-forward neural network (SL-FF). In this variant, we average the output vectors of the neural network to obtain a fixed size representation of q_0.

Reinforcement Learning (RL): We use multiple variants of the proposed RL method. RL-CNN and RL-RNN are the models described in Section 2.1, in which the former uses CNNs to encode query and term features and the latter uses RNNs (more specifically, bidirectional LSTMs). RL-FF is the model in which term and query vectors are encoded by single hidden layer feed-forward neural networks. In the RL-RNN-SEQ model, we add the sequential generator described in Section 2.2 to the RL-RNN variant.

4.3 Datasets

We summarize in Table 1 the datasets.

TREC - Complex Answer Retrieval (TREC-CAR) This is a publicly available dataset automatically created from Wikipedia whose goal is to encourage the development of methods that respond to more complex queries with longer answers (Dietz and Ben, 2017). A query is the concatenation of an article title and one of its section titles. The ground-truth documents are the paragraphs within that section. For example, a query is "*Sea Turtle, Diet*" and the ground truth documents are the paragraphs in the section "*Diet*" of the "*Sea Turtle*" article. The corpus consists of all the English Wikipedia paragraphs, except the abstracts. The released dataset has five predefined folds, and we use the first three as the training set and the remaining two as validation and test sets, respectively.

Jeopardy This is a publicly available Q&A dataset introduced by Nogueira and Cho (2016). A query is a question from the *Jeopardy!* TV Show and the corresponding document is a Wikipedia article whose title is the answer. For example, a query is *"For the last eight years of his life, Galileo was under house arrest for espousing this mans theory"* and the answer is the Wikipedia article titled *"Nicolaus Copernicus"*. The corpus consists of all the articles in the English Wikipedia.

Microsoft Academic (MSA) This dataset consists of academic papers crawled from Microsoft Academic API.[5] The crawler started at the paper Silver et al. (2016) and traversed the graph of references until 500,000 papers were crawled. We then removed papers that had no reference within or whose abstract had less than 100 characters. We ended up with 480,000 papers.

A query is the title of a paper, and the ground-truth answer consists of the papers cited within. Each document in the corpus consists of its title and abstract.[6]

4.4 Metrics and Reward

Three metrics are used to evaluate performance:

Recall@K: Recall of the top-K retrieved documents:

$$\text{R@}K = \frac{|D_K \cap D^*|}{|D^*|}, \tag{10}$$

[5] https://www.microsoft.com/cognitive-services/en-us/academic-knowledge-api

[6] This was done to avoid a large computational overhead for indexing full papers.

			Queries			Relevant Docs/Query		Words/Doc	
Dataset	Corpus	Docs	Train	Valid	Test	Avg.	Std.	Avg.	Std.
TREC-CAR	Wikipedia Paragraphs	3.5M	585k	195k	195k	3.6	5.7	84	68
Jeopardy	Wikipedia Articles	5.9M	118K	10k	10k	1.0	0.0	462	990
MSA	Academic Papers	480k	270k	20k	20k	17.9	21.5	165	158

Table 1: Summary of the datasets.

	TREC-CAR			Jeopardy			MSA		
Method	R@40	P@10	MAP@40	R@40	P@10	MAP@40	R@40	P@10	MAP@40
Raw-Lucene	43.6	7.24	19.6	23.4	1.47	7.40	12.9	7.24	3.36
Raw-Google	-	-	-	30.1	1.92	7.71	-	-	-
PRF-TFIDF	44.3	7.31	19.9	29.9	1.91	7.65	13.2	7.27	3.50
PRF-RM	45.1	7.35	19.5	30.5	1.96	7.64	12.3	7.22	3.38
PRF-Emb	44.5	7.32	19.0	30.1	1.92	7.74	12.2	7.22	3.20
Vocab-Emb	44.2	7.30	19.1	29.4	1.87	7.80	12.0	7.21	3.21
SL-FF	44.1	7.29	19.7	30.8	1.95	7.70	13.2	7.28	3.88
SL-CNN	45.3	7.35	19.8	31.1	1.98	7.79	14.0	7.42	3.99
SL-Oracle	50.8	8.25	21.0	38.8	2.50	9.92	17.3	10.12	4.89
RL-FF	44.1	7.29	20.0	31.0	1.98	7.81	13.9	7.33	3.81
RL-CNN	47.3	7.45	20.3	33.4	**2.14**	8.02	14.9	7.63	4.30
RL-RNN	**47.9**	**7.52**	**20.6**	**33.7**	2.12	**8.07**	**15.1**	**7.68**	**4.35**
RL-RNN-SEQ	47.4	7.48	20.3	33.4	2.13	8.01	14.8	7.63	4.27
RL-Oracle	55.9	9.06	23.0	42.4	2.74	10.3	24.6	12.83	6.33

Table 2: Results on Test sets. We use R@40 as a reward to the RL-based models.

where D_K are the top-K retrieved documents and D^* are the relevant documents. Since one of the goals of query reformulation is to increase the proportion of relevant documents returned, recall is our main metric.

Precision@K: Precision of the top-K retrieved documents:

$$\text{P@}K = \frac{|D_K \cap D^*|}{|D_K|} \tag{11}$$

Precision captures the proportion of relevant documents among the returned ones. Despite not being the main goal of a reformulation method, improvements in precision are also expected with a good query reformulation method. Therefore, we include this metric.

Mean Average Precision: The average precision of the top-K retrieved documents is defined as:

$$\text{AP@}K = \frac{\sum_{k=1}^{K} \text{P@}k \times \text{rel}(k)}{|D^*|}, \tag{12}$$

where

$$\text{rel}(k) = \begin{cases} 1, & \text{if the k-th document is relevant;} \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The mean average precision of a set of queries Q is then:

$$\text{MAP@}K = \frac{1}{|Q|} \sum_{q \in Q} \text{AP@}K_q, \tag{14}$$

where AP@K_q is the average precision at K for a query q. This metric values the position of a relevant document in a returned list and is, therefore, complementary to precision and recall.

Reward We use R@K as a reward when training the proposed RL-based models as this metric has shown to be effective in improving the other metrics as well.

SL-Oracle In addition to the baseline methods and proposed reinforcement learning approach, we report two oracle performance bounds. The first oracle is a supervised learning oracle (SL-Oracle). It is a classifier that perfectly selects terms that will increase performance according to the procedure described in Section 4.2. This measure serves as an upper-bound for the supervised methods. Notice that this heuristic assumes that each term contributes independently from all the other terms to the retrieval performance. There may be, however, other ways to explore the dependency of terms that would lead to a higher performance.

	TREC-CAR	Jeopardy	MSA
SL-Oracle	13%	5%	11%
RL-Oracle	29%	27%	31%

Table 3: Percentage of relevant terms over all the candidate terms according to SL- and RL-Oracle.

RL-Oracle Second, we introduce a reinforcement learning oracle (RL-Oracle) which estimates a conservative upper-bound performance for the RL models. Unlike the SL-Oracle, it does not assume that each term contributes independently to the retrieval performance. It works as follows: first, the *validation* or *test* set is divided into N small subsets $\{A_i\}_{i=1}^N$ (each with 100 examples, for instance). An RL model is trained on each subset A_i until it overfits, that is, until the reward R_i^* stops increasing or an early stop mechanism ends training.[7] Finally, we compute the oracle performance R^* as the average reward over all the subsets: $R^* = \frac{1}{N}\sum_{i=1}^N R_i^*$.

This upper bound by the RL-Oracle is, however, conservative since there might exist better reformulation strategies that the RL model was not able to discover.

4.5 Implementation Details

Search engine We use Lucene and BM25 as the search engine and the ranking function, respectively, for all PRF, SL and RL methods. For Raw-Google, we restrict the search to the *wikipedia.org* domain when evaluating its performance on the Jeopardy dataset. We could not apply the same restriction to the two other datasets as Google does not index Wikipedia paragraphs, and as it is not trivial to match papers from MS Academic to the ones returned by Google Search.

Candidate terms We use Wikipedia articles as a source for candidate terms since it is a well curated, clean corpus, with diverse topics.

At training and test times of SL methods, and at test time of RL methods, the candidate terms are from the first M words of the top-K Wikipedia articles retrieved. We select M and K using grid search on the validation set over $\{50, 100, 200, 300\}$ and $\{1, 3, 5, 7\}$, respectively. The best values are $M = 300$ and $K = 7$. These correspond to the maximum number of terms we could fit in a single GPU.

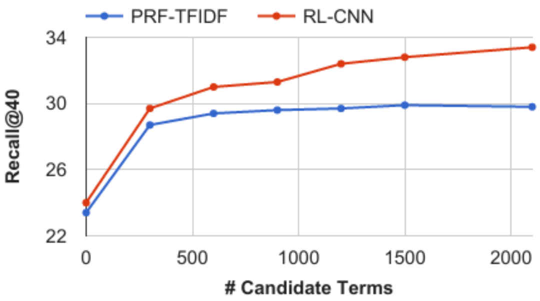


Figure 3: Our RL-based model continues to improve recall as more candidate terms are added, whereas a classical PRF method saturates.

At training time of an RL model, we use only *one* document uniformly sampled from the top-K retrieved ones as a source for candidate terms, as this leads to a faster learning.

For the PRF methods, the top-M terms according to a relevance metric (i.e., TF-IDF for PRF-TFIDF, cosine similarity for PRF-Emb, and conditional probability for PRF-RM) from each of the top-K retrieved documents are added to the original query. We select M and K using grid search over $\{10, 50, 100, 200, 300, 500\}$ and $\{1, 3, 5, 9, 11\}$, respectively. The best values are $M = 300$ and $K = 9$.

Multiple Reformulation Rounds Although our framework supports multiple rounds of search and reformulation, we did not find any significant improvement in reformulating a query more than once. Therefore, the numbers reported in the results section were all obtained from models running two rounds of search and reformulation.

Neural Network Setup For SL-CNN and RL-CNN variants, we use a 2-layer convolutional network for the original query. Each layer has a window size of 3 and 256 filters. We use a 2-layer convolutional network for candidate terms with window sizes of 9 and 3, respectively, and 256 filters in each layer. We set the dimension d of the weight matrices W, S, U, and V to 256. For the optimizer, we use ADAM (Kingma and Ba, 2014) with $\alpha = 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We set the entropy regularization coefficient λ to 10^{-3}.

For RL-RNN and RL-RNN-SEQ, we use a 2-layer bidirectional LSTM with 256 hidden units in each layer. We clip the gradients to unit norm. For RL-RNN-SEQ, we set the maximum possible

[7] The subset should be small enough, or the model should be large enough so it can overfit.

number of generated terms to 50 and we use beam search of size four at test time.

We fix the dictionary of pre-trained word embeddings during training, except the vector for out-of-vocabulary words. We found that this led to faster convergence and observed no difference in the overall performance when compared to learning embeddings during training.

5 Results and Discussion

Table 2 shows the main result. As expected, reformulation based methods work better than using the original query alone. Supervised methods (SL-FF and SL-CNN) have in general a better performance than unsupervised ones (PRF-TFIDF, PRF-RM, PRF-Emb, and Emb-Vocab), but perform worse than RL-based models (RL-FF, RL-CNN, RL-RNN, and RL-RNN-SEQ).

RL-RNN-SEQ performs slightly worse than RL-RNN but produces queries that are three times shorter, on average (15 vs 47 words). Thus, RL-RNN-SEQ is faster in retrieving documents and therefore might be a better candidate for a production implementation.

The performance gap between the oracle and best performing method (Table 2, RL-Oracle vs. RL-RNN) suggests that there is a large room for improvement. The cause for this gap is unknown but we suspect, for instance, an inherent difficulty in learning a good selection strategy and the partial observability from using a black box search engine.

5.1 Relevant Terms per Document

The proportion of relevant terms selected by the SL- and RL-Oracles over the total number of candidate terms (Table 3) indicates that only a small subset of terms are useful for the reformulation. Thus, we may conclude that the proposed method was able to learn an efficient term selection strategy in an environment where relevant terms are infrequent.

5.2 Scalability: Number of Terms vs Recall

Fig. 3 shows the improvement in recall as more candidate terms are provided to a reformulation method. The RL-based model benefits from more candidate terms, whereas the classical PRF method quickly saturates. In our experiments, the best performing RL-based model uses the maximum number of candidate terms that we could fit on a single GPU. We, therefore, expect further improvements with more computational resources.

Query	Top-3 Retrieved Documents
(Original) *The Cross Entropy Method for Fast Policy Search*	*-The Cross Entropy Method for Network Reliability Estim.* ***-Robot Weightlifting by Direct Policy Search*** *-Off-policy Policy Search*
(Reformulated) *Cross Entropy Fast Policy Reinforcement Learning policies global search optimization biased*	***-Near Optimal Reinforcement Learning in Polynom. Time*** *-The Cross Entropy Method for Network Reliability Estim.* ***-Robot Weightlifting by Direct Policy Search***
(Original) *Daikon Cultivation*	“*...many types of pickles are made with daikon, includ...*” “***Certain varieties of daikon can be grown as a winter...***” “*In Chinese cuisine, turnip cake and chai tow kway...*”
(Reformulated) *Daikon Cultivation root seed grow fast-growing Chinese leaves*	“*...many types of pickles are “made with daikon, includ...*” “***Certain varieties of daikon can be grown as a winter...***” “***The Chinese and Indian varieties tolerate higher....***”

Table 4: Top-3 retrieved documents using the original query and a query reformulated by our RL-CNN model. In the first example, we only show the titles of the retrieved MSA papers. In the second example, we only show some words of the retrieved TREC-CAR paragraphs. **Bold** corresponds to ground-truth documents.

5.3 Qualitative Analysis

We show two examples of queries and the probabilities of each candidate term of being selected by the RL-CNN model in Fig. 4.

Trained on	Selected Terms
TREC-CAR	*serves american national Winsted accreditation*
Jeopardy	*Tunxis Quinebaug Winsted NCCC*
MSA	*hospital library arts center cancer center summer programs*

Table 5: Given the query *“Northwestern Connecticut Community College”*, models trained on different tasks choose different terms.

Notice that terms that are more related to the query have higher probabilities, although common words such as ”*the*” are also selected. This is a consequence of our choice of a reward that does

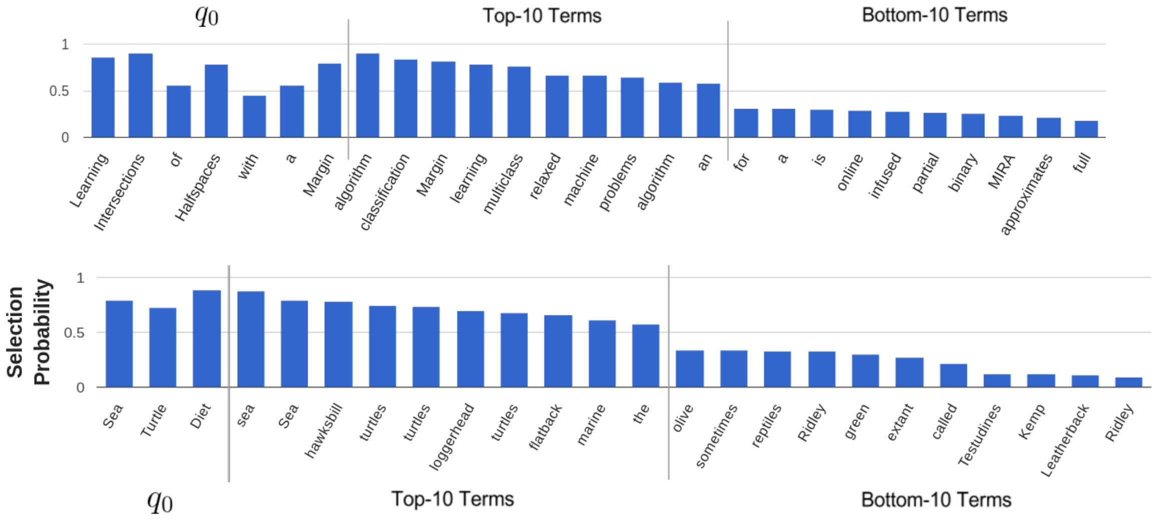


Figure 4: Probabilities assigned by the RL-CNN to candidate terms of two sample queries: "*Learning Intersections of Halfspaces with a Margin*" (top) and "*Sea Turtle Diet*" (bottom). We show the original query terms and the top-10 and bottom-10 document terms with respect to their probabilities.

not penalize the selection of neutral terms.

In Table 4 we show an original and reformulated query examples extracted from the MS Academic and TREC-CAR datasets, and their top-3 retrieved documents. Notice that the reformulated query retrieves more relevant documents than the original one. As we conjectured earlier, we see that a search engine tends to return a document simply with the largest overlap in the text, necessitating the reformulation of a query to retrieve semantically relevant documents.

Same query, different tasks We compare in Table 5 the reformulation of a sample query made by models trained on different datasets. The model trained on TREC-CAR selects terms that are similar to the ones in the original query, such as "*serves*" and "*accreditation*". These selections are expected for this task since similar terms can be effective in retrieving similar paragraphs. On the other hand, the model trained on Jeopardy prefers to select proper nouns, such as "*Tunxis*", as these have a higher chance of being an answer to the question. The model trained on MSA selects terms that cover different aspects of the entity being queried, such as "*arts center*" and "*library*", since retrieving a diverse set of documents is necessary for the task the of citation recommendation.

5.4 Training and Inference Times

Our best model, RL-RNN, takes 8-10 days to train on a single K80 GPU. At inference time, it takes approximately one second to reformulate a batch of 64 queries. Approximately 40% of this time is to retrieve documents from the search engine.

6 Conclusion

We introduced a reinforcement learning framework for task-oriented automatic query reformulation. An appealing aspect of this framework is that an agent can be trained to use a search engine for a specific task. The empirical evaluation has confirmed that the proposed approach outperforms strong baselines in the three separate tasks. The analysis based on two oracle approaches has revealed that there is a meaningful room for further development. In the future, more research is necessary in the directions of (1) iterative reformulation under the proposed framework, (2) using information from modalities other than text, and (3) better reinforcement learning algorithms for a partially-observable environment.

Acknowledgements

RN is funded by Coordenao de Aperfeioamento de Pessoal de Nvel Superior (CAPES). KC thanks support by Facebook, Google and NVIDIA. This work was partly funded by the Defense Advanced Research Projects Agency (DARPA) D3M program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. 2017. Ask the right questions: Active question reformulation with reinforcement learning. *arXiv preprint arXiv:1705.07830*.

Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. 2008. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 243–250. ACM.

Fernando Diaz. 2016. Pseudo-query reformulation. In *European Conference on Information Retrieval*, pages 521–532. Springer.

Laura Dietz and Gamari Ben. 2017. Trec car: A data set for complex answer retrieval. *http://trec-car.cs.unh.edu*.

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

Jeff Huang and Efthimis N Efthimiadis. 2009. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 77–86. ACM.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Saar Kuzi, Anna Shtok, and Oren Kurland. 2016. Query expansion using word embeddings. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1929–1932. ACM.

Victor Lavrenko and W Bruce Croft. 2001. Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127. ACM.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. *arXiv preprint arXiv:1603.07954*.

Rodrigo Nogueira and Kyunghyun Cho. 2016. End-to-end goal-driven web navigation. In *Advances in Neural Information Processing Systems*, pages 1903–1911.

Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. 2016. Using word embeddings for automatic query expansion. *arXiv preprint arXiv:1606.07608*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Alessandro Sordoni, Yoshua Bengio, and Jian-Yun Nie. 2014. Learning concept embeddings for query expansion by quantum entropy minimization. In *AAAI*, pages 1586–1592.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Jinxi Xu and W Bruce Croft. 1996. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 4–11. ACM.

Chengxiang Zhai and John Lafferty. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342. ACM.

Sentence Simplification with Deep Reinforcement Learning

Xingxing Zhang and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
x.zhang@ed.ac.uk,mlap@inf.ed.ac.uk

Abstract

Sentence simplification aims to make sentences easier to read and understand. Most recent approaches draw on insights from machine translation to learn simplification rewrites from monolingual corpora of complex and simple sentences. We address the simplification problem with an encoder-decoder model coupled with a deep reinforcement learning framework. Our model, which we call DRESS (as shorthand for **D**eep **RE**inforcement **S**entence **S**implification), explores the space of possible simplifications while learning to optimize a reward function that encourages outputs which are simple, fluent, and preserve the meaning of the input. Experiments on three datasets demonstrate that our model outperforms competitive simplification systems.[1]

1 Introduction

The main goal of *sentence simplification* is to reduce the linguistic complexity of text, while still retaining its original information and meaning. The simplification task has been the subject of several modeling efforts in recent years due to its relevance for NLP applications and individuals alike (Siddharthan, 2014; Shardlow, 2014). For instance, a simplification component could be used as a preprocessing step to improve the performance of parsers (Chandrasekar et al., 1996), summarizers (Beigman Klebanov et al., 2004), and semantic role labelers (Vickrey and Koller, 2008; Woodsend and Lapata, 2014). Automatic simplification would also benefit people with low-literacy skills (Watanabe et al., 2009), such as children and non-native speakers as well as individuals with autism (Evans et al., 2014), aphasia (Carroll et al., 1999), or dyslexia (Rello et al., 2013).

The most prevalent rewrite operations which give rise to simplified text include substituting rare words with more common words or phrases, rendering syntactically complex structures simpler, and deleting elements of the original text (Siddharthan, 2014). Earlier work focused on individual aspects of the simplification problem. For example, several systems performed syntactic simplification only, using rules aimed at sentence splitting (Carroll et al., 1999; Chandrasekar et al., 1996; Vickrey and Koller, 2008; Siddharthan, 2004) while others turned to lexical simplification by substituting difficult words with more common WordNet synonyms or paraphrases (Devlin, 1999; Inui et al., 2003; Kaji et al., 2002).

Recent approaches view the simplification process more holistically as a monolingual text-to-text generation task borrowing ideas from statistical machine translation. Simplification rewrites are learned automatically from examples of complex-simple sentences extracted from online resources such as the ordinary and simple English Wikipedia. For example, Zhu et al. (2010) draw inspiration from syntax-based translation and propose a model similar to Yamada and Knight (2001) which additionally performs simplification-specific rewrite operations (e.g., sentence splitting). Woodsend and Lapata (2011) formulate simplification in the framework of Quasi-synchronous grammar (Smith and Eisner, 2006) and use integer linear programming to score the candidate translations/simplifications. Wubben et al. (2012) propose a two-stage model: initially, a standard phrase-based machine translation (PBMT) model is trained on complex-simple sentence pairs. During inference, the K-best outputs of the PBMT model are reranked according

[1] Our code and data are publicly available at https://github.com/XingxingZhang/dress.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 584–594
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

to their dis-similarity to the (complex) input sentence. The hybrid model developed in Narayan and Gardent (2014) also operates in two phases. Initially, a probabilistic model performs sentence splitting and deletion operations over discourse representation structures assigned by Boxer (Curran et al., 2007). The resulting sentences are further simplified by a model similar to Wubben et al. (2012). Xu et al. (2016) train a syntax-based machine translation model on a large scale paraphrase dataset (Ganitkevitch et al., 2013) using simplification-specific objective functions and features to encourage simpler output.

In this paper we propose a simplification model which draws on insights from neural machine translation (Bahdanau et al., 2015; Sutskever et al., 2014). Central to this approach is an encoder-decoder architecture implemented by recurrent neural networks. The encoder reads the source sequence into a list of continuous-space representations from which the decoder generates the target sequence. Although our model uses the encoder-decoder architecture as its backbone, it must also meet constraints imposed by the simplification task itself, i.e., the predicted output must be simpler, preserve the meaning of the input, and grammatical. To incorporate this knowledge, the model is trained in a reinforcement learning framework (Williams, 1992): it explores the space of possible simplifications while learning to maximize an expected reward function that encourages outputs which meet simplification-specific constraints. Reinforcement learning has been previously applied to extractive summarization (Ryang and Abekawa, 2012), information extraction (Narasimhan et al., 2016), dialogue generation (Li et al., 2016), machine translation, and image caption generation (Ranzato et al., 2016).

We evaluate our system on three publicly available datasets collated automatically from Wikipedia (Zhu et al., 2010; Woodsend and Lapata, 2011) and human-authored news articles (Xu et al., 2015b). We experimentally show that the reinforcement learning framework is the key to successful generation of simplified text bringing significant improvements over strong simplification models across datasets.

2 Neural Encoder-Decoder Model

We will first define a basic encoder-decoder model for sentence simplification and then explain how to embed it in a reinforcement learning framework. Given a (complex) *source* sentence $X = (x_1, x_2, \ldots, x_{|X|})$, our model learns to predict its simplified *target* $Y = (y_1, y_2, \ldots, y_{|Y|})$. Inferring the target Y given the source X is a typical sequence to sequence learning problem, which can be modeled with attention-based encoder-decoder models (Bahdanau et al., 2015; Luong et al., 2015). Sentence simplification is slightly different from related sequence transduction tasks (e.g., compression) in that it can involve splitting operations. For example, a long source sentence (*In 1883, Faur married Marie Fremiet, with whom he had two sons.*) can be simplified as two sentences (*In 1883, Faur married Marie Fremiet. They had two sons.*). Nevertheless, we still view the target as a sequence, i.e., two or more sequences concatenated with full stops.

The encoder-decoder model has two parts (see left hand side in Figure 1). The *encoder* transforms the source sentence X into a sequence of hidden states $(\mathbf{h}_1^S, \mathbf{h}_2^S, \ldots, \mathbf{h}_{|X|}^S)$ with a Long Short-Term Memory Network (LSTM; Hochreiter and Schmidhuber 1997), while the *decoder* uses another LSTM to generate one word y_{t+1} at a time in the simplified target Y. Generation is conditioned on all previously generated words $y_{1:t}$ and a dynamically created context vector $\mathbf{c}_t$, which encodes the source sentence:

$$P(Y|X) = \prod_{t=1}^{|Y|} P(y_t|y_{1:t-1}, X) \tag{1}$$

$$P(y_{t+1}|y_{1:t}, X) = \text{softmax}(g(\mathbf{h}_t^T, \mathbf{c}_t)) \tag{2}$$

where $g(\cdot)$ is a one-hidden-layer neural network with the following parametrization:

$$g(\mathbf{h}_t^T, \mathbf{c}_t) = \mathbf{W}_o \tanh(\mathbf{U}_h \mathbf{h}_t^T + \mathbf{W}_h \mathbf{c}_t) \tag{3}$$

where $\mathbf{W}_o \in \mathbb{R}^{|V| \times d}$, $\mathbf{U}_h \in \mathbb{R}^{d \times d}$, and $\mathbf{W}_h \in \mathbb{R}^{d \times d}$; $|V|$ is the output vocabulary size and d the hidden unit size. $\mathbf{h}_t^T$ is the hidden state of the decoder LSTM which summarizes $y_{1:t}$, i.e., what has been generated so far:

$$\mathbf{h}_t^T = \text{LSTM}(y_t, \mathbf{h}_{t-1}^T) \tag{4}$$

The dynamic context vector $\mathbf{c}_t$ is the weighted sum of the hidden states of the source sentence:

$$\mathbf{c}_t = \sum_{i=1}^{|X|} \alpha_{ti} \mathbf{h}_i^S \tag{5}$$

whose weights α_{ti} are determined by an *attention* mechanism:

$$\alpha_{ti} = \frac{\exp(\mathbf{h}_t^T \cdot \mathbf{h}_i^S)}{\sum_i \exp(\mathbf{h}_t^T \cdot \mathbf{h}_i^S)} \quad (6)$$

where $\cdot$ is the dot product between two vectors. We use the dot product here mainly for efficiency reasons; alternative ways to compute attention scores have been proposed in the literature and we refer the interested reader to Luong et al. (2015). The model sketched above is usually trained by minimizing the negative log-likelihood of the training source-target pairs.

3 Reinforcement Learning for Sentence Simplification

In this section we present DRESS, our **D**eep **RE**inforcement **S**entence **S**implification model. Despite successful application in numerous sequence transduction tasks (Jean et al., 2015; Chopra et al., 2016; Xu et al., 2015a), a vanilla encoder-decoder model is not ideal for sentence simplification. Although a number of rewrite operations (e.g., copying, deletion, substitution, word reordering) can be used to simplify text, copying is by far the most common. We empirically found that 73% of the target words are copied from the source in the Newsela dataset. This number further increases to 83% when considering Wikipedia-based datasets (we provide details on these datasets in Section 5). As a result, a generic encoder-decoder model learns to copy all too well at the expense of other rewrite operations, often parroting back the source or making only a few trivial changes.

To encourage a wider variety of rewrite operations while remaining fluent and faithful to the meaning of the source, we employ a reinforcement learning framework (see Figure 1). We view the encoder-decoder model as an agent which first reads the source sentence X; then at each step, it takes an action $\hat{y}_t \in V$ (where V is the output vocabulary) according to a policy $P_{RL}(\hat{y}_t|\hat{y}_{1:t-1}, X)$ (see Equation (2)). The agent continues to take actions until it produces an **E**nd **O**f **S**entence (EOS) token yielding the action sequence $\hat{Y} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|\hat{Y}|})$, which is also the simplified output of our model. A reward r is then received and the REINFORCE algorithm (Williams, 1992) is used to update the agent. In the following, we first introduce our reward and then present the details of the REINFORCE algorithm.

3.1 Reward

The reward $r(\hat{Y})$ for system output $\hat{Y}$ is the weighted sum of the three components aimed at capturing key aspects of the target output, namely simplicity, relevance, and fluency:

$$r(\hat{Y}) = \lambda^S r^S + \lambda^R r^R + \lambda^F r^F \quad (7)$$

where $\lambda^S, \lambda^R, \lambda^F \in [0, 1]$; $r(\hat{Y})$ is a shorthand for $r(X, Y, \hat{Y})$ where X is the source, Y the reference (or target), and $\hat{Y}$ the system output. r^S, r^R, and r^F are shorthands for simplicity $r^S(X, Y, \hat{Y})$, relevance $r^R(X, \hat{Y})$, and fluency $r^F(\hat{Y})$. We provide details for each reward summand below.

Simplicity To encourage the model to apply a wide range of simplification operations, we use SARI (Xu et al., 2016), a recently proposed metric which compares **S**ystem output **A**gainst **R**eferences and against the **I**nput sentence. SARI is the arithmetic average of n-gram precision and recall of three rewrite operations: addition, copying, and deletion. It rewards addition operations where system output was not in the input but occurred in the references. Analogously, it rewards words retained/deleted in both the system output and the references. In experimental evaluation Xu et al. (2016) demonstrate that SARI correlates well with human judgments of simplicity, whilst correctly rewarding systems that both make changes and simplify the input.

One caveat with using SARI as a reward is the fact that it relies on the availability of multiple references which are rare for sentence simplification. Xu et al. (2016) provide eight references for 2,350 sentences, but these are primarily for system tuning and evaluation rather than training. The majority of existing simplification datasets (see Section 5 for details) have a single reference for each source sentence. Moreover, they are unavoidably noisy as they are mostly constructed automatically, e.g., by aligning sentences from the ordinary and simple English Wikipedias. When relying solely on a single reference, SARI will try to reward accidental n-grams that should never have occurred in it. To countenance the effect of noise, we apply SARI$(X, \hat{Y}, Y)$ in the expected direction, with X as the source, $\hat{Y}$ the system output, and Y the reference as well as in the reverse direction with Y as the system output and $\hat{Y}$ as the reference. Assuming our system can produce reasonably good simplifications, by swapping the output

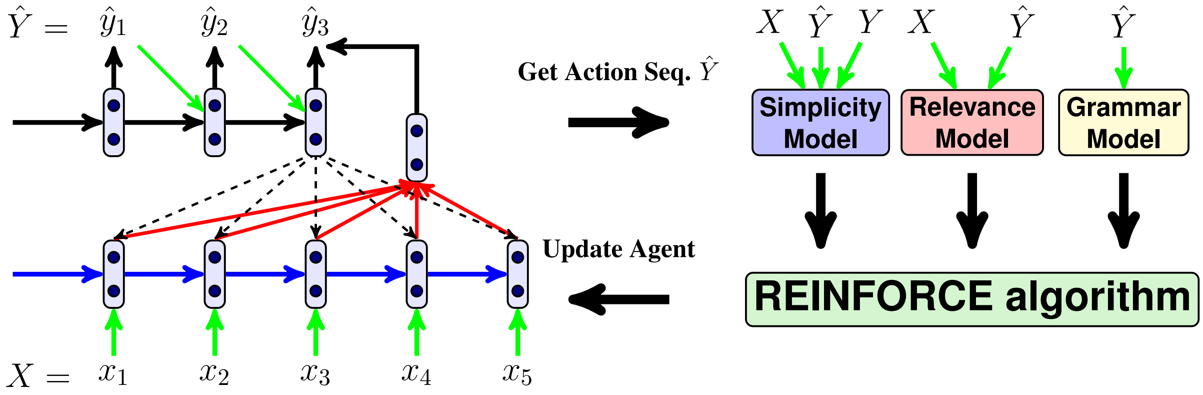

$\hat{Y} = \hat{y}_1 \; \hat{y}_2 \; \hat{y}_3$
Get Action Seq. $\hat{Y}$
$X \; \hat{Y} \; Y \; X \; \hat{Y} \; \hat{Y}$
Simplicity Model
Relevance Model
Grammar Model
Update Agent
REINFORCE algorithm
$X = x_1 \; x_2 \; x_3 \; x_4 \; x_5$

Figure 1: Deep reinforcement learning simplification model. X is the complex sentence, Y the reference (simple) sentence and $\hat{Y}$ the action sequence (simplification) produced by the encoder-decoder model.

and the reference, reverse SARI can be used to estimate how good a reference is with respect to the system output. Our first reward is therefore the weighted sum of SARI and reverse SARI:

$$r^S = \beta\, \textsc{Sari}(X, \hat{Y}, Y) + (1-\beta)\, \textsc{Sari}(X, Y, \hat{Y}) \quad (8)$$

Relevance While the simplicity-based reward r^S tries to encourage the model to make changes, the relevance reward r^R ensures that the generated sentences preserve the meaning of the source. We use an LSTM sentence encoder to convert the source X and the predicted target $\hat{Y}$ into two vectors $\mathbf{q}_X$ and $\mathbf{q}_{\hat{Y}}$. The relevance reward r^R is simply the cosine similarity between these two vectors:

$$r^R = \cos(\mathbf{q}_X, \mathbf{q}_{\hat{Y}}) = \frac{\mathbf{q}_X \cdot \mathbf{q}_{\hat{Y}}}{||\mathbf{q}_X||\;||\mathbf{q}_{\hat{Y}}||} \quad (9)$$

We use a sequence auto-encoder (SAE; Dai and Le 2015) to train the LSTM sentence encoder on both the complex and simple sentences. Specifically, the SAE uses sentence $X = (x_1, \ldots, x_{|X|})$ to infer itself via an encoder-decoder model (without an attention mechanism). Firstly, an encoder LSTM converts X into a sequence of hidden states $(\mathbf{h}_1, \ldots, \mathbf{h}_{|X|})$. Then, we use $\mathbf{h}_{|X|}$ to initialize the hidden state of the decoder LSTM and recover/generate X one word at a time.

Fluency Xu et al. (2016) observe that SARI correlates less with fluency compared to other metrics such as BLEU (Papineni et al., 2002). The fluency reward r^F models the well-formedness of the generated sentences explicitly. It is the normalized sentence probability assigned by an LSTM language model trained on simple sentences:

$$r^F = \exp\left(\frac{1}{|\hat{Y}|}\sum_{i=1}^{|\hat{Y}|} \log P_{LM}(\hat{y}_i|\hat{y}_{0:i-1})\right) \quad (10)$$

We take the exponential of $\hat{Y}$'s perplexity to ensure that $r^F \in [0, 1]$ as is the case with r^S and r^R.

3.2 The REINFORCE Algorithm

The goal of the REINFORCE algorithm is to find an agent that maximizes the expected reward. The training loss for one sequence is its negative expected reward:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(\hat{y}_1,\ldots,\hat{y}_{|\hat{Y}|})\sim P_{RL}(\cdot|X)}[r(\hat{y}_1, \ldots, \hat{y}_{|\hat{Y}|})]$$

where P_{RL} is our policy, i.e., the distribution produced by the encoder-decoder model (see Equation(2)) and $r(\cdot)$ is the reward function of an action sequence $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_{|\hat{Y}|})$, i.e., a generated simplification. Unfortunately, computing the expectation term is prohibitive, since there is an infinite number of possible action sequences. In practice, we approximate this expectation with a single sample from the distribution of $P_{LR}(\cdot|X)$. We refer to Williams (1992) for the full derivation of the gradients. The gradient of $\mathcal{L}(\theta)$ is:

$$\nabla\mathcal{L}(\theta) \approx \\ \textstyle\sum_{t=1}^{|\hat{Y}|} \nabla \log P_{RL}(\hat{y}_t|\hat{y}_{1:t-1}, X)[r(\hat{y}_{1:|\hat{Y}|}) - b_t]$$

To reduce the variance of gradients, we also introduce a baseline linear regression model b_t to estimate the expected future reward at time t (Ranzato et al., 2016). b_t takes the concatenation of $\mathbf{h}_t^T$ and $\mathbf{c}_t$ as input and outputs a real value as the expected reward. The parameters of the regressor are trained by minimizing mean squared error. We do not back-propagate this error to $\mathbf{h}_t^T$ or $\mathbf{c}_t$ during training (Ranzato et al., 2016).

3.3 Learning

Presented in its original form, the REINFORCE algorithm starts learning with a random policy. This assumption can make model training challenging for generation tasks like ours with large vocabularies (i.e., action spaces). We address this issue by pre-training our agent (i.e., the encoder-decoder model) with a negative log-likelihood objective (see Section 2), making sure it can produce reasonable simplifications, thereby starting off with a policy which is better than random. We follow prior work (Ranzato et al., 2016) in adopting a curriculum learning strategy. In the beginning of training, we give little freedom to our agent allowing it to predict the last few words for each target sentence. For every target sequence, we use negative log-likelihood to train the first L (initially, $L = 24$) tokens and apply the reinforcement learning algorithm to the $(L + 1)$th tokens onwards. Every two epochs, we set $L = L - 3$ and the training terminates when L is 0.

4 Lexical Simplification

Lexical substitution, the replacement of complex words with simpler alternatives, is an integral part of sentence simplification (Specia et al., 2012). The model presented so far learns lexical substitution and other rewrite operations *jointly*. In some cases, words are predicted because they seem natural in the their context, but are poor substitutes for the content of the complex sentence. To countenance this, we learn lexical simplifications *explicitly* and integrate them with our reinforcement learning-based model.

We use an *pre-trained* encoder-decoder model (which is trained on a parallel corpus of complex and simple sentences) to obtain probabilistic word alignments, aka attention scores (see α_t in Equation (6)). Let $X = (x_1, x_2, \ldots, x_{|X|})$ denote a source sentence and $Y = (y_1, y_2, \ldots, y_{|Y|})$ a target sentence. We convert X into $|X|$ hidden states $(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{|X|})$ with an LSTM. Note that $\mathbf{v}_t \in \mathbb{R}^{d \times 1}$ corresponds to the context dependent representation of x_t. Let α_t denote the alignment scores $\alpha_{t1}, \alpha_{t2}, \ldots, \alpha_{t|X|}$. The lexical simplification probability of y_t given the source sentence and the alignment scores is:

$$P_{LS}(y_t|X, \alpha_t) = \text{softmax}(\mathbf{W}_l\, \mathbf{s}_t) \tag{11}$$

where $\mathbf{W}_l \in \mathbb{R}^{|V| \times d}$ and $\mathbf{s}_t$ represents the source:

$$\mathbf{s}_t = \sum_{i=1}^{|X|} \alpha_{ti} \mathbf{v}_i \tag{12}$$

The lexical simplification model on its own encourages lexical substitutions, without taking into account what has been generated so far (i.e., $y_{1:t-1}$) and as a result fluency could be compromised. A straightforward solution is to integrate lexical simplification with our reinforcement learning trained model (Section 3) using linear interpolation, where $\eta \in [0, 1]$:

$$\begin{aligned} P(y_t|y_{1:t-1}, X) = (1 - \eta)\, P_{RL}(y_t|y_{1:t-1}, X) \\ + \eta\, P_{LS}(y_t|X, \alpha_t) \end{aligned} \tag{13}$$

5 Experimental Setup

In this section we present our experimental setup for assessing the performance of the simplification model described above. We give details on our datasets, model training, evaluation protocol, and the systems used for comparison.

Datasets We conducted experiments on three simplification datasets. *WikiSmall* (Zhu et al., 2010) is a parallel corpus which has been extensively used as a benchmark for evaluating text simplification systems (Wubben et al., 2012; Woodsend and Lapata, 2011; Narayan and Gardent, 2014; Zhu et al., 2010). It contains automatically aligned complex and simple sentences from the ordinary and simple English Wikipedias. The test set consists of 100 complex-simple sentence pairs. The training set contains 89,042 sentence pairs (after removing duplicates and test sentences). We randomly sampled 205 pairs for development and used the remaining sentences for training.

We also constructed *WikiLarge*, a larger Wikipedia corpus by combining previously created simplification corpora. Specifically, we aggregated the aligned sentence pairs in Kauchak (2013), the aligned and revision sentence pairs in Woodsend and Lapata (2011), and Zhu's (2010) WikiSmall dataset described above. We used the development and test sets created in Xu et al. (2016). These are complex sentences taken from WikiSmall paired with simplifications provided by Amazon Mechanical Turk workers. The dataset contains 8 (reference) simplifications for 2,359 sentences partitioned into 2,000 for development and 359 for testing. After removing duplicates and sentences in development and test sets, the resulting training set contains 296,402 sentence pairs.

Our third dataset is *Newsela*, a corpus collated by Xu et al. (2015b) who argue that Wikipedia-based resources are suboptimal due to the automatic sentence alignment which unavoidably introduces errors, and their uniform writing style which leads to systems that generalize poorly. Newsela[2] consists of 1,130 news articles, each rewritten four times by professional editors for children at different grade levels (0 is the most complex level and 4 is simplest). Xu et al. (2015b) provide multiple aligned complex-simple pairs within each article. We removed sentence pairs corresponding to levels 0–1, 1–2, and 2–3, since they were too similar to each other. The first 1,070 documents were used for training (94,208 sentence pairs), the next 30 documents for development (1,129 sentence pairs) and the last 30 documents for testing (1,076 sentence pairs).[3] We are not aware of any published results on this dataset.

Training Details We trained our models on an Nvidia GPU card. We used the same hyper-parameters across datasets. We first trained an encoder-decoder model, and then performed reinforcement learning training (Section 3), and trained the lexical simplification model (Section 4). Encoder-decoder parameters were uniformly initialized to $[-0.1, 0.1]$. We used Adam (Kingma and Ba, 2014) to optimize the model with learning rate 0.001; the first momentum coefficient was set to 0.9 and the second momentum coefficient to 0.999. The gradient was rescaled when the norm exceeded 5 (Pascanu et al., 2013). Both encoder and decoder LSTMs have two layers with 256 hidden neurons in each layer. We regularized all LSTMs with a dropout rate of 0.2 (Zaremba et al., 2014). We initialized the encoder and decoder word embedding matrices with 300 dimensional Glove vectors (Pennington et al., 2014).

During reinforcement training, we used plain stochastic gradient descent with a learning rate of 0.01. We set $\beta = 0.1$, $\lambda_S = 1$, $\lambda_R = 0.25$ and $\lambda_F = 0.5$.[4] Training details for the lexical simplification model are identical to the encoder-decoder model except that word embedding matrices were randomly initialized. The weight of the lexical simplification model was set to $\eta = 0.1$.

To reduce vocabulary size, named entities were tagged with the Stanford CoreNLP (Manning et al., 2014) and anonymized with a NE@N token, where NE $\in \{\text{PER}, \text{LOC}, \text{ORG}, \text{MISC}\}$ and N indicates NE@N is the N-th distinct NE typed entity. For example, "John and Bob are ..." becomes "PER@1 and PER@2 are ...". At test time, we de-anonymize NE@N tokens in the output by looking them up in their source sentences. Note that the de-anonymization may fail, but the chance is small (around 2% of the time on the Newsela development set). We replaced words occurring three times or less in the training set with UNK. At test time, when our models predict UNK, we adopt the UNK replacement method proposed in Jean et al. (2015).

Evaluation Following previous work (Woodsend and Lapata, 2011; Xu et al., 2016) we evaluated system output automatically adopting metrics widely used in the simplification literature. Specifically, we used BLEU[5] (Papineni et al., 2002) to assess the degree to which generated simplifications differed from gold standard references and the Flesch-Kincaid Grade Level index (FKGL; Kincaid et al. 1975) to measure the readability of the output (lower FKGL[6] implies simpler output). In addition, we used SARI (Xu et al., 2016), which evaluates the quality of the output by comparing it against the source and reference simplifications.[7] BLEU, FKGL, and SARI are all measured at corpus-level. We also evaluated system output by eliciting human judgments via Amazon's Mechanical Turk. Specifically (self-reported) native English speakers were asked to rate simplifications on three dimensions: *Fluency* (is the output grammatical and well formed?), *Adequacy* (to what extent is the meaning expressed in the original sentence preserved in the output?) and *Simplicity* (is the output simpler than the original sentence?). All ratings were obtained using a five point Likert scale.

Comparison Systems We compared our model against several systems previously proposed in the literature. These include PBMT-R, a mono-

[2] https://newsela.com

[3] If a sentence has multiple references in the development or test set, we use the reference with highest simplicity level.

[4] Weights were tuned on the development set of the Newsela dataset and kept fixed for the other two datasets.

[5] With the default mteval-v13a.pl settings.

[6] FKGL implementation at http://goo.gl/OHP7k3.

[7] We used he implementation of SARI in Xu et al. (2016).

Newsela	BLEU	FKGL	SARI
PBMT-R	18.19	7.59	15.77
Hybrid	14.46	**4.01**	**30.00**
EncDecA	21.70	5.11	24.12
Dress	23.21	4.13	27.37
Dress-Ls	**24.30**	4.21	26.63

WikiSmall	BLEU	FKGL	SARI
PBMT-R	46.31	11.42	15.97
Hybrid	**53.94**	9.20	**30.46**
EncDecA	47.93	11.35	13.61
Dress	34.53	**7.48**	27.48
Dress-Ls	36.32	7.55	27.24

WikiLarge	BLEU	FKGL	SARI
PBMT-R	81.11	8.33	38.56
Hybrid	48.97	**4.56**	31.40
SBMT-SARI	73.08	7.29	**39.96**
EncDecA	**88.85**	8.41	35.66
Dress	77.18	6.58	37.08
Dress-Ls	80.12	6.62	37.27

Table 1: Automatic evaluation on Newsela, WikiSmall, and WikiLarge test sets.

lingual phrase-based machine translation system with a reranking post-processing step[8] (Wubben et al., 2012) and Hybrid, a model which first performs sentence splitting and deletion operations over discourse representation structures and then further simplifies sentences with PBMT-R (Narayan and Gardent, 2014). Hybrid[9] is state of the art on the WikiSmall dataset. Comparisons with SBMT-SARI, a syntax-based translation model trained on PPDB (Ganitkevitch et al., 2013) and tuned with SARI (Xu et al., 2016), are problematic due to the size of PPDB which is considerably larger than any of the datasets used in this work (it contains 106 million sentence pairs with 2 billion words). Nevertheless, we compare[10] against SBMT-SARI, but only models trained on Wikilarge, our largest dataset.

6 Results

Since Newsela contains high quality simplifications created by professional editors, we performed the bulk of our experiments on this dataset. Specifically, we set out to answer two questions: (a) which neural model performs best and (b) how do neural models which are resource lean and do not have access to linguistic annotations fare against more traditional systems. We therefore compared the basic attention-based encoder-decoder model (EncDecA), with the deep reinforcement learning model (Dress; Section 3), and a linear combination of Dress and the lexical simplification model (Dress-Ls; Section 4). Neural models were further compared against two strong baselines, PBMT-R and Hybrid. Table 3 shows example output of all models on the Newsela dataset.

Newsela	Fluency	Adequacy	Simplicity	All
PBMT-R	3.56	**3.58****	2.09**	3.08**
Hybrid	2.70**	2.51**	2.99	2.73**
EncDecA	3.63	2.99	2.56**	3.06**
Dress	3.65	2.94	**3.10**	3.23
Dress-Ls	**3.71**	3.07	3.04	**3.28**
Reference	3.90	2.81**	3.42**	3.38

WikiSmall	Fluency	Adequacy	Simplicity	All
PBMT-R	3.91	**3.74****	2.80**	3.48*
Hybrid	3.26**	3.42	2.82**	3.17**
Dress-Ls	**3.92**	3.36	**3.55**	**3.61**
Reference	3.74*	3.34	3.13**	3.41**

WikiLarge	Fluency	Adequacy	Simplicity	All
PBMT-R	3.68	**3.63***	2.70**	3.34*
Hybrid	2.60**	2.42**	**3.52**	2.85**
SBMT-SARI	3.34**	3.51*	2.77**	3.21**
Dress-Ls	**3.70**	3.28	3.42	**3.46**
Reference	3.79	3.72**	2.86**	3.46

Table 2: Mean ratings elicited by humans on Newsela, WikiSmall, and WkiLarge test sets. Ratings significantly different from Dress-Ls are marked with * ($p < 0.05$) and ** ($p < 0.01$). Significance tests were performed using a student t-test.

The top block in Table 1 summarizes the results of our automatic evaluation. As can be seen, all neural models obtain higher BLEU, lower FKGL and higher SARI compared to PBMT-R. Hybrid has the lowest FKGL and highest SARI. Compared to EncDecA, Dress scores lower on FKGL and higher on SARI, which indicates that the model has indeed learned to optimize the reward function which includes SARI. Integrating lexical simplification (Dress-Ls) yields better BLEU, but slightly worse FKGL and SARI.

The results of our human evaluation are presented in the top block of Table 2. We elicited judgments for 100 randomly sampled test sentences. Aside from comparing system output (PBMT-R, Hybrid, EncDecA, Dress, and Dress-Ls), we also elicited ratings for the gold standard Reference as an upper bound. We report results for Fluency, Adequacy, and Simplicity individually and in combination (All is the average rating of the three dimensions). As can be seen, Dress and Dress-Ls outperform PBMT-R and

[8] We made a good-faith effort to re-implement their system following closely the details in Wubben et al. (2012).

[9] We are grateful to Shashi Narayan for running his system on our three datasets.

[10] The output of SBMT-SARI is publicly available.

Complex	There’s just one major hitch: the primary purpose of education is to develop citizens with a wide variety of skills.
Reference	The purpose of education is to develop a wide range of skills.
PBMT-R	It’s just one major hitch: the purpose of education is to **make people** with a wide variety of skills.
Hybrid	one hitch the purpose is to develop citizens.
EncDecA	The **key** of education is to develop **people** with a wide variety of skills.
Dress	There’s just one major hitch: the **main goal** of education is to develop **people** with **lots of** skills.
Dress-Ls	There’s just one major hitch: the **main goal** of education is to develop citizens with **lots of** skills.
Complex	“They were so burdened by the past they couldn’t think about the future,” said Barnet, 62, who was president of Columbia Records, the No.1 record label in the United States, before joining Capitol.
Reference	Capitol was stuck in the past. It could not think about the future, Barnett said.
PBMT-R	“They were so **affected** by the past they couldn’t think about the future,” said Barnett, 62, was president of Columbia Records, before joining Capitol **building**.
Hybrid	‘They were so burdened by the past they couldn’t think about the future,” said Barnett, 62, who was Columbia Records, president of the No.1 record label in the united states, before joining Capitol.
EncDecA	“They were so burdened by the past they couldn’t think about the future,” said Barnett, who was president of Columbia Records, the No.1 record labels in the United States.
Dress	“They were so **sicker** by the past they couldn’t think about the future,” said Barnett, who was president of Columbia Records.
Dress-Ls	“They were so burdened by the past they couldn’t think about the future,” said Barnett, who was president of Columbia Records.

Table 3: System output for two sentences (Newsela development set). Substitutions are shown in bold.

Hybrid on Fluency, Simplicity, and overall. The fact that neural models (EncDecA, Dress and Dress-Ls) fare well on Fluency, is perhaps not surprising given the recent success of LSTMs in language modeling and neural machine translation (Zaremba et al., 2014; Jean et al., 2015).

Neural models obtain worse ratings on Adequacy but are closest to the human references on this dimension. Dress-Ls (and Dress) are significantly better ($p < 0.01$) on Simplicity than EncDecA, PBMT-R, and Hybrid which indicates that our reinforcement learning based model is effective at creating simpler output. Combined ratings (All) for Dress-Ls are significantly different compared to the other models but not to Dress and the Reference. Nevertheless, integration of the lexical simplification model boosts performance as ratings increase almost across the board (Simplicity is slightly worse). Returning to our original questions, we find that neural models are more fluent than comparison systems, while performing non-trivial rewrite operations (see the SARI scores in Table 1) which yield simpler output (see the Simplicity column in Table 2). Based on our judgment elicitation study, neural models trained with reinforcement learning perform best, with Dress-Ls having a slight advantage.

We further analyzed model performance by computing various statistics on the simplified output. We measured average sentence length and the degree to which Dress and comparison systems perform rewriting operations. We approximated the latter with Translation Error Rate (TER; Snover et al. 2006), a measure commonly used to automatically evaluate the quality of machine translation output. We used TER to compute the (average) number of edits required to change an original complex sentence to simpler output. We also report the number of edits by type, i.e., the number of insertions, substitutions, deletions, and shifts needed (on average) to convert complex to simple sentences.

As shown in Table 4, Hybrid obtains the highest TER, followed by our models (Dress and

Models	Len	TER	Ins	Del	Sub	Shft
PBMT-R	23.1	0.13	0.68	0.68	1.50	0.09
Hybrid	12.4	0.90	0.01	10.19	0.12	0.41
EncDecA	17.0	0.36	0.13	5.96	1.69	0.09
Dress	14.2	0.46	0.07	8.53	1.37	0.11
Dress-Ls	14.4	0.44	0.07	8.38	1.11	0.09
Reference	12.7	0.67	0.40	10.26	3.44	0.73

Table 4: Output length (average number of tokens), TER scores and number of edits by type (Ins*ertions*, Del*etions*, Sub*stitutions*, Sh*i*ft*s*) on the Newsela test set. Higher TER means that more rewriting operations are performed.

Dress-Ls), which indicates that they actively perform rewriting. Perhaps Hybrid is too aggressive when simplifying a sentence, it obtains low Fluency and Adequacy scores in human evaluation (Table 2). There is a strong correlation between sentence length and number of deletion operations (i.e., more deleteions lead to shorter sentences) and PBMT-R performs very few deletions. Overall, reinforcement learning encourages deletion (see Dress and Dress-Ls), while performing a reasonable amount of additional operations (e.g., substitutions and shifts) compared to EncDecA and PBMT-R.

The middle blocks in Tables 1 and 2 report results on the WikiSmall dataset. FKGL and SARI follow a similar pattern as on Newsela. BLEU scores for PBMT-R, Hybrid, and EncDecA are much higher compared to Dress and Dress-Ls. Hybrid obtains best BLEU and SARI scores, while Dress and Dress-Ls do very well on FKGL. In human evaluation, we elicited judgments on the entire WikiSmall test set (100 sentences). We compared Dress-Ls, with PBMT-R, Hybrid, and gold standard Reference simplifications. As human experiments are time consuming and expensive, we did not include other neural models besides Dress-Ls based on our Newsela study which showed that EncDecA is inferior to variants trained with reinforcement learning and that Dress-Ls is the better performing model (however, we do compare *all* models in Table 1). Dress-Ls is significantly better on Simplicity than PBMT-R, Hybrid, and the Reference. It performs on par with PBMT-R on Fluency and worse on Adequacy (but still closer to the human Reference than PBMT-R or Hybrid). When combining all ratings (All in Table 2), Dress-Ls is significantly better than PBMT-R, Hybrid, and the Reference.

The bottom blocks in Tables 1 and 2 report results on Wikilarge. We compared our models with PBMT-R, Hybrid, and SBMT-SARI (Xu et al., 2016). The FKGL follows a similar pattern as in the previous datasets. PBMT-R and our models are best in terms of BLEU while SBMT-SARI outperforms all other systems on SARI.[11] Because there are 8 references for each complex sentence in the test set, BLEU scores are much higher compared to Newsela and WikiSmall. In human evaluation, we again elicited judgments for 100 randomly sampled test sentences. We randomly selected one of the 8 references as the Reference upper bound. On Simplicity, Dress-Ls is significantly better than all comparison systems, except Hybrid. On Adequacy, it is better than Hybrid but significantly worse than other comparison systems. On Fluency, it is on par with PBMT-R[12] but better than Hybrid and SBMT-SARI. On All dimension Dress-Ls significantly outperforms all comparison systems.

7 Conclusions

We developed a reinforcement learning-based text simplification model, which can jointly model simplicity, grammaticality, and semantic fidelity to the input. We also proposed a lexical simplification component that further boosts performance. Overall, we find that reinforcement learning offers a great means to inject prior knowledge to the simplification task achieving good results across three datasets. In the future, we would like to explicitly model sentence splitting and simplify entire documents (rather than individual sentences). Beyond sentence simplification, the reinforcement learning framework presented here is potentially applicable to generation tasks such as sentence compression (Chopra et al., 2016), generation of programming code (Ling et al., 2016), or poems (Zhang and Lapata, 2014).

Acknowledgments We would like to thank Li Dong, Jianpeng Cheng, Shashi Narayan and the EMNLP reviewers for their valuable feedback. We are also grateful to Shashi Narayan for supplying us with the output of his system and Wei Xu for her help with this work. The authors acknowledge the support of the European Research Council (award number 681760).

[11] BLEU and SARI scores reported in Xu et al. (2016) are 72.36 and 37.91, and measured at sentence-level.

[12] We used more data to train PBMT-R and maybe that is why PBMT-R performs better than Xu et al. (2016) reported.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*, San Diego, CA.

Beata Beigman Klebanov, Kevin Knight, and Daniel Marcu. 2004. Text simplification for information-seeking applications. In *Proceedings of ODBASE*, volume 3290 of *Lecture Notes in Computer Science*, pages 735–747, Agia Napa, Cyprus. Springer.

J. Carroll, G. Minnen, D. Pearce, Y. Canning, S. Devlin, and J Tait. 1999. Simplifying text for language-impaired readers. In *Proceedings of the 9th EACL*, pages 269–270, Bergen, Norway.

R. Chandrasekar, C. Doran, and B. Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the 16th COLING*, pages 1041–1044, Copenhagen, Denmark.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of NAACL: HLT*, pages 93–98, San Diego, CA.

James Curran, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale nlp with c&c and boxer. In *Proceedings of the 45th ACL Companion Volume Proceedings of the Demo and Poster Sessions*, pages 33–36, Prague, Czech Republic.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087.

Siobhan Devlin. 1999. *Simplifying Natural Language for Aphasic Readers*. Ph.D. thesis, University of Sunderland.

Richard Evans, Constantin Or asan, and Iustin Dornescu. 2014. An evaluation of syntactic simplification rules for people with autism. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 131–140, Gothenburg, Sweden.

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

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

Kentaro Inui, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. 2003. Text simplification for reading assistance: A project note. In *Proceedings of the 2nd International Workshop on Paraphrasing*, pages 9–16, Sapporo, Japan.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal neural machine translation systems for WMT15. In *Proceedings of the 10th Workshop on Statistical Machine Translation*, pages 134–140, Lisbon, Portugal.

Nobuhiro Kaji, Daisuke Kawahara, Sadao Kurohashi, and Satoshi Sato. 2002. Verb paraphrase based on case frame alignment. In *Proceedings of 40th ACL*, pages 215–222, Philadelphia, PA.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st ACL*, pages 1537–1546, Sofia, Bulgaria.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, DTIC Document.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 EMNLP*, pages 1192–1202, Austin, TX.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th ACL*, pages 599–609, Berlin, Germany.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 EMNLP*, pages 1412–1421, Lisbon, Portugal.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*, pages 55–60.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *Proceedings of the 2016 EMNLP*, pages 2355–2365, Austin, TX.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proceedings of the 52nd ACL*, pages 435–445, Baltimore, MD.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th ACL*, pages 311–318, Philadelphia, PA.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *In Proceedings of the EMNLP 2014*, volume 14, pages 1532–43, Doha, Qatar.

MarcAurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of ICLR*, San Juan, Puerto Rico.

Luz Rello, Clara Bayarri, Azuki Górriz, Ricardo Baeza-Yates, Saurabh Gupta, Gaurang Kanvinde, Horacio Saggion, Stefan Bott, Roberto Carlini, and Vasile Topac. 2013. Dyswebxia 2.0!: More accessible text for people with dyslexia. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, pages –, Brazil.

Seonggi Ryang and Takeshi Abekawa. 2012. Framework of automatic text summarization using reinforcement learning. In *Proceedings of the 2012 EMNLP-CoNLL*, pages 256–265, Jeju Island, Korea.

Matthew Shardlow. 2014. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, pages 581–701. Special Issue on Natural Language Processing.

Advaith Siddharthan. 2004. Syntactic simplification and text cohesion. in research on language and computation. *Research on Language and Computation*, 4(1):77–109.

Advaith Siddharthan. 2014. A survey of research on text simplification. *International Journal of Applied Linguistics*, 165(2):259–298.

David A Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the NAACL 206 Workshop on Statistical Machine Translation*, pages 23–30, New York City.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200.

Lucia Specia, Sujay Kumar Jauhar, and Rada Mihalcea. 2012. Semeval-2012 task 1: English lexical simplification. In *In Proceedings of *SEM 2012*, pages 347–355, Montréal, Canada.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

D. Vickrey and D. Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*, pages 344–352, Columbus, OH.

Willian Massami Watanabe, Arnaldo Candido Junior, Vinícius Rodriguez de Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Alu´sio. 2009. Facilita: reading assistance for low-literacy readers. In *Proceedings of the 27th ACM International Conference on Design of Communication*, Bloomington, IN.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 EMNLP*, pages 409–420, Edinburgh, Scotland.

Kristian Woodsend and Mirella Lapata. 2014. Text rewriting improves semantic role labeling. *Journal of Artificial Intelligence Research*, 51:133–164.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th ACL*, pages 1015–1024, Jeju Island, Korea.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015a. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2(3):5.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015b. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th ACL*, pages 523–530, Toulouse, France.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 EMNLP*, pages 670–680, Doha, Qatar.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd COLING*, pages 1353–1361, Beijing, China.

Learning how to Active Learn: A Deep Reinforcement Learning Approach

Meng Fang and **Yuan Li** and **Trevor Cohn**
School of Computing and Information Systems
The University of Melbourne
meng.fang@unimelb.edu.au, yuanl4@student.unimelb.edu.au,
t.cohn@unimelb.edu.au

Abstract

Active learning aims to select a small subset of data for annotation such that a classifier learned on the data is highly accurate. This is usually done using heuristic selection methods, however the effectiveness of such methods is limited and moreover, the performance of heuristics varies between datasets. To address these shortcomings, we introduce a novel formulation by reframing the active learning as a reinforcement learning problem and explicitly learning a data selection policy, where the policy takes the role of the active learning heuristic. Importantly, our method allows the selection policy learned using simulation on one language to be transferred to other languages. We demonstrate our method using cross-lingual named entity recognition, observing uniform improvements over traditional active learning.

1 Introduction

For most Natural Language Processing (NLP) tasks, obtaining sufficient annotated text for training accurate models is a critical bottleneck. Thus active learning has been applied to NLP tasks to minimise the expense of annotating data (Thompson et al., 1999; Tong and Koller, 2001; Settles and Craven, 2008). Active learning aims to reduce cost by identifying a subset of unlabelled data for annotation, which is selected to maximise the accuracy of a supervised model trained on the data (Settles, 2010). There have been many successful applications to NLP, e.g., Tomanek et al. (2007) used an active learning algorithm for CoNLL corpus to get an F_1 score 84% with a reduction of annotation cost of about 48%. In prior work most active learning algorithms are designed for English based on heuristics, such as using uncertainty or informativeness. There has been comparatively little work done about how to learn the active learning strategy itself.

It is no doubt that active learning is extremely important for other languages, particularly low-resource languages, where annotation is typically difficult to obtain, and annotation budgets more modest (Garrette and Baldridge, 2013). Such settings are a natural application for active learning, however there is little work to this end. A potential reason is that most active learning algorithms require a substantial 'seed set' of data for learning a basic classifier, which can then be used for active data selection. However, given the dearth of data in the low-resource setting, this assumption can make standard approaches infeasible.

In this paper,[1] we propose PAL, short for *Policy based Active Learning*, a novel approach for learning a dynamic active learning strategy from data. This allows for the strategy to be applied in other data settings, such as cross-lingual applications. Our algorithm does not use a fixed heuristic, but instead learns how to actively select data, formalised as a reinforcement learning (RL) problem. An intelligent agent must decide whether or not to select data for annotation in a streaming setting, where the decision policy is learned using a deep Q-network (Mnih et al., 2015). The policy is informed by observations including sentences' content information, the supervised model's classifications and its confidence. Accordingly, a rich and dynamic policy can be learned for annotating new data based on the past sequence of annotation decisions.

Furthermore, in order to reduce the dependence on the data in the target language, which may be low resource, we first learn the policy of active

[1] Source code available at https://github.com/mengf1/PAL

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 595–605
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

learning on another language and then transfer it to the target language. It is easy to learn a policy on a high resource language, where there is plentiful data, such as English. We use cross-lingual word embeddings to learn compatible data representations for both languages, such that the learned policy can be easily ported into the other language.

Our work is different for prior work in active learning for NLP. Most previous active learning algorithms developed for NER tasks is based on one language and then applied to the language itself. Another main difference is that many active learning algorithms use a fixed data selection heuristic, such as uncertainty sampling (Settles and Craven, 2008; Stratos and Collins, 2015; Zhang et al., 2016). However, in our algorithm, we implicitly use uncertainty information as one kind of observations to the RL agent.

The remainder of this paper is organised as follows. In Section 2, we briefly review some related work. In Section 3, we present active learning algorithms, which cross multiple languages. The experimental results are presented in Section 4. We conclude our work in Section 5.

2 Related work

As supervised learning methods often require a lot of training data, active learning is a technique that selects a subset of data to annotate for training the best classifier. Existing active learning (AL) algorithms can be generally considered as three categories: 1) uncertainty sampling (Lewis and Gale, 1994; Tong and Koller, 2001), which selects the data about which the current classifier is the most uncertain; 2) query by committee (Seung et al., 1992), which selects the data about which the "committee" disagree most; and 3) expected error reduction (Roy and McCallum, 2001), which selects the data that can contribute the largest model loss reduction for the current classifier once labelled. Applications of active learning to NLP include text classification (McCallumzy and Nigamy, 1998; Tong and Koller, 2001), relation classification (Qian et al., 2014), and structured prediction (Shen et al., 2004; Settles and Craven, 2008; Stratos and Collins, 2015; Fang and Cohn, 2017). Qian et al. used uncertainty sampling to jointly perform on English and Chinese. Stratos and Collins and Zhang et al. deployed uncertainty-based AL algorithms for languages with the minimal supervision.

Deep reinforcement learning (DRL) is a general-purpose framework for decision making based on representation learning. Recently, there are some notable examples include deep Q-learning (Mnih et al., 2015), deep visuomotor policies (Levine et al., 2016), attention with recurrent networks (Ba et al., 2015), and model predictive control with embeddings (Watter et al., 2015). Other important works include massively parallel frameworks (Nair et al., 2015), dueling architecture (Wang et al., 2016) and expert move prediction in the game of Go (Maddison et al., 2015), which produced policies matching those of the Monte Carlo tree search programs, and squarely beaten a professional player when combined with search (Silver et al., 2016). DRL has been also studied in NLP tasks. For example, recently, DRL has been studied for information extraction problem (Narasimhan et al., 2016). They designed a framework that can decide to acquire external evidence and the framework is under the reinforcement learning method. However, there has been fairly little work on using DRL to learn active learning strategies for language processing tasks, especially in cross-lingual settings.

Recent deep learning work has also looked at transfer learning (Bengio, 2012). More recent work in deep learning has also considered transferring policies by reusing policy parameters between environments (Parisotto et al., 2016; Rusu et al., 2016), using either regularization or novel neural network architectures, though this work has not looked at transfer active learning strategies between languages with shared feature space in state.

3 Methodology

We now show how active learning can be formalised as as a decision process, and then show how this allows for the active learning selection policy to be learned from data using deep reinforcement learning. Later we introduce a method for transferring the policy between languages.

3.1 Active learning as a decision process

Active learning is a simple technique for labelling data, which involves first selecting some instances from an unlabelled dataset, which are then annotated by a human oracle, which is then repeated many times until a termination criterion is satisfied, e.g., the annotation budget is exhausted. Most often the selection function is based on the predictions of a trained model, which has been fit to the labelled dataset at each stage in the algorithm, where datapoints are selected based on the model's predictive uncertainty (Lewis and Gale, 1994), or divergence in predictions over an ensemble (Seung et al., 1992). The key idea of these methods is to find the instances on which the model is most likely to make errors, such that after their labelling and inclusion in the training set, the model becomes more robust to these types of errors on unseen data.

The steps in active learning can be viewed as a decision process, a means of formalising the active learning algorithm as a sequence of decisions, where the stages of active learning correspond to the state of the system. Accordingly, the *state* corresponds to the selected data for labelling and their labels, and each step in the active learning algorithm corresponds to a selection *action*, wherein the heuristic selects the next items from a pool. This process terminates when the budget is exhausted.

Effectively the active learning heuristic is operating as a decision *policy*, a form of function taking as input the current state — comprising the labelled data, from which a model is trained — and a candidate unlabelled data point — e.g., the model uncertainty. This raises the opportunity to consider general policy functions, based on the state and data point inputs, and resulting in a labelling decision, and, accordingly a mechanism for learning such functions from data. We now elaborate on the components of this process, namely the formulation of the decision process, architecture of the policy function, and means of learning the decision policy automatically from data.

3.2 Stream-based learning

For simplicity, we make a streaming assumption, whereby unlabelled data (sentences) arrive in a stream (Lewis and Gale, 1994).[2] As each instance arrives, an agent must decide the action to take, namely whether or not the instance should be manually annotated. This process is illustrated in Figure 1, which illustrates the space of decision sequences for a small corpus. As part of this process, a separate model, p_ϕ, is trained on the labelled data, and updated accordingly as the labelled dataset is expanded as new annotations arrive. This model is central to the policy for choosing the labelling actions at each stage, and for determining the reward for a sequence of actions.

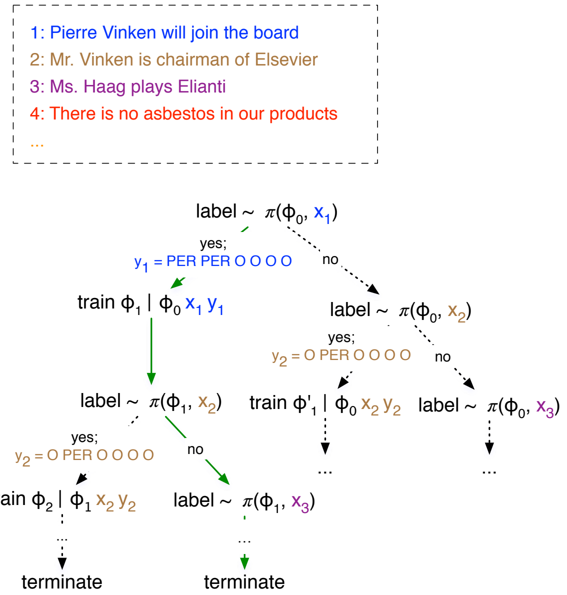

1: Pierre Vinken will join the board
2: Mr. Vinken is chairman of Elsevier
3: Ms. Haag plays Elianti
4: There is no asbestos in our products
...
label ~ $\pi(\phi_0, x_1)$
yes;
y_1 = PER PER O O O O
no
train ϕ_1 | ϕ_0 x_1 y_1
label ~ $\pi(\phi_0, x_2)$
yes;
y_2 = O PER O O O O
no
label ~ $\pi(\phi_1, x_2)$
train ϕ'_1 | ϕ_0 x_2 y_2
label ~ $\pi(\phi_0, x_3)$
yes;
y_2 = O PER O O O O
no
...
...
train ϕ_2 | ϕ_1 x_2 y_2
label ~ $\pi(\phi_1, x_3)$
...
...
terminate
terminate

Figure 1: Example illustrating sequential active learning as a Markov Decision process. Data arrives sequentially, and at each time the active learning policy, π, must decide whether it should be labelled or not, based on the state which includes a predictive model parameterised by ϕ, and an unlabelled data instance $\mathbf{x}$. The process continues until termination, e.g., when the annotation budget is exhausted. The solid green path shows the maximum scoring decision sequence.

This is a form of Markov Decision Process (MDP), which allows the learning of a policy that can dynamically select instances that are most informative. As illustrated in Figure 1 at each time, the agent observes the current state s_i which includes the sentence $\mathbf{x}_i$, and the learned model ϕ. The agent selects a binary action a_i, denoting whether to label $\mathbf{x}_i$, according to the policy π. For $a_i = 1$, the corresponding sentence is labelled and added to the labelled data, and the model p_ϕ updated to include this new training point. The process then repeats, terminating when either the dataset is exhausted or a fixed annotation budget is reached. After termination a reward is computed based on the accuracy of the final model, ϕ. We represent the MDP framework as a tuple $\langle S, A, Pr(s_{i+1}|s_i, a), R\rangle$, where $S = \{s\}$ is the space of all possible states, $A = \{0, 1\}$ is the set of actions, $R(s, a)$ is the reward function, and $Pr(s_{i+1}|s_i, a)$ is the transition function.

[2] This is different to pool-based active learning, where one of several options is chosen for annotation. Our setup permits simpler learning, while remaining sufficiently general.

3.2.1 State

The state at time i comprises the candidate instance being considered for annotation and the labelled dataset constructed in steps $1 \ldots i$. We represent the state using a continuous vector, using the concatenation of the vector representation of $\mathbf{x}_i$, and outputs of the model p_ϕ trained over the labelled data. These outputs use both the predictive marginal distributions of the model on the instance, and a representation of the model's confidence. We now elaborate on each component.

Content representation A key input to the agent is the content of the sentence, $\mathbf{x}_i$, which we encode using a convolutional neural network to arrive at a fixed sized vector representation, following Kim (2014). This involves embedding each of the n words in the sentence to produce a matrix $X_i = \{x_{i,1}, x_{i,2}, \cdots, x_{i,n}\}$, after which a series of wide convolutional filters is applied, using multiple filters with different gram sizes. Each filter uses a linear transformation with a rectified linear unit activation function. Finally the filter outputs are merged using a max-pooling operation to yield a hidden state $\mathbf{h}_c$, which is used to represent the sentence.

Representation of marginals The prediction outputs of the training model, $p_\phi(\mathbf{y}|\mathbf{x}_i)$, are central to all active learning heuristics, and accordingly, we include this in our approach. In order to generalise existing techniques, we elect to use the predictive marginals directly, rather than only using statistics thereof, e.g., entropy. This generality allows for different and more nuanced concepts to be learned, including patterns of probabilities that span several adjacent positions in the sentence (e.g., the uncertainty about the boundary of a named entity).

We use another convolutional neural network to process the predictive marginals, as shown in Figure 2. The convolutional layer contains j filters with ReLU activation, based on a window of width 3 and height equal to the number of classes, and with a stride of one token. We use a wide convolution, by padding the input matrix to either size with vectors of zeros. These j feature maps are then subsampled with mean pooling, such that the network is easily able to capture the average uncertainty in each window. The final hidden layer $\mathbf{h}_e$ is used to represent the predictive marginals.

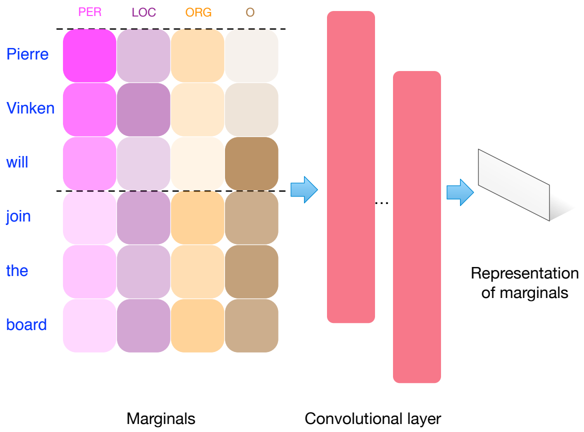


Figure 2: The architecture for representing predictive marginal distributions, $p_\phi(\mathbf{y}|\mathbf{x}_i)$, as a fixed dimensional vector, to form part of the MDP state.

Confidence of sequential prediction The last component is a score C which indicates the confidence of the model prediction. This is defined based on the most probable label sequence under the model, e.g., using Viterbi algorithm with a CRF, and the probability of this sequence is used to represent the confidence, $C = \sqrt[n]{\max_{\mathbf{y}} p_\phi(\mathbf{y}|\mathbf{x}_i)}$, where $n = |\mathbf{x}_i|$ is the length of the sentence.

3.2.2 Action

We now turn to the action, which denotes whether the human oracle must annotate the current sentence. The agent selects either to annotate $\mathbf{x}_i$, in which case $a_i = 1$, or not, with $a_i = 0$, after which the agent proceeds to consider the next instance, $\mathbf{x}_{i+1}$. When action $a_i = 1$ is chosen, an oracle is requested to annotate the sentence, and the newly annotated sentence is added to the training data, and ϕ updated accordingly. A special 'terminate' option applies when no further data remains or the annotation budget is exhausted, which concludes the active learning run (referred to as an 'episode' or 'game' herein).

3.2.3 Reward

The training signal for learning the policy takes the form of a scalar 'reward', which provides feedback on the quality of the actions made by the agent. The most obvious reward is to wait for a game to conclude, then measure the held-out performance of the model, which has been trained on the labelled data. However, this reward is delayed, and is difficult to related to individual actions after a long game. To compensate for this, we use reward shaping, whereby small intermediate rewards are assigned which speeds up the learning process (Ng, 2003; Lample and Chaplot, 2016). At each step, the intermediate reward is defined as the change in held-out performance, i.e., $R(s_{i-1}, a) = \text{Acc}(\phi_i) - \text{Acc}(\phi_{i-1})$, where Acc denotes predictive accuracy (here F1 score), and ϕ_i is the trained model after action a has take place, which may include an additional training instance. Accordingly, when considering the aggregate reward over a game, the intermediate terms cancel, such that the total reward measures the performance improvement over the whole game. Note that the value of $R(s, a)$ can be positive or negative, indicating a beneficial or detrimental effect on the performance.

3.2.4 Budget

There is a fixed budget $\mathcal{B}$ for the total number of instances annotated, which corresponds to the terminal state in the MDP. It is a predefined number and chosen according to time and cost constraints. A game is finished when the data is exhausted or the budget reached, and with the final result being the dataset thus created, upon which the final model is trained.

3.2.5 Reinforcement learning

The remaining question is how the above components can be used to *learn* a good policy. Different policies make different data selections, and thus result in models with different performance. We adopt a reinforcement learning (RL) approach to learn a policy resulting a highly accurate model.

Having represented the problem as a MDP, episode as a sequence of transitions (s_i, a, r, s_{i+1}). One episode of active learning produces a finite sequence of states, actions and rewards. We use a deep Q-learning approach (Mnih et al., 2015), which formalises the policy using function $Q^\pi(s, a) \to \mathcal{R}$ which determines the utility of taking a from state s according to a policy π. In Q-learning, the agent iteratively updates $Q(s, a)$ using rewards obtained from each episode, with updates based on the recursive Bellman equation for the optimal Q:

$$Q^\pi(s, a) = \mathbb{E}[R_i | s_i = s, a_i = a, \pi]. \quad (1)$$

Here, $R_i = \sum_{t=i}^{T} \gamma^{t-i} r_t$ is the discounted future reward and $\gamma \in [0, 1]$ is a factor discounting the value of future rewards and the expectation is taken over all transitions involving state s and action a.

Algorithm 1 Learn an active learning policy

```
Input: data $\mathcal{D}$, budget $\mathcal{B}$
Output: $\pi$
for episode = $1, 2, \ldots, N$ do
    $\mathcal{D}_l \leftarrow \emptyset$ and shuffle $\mathcal{D}$
    $\phi \leftarrow$ Random
    for $i \in \{0, 1, 2, \ldots, |\mathcal{D}|\}$ do
        Construct the state $s_i$ using $\mathbf{x}_i$
        The agent makes a decision according to
        $a_i = \arg\max Q^\pi(s_i, a)$
        if $a_i = 1$ then
            Obtain the annotation $\mathbf{y}_i$
            $\mathcal{D}_l \leftarrow \mathcal{D}_l + (\mathbf{x}_i, \mathbf{y}_i)$
            Update model $\phi$ based on $\mathcal{D}_l$
        end if
        Receive a reward $r_i$ using held-out set
        if $|\mathcal{D}_l| = \mathcal{B}$ then
            Store $(s_i, a_i, r_i, \text{Terminate})$ in $\mathcal{M}$
            Break
        end if
        Construct the new state $s_{i+1}$
        Store transition $(s_i, a_i, r_i, s_{i+1})$ in $\mathcal{M}$
        Sample random minibatch of transitions
        $\{(s_j, a_j, r_j, s_{j+1})\}$ from $\mathcal{M}$, and per-
        form gradient descent step on $\mathcal{L}(\theta)$
        Update policy $\pi$ with $\theta$
    end for
end for
return the latest policy $\pi$
```

Following Deep Q-learning (Mnih et al., 2015), we make use of a deep neural network to compute the expected Q-value, in order to update the parameters. We implement the Q-function using a single hidden layer neural network, taking as input the state representation $(\mathbf{h}_c, \mathbf{h}_e, C)$ (defined in §3.2.1), and outputting two scalar values corresponding to the values $Q(s, a)$ for $a \in \{0, 1\}$. This network uses a rectified linear unit (ReLU) activation function in its hidden layer.

The parameters in the DQN are learnt using stochastic gradient descent, based on a regression objective to match the Q-values predicted by the DQN and the expected Q-values from the Bellman equation, $r_i + \gamma \max_a Q(s_{i+1}, a; \theta)$. Following (Mnih et al., 2015), we use an experience replay memory $\mathcal{M}$ to store each transition (s, a, r, s') as it is used in an episode, after which

Algorithm 2 Active learning by policy transfer

```
Input: unlabelled data $\mathcal{D}$, budget $\mathcal{B}$, policy $\pi$
Output: $\mathcal{D}_l$
$\mathcal{D}_l \leftarrow \emptyset$
$\phi \leftarrow$ Random
for $|\mathcal{D}_l| \neq \mathcal{B}$ and $\mathcal{D}$ not empty do
    Randomly sample $\mathbf{x}_i$ from the data pool $\mathcal{D}$ and construct the state $s_i$
    The agent chooses an action $a_i$ according to $a_i = \arg\max Q^{\pi}(s_i, a)$
    if $a_i = 1$ then
        Obtain the annotation $\mathbf{y}_i$
        $\mathcal{D}_l \leftarrow \mathcal{D}_l + (\mathbf{x}_i, \mathbf{y}_i)$
        Update model $\phi$ based on $\mathcal{D}_l$
    end if
    $\mathcal{D} \leftarrow \mathcal{D} \backslash \mathbf{x}_i$
    Receive a reward $r_i$ using held-out set
    Update policy $\pi$
end for
return $\mathcal{D}_l$
```

we sample a mini-batch of transitions from the memory and then minimize the loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'}\left[\left(y_i(r,s') - Q(s,a;\theta)\right)^2\right], \quad (2)$$

where $y_i(r,s') = r + \gamma \max_{a'} Q(s',a';\theta_{i-1})$ is the target Q-value, based on the current parameters θ_{i-1}, and the expectation is over the mini-batch. Learning updates are made every training step, based on stochastic gradient descent to minimise Eq. 2 w.r.t. parameters θ.

The algorithm for learning is summarised in Algorithm 1. We train the policy by running multiple active learning episodes over the training data, where each episode is a simulated active learning run. For each episode, we shuffle the data, and hide the known labels, which are revealed as requested during the run. A disjoint held-out set is used to compute the reward, i.e., model accuracy, which is fixed over the episodes. Between each episode the model is reset to its initialisation condition, with the main changes being the different (random) data ordering and the evolving policy function.

3.3 Cross-lingual policy transfer

We now turn to the question of how the learned policy can be applied to another dataset. Given the extensive use of the training dataset, the policy application only makes sense when employed in a different data setting, e.g., where the domain, task or language is different. For this paper, we consider a cross-lingual application of the same task (NER), where we train a policy on a source language (e.g., English), and then transfer the learned policy to a different target language. Cross-lingual word embeddings provide a common shared representation to facilitate application of the policy to other languages.

Algorithm 3 Active learning by policy *and* model transfer, for 'cold-start' scenario

```
Input: unlabelled data $\mathcal{D}$, budget $\mathcal{B}$, policy $\pi$, model $\phi$
Output: $\mathcal{D}_l$
$\mathcal{D}_l \leftarrow \emptyset$
for $|\mathcal{D}_l| \neq \mathcal{B}$ and $\mathcal{D}$ not empty do
    Randomly sample $\mathbf{x}_i$ from the data pool $\mathcal{D}$ and construct the state $s_i$
    The agent chooses an action $a_i$ according to $a_i = \arg\max Q^{\pi}(s_i, a)$
    if $a_i = 1$ then
        $\mathcal{D}_l \leftarrow \mathcal{D}_l + (\mathbf{x}_i, -)$
    end if
    $\mathcal{D} \leftarrow \mathcal{D} \backslash \mathbf{x}_i$
end for
Obtain all the annotations for $\mathcal{D}_l$
return $\mathcal{D}_l$
```

We illustrate the policy transfer algorithm in Algorithm 2. This algorithm is broadly similar to Algorithm 1, but has two key differences. Firstly, Algorithm 2 makes only one pass over the data, rather than several passes, as befits an application to a low-resource language where oracle labelling is costly. Secondly, the algorithm also assumes an initial policy, π, which is fine tuned during the episode based on held-out performance such that the policy can adapt to the test scenario.[3]

3.4 Cold-start transfer

The above transfer algorithm has some limitations, which may not be realistic for low-resource settings: the requirement for held-out evaluation data and the embedding of the oracle annotator inside the learning loop. The former implies more supervision than is ideal in a low-resource setting,

[3] Moreover, the algorithm can be extended to a traditional batch setting by evaluating a batch of data instances and selectinag the best k instances for labelling under the policy. This could be applied in either the transfer step (Algorithm 2) or initial policy training (Algorithm 1), or both.

while the latter places limitations on the communication with annotator as well as a necessity for real-time processing, both which are unlikely in a field linguistics setting.

For this data and- communication-impoverished setting, denoted as *cold-start*, we allow only one chance to request labels for the target data, and, having no held-out data, do not allow policy updates. The agent needs to select a batch of unlabelled target instances for annotations, but cannot use these resulting annotations or any other feedback to refine the selection. In this, more difficult cold-start setting, we bootstrap the process with an initial model, such that the agent can make informative decisions in the absence of feedback.

The procedure is outlined in Algorithm 3. Using the cross-lingual word embeddings, we transfer both a policy and a model into the target language. The model, ϕ, is trained on one source language, and the policy is learned on a different source language. Policy learning uses Alg 1, with the small change that in step 3 the model is initialised using ϕ. Consequently the learned policy can exploit the knowledge from cross-lingual initialisation, such that it can figure out which aspects that need to be corrected using target annotated data. Overall this allows for estimates and confidence values to be produced by the model, thus providing the agent with sufficient information for data selection.

4 Experiments

We conduct experiments to validate the proposed active learning method in a cross-lingual setting, whereby an active learning policy trained on a source language is transferred to a target language. We allow repeated active learning simulations on the source language, where annotated corpora are plentiful, to learn a policy, while for target languages we only permit a single episode, to mimic a language without existing resources.

We use NER corpora from CoNLL2002/2003 shared tasks,[4] which comprise NER annotated text in English (en), German (de), Spanish (es), and Dutch (nl), each annotated using the IOB1 labelling scheme, which we convert to the IO labeling scheme. We use the existing corpus partions, with `train` used for policy training, `testb` used as held-out for computing rewards, and final results are reported on `testa`.

Bilingual		Multilingual		Cold-start		
tgt	**src**	**tgt**	**src**	**tgt**	**src**	**pre**
de	en	de	en,nl,es	de	nl	en
nl	en	nl	en,de,es	nl	de	en
es	en	es	en,de,nl	es	de	en
-	-	-	-	de	es	en
-	-	-	-	nl	es	en
-	-	-	-	es	nl	en

Table 1: Experimental configuration for the three settings, showing target language (**tgt**), source language (**src**) as used for policy learning, and language used for pre-training the model (**pre**).

We consider three experimental conditions, as illustrated in Table 1:

bilingual where English is the source (used for policy learning) and we vary the target language;

multilingual where several source languages are the used in joint learning of the policy, and a separate language is used as target; and

cold-start where a pretrained English NER tagger is used to initialise policy learning on a source language, and in cold-start application to a separate target language.

Configuration We now outline the parameter settings for the experimental runs. For learning an active learning policy, we run $N = 10,000$ episodes with budget $\mathcal{B} = 200$ sentences using Alg. 1. Content representations use three convolutional filters of size $3, 4$ and 5, using 128 filters for each size, while for predictive marginals, the convolutional filters are of width 3, using 20 filters. The size of the last hidden layer is 256. The discount factor is set to $\gamma = 0.99$. We used the ADAM algorithm with mini-batches of size 32 for training the neural network. To report performance, we apply the learned policy to the target training set (using Alg. 2 or 3, again with budget 200),[5] after which we use the final trained model for which we report F_1 score.

For word embeddings, we use off the shelf CCA trained multilingual embeddings (Ammar et al.,

[4] http://www.cnts.ua.ac.be/conll2002/ner/, http://www.cnts.ua.ac.be/conll2003/ner/

[5] Although it is possible the policy may learn not to use the full budget, this does not occur in practise.

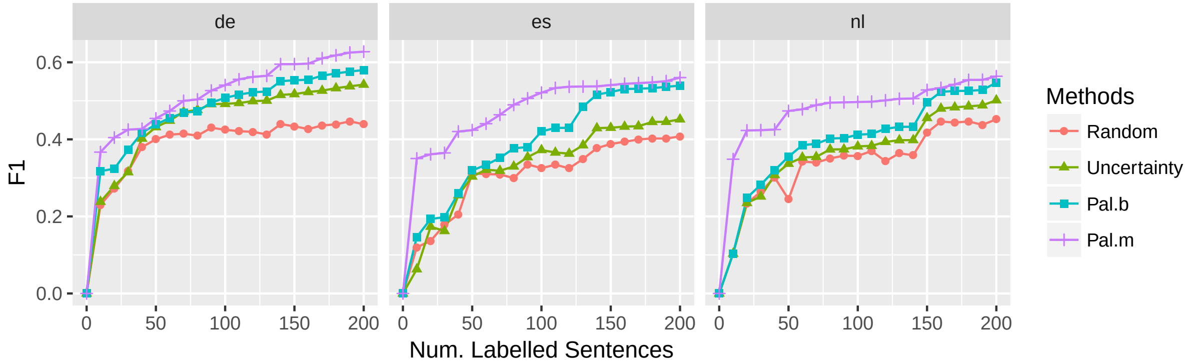


Figure 3: The performance of active learning methods on the bilingual and multilingual settings for three target languages, whereby the active learning policy is trained on only en, or all other languages excluding the target, respectively.

2016),[6] using a 40 dimensional embedding and fixing these during training of both the policy and model. As the model, we use a standard linear chain CRF (Lafferty et al., 2001) for the first two sets of experiments, while for cold-start case we use a basic RNN classifier with the same multilingual embeddings as before, and a 128 dimensional hidden layer.

The proposed method is referred to as PAL, as shorthand *Policy based Active Learning*. Subscripts b, m, c are used to denote the bilingual, multilingual and cold-start experimental configurations. For comparative baselines, we use the following methods:

Uncertainty sampling we use the total token entropy measure (Settles and Craven, 2008), which takes the instance $\mathbf{x}$ maximising $\sum_{t=1}^{|\mathbf{x}|} H(y_t|\mathbf{x}, \phi)$, where H is the token entropy. We use the whole training set as the data pool, and select a single instance for labelling in each active learning step. This method was shown to achieve the best result among model-independent active learning methods on the CoNLL data.

Random sampling which randomly selects examples from the unlabelled pool.

Results Figure 3 shows results the bilingual case, where PAL$_b$ consistently outperforms the Random and Uncertainty baselines across the three target languages. Uncertainty sampling is ineffective, particularly towards the start of the run, as a consequence of its dependence on a high quality model. The use of content information allows PAL$_b$ to make a stronger start, despite the poor initial model.

Also shown in Figure 3 are results for multilingual policy learning, PAL$_m$, which outperform all other approaches including PAL$_b$. This illustrates that the additional training over several languages gives rise to a better policy, than only using one source language. The superior performance is particularly marked in the early stages of the runs for Spanish and Dutch, which may indicate that the approach was better able to learn to exploit the sentence content information.

We evaluate the cold-start setting in Figure 4. Recall that in this setting there are no policy or model updates, as no heldout data is used, and all annotations arrive in a batch. The model, however, is initialised with a NER tagger trained on a different language, which explains why the performance for all methods starts from around 40% rather than 0%. Even in this challenging evaluation setting, our algorithm PAL$_c$ outperforms both baseline methods, showing that deep Q learning allows for better exploitation of the pretrained classifier, alongside the sentence content.

Lastly, we report the results for all approaches in Table 2, based on training on the full 200 labelled sentences as selected under the different methods. It is clear that the PAL methods all outperform the baselines, and among these the multilingual training of PAL$_m$ outperforms the bilingual setting in PAL$_b$. Surprisingly, PAL$_c$ gives the overall best results, despite using a static policy and model during target application, underscoring the importance of model pretraining. Table 2 also re-

[6] http://128.2.220.95/multilingual

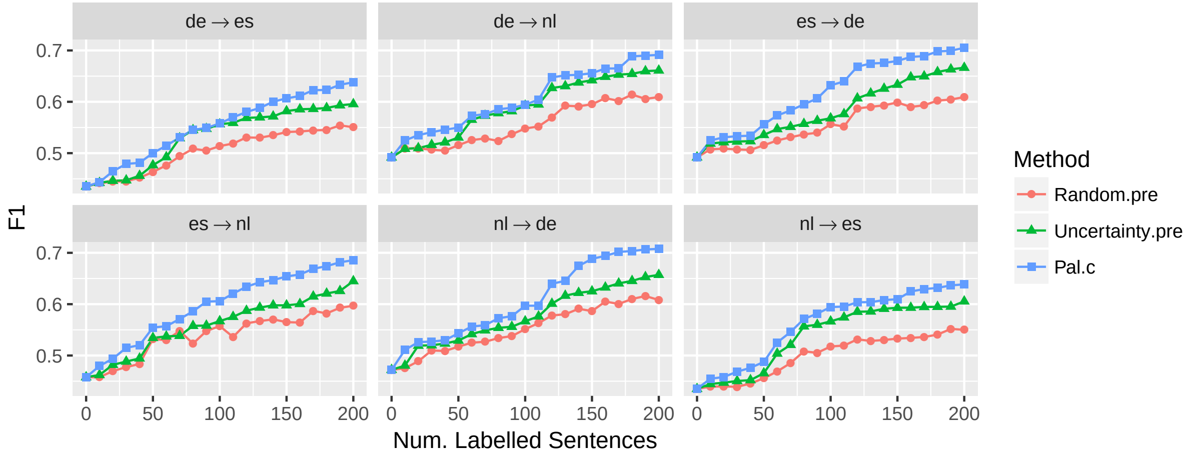


Figure 4: The performance of active learning methods on the cold-start setting, each showing different source → target configurations, in all cases pretraining in en.

	de		nl		es	
	F1	C/R	F1	C/R	F1	C/R
Rand.	44.6	100	45.2	100	40.7	100
Uncert.	54.2	60	50.1	25	45.1	30
PAL$_b$	57.9	60	54.7	25	53.9	40
PAL$_m$	62.7	25	56.3	30	56.0	25
PAL$_c$	70.7	10	69.1	10	63.8	10

Table 2: Results from active learning using the different methods, where each approach constructs a training set of 200 sentences. The three target languages are shown as columns, reporting in each F$_1$ score (%) and the relative cost reduction to match the stated performance of the Random strategy.

ports the cost reduction versus random sampling, showing that the PAL methods can reduce the annotation burden to as low as 10%.

5 Conclusion

In this paper, we have proposed a new active learning algorithm capable of learning active learning strategies from data. We formalise active learning under a Markov decision framework, whereby active learning corresponds to a sequence of binary annotation decisions applied to a stream of data. Based on this, we design an active learning algorithm as a policy based on deep reinforcement learning. We show how these learned active learning policies can be transferred between languages, which we empirically show provides consistent and sizeable improvements over baseline methods, including traditional uncertainty sampling. This holds true even in a very difficult cold-start setting, where no evaluation data is available, and there is no ability to react to annotations.

Acknowledgments

This work was sponsored by the Defense Advanced Research Projects Agency Information Innovation Office (I2O) under the Low Resource Languages for Emergent Incidents (LORELEI) program issued by DARPA/I2O under Contract No. HR0011-15-C-0114. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Trevor Cohn was supported by an Australian Research Council Future Fellowship.

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925* .

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2015. Multiple object recognition with visual attention. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yoshua Bengio. 2012. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. pages 17–36.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. In *Proceedings of the 55th Annual Meeting on Association for Computational Linguistics (ACL)*.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*. pages 138–147.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*. pages 282–289.

Guillaume Lample and Devendra Singh Chaplot. 2016. Playing FPS games with deep reinforcement learning. *arXiv preprint arXiv:1609.05521* .

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17(39):1–40.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th International ACM SIGIR Conference on Research and Development in Information Retrieval*. pages 3–12.

Chris J Maddison, Aja Huang, Ilya Sutskever, and David Silver. 2015. Move evaluation in go using deep convolutional neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Andrew Kachites McCallumzy and Kamal Nigamy. 1998. Employing em and pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning (ICML)*. pages 359–367.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, Shane Legg, Volodymyr Mnih, Koray Kavukcuoglu, and David Silver. 2015. Massively parallel methods for deep reinforcement learning. In *Proceedings of ICML Workshop on Deep Learning*.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *Proceedings of the 2016 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Andrew Y. Ng. 2003. *Shaping and Policy Search in Reinforcement Learning*. Ph.D. thesis, University of California, Berkeley.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. 2016. Actor-mimic: Deep multitask and transfer reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Longhua Qian, Haotian Hui, Yanan Hu, Guodong Zhou, and Qiaoming Zhu. 2014. Bilingual active learning for relation classification via pseudo parallel corpora. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 582–592.

Nicholas Roy and Andrew McCallum. 2001. Toward optimal active learning through monte carlo estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*. pages 441–448.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671* .

Burr Settles. 2010. Active learning literature survey. *University of Wisconsin, Madison* 52(55-66):11.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1070–1079.

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Proceedings of the 5th annual workshop on Computational Learning Theory*. pages 287–294.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. 2004. Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL)*.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Karl Stratos and Michael Collins. 2015. Simple semi-supervised pos tagging. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. pages 79–87.

Cynthia A Thompson, Mary Elaine Califf, and Raymond J Mooney. 1999. Active learning for natural language parsing and information extraction. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*. pages 406–414.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. pages 486–495.

Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research* 2(Nov):45–66.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. 2015. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems (NIPS)*. pages 2746–2754.

Boliang Zhang, Xiaoman Pan, Tianlu Wang, Ashish Vaswani, Heng Ji, Kevin Knight, and Daniel Marcu. 2016. Name tagging for low-resource incident languages based on expectation-driven learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Language Technologies (NAACL-HLT)*. pages 249–259.

Split and Rephrase

Shashi Narayan[†] **Claire Gardent**[‡] **Shay B. Cohen**[†] **Anastasia Shimorina**[‡]
[†] School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, UK
[‡] CNRS, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, F-54500, France
shashi.narayan@ed.ac.uk claire.gardent@loria.fr
scohen@inf.ed.ac.uk anastasia.shimorina@loria.fr

Abstract

We propose a new sentence simplification task (Split-and-Rephrase) where the aim is to split a complex sentence into a meaning preserving sequence of shorter sentences. Like sentence simplification, splitting-and-rephrasing has the potential of benefiting both natural language processing and societal applications. Because shorter sentences are generally better processed by NLP systems, it could be used as a preprocessing step which facilitates and improves the performance of parsers, semantic role labelers and machine translation systems. It should also be of use for people with reading disabilities because it allows the conversion of longer sentences into shorter ones. This paper makes two contributions towards this new task. First, we create and make available a benchmark consisting of 1,066,115 tuples mapping a single complex sentence to a sequence of sentences expressing the same meaning.[1] Second, we propose five models (vanilla sequence-to-sequence to semantically-motivated models) to understand the difficulty of the proposed task.

1 Introduction

Several sentence rewriting operations have been extensively discussed in the literature: sentence compression, multi-sentence fusion, sentence paraphrasing and sentence simplification.

Sentence compression rewrites an input sentence into a shorter paraphrase (Knight and Marcu, 2000; Cohn and Lapata, 2008; Filippova and Strube, 2008; Pitler, 2010; Filippova et al., 2015; Toutanova et al., 2016). Sentence fusion consists of combining two or more sentences with overlapping information content, preserving common information and deleting irrelevant details (McKeown et al., 2010; Filippova, 2010; Thadani and McKeown, 2013). Sentence paraphrasing aims to rewrite a sentence while preserving its meaning (Dras, 1999; Barzilay and McKeown, 2001; Bannard and Callison-Burch, 2005; Wubben et al., 2010; Mallinson et al., 2017). Finally, sentence (or text) simplification aims to produce a text that is easier to understand (Siddharthan et al., 2004; Zhu et al., 2010; Woodsend and Lapata, 2011; Wubben et al., 2012; Narayan and Gardent, 2014; Xu et al., 2015; Narayan and Gardent, 2016; Zhang and Lapata, 2017). Because the vocabulary used, the length of the sentences and the syntactic structures occurring in a text are all factors known to affect readability, simplification systems mostly focus on modelling three main text rewriting operations: simplifying paraphrasing, sentence splitting and deletion.

We propose a new sentence simplification task, which we dub Split-and-Rephrase, where the goal is to split a complex input sentence into shorter sentences while preserving meaning. In that task, the emphasis is on sentence splitting and rephrasing. There is no deletion and no lexical or phrasal simplification but the systems must learn to split complex sentences into shorter ones and to make the syntactic transformations required by the split (e.g., turn a relative clause into a main clause). Table 1 summarises the similarities and differences between the five sentence rewriting tasks.

Like sentence simplification, splitting-and-rephrasing could benefit both natural language processing and societal applications. Because shorter sentences are generally better processed by NLP systems, it could be used as a preprocess-

[1] The Split-and-Rephrase dataset is available here: https://github.com/shashiongithub/Split-and-Rephrase.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 606–616
Copenhagen, Denmark, September 7–11, 2017.

	Split	Delete	Rephr.	MPre.
Compression	N	Y	?Y	N
Fusion	N	Y	Y	?Y
Paraphrasing	N	N	Y	Y
Simplification	Y	Y	Y	N
Split-and-Rephrase	Y	N	Y	Y

Table 1: Similarities and differences between sentence rewriting tasks with respect to splitting (Split), deletion (Delete), rephrasing (Rephr.) and meaning preserving (MPre.) operations (Y: yes, N: No, ?Y: should do but most existing approaches do not).

ing step which facilitates and improves the performance of parsers (Tomita, 1985; Chandrasekar and Srinivas, 1997; McDonald and Nivre, 2011; Jelínek, 2014), semantic role labelers (Vickrey and Koller, 2008) and statistical machine translation (SMT) systems (Chandrasekar et al., 1996). In addition, because it allows the conversion of longer sentences into shorter ones, it should also be of use for people with reading disabilities (Inui et al., 2003) such as aphasia patients (Carroll et al., 1999), low-literacy readers (Watanabe et al., 2009), language learners (Siddharthan, 2002) and children (De Belder and Moens, 2010).

Contributions. We make two main contributions towards the development of Split-and-Rephrase systems.

Our first contribution consists in creating and making available a benchmark for training and testing Split-and-Rephrase systems. This benchmark (WEBSPLIT) differs from the corpora used to train sentence paraphrasing, simplification, compression or fusion models in three main ways.

First, it contains a high number of splits and rephrasings. This is because (i) each complex sentence is mapped to a rephrasing consisting of at least two sentences and (ii) as noted above, splitting a sentence into two usually imposes a syntactic rephrasing (e.g., transforming a relative clause or a subordinate into a main clause).

Second, the corpus has a vocabulary of 3,311 word forms for a little over 1 million training items which reduces sparse data issues and facilitates learning. This is in stark contrast to the relatively small size corpora with very large vocabularies used for simplification (cf. Section 2).

Third, complex sentences and their rephrasings are systematically associated with a meaning representation which can be used to guide learning. This allows for the learning of semantically-informed models (cf. Section 5).

Our second contribution is to provide five models to understand the difficulty of the proposed Split-and-Rephrase task: (i) A basic encoder-decoder taking as input only the complex sentence; (ii) A hybrid probabilistic-SMT model taking as input a deep semantic representation (Discourse representation structures, Kamp 1981) of the complex sentence produced by Boxer (Curran et al., 2007); (iii) A multi-source encoder-decoder taking as input both the complex sentence and the corresponding set of RDF (Resource Description Format) triples; (iv,v) Two partition-and-generate approaches which first, partition the semantics (set of RDF triples) of the complex sentence into smaller units and then generate a text for each RDF subset in that partition. One model is multi-source and takes the input complex sentence into account when generating while the other does not.

2 Related Work

We briefly review previous work on sentence splitting and rephrasing.

Sentence Splitting. Of the four sentence rewriting tasks (paraphrasing, fusion, compression and simplification) mentioned above, only sentence simplification involves sentence splitting. Most simplification methods learn a statistical model (Zhu et al., 2010; Coster and Kauchak, 2011; Woodsend and Lapata, 2011; Wubben et al., 2012; Narayan and Gardent, 2014) from the parallel dataset of complex-simplified sentences derived by Zhu et al. (2010) from Simple English Wikipedia[2] and the traditional one[3].

For training Split-and-Rephrase models, this dataset is arguably ill suited as it consists of 108,016 complex and 114,924 simplified sentences thereby yielding an average number of simple sentences per complex sentence of 1.06. Indeed, Narayan and Gardent (2014) report that only 6.1% of the complex sentences are in fact split in the corresponding simplification. A more detailed evaluation of the dataset by Xu et al. (2015) further shows that (i) for a large number of pairs, the

[2] Simple English Wikipedia (http://simple.wikipedia.org) is a corpus of simple texts targeting "children and adults who are learning English Language" and whose authors are requested to "use easy words and short sentences".

[3] English Wikipedia (http://en.wikipedia.org).

simplifications are in fact not simpler than the input sentence, (ii) automatic alignments resulted in incorrect complex-simplified pairs and (iii) models trained on this dataset generalised poorly to other text genres. Xu et al. (2015) therefore propose a new dataset, Newsela, which consists of 1,130 news articles each rewritten in four different ways to match 5 different levels of simplicity. By pairing each sentence in that dataset with the corresponding sentences from simpler levels (and ignoring pairs of contiguous levels to avoid sentence pairs that are too similar to each other), it is possible to create a corpus consisting of 96,414 distinct complex and 97,135 simplified sentences. Here again however, the proportion of splits is very low.

As we shall see in Section 3.3, the new dataset we propose differs from both the Newsela and the Wikipedia simplification corpus, in that it contains a high number of splits. In average, this new dataset associates 4.99 simple sentences with each complex sentence.

Rephrasing. Sentence compression, sentence fusion, sentence paraphrasing and sentence simplification all involve rephrasing.

Paraphrasing approaches include bootstrapping approaches which start from slotted templates (e.g.,"X is the author of Y") and seed (e.g.,"X = Jack Kerouac, Y = "On the Road"") to iteratively learn new templates from the seeds and new seeds from the new templates (Ravichandran and Hovy, 2002; Duclaye et al., 2003); systems which extract paraphrase patterns from large monolingual corpora and use them to rewrite an input text (Duboue and Chu-Carroll, 2006; Narayan et al., 2016); statistical machine translation (SMT) based systems which learn paraphrases from monolingual parallel (Barzilay and McKeown, 2001; Zhao et al., 2008), comparable (Quirk et al., 2004) or bilingual parallel (Bannard and Callison-Burch, 2005; Ganitkevitch et al., 2011) corpora; and a recent neural machine translation (NMT) based system which learns paraphrases from bilingual parallel corpora (Mallinson et al., 2017).

In sentence simplification approaches, rephrasing is performed either by a machine translation (Coster and Kauchak, 2011; Wubben et al., 2012; Narayan and Gardent, 2014; Xu et al., 2016; Zhang and Lapata, 2017) or by a probabilistic model (Zhu et al., 2010; Woodsend and Lapata, 2011). Other approaches include symbolic approaches where hand-crafted rules are used e.g., to split coordinated and subordinated sentences into several, simpler clauses (Chandrasekar and Srinivas, 1997; Siddharthan, 2002; Canning, 2002; Siddharthan, 2010, 2011) and lexical rephrasing rules are induced from the Wikipedia simplification corpus (Siddharthan and Mandya, 2014).

Most sentence compression approaches focus on deleting words (the words appearing in the compression are words occurring in the input) and therefore only perform limited paraphrasing. As noted by Pitler (2010) and Toutanova et al. (2016) however, the ability to paraphrase is key for the development of abstractive summarisation systems since summaries written by humans often rephrase the original content using paraphrases or synonyms or alternative syntactic constructions. Recent proposals by Rush et al. (2015) and Bingel and Søgaard (2016) address this issue. Rush et al. (2015) proposed a neural model for abstractive compression and summarisation, and Bingel and Søgaard (2016) proposed a structured approach to text simplification which jointly predicts possible compressions and paraphrases.

None of these approaches requires that the input be split into shorter sentences so that both the corpora used, and the models learned, fail to adequately account for the various types of specific rephrasings occurring when a complex sentence is split into several shorter sentences.

Finally, sentence fusion does induce rephrasing as one sentence is produced out of several. However, research in that field is still hampered by the small size of datasets for the task, and the difficulty of generating one (Daume III and Marcu, 2004). Thus, the dataset of Thadani and McKeown (2013) only consists of 1,858 fusion instances of which 873 have two inputs, 569 have three and 416 have four. This is arguably not enough for learning a general Split-and-Rephrase model.

In sum, while work on sentence rewriting has made some contributions towards learning to split and/or to rephrase, the interaction between these two subtasks have never been extensively studied nor are there any corpora available that would support the development of models that can both split and rephrase. In what follows, we introduce such a benchmark and present some baseline models which provide some interesting insights on how to address the Split-and-Rephrase problem.

3 The WEBSPLIT Benchmark

We derive a Split-and-Rephrase dataset from the WEBNLG corpus presented in Gardent et al. (2017).

3.1 The WEBNLG Dataset

In the WEBNLG dataset, each item consists of a set of RDF triples (M) and one or more texts (T_i) verbalising those triples.

An RDF (Resource Description Format) triple is a triple of the form *subject|property|object* where the subject is a URI (Uniform Resource Identifier), the property is a binary relation and the object is either a URI or a literal value such as a string, a date or a number. In what follows, we refer to the sets of triples representing the meaning of a text as its meaning representation (MR). Figure 1 shows three example WEBNLG items with M_1, M_2, M_3 the sets of RDF triples representing the meaning of each item, and $\{T_1^1, T_1^2\}$, $\{T_2\}$ and $\{T_3\}$ listing possible verbalisations of these meanings.

The WEBNLG dataset[4] consists of 13,308 MR-Text pairs, 7049 distinct MRs, 1482 RDF entities and 8 DBpedia categories (Airport, Astronaut, Building, Food, Monument, SportsTeam, University, WrittenWork). The number of RDF triples in MRs varies from 1 to 7. The number of distinct RDF tree shapes in MRs is 60.

3.2 Creating the WEBSPLIT Dataset

To construct the Split-and-Rephrase dataset, we make use of the fact that the WEBNLG dataset (i) associates texts with sets of RDF triples and (ii) contains texts of different lengths and complexity corresponding to different subsets of RDF triples. The idea is the following. Given a WEBNLG MR-Text pair of the form (M, T) where T is a single complex sentence, we search the WEBNLG dataset for a set $\{(M_1, T_1), \ldots, (M_n, T_n)\}$ such that $\{M_1, \ldots, M_n\}$ is a partition of M and $\langle T_1, \ldots, T_n \rangle$ forms a text with more than one sentence. To achieve this, we proceed in three main steps as follows.

Sentence segmentation We first preprocess all 13,308 distinct verbalisations contained in the WEBNLG corpus using the Stanford CoreNLP pipeline (Manning et al., 2014) to segment each verbalisation T_i into sentences.

Sentence segmentation allows us to associate each text T in the WEBNLG corpus with the number of sentences it contains. This is needed to identify complex sentences with no split (the input to the Split-and-Rephrase task) and to know how many sentences are associated with a given set of RDF triples (e.g., 2 triples may be realised by a single sentence or by two). As the CoreNLP sentence segmentation often fails on complex/rare named entities thereby producing unwarranted splits, we verified the sentence segmentations produced by the CoreNLP sentence segmentation module for each WEBNLG verbalisation and manually corrected the incorrect ones.

Pairing Using the semantic information given by WEBNLG RDF triples and the information about the number of sentences present in a WEBNLG text produced by the sentence segmentation step, we produce all items of the form $\langle (M_C, C), \{(M_1, T_1) \ldots (M_n, T_n)\} \rangle$ such that:

- C is a single sentence with semantics M_C.
- $T_1 \ldots T_n$ is a sequence of texts that contains at least two sentences.
- The disjoint union of the semantics $M_1 \ldots M_n$ of the texts $T_1 \ldots T_n$ is the same as the semantics M_C of the complex sentence C. That is, $M_C = M_1 \uplus \ldots \uplus M_n$.

This pairing is made easy by the semantic information contained in the WEBNLG corpus and includes two subprocesses depending on whether complex and split sentences come from the same WEBNLG entry or not.

Within entries. Given a set of RDF triples M_C, a WEBNLG entry will usually contain several alternative verbalisations for M_C (e.g., T_1^1 and T_1^2 in Figure 1 are two possible verbalisations of M_1). We first search for entries where one verbalisation T_C consists of a single sentence and another verbalisation T contains more than one sentence. For such cases, we create an entry of the form $\langle (M_C, T_C), \{(M_C, T)\} \rangle$ such that, T_C is a single sentence and T is a text consisting of more than one sentence. The second example item for WEBSPLIT in Figure 1 presents this case. It uses different verbalisations (T_1^1 and T_1^2) of the same meaning representation M_1 in WEBNLG to construct

[4] We use a version from February 2017 given to us by the authors. A more recent version is available here: http://talc1.loria.fr/webnlg/stories/challenge.html.

<table>
<tr><th></th><th>WebNLG</th></tr>
<tr><td>M_1</td><td><i>{ Birmingham|leaderName|John_Clancy_(Labour_politician), John_Madin|birthPlace|Birmingham, 103_Colmore_Row|architect|John_Madin}</i></td></tr>
<tr><td>T_1^1</td><td>John Clancy is a labour politican who leads Birmingham, where architect John Madin, who designed 103 Colmore Row, was born.</td></tr>
<tr><td>T_1^2</td><td>Labour politician, John Clancy is the leader of Birmingham.
John Madin was born in this city.
He was the architect of 103 Colmore Row.</td></tr>
<tr><td>M_2</td><td><i>{ Birmingham|leaderName|John_Clancy_(Labour_politician)}</i></td></tr>
<tr><td>T_2</td><td>Labour politician, John Clancy is the leader of Birmingham.</td></tr>
<tr><td>M_3</td><td><i>{ John_Madin|birthPlace|Birmingham, 103_Colmore_Row|architect|John_Madin}</i></td></tr>
<tr><td>T_3</td><td>John Madin was born in Birmingham.
He was the architect of 103 Colmore Row.</td></tr>
<tr><th></th><th>WebSplit</th></tr>
<tr><td>$M_C(= M_1)$</td><td><i>{ Birmingham|leaderName|John_Clancy_(Labour_politician), John_Madin|birthPlace|Birmingham, 103_Colmore_Row|architect|John_Madin}</i></td></tr>
<tr><td>$C(= T_1^1)$</td><td>John Clancy is a labour politican who leads Birmingham, where architect John Madin, who designed 103 Colmore Row, was born.</td></tr>
<tr><td>M_2</td><td><i>{ Birmingham|leaderName|John_Clancy_(Labour_politician)}</i></td></tr>
<tr><td>T_2</td><td>Labour politician, John Clancy is the leader of Birmingham.</td></tr>
<tr><td>M_3</td><td><i>{ John_Madin|birthPlace|Birmingham, 103_Colmore_Row|architect|John_Madin}</i></td></tr>
<tr><td>T_3</td><td>John Madin was born in Birmingham.
He was the architect of 103 Colmore Row.</td></tr>
<tr><td>$M_C(= M_1)$</td><td><i>{ Birmingham|leaderName|John_Clancy_(Labour_politician), John_Madin|birthPlace|Birmingham, 103_Colmore_Row|architect|John_Madin}</i></td></tr>
<tr><td>$C(= T_1^1)$</td><td>John Clancy is a labour politican who leads Birmingham, where architect John Madin, who designed 103 Colmore Row, was born.</td></tr>
<tr><td>M_1</td><td><i>{ Birmingham|leaderName|John_Clancy_(Labour_politician), John_Madin|birthPlace|Birmingham, 103_Colmore_Row|architect|John_Madin}</i></td></tr>
<tr><td>T_1^2</td><td>Labour politician, John Clancy is the leader of Birmingham.
John Madin was born in this city.
He was the architect of 103 Colmore Row.</td></tr>
</table>

Figure 1: Example entries from the WebNLG benchmark and their pairing to form entries in the WebSplit benchmark.

a WebSplit item associating the complex sentence (T_1^1) with a text (T_1^2) made of three short sentences.

Across entries. Next we create $\langle (M, C), \{(M_1, T_1) \ldots (M_n, T_n)\}\rangle$ entries by searching for all WebNLG texts C consisting of a single sentence. For each such text, we create all possible partitions of its semantics M_C and for each partition, we search the WebNLG corpus for matching entries i.e., for a set S of (M_i, T_i) pairs such that (i) the disjoint union of the semantics M_i in S is equal to M_C and (ii) the resulting set of texts contains more than one sentence. The first example item for WebSplit in Figure 1 is a case in point. $C(= T_1^1)$ is the single, complex sentence whose meaning is represented by the three triples M. $\langle T_2, T_3\rangle$ is the sequence of shorter texts C is mapped to. And the semantics M_2 and M_3 of these two texts forms a partition over M.

Ordering. For each item $\langle (M_C, C), \{(M_1, T_1) \ldots (M_n, T_n)\}\rangle$ produced in the preceding step, we determine an order on $T_1 \ldots T_n$ as follows. We observed that the WebNLG texts mostly[5] follow the order in which the RDF triples are presented. Since this order corresponds to a left-to-right depth-first traversal of the RDF tree, we use this order to order the sentences in the T_i texts.

3.3 Results

By applying the above procedure to the WebNLG dataset, we create 1,100,166 pairs of the form $\langle (M_C, T_C), \{(M_1, T_1) \ldots (M_n, T_n)\}\rangle$ where T_C is a complex sentence and $T_1 \ldots T_n$ is a sequence of texts with semantics $M_1, \ldots M_n$ expressing the same content M_C as T_C. 1,945 of these pairs were of type "Within entries" and the rest were of type "Across entries". In total, there are 1,066,115 distinct $\langle T_C, T_1 \ldots T_n\rangle$ pairs with 5,546 distinct complex sentences. Complex sentences are associated with 192.23 rephrasings in average (min: 1, max: 76283, median: 16). The number of sentences in the rephrasings varies between 2 and 7 with an average of 4.99. The vocabulary size is 3,311.

[5] As shown by the examples in Figure 1, this is not always the case. We use this constraint as a heuristic to determine an ordering on the set of sentences associated with each input.

4 Problem Formulation

The Split-and-Rephrase task can be defined as follows. Given a complex sentence C, the aim is to produce a simplified text T consisting of a sequence of texts $T_1 \dots T_n$ such that T forms a text of at least two sentences and the meaning of C is preserved in T. In this paper, we proposed to approach this problem in a supervised setting where we aim to maximise the likelihood of T given C and model parameters θ: $P(T|C;\theta)$. To exploit the different levels of information present in the WEBSPLIT benchmark, we break the problem in the following ways:

$$\begin{aligned} P(T|C;\theta) &= \sum_{M_C} P(T|C;M_C;\theta)P(M_C|C;\theta) && (1) \\ &= P(T|C;M_C;\theta), \text{if } M_C \text{ is known.} && (2) \\ &= \sum_{M_{1-n}} \begin{matrix} P(T|C;M_C;M_{1-n};\theta)\times \\ P(M_{1-n}|C;M_C;\theta) \end{matrix} && (3) \end{aligned}$$

where, M_C is the meaning representation of C and M_{1-n} is a set $\{M_1, \dots, M_n\}$ which partitions M_C.

5 Split-and-Rephrase Models

In this section, we propose five different models which aim to maximise $P(T|C;\theta)$ by exploiting different levels of information in the WEBSPLIT benchmark.

5.1 A Probabilistic, Semantic-Based Approach

Narayan and Gardent (2014) describes a sentence simplification approach which combines a probabilistic model for splitting and deletion with a phrase-based statistical machine translation (SMT) and a language model for rephrasing (reordering and substituting words). In particular, the splitting and deletion components exploit the deep meaning representation (a Discourse Representation Structure, DRS) of a complex sentence produced by Boxer (Curran et al., 2007).

Based on this approach, we create a Split-and-Rephrase model (aka HYBRIDSIMPL) by (i) including only the splitting and the SMT models (we do not learn deletion) and (ii) training the model on the WEBSPLIT corpus.

5.2 A Basic Sequence-to-Sequence Approach

Sequence-to-sequence models (also referred to as encoder-decoder) have been successfully applied to various sentence rewriting tasks such as machine translation (Sutskever et al., 2011; Bahdanau et al., 2014), abstractive summarisation (Rush et al., 2015) and response generation (Shang et al., 2015). They first use a recurrent neural network (RNN) to convert a source sequence to a dense, fixed-length vector representation (encoder). They then use another recurrent network (decoder) to convert that vector to a target sequence.

We use a three-layered encoder-decoder model with LSTM (Long Short-Term Memory, (Hochreiter and Schmidhuber, 1997)) units for the Split-and-Rephrase task. Our decoder also uses the local-p attention model with feed input as in (Luong et al., 2015). It has been shown that the local attention model works better than the standard global attention model of Bahdanau et al. (2014). We train this model (SEQ2SEQ) to predict, given a complex sentence, the corresponding sequence of shorter sentences.

The SEQ2SEQ model is learned on pairs $\langle C, T\rangle$ of complex sentences and the corresponding text. It directly optimises $P(T|C;\theta)$ and does not take advantage of the semantic information available in the WEBSPLIT benchmark.

5.3 A Multi-Source Sequence-to-Sequence Approach

In this model, we learn a multi-source model which takes into account not only the input complex sentence but also the associated set of RDF triples available in the WEBSPLIT dataset. That is, we maximise $P(T|C;M_C;\theta)$ (Eqn. 2) and learn a model to predict, given a complex sentence C and its semantics M_C, a rephrasing of C.

As noted by Gardent et al. (2017), the shape of the input may impact the syntactic structure of the corresponding text. For instance, an input containing a path $(X|P_1|Y)(Y|P_2|Z)$ equating the object of a property P_1 with the subject of a property P_2 may favour a verbalisation containing a subject relative ("x V_1 y who V_2 z"). Taking into account not only the sentence C that needs to be rephrased but also its semantics M_C may therefore help learning.

We model $P(T|C;M_C;\theta)$ using a multi-source sequence-to-sequence neural framework (we refer to this model as MULTISEQ2SEQ). The core idea comes from Zoph and Knight (2016) who show that a multi-source model trained on trilingual translation pairs $((f,g),h)$ outperforms sev-

Model	Task	Training Size
HYBRIDSIMPL	Given C, predict T	886,857
SEQ2SEQ	Given C, predict T	886,857
MULTISEQ2SEQ	Given C and M_C, predict T	886,866
SPLIT-MULTISEQ2SEQ	Given C and M_C, predict $M_1 \dots M_n$	13,051
	Given C and M_i, predict T_i	53,470
SPLIT-SEQ2SEQ	Given C and M_C, predict $M_1 \dots M_n$	13,051
	Given M_i, predict T_i	53,470

Table 2: Tasks modelled and training data used by Split-and-Rephrase models.

eral strong single source baselines. We explore a similar "trilingual" setting where f is a complex sentence (C), g is the corresponding set of RDF triples (M_C) and h is the output rephrasing (T).

We encode C and M_C using two separate RNN encoders. To encode M_C using RNN, we first linearise M_C by doing a depth-first left-right RDF tree traversal and then tokenise using the Stanford CoreNLP pipeline (Manning et al., 2014). Like in SEQ2SEQ, we model our decoder with the local-p attention model with feed input as in (Luong et al., 2015), but now it looks at both source encoders simultaneously by creating separate context vector for each encoder. For a detailed explanation of multi-source encoder-decoders, we refer the reader to Zoph and Knight (2016).

5.4 Partitioning and Generating

As the name suggests, the Split-and-Rephrase task can be seen as a task which consists of two subtasks: (i) splitting a complex sentence into several shorter sentences and (ii) rephrasing the input sentence to fit the new sentence distribution. We consider an approach which explicitly models these two steps (Eqn. 3). A first model $P(M_1, \dots, M_n|C; M_C; \theta)$ learns to partition a set M_C of RDF triples associated with a complex sentence C into a disjoint set $\{M_1, \dots, M_n\}$ of sets of RDF triples. Next, we generate a rephrasing of C as follows:

$$
\begin{aligned}
& P(T|C; M_C; M_1, \dots, M_n; \theta) && (4)\\
& \approx P(T|C; M_1, \dots, M_n; \theta) && (5)\\
& = P(T_1, \dots, T_n|C; M_1, \dots, M_n; \theta) && (6)\\
& = \prod_i^n P(T_i|C; M_i; \theta) && (7)
\end{aligned}
$$

where, the approximation from Eqn. 4 to Eqn. 5 derives from the assumption that the generation of T is independent of M_C given $(C; M_1, \dots, M_n)$. We propose a pipeline model to learn parameters θ. We first learn to split and then learn to generate from each RDF subset generated by the split.

Learning to split. For the first step, we learn a probabilistic model which given a set of RDF triples M_C predicts a partition $M_1 \dots M_n$ of this set. For a given M_C, it returns the partition $M_1 \dots M_n$ with the highest probability $P(M_1, \dots, M_n|M_C)$.

We learn this split module using items $\langle (M_C, C), \{(M_1, T_1) \dots (M_n, T_n)\} \rangle$ in the WEBSPLIT dataset by simply computing the probability $P(M_1, \dots, M_n|M_C)$. To make our model robust to an unseen M_C, we strip off named-entities and properties from each RDF triple and only keep the tree skeleton of M_C. There are only 60 distinct RDF tree skeletons, 1,183 possible split patterns and 19.72 split candidates in average for each tree skeleton, in the WEBSPLIT dataset.

Learning to rephrase. We proposed two ways to estimate $P(T_i|C; M_i; \theta)$: (i) we learn a multi-source encoder-decoder model which generates a text T_i given a complex sentence C and a set of RDF triples $M_i \in M_C$; and (ii) we approximate $P(T_i|C; M_i; \theta)$ by $P(T_i|M_i; \theta)$ and learn a simple sequence-to-sequence model which, given M_i, generates a text T_i. Note that as described earlier, M_i's are linearised and tokenised before we input them to RNN encoders. We refer to the first model by SPLIT-MULTISEQ2SEQ and the second model by SPLIT-SEQ2SEQ.

6 Experimental Setup and Results

This section describes our experimental setup and results. We also describe the implementation details to facilitate the replication of our results.

6.1 Training, Validation and Test sets

To ensure that complex sentences in validation and test sets are not seen during training, we split the 5,546 distinct complex sentences in the WEBSPLITdata into three subsets: Training set (4,438, 80%), Validation set (554, 10%) and Test set (554, 10%).

Table 2 shows, for each of the 5 models, a summary of the task and the size of the training corpus. For the models that directly learn to map a complex sentence into a meaning preserving sequence of at least two sentences (HybridSimpl, Seq2Seq and MultiSeq2Seq), the training set consists of 886,857 $\langle C, T \rangle$ pairs with C a complex sentence and T, the corresponding text. In contrast, for the pipeline models which first partition the input and then generate from RDF data (Split-MultiSeq2Seqand Split-Seq2Seq), the training corpus for learning to partition consists of 13,051 $\langle M_C, \langle M_1 \dots M_n \rangle \rangle$ pairs while the training corpus for learning to generate contains 53,470 $\langle M_i, T_i \rangle$ pairs.

6.2 Implementation Details

For all our neural models, we train RNNs with three-layered LSTM units, 500 hidden states and a regularisation dropout with probability 0.8. All LSTM parameters were randomly initialised over a uniform distribution within [-0.05, 0.05]. We trained our models with stochastic gradient descent with an initial learning rate 0.5. Every time perplexity on the held out validation set increased since it was previously checked, then we multiply the current learning rate by 0.5. We performed mini-batch training with a batch size of 64 sentences for Seq2Seq and MultiSeq2Seq, and 32 for Split-Seq2Seq and Split-MultiSeq2Seq. As the vocabulary size of the WebSplit data is small, we train both encoder and decoder with full vocabulary. We randomly initialise word embeddings in the beginning and let the model train them during training. We train our models for 20 epochs and keep the best model on the held out set for the testing purposes. We used the system of Zoph and Knight (2016) to train both simple sequence-to-sequence and multi-source sequence-to-sequence models[6], and the system of Narayan and Gardent (2014) to train our HybridSimpl model.[7]

Model	BLEU	#S/C	#Tokens/S
Source	55.67	1.0	21.11
HybridSimpl	39.97	1.26	17.55
Seq2Seq	48.92	2.51	10.32
MultiSeq2Seq	42.18	2.53	10.69
Split-MultiSeq2Seq	77.27	**2.84**	11.63
Split-Seq2Seq	**78.77**	**2.84**	**9.28**

Table 3: Average BLEU scores for rephrasings, average number of sentences in the output texts (#S/C) and average number of tokens per output sentences (#Tokens/S). Source are the complex sentences from the WebSplit corpus.

6.3 Results

We evaluate all models using multi-reference BLEU-4 scores (Papineni et al., 2002) based on all the rephrasings present in the Split-and-Rephrase corpus for each complex input sentence.[8] As BLEU is a metric for n-grams precision estimation, it is not an optimal metric for the Split-and-Rephrase task (sentences even without any split could have a high BLEU score). We therefore also report on the average number of output simple sentences per complex sentence and the average number of output words per output simple sentence. The first one measures the ability of a system to split a complex sentence into multiple simple sentences and the second one measures the ability of producing smaller simple sentences.

Table 3 shows the results. The high BLEU score for complex sentences (Source) from the WebSplit corpus shows that using BLEU is not sufficient to evaluate splitting and rephrasing. Because the short sentences have many n-grams in common with the source, the BLEU score for complex sentences is high but the texts are made of a single sentence and the average sentence length is high. HybridSimpl performs poorly – we conjecture that this is linked to a decrease in semantic parsing quality (DRSs) resulting from complex named entities not being adequately recognised. The simple sequence-to-sequence model does not perform very well neither does the multi-source model trained on both complex sentences and their semantics. Typically, these two models often produce non-meaning preserving outputs (see example in Table 4) for input of longer length. In contrast, the two partition-and-generate models outperform all other models by a wide mar-

[6] We used the code available at https://github.com/isi-nlp/Zoph_RNN.

[7] We used the code available at https://github.com/shashiongithub/Sentence-Simplification-ACL14.

[8] We used https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl to estimate BLEU scores against multiple references.

SOURCE	Alan Shepard was born in New Hampshire and he served as the Chief of the Astronaut Office .
HYBRIDSIMPL	Alan Shepard was born in New Hampshire and he served as of the the chief astronaut office .
SEQ2SEQ	Alan Shepard 's occupation was a test pilot . Alan Shepard was born in New Hampshire . Alan Shepard was born on Nov 18 , 1923 .
MULTISEQ2SEQ	Alan Shepard served as a test pilot . Alan Shepard 's birth place was New Hampshire .
SPLIT-MULTISEQ2SEQ	Alan Shepard served as the Chief of the Astronaut Office . Alan Shepard was born in New Hampshire .
SPLIT-SEQ2SEQ	Alan Shepard served as the Chief of the Astronaut Office . Alan Shepard 's birth place was New Hampshire .

Table 4: Example outputs from different models.

gin. This suggests that the ability to split is key to a good rephrasing: by first splitting the input semantics into smaller chunks, the two partition-and-generate models permit reducing a complex task (generating a sequence of sentences from a single complex sentence) to a series of simpler tasks (generating a short sentence from a semantic input).

Unlike in neural machine translation setting, multi-source models in our setting do not perform very well. SEQ2SEQ and SPLIT-SEQ2SEQ outperform MULTISEQ2SEQ and SPLIT-MULTISEQ2SEQ respectively, despite using less input information than their counterparts. The multi-source models used in machine translation have as a multi-source, two translations of the same content (Zoph and Knight, 2016). In our approach, the multi-source is a complex sentence and a set of RDF triples, e.g., $(C; M_C)$ for MULTISEQ2SEQ and $(C; M_i)$ for SPLIT-MULTISEQ2SEQ. We conjecture that the poor performance of multi-source models in our case is due either to the relatively small size of the training data or to a stronger mismatch between RDF and complex sentence than between two translations.

Table 4 shows an example output for all 5 systems highlighting the main differences. HYBRIDSIMPL's output mostly reuses the input words suggesting that the SMT system doing the rewriting has limited impact. Both the SEQ2SEQ and the MULTISEQ2SEQ models “hallucinate” new information (“served as a test pilot”, “born on Nov 18, 1983”). In contrast, the partition-and-generate models correctly render the meaning of the input sentence (SOURCE), perform interesting rephrasings (“X was born in Y” $\rightarrow$ “X's birth place was Y”) and split the input sentence into two.

7 Conclusion

We have proposed a new sentence simplification task which we call “Split-and-Rephrase”. We have constructed a new corpus for this task which is built from readily-available data used for NLG (Natural Language Generation) evaluation. Initial experiments indicate that the ability to split is a key factor in generating fluent and meaning preserving rephrasings because it permits reducing a complex generation task (generating a text consisting of at least two sentences) to a series of simpler tasks (generating short sentences). In future work, it would be interesting to see whether and if so how, sentence splitting can be learned in the absence of explicit semantic information in the input.

Another direction for future work concerns the exploitation of the extended WebNLG corpus. While the results presented in this paper use a version of the WebNLG corpus consisting of 13,308 MR-Text pairs, 7049 distinct MRs and 8 DBpedia categories, the current WebNLG corpus encompasses 43,056 MR-Text pairs, 16,138 distinct MRs and 15 DBpedia categories. We plan to exploit this extended corpus to make available a correspondingly extended WEBSPLIT corpus, to learn optimised Split-and-Rephrase models and to explore sentence fusion (converting a sequence of sentences into a single complex sentence).

Acknowledgements

We thank Bonnie Webber and Annie Louis for early discussions on the ideas presented in the paper. We thank Rico Sennrich for directing us to multi-source NMT models. This work greatly benefited from discussions with the members of the Edinburgh NLP group. We also thank the three anonymous reviewers for their comments to improve the paper. The research presented in this paper was partially supported by the H2020 project SUMMA (under grant agreement 688139) and the French National Research Agency within the framework of the WebNLG Project (ANR-14-CE24-0033).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of ACL*.

Regina Barzilay and Kathleen R McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of ACL*.

Joachim Bingel and Anders Søgaard. 2016. Text simplification as tree labeling. In *Proceedings of ACL*.

Yvonne Margaret Canning. 2002. *Syntactic simplification of Text*. Ph.D. thesis, University of Sunderland.

John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying text for language-impaired readers. In *Proceedings of EACL*.

Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of COLING*.

Raman Chandrasekar and Bangalore Srinivas. 1997. Automatic induction of rules for text simplification. *Knowledge-Based Systems*, 10(3):183–190.

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of COLING*.

William Coster and David Kauchak. 2011. Learning to simplify sentences using wikipedia. In *Proceedings of Monolingual Text-To-Text Generation*.

James R Curran, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with C&C and Boxer. In *Proceedings of ACL*.

Hal Daume III and Daniel Marcu. 2004. Generic sentence fusion is an ill-defined summarization task. Technical report, DTIC.

Jan De Belder and Marie-Francine Moens. 2010. Text simplification for children. In *Proceedings of the SIGIR Workshop on Accessible Search Systems*.

Mark Dras. 1999. *Tree adjoining grammar and the reluctant paraphrasing of text*. Ph.D. thesis, Macquarie University, Australia.

Pablo Ariel Duboue and Jennifer Chu-Carroll. 2006. Answering the question you wish they had asked: The impact of paraphrasing for question answering. In *Proceedings of NAACL-HLT*.

Florence Duclaye, François Yvon, and Olivier Collin. 2003. Learning paraphrases to improve a question-answering system. In *Proceedings of the EACL Workshop on Natural Language Processing for Question Answering Systems*.

Katja Filippova. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of COLING*.

Katja Filippova, Enrique Alfonseca, Carlos Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with LSTMs. In *Proceedings of EMNLP*.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *Proceedings of INLG*.

Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles, and Benjamin Van Durme. 2011. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proceedings of EMNLP*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for nlg micro-planning. In *Proceedings of ACL*.

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

Kentaro Inui, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. 2003. Text simplification for reading assistance: A project note. In *Proceedings of the workshop on Paraphrasing*.

Tomáš Jelínek. 2014. Improvements to dependency parsing using automatic simplification of data. In *Proceedings of LREC*.

Hans Kamp. 1981. A theory of truth and semantic representation. In *Formal Methods in the Study of Language*, volume 1, pages 277–322. Mathematisch Centrum.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization-step one: Sentence compression. In *Proceedings of AAAI-IAAI*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of EACL*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL System Demonstrations*.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Linguistics*, 37(1):197–230.

Kathleen McKeown, Sara Rosenthal, Kapil Thadani, and Coleman Moore. 2010. Time-efficient creation of an accurate sentence fusion corpus. In *Proceedings of NAACL-HLT*.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proceedings of ACL*.

Shashi Narayan and Claire Gardent. 2016. Unsupervised sentence simplification using deep semantics. In *Proceedings of INLG*.

Shashi Narayan, Siva Reddy, and Shay B. Cohen. 2016. Paraphrase generation from Latent-Variable PCFGs for semantic parsing. In *Proceedings of INLG*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Emily Pitler. 2010. Methods for sentence compression. Technical report, University of Pennsylvania.

Chris Quirk, Chris Brockett, and William B Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of EMNLP*.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *CoRR*, abs/1503.02364.

Advaith Siddharthan. 2002. An architecture for a text simplification system. In *Proceedings of Language Engineering Conference*. IEEE Computer Society.

Advaith Siddharthan. 2010. Complex lexico-syntactic reformulation of sentences using typed dependency representations. In *Proceedings of INLG*.

Advaith Siddharthan. 2011. Text simplification using typed dependencies: A comparison of the robustness of different generation strategies. In *Proceedings of ENLG*.

Advaith Siddharthan and Angrosh Mandya. 2014. Hybrid text simplification using synchronous dependency grammars with hand-written and automatically harvested rules. In *Proceedings of EACL*.

Advaith Siddharthan, Ani Nenkova, and Kathleen McKeown. 2004. Syntactic simplification for improving content selection in multi-document summarization. In *Proceedings of COLING*.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of ICML*.

Kapil Thadani and Kathleen McKeown. 2013. Supervised sentence fusion with single-stage inference. In *Proceedings of IJCNLP*.

Masaru Tomita. 1985. *Efficient Parsing for Natural Language: A Fast Algorithm for Practical Systems*. The Springer International Series in Engineering and Computer Science. Springer US.

Kristina Toutanova, Chris Brockett, Ke M. Tran, and Saleema Amershi. 2016. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. In *Proceedings of EMNLP*.

David Vickrey and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of ACL-HLT*.

Willian Massami Watanabe, Arnaldo Candido Junior, Vinícius Rodriguez Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Aluísio. 2009. Facilita: reading assistance for low-literacy readers. In *Proceedings of ACM*.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of EMNLP*.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of ACL*.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2010. Paraphrase generation as monolingual translation: Data and evaluation. In *Proceedings of INLG*.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of EMNLP*.

Shiqi Zhao, Haifeng Wang, Ting Liu, and Sheng Li. 2008. Pivot approach for extracting paraphrase patterns from bilingual corpora. In *Proceedings of ACL*.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of COLING*.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of NAACL-HLT*.

Neural Response Generation via GAN with an Approximate Embedding Layer[*]

Zhen Xu[1], **Bingquan Liu**[1], **Baoxun Wang**[2], **Chengjie Sun**[1], **Xiaolong Wang**[1],
Zhuoran Wang[2] and **Chao Qi**[2]
[1]School of Computer Science and Technology,
Harbin Institute of Technology, Harbin, China
[2]Tricorn (Beijing) Technology Co., Ltd, Beijing, China
[1]{zxu,bqliu,cjsun,wangxl}@insun.hit.edu.cn
[2]{wangbaoxun, wangzhuoran, qichao}@trio.ai

Abstract

This paper presents a Generative Adversarial Network (GAN) to model single-turn short-text conversations, which trains a sequence-to-sequence (Seq2Seq) network for response generation simultaneously with a discriminative classifier that measures the differences between human-produced responses and machine-generated ones. In addition, the proposed method introduces an approximate embedding layer to solve the non-differentiable problem caused by the sampling-based output decoding procedure in the Seq2Seq generative model. The GAN setup provides an effective way to avoid non-informative responses (a.k.a "safe responses"), which are frequently observed in traditional neural response generators. The experimental results show that the proposed approach significantly outperforms existing neural response generation models in diversity metrics, with slight increases in relevance scores as well, when evaluated on both a Mandarin corpus and an English corpus.

1 Introduction

After achieving remarkable successes in Machine Translation (Sutskever et al., 2014; Cho et al., 2014), neural networks with the encoder-decoder architectures (a.k.a sequence-to-sequence models, Seq2Seq) have been proven to be a functioning method to model short-text conversations (Vinyals and Le, 2015; Shang et al., 2015), where the corresponding task is often called Neural Response Generation. The advantage of applying Seq2Seq models to conversation generation is that the training procedure can be performed end-to-end in an unsupervised manner, based on human-generated conversational utterances (typically query-response pairs mined from social networks). One of the potential applications of such neural response generators is to improve the capability of existing conversational interfaces (informally also known as chatbots) by enabling them to go beyond predefined tasks and chat with human users in an open domain.

However, previous research has indicated that naïve implementations of Seq2Seq based conversation models tend to suffer from the so-called "safe response" problem (Li et al., 2016a), i.e. such models tend to generate non-informative responses that can be associated to most queries, e.g. "I don't know", "I think so", etc. This is due to the fundamental nature of statistical models, which fit sufficiently observed examples better than insufficiently observed ones. Concretely, the space of open-domain conversations is so large that in any sub-sample of it (i.e. a training set), the distribution of most pieces of information are relatively much sparser when compared to safe response patterns. Furthermore, since a safe response can be of relevance to a large amount of diverse queries, a statistical learner will tend to minimize its empirical risk in the response generation process by capturing those safe responses if naïve relevance-oriented loss metrics are employed.

Frequent occurrences of safe responses can dramatically reduce the attractiveness of a chat agent, which therefore should be avoided to the best extent possible when designing the learning algorithms. The pathway to achieve this purpose is to seek a more expressive model with better capacity that can take relevance and diversity (or informativeness) into account simultaneously

[*]The work was done when the first author was an intern at Tricorn (Beijing) Technology Co., Ltd.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 617–626
Copenhagen, Denmark, September 7–11, 2017.

when modelling the underlying distribution of human conversations.

Generative Adversarial Nets (GANs) (Goodfellow et al., 2014; Chen et al., 2016) offers an effective architecture of jointly training a generative model and a discriminative classifier to generate sharp and realistic images. This architecture could also potentially be applied to conversational response generation to relieve the safe response problem, where the generative part can be an Seq2Seq-based model that generates response utterances for given queries, and the discriminative part can evaluate the quality of the generated utterances from diverse dimensions according to human-produced responses. However, unlike the image generation problems, training such a GAN for text generation here is not straightforward. The decoding phase of the Seq2Seq model usually involves sampling discrete words from the predicted distributions, which will be fed into the training of the discriminator. The sampling procedure is non-differentiable, and will therefore break the back-propagation.

To the best of our knowledge, Reinforcement Learning (RL) is first introduced to address the above problem (Li et al., 2017; Yu et al., 2017), where the score predicted by a discriminator was used as the reinforcement to train the generator, yielding a hybrid model of GAN and RL. But to train the RL phrase, Li et al. (2017) introduced two approximations for reward computing at each action (word) selection step, including a Markov Chain Monte Carlo (MCMC) sampling method and a partial utterance scoring approach. It has been stated in their work that the former approach is time-consuming and the latter one will result in lower performance due to the overfitting problem caused by adding a large amount of partial utterances into the training set. Nevertheless, we also want to argue that, besides the time complexity issue of MCMC, RL itself is not an optimal choice either. As shown in our experimental results in Section 5.1, a more elegant design of an end-to-end differentiable GAN can significantly increase the model's performance in this text generation task.

In this paper, we propose a novel variant of GAN for conversational response generation, which introduces an approximate embedding layer to replace the sampling-based decoding phase, such that the entire model is continuous and differentiable. Empirical experiments are conducted based on two datasets, of which the results show that the proposed method significantly outperforms three representative existing approaches in both relevance and diversity oriented automatic metrics. In addition, human evaluations are carried out as well, demonstrating the potential of the proposed model.

2 Related Work

Inspired by recent advances in Neural Machine Translation (NMT), Ritter et al. (2011) and Vinyals and Le (2015) have shown that single-turn short-text conversations can be modelled as a generative process trained using query-response pairs accumulated on social networks. Earlier works focused on paired word sequences only, while Zhou et al. (2016) and Iulian et al. (2017) have demonstrated that the comprehensibility of the generated responses can benefit from multi-view training with respect to words, coarse tokens and utterances. Moreover, Sordoni et al. (2015) proposed a context-aware response generation model that goes beyond single-turn conversations.

In addition, attention mechanisms were introduced to Seq2Seq-based models to capture topic and dialog focus information by Shang et al. (2015) and Chen et al. (2017), which had been proven to be helpful for improving query-response relevance (Wu et al., 2016). Additional features such as persona information (Li et al., 2016b) and latent semantics (Zhou et al., 2017; Serban et al., 2017) have also been proven beneficial within this context.

When compared to previous work, this paper is focused on single-turn conversation modeling, and employs a GAN to yield informative responses.

3 Building a Conversational Response Generator via GAN

3.1 Notations

Let $\mathcal{D} = \{(q_i, r_i)\}_{i=1}^{N}$ be a set of N single-turn human-human conversations, where $q_i = (w_{q_i,1}, \ldots, w_{q_i,t}, \ldots, w_{q_i,m})$ is a query, $r_i = (w_{r_i,1}, \ldots, w_{r_i,t}, \ldots, w_{r_i,n})$ stands for the response to q_i, and $w_{q_i,t}$ and $w_{r_i,t}$ denote the t-th words in q_i and r_i, respectively. This paper aims to learn a generative model $G(r|q)$ based on a discriminator D that can predict informative responses with good diversity for arbitrary input queries.

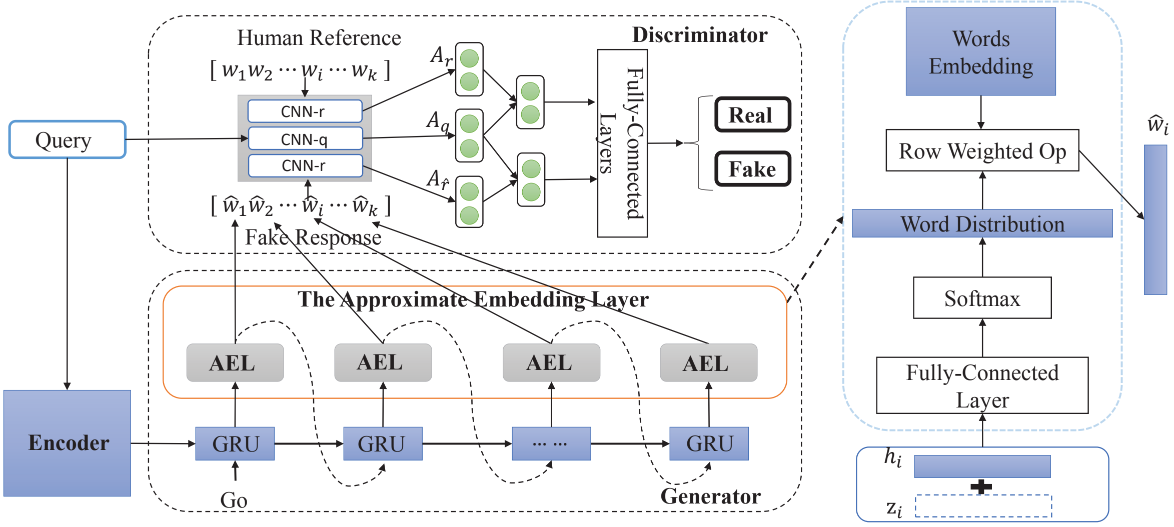


Figure 1: The Framework of GAN for the Response Generator.

3.2 Model Overview

We name the proposed model Generative Adversarial Network with an Approximate Embedding Layer (GAN-AEL), of which Figure 1 illustrates the overall framework. Generally speaking, the whole framework consists of a response generator G, a discriminator D and an embedding approximation layer that connects the G and the D. We explain each of the components in detail as follows. The generator adopts the Gated Recurrent Unit (GRU) (Cho et al., 2014) based encoder-decoder architecture, where the encoder projects the input query (a discrete word sequence) into a real-valued vector, on which the output will be generated conditionally in the decoding process, activated by a starting signal (denoted as "Go" in Figure 1). An approximate embedding layer is designed to guarantee that the response generation procedure is continuous and differentiable, serving as an interface for the discriminator to propagate its loss to the generator. The Convolutional Neural Network (CNN) based discriminator is attached on top of the approximation layer, which aims to distinguish the fake responses output by the approximation layer and the corresponding human-generated references, conditioned on the input query. The judgement of the CNN can be propagated to the Seq2Seq generator through the proposed approximate embedding layer, and forces the generator to be fine-tuned to produce more attractive results.

The proposed GAN framework possesses several advantages over existing conversational response generation models. Firstly, both the generator and the discriminator are conditioned on the input query, which guarantees the relevance of the generated responses. Secondly, the discriminator enforces the generator to produce a response according to the true distribution in better granularity, such that the state of promoting safe responses is leaped out. Thirdly, the approximation layer yields a smooth connection between the generator and the discriminator, avoiding the non-differentiable discrete sampling process.

3.3 Pre-training the Generator by MLE

In our proposed encoder-decoder framework, both the encoder and the generator (i.e. the decoder) G is composed of GRU (Cho et al., 2014) units, which is designed to generate responses $r = \{w_{r,1}, w_{r,2}, \cdots, w_{r,K}\}$ conditioned on an input query $q = \{w_{q,1}, w_{q,2}, \cdots, w_{q,J}\}$. For a given query-response pair (q, r), the target is to maximum the conditional probability $p(r|q)$ in the generative process. Concretely, in this model, q is firstly encoded into a vector representation q_v by the GRU-based encoder as shown in Figure 1, which is actually the last hidden state of the encoder. Then the generator estimates the probability of each word occurring in r conditioned on q_v. Hence $p(r|q)$ can be formulated as follows:

$$p(r|q) = \prod_{t=1}^{K} p(w_{r,t}|q_v, w_{r,1}, \cdots, w_{r,t-1}) \quad (1)$$

Taking the logarithm of the probabilities for effective computation, the generator is trained by optimising the Maximum Likelihood Estimation (MLE) objective defined as:

$$\frac{1}{|\mathcal{D}|}\sum_{(q,r)\in\mathcal{D}}\sum_{t=1}^{K}\log p(w_{r,t}|q_v, w_{r,1},\cdots,w_{r,t-1}) \tag{2}$$

Note here, we need to pre-train the generator using Equation 2 as the loss function to guarantee the generator to produce grammatical utterances. Otherwise, the discriminator will tend to learn a rule with ease to distinguish human-produced utterances from those ungrammatical responses generated in the early stages of the training phase, which would cause the failure of the training in satisfying Nash Equilibrium (Goodfellow et al., 2014).

3.4 The Approximate Embedding Layer

In order to smoothly connect the output layer of the generator to the input layer of the discriminator to yield an end-to-end differentiable GAN, one needs to solve the following critical problem. The output of the generator is a sequence of discrete words, which is usually sampled from the distributions predicted by the decoder's RNN units in the Softmax layer, and is non-differentiable.

Since afterward those words will be projected into embedding vectors to feed the CNN-based discriminator, we introduce an embedding approximation layer to merge the generation process of the decoder and the word embedding phrase of the discriminator. This can be done by multiplying the word probabilities in the distributions obtained from the decoder's Softmax layer to the corresponding word vectors, to directly yield an approximately vectorized representation of the generated word sequences for further convolutional computations in the discriminative process. This approximation is based on the assumption that ideally the word distributions should be trained to reasonably approach the one-hot representations of the discrete words.

The structure of the approximation layer is illustrated on the right-hand side of Figure 1. Concretely, the approximation layer takes the output h_i of the generator and a random noise z_i as the input, and reuses the word projection layer (pre-trained in the standard generator) to estimate the probability distribution of w_i. Note that, the noise z_i added to h_i forms a latent feature for the word embedding approximation process to enforce the diversity of the generated responses. The overall word embedding approximation is computed as:

$$\hat{e}_{w_i} = \sum_{j=1}^{V} e_j \cdot \text{Softmax}(W_p(h_i + z_i) + b_p)_j \tag{3}$$

where W_p and b_p are the weight and bias parameters of the word projection layer, respectively, and h_i is the hidden representation of word w_i, from the decoding procedure of the generator G, which is computed as:

$$h_i = g(h_{i-1}, \hat{e}_{w_{i-1}}) \tag{4}$$

where $g(\cdot)$ is the standard GRU inference step in G (Cho et al., 2014).

3.5 Pre-training the CNN-based Discriminator

CNN has been proven to be an appropriate classifier for many NLP tasks, such as sentence classification (Kim, 2014) and matching (Hu et al., 2014). Therefore, in this paper we adopt a CNN-based discriminator as shown in Figure 1.

For the convenience of further discussions, we introduce $\hat{r}$ to denote the underlying (distributional) fake response produced by the decoder. In other words, $\hat{r}$ stands for a sequence of word distributions projected from the hidden layers of the decoder RNN, based on which one would sample the output response utterance in a traditional Seq2Seq generator. The detailed architecture of the discriminator is described as follows. Firstly, the input of the discriminator consist of the word embedding vector sequence V_q for a given query q and the word embedding vector sequence V_r for its human-produced response r, as well as the approximate word embedding vector sequence $V_{\hat{r}}$ produced by the approximate embedding layer for the corresponding fake response $\hat{r}$. All the word embedding vector sequences here are zero-padded or truncated to a same fixed length. After this, two CNNs with shared parameters are employed to encode V_r and $V_{\hat{r}}$ into higher-level abstractions, respectively. In addition, a separate CNN is used to abstract V_q in a similar way. We denote such abstraction layers (i.e. the max-pooling layers before the fully-connected layers) in the above CNNs as A_r, $A_{\hat{r}}$ and A_q, corresponding to r, $\hat{r}$ and q, respectively. Finally, we concatenate A_q

to A_r and $A_{\hat{r}}$ separately, and feed the resulting vectors to their respective fully-connected layers, as illustrated in Figure 1. Here, we make the two fully-connected layers share common parameters and predict probabilities $D(r|q)$ and $D(\hat{r}|q)$, respectively, for r and $\hat{r}$ being true responses of the given q.

In practice, when the Seq2Seq generative network G is pre-trained, we also pre-train the above discriminator D by maximising the following objective function:

$$D_{loss} = \log D(r|q) + \log(1 - D(\hat{r}|q)) \qquad (5)$$

with the parameters of G frozen, before the adversarial training procedure described in Section 3.6.

3.6 Adversarial Training of the Generator

After the pre-training of the generator G and the discriminator D as described above, the entire network is trained adversarially. Concretely, we iteratively train G and D, where at each iteration, the parameters of the non-training network will be frozen. The following tricks are utilised in the adversarial training phase to achieve better convergence. Firstly, when training G, we replace the objective function given in Equation 5 with the l_2-loss between A_r and $A_{\hat{r}}$, to maintain a reasonable scale of the gradient. Secondly, we freeze the parameters of the encoder network and the projection layer of the decoder network, but only tune the parameters of decoder's hidden layers. This is based on the assumption that, in principle, after the pre-training, the encoder network is sufficiently effective to represent the entire input utterance, while the projection layer of the decoder is also adequate to decode words from its hidden states. Therefore, the adversarial training here is to adjust the "wording strategy" of the generative model, i.e. the way it organises the semantic contents during the decoding (or in other words, the way it realises the hidden states). Preliminary experiments show that this trick significantly improves the grammaticalness of the generated responses.

The gradient of the generator can be computed as:

$$\begin{aligned} \nabla_{g_{D,G}(\theta_G)} &= \frac{\partial G_{loss}}{\partial V_{\hat{r}}} \frac{\partial V_{\hat{r}}}{\partial \theta_G} \\ &= \frac{\partial G_{loss}}{\partial V_{\hat{r}}} \frac{\partial V_{\hat{r}}}{\partial G} \frac{\partial G}{\partial \theta_G} \end{aligned} \qquad (6)$$

where θ_G denotes the active parameters of the generator G, $G_{loss} = \|A_r - A_{\hat{r}}\|$ and $g_{D,G}(\cdot)$ stands for the inference step of the entire GAN. It can be seen that the feedback signals from D can be propagated to G effectively through the approximate embedding layer, which connects G and D smoothly, and avoids the discrete sampling procedure.

4 Experiment Setup

4.1 Datasets

We test our model on two datasets: **Baidu Tieba** and **OpenSubtitles** (Lison and Tiedemann, 2016). The Baidu Tieba dataset is composed of single-turn conversations collected from the threads of Baidu Tieba[1], of which the utterance length ranging from 3 to 30 words. The OpenSubtitles dataset contains movie scripts organised by characters, where we follow Li et al. (2016a) to retain subtitles containing 5-50 words in the following experiments. From each of the two datasets, we sample 5,000,000 unique single-turn conversations as the training data, 200,000 additional unique pairs for validation, and another 10,000 as the test set.

4.2 Baselines

To illustrate the performance of the proposed model, we introduce three existing approaches as baselines.

- **Seq2Seq:** the standard sequence-to-sequence model (Sutskever et al., 2014).
- **MMI-anti:** a Seq2Seq model with a Maximum Mutual Information (MMI) criterion (implemented as an anti-language model) (Li et al., 2016a) in the decoding process, which reduces the probability of generating "safe responses".
- **Adver-REGS:** another adversarial strategy proposed by Li et al. (2017)[2], which links the generator and the discriminator together with a reinforcement learning framework, and takes the discriminator's output probability as the reward to train the generator.

[1] https://tieba.baidu.com/index.html

[2] The codes can be accessed at https://github.com/jiweil/Neural-Dialogue-Generation/tree/master/Adversarial

4.3 Evaluation Metrics

For automatic evaluations, the following commonly accepted metrics are employed. Note here, the goal of our model is to obtain responses not only semantically relevant to the corresponding queries, but also of good diversity and novelty. Therefore, in this work, embedding-based metrics (Liu et al., 2016) are adopted to evaluate semantic the relevance between queries and their corresponding generated responses, while **dist-1**, **dist-2** (Li et al., 2016a) are used as diversity measures. In addition, we also introduce a **Novelty** measure as detailed below.

Relevance Metrics: The following three word embedding based metrics[3] are used to compute the semantic relevance of two utterances. The **Greedy** metric is to greedily match words in two given utterances based on the cosine similarities of their embeddings, and to average the obtained scores (Rus and Lintean, 2012). Alternatively, an utterance representation can be obtained by averaging the embeddings of all the words in that utterance, of which the cosine similarity gives the **Average** metric (Mitchell and Lapata, 2008). In addition, one can also achieve an utterance representation by taking the largest extreme values among the embedding vectors of all the words it contains, before computing the cosine similarities between utterance vectors, which yields the **Extreme** metric (Forgues et al., 2014).

Diversity Metrics: To measure the informativeness and diversity of the generated responses, we follow the **dist-1** and **dist-2** metrics proposed by Li et al. (2016a) and Chen et al. (2017), and introduce a **Novelty** metric. The **dist-1** (**dist-2**) is defined as the number of unique unigrams (bigrams for dist-2). A common drawback of dist-1 and dist-2 is that in the computation, less informative words (such as "I", "is", etc.) are considered equally with those more informative ones. Therefore, in this paper, we define an extra **Novelty** metric, which is the number of infrequent words observed in the generated responses. Here we take all the words except the top 2000 most frequent ones in the vocabulary as infrequent words. Note here, the dist-1 and Novelty values are normalised by utterance length, and dist-2 is normalised by the total number of bigrams in the generated response.

Human Evaluation: To evaluate the performance of our model from human perspectives, this paper conducts a human subject experiement by comparing the responses generated by Adver-REGS (which is one of the most competitive existing approaches) with those by the proposed model. Three experienced annotators are invited to evaluate 200 groups of examples. In the evaluation, for every given query, the annotators will see 10 generated responses from each model. Since the proposed method aims at improving the diversity of the responses generated by Seq2Seq models, while maintaining their relevance to the input queries, we ask the annotators to evaluate the diversity performance of the two systems only if there is no obvious difference between the performance of their relevance. This experimental setting is due to the following two reasons. Firstly, it is difficult to judge a systems diversity based on one single response (Li et al., 2016a; Zhou et al., 2017). Secondly, the practical deployment of a chat-oriented conversational system will usually decode an N-best list of candidate responses, from which it random samples the final reply. Considering that all the annotators use Mandarin as their first language, the above evaluation is only done on the Tieba dataset.

4.4 Hyperparameters & Training Strategies

Hyperparameter Settings: The hyperparameters of the networks used in all the experiments below are described as follows. The vocabulary sizes for Tieba and OpenSubtitles are truncated to 100,000 and 150,000, respectively. The dimensions of word embedding vectors are set to 100 for Tieba and 300 for OpenSubtitles. The size of the hidden layers in the generator is set to 200 in the all experiments on both datasets. We experiments subsets of {1,2,3,4} for the filter sizes of the CNNs, and fixed the filter number to 128. As shown in subsection 5.3, CNNs with filter sizes {1,2} are the best choice here. Max-pooling is used in all the CNN settings here. The noise Z is sampled from a normal distribution with 0 mean and 0.1 variance.

Training Strategies: To train the proposed GAN, the parameters of the generator G are initialised based on the pre-training mentioned in subsection 3.3, while those of the discriminator D are randomly initialised. The adversarial training

[3] The implementation of all these metrics follows the code at https://github.com/julianser/hed-dlg-truncated/tree/master/Evaluation.

Model	Relevance			Diversity		
	Average	**Greedy**	**Extreme**	**Dist-1**	**Dist-2**	**Novelty**
Seq2Seq	0.720	0.614	0.571	0.0037	0.0121	0.0102
MMI-anti	0.713	0.592	0.552	0.0127	0.0495	0.0250
Adver-REGS	0.722	0.660	0.574	0.0153	0.0658	0.0392
GAN-AEL	**0.736**	**0.689**	**0.580**	**0.0214**	**0.0963**	**0.0635**

Table 1: Relevance and diversity evaluation on the Tieba dataset.

Model	Relevance			Diversity		
	Average	**Greedy**	**Extreme**	**Dist-1**	**Dist-2**	**Novelty**
Seq2Seq	0.719	0.578	0.505	0.0054	0.0141	0.0045
MMI-anti	0.710	0.569	0.499	0.0175	0.0586	0.0097
Adver-REGS	0.726	0.590	0.507	0.0223	0.0725	0.0147
GAN-AEL	**0.734**	**0.621**	**0.514**	**0.0296**	**0.0955**	**0.0216**

Table 2: Relevance and diversity evaluation on the OpenSubtitles dataset.

starts from pre-training D with the parameters of G fixed. After this, G and D will be trained iteratively with different learning rates, which are 0.0001 for D and 0.00002 for G. In addition, we update D at a frequency of every 5 batches instead of every single batch.

5 Experimental Results

5.1 Automatic Evaluation & Analysis

From Table 1 and 2, it can be observed that the proposed GAN-AEL model outperforms the baselines on both datasets in all metrics, especially for the diversity oriented scores. The improvements can be explained from the following two angles.

a) Since a vanilla Seq2Seq model does not take diversity, novelty or informativeness into account, the discriminator tends to capture such information to distinguish model-generated responses and human responses. By backpropagating the discriminator's feedback to the generator, the adversarially trained generator gains significantly better performance in such aspects. On the other hand, the relevance is also retained during the adversarial training, as one can imagine that the human produced references given to the discriminator are usually semantically highly relevant to the corresponding queries.

b) The proposed approximation layer is an effective way to couple the response generator and the discriminator. Through this differentiable component, the loss of the discriminator is properly propagated to the generator and guide the tuning of the latter's parameters.

It can also be seen from the results that the performance of all the models on the three semantic relevance oriented metrics are comparable to each other. This implies that all the models, including the baseline methods and the proposed model, have the capability to generate responses of reasonable relevance to given queries, which satisfies the primary goal of the response generation task. It further suggests that the Seq2Seq architecture works properly in modelling the semantics of entire utterances. Nevertheless, although the decoder mechanism can select topic-relevant words to construct responses based on the given query, the limitation of naïve Seq2Seq models tend to yield less diverse or informative outputs.

Furthermore, when compared to Adver-REGS, the proposed GAN-AEL gains 30%-60% relative improvement in the dist-1, dist-2 and novelty metrics on both datasets, which indicates that coupling the generator and the discriminator with a differentiable component is a more preferable methodology for text generation tasks, and is a meaningful analogy to standard GANs for image generation. Interestingly, all the models achieve significantly higher novelty scores on the Tieba dataset than on the OpenSubtitle dataset. This is due to the difference of the coverages of high-frequency words in the two corpora. Concretely, since we exclude top 2,000 most frequent words when computing the novelty scores on both datasets, which covers 70% and 82% of the words in Tieba and OpenSubtitle respectively, it is more likely to observe novel words on the Tieba data.

In addition, it can be seen that GAN-AEL improves the greedy score to a much greater extent than the average and extreme scores, which further suggests that the responses generated by GAN-AEL are more informative. Concretely, the calculations of the average and extreme scores may be dominated by generic non-informative words. By contrast, since the greedy metric is computed based on a (simple and greedy) word-wise semantic alignments between two utterances, the influence of those generic words will be reduced.

5.2 Human Evaluation Results

Table 3 gives the human evaluation results, which indicates that the proposed GAN-AEL is more preferable than Adver-REGS from human perspectives. This again implies that the approximate embedding layer is more effective in propagating the discriminator's feedback to the generator than the reinforcement learning mechanism of (Li et al., 2017). The result is statistically significant with $p < 0.01$ according to sign test.

GAN-AEL vs Adver-REGS		
Wins	Losses	Ties
0.61	0.13	0.26

Table 3: Evaluations of GAN-AEL and Adver-REGS based on human subjects,

5.3 The Influence of the Discriminator to Adversarial Training

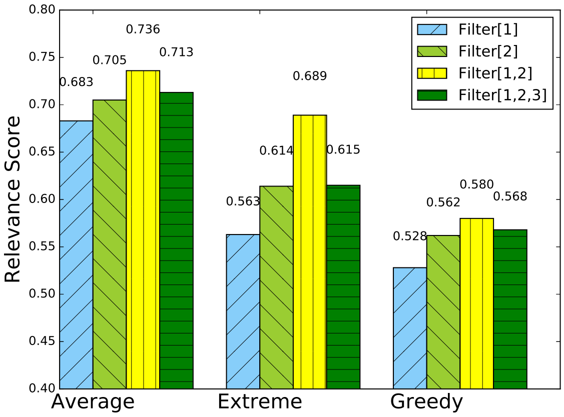


Figure 2: Relevance scores of GAN-AEL on the Tieba corpus with respect to different CNN window sizes.

The discriminator plays an important role in the adversarial training process, which determines whether the GAN model converges to a Nash Equilibrium (Chen et al., 2016). We conduct a set of experiments to explore the influence of the discriminator's capacities to the adversarial training. Figure 2 shows the relevance scores with respect to different convolution window sizes for the CNN discriminator, where "Filter[x]" denotes the CNN with its convolution window(s) set to x.

It can be found that the discriminator with "Filter[1, 2]" achieves the best performance. Two facts based on the principle of GAN could be taken to explain this observation: On one hand, a discriminator with too low capacity (such as "Filter[1]") is less capable in distinguishing human responses from generated ones, which will backpropagate inappropriate signals that misleads the generator. On the other hand, if the capacity of the discriminator is too high (such as "Filter[1, 2, 3]"), in the adversarial training, the training of the discriminator will converge too fast before the generator being sufficiently trained (Durugkar et al., 2016). A similar phenomenon is also observed on the OpenSubtitles dataset. Since the relevance between queries and their corresponding responses is essential in this task, discriminators with "Filter[1,2]" are adopted in the experiments reported in subsections 5.1 and 5.2.

5.4 Case Study

To demonstrate the intuitive performance of the proposed model in comparison with the naïve Seq2Seq model, we provide some example cases in Figure 3, where for each query the response is the top hypothesis obtained via beam search. Especially, we show that when the vanilla approach generates safe responses such as "I don't know what to do", the adversarially trained method can still yield more informative outputs. Furthermore, when both models produces informative responses, the utterance generated by GAN-AEL tends to be of more interestingness, which is a reflection of novelty and diversity.

6 Conclusion

In this paper, we proposed a GAN framework to model single-turn short-text conversations. An approximation embedding layer is introduced to force the entire network differentiable, which significantly overcomes the drawbacks found in

Example responses generated based on the OpenSubtitle data	Example responses generated based on the Baidu Tieba data
Query: You swore an oath when you put that uniform on. **Seq2Seq:** I don't know what to do. **GAN-AEL:** I think I had a good reason.	**Query:** 你喜欢猫还是狗？Do you like cats or dogs? **Seq2Seq:** 喜欢养猫。I Like cats. **GAN-AEL:** 我喜欢金毛。I like golden retriever.
Query: Entire town knows your son is a goon. **Seq2Seq:** What do you mean? **GAN-AEL:** His life is really a sad story.	**Query:** 你像奥巴马的妻子。You look like Obama's wife. **Seq2Seq:** 哈哈哈哈哈哈。Haha... **GAN-AEL:** 我是大众脸。I have a common face.
Query: Turn his face away from me. **Seq2Seq:** I don't know what to do. **GAN-AEL:** Let me explain to you about the mistake.	**Query:** 我不光搭讪，还请吃饭呢。 I wish to invite you to dinner rather than chat up only. **Seq2Seq:** 我也想吃。I want to have dinner too. **GAN-AEL:** 我是胖子，减肥。I am fat, and I am on a diet.

Figure 3: Example responses generated by Seq2Seq and GAN-AEL.

the previous RL-based attempts (Li et al., 2017). The superiority of the proposed method has been demonstrated by empirical experiments based on both automatic evaluation metrics and human judgements. Further explorations of GAN-based techniques to model contextual information in dialogue problems will be addressed in our future research.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. We also thank Dr. Deyuan Zhang and Dr. Xin Wang for their great help. This research is partially supported by National Natural Science Foundation of China (No.61572151, No.61602131, No.61672192) and the National High Technology Research and Development Program ("863" Program) of China (No.2015AA015405).

References

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems 29*, pages 2172–2180.

Xing Chen, Wu Wei, Wu Yu, Liu Jie, Huang Yalou, Zhou Ming, and Ma Wei-Ying. 2017. Topic aware neural response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3351–3357.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. 2016. Generative multi-adversarial networks. *Proceedings of the 4th International Conference on Learning Representations (ICLR)*.

Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *NIPS, Modern Machine Learning and Natural Language Processing Workshop*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems 27*, pages 2672–2680.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27*, pages 2042–2050.

Serban Iulian, Vlad, Klinger Tim, Tesauro Gerald, Talamadupula Kartik, Zhou Bowen, Bengio Yoshua, and C. Courville Aaron. 2017. Multiresolution recurrent neural networks: An application to dialogue response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3288–3294.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and William B. Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2122–2132.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the 2011 Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 583–593.

Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 157–162.

Iulian Vlad Serban, Sordoni Alessandro, Lowe Ryan, Charlin Laurent, Pineau Joelle, C. Courville Aaron, and Bengio Yoshua. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3295–3301.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 1577–1586.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 14th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 196–205.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Bowen Wu, Baoxun Wang, and Hui Xue. 2016. Ranking responses oriented to conversational relevance in chat-bots. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 652–662.

Lantao Yu, Weinan Zhang, and and Yong Yu Jun Wang. 2017. SeqGAN: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, volume 31.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017. Mechanism-aware neural machine for dialogue response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3400–3407.

Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view response selection for human-computer conversation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 372–381.

A Hybrid Convolutional Variational Autoencoder for Text Generation

Stanislau Semeniuta[1] **Aliaksei Severyn**[2] **Erhardt Barth**[1]
[1]Universität zu Lübeck, Institut für Neuro- und Bioinformatik
{stas,barth}@inb.uni-luebeck.de
[2]Google Research Europe
severyn@google.com

Abstract

In this paper we explore the effect of architectural choices on learning a variational autoencoder (VAE) for text generation. In contrast to the previously introduced VAE model for text where both the encoder and decoder are RNNs, we propose a novel hybrid architecture that blends fully feed-forward convolutional and deconvolutional components with a recurrent language model. Our architecture exhibits several attractive properties such as faster run time and convergence, ability to better handle long sequences and, more importantly, it helps to avoid the issue of the VAE collapsing to a deterministic model.

1 Introduction

Generative models of texts are currently at the cornerstone of natural language understanding enabling recent breakthroughs in machine translation (Bahdanau et al., 2014; Wu et al., 2016), dialogue modelling (Serban et al., 2016), abstractive summarization (Rush et al., 2015), etc.

Currently, RNN-based generative models hold state-of-the-art results in both unconditional (Józefowicz et al., 2016; Ha et al., 2016) and conditional (Vinyals et al., 2014) text generation. At a high level, these models represent a class of autoregressive models that work by generating outputs sequentially one step at a time where the next predicted element is conditioned on the history of elements generated thus far.

Variational autoencoders (VAE), recently introduced by (Kingma and Welling, 2013; Rezende et al., 2014), offer a different approach to generative modeling by integrating stochastic latent variables into the conventional autoencoder architecture. The primary purpose of learning VAE-based generative models is to be able to generate realistic examples as if they were drawn from the input data distribution by simply feeding noise vectors through the decoder. Additionally, the latent representations obtained by applying the encoder to input examples give a fine-grained control over the generation process that is harder to achieve with more conventional autoregressive models. Similar to compelling examples from image generation, where it is possible to condition generated human faces on various attributes such as hair, skin color and style (Yan et al., 2015; Larsen et al., 2015), in text generation it should be possible to also control various attributes of the generated sentences, such as, for example, sentiment or writing style.

While training VAE-based models seems to pose little difficulty when applied to the tasks of generating natural images (Bachman, 2016; Gulrajani et al., 2016) and speech (Fraccaro et al., 2016), their application to natural text generation requires additional care (Bowman et al., 2016; Miao et al., 2015). As discussed by Bowman et al. (2016), the core difficulty of training VAE models is the collapse of the latent loss (represented by the KL divergence term) to zero. In this case the generator tends to completely ignore latent representations and reduces to a standard language model. This is largely due to the high modeling power of the RNN-based decoders which with sufficiently small history can achieve low reconstruction errors while not relying on the latent vector provided by the encoder.

In this paper, we propose a novel VAE model for texts that is more effective at forcing the decoder to make use of latent vectors. Contrary to existing work, where both encoder and decoder layers are LSTMs, the core of our model is a feed-forward architecture composed of one-dimensional convolutional and deconvolutional (Zeiler et al., 2010) layers. This choice of architecture helps to gain

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 627–637
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

more control over the KL term, which is crucial for training a VAE model. Given the difficulty of generating long sequences in a fully feed-forward manner, we augment our network with an RNN language model layer. To the best of our knowledge, this paper is the first work that successfully applies deconvolutions in the decoder of a latent variable generative model of natural text. We empirically verify that our model is easier to train than its fully recurrent alternative, which, in our experiments, fails to converge on longer texts. To better understand why training VAEs for texts is difficult we carry out detailed experiments, discuss optimization difficulties, and propose effective ways to address them. Finally, we demonstrate that sampling from our model yields realistic texts.

2 Related Work

So far, the majority of neural generative models of text are built on the autoregressive assumption (Larochelle and Murray, 2011; van den Oord et al., 2016). Such models assume that the current data element can be accurately predicted given sufficient history of elements generated thus far. The conventional RNN based language models fall into this category and currently dominate the language modeling and generation problem in NLP. Neural architectures based on recurrent (Józefowicz et al., 2016; Zoph and Le, 2016; Ha et al., 2016) or convolutional decoders (Kalchbrenner et al., 2016; Dauphin et al., 2016) provide an effective solution to this problem.

A recent work by Bowman et al. (2016) tackles language generation problem within the VAE framework (Kingma and Welling, 2013; Rezende et al., 2014). The authors demonstrate that with some care it is possible to successfully learn a latent variable generative model of text. Although their model is slightly outperformed by a traditional LSTM (Hochreiter and Schmidhuber, 1997) language model, their model achieves a similar effect as in computer vision where one can (i) sample realistic sentences by feeding randomly generated novel latent vectors through the decoder and (ii) linearly interpolate between two points in the latent space. Miao et al. (2015) apply VAE to bag-of-words representations of documents and the answer selection problem achieving good results on both tasks. Yang et al. (2017) discuss a VAE consisting of RNN encoder and CNN decoder so that the decoder's receptive field is limited. They demonstrate that this allows for a better control of KL and reconstruction terms. Hu et al. (2017) build a VAE for text generation and design a cost function that encourages interpretability of the latent variables. Zhang et al. (2016), Serban et al. (2016) and Zhao et al. (2017) apply VAE to sequence-to-sequence problems, improving over deterministic alternatives. Chen et al. (2016) propose a hybrid model combining autoregressive convolutional layers with the VAE. The authors make an argument based on the Bit-Back coding (Hinton and van Camp, 1993) that when the decoder is powerful enough the best thing for the encoder to do is to make the posterior distribution equivalent to the prior. While they experiment on images, this argument is very relevant to the textual data. A recent work by Bousquet et al. (2017) approaches VAEs and GANs from the optimal transport point of view. The authors show that commonly known blurriness of samples from VAEs trained on image data are a necessary property of the model. While the implications of their argument to models combining latent variables and autoregressive layers trained on non-image data are still unclear, the argument supports the hypothesis of Chen et al. (2016) that difficulty of training a hybrid model is not caused by a simple optimization difficulty but rather may be a more principled issue.

Various techniques to improve training of VAE models where the total cost represents a trade-off between the reconstruction cost and KL term have been used so far: KL-term annealing and input dropout (Bowman et al., 2016; Sønderby et al., 2016), imposing structured sparsity on latent variables (Yeung et al., 2016) and more expressive formulations of the posterior distribution (Rezende and Mohamed, 2015; Kingma et al., 2016). A work by (Mescheder et al., 2017) follows the same motivation and combines GANs and VAEs allowing a model to use arbitrary complex formulations of both prior and posterior distributions. In Section 3.4 we propose another efficient technique to control the trade-off between KL and reconstruction terms.

3 Model

In this section we first briefly explain the VAE framework of Kingma and Welling (2013), then describe our hybrid architecture where the feed-

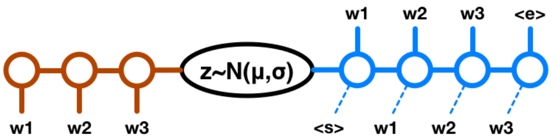


Figure 1: LSTM VAE model of (Bowman et al., 2016)

forward part is composed of a fully convolutional encoder and a decoder that combines deconvolutional layers and a conventional RNN. Finally, we discuss optimization recipes that help VAE to respect latent variables, which is critical training a model with a meaningful latent space and being able to sample realistic sentences.

3.1 Variational Autoencoder

The VAE is a recently introduced latent variable generative model, which combines variational inference with deep learning. It modifies the conventional autoencoder framework in two key ways. Firstly, a deterministic internal representation $\mathbf{z}$ (provided by the encoder) of an input $\mathbf{x}$ is replaced with a posterior distribution $q(\mathbf{z}|\mathbf{x})$. Inputs are then reconstructed by sampling $\mathbf{z}$ from this posterior and passing them through a decoder. To make sampling easy, the posterior distribution is usually parametrized by a Gaussian with its mean and variance predicted by the encoder. Secondly, to ensure that we can sample from any point of the latent space and still generate valid and diverse outputs, the posterior $q(\mathbf{z}|\mathbf{x})$ is regularized with its KL divergence from a prior distribution $p(\mathbf{z})$. The prior is typically chosen to be also a Gaussian with zero mean and unit variance, such that the KL term between posterior and prior can be computed in closed form (Kingma and Welling, 2013). The total VAE cost is composed of the reconstruction term, i.e., negative log-likelihood of the data, and the KL regularizer:

$$\begin{aligned} J_{vae} = & KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \\ & -\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log\ p(\mathbf{x}|\mathbf{z})] \end{aligned} \quad (1)$$

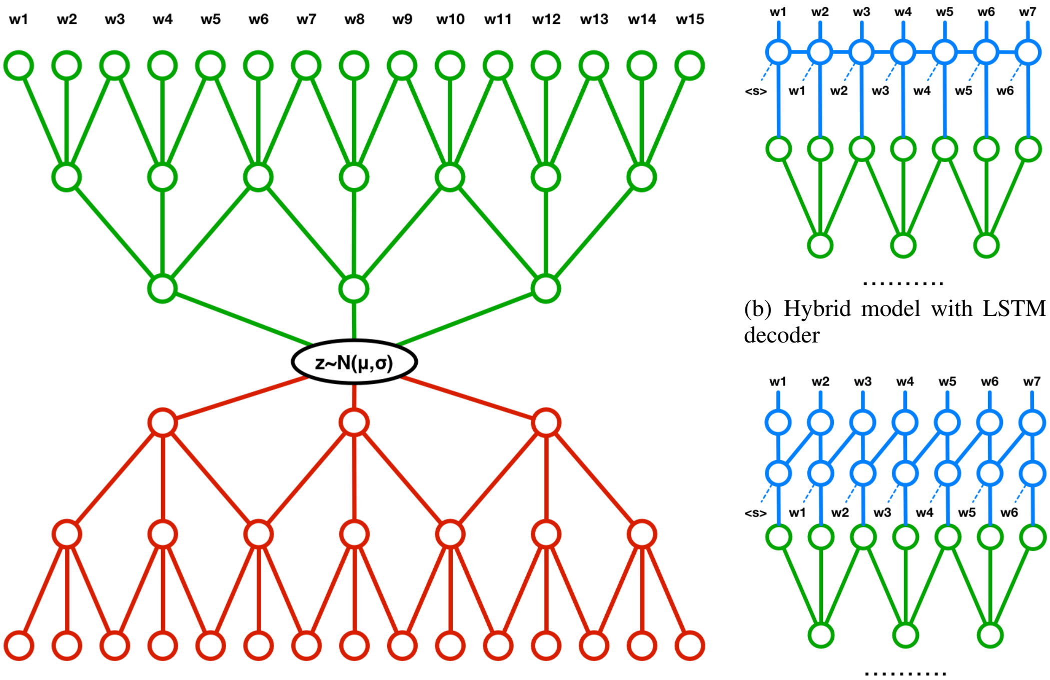
Kingma and Welling (2013) show that the loss function from Eq (1) can be derived from the probabilistic model perspective and it is an upper bound on the true negative likelihood of the data.

One can view a VAE as a traditional Autoencoder with some restrictions imposed on the internal representation space. Specifically, using a sample from the $q(\mathbf{z}|\mathbf{x})$ to reconstruct the input instead of a deterministic $\mathbf{z}$, forces the model to map an input to a region of the space rather than to a single point. The most straight-forward way to achieve a good reconstruction error in this case is to predict a very sharp probability distribution effectively corresponding to a single point in the latent space (Raiko et al., 2014). The additional KL term in Eq (1) prevents this behavior and forces the model to find a solution with, on one hand, low reconstruction error and, on the other, predicted posterior distributions close to the prior. Thus, the decoder part of the VAE is capable of reconstructing a sensible data sample from every point in the latent space that has non-zero probability under the prior. This allows for straightforward generation of novel samples and linear operations on the latent codes. Bowman et al. (2016) demonstrate that this does not work in the fully deterministic Autoencoder framework . In addition to regularizing the latent space, KL term indicates how much information the VAE stores in the latent vector.

Bowman et al. (2016) propose a VAE model for text generation where both encoder and decoder are LSTM networks (Figure 1). We will refer to this model as LSTM VAE in the remainder of the paper. The authors show that adapting VAEs to text generation is more challenging as the decoder tends to ignore the latent vector (KL term is close to zero) and falls back to a language model. Two training tricks are required to mitigate this issue: (i) KL-term annealing where its weight in Eq (1) gradually increases from 0 to 1 during the training; and (ii) applying dropout to the inputs of the decoder to limit its expressiveness and thereby forcing the model to rely more on the latent variables. We will discuss these tricks in more detail in Section 3.4. Next we describe a deconvolutional layer, which is the core element of the decoder in our VAE model.

3.2 Deconvolutional Networks

A deconvolutional layer (also referred to as transposed convolutions (Gulrajani et al., 2016) and fractionally strided convolutions (Radford et al., 2015)) performs spatial up-sampling of its inputs and is an integral part of latent variable generative models of images (Radford et al., 2015; Gulrajani et al., 2016) and semantic segmentation algorithms (Noh et al., 2015). Its goal is to perform an “inverse” convolution operation and increase spatial size of the input while decreasing the number of feature maps. This operation can be viewed as



(a) Fully feed-forward component of our VAE model

(b) Hybrid model with LSTM decoder

(c) Hybrid model with ByteNet decoder

Figure 2: Illustrations of our proposed models.

a backward pass of a convolutional layer and can be implemented by simply switching the forward and backward passes of the convolution operation. In the context of generative modeling based on global representations, the deconvolutions are typically used as follows: the global representation is first linearly mapped to another representation with small spatial resolution and large number of feature maps. A stack of deconvolutional layers is then applied to this representation, each layer progressively increasing spatial resolution and decreasing the amount of feature channels. The output of the last layer is an image or, in our case, a text fragment. A notable example of such a model is the deep network of (Radford et al., 2015) trained with adversarial objective. Our model uses a similar approach but is instead trained with the VAE objective.

There are two primary motivations for choosing deconvolutional layers instead of the dominantly used recurrent ones: firstly, such layers have extremely efficient GPU implementations due to their fully parallel structure. Secondly, feed-forward architectures are typically easier to optimize than their recurrent counterparts, as the number of back-propagation steps is constant and potentially much smaller than in RNNs. Both points become significant as the length of the generated text increases. Next, we describe our VAE architecture that blends deconvolutional and RNN layers in the decoder to allow for better control over the KL-term.

3.3 Hybrid Convolutional-Recurrent VAE

Our model is composed of two relatively independent modules. The first component is a standard VAE where the encoder and decoder modules are parametrized by convolutional and deconvolutional layers respectively (see Figure 2(a)). This architecture is attractive for its computational efficiency and simplicity of training.

The other component is a recurrent language model consuming activations from the deconvolutional decoder concatenated with the previous output characters. We consider two flavors of recurrent functions: a conventional LSTM network (Figure 2(b)) and a stack of masked convolutions also known as the ByteNet decoder from Kalchbrenner et al. (2016) (Figure 2(c)). The primary reason for having a recurrent component in the decoder is to capture dependencies between elements of the text sequences – a hard task for a fully feed-forward architecture. Indeed, the conditional distribution $P(\mathbf{x}|\mathbf{z}) = P(x_1, \ldots, x_n|\mathbf{z})$ of generated sentences cannot be richly represented with a feed-forward network. Instead, it factorizes as: $P(x_1, \ldots, x_n|\mathbf{z}) = \prod_i P(x_i|\mathbf{z})$ where components are independent of each other and are conditioned only on $\mathbf{z}$. To minimize the reconstruction cost the model is forced to encode every detail of a text fragment. A recurrent language model instead models the full joint distribution of output sequences without having to make independence assumptions $P(x_1, \ldots, x_n|\mathbf{z}) = \prod_i P(x_i|x_{i-1}, \ldots, x_1, \mathbf{z})$. Thus, adding a recurrent layer on top of our fully feed-forward encoder-decoder architecture relieves it from encoding every aspect of a text fragment into the latent vector and allows it to instead focus on more high-level semantic and stylistic features.

Note that the feed-forward part of our model is different from the existing fully convolutional approaches of Dauphin et al. (2016) and Kalchbrenner et al. (2016) in two respects: firstly, while being fully parallelizable during training, these models still require predictions from previous time steps during inference and thus behave as a variant of recurrent networks. In contrast, expansion of the z vector is fully parallel in our model (except for the recurrent component). Secondly, our model down- and up-samples a text fragment during processing while the existing fully convolutional decoders do not. Preserving spatial resolution can be beneficial to the overall result, but comes at a higher computational cost. Lastly, we note that our model imposes an upper bound on the size of text samples it is able to generate. While it is possible to model short texts by adding special padding characters at the end of a sample, generating texts longer than certain thresholds is not possible by design. This is not an unavoidable restriction, since the model can be extended to generate variable sized text fragments by, for example, variable sized latent codes. These extensions however are out of scope of this work.

3.4 Optimization Difficulties

The addition of the recurrent component results in optimization difficulties that are similar to those described by Bowman et al. (2016). In most cases the model converges to a solution with a vanishingly small KL term, thus effectively falling back to a conventional language model. Bowman et al. (2016) have proposed to use input dropout and KL term annealing to encourage their model to encode meaningful representations into the $\mathbf{z}$ vector. We found that these techniques also help our model to achieve solutions with non-zero KL term.

KL term annealing can be viewed as a gradual transition from conventional deterministic Autoencoder to a full VAE. In this work we use linear annealing from 0 to 1. We have experimented with other schedules but did not find them to have a significant impact on the final result. As long as the KL term weight starts to grow sufficiently slowly, the exact shape and speed of its growth does not seem to affect the overall result. We have found the following heuristic to work well: we first run a model with KL weight fixed to 0 to find the number of iterations it needs to converge. We then configure the annealing schedule to start after the unregularized model has converged and last for no less than 20% of that amount.

While helping to regularize the latent vector, input dropout tends to slow down convergence. We propose an alternative technique to encourage the model to compress information into the latent vector: in addition to the reconstruction cost computed on the outputs of the recurrent language model, we also add an auxiliary reconstruction term computed from the activations of the last deconvolutional layer:

$$\begin{aligned} J_{aux} &= -\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log\ p(\mathbf{x}|\mathbf{z})] \\ &= -\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\sum_t \log\ p(x_t|\mathbf{z})]. \end{aligned} \tag{2}$$

Since at this layer the model does not have access to previous output elements it needs to rely on the $\mathbf{z}$ vector to produce a meaningful reconstruction. The final cost minimized by our model is:

$$J_{hybrid} = J_{vae} + \alpha J_{aux} \tag{3}$$

where α is a hyperparameter, J_{aux} is the intermediate reconstruction term and J_{vae} is the bound from Eq (1). Expanding the two terms from Eq (3) gives:

$$\begin{aligned} J_{hybrid} &= KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \\ &-\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\sum_t \log\ p(x_t|\mathbf{z}, x_{<t})] \\ &-\alpha\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\sum_t \log p(x_t|\mathbf{z})]. \end{aligned} \tag{4}$$

The objective function from Eq (4) puts a mild constraint on the latent vector to produce features useful for historyless reconstruction. Since the autoregressive part reuses these features, it also improves the main reconstruction term. We are thus able to encode information in the latent vector without hurting expressiveness of the decoder.

One can view the objective function in Eq 4 as a joint objective for two VAEs: one only feed-forward, as in Figure 2(a), and the other combining feed-forward and recurrent parts, as in Figures 2(b) and 2(c), that partially share parameters. Since the feed-forward VAE is incapable of producing reasonable reconstructions without making use of the latent vector, the full architecture also gains access to the latent vector through shared parameters. We note that this trick comes at a cost of worse result on the density estimation task, since part of the parameters of the full model are trained to optimize an objective that does not capture all the dependencies that exist in the textual data. However, the gap between purely deterministic LM and our model is small and easily controllable by the α hyperparameter. We refer the reader to Figure 4 for quantitative results regarding the effect of α on the performance of our model on the LM task.

4 Experiments

We use KL term annealing and input dropout when training the LSTM VAE models from Bowman et al. (2016) and KL term annealing and regularized objective function from Eq (3) when training our models. All models were trained with the Adam optimization algorithm (Kingma and Ba, 2014) with decaying learning rate. We use Layer Normalization (Ba et al., 2016) in LSTM layers and Batch Normalization (Ioffe and Szegedy, 2015) in convolutional and deconvolutional layers. To make our results easy to reproduce we have released the source code of all our experiments[1].

Data. Our first task is character-level language generation performed on the standard Penn Treebank dataset (Marcus et al., 1993). One of the goals is to test the ability of the models to successfully learn the representations of long sequences. For training, fixed-size data samples are selected from random positions in the standard training and validation sets.

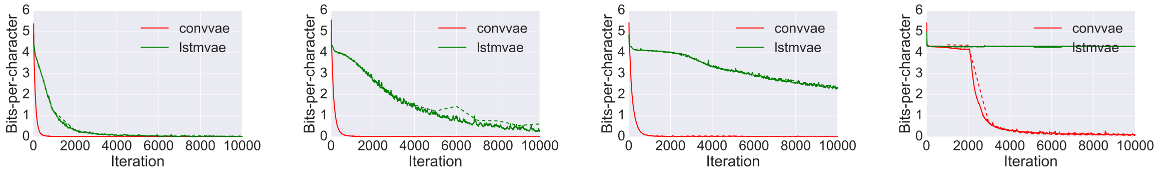
4.1 Comparison with LSTM VAE

Historyless decoding. We start with an experiment where the decoder is forced to ignore the history and has to rely fully on the latent vector. By conditioning the decoder only on the latent vector $\mathbf{z}$ we can directly compare the expressiveness of the compared models. For the LSTM VAE model historyless decoding is achieved by using the dropout on the input elements with the dropout rate equal to 1 so that its decoder is only conditioned on the $\mathbf{z}$ vector and, implicitly, on the number tokens generated so far. We compare it to our fully-feedforward model without the recurrent layer in the decoder (Figure 2(a)). Both networks are parametrized to have comparable number of parameters.

To test how well both models can cope with the stochasticity of the latent vectors, we minimize only the reconstruction term from Eq. (1). This is equivalent to a pure autoencoder setting with stochastic internal representation and no regularization of the latent space. This experiment corresponds to an initial stage of training with KL term annealing when its weight is set to 0. We pursue two goals with this experiment: firstly, we investigate how do the two alternative encoders behave in the beginning of training and establish a lower bound on the quality of the reconstructions. Secondly, we attempt to put the Bit Back coding argument from Chen et al. (2016) in context. The authors assume the encoder to be powerful enough to produce a good representation of the data. One interpretation of this argument applied to textual data is that factorizing the joint probability as $p(\mathbf{x}) = \prod_t p(x_t|x_{<t})$ provides the model with a sufficiently powerful decoder that does not need the latent variables. However, our experimental results suggest that LSTM encoder may not be a sufficiently expressive encoder for VAEs for textual data, potentially making the argument inapplicable.

The results are presented in Figure 3. Note that when the length of input samples reaches 30 characters, the historyless LSTM autoencoder fails to fit the data well, while the convolutional architecture converges almost instantaneously. The results appear even worse for LSTMs on sequences of 50 characters. To make sure that this effect is not caused by optimization difficulties, i.e. exploding gradients (Pascanu et al., 2013), we have searched over learning rates, gradient clipping thresholds and sizes of LSTM layers but were only able to get results comparable to those shown in Figure 3. Note that LSTM networks make use of Layer Nor-

[1] https://github.com/stas-semeniuta/textvae



(a) 10 characters (b) 20 characters (c) 30 characters (d) 50 characters

Figure 3: Training curves of LSTM autoencoder and our model on samples of different length. Solid and dashed lines show training and validation curves respectively. Note that the model exhibits little to no overfitting since the validation curve follows the training one almost perfectly.

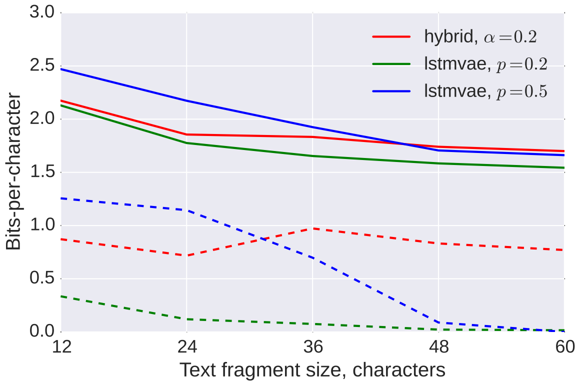


Figure 4: The full cost (solid lines) and its KL component (dashed lines) in bits-per-character of our Hybrid model trained with 0.2 α hyper-parameter vs. LSTM based VAE trained with 0.2 and 0.5 input dropout, measured on validation partition.

malization (Ba et al., 2016) which has been shown to make training of such networks easier. These results suggest that our model is easier to train than the LSTM-based model, especially for modeling longer pieces of text. Additionally, our model is computationally faster by a factor of roughly two, since we run only one recurrent network per sample and time complexity of the convolutional part is negligible in comparison.

Decoding with history. We now move to a case where the decoder is conditioned on both the latent vector and previous output elements. In these experiments we pursue two goals: firstly, we verify whether the results obtained on the historyless decoding task also generalize to a less restricted case. Secondly, we study how well the models cope with stochasticity introduced by the latent variables. Note that we do not attempt to improve the state-of-the-art result on the Language Modeling task but instead focus on providing an approach capable of generating long and diverse sequences. We experiment on the task to obtain a detailed picture of how are our model and LSTM VAE affected by various choices and compare the two models, focusing on how effective is the encoder at producing meaningful latent vector. However, we note that our model performs fairly well on the LM task and is only slightly worse than purely deterministic Language Model, trained in the same environment, and is comparable to the one of Bowman et al. (2016) in this regard.

We fix input dropout rates at 0.2 and 0.5 for LSTM VAE and use auxiliary reconstruction loss (Section 3.4) with 0.2 weight in our Hybrid model. The bits-per-character scores on differently sized text samples are presented in Figure 4. As discussed in Section 3.1, the KL term value indicates how much information the network stores in the latent vector. We observe that the amount of information stored in the latent vector by our model and the LSTM VAE is comparable when we train on short samples and largely depends on hyper-parameters α and p. When the length of a text fragment increases, LSTM VAE is able to put less information into the latent vector (i.e., the KL component is small) and for texts longer than 48 characters, the KL term drops to almost zero while for our model the ratio between KL and reconstruction terms stays roughly constant. This suggests that our model is better at encoding latent representations of long texts since the amount of information in the latent vector does not decrease as the length of a text fragment grows. In contrast, there is a steady decline of the KL term of the LSTM VAE model. This result is consistent with our findings from the historyless decoding experiment. Note that in both of these experiments the LSTM VAE model fails to produce meaningful latent vectors with inputs over 50 characters long. This further suggests that our Hybrid model en-

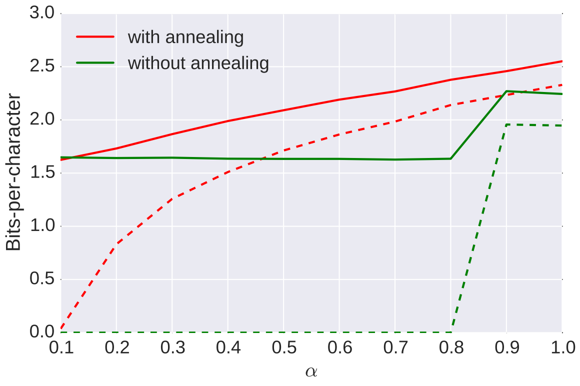


Figure 5: The full cost (solid line) and the KL component (dashed line) of our Hybrid model with LSTM decoder trained with various α, with and without KL term weight annealing, measured on the validation partition.

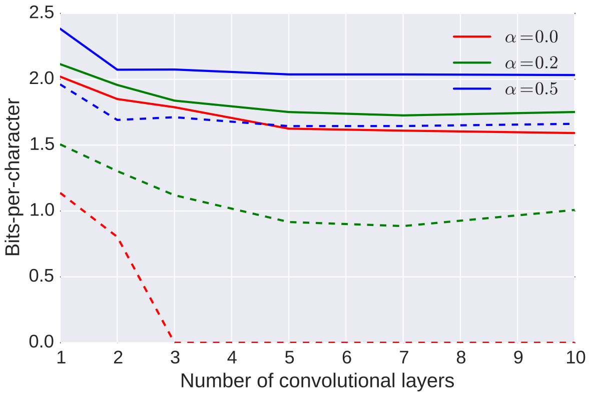


Figure 6: The full cost (solid line) and the KL component (dashed line) of our Hybrid model with ByteNet decoder trained with various number of convolutional layers, measured on the validation partition

codes long texts better than the LSTM VAE.

4.2 Controlling the KL term

We study the effect of various training techniques that help control the KL term which is crucial for training a generative VAE model.

Aux cost weight. First, we provide a detailed view of how optimization tricks discussed in Section 3.4 affect the performance of our Hybrid model. Figure 5 presents results of our model trained with different values of α from Eq. (3). Note that the inclusion of the auxiliary reconstruction loss slightly harms the bound on the likelihood of the data but helps the model to rely more on the latent vector as α grows. A similar effect on model's bound was observed by Bowman et al. (2016): increased input dropout rates force their model to put more information into the $\mathbf{z}$ vector but at the cost of increased final loss values. This is a trade-off that allows for sampling outputs in the VAE framework. Note that our model can find a solution with non-trivial latent vectors when trained with the full VAE loss provided that the α hyper-parameter is large enough. Combining it with KL term annealing helps to find non-zero KL term solutions at smaller α values.

Receptive field. The goal of this experiment is to study the relationship between the KL term values and the expressiveness of the decoder. Without KL term annealing and input dropout, the RNN decoder in LSTM VAE tends to completely ignore information stored in the latent vector and essentially falls back to an RNN language model. To have a full control over the receptive field size of the recurrent component in our decoder, we experiment with masked convolutions (Figure 2(c)), which is similar to the decoder in ByteNet model from Kalchbrenner et al. (2016). We fix the size of the convolutional kernels to 2 and do not use dilated convolutions and skip connections as in the original ByteNet.

The resulting receptive field size of the recurrent layer in our decoder is equal to $N + 1$ characters, where N is the number of convolutional layers. We vary the number of layers to find the amount of preceding characters that our model can consume without collapsing the KL term to zero.

Results of these experiments are presented in Figure 6. Interestingly, with the receptive field size larger than 3 and without the auxiliary reconstruction term from Eq. (3) ($\alpha = 0$) the KL term collapses to zero and the model falls back to a pure language model. This suggests that the training signal received from the previous characters is much stronger than that from the input to be reconstructed. Using the auxiliary reconstruction term, however, helps to find solutions with non-zero KL term component irrespective of receptive field size. Note that increasing the value of α results in stronger values of KL component. This is consistent with the results obtained with LSTM decoder in Figure 5.

@userid @userid @userid @userid @userid ...
I want to see you so much @userid #FollowMeCam ...
@userid @userid @userid @userid @userid ...
Why do I start the day today?
@userid thanks for the follow back
no matter what I'm doing with my friends they are so cute
@userid Hello How are you doing
I wanna go to the UK tomorrow!! #feelinggood #selfie #instago
@userid @userid I'll come to the same time and it was a good day too xx

Table 1: Random sample tweets generated by LSTM VAE (top) and our Hybrid model (bottom).

	Rec	KL
LSTM, $p = 0.2$	67.4	1.0
LSTM, $p = 0.5$	77.1	2.1
LSTM, $p = 0.8$	93.7	3.8
Hybrid, $\alpha = 0.2$	58.5	12.5

Table 2: Breakdown into KL and reconstruction terms for char-level tweet generation. p refers to input dropout rate.

4.3 Generating Tweets

In this section we present qualitative results on the task of generating tweets.

Data. We use 1M tweets[2] to train our model and test it on a held out dataset of 10k samples. We minimally preprocess tweets by only replacing user ids and urls with "@userid" and "url".

Setup. We use 5 convolutional layers with the ReLU non-linearity, kernel size 3 and stride 2 in the encoder. The number of feature maps is [128, 256, 512, 512, 512] for each layer respectively. The decoder is configured equivalently but with the amount of feature maps decreasing in each consecutive layer. The top layer is an LSTM with 1000 units. We have not observed significant overfitting. The baseline LSTM VAE model contained two distinct LSTMs both with 1000 cells. The models have comparable number of parameters: 10.5M for the LSTM VAE model and 10.8M for our hybrid model.

Results. Both VAE models are trained on the character-level generation. The breakdown of total cost into KL and reconstruction terms is given in Table 2. Note that while the total cost values are comparable, our model puts more information into the latent vector, further supporting our observations from Section 4.1. This is reflected in the random samples from both models, presented in Table 1. We perform greedy decoding during generation so any variation in samples is only due to the latent vector. LSTM VAE produces very limited range of tweets and tends to repeat "@userid" sequence, while our model produces much more diverse samples.

5 Conclusions

We have introduced a novel generative model of natural texts based on the VAE framework. Its core components are a convolutional encoder and a deconvolutional decoder combined with a recurrent layer. We have shown that the feed-forward part of our model architecture makes it easier to train a VAE and avoid the problem of KL-term collapsing to zero, where the decoder falls back to a standard language model thus inhibiting the sampling ability of VAE. Additionally, we propose an efficient way to encourage the model to rely on the latent vector by introducing an additional cost term in the training objective. We observe that it works well on long sequences which is hard to achieve with purely RNN-based VAEs using the previously proposed tricks such as KL-term annealing and input dropout. Finally, we have extensively evaluated the trade-off between the KL-term and the reconstruction loss. In particular, we investigated the effect of the receptive field size on the ability of the model to respect the latent vector which is crucial for being able to generate realistic and diverse samples. In future work we plan to apply our VAE model to semi-supervised NLP tasks and experiment with conditioning generation on text attributes such as sentiment and writing style.

Acknowledgments

We thank Enrique Alfonseca, Katja Filippova, Sylvain Gelly and Jason Lee for their useful feedback while preparing this paper. This project has received funding from the European Union's Framework Programme for Research and Innovation HORIZON 2020 (2014-2020) under the Marie Skodowska-Curie Agreement No. 641805. Stanislau Semeniuta thanks the support from Pattern Recognition Company GmbH. We thank the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

[2] a random sample collected using the Twitter API

References

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Philip Bachman. 2016. An architecture for deep, hierarchical generative models. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *NIPS*, pages 4826–4834.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Olivier Bousquet, Sylvain Gelly, Ilya Tolstikhin, Carl-Johann Simon-Gabriel, and Bernhard Schoelkopf. 2017. From optimal transport to generative modeling: the vegan cookbook. *CoRR*, abs/1705.07642.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *CONLL*, pages 10–21.

Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Variational lossy autoencoder. *CoRR*, abs/1611.02731.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *CoRR*, abs/1612.08083.

Marco Fraccaro, Søren Kaae Sø nderby, Ulrich Paquet, and Ole Winther. 2016. Sequential neural models with stochastic layers. In *NIPS*, pages 2199–2207.

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vázquez, and Aaron C. Courville. 2016. Pixelvae: A latent variable model for natural images. *CoRR*, abs/1611.05013.

David Ha, Andrew M. Dai, and Quoc V. Le. 2016. Hypernetworks. *CoRR*, abs/1609.09106.

Geoffrey E. Hinton and Drew van Camp. 1993. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, COLT 1993, Santa Cruz, CA, USA, July 26-28, 1993.*, pages 5–13.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, pages 1735–1780.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Controllable text generation. *CoRR*, abs/1703.00955.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *CoRR*, abs/1610.10099.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Diederik P. Kingma, Tim Salimans, and Max Welling. 2016. Improving variational inference with inverse autoregressive flow. *CoRR*, abs/1606.04934.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

Hugo Larochelle and Iain Murray. 2011. The neural autoregressive distribution estimator. In *AISTATS*, pages 29–37.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, and Ole Winther. 2015. Autoencoding beyond pixels using a learned similarity metric. *CoRR*, abs/1512.09300.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Lars M. Mescheder, Sebastian Nowozin, and Andreas Geiger. 2017. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. *CoRR*, abs/1701.04722.

Yishu Miao, Lei Yu, and Phil Blunsom. 2015. Neural variational inference for text processing. *CoRR*, abs/1511.06038.

Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. 2015. Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366.

Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. *CoRR*, abs/1601.06759.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*, pages 1310–1318.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434.

Tapani Raiko, Mathias Berglund, Guillaume Alain, and Laurent Dinh. 2014. Techniques for learning binary stochastic feedforward neural networks. *CoRR*, abs/1406.2989.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1530–1538.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, pages 1278–1286.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2016. A hierarchical latent variable encoder-decoder model for generating dialogues. *CoRR*, abs/1605.06069.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. 2016. Ladder variational autoencoders. *CoRR*, abs/1602.02282.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2014. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. 2015. Attribute2image: Conditional image generation from visual attributes. *CoRR*, abs/1512.00570.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. *CoRR*, abs/1702.08139.

Serena Yeung, Anitha Kannan, Yann Dauphin, and Li Fei-Fei. 2016. Epitomic variational autoencoder. *In submission to ICLR 2017*.

Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Robert Fergus. 2010. Deconvolutional networks. In *CVPR*, pages 2528–2535.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2016. Variational neural machine translation. *CoRR*, abs/1605.07869.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *CoRR*, abs/1703.10960.

Barret Zoph and Quoc V. Le. 2016. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578.

Filling the <u>`Blanks`</u> (*hint*: `plural noun`) for Mad Libs® Humor

Nabil Hossain*, **John Krumm**†, **Lucy Vanderwende**†, **Eric Horvitz**† and **Henry Kautz***

*Department of Computer Science
University of Rochester
{nhossain,kautz}@cs.rochester.edu

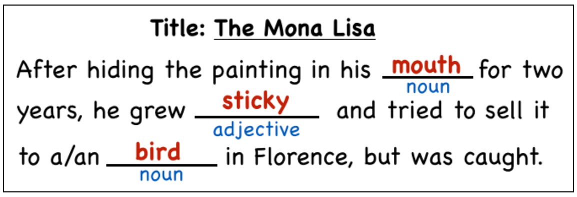
†Microsoft Research AI
Redmond, Washington
{jckrumm,lucyv,horvitz}@microsoft.com

Abstract

Computerized generation of humor is a notoriously difficult AI problem. We develop an algorithm called *Libitum* that helps humans generate humor in a Mad Lib®, which is a popular fill-in-the-blank game. The algorithm is based on a machine learned classifier that determines whether a potential fill-in word is funny in the context of the Mad Lib story. We use Amazon Mechanical Turk to create ground truth data and to judge humor for our classifier to mimic, and we make this data freely available. Our testing shows that Libitum successfully aids humans in filling in Mad Libs that are usually judged funnier than those filled in by humans with no computerized help. We go on to analyze why some words are better than others at making a Mad Lib funny.

1 Introduction

As technologists attempt to build more natural human-computer interfaces, the inclusion of computer-generated humor becomes more important for creating personable interactions. However, computational humor remains a long-standing challenge in AI. Despite decades devoted to theories and algorithms for humor, the best computerized humor is still mediocre compared to humans. Humor requires creativity, sophistication of language, world knowledge, empathy and cognitive mechanisms, which are extremely difficult to model theoretically. A more modest goal for computational humor is to build machines that help humans create humor rather than replace them.



Title: <u>The Mona Lisa</u>

After hiding the painting in his <u>mouth</u> for two
noun
years, he grew <u>sticky</u> and tried to sell it
adjective
to a/an <u>bird</u> in Florence, but was caught.
noun

Figure 1: Example of a Mad Lib sentence. The original words for the blanks, in order, were "apartment", "impatient" and "gallery".

We develop and test an algorithm, called Libitum[1], for a computer-aided approach to humor generation. Its aim is to help a human player fill in the blanks of a Mad Lib® story to make it funny. The algorithm generates candidate words, and its core component is a machine-trained classifier that can assess whether a potential fill-in-the-blank word is funny, based on several features, including the blank's surrounding context. We trained the classifier on Mad Lib stories that were filled in and judged by humans. We note that in our work, we give players, both human and computer, access to the full Mad Lib story, including the sentences surrounding the blanks. In regular Mad Libs, players do not see the surrounding sentences. Figure 1 shows a sentence from a typical Mad Lib, completed by a human player.

The work presented here makes three contributions. The first is the creation of a challenging benchmark for humor generation, a vital aspect of human communication which has received relatively little attention in the NLP community. This benchmark, based on Mad Libs, (i) is challenging, but doable by both humans and machines, (ii) provides quantitative results so that progress can be measured, and (iii) cannot be gamed by trivial strategies, such as filling in random words. The benchmark dataset is annotated and judged us-

[1] Libitum is Latin for "pleased". Mad Lib is a humorous variation of the Latin ad lib, where lib is short for Libitum.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 638–647
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

ing Amazon Mechanical Turk, with several steps taken to reduce effects of human variation in humor taste. Our second contribution is that we create and demonstrate a computer-aided humor algorithm that, in most cases, allows humans to generate funnier Mad Lib stories than they can on their own without computer assistance. The third contribution is an analysis of our test data and algorithm that helps to quantitatively explain what makes our results humorous.

Our work goes beyond the modeling and generation of language to simply convey information. Instead, we are trying to create a pleasurable feeling using humor. This is analogous to the Story Cloze Test (Mostafazadeh et al., 2017), where the task is to choose a *satisfying* ending to a story.

We note the previous work called "Visual Madlibs" (Yu et al., 2015). Although its title implies similarity to our work, it is about an image data set augmented with fill-in-the-blank questions, such as "This place is a park."

2 Related Work

Developing a general humor recognizer or generator is hard, and some researchers consider it an AI-complete problem (Stock and Strapparava, 2003), *i.e.*, "solving" computational humor is as difficult as making computers as intelligent as people.

While our goal is generating humor, there is a growing literature of projects for recognizing humor, mostly aimed at specific types of jokes.

Taylor and Mazlack's (2004) work on knock-knock jokes is based on Raskin's (2012) "semantic theory of verbal humor", which says that humor comes from script opposition. Here, the joke posits two different interpretations that oppose each other, resulting in humor. They analyze the two key parts of knock-knock jokes to discover the wordplay that results in the humorous opposition.

Mihalcea and Strapparava (2006) developed a classifier to recognize one-liner jokes by looking for alliteration, antonymy, and adult slang. Compared to different classes of non-jokes, they achieved an accuracy of between 79% and 97%.

Davidov *et al.* (2010) decompose potential sarcastic sentences into specialized parts of speech and match them against sentence patterns that they discovered in their corpus of sarcasm. Like us, they used Mechanical Turk to find ground truth.

Kiddon and Brun (2011) present a classifier that recognizes double entendres that become funny after adding "That's what she said" at the end.

Other humor recognition efforts do not start with constraints on the type of humor they recognize. For instance, Zhang and Liu's (2014) system recognized humorous tweets based on 50 linguistic features from humor theory, achieving a classification accuracy of 82%. Raz (2012) shows how to classify humorous tweets into one of 12 joke categories. Betero and Fung (2016) take on the challenging task of recognizing humor from TV sitcoms, using a neural network. They use the sitcom's laugh track to identify ground truth.

Our work is aimed at generating humor rather than recognizing it. As with previous work on humor recognition, humor generation has been largely limited to specific types of jokes. The three examples below, along with ours, show that automatically generating humor still relies on specializing around a subgenre of jokes with customized approaches that have yet to yield a general method for generating humor.

Binsted *et al.*'s Joke Analysis and Production Engine (JAPE) produced punning riddles, such as:

Question: What do you call a weird market?

Answer: A bizarre bazaar. (Binsted et al., 1997)

JAPE worked by discovering candidate ambiguities in either spelling (*e.g.*, cereal *vs.* serial) or word sense (e.g. river bank vs. savings bank). Evaluated by children, JAPEs jokes were funnier than non-jokes, but not as funny as human-generated jokes.

Petrovic and Matthews (2013) created an algorithm to generate jokes such as "I like my coffee like I like my war, cold," filling in the three blanks. They encoded four assumptions about what makes a joke funny, using discrete probability tables learned from a large corpus of regular text data along with part-of-speech tagging and an estimate of different possible senses of the words. 16% of their automatically generated jokes were considered funny, compared to 33% when the same type of jokes were generated by people.

HAHAcronyn, from Stock and Strapparava (2003) attempted to take existing acronyms (e.g. DMSO for Defense Modeling and Simulation Office) and make an alternate, funny version (e.g. Defense Meat-eating and Salivation Office). Their algorithm keeps part of the acronym and then looks for what to change in the remainder, with goals of different semantics, rhymes, antonyms, and extreme-sounding adverbs.

Besides the novelty of looking at Mad Libs, our work is different in that it seeks to generate humor for *longer* passages than the acronyms, one-liners, and short riddles of previous work.

In addition to algorithmic work, there is a long history of research into general theories of humor (O'Shannon, 2012; Weems, 2014; Wilkins and Eisenbraun, 2009). One of the main thrusts is incongruity theory, which says that a joke is funny when it has a surprise that violates the conventional expectation. According to the Benign Violation Theory (Raskin, 2008), the unexpected must logically follow from the set up and must not be offensive to the reader, otherwise the reader is left confused and the joke is not funny. Similarly, the Sematic Script Theory of Humor (SSTH) says that a joke emerges when it can be interpreted according to two different, generic scripts, one of which is less obvious (Attardo and Raskin, 1991). Labutov and Lipson make a first step at exploiting the SSTH theory to automatically generate two-line jokes (Labutov and Lipson, 2012).

In general, it is difficult to apply these theories directly to humor recognition and generation, however, because they require a high degree of common sense understanding of the world. Because of this, most successful algorithmic work on humor is limited to using relatively shallow linguistic rules on specific types of jokes. This is also true of our work, which concentrates on filling the blanks in Mad Libs, described next.

3 Mad Libs®

Invented in 1953 by Roger Price and Leonard Stern (2008a), Mad Libs is a fill-in-the blank game intended to be humorous. A Mad Lib consists of a story of several sentences and a title. Some of the words are replaced with blanks, each of which has a **hint** belonging to a certain **hint type**, such as a part of speech. Players fill in each blank with a word that agrees with the hint. A player can see only the story's title and the list of blanks with hints. The resulting filled-in Mad Lib is usually funny, because players fill in the blanks with no knowledge of the story (except for its title). The humor comes from the nonsensical filled-in words in the context of a sensible, coherent story. Figure 1 shows part of a filled-in Mad Lib created from a story describing the theft of the Mona Lisa.

Filling in Mad Libs is a novel challenge for automatic humor generation. The title and words surrounding the blanks in a Mad Lib provide a contextual scaffolding that an algorithm can exploit to choose appropriate words for the blanks that make the resulting story humorous.

In order to incorporate such context, our rules for playing Mad Libs differ from the original ones: both our algorithmic and human players are allowed to look at the story as a guide to filling in the blanks. This makes the problem much richer, because players can take advantage of the story's text in choosing which words to fill in. Without looking at the story, our algorithm would be reduced to one that chooses only *a priori* funny words.

3.1 Fun Libs Generation

Mad Libs are copyrighted, and therefore it is difficult to release a data set by using stories from Mad Libs books. Instead we studied original Mad Libs to develop our own dataset, which we call **Fun Libs**. This data set, including filled-in words and funniness assessments from Mechanical Turkers, is available online.[2]

Designing Mad Lib-like stories requires skill, because the Mad Lib context is usually designed in a way to help generate humor. To create our own stories, we first examined 50 Mad Libs from one of the many Mad Libs books (Price and Stern, 2008b). We found that the mean number of blanks, observed words and sentences per Mad Lib were, respectively, 16.0 ($\sigma = 2.25$), 114.84 ($\sigma = 20.58$) and 9.04 ($\sigma = 2.38$). There were 14 unique hint types.

One of our main goals is to build a system that can create meaningful, diverse, and funny stories which apply to a broad audience. However, in pilot tests with human players, we found that six of the original hint types restricted the variety of humor that can be generated by filling in their blanks, by: (i) not affording a variety of possibilities to fill in (hint types: color, silly word, exclamation) and subtlety in humor generation (number), and (ii) generally requiring the audience to have knowledge of cultural references and specifics (person name, place). Hence, we discarded them, leaving eight hint types in our Fun Libs dataset, as shown in Table 1. Some of them have variants such as plurality for nouns and tenses for verbs.

Next, we created our dataset of 50 Fun Libs using simple Wikipedia articles because, similar to

[2] Fun Libs dataset: https://www.microsoft.com/en-us/download/details.aspx?id=55593

Hint Type	**Mad Libs**	**Fun Libs**
Noun	7.06	7.00
Adjective	4.06	3.12
Verb	1.22	3.10
Part of the Body	0.98	0.46
Adverb	0.44	0.38
Type of Food	0.22	0.20
Animal	0.20	0.36
Type of Liquid	0.18	0.16
All Blanks	**16.0***	**14.78**

Table 1: Mean hint types per story in the two datasets. *The mean number of blanks in the Mad Libs dataset was computed based on 14 hint types.

Mad Libs, these articles have a title and text. Creating the stories involved finding a Wikipedia article and picking sentences which have potential to generate humor (with very minimal edits such as reducing verbosity), and then replacing some words with blanks in a way such that the overall blank, sentence, word and hint type distributions are similar to their respective distributions in the 50 original Mad Libs. Table 1 shows the means of the hint type distributions for the two datasets. We randomly sampled 40 Fun Libs for training and kept the remaining 10 for evaluation.

4 Data Annotation

With a set of Mad Lib-like stories in place, our next task was to objectively create filled-in stories to use as the basis for the remainder of our analysis. We used Amazon Mechanical Turk workers, bootstrapping from an initial set of filled-in stories, to then finding qualified turker judges, and then finding qualified turker players to fill in the blanks. Our goal was to create a labeled dataset with filled-in blanks and a **funniness grade** for each filled-in word. We selected turkers who are native English speakers (from USA, Canada, Great Britain and Australia), have a HIT approval rate above 97%, and have completed at least 10,000 HITs. We now describe how we further selected qualified turkers and the methods we applied to obtain better quality for the labeled data.

4.1 Judge Selection

To grade how funny a filled-in story is, we needed turker judges who were unbiased referees of general humor. We selected judges before players, because we used the judges to find qualified players. Finding good judges is challenging, because humor is subjective. We launched a qualification task to select qualified judges, with clear instructions on what makes a desirable judge. This task involved giving turkers a set of seven filled-in stories to be graded for funniness. Out of these seven, three were direct excerpts from Wikipedia articles with some words marked as filled-in blanks. Except for designating some words as filled-in, these stories were unaltered from Wikipedia, and therefore were not funny. The remaining four stories were filled in by humans to make them funny. Turkers were allowed to select, for each story, a grade from $\{0,1,2,3\}$ (a scale which we used throughout our work) described as follows:

0 - Not funny
1 - Somewhat funny
2 - Moderately funny
3 - Funny

We marked the ground-truth grade of the Wikipedia excerpts as 0, and we used volunteers from our research group to decide the ground-truth grade of the other four stories between one and three. We used 60 candidate turkers to do this qualification task, out of which 27 successfully assigned 0 to all the Wikipedia excerpts and 1-3 to the other four stories. We selected all of them as qualified judges. 16 others graded all the Wikipedia excerpts as 0 but considered 1 of the other four stories to be "Not funny". We sorted these turkers using the total Euclidean distance of their grades from the ground truth, and chose the top 13 among them to get 40 judges in total. To make sure that turkers read each story, we also asked a question that could be answered correctly only by understanding the story's context. During our initial tests, we removed two judges who were repeatedly failing to answer these questions correctly and took very little time to grade. Therefore, our final judge pool included 38 judges. This selection process was designed to find judges that were careful, consistent, and representative in objectively judging the funniness of a filled-in story.

4.2 Player Selection

Players are the people who fill in the blanks in a story. Our goal was to find turker players who were good at creating funny stories by filling in blanks. There was no overlap between the qualified judges and the qualified players. To select qualified players, we created two extra fill-in-the-blank stories to be used for player selection alone. We gave both stories to 50 candidate turkers to fill in. We instructed candidate turkers to avoid using slang words, sexual references or bathroom humor as these are crude, easy techniques for generating humor, and do not require creativity. Further, we required turkers to fill in each blank with exactly one word (using alphabetic characters only) that can be found in a US English dictionary, that is grammatically correct, and that is not colloquial.

To lessen the impact of story contexts and to have a variety in the humor generated, another task was launched with two new stories and 30 new candidate turkers. For each filled-in story, we graded the funniness of each story on our 0-3 scale using 10 qualified judges (described above) to mitigate the effects of variations in humor taste. We ranked the potential players and selected the highest ranked as qualified players. A total of 25 qualified players were obtained from the two batches.

For both the judge and player selection phase, we launched several small pilot tests and used turkers' feedback to design a better data labeling phase, which we cover next.

4.3 Labeling Fun Libs

Our goal in labeling filled-in stories was to assess the funniness of the overall filled-in story, the funniness contribution of each filled-in word, and other aspects of the humor. For each of the 40 stories in the training set, we used 5 players to fill the blanks, giving us a total of 200 filled-in stories. The players also self-graded the humor in their completed stories.

Each of these stories was graded by 9 judges to represent an audience rather than an individual and to reduce effects of different humor tastes. Judges answered the following:

- Which of the filled-in words caused humor.
- Funniness of the story (integer scale of 0-3).
- How coherent the story is, with the filled-in words (integer scale of 0-3).
- To what extent the filled-in words caused the story to deviate from its original topic as suggested by the title (integer scale of 0-3).
- Whether the humor generation technique of **incongruity**[3] was applied by the player.
- A verification quiz, which can be answered using the context of the story, to help ensure that judges read the filled-in story carefully.

We asked judges about coherence because we expected incongruity would play a significant role in the humor of Mad Libs due to their nature.

Judges and players each received 60 U.S. cents per HIT. We also announced bonuses for the top 10 judges selected based on other judges' agreements with them and the top 10 players based on how funny their filled-in stories were, as graded by the judges. The total Mechanical Turk cost, including the bonuses, was approximately US$2000.

5 Computer-aided Humor

In this section, we will discuss our machine learning approach to generating humorous Fun Libs. First we trained a random forest classifier (Breiman, 2001) to predict whether a filled-in word is funny using features from the story and the title. Then our technique for generating complete, funny Fun Libs involves, for each blank:

1. Use a language model to generate words
2. Keep the top 20 funny candidate words as ranked by our classifier
3. Use humans to decide which candidate to fill in the blank with

5.1 Language Model

To supply reasonable words for each blank, we used the Microsoft Web Language Models API (Wang et al., 2010), trained using Web Page titles and available as a Web service. To generate candidate words for a blank, we use the API's word completion tool to get a list of all possible candidate words using context windows up to four previous words. Next, for each word we computed the **joint probability score**, using the API's joint probability tool to get a probability estimate of how well each candidate fits into the containing sentence. Then we ranked candidates using this score. We expect the words with high scores to be the more fitting (less humorous) words for the context, while those with low scores are more likely to be the incongruous words that may generate surprise or humor when used in the sentence.

5.1.1 Candidate Refinement

Since Web page titles for our language model are not forced to be grammatically correct, the generated words are not all appropriate. Thus we applied the following constraints to get a cleaner candidate list:

[3] Incongruity theory of humor says that a joke is funny when it has a surprise, often at the end, that violates the conventional expectation, often set up at the start (Weems, 2014).

- The candidate must be in WordNet (Fellbaum, 1998) or a US English dictionary.
- For a blank that is a sub-category of nouns (*e.g.*, “animal”), the candidate must have the same sub-category as its WordNet hypernym.
- The candidate must have a part of speech tag that agrees with the hint.
- The candidate must not be a word with non-alphabetic characters or a stop word.
- The candidates for nouns and sub-categories have to fit the context in terms of plurality .
- Candidate must not be a slang, adult or bathroom-related word (filtered using a list of 4,012 words), because such words would be too crude, producing shallow, easy humor.

5.2 Features

To predict whether a filled-in word is funny or not, our classifier uses the following ten features extracted from the **(word, story)** pair:

1. Hint type.
2. Length of the word.
3. Language model’s joint probability for the containing sentence with the word filled-in.
4. The word’s relative position, in terms of joint probability, in the ranked candidate list generated by the language model: a value in the interval [0,1], with 0 implying the candidate has the highest joint probability in the list.
5. Minimum, maximum and average letter trigram probabilities using letter bigram and letter trigram counts from the Google Web Trillion Word Corpus (Brants and Franz, 2006). The purpose of these features is to capture the *phonetic funniness* of words (*e.g.*, “whacking” instead of “fighting”).
6. The candidate’s similarity to the three contexts — title, overall story, and the containing sentence. This is the cosine similarity of the candidate’s word embedding vector with the average word embedding vector for each context. The vectors were computed using GloVe vectors (Pennington et al., 2014) trained with 840 billion tokens from Web data.

5.3 Classification

The full training dataset includes 40 labeled Fun Libs, each filled in by 5 different players, each of which was graded by 9 different judges. We split the data randomly by story titles, keeping 30 for training and 10 in a validation set. This ensures that the classifier does not see the validation stories’ contexts during training. Thus, our training set consisted of features and funniness labels for filled-in words from 150 stories. Further, we assigned labels using a **vast majority** vote, *i.e.*, a filled-in word having, out of nine judges’ votes:

- six or more “funny votes” is **funny**
- three or fewer funny votes is **not funny**

Otherwise the word was discarded. As a result, the final dataset includes 1939 instances of filled-in words: 1449 for training and 490 for validation.

	Train	**Validation**
Precision (Funny)	0.712	0.715
Recall (Funny)	0.856	0.801
F1 score (Funny)	0.778	0.756
Accuracy	**0.727**	**0.695**

Table 2: Filled-in word classification results.

We experimented with several machine learning models, including linear regression, to predict the number of positive votes for each word. Among these, the random forest classifier worked best. We performed 10-fold cross validation to train this classifier, optimizing based on the F1 score. The results are shown in Table 2. On the validation data, the classifier has an F1 score of 0.756 and an overall classification accuracy of 69.5%. While these quality measures leave room for improvement, they show that the classifier is clearly biased toward choosing funnier words.

We show results from three baseline classifiers in Table 3. Among these, the “Chance” classifiers always predict the most frequent class found in the training set, and “Chance Hint” predicts the most frequent class for each hint type. The other baseline is a Linear SVM classifier trained using the three most important features of the trained random forest as shown in Figure 2. The SVM’s learned weights for the similarity features between the word and the containing sentence, the entire story, and the title, respectively, were -0.449,

Baseline	**Train**	**Validation**
Chance	0.571	0.543
Chance Hint	0.595	0.591
Linear SVM (3 feat.)	0.606	0.600

Table 3: Baseline classification accuracies.

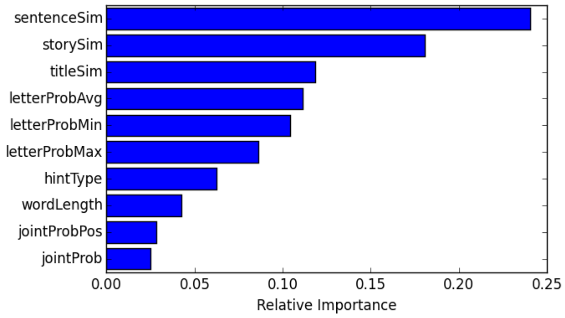


Figure 2: Feature importance for our classifier.

-0.883 and 1.344, showing that funnier filled-in words are similar to the title but not to the body of the story. This suggests that incongruity is playing a significant role in generating humor.

6 Evaluation and Discussion

In this section, we define three approaches for generating humor in Mad Libs, and then we compare and analyze the results we obtain from them. These approaches are:

1. **FreeText**: players fill in the blanks without any restrictions.
2. **LMC** (Language Model + Multiple Choice): for each blank, players choose a word from up to 20 candidate words generated by the language model and sorted by their joint probability score[4].
3. **Libitum**: Similar to the LMC method, except here we rank all the words up to the top 500 words generated by the language model using our classifier, keeping up to the top 20 "humorous" candidates[4].

These methods are designed to study the outcome of humor generated by humans only vs. humans with machine help. The purpose of the LMC model is to study whether the language model alone is a good aid to humans when creating humor, and the benefit of adding machine learning.

For evaluation, we used our ten test Fun Libs, and for each of our three approaches, each of the ten stories was completed by three players. Figure 3 shows the mean grade for these 30 stories. In the figure, the titles are sorted left to right based on the maximum mean story grade among the titles in the Libitum approach. The stories per title are also sorted left to right in descending mean grade. In only one story (ID = 21), the Libitum model receives a lower mean grade than the LMC model, suggesting that adding the machine learning to the language model helps generate significantly more humor than the language model alone. The FreeText model is fairly consistent in generating more humor than the LMC model, which beats the FreeText model in only 7 out of 30 instances. However, the Libitum approach frequently outperforms the FreeText (human only) approach, and it achieves significant gain in generating humor in the stories with the titles "Valentine's Day" and "Cats".

Interestingly, the two stories that received the highest mean grade ("Batman" and "Ducks") are from the FreeText format. This suggests that given more freedom, humans are capable of generating a stronger degree of humor than when they are restricted to a limited number of choices. Excerpts from the best "Batman" story are shown in Figure 4. Here, the strategy employed by the FreeText player is to consistently portray Batman as an obese person obsessed with eating, exploiting the superiority theory of humor[5] (Mulder and Nijholt, 2002). This is remarkable, because it shows how skilled humans are at finding and relating multiple coherent concepts, achieving meaningful and steady humor via continuity, something which is very difficult for machines to do. Much of the humor generated by the Libitum approach here is via incongruity — the filled-in words are quite humorous mainly because they fit their contexts but do not match the expectation of the reader (*e.g.*, Batman is an inefficient superhero). At times, some of the filled-in words in the Libitum approach coherently generate humor, for instance, in the last sentence when Batman is described as wearing a veil to fight acne.

Since each human has a bias towards his/her own understanding of humor, we also studied how the stories appealed to the judges individually by counting the total number of judges for each grade in the three approaches. Table 4 shows the results, where the difference between the mean fun-

[4] In rare cases, for a blank, the language model was only able to generate less than 20 candidate words that pass the candidate refinement step. For such cases, the LMC candidate list was expanded to at least 10 words by adding words randomly from all possible words in WordNet that fit the blank's hint type, if necessary. For Libitum, WordNet was used to randomly add words fitting the blank's hint type to make a list of 50 words that pass the candidate refinement phase. These words were sorted using Libitum and used to ensure the final candidate list had at least 10 words.

[5] Superiority humor comes from the misfortunes or shortcomings of others

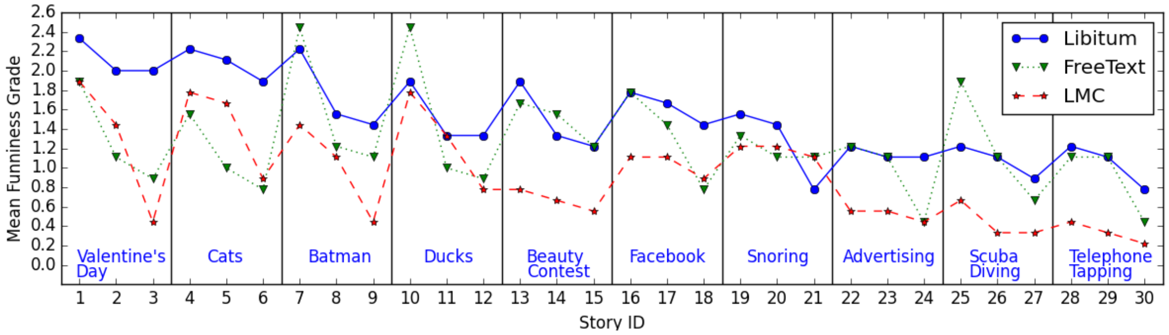


Figure 3: Mean funniness grade for the 30 test stories and their titles. The maximum mean grade for a story in the Libitum format was used to order the titles. For each title, the three stories are sorted in descending mean grade.

Grade	FreeText	LMC	Libitum
0	65	102	37
1	97	100	103
2	84	56	86
3	24	12	44
mean	**1.25**	**0.92**	**1.51**

Table 4: Funniness grades for the 30 stories for the three humor generation formats.

niness grades for each pair of approaches is statistically significant with $p < 0.005$ when a 2-sample t-test was performed. Using Krippendorff's Alpha (Krippendorff, 1970), we also found positive agreements between judges for these and training set ratings.

As expected, the LMC model is the poorest in terms of generating humor. Further, for each non-zero grade, the Libitum model received more votes than the FreeText model. A possible reason is that the judges and players have different perceptions of humor. In the FreeText approach, the common technique employed by players was to use words that are coherent, belonging to a specific topic or domain, and to guide the story towards one conclusion (*e.g.*, the Batman story in Table 4). When this technique worked well, the humor generated was very strong. However, the players have their own biases towards what is humorous, and having more freedom in the FreeText format allowed them to explore their own concept of humor, which could be too narrow to appeal to a broader audience. The Libitum approach, by restricting the players, prevented them from inserting words that they themselves thought were funny, but were not actually funny to people in general.

Table 5 shows passages from stories containing filled-in words that received 9 funniness votes (the maximum possible) from the judges. The story ID and the algorithm used are also provided. Here, in the "Ducks" story, the FreeText player chose to generate humor by developing a steady mockery by satirizing ducks as politicians, whereas the LMC player chose the incongruity approach. The "Beauty Contest" story shows the outstanding skills of humans in generating humor when two blanks are directly connected to each other (*i.e.*, "brawler" and "deadly"). For the same segment, Libitum was also able to aid the players in generating a very funny word, however, the coherence between the blanks does not appear strong. With the computer aided approaches, it is quite difficult to suggest candidates for pairs of (or more) blanks such that the choices are coherent.

6.1 Correlations

Table 6 shows correlations between different ratings by judges (coherence, topic change and incongruity) and the stories' funniness grades. Incongruity had the strongest positive, and statistically significant (with $p < 0.001$), correlation with the graded humor of a story. Coherence also appears very important for generating humor in all the datasets except LMC, where it is difficult to generate coherent words using a language model only. Libitum is likely aiding players in achieving funniness by providing coherent words. In LMC, most of the words generating humor are probably random, incongruous words since the change of topic strongly positively correlates with increasing the humor but coherence does not. Lastly, the player's self grade of humor has no significant relationship with the judges' grade of humor, suggesting that each person has his/her own biases about what is humorous.

Alg.	ID	Text
FText	10	Most ducks are `republican` (adjective) birds, they can be found in both saltwater and fresh **vodka** (liquid). Ducks are omnivorous, eating **talkative** (adjective) plants . . . People commonly feed ducks in ponds stale **jokes** (noun), thinking that the ducks will like to have something to **ridicule** (verb)
LMC	10	Most ducks are `evil` (adjective) birds, they can be found in both saltwater and fresh **coffee** (liquid) . . . Some ducks are not `evil` (adjective), and they are bred and kept by **jesus** (noun).
FText Libtm	14 15	A beauty contest is a/an `elaborate` (adj) contest to decide which **brawler** (n) is the most **deadly** (adj) `observational` `organism` **flammable**
Libtm	5	Cats are the most **barbaric** (adjective) pets in the world. They were probably first kept because they ate **humans** (animal). Later cats were `bullied` (verb [past]) because they are **corrupt** (adjective) . . . Cats are active carnivores, meaning they hunt **online** (adjective) prey.

Table 5: Excerpts from test set stories, showing filled-in words receiving 9 (out of 9) funniness votes (boldfaced) from our judges. The story ID and the approach provided can be used to trace the mean funniness grade of the story in Figure 3.

Batman is a fictional character and one of the most famous / **obese** / sexiest / **inefficient** (adjective) superheroes . . . Batman began in comic books and he was later used / liposuctioned / arrested / imprisoned (verb [past]) in several movies, TV programs, and books. Batman lives in the fictional / edible / holy / abyssal (adjective) city of Gotham. . . . Batman's origin story is that as a/an young / obnoxious / adult / algebraic (adjective) child, Bruce Wayne saw a robber murder / eat / play / prank (verb) his parents after the family left a/an theater / sauna / trail / bag (noun). Bruce decided that he did not want that kind of violence / meal / luck / poem (noun) to happen to anyone else. He dedicated his life to protect / devouring / pronounce / demolish (verb) Gotham City. Wayne learned many different ways to fight / nibble / spell / crochet (verb) as he grew up. As an adult, he wore a/an costume / prosthesis / wig / **veil** (noun) to protect his identity / belly / head / jaw (noun) while fighting crime / gelatin / poverty / **acne** (noun) in Gotham.

Figure 4: Portions of the best "Batman" story, with the filled-in words ordered based on their sources as follows: Original / FreeText / LMC / Libitum. Each boldfaced word was rated "funny" by all 9 judges.

Assessment	Train	FText	LMC	Libtm
Coherence	**0.390**	**0.345**	0.023	**0.578**
Topic change	**0.266**	**0.281**	**0.467**	0.028
Self-grade	0.001	-0.054	0.024	0.151
Incongruity	**0.638**	**0.620**	**0.650**	**0.515**

Table 6: Correlation of different assessments of stories with their funniness grade. The boldfaced and underlined correlations are statistically significant, respectively, with $p < 0.001$ and $p < 0.05$.

6.2 Fully Automated System

We also tested an automated Mad Lib humor generation system, where we filled-in each test blank with the most funny candidate word from Libitum. The results were poor, with the stories having a mean funny grade of 0.80. This is expected, because without human support, Libitum cannot analyze complex word associations in order to come up with words that preserve meaning and coherence. This is evidenced by the statistically significant correlation ($p < 0.001$) of 0.456 between coherence and mean funniness grade scores from judges, suggesting that the judges mostly found those stories funny which were coherent.

7 Conclusion

Our goal was to create and explore a new benchmark for computational humor research. We created a copyright-free dataset that approximately matches the statistical characteristics of Mad Libs. Then we vetted a pool of Mechanical Turk judges and players to create ground truth data. We developed an algorithm called Libitum that generates and classifies potential blank-filling words as either funny or not in the context of a Mad Lib. For each blank, Libitum supplied a list of potentially funny words from which a human could choose. As judged by humans, the Libitum-aided words easily worked better than words from a simple language model and were usually better than even words generated by human players who could fill in the blanks in whichever way they liked.

Our three contributions, the benchmark, Libitum, and the analysis of what makes it funny, advance the state of the art in computer humor by demonstrating a successful computer aided humor technique and quantitatively analyzing what makes for funny fill-in-the-blank words for Mad Libs. These analyses show that coherent stories have tremendous potential in making a Mad Lib humorous, a promising direction for future work.

References

Salvatore Attardo and Victor Raskin. 1991. Script theory revis (it) ed: Joke similarity and joke representation model. *Humor-International Journal of Humor Research*, 4(3-4):293–348.

Dario Bertero and Pascale Fung. 2016. A long short-term memory framework for predicting humor in dialogues. In *Proceedings of NAACL-HLT*, pages 130–135.

Kim Binsted, Helen Pain, and Graeme D Ritchie. 1997. Children's evaluation of computer-generated punning riddles. *Pragmatics & Cognition*, 5(2):305–354.

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram version 1. Linguistic Data Consortium, Philadelphia.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 107–116. Association for Computational Linguistics.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Chloe Kiddon and Yuriy Brun. 2011. That's what she said: double entendre identification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 89–94. Association for Computational Linguistics.

Klaus Krippendorff. 1970. Estimating the reliability, systematic error and random error of interval data. *Educational and Psychological Measurement*, 30(1):61–70.

Igor Labutov and Hod Lipson. 2012. Humor as circuits in semantic networks. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 150–155. Association for Computational Linguistics.

Rada Mihalcea and Carlo Strapparava. 2006. Learning to laugh (automatically): Computational models for humor recognition. *Computational Intelligence*, 22(2):126–142.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James F Allen. 2017. Lsdsem 2017 shared task: The story cloze test. *LSDSem 2017*, page 46.

Matthijs P Mulder and Antinus Nijholt. 2002. Humour research: State of art. Technical report, University of Twente, Centre for Telematics and Information Technology.

Dan O'Shannon. 2012. *What are You Laughing At?: A Comprehensive Guide to the Comedic Event*. A&C Black.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Sasa Petrovic and David Matthews. 2013. Unsupervised joke generation from big data. In *ACL (2)*, pages 228–232.

Roger Price and Leonard Stern. 2008a. *50 Years of Mad Libs*. Penguin Group.

Roger Price and Leonard Stern. 2008b. *Best of Mad Libs*. Penguin Group.

Victor Raskin. 2008. *The primer of humor research*, volume 8. Walter de Gruyter.

Victor Raskin. 2012. *Semantic mechanisms of humor*, volume 24. Springer Science & Business Media.

Yishay Raz. 2012. Automatic humor classification on twitter. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop*, pages 66–70. Association for Computational Linguistics.

Oliviero Stock and Carlo Strapparava. 2003. Hahacronym: Humorous agents for humorous acronyms. *Humor*, 16(3):297–314.

Julia M Taylor and Lawrence J Mazlack. 2004. Computationally recognizing wordplay in jokes. In *Proceedings of the Cognitive Science Society*, volume 26.

Kuansan Wang, Christopher Thrasher, Evelyne Viegas, Xiaolong Li, and Bo-june Paul Hsu. 2010. An overview of microsoft web n-gram corpus and applications. In *Proceedings of the NAACL HLT 2010 Demonstration Session*, pages 45–48. Association for Computational Linguistics.

Scott Weems. 2014. *Ha!: The science of when we laugh and why*. Basic Books.

Julia Wilkins and Amy Janel Eisenbraun. 2009. Humor theories and the physiological benefits of laughter. *Holistic nursing practice*, 23(6):349–354.

Licheng Yu, Eunbyung Park, Alexander C Berg, and Tamara L Berg. 2015. Visual madlibs: Fill in the blank description generation and question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2461–2469.

Renxian Zhang and Naishi Liu. 2014. Recognizing humor on twitter. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 889–898. ACM.

Measuring Thematic Fit with Distributional Feature Overlap

Enrico Santus[1], **Emmanuele Chersoni**[2], **Alessandro Lenci**[3] and **Philippe Blache**[2]
enrico_santus@sutd.edu.sg
emmanuelechersoni@gmail.com
alessandro.lenci@unipi.it
philippe.blache@univ-amu.fr
[1] Singapore University of Technology and Design
[2] Aix-Marseille University
[3] University of Pisa

Abstract

In this paper, we introduce a new distributional method for modeling predicate-argument thematic fit judgments. We use a syntax-based DSM to build a prototypical representation of verb-specific roles: for every verb, we extract the most salient *second order contexts* for each of its roles (i.e. the most salient dimensions of *typical role fillers*), and then we compute thematic fit as a *weighted overlap* between the top features of candidate fillers and role prototypes. Our experiments show that our method consistently outperforms a baseline re-implementing a state-of-the-art system, and achieves better or comparable results to those reported in the literature for the other unsupervised systems. Moreover, it provides an explicit representation of the features characterizing verb-specific semantic roles.

1 Introduction

Several psycholinguistic studies in the last two decades have brought extensive evidence that humans activate a rich array of event knowledge during sentence processing: verbs (e.g. *arrest*) activate expectations about their typical arguments (e.g. *cop, thief*) (McRae et al., 1998; Altmann and Kamide, 1999; Ferretti et al., 2001; McRae et al., 2005; Hare et al., 2009; Matsuki et al., 2011), and nouns activate other nouns typically co-occurring in the same events (Kamide et al., 2003; Bicknell et al., 2010). Subjects are able to determine the plausibility of a noun for a given argument role and quickly use this knowledge to anticipate upcoming linguistic input (McRae and Matsuki, 2009). This phenomenon is referred to in the literature as *thematic fit*. Thematic fit estimation has been extensively used in sentence comprehension studies on constraint-based models, mainly as a predictor variable allowing to disambiguate between possible structural analyses.[1] More in general, thematic fit is considered as a key factor in a variety of studies concerned with structural ambiguity (Vandekerckhove et al., 2009).

Starting from the work of Erk et al. (2010), several distributional semantic methods have been proposed to compute the extent to which nouns fulfill the requirements of verb-specific thematic roles, and their performances have been evaluated against human-generated judgments (Baroni and Lenci, 2010; Lenci, 2011; Sayeed and Demberg, 2014; Sayeed et al., 2015, 2016; Greenberg et al., 2015a,b). Most research on thematic fit estimation has focused on *count-based* vector representations (as distinguished from *prediction-based* vectors).[2] Indeed, in their comparison between high-dimensional explicit vectors and low-dimensional neural embeddings, Baroni et al. (2014) found that thematic fit estimation is the only benchmark on which prediction models are lagging behind state-of-the-art performance. This is consistent with Sayeed et al. (2016)'s observation that "thematic fit modeling is particularly sensitive to linguistic detail and interpretability of the vector space".

The present work sets itself among the unsupervised approaches to thematic fit estimation. By relying on explicit and interpretable *count-based* vector representations, we propose a simple, cognitively-inspired, and efficient thematic fit model using information extracted from dependency-parsed corpora. The key features of our proposal are *a*) prototypical representations of verb-specific thematic roles, based on feature weighting and filtering of *second order contexts*

[1] For an overview on constraint-based models, see MacDonald and Seidenberg (2006).

[2] We adopt the terminology from Baroni et al. (2014).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 648–658
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

(i.e. contexts that are salient for many of the typical fillers of a given verb-specific thematic role), and *b*) a similarity measure which computes the *Weighted Overlap* (WO) between prototypes and candidate fillers.[3]

2 Related Work

Erk et al. (2010) were, at the best of our knowledge, the first authors to measure the correlation between human-elicited thematic fit ratings and the scores assigned by a syntax-based Distributional Semantic Model (DSM). More specifically, their gold standard consisted of the human judgments collected by McRae et al. (1998) and Padó (2007). The plausibility of each verb-filler pair was computed as the similarity between new candidate nouns and previously attested exemplars for each specific verb-role pairing (as already proposed in Erk (2007)).

Baroni and Lenci (2010) evaluated their Distributional Memory (henceforth DM)[4] framework on the same datasets, adopting an approach to the task that has become dominant in the literature: for each verb role, they built a prototype vector by averaging the dependency-based vectors of its most typical fillers. The higher the similarity of a noun with a role prototype, the higher its plausibility as a filler for that role. Lenci (2011) has later extended the model to account for the dynamic update of the expectations on an argument, depending on how another role is filled. By using the same DM tensor, this study tested an additive and a multiplicative model (Mitchell and Lapata, 2010) to compose and update the expectations on the patient filler of the subject-verb-object triples of the Bicknell dataset (Bicknell et al., 2010).

The thematic fit models proposed by Sayeed and Demberg (2014) and Sayeed et al. (2015) are similar to Baroni and Lenci's, but their DSMs were built by using the roles assigned by the SENNA semantic role labeler (Collobert et al., 2011) to define the feature space. These authors argued that the prototype-based method with dependencies works well when applied to the agent and to the patient role (which are almost always syntactically realized as subjects and objects), but that it might be problematic to apply it to different roles, such as instruments and locations, as the construction of the prototype would have to rely on prepositional complements as typical fillers, and the meaning of prepositions can be ambiguous. Comparing their results with Baroni and Lenci (2010), the authors showed that their system outperforms the syntax-based model DepDM and almost matches the scores of the best performing TypeDM, which uses hand-crafted rules. Moreover, they were the first to evaluate thematic role plausibility for roles other than agent and patient, as they computed the scores also for the instruments and for the locations of the Ferretti datasets (Ferretti et al., 2001).

Greenberg et al. (2015a,b) further developed the TypeDM and the role-based models, investigating the effects of verb polysemy on human thematic fit judgments and introducing a hierarchical agglomerative clustering algorithm into the prototype creation process. Their goal was to cluster together typical fillers into multiple prototypes, corresponding to different verb senses, and their results showed constant improvements of the performance of the DM-based model.

Finally, Tilk et al. (2016) presented two neural network architectures for generating probability distributions over selectional preferences for each thematic role. Their models took advantage of supervised training on two role-labeled corpora to optimize the distributional representation for thematic fit modeling, and managed to obtain significant improvements over the other systems on almost all the evaluation datasets. They also evaluated their model on the task of composing and updating verb argument expectations, obtaining a performance comparable to Lenci (2011).

3 Methodology

As pointed out by Sayeed et al. (2016), most works on unsupervised thematic fit estimation vary in the method adopted for constructing the prototypes. The semantic role prototype is usually a vector, obtained by averaging the most typical fillers, and plausibility of new fillers depends on their similarity to the prototype, assessed by means of vector cosine (the standard similarity measure for DSMs; see Turney and Pantel (2010)).

Its merits notwithstanding, we argue that this method is not optimal for characterizing roles. Distributional vectors are typically built as out-of-

[3] Code: https://github.com/esantus/Thematic_Fit

[4] In this paper, we will make reference to two different models of DM: DepDM and TypeDM. DepDM counts the frequency of dependency links between words (e.g. *read, obj, book*), while TypeDM uses the variety of surface forms that express the link between words, rather than the link itself.

context representations, and they conflate different senses. By building the prototype as the centroid of a cluster of vectors and measuring then the thematic fit with vector cosine, the plausibility score is inevitably affected by many contexts that are irrelevant for the specific verb-argument combination.[5] This is likely to be one of the main reasons behind the difficulties of modeling roles other than agent and patient with syntax-based DSMs. We claim that improving the prototype representation might lead to a better characterization of thematic roles, and to a better treatment of polysemy.

When a verb and an argument are composed, humans are intuitively able to select only the part of the potential meaning of the words that is relevant for the concept being expressed (e.g. in *The player hit the ball*, humans would certainly exclude from the meaning of *ball* semantic dimensions that are strictly related to its dancing sense). In other words, not all the features of the semantic representations are active, and the composition process makes some features more 'prominent', while moving others to the background.[6]

Although we are not aware of experimental works specifically dedicated to verb-argument composition, a similar idea has been supported in studies on conceptual combinations (Hampton, 1997, 2007): when a head and a modifier are combined, their interaction affects the saliency of the features in the original concepts. For example, in *racing car*, the most salient properties would be those related to SPEED, whereas in *family car* SPACE properties would probably be more prominent. Yeh and Barsalou (2006) used a property priming experiment to show how the concept features activated during language comprehension vary across the background situations described by the sentence they occur in. When concepts are combined in a sentence, the features that are relevant for the specific combination are activated and are then easier to verify for human subjects.

The same could be true for linguistically-derived properties of lexical meaning: Simmons et al. (2008) brought neuroimaging evidence of the early activation of word association areas during property generation tasks, and Santos et al. (2011) showed that word associates are often among the properties generated for a given concept. Such findings suggest that, while we combine concepts, both embodied simulations and word distributions influence property salience (Barsalou et al., 2008).

Our model makes the following assumptions:

- the composition between a verb role representation and an argument shares the same cognitive mechanism underlying conceptual combinations;

- at least part of semantic representations is derived from, and/or mirrored in, linguistic data.[7] Consistently, the process of selecting the relevant features of the concepts being composed corresponds to modify the salience of the dimensions of distributional vectors;

- thematic fit computation is carried out on the basis of the activation and selection of salient features of a verb thematic role prototype and of the candidate argument filler vectors.

We rely on syntax-based DSMs, using dependency relations to approximate verb-specific roles and to identify their most typical fillers: for agents/patients, we extract the most frequent subjects/objects, for instruments we use the prepositional complements introduced by *with*, and for locations those introduced by either *on, at* or *in*.

Assuming that the linguistic features of distributional vectors correspond to the properties of conceptual composition processes, a candidate filler can be represented as a sorted distributional vector of the filler term, in which the most salient contexts occupy the top positions. Similarly, the abstract representation of a verb-specific role is a sorted prototype-vector, whose features derive from the sum of the most typical filler vectors for that verb-specific role.

Differently from Baroni and Lenci, the core and novel aspect of our proposal, described in the following subsections, is that we do not simply measure the correlation between all the features of candidate and prototype vectors (as vector cosine would do on unsorted vectors), but rather we *rank* and *filter* the features, computing the *weighted overlap* with a rank-based similarity measure inspired by $APSyn$, a recent proposal by Santus

[5] For an overview on the limitations of vector cosine, see: Li and Han (2013); Dinu et al. (2015); Schnabel et al. (2015); Faruqui et al. (2016); Santus et al. (2016a).

[6] An early proposal going in this direction is the predication theory by Kintsch (2001), which exploited Latent Semantic Analysis to select only the vector features that are appropriate for predicate-argument composition.

[7] See also the so-called 'strong version' of the Distributional Hypothesis (Miller and Charles, 1991; Lenci, 2008).

et al. (2016a,b,c) which has shown interesting results in synonymy detection and similarity estimation. As we will show in the next sections, the new metric assigns high scores to candidate fillers sharing many salient contexts with the verb-specific role prototype.

3.1 Typical Fillers

The first step of our method consists in identifying the typical fillers of a verb-specific role. Following Baroni and Lenci (2010), we weighted the raw co-occurrences between verbs, syntactic relations and fillers in the TypeDM tensor of DM with Positive Local Mutual Information (PLMI; Evert (2004)).

Given the co-occurrence count O_{vrf} of the verb v, a syntactic relation r and the filler f, we computed the expected count E_{vrf} under the assumption of statistical independence:

$$PLMI(v,r,f) = log\left(\frac{O_{v,r,f}}{E_{v,r,f}}\right) * O_{v,r,f} \tag{1}$$

From the ranked list of *(v,r,f)* tuples, for each slot, we selected as typical fillers the top k lexemes with the highest PLMI scores (see examples in Table 1, *Typical Fillers* column). In our experiments, we report results for $k = \{10, 30, 50\}$.

3.2 Role Prototype Vectors

To represent the typical fillers, the candidate fillers and the verb-specific role prototypes (which are obtained by summing their typical filler vectors), we built a syntax-based DSM. This includes *relation:word* contexts, like *sbj:dog, obj:apple, etc.*.

Contexts were weighted with Positive Pointwise Mutual Information (PPMI; Church and Hanks (1990), Bullinaria and Levy (2012), Levy et al. (2015)). Given a context c and a word w, the PPMI is defined as follows:

$$PPMI(w,c) = max(PMI(w,c), 0) \tag{2}$$

$$PMI(w,c) = log\left(\frac{P(w,c)}{P(w)P(c)}\right) = log\left(\frac{|w,c|D}{|w||c|}\right) \tag{3}$$

where *w* is the target word, *c* is the given context, *P(w,c)* is the probability of co-occurrence, and *D* is the collection of observed word-context pairs.[8]

The context c of the prototype vector P representing a thematic role has a value corresponding to the sum of the values of c for each of the k typical fillers used to build P. The contexts of P are then sorted according to their weight. Desirably, the highest-ranking contexts for a role prototype will be those that are more strongly associated with many of its typical fillers. Such *second order contexts* correspond to the most salient features of the verb-specific thematic role, as they are salient for many role fillers (some examples are reported in Table 1, *Top Second Order Contexts* column).

In summary, we built centroid vectors for our verb-specific thematic roles by means of *second order contexts*, which are first order dependency-based contexts of the most typical fillers of a verb-specific role. Since we are interested only in the most salient contexts, we ranked the centroid contexts according to their PPMI score, and we took the resulting rank as a distributional characterization of the thematic roles.

3.3 Filtering the Contexts

Filtering the prototype dimensions according to syntactic criteria might be useful to improve our role representations. It is, indeed, reasonable to hypothesize that predicates co-occurring with the typical patients of a verb are more relevant for the characterization of its patient role than – let's say – prepositional complements, as they correspond to other actions that are typically performed on the same patients.

Imagine that *apple, pizza, cake* etc. are among the most salient fillers for the OBJ slot of *to eat*, and that *OBJ-1:slice-v, OBJ-1:devour-v, SBJ:kid-n, INSTRUMENT:fork-n, LOCATION:table-n* are some of the most salient contexts of the prototype.[9] Things that are typically *sliced* and/or *devoured* are more likely to be good fillers for the patient role *to eat* than things that are simply located on a *table* or that are patients of actions performed by *kids*. To test this hypothesis, we evaluated the performance of the system in three different settings, each of which selecting:

[8] A variant of this DSM weighted with PLMI (which is simply the PPMI multiplied by the word-context frequency) was also built, but because of its lower and inconsistent performance we will not discuss it further. Santus et al. (2016c) previously showed that their rank-based measure performs worse on PLMI-weighted vectors, as they are biased towards frequent contexts.

[9] Our DSM also makes use of inverse syntactic dependencies: *target SYN-1 context* means that *target* is linked to *context* by the dependency relation *SYN* (e.g. *meal OBJ-1 devour* means that *meal* is *OBJ* of *devour*).

	Typical Fillers	Top Second Order Contexts
subject: cure-v	treatment-n, drug-n, resin-n, doctor-n, surgery-n, medicine-n, therapy-n, antibiotic-n, dose-n, operation-n, water-n...	obj-1:prescribe-v, sbj-1:prescribe-v, sbj-1:prevent-v, sbj-1:contraindicate-v, [...]
object: abandon-v	plan-n, idea-n, project-n, attempt-n, position-n, principle-n, policy-n, ship-n, practice-n, hope-n, fort-n, claim-n...	obj-1:revive-v, obj-1:defend-v, obj-1:renounce-v, obj-1:espouse-v, sbj-1:entail-v...
instrument: eat-v	bread-n, hand-n, spoon-n, sauce-n, relish-n, fork-n, finger-n, meal-n, knife-n, friend-n, chopstick-n, rice-n, food-n...	obj-1:flavour-v, obj-1:taste-v, obj-1:spoon-v, sbj-1:taste-v, obj-1:slice-v in:bowl-n...
location: walk-v	in:direction-n, at:time, at:pace-n, on:path-n, at:night, on:side-n, at:end, on:beach-n, on:leg, in:area, in:way...	obj-1:wander-v, obj-1:stroll-v, obj-1:litter-v, obj-1:sweep-v, sbj-1:slope-v, obj-1:tread-v...

Table 1: Typical fillers and top *second order contexts* for several verb-specific roles.

- only predicates in a subject/object relation (**SO** setting);
- only prepositional complements (**PREP** setting);
- both of them (**ALL** setting).

3.4 Computing the Thematic Fit

Our hypothesis is that fillers whose salience-ranked vector has a large overlap with the prototype representation should have a high thematic fit. Such overlap should take into account not only the number of shared features, but also their respective ranks in the salience-ranked vectors.

When the prototype has been computed and the candidate filler vector has also been sorted, we can measure the *Weighted Overlap* by adapting $APSyn$ (Santus et al., 2016a,b,c) to our needs:

$$WO(w_x, w_y) = \sum_{\forall f \epsilon (x_{[1:N]} \cap y_{[1:N]})} \frac{1}{avg(r_x(f), r_y(f))} \qquad (4)$$

where for every feature f in the intersection between the top N features of the sorted vectors x, $x_{[1:N]}$, and y, $y_{[1:N]}$, we sum 1 divided by the average rank of the shared feature in x and y, $r_x(f)$ and $r_y(f)$ (N is a tunable parameter).

This measure assigns the maximum score to vectors sharing exactly the same dimensions, in the same salience ranking. The lower the rank of a shared context in the sorted vector, the smaller its contribution to the thematic fit score. If the feature set intersection is empty, the score will be 0.

Differently from cosine similarity, which conflates multiple senses, measuring the *Weighted Overlap* between prototype and candidate filler can improve the estimation of the thematic fit by favoring the appropriate word senses: for example, for a verb-argument pair like *embrace-v–communism-n*, *communism-n* is likely to intersect and to increase the saliency (through the average rank) only of the second-order features of *embrace-v* referring to its abstract sense.

Data	Our system	BL2010	SD2014	G2015	T2016
Padó	96	100	99	100	99
McRae	100	95	96	95	96
Instr.	100	93	94	93	94
Loc.	96	99	100	99	100

Table 2: Dataset coverage (%) for all systems.

4 Experiments

Datasets. We tested our method on three popular datasets for thematic fit estimation, namely McRae et al. (1998), Ferretti et al. (2001) and Padó (2007). All the datasets contain human plausibility judgments for verb-role-filler triples. McRae and Padó include scores for agent and patient roles, whereas Ferretti includes instruments and locations (see Table 2 for the coverage of each system for the datasets).

Metrics. Performance is evaluated as the Spearman correlation between the scores of the systems and the human plausibility judgments.

Fillers. In order to make our results more comparable with previous studies, the typical fillers for each verb role were extracted from the TypeDM tensor of the Distributional Memory framework (see Section 3.1).[10] Those were the same fillers used by Baroni and Lenci (2010) and Greenberg et al. (2015b).

DSM. Distributional information is derived from the concatenation of two corpora: the British National Corpus (Leech, 1992) and Ukwac (Baroni et al., 2009). Both were parsed with the Maltparser (Nivre and Hall, 2005). From this concatenation, we built a dependency-based DSMs, weighted with PPMI, containing 20,145 targets (i.e. nouns and verbs with frequency above 1000) and 94,860 contexts. The syntactic relations taken into account were: *sbj, sbj-1, obj, obj-1, at-1, in-1, on-1, with-1*.

Settings. To prove our hypotheses and verify the consistency of the system, we tested a large range of settings, varying:

[10] http://clic.cimec.unitn.it/dm/

Weight	N	# Fillers	Padó			Mcrae			Ferretti - Instruments			Ferretti - Locations		
			ALL	SO	PREP	ALL	SO	PREP	ALL	SO	PREP	ALL	SO	PREP
PPMI	**2000**	**10**	0.43	0.45	0.26	0.25	0.27	0.19	0.43	0.41	0.46	0.25	0.27	0.28
		30	0.47	0.49	0.33	0.26	0.28	0.22	0.42	0.41	**0.50**	0.28	0.31	0.37
		50	0.46	**0.50**	0.35	0.27	**0.29**	0.24	0.39	0.38	0.47	0.28	0.32	**0.39**
Vector Cosine (Baseline)		**10**	0.43			0.25			0.42			0.29		
		30	0.47			0.26			0.41			0.32		
		50	0.48			0.26			0.38			0.31		
State of the Art														
Baroni and Lenci (2010)			0.53			0.33			0.36			0.23		
Sayeed and Demberg (2014)			0.56			0.27			0.28			0.13		
Greenberg et al. (2015)			0.53			0.36			0.42			0.29		
Tilk et al. (2016)			0.52			0.38			0.45			0.44		

Table 3: Results for Padó, McRae and Ferretti, Instruments and Locations, with WO computed on PPMI matrix, varying the number of fillers (i.e. 10, 30 and 50) and the types of dependency contexts (i.e. ALL, SO and PREP). The best results of our system are in bold. A baseline reimplementing Baroni and Lenci (2010) – with 10, 30 and 50 fillers – and state of the art results from previous literature are reported for comparison.

- the number of fillers used to build the prototype, with the most typical values in the literature ranging between 10 and 50. We report the results for 10, 30 and 50 fillers
- the types of the dependency relations used for calculating the overlap: we report results for the SO, PREP and ALL settings;
- the value of N, that is the number of top contexts that we take into account when computing the weighted overlap. Table 3 reports the scores for our best setting, while the performances for other values of N are discussed in the Section 5.

Baseline and State of the Art. As a baseline, we use the thematic fit model by Baroni and Lenci (2010), with no ranking of the features of the prototypes and with vector cosine as a similarity metric.[11] Results are reported for 10, 30 and 50 fillers. For reference, we also report the results of state-of-the-art models, both the unsupervised (Baroni and Lenci, 2010; Sayeed and Demberg, 2014; Greenberg et al., 2015b) and the supervised ones (Tilk et al., 2016).

5 Results

Table 3 describes the performance of the best setting (weight: PPMI; N=2000). In the first three rows, the table shows the scores obtained by our system varying the types of dependency contexts (i.e. ALL, SO, PREP) and the number of fillers considered for the prototype (i.e. 10, 30 and 50). The other rows respectively show i) the scores obtained by calculating the vector cosine between the role prototype vector (i.e. the vector obtained by summing the most typical fillers, with no salience ranking of the dimensions) and the candidate filler vector and ii) the scores reported in the literature for the best unsupervised and supervised models.

At a glance, our best scores always outperform the reimplementation of Baroni and Lenci, being mostly competitive with the state of the art models. More precisely, for agents and patients the performance is close to the reported scores for DM, when only predicates are used in the WO calculation, as hypothesized in Section 3.3. The neural network of Tilk and colleagues retains a significant advantage on our models only for the McRae dataset. Our system, however, shows a remarkable improvements on the Ferretti's datasets, and specifically on Ferretti-Instruments, when only complements are used (see Section 3.3), outperforming even the supervised and more complex model by Tilk et al. (2016), which has access to semantic roles information. Compared to the other unsupervised models, our system has a statistically significant advantage over Baroni and Lenci (2010) on the locations dataset and over Sayeed and Demberg (2014) on the locations and on the instruments dataset ($p < 0.05$).[12]

At the best of our knowledge, the result for the

[11] This baseline is equivalent to the approach of Baroni and Lenci (2010), except for the fact that it is applied on a standard dependency-based DSM and not on TypeDM, which combines dependency links and handcrafted lexico-syntactic patterns: see Section 2.

[12] p-values computed with Fisher's r-to-z transformation.

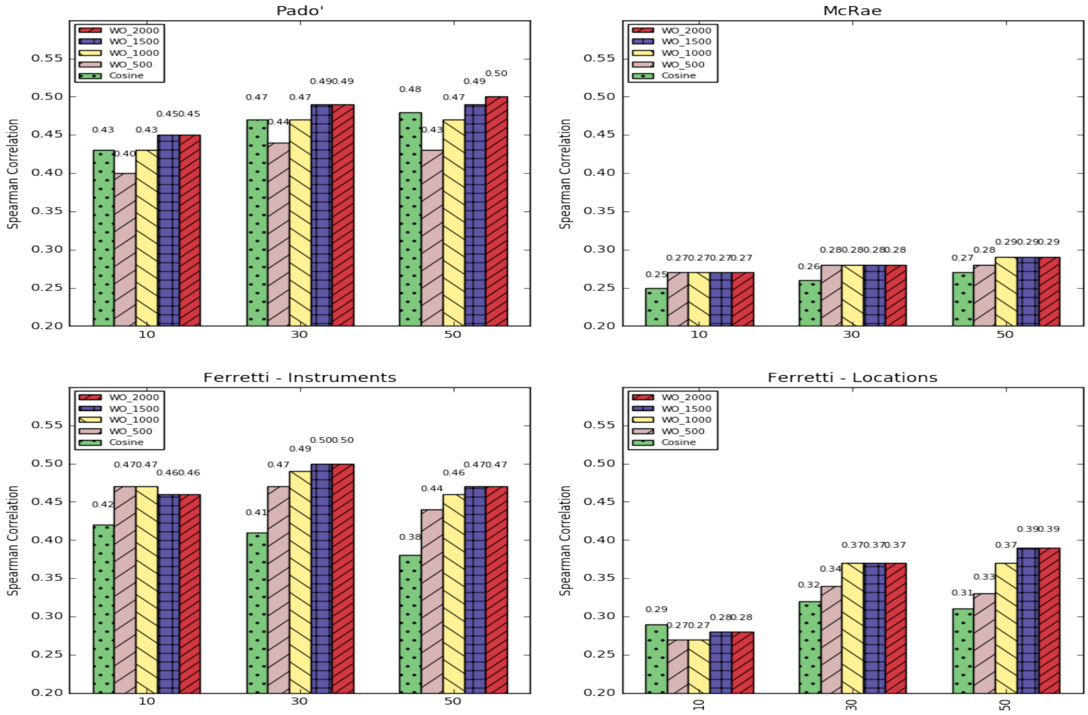


Figure 1: Results for Padó, McRae and Ferretti, Instruments and Locations, with WO (respectively SO and PREP) computed on PPMI matrix, varying the number of fillers (i.e. 10, 30 and 50) and the value of N (i.e. 500, 1000, 1500 and 2000). A baseline reimplementing Baroni and Lenci (2010) – with 10, 30 and 50 fillers – is also reported in every test for comparison.

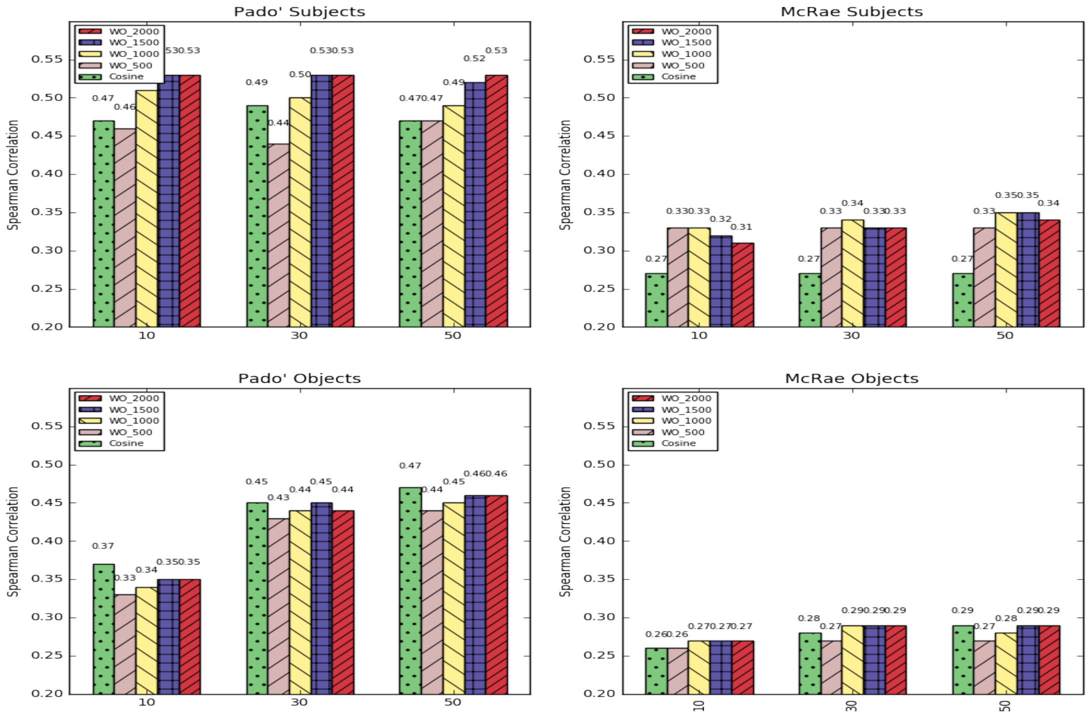


Figure 2: Results for the agent and patient roles in Padó and McRae, with WO (SO) computed on PPMI matrix, varying the number of fillers (i.e. 10, 30 and 50) and the value of N (i.e. 500, 1000, 1500 and 2000). A baseline reimplementing Baroni and Lenci (2010) – with 10, 30 and 50 fillers – is also reported in every test for comparison.

		BEST 35				WORST 35			
	Metric	Avg. Gold	Overlap	Syntax	Lexemes	Avg. Gold	Overlap	Syntax	Lexemes
McRae (k=50, Pred)	Cos	4.90	3	14 sbj, 21 obj	3 sentence, 2 devour, 2 scratch...	4.75	26	24 sbj, 11 obj	2 consider, 2 entertain, 2 scrub
	WO 2000	4.20	4	17 sbj, 18 obj	2 haunt	4.15	26	23 sbj, 12 obj	2 admire, 2 arrest, 2 consider, 2 entertain
Padó (k=50, Pred)	Cos	4.07	10	12 sbj, 23 obj	4 advise, 4 eat, 4 embarrass	4.77	16	17 sbj, 18 obj	9 tell, 7 kill, 4 see
	WO 2000	4.35	10	21 sbj, 14 obj	3 confuse, 3 hear, 3 promise, 3 raise	4.68	16	15 sbj, 20 obj	7 resent, 5 increase, 4 hear, 4 see
Ferretti - Instruments (k=30, Compl)	Cos	4.53	16	35 with	3 hung, 3 eat, 3 teach	4.51	22	35 with	4 repair, 3 teach, 3 inflate
	WO 2000	5.06	15	35 with	3 dig, 3 hunt	4.49	22	35 with	3 repair, 2 attract, 2 dig, 2 draw, 2 drink...
Ferretti - Locations (k=50, Compl)	Cos	5.15	11	35 on/at/in	3 draw, 3 rescue	4.72	23	35 on/at/in	3 run, 2 wait, 2 wash, 2 shower...
	WO 2000	4.97	11	35 on/at/in	3 browse, 3 eat, 3 mingle, 3 rescue	4.47	23	35 on/at/in	3 run, 2 draw, 2 exercise, 2 shower, 2 wait...

Table 4: Average gold values, number of items listed for both metrics, and distribution of syntactic and lexical forms among the 35 best and worst correlated items for every measure in the given datasets.

instruments is the best reported until now in the literature. This is particularly interesting because – as pointed out by Sayeed and Demberg (2014) – instruments and locations are difficult to model for a dependency-based system, given the ambiguity of prepositional phrases (e.g. *with* does not only encode instruments, but it can also encode other roles, such as in *I ate a pizza with Mark*). We think this is the main reason behind the different trend observed for the Instruments datasets with respect to the number of the fillers (see Table 3 and Figure 1). Unlike all the other datasets, instrument prototypes built with more fillers tend to be more noisy and therefore to pull down both the vector cosine and WO performance (this is partially true also for locations, where the performances – for cosine and WO with a lower number of contexts – drop with more than 30 fillers: see Figure 1). Systems based on semantic role labeling have an advantage in this sense, as they do not have to deal with prepositional ambiguity.

Our results show that, by weighting and filtering the features of the role prototype, dependency-based approaches can be successful in modeling roles other than agent and patient, eventually dealing also with the ambiguity of prepositional phrases.

Settings. Apart from the above-mentioned exceptions, the best scores are obtained building the prototypes with a higher number of fillers, typically with 50, and calculating the WO only with a syntactically-filtered set of contexts. More specifically, Padó and McRae benefit from the calculation of WO using only second order subject-object predicates (i.e. SO), while Ferretti-Instruments and Ferretti-Locations benefit from the exclusive use of prepositional complements (i.e. PREP). On the other hand, the opposite setting (e.g. SO for Ferretti-Instruments and Ferretti-Locations and PREP for Padó and McRae) leads to much lower scores, whereas the full vectors (i.e. ALL) tend to have a stable-but-not-excellent performances on all datasets.

As briefly mentioned above, in our experiments, we tested both PPMI and PLMI as weighting measures. Table 3 only reports PPMI scores because it performs more regularly than PLMI, whose behaviour is often unpredictable.

A parameter that has an impact on the performance of our system is the value of N, which is the number of *second order contexts* that are considered when calculating the WO. We have noticed that the performance of WO is directly related to the growth of N, and this can be noticed in Figure 1, where WO is plotted for the different values of N with every combination of dataset and number of fillers. For space reasons, the plot only contains the performance for the best type of *second order contexts* for each dataset (i.e. SO for Padó and McRae and COMP for Ferretti-Locations and Ferretti-Instruments). As it can be seen in Figure 1, the scores of WO tend to grow with the growth of N in all datasets. Interestingly, they are largely above the competitive baseline in most of the cases, the only exceptions being Padó (where a large N is necessary to outperform the baseline) and Ferretti-Locations with 10 fillers (prepositional ambiguity might have caused the introduction of noisy fillers among the top ones).

Agent & Patient. In order to further evaluate our system, we have split Padó and McRae datasets into agent and patient subsets. Figure 2 describes the performance of WO and vector cosine baseline while varying N and the number of fillers. The plot shows a clearly better performance of WO for the agent role (i.e. *subject*), especially when N is equal or over 1000 (note that the value of N has little impact in the agent subset of the McRae dataset). Such advantage, however, is reduced for the patient role (i.e. *object*). This is particularly interesting because we do not observe large drops in performance for the vector cosine between agent and patient role (except for Padó, $k = 10$). The drop is particularly noticeable in Padó, a dataset which has several non-constraining verbs (especially for the patient role: a similar observation was also made by Tilk et al. (2016)). As the constraints on the typical fillers of such verbs are very loose, we hypothesize that it is more difficult to find a set of salient features that are shared by many typical fillers. Therefore, estimations based on the whole vectors turn out to be more reliable. This can be confirmed by looking at the worst correlated words reported in *Lexemes* column, in Table 4.

5.1 Error Analysis

We performed an error analysis to verify – for the best settings of WO in each dataset – the correlation between vector cosine and WO scores (see Table 5), and the peculiarities of the entries with the strongest and the weakest correlation (see Table 4).

We found that WO and vector cosine always have a high correlation (i.e. above 0.80), with the highest correlations reported for McRae and Ferretti-Instruments. Looking at Table 4 we can also observe that:

- the average gold value of the 35 most (4.65) and least (4.56) correlated items does not substantially differ from the average gold value calculated on the full datasets (4.31), meaning that the distribution of likely and unlikely fillers among the best and worst correlated items is similar to the one in the datasets (i.e. no bias can be identified);
- both measures have difficulties on the same test items (probably because of loose semantic constraints), but report their best performances on different pairs (see *Overlap* and *Lexemes* columns);
- syntactically, vector cosine correlates better with objects, while WO is more balanced between objects and subjects, often showing a preference for the latter (see the distribution in *Syntax* column).

6 Conclusions

In this paper, we have introduced an unsupervised distributional method for modeling predicate-argument thematic fit judgments which works purely on syntactic information.

Dataset	Correlation
McRae	0.88
Padó	0.81
Ferretti - Instruments	0.90
Ferretti - Locations	0.83

Table 5: Correlation between WO and vector cosine in WO best settings for all datasets

The method, inspired by cognitive and psycholinguistic findings, consists in: i) extracting and filtering the most salient *second order contexts* for each verb-specific role, i.e. the most salient semantic dimensions of *typical verb-specific role fillers*; and then ii) estimating the thematic fit as a *weighted overlap* between the top features of the candidate fillers and of the prototypes. Once tested on some popular datasets of thematic fit judgments, our method consistently outperforms a baseline re-implementing the thematic fit model of Baroni and Lenci (2010) and proves to be competitive with state of the art models. It even registered the best performance on the Ferretti-Instruments dataset and it is the second best on the Ferretti-Locations, which were known to be particularly hard to model for dependency-based approaches.

Our method is simple, economic and efficient, it works purely on syntactic dependencies (so it does not require a role-labeled corpus) and achieves good results even with no supervised training. Finally, it offers linguistically and cognitively grounded insights on the process of prototype creation and contextual feature salience, preparing the ground for further speculations and optimizations. For example, future work might aim at identifying strategies for tuning the parameter N to account for the different degrees of selectivity of each verb-specific role. Another possible extension would be the inclusion of a mechanism for updating the role prototypes depending on how the other roles are filled, which would be the key for a more realistic and dynamic model of thematic fit expectations (Lenci, 2011).

Acknowledgments

We would like to thank the anonymous reviewers for their helpful suggestions.

This work has been carried out thanks to the support of the A*MIDEX grant (nANR-11-IDEX-0001-02) funded by the French Government “Investissements d’Avenir” program.

References

Gerry T.M Altmann and Yuki Kamide. 1999. Incremental Interpretation at Verbs: Restricting the Domain of Subsequent Reference . *Cognition* 73(3):247 – 264.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: a Collection of Very Large Linguistically Processed Web-crawled Corpora. *Language Resources and Evaluation* 43(3):209–226.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Dont Count, Predict! A Systematic Comparison of Context-counting vs. Context-predicting Semantic Vectors. In *Proceedings of ACL*. volume 1.

Marco Baroni and Alessandro Lenci. 2010. Distributional Memory: A General Framework for Corpus-based Semantics. *Computational Linguistics* 36(4):673–721.

Lawrence W Barsalou, Ava Santos, W Kyle Simmons, and Christine D Wilson. 2008. Language and Simulation in Conceptual Processing. *Symbols, embodiment, and meaning* pages 245–283.

Klinton Bicknell, Jeffrey L Elman, Mary Hare, Ken McRae, and Marta Kutas. 2010. Effects of Event Knowledge in Processing Verbal Arguments. *Journal of Memory and Language* 63(4):489–505.

John A. Bullinaria and Joseph P. Levy. 2012. Extracting Semantic Representations from Word Co-occurrence Statistics: Stop-lists, Stemming, and SVD. *Behavior Research Methods* 44(3):890–907.

Kenneth Ward Church and Patrick Hanks. 1990. Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics* 16(1):22–29.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research* 12:2493–2537.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving Zero-shot Learning by Mitigating the Hubness Problem. In *Proceedings of ICLR*.

Katrin Erk. 2007. A Simple, Similarity-based Model for Selectional Preferences. In *Proceedings of ACL*.

Katrin Erk, Sebastian Padó, and Ulrike Padó. 2010. A flexible, corpus-driven model of regular and inverse selectional preferences. *Computational Linguistics* 36:723–763.

Stefan Evert. 2004. *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. Ph.D. thesis.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems With Evaluation of Word Embeddings Using Word Similarity Tasks. In *Proceedings of ACL Workshop on Evaluating Vector Space Representations for NLP*.

Todd R. Ferretti, Ken McRae, and Andrea Hatherell. 2001. Integrating Verbs, Situation Schemas, and Thematic Role Concepts . *Journal of Memory and Language* 44(4):516 – 547.

Clayton Greenberg, Vera Demberg, and Asad Sayeed. 2015a. Verb Polysemy and Frequency Effects in Thematic Fit Modeling. In *Proceedings of NAACL Workshop on Cognitive Modeling and Computational Linguistics*.

Clayton Greenberg, Asad B. Sayeed, and Vera Demberg. 2015b. Improving Unsupervised Vector-space Thematic Fit Evaluation via Role-filler Prototype Clustering. In *Proceedings of HLT-NAACL*.

James A Hampton. 1997. Conceptual Combination. *Knowledge, Concepts, and Categories* pages 133–159.

James A. Hampton. 2007. Typicality, Graded Membership, and Vagueness. *Cognitive Science* 31:355–384.

Mary Hare, Michael Jones, Caroline Thomson, Sarah Kelly, and Ken McRae. 2009. Activating Event Knowledge. *Cognition* 111 2:151–67.

Yuki Kamide, Gerry T.M Altmann, and Sarah L Haywood. 2003. The Time-course of Prediction in Incremental Sentence Processing: Evidence from Anticipatory Eye Movements . *Journal of Memory and Language* 49(1):133 – 156.

Walter Kintsch. 2001. Predication. *Cognitive Science* 25:173–202.

Geoffrey Leech. 1992. 100 Million Words of English: the British National Corpus (BNC). *Language Research* 28(1):1–13.

Alessandro Lenci. 2008. Distributional Semantics in Linguistic and Cognitive Research. *Italian Journal of Linguistics* 20(1):1–31.

Alessandro Lenci. 2011. Composing and Updating Verb Argument Expectations: A Distributional Semantic Model. In *Proceedings of ACL Workshop on Cognitive Modeling and Computational Linguistics*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *TACL* 3:211–225.

Baoli Li and Liping Han. 2013. Distance Weighted Cosine Similarity Measure for Text Classification. In *Intelligent Data Engineering and Automated Learning*. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 611–618.

Maryellen C MacDonald and Mark S Seidenberg. 2006. Constraint Satisfaction Accounts of Lexical and Sentence Comprehension. *Handbook of psycholinguistics* 2:581–611.

Kazunaga Matsuki, Tracy Chow, Mary Hare, Jeffrey L Elman, Christoph Scheepers, and Ken McRae. 2011. Event-based Plausibility Immediately Influences On-line Language Comprehension. *Journal of experimental psychology. Learning, memory, and cognition* 37 4:913–34.

Ken McRae, Mary Hare, Jeffrey L. Elman, and Todd Ferretti. 2005. A basis for generating expectancies for verbs from nouns. *Memory & Cognition* 33(7):1174–1184.

Ken McRae and Kazunaga Matsuki. 2009. People Use their Knowledge of Common Events to Understand Language, and Do So as Quickly as Possible. *Language and Linguistics Compass* 3(6):1417–1429.

Ken McRae, Michael J. Spivey-Knowlton, and Michael K. Tanenhaus. 1998. Modeling the Influence of Thematic Fit (and Other Constraints) in Online Sentence Comprehension. *Journal of Memory and Language* 38(3):283–312.

George A. Miller and Walter G. Charles. 1991. Contextual Correlates of Semantic Similarity. *Language and Cognitive Processes* 6(1):1–28.

Jeff Mitchell and Mirella Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive science* 34(8):1388–1429.

Joakim Nivre and Johan Hall. 2005. Maltparser: A Language-independent System for Data-driven Dependency Parsing. In *Proceedings of Workshop on Treebanks and Linguistic Theories*. pages 13–95.

Ulrike Padó. 2007. *The Integration of Syntax and Semantic Plausibility in a Wide-coverage Model of Human Sentence Processing*. Ph.D. thesis.

Ava Santos, Sergio E. Chaigneau, W. Kyle Simmons, and Lawrence W. Barsalou. 2011. Property Generation Reflects Word Association and Situated Simulation. *Language and Cognition* 3(1):83119.

Enrico Santus, Emmanuele Chersoni, Alessandro Lenci, Chu-Ren Huang, and Philippe Blache. 2016a. Testing APSyn against Vector Cosine on Similarity Estimation. In *Proceedings of PACLIC*.

Enrico Santus, Tin-Shing Chiu, Qin Lu, Alessandro Lenci, and Chu-Ren Huang. 2016b. Unsupervised Measure of Word Similarity: how to Outperform Co-occurrence and Vector Cosine in VSMs. In *Proceedings of the AAAI*. AAAI Press, pages 4260–4261.

Enrico Santus, Tin-Shing Chiu, Qin Lu, Alessandro Lenci, and Chu-Ren Huang. 2016c. What a Nerd! Beating Students and Vector Cosine in the ESL and TOEFL Datasets. In *Proceedings of LREC*.

Asad Sayeed and Vera Demberg. 2014. Combining Unsupervised Syntactic and Semantic Models of Thematic Fit. In *Proceedings of CLIC*.

Asad Sayeed, Vera Demberg, and Pavel Shkadzko. 2015. An Exploration of Semantic Features in an Unsupervised Thematic Fit Evaluation Framework. In *Italian Journal of Linguistics*.

Asad Sayeed, Clayton Greenberg, and Vera Demberg. 2016. Thematic Fit Evaluation: an Aspect of Selectional Preferences. In *Proceedings of ACL Workshop for Evaluating Vector Space Representations for NLP*.

Tobias Schnabel, Igor Labutov, David M. Mimno, and Thorsten Joachims. 2015. Evaluation Methods for Unsupervised Word Embeddings. In *Proceedings of EMNLP*.

W Kyle Simmons, Stephan B Hamann, Carla L Harenski, Xiaoping P Hu, and Lawrence W Barsalou. 2008. fMRI Evidence for Word Association and Situated Simulation in Conceptual Processing. *Journal of Physiology* 102 1-3:106–19.

Ottokar Tilk, Vera Demberg, Asad B. Sayeed, Dietrich Klakow, and Stefan Thater. 2016. Event Participant Modelling with Neural Networks. In *Proceedings of EMNLP*.

Peter D. Turney and Patrick Pantel. 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research* 37:141–188.

Bram Vandekerckhove, Dominiek Sandra, and Walter Daelemans. 2009. A Robust and Extensible Exemplar-Based Model of Thematic Fit. In *Proceedings of EACL*.

Wenchi Yeh and Lawrence W Barsalou. 2006. The Situated Nature of Concepts. *The American Journal of Psychology* 119:3:349–84.

SCDV : Sparse Composite Document Vectors using soft clustering over distributional representations

Dheeraj Mekala*
IIT Kanpur
dheerajm@iitk.ac.in

Vivek Gupta*
Microsoft Research
t-vigu@microsoft.com

Bhargavi Paranjape
Microsoft Research
t-bhpara@microsoft.com

Harish Karnick
IIT Kanpur
hk@iitk.ac.in

Abstract

We present a feature vector formation technique for documents - Sparse Composite Document Vector (SCDV) - which overcomes several shortcomings of the current distributional paragraph vector representations that are widely used for text representation. In SCDV, word embeddings are clustered to capture multiple semantic contexts in which words occur. They are then chained together to form document topic-vectors that can express complex, multi-topic documents. Through extensive experiments on multi-class and multi-label classification tasks, we outperform the previous state-of-the-art method, NTSG (Liu et al., 2015a). We also show that SCDV embeddings perform well on heterogeneous tasks like Topic Coherence, context-sensitive Learning and Information Retrieval. Moreover, we achieve significant reduction in training and prediction times compared to other representation methods. SCDV achieves best of both worlds - better performance with lower time and space complexity.

1 Introduction

Distributed word embeddings represent words as dense, low-dimensional and real-valued vectors that can capture their semantic and syntactic properties. These embeddings are used abundantly by machine learning algorithms in tasks such as text classification and clustering. Traditional bag-of-word models that represent words as indices into a vocabulary don't account for word ordering and long-distance semantic relations. Representations based on neural network language models (Mikolov et al., 2013b) can overcome these flaws and further reduce the dimensionality of the vectors. The success of the method is recently mathematically explained using the random walk on discourses model (Arora et al., 2016a). However, there is a need to extend word embeddings to entire paragraphs and documents for tasks such as document and short-text classification.

Representing entire documents in a dense, low-dimensional space is a challenge. A simple weighted average of the word embeddings in a large chunk of text ignores word ordering, while a parse tree based combination of embeddings (Socher et al., 2013) can only extend to sentences. (Le and Mikolov, 2014) trains word and paragraph vectors to predict context but shares word-embeddings across paragraphs. However, words can have different semantic meanings in different contexts. Hence, vectors of two documents that contain the same word in two distinct senses need to account for this distinction for an accurate semantic representation of the documents. (Ling et al., 2015), (Liu et al., 2015a) map word embeddings to a latent topic space to capture different senses in which words occur. However, they represent complex documents in the same space as words, reducing their expressive power. These methods are also computationally intensive.

In this work, we propose the Sparse Composite Document Vector(SCDV) representation learning technique to address these challenges and create efficient, accurate and robust semantic representations of large texts for document classification tasks. SCDV combines syntax and semantics learnt by word embedding models together with a latent topic model that can handle different senses of words, thus enhancing the expressive power of document vectors. The topic space is learnt efficiently using a soft clustering technique over embeddings and the final document vectors are made

*Represents equal contribution

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 659–669
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

sparse for reduced time and space complexity in tasks that consume these vectors.

The remaining part of the paper is organized as follows. Section 2 discusses related work in document representations. Section 3 introduces and explains SCDV in detail. This is followed by extensive and rigorous experiments together with analysis in section 4 and 5 respectively.

2 Related Work

(Le and Mikolov, 2014) proposed two models for distributional representation of a document, namely, *Distributed Memory Model Paragraph Vectors (PV-DM)* and *Distributed BoWs paragraph vectors (PV-DBoW)*. In *PV-DM*, the model is learned to predict the next context word using word and paragraph vectors. In *PV-DBoW*, the paragraph vector is directly learned to predict randomly sampled context words. In both models, word vectors are shared across paragraphs. While word vectors capture semantics across different paragraphs of the text, documents vectors are learned over context words generated from the same paragraph and potentially capture only local semantics (Singh and Mukerjee, 2015). Moreover, a paragraph vector is embedded in the same space as word vectors though it can contain multiple topics and words with multiple senses. As a result, doc2vec (Le and Mikolov, 2014) doesn't perform well on Information Retrieval as described in (Ai et al., 2016a) and (Roy et al., 2016). Consequently, we expect a paragraph vector to be embedded in a higher dimensional space.

A paragraph vector also assumes all words contribute equally, both quantitatively (weight) and qualitatively (meaning). They ignore the importance and distinctiveness of a word across all documents (Singh and Mukerjee, 2015). Mukerjee et al. (Singh and Mukerjee, 2015) proposed idf-weighted averaging of word vectors to form document vectors. This method tries to address the above problem. However, it assumes that all words within a document belong to the same semantic topic. Intuitively, a paragraph often has words originating from several semantically different topics. In fact, Latent Dirichlet Allocation (Blei et al., 2003) models a document as a distribution of multiple topics.

These shortcomings are addressed in three novel composite document representations called *Topical word embedding (TWE-1,TWE-2 and TWE-3)* by (Liu et al., 2015a). *TWE-1* learns word and topic embeddings by considering each topic as a pseudo word and builds the topical word embedding for each word-topic assignment. Here, the interaction between a word and the topic to which it is assigned is not considered. *TWE-2* learns a topical word embedding for each word-topic assignment directly, by considering each word- topic pair as a pseudo word. Here, the interaction between a word and its assigned topic is considered but the vocabulary of pseudo-words blows up. For each word and each topic, *TWE-3* builds distinct embeddings for the topic and word and concatenates them for each word-topic assignment. Here, the word embeddings are influenced by the corresponding topic embeddings, making words in the same topic less discriminative.

(Liu et al., 2015a) proposed an architecture called *Neural tensor skip-gram model (NTSG-1, NTSG-2, NTSG-3, NTSG-4)*, that learns multi-prototype word embeddings and uses a tensor layer to model the interaction of words and topics to capture different senses. $NTSG$ outperforms other embedding methods like $TWE-1$ on the 20 newsgroup data-set by modeling context-sensitive embeddings in addition to topical-word embeddings. $LTSG$ (Law et al., 2017) builds on $NTSG$ by jointly learning the latent topic space and context-sensitive word embeddings. All three, TWE, $NTSG$ and $LTSG$ use LDA and suffer from computational issues like large training time, prediction time and storage space. They also embed document vectors in the same space as terms. Other works that harness topic modeling like WTM (Fu et al., 2016), $w2v-LDA$ (Nguyen et al., 2015), $TV + MeanWV$ (Li et al., 2016a), $LTSG$ (Law et al., 2017), $Gaussian - LDA$ (Das et al., 2015), $Topic2Vec$ (Niu et al., 2015), (Moody, 2016) and $MvTM$ (Li et al., 2016b) also suffer from similar issues.

(Gupta et al., 2016) proposed a method to form a composite document vector using word embeddings and tf-idf values, called the *Bag of Words Vector (BoWV)*. In $BoWV$, each document is represented by a vector of dimension $D = K*d+K$, where K is the number of clusters and d is the dimension of the word embeddings. The core idea behind $BoWV$ is that semantically different words belong to different topics and their word vectors should not be averaged. Further, $BoWV$ computes inverse cluster frequency of each cluster (icf) by averaging the idf values of its member terms to capture the importance of words in the corpus. However, $BoWV$ does hard clustering using K-means algorithm, assigning each word to only one cluster or semantic topic but a word can belong to multiple topics. For example, the word *apple* belongs to topic *food* as a fruit, and belongs to topic *Information Technology* as an IT company. Moreover, $BoWV$ is a non-sparse, high dimensional continuous vector and suffers from computational problems like large training time, prediction time and storage requirements.

3 Sparse Composite Document Vectors

In this section, we present the proposed Sparse Composite Document Vector (SCDV) representation as a novel document vector learning algorithm. The feature formation algorithm can be divided into three steps.

3.1 Word Vector Clustering

We begin by learning d dimensional word vector representations for every word in the vocabulary V using the skip-gram algorithm with negative sampling (SGNS) (Mikolov et al., 2013a). We then cluster these word embeddings using the Gaussian Mixture Models(GMM) (Reynolds, 2015) soft clustering technique. The number of clusters, *K*, to be formed is a parameter of the SCDV model. By inducing soft clusters, we ensure that each word belongs to every cluster with some probability $P(c_k|w_i)$.

$$p(c_k = 1) = \pi_k$$

$$p(c_k = 1|w) = \frac{\pi_k \mathcal{N}(w|\mu_k, \Sigma_k)}{\Sigma_{j=1}^{K} \pi_j \mathcal{N}(w|\mu_j, \Sigma_j)}$$

3.2 Document Topic-vector Formation

For each word w_i, we create K different word-cluster vectors of d dimensions ($\vec{wcv_{ik}}$) by weighting the word's embedding with its probability distribution in the kth cluster, $P(c_k|w_i)$. We then concatenate all K word-cluster vectors ($\vec{wcv_{ik}}$) into a K×d dimensional embedding and weight it with inverse document frequency of w_i to form a word-topics vector ($\vec{wtv_i}$). Finally, for all words appearing in document D_n, we sum their word-topic vectors $\vec{wtv_i}$ to obtain the document vector $\vec{dv_{D_n}}$.

$$\vec{wcv_{ik}} = \vec{wv_i} \times P(c_k|w_i)$$

Algorithm 1: Sparse Composite Document Vector

```
Data: Documents $D_n$, n = 1 ... N
Result: Document vectors $\vec{SCDV}_{D_n}$, n = 1
        ... N
1  Obtain word vector ($\vec{wv_i}$), for each word $w_i$;
2  Calculate idf values, $idf(w_i)$, $i = 1..|V|$ ;
     /* $|V|$ is vocabulary size */
3  Cluster word vectors $\vec{wv}$ using GMM
     clustering into K clusters;
4  Obtain soft assignment $P(c_k|w_i)$ for word $w_i$
     and cluster $c_k$;
   /* Loop 5-10 can be
      pre-computed                          */
5  for each word $w_i$ in vocabulary $V$ do
6      for each cluster $c_k$ do
7          $\vec{wcv_{ik}} = \vec{wv_i} \times P(c_k|w_i)$;
8      end
9      $\vec{wtv_i} = idf(w_i) \times \bigoplus_{k=1}^{K} \vec{wcv_{ik}}$ ;
       /* $\bigoplus$ is concatenation      */
10 end
11 for $n \in (1..N)$ do
12     Initialize document vector $\vec{dv_{D_n}} = \vec{0}$;
13     for word $w_i$ in $D_n$ do
14         $\vec{dv_{D_n}}$ += $\vec{wtv_i}$;
15     end
16     $\vec{SCDV}_{D_n}$ = make-sparse($\vec{dv_{D_n}}$);
       /* as mentioned in sec 3     */
17 end
```

$$\vec{wtv_i} = idf(w_i) \times \bigoplus_{k=1}^{K} \vec{wcv_{ik}}$$

where, $\bigoplus$ is concatenation

3.3 Sparse Document Vectors

After normalizing the vector, we observed that most values in $\vec{dv_{D_n}}$ are very close to zero. Figure 3 verifies this observation. We utilize this fact to make the document vector $\vec{dv_{D_n}}$ sparse by zeroing attribute values whose absolute value is close to a threshold (specified as a parameter), which results in the Sparse Composite Document Vector $\vec{SCDV}_{D_n}$.

In particular, let p be percentage sparsity threshold parameter, a_i the value of the i^{th} attribute of the non-Sparse Composite Document Vector and n represent the n^{th} document in the training set:

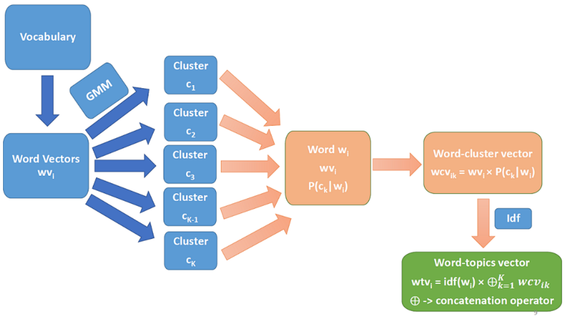


Figure 1: Word-topics vector formation.

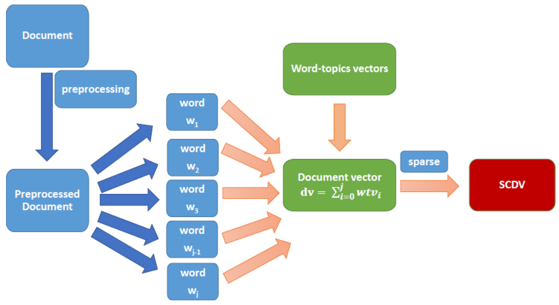


Figure 2: Sparse Composite Document Vector formation.

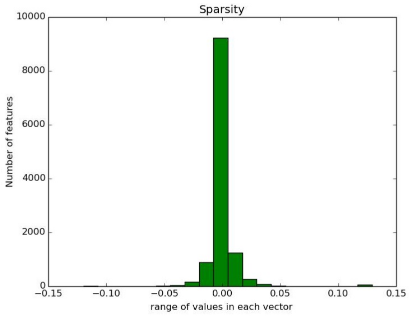


Figure 3: Distribution of attribute feature vector values.

$$a_i = \begin{cases} a_i & \text{if } |a_i| \geq \frac{p}{100} * t \\ 0 & \text{otherwise} \end{cases}$$

$$t = \frac{|a_{min}| + |a_{max}|}{2}$$

$$a_{min} = avg_n(min_i(a_i))$$

$$a_{max} = avg_n(max_i(a_i))$$

Flowcharts depicting the formation of word-topics vector and Sparse Composite Document Vectors are shown in figure 1 and figure 2 respectively. Algorithm 1 describes SCDV in detail.

4 Experiments

We perform multiple experiments to show the effectiveness of SCDV representations for multi-class and multi-label text classification. For all experiments and baselines, we use Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz, 40 working cores, 128GB RAM machine with Linux Ubuntu 14.4. However, we utilize multiple cores only during Word2Vec training and when we run the one-vs-rest classifier for Reuters.

4.1 Baselines

We consider the following baselines: Bag-of-Words (BoW) model (Harris, 1954), Bag of Word Vector (BoWV) (Gupta et al., 2016) model, paragraph vector models (Le and Mikolov, 2014), Topical word embeddings (TWE-1) (Liu et al., 2015b), Neural Tensor Skip-Gram Model (NTSG-1 to NTSG-3) (Liu et al., 2015a), tf-idf weighted average word-vector model (Singh and Mukerjee, 2015) and weighted Bag of Concepts (weight-BoC) (Kim et al., 2017), where we build topic-document vectors by counting the member words in each topic.

We use the best parameter settings as reported in all our baselines to generate their results. We use 200 dimensions for tf-idf weighted word-vector model, 400 for paragraph vector model, 80 topics and 400 dimensional vectors for TWE, NTSG, LTSG and 60 topics and 200 dimensional word vectors for BOWV. We also compare our results with reported results of other topic modeling based document embedding methods like WTM (Fu et al., 2016), $w2v-LDA$ (Nguyen et al., 2015), LDA (Chen and Liu, 2014), $TV+MeanWV$ (Li et al., 2016a), $LTSG$ (Law et al., 2017), $Gaussian-LDA$ (Das et al., 2015), $Topic2Vec$ (Niu et al., 2015), (Moody, 2016) and $MvTM$ (Li et al., 2016b). Implementation of SCDV and related experiments is available here [1].

4.2 Text Classification

We run multi-class experiments on 20NewsGroup dataset [2] and multi-label classification experiments on Reuters-21578 dataset [3]. We use the script[4] for preprocessing the Reuters-21578 dataset. We use LinearSVM for multi-class classi-

[1] https://github.com/dheeraj7596/SCDV
[2] http://qwone.com/∼jason/20Newsgroups/
[3] https://goo.gl/NrOfu
[4] https://gist.github.com/herrfz/7967781

fication and Logistic regression with OneVsRest setting for multi-label classification in baselines and SCDV.

For SCDV, we set the dimension of word-embeddings to 200 and the number of mixture components in GMM to 60. All mixture components share the same spherical co-variance matrix. We learn word vector embedding using Skip-Gram with window size of 10, Negative Sampling (SGNS) of 10 and minimum word frequency as 20. We use 5-fold cross-validation on F1 score to tune parameter C of SVM and the sparsity threshold for SCDV.

4.2.1 Multi-class classification

We evaluate classifier performance using standard metrics like accuracy, macro-averaging precision, recall and F-measure. Table 1 shows a comparison with the current state-of-art (NTSG) document representations on the 20Newsgroup dataset. We observe that SCDV outperforms all other current models by fair margins. We also present the class-wise precision and recall for 20Newsgroup on an almost balanced dataset with SVM over Bag of Words model and the SCDV embeddings in Table 2 and observe that SCDV improves consistently over all classes.

Table 1: Performance on multi-class classification (Values in red show best performance, the SCDV algorithm of this paper)

Model	Acc	Prec	Rec	F-mes
SCDV	84.6	84.6	84.5	84.6
NTSG-1	82.6	82.5	81.9	81.2
NTSG-2	82.5	83.7	82.8	82.4
BoWV	81.6	81.1	81.1	80.9
NTSG-3	81.9	83.0	81.7	81.1
LTSG	82.8	82.4	81.8	81.8
WTM	80.9	80.3	80.3	80.0
w2v-LDA	77.7	77.4	77.2	76.9
TV+MeanWV	72.2	71.8	71.5	71.6
MvTM	72.2	71.8	71.5	71.6
TWE-1	81.5	81.2	80.6	80.6
lda2Vec	81.3	81.4	80.4	80.5
lda	72.2	70.8	70.7	70.0
weight-AvgVec	81.9	81.7	81.9	81.7
BoW	79.7	79.5	79.0	79.0
weight-BOC	71.8	71.3	71.8	71.4
PV-DBoW	75.4	74.9	74.3	74.3
PV-DM	72.4	72.1	71.5	71.5

Table 2: Class-level results on the balanced 20newsgroup dataset.

	BoW		SCDV	
Class Name	**Pre.**	**Rec.**	**Pre.**	**Rec.**
alt.atheism	67.8	72.1	80.2	79.5
comp.graphics	67.1	73.5	75.3	77.4
comp.os.ms-windows.misc	77.1	66.5	78.6	77.2
comp.sys.ibm.pc.hardware	62.8	72.4	75.6	73.5
comp.sys.mac.hardware	77.4	78.2	83.4	85.5
comp.windows.x	83.2	73.2	87.6	78.6
misc.forsale	81.3	88.2	81.4	85.9
rec.autos	80.7	82.8	91.2	90.6
rec.motorcycles	92.3	87.9	95.4	95.7
rec.sport.baseball	89.8	89.2	93.2	94.7
rec.sport.hockey	93.3	93.7	96.3	99.2
sci.crypt	92.2	86.1	92.5	94.7
sci.electronics	70.9	73.3	74.6	74.9
sci.med	79.3	81.3	91.3	88.4
sci.space	90.2	88.3	88.5	93.8
soc.religion.christian	77.3	87.9	83.3	92.3
talk.politics.guns	71.7	85.7	72.7	90.6
talk.politics.mideast	91.7	76.9	96.2	95.4
talk.politics.misc	71.7	56.5	80.9	59.7
talk.religion.misc	63.2	55.4	73.5	57.2

4.2.2 Multi-label classification

We evaluate multi-label classification performance using Precision@K, nDCG@k (Bhatia et al., 2015), Coverage error, Label ranking average precision score (LRAPS)[5] and F1-score. All measures are extensively used for the multi-label classification task. However, F1-score is an appropriate metric for multi-label classification as it considers label biases when train-test splits are random. Table 3 show evaluation results for multi-label text classification on the Reuters-21578 dataset.

4.2.3 Effect of Hyper-Parameters

SCDV has three parameters: the number of clusters, word vector dimension and sparsity threshold parameter. We vary one parameter by keeping the other two constant. Performance on varying all three parameters in shown in Figure 4. We observe that performance improves as we increase the number of clusters and saturates at 60. The performance improves until a word vector dimension of 300 after which it saturates. Similarly, we observe that the performance improves as we increase p till 4 after which it declines. At 4% thresholding, we reduce the storage space by 80% compared to the dense vectors. We observe that SCDV is robust to variations in training Word2Vec

[5] Section 3.3.3.2 of https://goo.gl/4GrR3M

Table 3: Performance on various metrics for multi-label classification for Reuters(Values in red show best performance, the SCDV algorithm of this paper)

Model	Prec@1 nDCG@1	Prec @5	nDCG @5	Coverage Error	LRAPS	F1-Score
SCDV	**94.20**	**36.98**	**49.55**	**6.48**	**93.30**	**81.75**
BoWV	92.90	36.14	48.55	8.16	91.46	79.16
TWE-1	90.91	35.49	47.54	8.16	91.46	79.16
PV-DM	87.54	33.24	44.21	13.15	86.21	70.24
PV-DBoW	88.78	34.51	46.42	11.28	87.43	73.68
AvgVec	89.09	34.73	46.48	9.67	87.28	71.91
tfidf AvgVec	89.33	35.04	46.83	9.42	87.90	71.97

and GMM. The performance metrics reported in Tables 1, 3 are the average values obtained across 5 separate runs of $SCDV$, each run training a different Word2Vec and GMM model with identical hyper-parameters.

4.3 Topic Coherence

We evaluate the topics generated by GMM clustering on 20NewsGroup for quantitative and qualitative analysis. Instead of using perplexity (Chang et al., 2011), which doesn't correlate with semantic coherence and human judgment of individual topics, we used the popular topic coherence (Mimno et al., 2011), (Arora et al., 2013), (Chen and Liu, 2014) measure. A higher topic coherence score indicates a more coherent topic.

We used Bayes rule to compute the $P(w_k|c_i)$ for a given topic c_i and given word w_j and compute the score of the top 10 words for each topic.

$$P(w_k|c_i) = \frac{P(c_i|w_k)P(w_k)}{P(c_i)}$$

where,

$$P(c_i) = \sum_{i=1}^{K} P(c_i|w_k)P(w_k)$$

$$P(w_k) = \frac{\#(w_k)}{\sum_{i=1}^{V} \#(w_i)}$$

Here, $\#(w_k)$ denotes the number of times word w_k appears in the corpus and V represents vocabulary size.

We calculated the topic coherence score for all topics for $SCDV$, LDA and $LTSG$ (Law et al., 2017). Averaging the score of all 80 topics, GMM clustering scores -85.23 compared to -108.72 of LDA and -92.23 of LTSG. Thus, SCDV creates more coherent topics than both LDA and LTSG.

Table 4 shows top 10 words of 3 topics from GMM clustering, LDA model and $LTSG$ model on 20NewsGroup and $SCDV$ shows higher topic coherence. Words are ranked based on their probability distribution in each topic. Our results also support the qualitative results of (Randhawa et al., 2016), (Sridhar, 2015) paper, where k-means, GMM was used respectively over word vectors to find topics.

4.4 Context-Sensitive Learning

In order to demonstrate the effects of soft clustering (GMM) during SCDV formation, we select some words (w$_j$) with multiple senses from 20Newsgroup and their soft cluster assignments to find the dominant clusters. We also select top scoring words (w$_k$) from each cluster (c$_i$) to represent the meaning of that cluster. Table 5 shows polysemic words and their dominant clusters with assignment probabilities. This indicates that using soft clustering to learn word vectors helps combine multiple senses into a single embedding vector. (Arora et al., 2016b) also reported similar results for polysemous words.

4.5 Information Retrieval

(Ai et al., 2016b) used (Mikolov et al., 2013b)'s paragraph vectors to enhance the basic language model based retrieval model. The language model(LM) probabilities are estimated from the corpus and smoothed using a Dirichlet prior (Zhai and Lafferty, 2004). In (Ai et al., 2016b), this language model is then interpolated with the paragraph vector (PV) language model as follows.

$$P(w|d) = (1 - \lambda)P_{LM}(w|d) + \lambda P_{PV}(w|d)$$

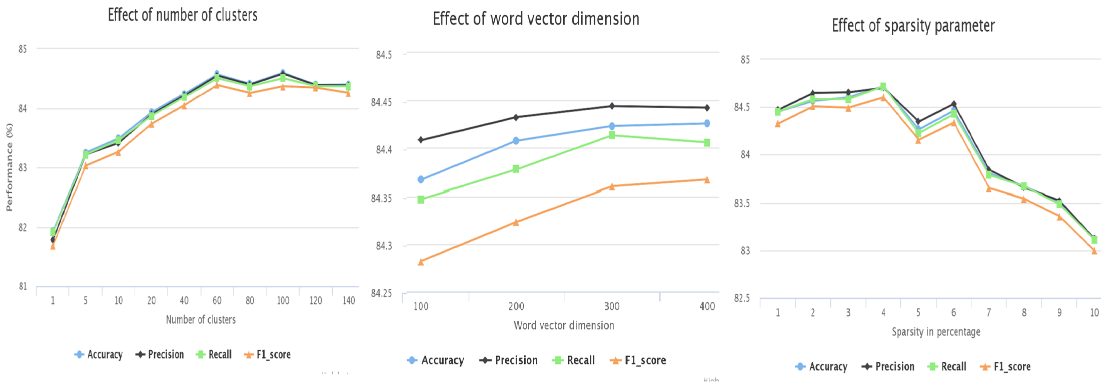


Figure 4: Effect of varying number of clusters (left), varying word vector dimension (center) and varying sparsity parameter (right) on performance for 20NewsGroup with SCDV

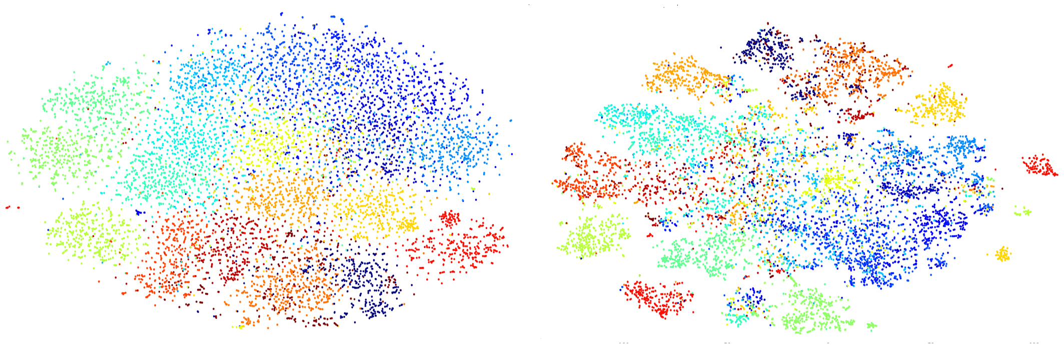


Figure 5: Visualization of paragraph vectors(left) and SCDV(right) using t-SNE

where,

$$P_{PV}(w|d) = \frac{exp(\vec{w}.\vec{d})}{\sum_{i=1}^{V} exp(\vec{w_i}.\vec{d})}$$

and the score for document d and query string Q is given by

$$score(q,d) = \sum_{w \in Q} P(w) P(w|d)$$

where $P(w)$ is obtained from the unigram query model and $score(q,d)$ is used to rank documents. (Ai et al., 2016b) do not directly make use of paragraph vectors for the retrieval task, but improve the document language model. To directly make use of paragraph vectors and make computations more tractable, we directly interpolate the language model query-document score $score(q,d)$ with the similarity score between the normalized query and document vectors to generate $score_{PV}(q,d)$, which is then used to rank documents.

$$score_{PV}(q,d) = (1-\lambda) score(q,d) + \lambda \vec{q}.\vec{d}$$

Directly evaluating the document similarity score with the query paragraph vector rather than collecting similarity scores for individual words in the query helps avoid confusion amongst distinct query topics and makes the interpolation operation faster. In Table 6, we report Mean Average Precision(MAP) values for four datasets, Associated Press 88-89 (topics 51-200), Wall Street Journal (topics 51-200), San Jose Mercury (topics 51-150) and Disks 4 & 5 (topics 301-450) in the TREC collection. We learn λ on a held out set of topics. We observe consistent improvement in MAP for all datasets. We marginally improve the MAP reported by (Ai et al., 2016b) on the Robust04 task. In addition, we also report the improvements in MAP score when Model based relevance feedback (Zhai and Lafferty, 2001) is applied over the initially retrieved results from both models. Again, we notice a consistent improvement in MAP.

5 Analysis and Discussion

SCDV overcomes several challenges encountered while training document vectors, which we had mentioned above.

Table 4: Top words of some topics from GMM and LDA on 20NewsGroup for K = 80. Higher score represent better coherent topics.

Topic Image			Topic Health			Topic Mail		
GMM	**LTSG**	**LDA**	**GMM**	**LTSG**	**LDA**	**GMM**	**LTSG**	**LDA**
file	image	image	heath	stimulation	doctor	ftp	anonymous	list
bit	jpeg	file	study	diseases	disease	mail	faq	mail
image	gif	color	medical	disease	coupons	internet	send	information
files	format	gif	drug	toxin	treatment	phone	ftp	internet
color	file	jpeg	test	toxic	pain	email	mailing	send
format	files	file	drugs	newsletter	medical	send	server	posting
images	convert	format	studies	staff	day	opinions	mail	email
jpeg	color	bit	disease	volume	microorganism	fax	alt	group
gif	formats	images	education	heaths	medicine	address	archive	news
program	images	quality	age	aids	body	box	email	anonymous
-67.16	-75.66	-88.79	-66.91	-96.98	-100.39	-77.47	-78.23	-95.47

Table 5: Words with multiple senses assigned to multiple clusters with significant probabilities

Word	Cluster Words	P($c_i\|w_j$)
subject:1	physics, chemistry, math, science	0.27
subject:2	mail, letter, email, gmail	0.72
interest:1	information, enthusiasm, question	0.65
interest:2	bank, market, finance, investment	0.32
break:1	vacation, holiday, trip, spring	0.52
break:2	encryption, cipher, security, privacy	0.22
break:3	if, elseif, endif, loop, continue	0.23
unit:1	calculation, distance, mass, length	0.25
unit:2	electronics, KWH, digital, signal	0.69

1. Clustering word-embeddings to discover topics improves performance of classification as Figure 4 (left) indicates, while also generating coherent clusters of words (Table 4). Figure 5 shows that clustering gives more discriminative representations of documents than paragraph vectors do since it uses K × d dimensions while paragraph vectors embed documents and words in the same space. This enables SCDV to represent complex documents. Fuzzy clustering allows words to belong to multiple topics, thereby recognizing polysemic words, as Table 5 indicates. Thus it mimics the word-context interaction in NTSG and LTSG.

2. Semantically different words are assigned to different topics. Moreover, a single document can contain words from multiple different topics. Instead of a weighted averaging of word embeddings to form document vectors, as in most previous work, concatenating word embeddings for each topic (cluster) avoids merging of semantically different topics.

3. It is well-known that in higher dimensions, structural regularizers such as sparsity help overcome the curse of dimensionality (Wainwright, 2014).Figure 3 demonstrates this, since majority of the features are close to zero. Sparsity also enables linear SVM to scale to large dimensions. On 20NewsGroups, BoWV model takes up 1.1 GB while SCDV takes up only 236MB(80% decrease). Since GMM assigns a non-zero probability to every topic in the word embedding, noise can accumulate when document vectors are created and tip the scales in favor of an unrelated topic. Sparsity helps to reduce this by zeroing out very small values of probability.

4. SCDV uses Gaussian Mixture Model (GMM) while TWE, $NTSG$ and $LTSG$ use LDA for finding semantic topics respectively. GMM time complexity is $\mathcal{O}(VNT^2)$ while that of LDA is $\mathcal{O}(V^2NT)$. Here, V = Vocabulary size, N = number of documents and T = number of topics. Since number of topics T $<$ vocabulary size V, GMM is faster. Empirically, compared to TWE, $SCDV$ reduces document vector formation, training and prediction time significantly. Table 7 shows training and prediction times for BoWV, SCDV and TWE models.

6 Conclusion

In this paper, we propose a document feature formation technique for topic-based document representation. SCDV outperforms state-of-the-art models in multi-class and multi-label classification tasks. SCDV introduces sparsity in document vectors to handle high dimensionality. Table 7 in-

Table 6: Mean average precision (MAP) for IR on four IR datasets

Dataset	LM	LM+SCDV	MB	MB + SCDV
AP	0.2742	**0.2856**	0.3283	**0.3395**
SJM	0.2052	**0.2105**	0.2341	**0.2409**
WSJ	0.2618	**0.2705**	0.3027	**0.3126**
Robust04	0.2516	**0.2684**	0.2819	**0.2933**

Table 7: Time Comparison (20NewsGroup) (Values in red show least time, the SCDV algorithm of this paper)

Time (sec)	BoWV	TWE-1	SCDV
DocVec Formation	1250	700	**160**
Total Training	1320	740	**200**
Total Prediction	780	120	**25**

dicates that SCDV shows considerable improvements in feature formation, training and prediction times for the 20NewsGroups dataset. We show that fuzzy GMM clustering on word-vectors lead to more coherent topic than LDA and can also be used to detect Polysemic words. SCDV embeddings also provide a robust estimation of the query and document language models, thus improving the MAP of language model based retrieval systems. In conclusion, SCDV is simple, efficient and creates a more accurate semantic representation of documents.

Acknowledgments

The authors want to thank Nagarajan Natarajan (Post-Doc, Microsoft Research, India), Praneeth Netrapalli (Researcher, Microsoft Research, India), Raghavendra Udupa (Researcher, Microsoft Research, India), Prateek Jain (Researcher, Microsoft Research, India) for encouraging and valuable feedback .

References

Qingyao Ai, Liu Yang, Jiafeng Guo, and W Bruce Croft. 2016a. Analysis of the paragraph vector model for information retrieval. In *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval*, pages 133–142. ACM.

Qingyao Ai, Liu Yang, Jiafeng Guo, and W Bruce Croft. 2016b. Improving language estimation with the paragraph vector model for ad-hoc retrieval. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 869–872. ACM.

Sanjeev Arora, Rong Ge, Yonatan Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. 2013. A practical algorithm for topic modeling with provable guarantees. In *International Conference on Machine Learning*, pages 280–288.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016a. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016b. Linear algebraic structure of word senses, with applications to polysemy. *arXiv preprint arXiv:1601.03764*.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*, pages 730–738.

David M. Blei, David M. Ng, Michael I. Jordan, and John Lafferty. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003.

Jonathan Chang, Jordan L Boyd-Graber, Sean Gerrish, Chong Wang, and David M Blei. 2011. Reading tea leaves: How humans interpret topic models. pages 262–272.

Zhiyuan Chen and Bing Liu. 2014. Topic modeling using topics from many domains, lifelong learning and big data. In *International Conference on Machine Learning*, pages 703–711.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian lda for topic models with word embeddings. In *Proceedings of The 53th Annual Meeting of the Association of Computational Linguistic (ACL)*, pages 795–804. Association of Computational Linguistic.

Xianghua Fu, Ting Wang, Jing Li, Chong Yu, and Wangwang Liu. 2016. Improving distributed word representation and topic model by word-topic mixture model. In *Proceedings of The 8th Asian Conference on Machine Learning*, pages 190–205.

Vivek Gupta, Harish Karnick, Ashendra Bansal, and Pradhuman Jhala. 2016. Product classification in e-commerce using distributional semantics. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 536–546.

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

Han Kyul Kim, Hyunjoong Kim, and Sungzoon Cho. 2017. Bag-of-concepts: Comprehending document representation through clustering words in distributed representation. *Neurocomputing*.

Jarvan Law, Hankz Hankui Zhuo, Junhua He, and Erhu Rong. 2017. Ltsg: Latent topical skip-gram for mutually learning topic model and vector representations. *arXiv preprint arXiv:1702.07117*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Shaohua Li, Tat-Seng Chua, Jun Zhu, and Chunyan Miao. 2016a. Generative topic embedding: a continuous representation of documents. In *Proceedings of The 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association of Computational Linguistic.

Ximing Li, Jinjin Chi, Changchun Li, Jihong Ouyang, and Bo Fu. 2016b. Integrating topic modeling with word embeddings by mixtures of vmfs. *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 151–160.

Wand Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word-vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. North American Association for Computational Linguistics.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2015a. Learning context-sensitive word embeddings with neural tensor skip-gram model. In *Proceedings of IJCAI 2014, the 24th International Joint Conference on Artificial Intelligence*, pages 1284–1290.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015b. Topical word embeddings. In *Proceedings of AAAI 2015, the 29th Twenty-Ninth Association for the Advancement of Artificial Intelligence*, pages 2418–2424.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 262–272. Association for Computational Linguistics.

Christopher E Moody. 2016. Mixing dirichlet topic models and word embeddings to make lda2vec. *arXiv preprint arXiv:1605.02019*.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics*, 3:299–313.

Liqiang Niu, Xinyu Dai, Jianbing Zhang, and Jiajun Chen. 2015. Topic2vec: learning distributed representations of topics. In *2015 International Conference on Asian Language Processing (IALP)*, pages 193–196. IEEE.

Ramandeep S Randhawa, Parag Jain, and Gagan Madan. 2016. Topic modeling using distributed word embeddings. *arXiv preprint arXiv:1603.04747*.

Douglas Reynolds. 2015. Gaussian mixture models. *Encyclopedia of biometrics*, pages 827–832.

Dwaipayan Roy, Debasis Ganguly, Mandar Mitra, and Gareth JF Jones. 2016. Representing documents and queries as sets of word embedded vectors for information retrieval. *arXiv preprint arXiv:1606.07869*.

Pranjal Singh and Amitabha Mukerjee. 2015. Words are not equal: Graded weighting model for building composite document vectors. In *Proceedings of the twelfth International Conference on Natural Language Processing (ICON-2015)*.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642.

Vivek Kumar Rangarajan Sridhar. 2015. Unsupervised topic modeling for short texts using distributed representations of words. In *VS@ HLT-NAACL*, pages 192–200.

Martin J Wainwright. 2014. Structured regularizers for high-dimensional problems: Statistical and computational issues. *Annual Review of Statistics and Its Application*, 1:233–253.

Chengxiang Zhai and John Lafferty. 2001. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 403–410. ACM.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214.

Supervised Learning of Universal Sentence Representations from Natural Language Inference Data

Alexis Conneau
Facebook AI Research
aconneau@fb.com

Douwe Kiela
Facebook AI Research
dkiela@fb.com

Holger Schwenk
Facebook AI Research
schwenk@fb.com

Loïc Barrault
LIUM, Université Le Mans
loic.barrault@univ-lemans.fr

Antoine Bordes
Facebook AI Research
abordes@fb.com

Abstract

Many modern NLP systems rely on word embeddings, previously trained in an unsupervised manner on large corpora, as base features. Efforts to obtain embeddings for larger chunks of text, such as sentences, have however not been so successful. Several attempts at learning unsupervised representations of sentences have not reached satisfactory enough performance to be widely adopted. In this paper, we show how universal sentence representations trained using the supervised data of the Stanford Natural Language Inference datasets can consistently outperform unsupervised methods like SkipThought vectors (Kiros et al., 2015) on a wide range of transfer tasks. Much like how computer vision uses ImageNet to obtain features, which can then be transferred to other tasks, our work tends to indicate the suitability of natural language inference for transfer learning to other NLP tasks. Our encoder is publicly available[1].

1 Introduction

Distributed representations of words (or word embeddings) (Bengio et al., 2003; Collobert et al., 2011; Mikolov et al., 2013; Pennington et al., 2014) have shown to provide useful features for various tasks in natural language processing and computer vision. While there seems to be a consensus concerning the usefulness of word embeddings and how to learn them, this is not yet clear with regard to representations that carry the meaning of a full sentence. That is, how to capture the relationships among multiple words and phrases in a single vector remains an question to be solved.

In this paper, we study the task of learning universal representations of sentences, i.e., a sentence encoder model that is trained on a large corpus and subsequently transferred to other tasks. Two questions need to be solved in order to build such an encoder, namely: what is the preferable neural network architecture; and how and on what task should such a network be trained. Following existing work on learning word embeddings, most current approaches consider learning sentence encoders in an unsupervised manner like SkipThought (Kiros et al., 2015) or FastSent (Hill et al., 2016). Here, we investigate whether supervised learning can be leveraged instead, taking inspiration from previous results in computer vision, where many models are pretrained on the ImageNet (Deng et al., 2009) before being transferred. We compare sentence embeddings trained on various supervised tasks, and show that sentence embeddings generated from models trained on a natural language inference (NLI) task reach the best results in terms of transfer accuracy. We hypothesize that the suitability of NLI as a training task is caused by the fact that it is a high-level understanding task that involves reasoning about the semantic relationships within sentences.

Unlike in computer vision, where convolutional neural networks are predominant, there are multiple ways to encode a sentence using neural networks. Hence, we investigate the impact of the sentence encoding architecture on representational transferability, and compare convolutional, recurrent and even simpler word composition schemes. Our experiments show that an encoder based on a bi-directional LSTM architecture with max pooling, trained on the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015), yields state-of-the-art sentence embeddings com-

[1] https://www.github.com/facebookresearch/InferSent

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 670–680
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

pared to all existing alternative unsupervised approaches like SkipThought or FastSent, while being much faster to train. We establish this finding on a broad and diverse set of transfer tasks that measures the ability of sentence representations to capture general and useful information.

2 Related work

Transfer learning using supervised features has been successful in several computer vision applications (Razavian et al., 2014). Striking examples include face recognition (Taigman et al., 2014) and visual question answering (Antol et al., 2015), where image features trained on ImageNet (Deng et al., 2009) and word embeddings trained on large unsupervised corpora are combined.

In contrast, most approaches for sentence representation learning are unsupervised, arguably because the NLP community has not yet found the best supervised task for embedding the semantics of a whole sentence. Another reason is that neural networks are very good at capturing the biases of the task on which they are trained, but can easily forget the overall information or semantics of the input data by specializing too much on these biases. Learning models on large unsupervised task makes it harder for the model to specialize. Littwin and Wolf (2016) showed that co-adaptation of encoders and classifiers, when trained end-to-end, can negatively impact the generalization power of image features generated by an encoder. They propose a loss that incorporates multiple orthogonal classifiers to counteract this effect.

Recent work on generating sentence embeddings range from models that compose word embeddings (Le and Mikolov, 2014; Arora et al., 2017; Wieting et al., 2016b) to more complex neural network architectures. SkipThought vectors (Kiros et al., 2015) propose an objective function that adapts the skip-gram model for words (Mikolov et al., 2013) to the sentence level. By encoding a sentence to predict the sentences around it, and using the features in a linear model, they were able to demonstrate good performance on 8 transfer tasks. They further obtained better results using layer-norm regularization of their model in (Ba et al., 2016). Hill et al. (2016) showed that the task on which sentence embeddings are trained significantly impacts their quality. In addition to unsupervised methods, they included supervised training in their comparison—namely, on machine translation data (using the WMT'14 English/French and English/German pairs), dictionary definitions and image captioning data from the COCO dataset (Lin et al., 2014). These models obtained significantly lower results compared to the unsupervised Skip-Thought approach.

Recent work has explored training sentence encoders on the SNLI corpus and applying them on the SICK corpus (Marelli et al., 2014), either using multi-task learning or pretraining (Mou et al., 2016; Bowman et al., 2015). The results were inconclusive and did not reach the same level as simpler approaches that directly learn a classifier on top of unsupervised sentence embeddings instead (Arora et al., 2017). To our knowledge, this work is the first attempt to fully exploit the SNLI corpus for building generic sentence encoders. As we show in our experiments, we are able to consistently outperform unsupervised approaches, even if our models are trained on much less (but human-annotated) data.

3 Approach

This work combines two research directions, which we describe in what follows. First, we explain how the NLI task can be used to train universal sentence encoding models using the SNLI task. We subsequently describe the architectures that we investigated for the sentence encoder, which, in our opinion, covers a suitable range of sentence encoders currently in use. Specifically, we examine standard recurrent models such as LSTMs and GRUs, for which we investigate mean and max-pooling over the hidden representations; a self-attentive network that incorporates different views of the sentence; and a hierarchical convolutional network that can be seen as a tree-based method that blends different levels of abstraction.

3.1 The Natural Language Inference task

The SNLI dataset consists of 570k human-generated English sentence pairs, manually labeled with one of three categories: entailment, contradiction and neutral. It captures natural language inference, also known in previous incarnations as Recognizing Textual Entailment (RTE), and constitutes one of the largest high-quality labeled resources explicitly constructed in order to require understanding sentence semantics. We hypothesize that the semantic nature of NLI makes it a good candidate for learning universal sentence embeddings in a supervised way. That is, we aim to demonstrate that sentence encoders trained on natural language inference are able to learn sentence representations that capture universally useful features.

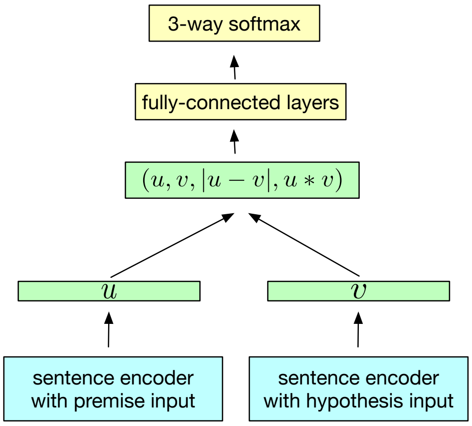

3-way softmax
fully-connected layers
$(u, v, |u - v|, u * v)$
u
v
sentence encoder with premise input
sentence encoder with hypothesis input

Figure 1: **Generic NLI training scheme.**

Models can be trained on SNLI in two different ways: (i) sentence encoding-based models that explicitly separate the encoding of the individual sentences and (ii) joint methods that allow to use encoding of both sentences (to use cross-features or attention from one sentence to the other).

Since our goal is to train a generic sentence encoder, we adopt the first setting. As illustrated in Figure 1, a typical architecture of this kind uses a shared sentence encoder that outputs a representation for the premise u and the hypothesis v. Once the sentence vectors are generated, 3 matching methods are applied to extract relations between u and v : (i) concatenation of the two representations (u, v); (ii) element-wise product $u * v$; and (iii) absolute element-wise difference $|u - v|$. The resulting vector, which captures information from both the premise and the hypothesis, is fed into a 3-class classifier consisting of multiple fully-connected layers culminating in a softmax layer.

3.2 Sentence encoder architectures

A wide variety of neural networks for encoding sentences into fixed-size representations exists, and it is not yet clear which one best captures generically useful information. We compare 7 different architectures: standard recurrent encoders with either Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU), concatenation of last hidden states of forward and backward GRU, Bi-directional LSTMs (BiLSTM) with either mean or max pooling, self-attentive network and hierarchical convolutional networks.

3.2.1 LSTM and GRU

Our first, and simplest, encoders apply recurrent neural networks using either LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014) modules, as in sequence to sequence encoders (Sutskever et al., 2014). For a sequence of T words $(w_1, \ldots, w_T)$, the network computes a set of T hidden representations $h_1, \ldots, h_T$, with $h_t = \overrightarrow{\text{LSTM}}(w_1, \ldots, w_T)$ (or using GRU units instead). A sentence is represented by the last hidden vector, h_T.

We also consider a model BiGRU-last that concatenates the last hidden state of a forward GRU, and the last hidden state of a backward GRU to have the same architecture as for SkipThought vectors.

3.2.2 BiLSTM with mean/max pooling

For a sequence of T words $\{w_t\}_{t=1,\ldots,T}$, a bidirectional LSTM computes a set of T vectors $\{h_t\}_t$. For $t \in [1, \ldots, T]$, h_t, is the concatenation of a forward LSTM and a backward LSTM that read the sentences in two opposite directions:

$$\begin{aligned}\overrightarrow{h_t} &= \overrightarrow{\text{LSTM}}_t(w_1, \ldots, w_T)\\ \overleftarrow{h_t} &= \overleftarrow{\text{LSTM}}_t(w_1, \ldots, w_T)\\ h_t &= [\overrightarrow{h_t}, \overleftarrow{h_t}]\end{aligned}$$

We experiment with two ways of combining the varying number of $\{h_t\}_t$ to form a fixed-size vector, either by selecting the maximum value over each dimension of the hidden units (max pooling) (Collobert and Weston, 2008) or by considering the average of the representations (mean pooling).

3.2.3 Self-attentive network

The self-attentive sentence encoder (Liu et al., 2016; Lin et al., 2017) uses an attention mechanism over the hidden states of a BiLSTM to generate a representation u of an input sentence. The attention mechanism is defined as :

$$\begin{aligned}\bar{h}_i &= \tanh(Wh_i + b_w)\\ \alpha_i &= \frac{e^{\bar{h}_i^T u_w}}{\sum_i e^{\bar{h}_i^T u_w}}\\ u &= \sum_t \alpha_i h_i\end{aligned}$$

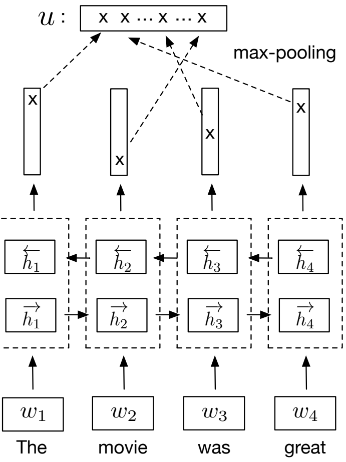

u: x x ... x ... x
max-pooling
x
x
x
x
$\overleftarrow{h_1}$ $\overleftarrow{h_2}$ $\overleftarrow{h_3}$ $\overleftarrow{h_4}$
$\overrightarrow{h_1}$ $\overrightarrow{h_2}$ $\overrightarrow{h_3}$ $\overrightarrow{h_4}$
w_1 w_2 w_3 w_4
The
movie
was
great

Figure 2: **Bi-LSTM max-pooling network.**

where $\{h_1, \ldots, h_T\}$ are the output hidden vectors of a BiLSTM. These are fed to an affine transformation (W, b_w) which outputs a set of keys $(\bar{h}_1, \ldots, \bar{h}_T)$. The $\{\alpha_i\}$ represent the score of similarity between the keys and a learned context query vector u_w. These weights are used to produce the final representation u, which is a weighted linear combination of the hidden vectors.

Following Lin et al. (2017) we use a self-attentive network with multiple views of the input sentence, so that the model can learn which part of the sentence is important for the given task. Concretely, we have 4 context vectors $u_w^1, u_w^2, u_w^3, u_w^4$ which generate 4 representations that are then concatenated to obtain the sentence representation u. Figure 3 illustrates this architecture.

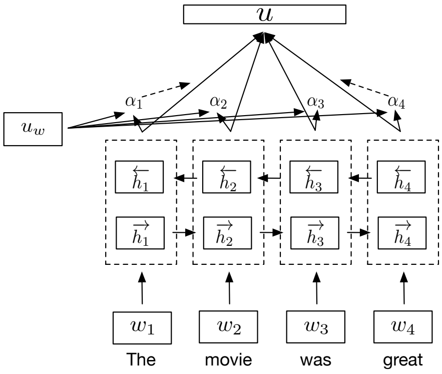

u
α_1
α_2
α_3
α_4
u_w
$\overleftarrow{h_1}$ $\overleftarrow{h_2}$ $\overleftarrow{h_3}$ $\overleftarrow{h_4}$
$\overrightarrow{h_1}$ $\overrightarrow{h_2}$ $\overrightarrow{h_3}$ $\overrightarrow{h_4}$
w_1 w_2 w_3 w_4
The
movie
was
great

Figure 3: **Inner Attention network architecture.**

3.2.4 Hierarchical ConvNet

One of the currently best performing models on classification tasks is a convolutional architecture termed *AdaSent* (Zhao et al., 2015), which concatenates different representations of the sentences at different level of abstractions. Inspired by this architecture, we introduce a faster version consisting of 4 convolutional layers. At every layer, a representation u_i is computed by a max-pooling operation over the feature maps (see Figure 4).

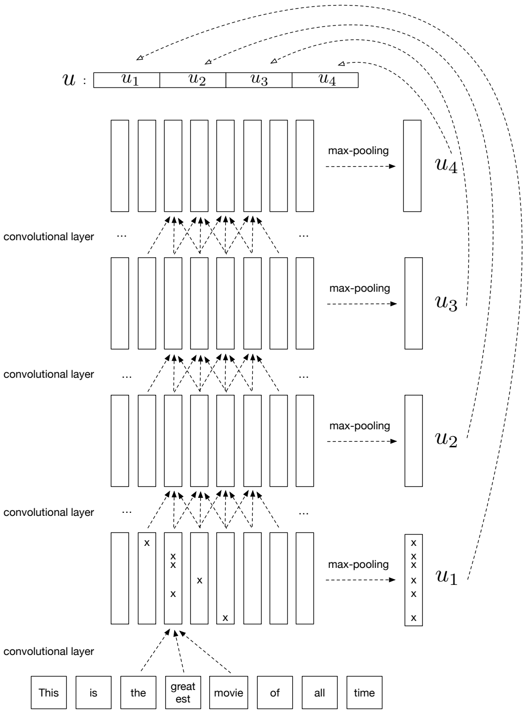


Figure 4: **Hierarchical ConvNet architecture.**

The final representation $u = [u_1, u_2, u_3, u_4]$ concatenates representations at different levels of the input sentence. The model thus captures hierarchical abstractions of an input sentence in a fixed-size representation.

3.3 Training details

For all our models trained on SNLI, we use SGD with a learning rate of 0.1 and a weight decay of 0.99. At each epoch, we divide the learning rate by 5 if the dev accuracy decreases. We use mini-batches of size 64 and training is stopped when the learning rate goes under the threshold of 10^{-5}. For the classifier, we use a multi-layer perceptron with 1 hidden-layer of 512 hidden units. We use open-source GloVe vectors trained on Common Crawl 840B[2] with 300 dimensions as fixed word embeddings.

4 Evaluation of sentence representations

Our aim is to obtain general-purpose sentence embeddings that capture generic information that is

[2] https://nlp.stanford.edu/projects/glove/

name	N	task	C	examples
MR	11k	sentiment (movies)	2	”Too slow for a younger crowd , too shallow for an older one.” (neg)
CR	4k	product reviews	2	”We tried it out christmas night and it worked great .” (pos)
SUBJ	10k	subjectivity/objectivity	2	”A movie that doesn’t aim too high , but doesn’t need to.” (subj)
MPQA	11k	opinion polarity	2	”don’t want”; ”would like to tell”; (neg, pos)
TREC	6k	question-type	6	”What are the twin cities ?” (LOC:city)
SST	70k	sentiment (movies)	2	”Audrey Tautou has a knack for picking roles that magnify her [..]” (pos)

Table 1: **Classification tasks**. C is the number of class and N is the number of samples.

useful for a broad set of tasks. To evaluate the quality of these representations, we use them as features in 12 transfer tasks. We present our sentence-embedding evaluation procedure in this section. We constructed a sentence evaluation tool[3] to automate evaluation on all the tasks mentioned in this paper. The tool uses Adam (Kingma and Ba, 2014) to fit a logistic regression classifier, with batch size 64.

Binary and multi-class classification We use a set of binary classification tasks (see Table 1) that covers various types of sentence classification, including sentiment analysis (MR, SST), question-type (TREC), product reviews (CR), subjectivity/objectivity (SUBJ) and opinion polarity (MPQA). We generate sentence vectors and train a logistic regression on top. A linear classifier requires fewer parameters than an MLP and is thus suitable for small datasets, where transfer learning is especially well-suited. We tune the L2 penalty of the logistic regression with grid-search on the validation set.

Entailment and semantic relatedness We also evaluate on the SICK dataset for both entailment (SICK-E) and semantic relatedness (SICK-R). We use the same matching methods as in SNLI and learn a Logistic Regression on top of the joint representation. For semantic relatedness evaluation, we follow the approach of (Tai et al., 2015) and learn to predict the probability distribution of relatedness scores. We report Pearson correlation.

STS14 - Semantic Textual Similarity While semantic relatedness is supervised in the case of SICK-R, we also evaluate our embeddings on the 6 unsupervised SemEval tasks of STS14 (Agirre et al., 2014). This dataset includes subsets of news articles, forum discussions, image descriptions and headlines from news articles containing pairs of sentences (lower-cased), labeled with a similarity score between 0 and 5. These tasks evaluate how the cosine distance between two sentences correlate with a human-labeled similarity score through Pearson and Spearman correlations.

Paraphrase detection The Microsoft Research Paraphrase Corpus is composed of pairs of sentences which have been extracted from news sources on the Web. Sentence pairs have been human-annotated according to whether they capture a paraphrase/semantic equivalence relationship. We use the same approach as with SICK-E, except that our classifier has only 2 classes.

Caption-Image retrieval The caption-image retrieval task evaluates joint image and language feature models (Hodosh et al., 2013; Lin et al., 2014). The goal is either to rank a large collection of images by their relevance with respect to a given query caption (Image Retrieval), or ranking captions by their relevance for a given query image (Caption Retrieval). We use a pairwise ranking-loss $\mathcal{L}_{\text{cir}}(x, y)$:

$$\sum_y \sum_k \max(0, \alpha - s(Vy, Ux) + s(Vy, Ux_k)) + \\ \sum_x \sum_{k'} \max(0, \alpha - s(Ux, Vy) + s(Ux, Vy_{k'}))$$

where (x, y) consists of an image y with one of its associated captions x, $(y_k)_k$ and $(y_{k'})_{k'}$ are negative examples of the ranking loss, α is the margin and s corresponds to the cosine similarity. U and V are learned linear transformations that project the caption x and the image y to the same embedding space. We use a margin $\alpha = 0.2$ and 30 contrastive terms. We use the same splits as in (Karpathy and Fei-Fei, 2015), i.e., we use 113k images from the COCO dataset (each containing 5 captions) for training, 5k images for validation and 5k images for test. For evaluation, we split the 5k images in 5 random sets of 1k images on which we compute Recall@K, with K $\in \{1, 5, 10\}$ and

[3] https://www.github.com/facebookresearch/SentEval

name	task	N	premise	hypothesis	label
SNLI	NLI	560k	”Two women are embracing while holding to go packages.”	”Two woman are holding packages.”	entailment
SICK-E	NLI	10k	A man is typing on a machine used for stenography	The man isn’t operating a stenograph	contradiction
SICK-R	STS	10k	”A man is singing a song and playing the guitar”	”A man is opening a package that contains headphones”	1.6
STS14	STS	4.5k	”Liquid ammonia leak kills 15 in Shanghai”	”Liquid ammonia leak kills at least 15 in Shanghai”	4.6

Table 2: **Natural Language Inference and Semantic Textual Similarity tasks**. NLI labels are contradiction, neutral and entailment. STS labels are scores between 0 and 5.

median (Med r) over the 5 splits. For fair comparison, we also report SkipThought results in our setting, using 2048-dimensional pretrained ResNet-101 (He et al., 2016) with 113k training images.

Model		NLI		Transfer	
	dim	dev	test	micro	macro
LSTM	2048	81.9	80.7	79.5	78.6
GRU	4096	82.4	81.8	81.7	80.9
BiGRU-last	4096	81.3	80.9	82.9	81.7
BiLSTM-Mean	4096	79.0	78.2	83.1	81.7
Inner-attention	4096	82.3	82.5	82.1	81.0
HConvNet	4096	83.7	83.4	82.0	80.9
BiLSTM-Max	4096	**85.0**	**84.5**	**85.2**	**83.7**

Table 3: **Performance of sentence encoder architectures** on SNLI and (aggregated) transfer tasks. Dimensions of embeddings were selected according to best aggregated scores (see Figure 5).

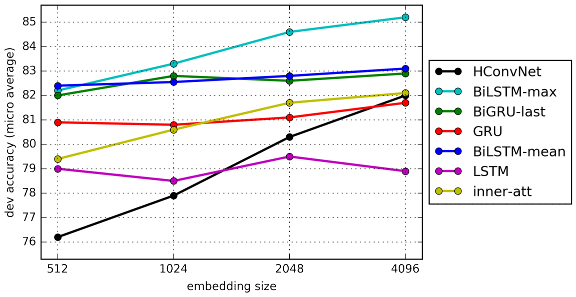


Figure 5: **Transfer performance w.r.t. embedding size** using the micro aggregation method.

5 Empirical results

In this section, we refer to ”micro” and ”macro” averages of development set (dev) results on transfer tasks whose metrics is accuracy: we compute a ”macro” aggregated score that corresponds to the classical average of dev accuracies, and the ”micro” score that is a sum of the dev accuracies, weighted by the number of dev samples.

5.1 Architecture impact

Model We observe in Table 3 that different models trained on the same NLI corpus lead to different transfer tasks results. The BiLSTM-4096 with the max-pooling operation performs best on both SNLI and transfer tasks. Looking at the micro and macro averages, we see that it performs significantly better than the other models LSTM, GRU, BiGRU-last, BiLSTM-Mean, inner-attention and the hierarchical-ConvNet.

Table 3 also shows that better performance on the training task does not necessarily translate in better results on the transfer tasks like when comparing inner-attention and BiLSTM-Mean for instance.

We hypothesize that some models are likely to over-specialize and adapt too well to the biases of a dataset without capturing general-purpose information of the input sentence. For example, the inner-attention model has the ability to focus only on certain parts of a sentence that are useful for the SNLI task, but not necessarily for the transfer tasks. On the other hand, BiLSTM-Mean does not make sharp choices on which part of the sentence is more important than others. The difference between the results seems to come from the different abilities of the models to incorporate general information while not focusing too much on specific features useful for the task at hand.

For a given model, the transfer quality is also sensitive to the optimization algorithm: when training with Adam instead of SGD, we observed that the BiLSTM-max converged faster on SNLI (5 epochs instead of 10), but obtained worse results on the transfer tasks, most likely because of the model and classifier’s increased capability to over-specialize on the training task.

Embedding size Figure 5 compares the overall performance of different architectures, showing the evolution of micro averaged performance with

Model	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	SICK-R	SICK-E	STS14
Unsupervised representation training (unordered sentences)										
Unigram-TFIDF	73.7	**79.2**	90.3	82.4	-	85.0	73.6/81.7	-	-	.58/.57
ParagraphVec (DBOW)	60.2	66.9	76.3	70.7	-	59.4	72.9/81.1	-	-	.42/.43
SDAE	74.6	78.0	90.8	86.9	-	78.4	**73.7**/80.7	-	-	.37/.38
SIF (GloVe + WR)	-	-	-	-	82.2	-	-	-	**84.6**	**.69**/ -
word2vec BOW†	77.7	79.8	90.9	88.3	79.7	83.6	72.5/81.4	0.803	78.7	.65/.64
fastText BOW†	76.5	78.9	91.6	87.4	78.8	81.8	72.4/81.2	0.800	77.9	.63/.62
GloVe BOW†	**78.7**	78.5	**91.6**	87.6	79.8	83.6	72.1/80.9	0.800	78.6	.54/.56
GloVe Positional Encoding†	78.3	77.4	91.1	87.1	80.6	83.3	72.5/81.2	0.799	77.9	.51/.54
BiLSTM-Max (untrained)†	77.5	**81.3**	89.6	**88.7**	80.7	**85.8**	73.2/81.6	**0.860**	83.4	.39/.48
Unsupervised representation training (ordered sentences)										
FastSent	70.8	78.4	88.7	80.6	-	76.8	72.2/80.3	-	-	**.63/.64**
FastSent+AE	71.8	76.7	88.8	81.5	-	80.4	71.2/79.1	-	-	.62/.62
SkipThought	76.5	80.1	93.6	87.1	82.0	**92.2**	**73.0/82.0**	**0.858**	82.3	.29/.35
SkipThought-LN	**79.4**	**83.1**	**93.7**	**89.3**	82.9	88.4	-	**0.858**	79.5	.44/.45
Supervised representation training										
CaptionRep (bow)	61.9	69.3	77.4	70.8	-	72.2	-	-	-	.46/.42
DictRep (bow)	76.7	78.7	90.7	87.2	-	81.0	68.4/76.8	-	-	**.67/.70**
NMT En-to-Fr	64.7	70.1	84.9	81.5	-	82.8	-	-		.43/.42
Paragram-phrase	-	-	-	-	79.7	-	-	0.849	83.1	**.71**/ -
BiLSTM-Max (on SST)†	(*)	83.7	90.2	89.5	(*)	86.0	72.7/80.9	0.863	83.1	.55/.54
BiLSTM-Max (on SNLI)†	79.9	84.6	92.1	**89.8**	83.3	**88.7**	75.1/82.3	**0.885**	**86.3**	.68/.65
BiLSTM-Max (on AllNLI)†	**81.1**	**86.3**	**92.4**	**90.2**	**84.6**	88.2	**76.2/83.1**	**0.884**	**86.3**	**.70/.67**
Supervised methods (directly trained for each task – no transfer)										
Naive Bayes - SVM	79.4	81.8	93.2	86.3	83.1	-	-	-	-	-
AdaSent	83.1	86.3	95.5	93.3	-	92.4	-	-	-	-
TF-KLD	-	-	-	-	-	-	80.4/85.9	-	-	-
Illinois-LH	-	-	-	-	-	-	-	-	84.5	-
Dependency Tree-LSTM	-	-	-	-	-	-	-	0.868	-	-

Table 4: **Transfer test results for various architectures trained in different ways**. Underlined are best results for transfer learning approaches, in bold are best results among the models trained in the same way. † indicates methods that we trained, other transfer models have been extracted from (Hill et al., 2016). For best published supervised methods (no transfer), we consider AdaSent (Zhao et al., 2015), TF-KLD (Ji and Eisenstein, 2013), Tree-LSTM (Tai et al., 2015) and Illinois-LH system (Lai and Hockenmaier, 2014). (*) Our model trained on SST obtained 83.4 for MR and 86.0 for SST (MR and SST come from the same source), which we do not put in the tables for fair comparison with transfer methods.

regard to the embedding size.

Since it is easier to linearly separate in high dimension, especially with logistic regression, it is not surprising that increased embedding sizes lead to increased performance for almost all models. However, this is particularly true for some models (BiLSTM-Max, HConvNet, inner-att), which demonstrate unequal abilities to incorporate more information as the size grows. We hypothesize that such networks are able to incorporate information that is not directly relevant to the objective task (results on SNLI are relatively stable with regard to embedding size) but that can nevertheless be useful as features for transfer tasks.

5.2 Task transfer

We report in Table 4 transfer tasks results for different architectures trained in different ways. We group models by the nature of the data on which they were trained. The first group corresponds to models trained with unsupervised unordered sentences. This includes bag-of-words models such as word2vec-SkipGram, the Unigram-TFIDF model, the Paragraph Vector model (Le and Mikolov, 2014), the Sequential Denoising Auto-Encoder (SDAE) (Hill et al., 2016) and the SIF model (Arora et al., 2017), all trained on the Toronto book corpus (Zhu et al., 2015). The second group consists of models trained with unsu-

		Caption Retrieval				Image Retrieval			
Model		**R@1**	**R@5**	**R@10**	**Med r**	**R@1**	**R@5**	**R@10**	**Med r**
Direct supervision of sentence representations									
m-CNN	(Ma et al., 2015)	38.3	-	81.0	2	27.4	-	79.5	3
m-CNN$_{\mathrm{ENS}}$	(Ma et al., 2015)	42.8	-	84.1	2	32.6	-	82.8	3
Order-embeddings	(Vendrov et al., 2016)	**46.7**	-	**88.9**	2	**37.9**	-	**85.9**	2
Pre-trained sentence representations									
SkipThought	+ VGG19 (82k)	33.8	67.7	82.1	3	25.9	60.0	74.6	4
SkipThought	+ ResNet101 (113k)	37.9	72.2	84.3	2	30.6	66.2	81.0	3
BiLSTM-Max (on SNLI)	+ ResNet101 (113k)	42.4	**76.1**	87.0	2	33.2	**69.7**	83.6	3
BiLSTM-Max (on AllNLI)	+ ResNet101 (113k)	**42.6**	75.3	**87.3**	2	**33.9**	**69.7**	**83.8**	3

Table 5: **COCO retrieval results**. SkipThought is trained either using 82k training samples with VGG19 features, or with 113k samples and ResNet-101 features (our setting). We report the average results on 5 splits of 1k test images.

pervised ordered sentences such as FastSent and SkipThought (also trained on the Toronto book corpus). We also include the FastSent variant "FastSent+AE" and the SkipThought-LN version that uses layer normalization. We report results from models trained on supervised data in the third group, and also report some results of supervised methods trained directly on each task for comparison with transfer learning approaches.

Comparison with SkipThought The best performing sentence encoder to date is the SkipThought-LN model, which was trained on a very large corpora of ordered sentences. With much less data (570k compared to 64M sentences) but with high-quality supervision from the SNLI dataset, we are able to consistently outperform the results obtained by SkipThought vectors. We train our model in less than a day on a single GPU compared to the best SkipThought-LN network trained for a month. Our BiLSTM-max trained on SNLI performs much better than released SkipThought vectors on MR, CR, MPQA, SST, MRPC-accuracy, SICK-R, SICK-E and STS14 (see Table 4). Except for the SUBJ dataset, it also performs better than SkipThought-LN on MR, CR and MPQA. We also observe by looking at the STS14 results that the cosine metrics in our embedding space is much more semantically informative than in SkipThought embedding space (pearson score of 0.68 compared to 0.29 and 0.44 for ST and ST-LN). We hypothesize that this is namely linked to the matching method of SNLI models which incorporates a notion of distance (element-wise product and absolute difference) during training.

NLI as a supervised training set Our findings indicate that our model trained on SNLI obtains much better overall results than models trained on other supervised tasks such as COCO, dictionary definitions, NMT, PPDB (Ganitkevitch et al., 2013) and SST. For SST, we tried exactly the same models as for SNLI; it is worth noting that SST is smaller than NLI. Our representations constitute higher-quality features for both classification and similarity tasks. One explanation is that the natural language inference task constrains the model to encode the semantic information of the input sentence, and that the information required to perform NLI is generally discriminative and informative.

Domain adaptation on SICK tasks Our transfer learning approach obtains better results than previous state-of-the-art on the SICK task - can be seen as an out-domain version of SNLI - for both entailment and relatedness. We obtain a pearson score of 0.885 on SICK-R while (Tai et al., 2015) obtained 0.868, and we obtain 86.3% test accuracy on SICK-E while previous best hand-engineered models (Lai and Hockenmaier, 2014) obtained 84.5%. We also significantly outperformed previous transfer learning approaches on SICK-E (Bowman et al., 2015) that used the parameters of an LSTM model trained on SNLI to fine-tune on SICK (80.8% accuracy). We hypothesize that our embeddings already contain the information learned from the in-domain task, and that learning only the classifier limits the number of parameters learned on the small out-domain task.

Image-caption retrieval results In Table 5, we report results for the COCO image-caption retrieval task. We report the mean recalls of 5 random splits of 1K test images. When trained with

ResNet features and 30k more training data, the SkipThought vectors perform significantly better than the original setting, going from 33.8 to 37.9 for caption retrieval R@1, and from 25.9 to 30.6 on image retrieval R@1. Our approach pushes the results even further, from 37.9 to 42.4 on caption retrieval, and 30.6 to 33.2 on image retrieval. These results are comparable to previous approach of (Ma et al., 2015) that did not do transfer but directly learned the sentence encoding on the image-caption retrieval task. This supports the claim that pre-trained representations such as ResNet image features and our sentence embeddings can achieve competitive results compared to features learned directly on the objective task.

MultiGenre NLI The MultiNLI corpus (Williams et al., 2017) was recently released as a multi-genre version of SNLI. With 433K sentence pairs, MultiNLI improves upon SNLI in its coverage: it contains ten distinct genres of written and spoken English, covering most of the complexity of the language. We augment Table 4 with our model trained on both SNLI and MultiNLI (AllNLI). We observe a significant boost in performance overall compared to the model trained only on SLNI. Our model even reaches AdaSent performance on CR, suggesting that having a larger coverage for the training task helps learn even better general representations. On semantic textual similarity STS14, we are also competitive with PPDB based paragram-phrase embeddings with a pearson score of 0.70. Interestingly, on caption-related transfer tasks such as the COCO image caption retrieval task, training our sentence encoder on other genres from MultiNLI does not degrade the performance compared to the model trained only SNLI (which contains mostly captions), which confirms the generalization power of our embeddings.

6 Conclusion

This paper studies the effects of training sentence embeddings with supervised data by testing on 12 different transfer tasks. We showed that models learned on NLI can perform better than models trained in unsupervised conditions or on other supervised tasks. By exploring various architectures, we showed that a BiLSTM network with max pooling makes the best current universal sentence encoding methods, outperforming existing approaches like SkipThought vectors.

We believe that this work only scratches the surface of possible combinations of models and tasks for learning generic sentence embeddings. Larger datasets that rely on natural language understanding for sentences could bring sentence embedding quality to the next level.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *Advances in neural information processing systems (NIPS)*.

Yoshua Bengio, Rejean Ducharme, and Pascal Vincent. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

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

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, page 8. IEEE.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*.

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

Miach Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems (NIPS)*, pages 3294–3302.

Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. *Proc. SemEval*, 2:5.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 14, pages 1188–1196.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer International Publishing.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *International Conference on Learning Representations (ICLR)*.

Etai Littwin and Lior Wolf. 2016. The multiverse loss for robust transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3957–3966.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv preprint arXiv:1605.09090*.

Lin Ma, Zhengdong Lu, Lifeng Shang, and Hang Li. 2015. Multimodal convolutional neural networks for matching image and sentence. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2623–2631.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC)*, pages 216–223.

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

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 14, pages 1532–1543.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813.

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

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. Deepface: Closing the gap to human-level performance in face verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, page 8. IEEE.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. *International Conference on Learning Representations (ICLR)*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016a. Charagram: Embedding words and sentences via character n-grams. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016b. Towards universal paraphrastic sentence embeddings. *International Conference on Learning Representations (ICLR)*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 4069–4076. AAAI Press.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Determining Semantic Textual Similarity using Natural Deduction Proofs

Hitomi Yanaka[1]
hitomiyanaka@g.ecc.u-tokyo.ac.jp

Koji Mineshima[2]
mineshima.koji@ocha.ac.jp

Pascual Martínez-Gómez[3]
pascual.mg@aist.go.jp

Daisuke Bekki[2]
bekki@is.ocha.ac.jp

[1]The University of Tokyo
[2]Ochanomizu University
[3]Artificial Intelligence Research Center, AIST
Tokyo, Japan

Abstract

Determining semantic textual similarity is a core research subject in natural language processing. Since vector-based models for sentence representation often use shallow information, capturing accurate semantics is difficult. By contrast, logical semantic representations capture deeper levels of sentence semantics, but their symbolic nature does not offer graded notions of textual similarity. We propose a method for determining semantic textual similarity by combining shallow features with features extracted from natural deduction proofs of bidirectional entailment relations between sentence pairs. For the natural deduction proofs, we use ccg2lambda, a higher-order automatic inference system, which converts Combinatory Categorial Grammar (CCG) derivation trees into semantic representations and conducts natural deduction proofs. Experiments show that our system was able to outperform other logic-based systems and that features derived from the proofs are effective for learning textual similarity.

1 Introduction

Determining semantic textual similarity (STS) is one of the most critical tasks in information retrieval and natural language processing. Vector-based sentence representation models have been widely used to compare and rank words, phrases or sentences using various similarity and relatedness scores (Wong and Raghavan, 1984; Mitchell and Lapata, 2010; Le and Mikolov, 2014). Recently, neural network-based sentence representation models (Mueller and Thyagarajan, 2016; Hill et al., 2016) have been proposed for learning textual similarity. However, these vector-based models often use shallow information, such as words and characters, and whether they can account for phenomena such as negation and quantification is not clear. Consider the sentences: *Tom did not meet some of the players* and *Tom did not meet any of the players*. If functional words such as *some* or *any* are ignored or represented as the same vector, then these sentences are to be represented by identical vectors. However, the first sentence implies that there is a player who Tom did not meet, whereas the second sentence means that Tom did not meet anyone, so the sentences have different meanings.

Conversely, logic-based approaches have been successful in representing the meanings of complex sentences, having had a positive impact for applications such as recognizing textual entailment (Mineshima et al., 2015, 2016; Abzianidze, 2015, 2016). However, purely logic-based approaches only assess entailment or contradiction relations between sentences and do not offer graded notions of semantic similarity.

In this paper, we propose to leverage logic cues to learn textual similarity. Our hypothesis is that *observing proof processes when testing the semantic relations is predictive of textual similarity*. We show that our approach can be more effective than systems that ignore these logic cues.

2 Related Work

Vector-based models of semantic composition have been widely studied with regards to calculating STS. Mitchell and Lapata (2008, 2010) pro-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 681–691
Copenhagen, Denmark, September 7–11, 2017.

posed a sentence vector model involving word vector addition or component-wise multiplication. Addition and multiplication are commutative and associative and thus ignore word order. Polajnar et al. (2015) proposed a discourse-based sentence vector model considering extra-intra sentential context. Also, a categorical compositional distributional semantic model has been developed for recognizing textual entailment and for calculating STS (Grefenstette and Sadrzadeh, 2011; Kartsaklis et al., 2014; Kartsaklis and Sadrzadeh, 2016). However, these previous studies are mostly concerned with the structures of basic phrases or sentences and do not address logical and functional words such as negations and connectives. Neural network-based models of semantic composition (Mueller and Thyagarajan, 2016; Hill et al., 2016) have also been proposed. Although these models achieve higher accuracy, their end-to-end nature introduces challenges in the diagnosis of the reasons that make two sentences to be similar or dissimilar to each other. These diagnosis capabilities may play an important role in making the system explainable and also to guide future system improvements in a more precise manner. Our approach presented in this paper is partially inspired by the latter two objectives.

Meanwhile, some previous studies have proposed logic systems for capturing the semantic relatedness of sentences. The Meaning Factory (Bjerva et al., 2014) uses both shallow and logic-based features for learning textual similarity. In this system, the overlap of predicates and entailment judgments are extracted as logic-based features. UTexas (Beltagy et al., 2014b) uses Probabilistic Soft Logic for learning textual similarity. In this system, each ground atom in the logical formulas has a probability based on distributional semantics of a word. The weights of the logical formulas are calculated from the probabilities of their ground atoms and are extracted as features. These previous studies improved the accuracy by using logic-based features derived from the entailment results of first-order theorem proving in addition to using shallow features such as sentence lengths.

In our study, we determine the semantic similarity of sentences based on the conception of proof-theoretic semantics (Bekki and Mineshima, 2017). The key idea is that not only the entailment results but also the *theorem proving process* can be considered as features for learning textual similarity. That is, by taking into account not only whether a theorem is proved but also *how* it is proved, we can capture the semantic relationships between sentence pairs in more depth.

Another difference between our study and previous logic systems is that we use higher-order predicate logic. Higher-order predicate logic is able to represent complex sentence semantics such as generalized quantifiers more precisely than first-order predicate logic. In addition, higher-order predicate logic makes the logical structure of a sentence more explicit than first-order predicate logic does, so it can simplify the process of proof search (Miller and Nadathur, 1986).

3 System Overview

Figure 1 shows an overview of the system which extracts features for learning textual similarity from logical proofs. To produce semantic representations of sentences and prove them automatically, we use ccg2lambda (Martínez-Gómez et al., 2016), which is a semantic parser combined with an inference system based on natural deduction.

First, sentences are parsed into syntactic trees based on Combinatory Categorial Grammar (CCG) (Steedman, 2000). CCG is a syntactic theory suitable for semantic composition from syntactic structures. Meaning representations are obtained based on semantic templates and combinatory rules for the CCG trees. Semantic templates are defined manually based on formal semantics. Combinatory rules specify the syntactic behaviors of words and compositional rules for the CCG trees. In ccg2lambda, two wide-coverage CCG parsers, C&C (Clark and Curran, 2007) and EasyCCG (Lewis and Steedman, 2014), are used for converting tokenized sentences into CCG trees robustly. According to a previous study (Martínez-Gómez et al., 2017), EasyCCG achieves higher accuracy. Thus, when the output of both C&C and EasyCCG can be proved, we use EasyCCG's output for creating features.

Second, the meanings of words are described using lambda terms. Semantic representations are obtained by combining lambda terms in accordance with the meaning composition rules specified in the CCG tree. The semantic representations are based on Neo-Davidsonian event semantics (Parsons, 1990; Mineshima et al., 2015), in which every verb is decomposed into a predicate over events and a set of functional expressions

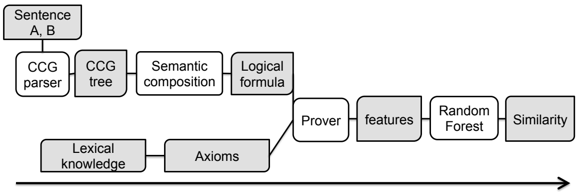


Figure 1: System overview.

relating the events. Adverbs and prepositions are also represented as predicates over events.

Third, we attempt to prove entailment relations between sentence pairs. For this purpose, we use Coq (Bertot and Castran, 2010), which can be used for efficient theorem-proving for natural language inference using both first-order and higher-order logic (Mineshima et al., 2015). Coq's proof calculus is based on natural deduction (Prawitz, 1965), a proof system based on inference rules called introduction and elimination rules for logical connectives. The inference system implemented in ccg2lambda using Coq achieves efficient automatic inference by feeding a set of predefined tactics and user-defined proof-search tactics to its interactive mode. The natural deduction system is particularly suitable for injecting external axioms during the theorem-proving process (Martínez-Gómez et al., 2017).

Finally, features for learning textual similarity are extracted from the proofs produced by ccg2lambda during the theorem-proving process. In this study, we experimented with logistic regression, support vector regression and random forest regression, finding that random forest regression was the most effective. We therefore chose random forest regression for learning textual similarity, with its hyperparameters being optimized by grid search. The mean squared error (MSE) was used to measure the prediction performance of our system.

4 Proof Strategy for Learning Textual Similarity

4.1 Overview of the proof strategy

Sentence similarity depends on complex elements, such as word overlaps and semantic relations. We capture the similarity between the sentence pair (A, B) as a function of the provability of bidirectional entailment relations for (A, B) and combine it with shallow features. After obtaining logical formulas A' and B' from A and B, we attempt to prove the bidirectional entailment relations, $A' \Rightarrow B'$ and $B' \Rightarrow A'$. If the initial natural deduction proofs fail, we re-run the proof, adding relevant external axioms or skipping unproved sub-goals until the proof is completed. After that, features for learning textual similarity are extracted by quantifying the provability of the bidirectional entailment relations.

The details of the procedure are as follows. First, we attempt a natural deduction proof without using external axioms, aiming to prove entailment relations, $A' \Rightarrow B'$ and $B' \Rightarrow A'$. If both fail, then we check whether A' contradicts B', which amounts to proving the negation of the original conclusion, namely $A' \Rightarrow \neg B'$ and $B' \Rightarrow \neg A'$.

The similarity of a sentence pair tends to be higher when the negation of the conclusion can be proved, compared with the case where neither the conclusion nor its negation can be proved. In the SICK (Sentences Involving Compositional Knowledge) dataset (Marelli et al., 2014) (see Section 6.1 for details), 70% of the sentence pairs annotated as contradictory are assigned a relatedness score in $[3, 5)$.

Next, if we fail to prove entailment or contradiction, that is, we cannot prove the conclusion or its negation, we identify an unproved sub-goal which is not matched by any predicate in the premise. We then attempt to prove $A' \Rightarrow B'$ and $B' \Rightarrow A'$ using axiom injection, following the method introduced in Martínez-Gómez et al. (2017). In axiom injection, unproved sub-goals are candidates to form axioms. We focus only on predicates that share at least one argument with both the premise and the conclusion. This means that an axiom can be generated only if there is a predicate p in the pool of premises and a predicate q in a sub-goal and p and q share a variable in an argument position, possibly with the same case (e.g., Subject or Object).

In generating axioms, the semantic relationships between the predicates in the premise and those in the conclusion are checked using lexical knowledge. In this study, we use WordNet (Miller, 1995) as the source of lexical knowledge. Linguistic relations between predicates are checked in the following order: inflections, derivationally related forms, synonyms, antonyms, hypernyms, similarities, and hyponyms. If any one of these relations is found in the lexical knowledge, an axiom can be generated. Again, if the proof fails, we attempt

$$\begin{tikzcd}
G : \mathcal{A} \wedge \mathcal{B} \arrow[d, "\wedge\text{-INTRO}"] & P : \mathcal{A}_1 \wedge \mathcal{A}_2 \wedge \cdots \wedge \mathcal{A}_n \arrow[d, "\wedge\text{-ELIM}"] \\
\begin{array}{c} G_1 : \mathcal{A} \\ G_2 : \mathcal{B} \end{array} & P_1 : \mathcal{A}_1,\ P_2 : \mathcal{A}_2,\ \ldots,\ P_n : \mathcal{A}_n
\end{tikzcd}$$

$$\begin{tikzcd}
G : \mathcal{A} \to \mathcal{B} \arrow[d, "\to\text{-INTRO}"] & \begin{array}{c} P_1 : \mathcal{A} \to \mathcal{B} \\ P_2 : \mathcal{A} \end{array} \arrow[d, "\to\text{-ELIM}"] \\
\begin{array}{c} P : \mathcal{A} \\ G : \mathcal{B} \end{array} & P : \mathcal{B}
\end{tikzcd}$$

$$\begin{tikzcd}
G : \exists x \mathcal{A}(x) \arrow[d, "\exists\text{-INTRO}"] & P : \exists x \mathcal{A}(x) \arrow[d, "\exists\text{-ELIM}"] & \begin{array}{c} P_1 : \mathcal{A}(t) \\ P_2 : t = u \end{array} \arrow[d, "=\text{-ELIM}"] \\
G_1 : \mathcal{A}(x) & P_1 : \mathcal{A}(x) & P : \mathcal{A}(u)
\end{tikzcd}$$

Figure 2: Example of the inference rules used in natural deduction. $P, P_1, \ldots P_n$ are formulas in the premise, while G, G_1, G_2 are formulas in the goal. The initial formulas are at the top, with the formulas obtained by applying the inference rules shown below.

to prove the negation of the conclusion using the axiom injection mechanism.

If the proof by axiom injection fails because of a lack of lexical knowledge, we obtain sentence similarity information from partial proofs by simply accepting the unproved sub-goals and forcibly completing the proof. After the proof is completed, information about the generated axioms and skipped sub-goals is used to create features.

4.2 Proving entailment relations

As an illustration of how our natural deduction proof works, consider the case of proving entailment for the following sentence pair:

A: A man is singing in a bar.
B: A man is singing.

The sentences A and B are mapped onto logical formulas A' and B' based on event semantics via CCG-based semantic composition, as follows.

$$A' : \exists e_1 x_1 x_2 (\mathbf{man}(x_1) \wedge \mathbf{sing}(e_1) \wedge (\mathbf{subj}(e_1) = x_1) \wedge \mathbf{bar}(x_2) \wedge \mathbf{in}(e_1, x_2))$$
$$B' : \exists e_1 x_1 (\mathbf{man}(x_1) \wedge \mathbf{sing}(e_1) \wedge (\mathbf{subj}(e_1) = x_1))$$

First, we attempt a natural deduction proof of $A' \Rightarrow B'$, setting A' as the premise and B' as the goal of the proof. Then A' and B' are decomposed according to the inference rules.

Figure 2 shows the major inference rules we use in the proofs. Inference rules in natural deduction are divided into two types: introduction rules and elimination rules. Introduction rules specify how to prove a formula in the goal, decomposing a goal formula into smaller sub-goals. Elimination rules specify how to use a premise, decomposing a formula in the pool of premises into smaller ones.

$$\begin{tikzcd}
\begin{array}{l} P_0 : \exists e_1 x_1 x_2 (\mathbf{man}(x_1) \wedge \mathbf{sing}(e_1) \wedge (\mathbf{subj}(e_1) = x_1) \\ \quad \wedge\, \mathbf{bar}(x_2) \wedge \mathbf{in}(e_1, x_2)) \\ G_0 : \exists e_1 x_1 (\mathbf{man}(x_1) \wedge \mathbf{sing}(e_1) \wedge (\mathbf{subj}(e_1) = x_1)) \end{array} \arrow[d, "\exists\text{-ELIM } (P_0),\ \exists\text{-INTRO } (G_0)"] \\
\begin{array}{l} P_1 : \mathbf{man}(x_1) \wedge \mathbf{sing}(e_1) \wedge (\mathbf{subj}(e_1) = x_1) \\ \quad \wedge\, \mathbf{bar}(x_2) \wedge \mathbf{in}(e_1, x_2) \\ G_1 : \mathbf{man}(x_1) \wedge \mathbf{sing}(e_1) \wedge (\mathbf{subj}(e_1) = x_1) \end{array} \arrow[d, "\wedge\text{-ELIM } (P_1),\ \wedge\text{-INTRO } (G_1)"] \\
\begin{array}{l} P_2 : \mathbf{man}(x_1),\ P_3 : \mathbf{sing}(e_1),\ P_4 : \mathbf{subj}(e_1) = x_1 \\ \quad P_5 : \mathbf{bar}(x_2),\ P_6 : \mathbf{in}(e_1, x_2), \\ G_2 : \mathbf{man}(x_1),\ G_3 : \mathbf{sing}(e_1),\ G_4 : \mathbf{subj}(e_1) = x_1 \end{array}
\end{tikzcd}$$

Figure 3: The proof process for the example entailment relation.

The proof process for $A' \Rightarrow B'$ is shown in Figure 3. Here A' is initially set to the premise P_0 and B' to the goal G_0. P_0 and G_0 are then decomposed using elimination rules (∧-ELIM, ∃-ELIM) and introduction rules (∧-INTRO, ∃-INTRO). Then we obtain a set of premise formulas $\mathcal{P} = \{P_2, P_3, P_4, P_5, P_6\}$, and a set of sub-goals $\mathcal{G} = \{G_2, G_3, G_4\}$. The proof is performed by searching for a premise P_i whose predicate and arguments match those of a given sub-goal G_j. If such a logical premise is found, the sub-goal is removed. In this example, the sub-goals G_2, G_3, and G_4 match the premises P_2, P_3, and P_4, respectively. Thus, $A' \Rightarrow B'$ can be proved without introducing axioms.

Second, we attempt the proof in the opposite direction, $B' \Rightarrow A'$, by switching P_0 and G_0 in Figure 3. Again, by applying inference rules, we obtain the following sets of premises $\mathcal{P}$ and sub-goals $\mathcal{G}$:

$$\begin{aligned} \mathcal{P} = \{&P_2 : \mathbf{man}(x_1),\ P_3 : \mathbf{sing}(e_1), \\ &P_4 : \mathbf{subj}(e_1) = x_1\} \\ \mathcal{G} = \{&G_2 : \mathbf{man}(x_1),\ G_3 : \mathbf{sing}(e_1), \\ &G_4 : \mathbf{subj}(e_1) = x_1, \\ &G_5 : \mathbf{bar}(x_2), G_6 : \mathbf{in}(e_1, x_2))\} \end{aligned}$$

Here, the two sub-goals G_5 and G_6 do not match any of the premises, so the attempted proof of $B' \Rightarrow A'$ fails. We therefore attempt to inject additional axioms, but in this case no predicate in $\mathcal{P}$ shares the argument x_2 of the predicates $\mathbf{bar}(x_2)$ and $\mathbf{in}(e_1, x_2)$ in $\mathcal{G}$. Thus, no axiom can be generated. To obtain information from a partial proof, we forcibly complete the proof of $B' \Rightarrow A'$ by skipping the unproved sub-goals $\mathbf{bar}(x)$ and $\mathbf{in}(e_1, x_2)$.

4.3 Proving the contradiction

The proof strategy illustrated here can be straightforwardly applied to proving the contradiction. In natural deduction, a negative formula of the form $\neg A$ can be defined as $A \rightarrow \textbf{False}$ ("the formula A implies the contradiction"), by using a propositional constant **False** to encode the contradiction. Thus, the inference rules for negation can be taken as special cases of implication rules, as shown in Figure 4.

As an illustration, let us consider the following sentence pair:

A: No man is singing.

B: There is a man singing loudly.

Figure 5 shows the proof process. The sentences A and B are mapped to P_0 and P_1, respectively, via compositional semantics and the goal G_0 is set to **False**. By decomposing P_1 using elimination rules and then by combining P_2, P_3, and P_4, we can obtain P_6. From P_0 and P_6 we can then derive the contradiction.

These proofs are performed by an automated prover implemented on Coq, using tactics for first-order theorem proving. When a proof is successful, Coq outputs the resulting proof (a proof term), from which we can extract detailed information such as the number of proof steps and the types of inference rules used. In addition to the entailment/contradiction result, information about the proof process is used to create features.

5 Description of the Features

To maximize accuracy when learning textual similarity, we adopt a hybrid approach that uses both logic-based features extracted from the natural deduction proof and other, non-logic-based features. All features are scaled to the $[0, 1]$ range.

5.1 Logic-based Features

We propose 15 features consisting of nine different types of logic-based features. Six of these feature types are derived from the bidirectional natural deduction proofs: six features are extracted from the direct proof ($A' \Rightarrow B'$) and another six from the reverse proof ($B' \Rightarrow A'$). The remaining three feature types are derived from semantic representations of the sentence pairs. The feature types are as follows.

Logical inference result. As stated in Section 4, we include features to distinguish the case where either the conclusion or its negation can be proved from the one where neither can be proved. If the conclusion can be proved, the feature is set to 1.0. If the negation of the conclusion can be proved, the feature is set to 0.5. If neither can be proved, the feature is set to 0.0.

$$\begin{array}{ccc} \begin{array}{l} G : \neg\mathcal{A} \\ \quad\Big\downarrow \neg\text{-}\textsc{Intro} \\ P : \mathcal{A} \\ G : \textbf{False} \end{array} & \qquad & \begin{array}{l} P_1 : \neg\mathcal{A} \\ P_2 : \mathcal{A} \\ \quad\Big\downarrow \neg\text{-}\textsc{Elim} \\ P : \textbf{False} \end{array} \end{array}$$

Figure 4: Inference rules of negation.

$$\begin{array}{l} P_0 : \neg\exists e_1 \exists x_1 (\mathbf{man}(x_1) \land \mathbf{sing}(e_1) \land (\mathbf{subj}(e_1) = x_1)) \\ P_1 : \exists e_1 \exists x_1 (\mathbf{man}(x_1) \land \mathbf{sing}(e_1) \land (\mathbf{subj}(e_1) = x_1) \\ \quad \land\, \mathbf{loudly}(e_1)) \\ G_0 : \textbf{False} \\ \qquad\qquad\qquad\Big\downarrow \exists\text{-}\textsc{Elim}, \land\text{-}\textsc{Elim}\ (P_2) \\ P_2 : \mathbf{man}(x_1),\ P_3 : \mathbf{sing}(e_1),\ P_4 : \mathbf{subj}(e_1) = x_1, \\ P_5 : \mathbf{loudly}(e_1) \\ \qquad\qquad\qquad\Big\downarrow \exists\text{-}\textsc{Intro}, \land\text{-}\textsc{Intro}\ (P_2) \\ P_6 : \exists e_1 \exists x_1 (\mathbf{man}(x_1) \land \mathbf{sing}(e_1) \land \mathbf{subj}(e_1) = x_1) \end{array}$$

Figure 5: Proof process for the contradiction example.

Axiom probabilities. The probability of an axiom and the number of axioms appearing in the proof are used to create features. The probability of an axiom is defined as the inverse of the length of the shortest path that connects the senses in the is-a (hypernym/hyponym) taxonomy in WordNet. When multiple axioms are used in the proof, the average of the probabilities of the axioms is extracted as a feature. If the proof can be completed without using axioms, the feature is set to 1.0.

Proved sub-goals. Given that proofs can be obtained either by proving all the sub-goals or skipping unproved sub-goals, we use the proportion of proved sub-goals as a feature. Our assumption is that if there are more unproved sub-goals then the sentence pair is less similar. When there are m logical formulas in the premise pool and n proved sub-goals, we set the feature to n/m. If the theorem can be proved without skipping any sub-goals, the feature is set to 1.0. It may be the case that the number of sub-goals is so large that some sub-goals remain unproved even after axiom injection. Since the proportion of unproved sub-goals is decreased by axiom injection, we use the proportion of unproved sub-goals both with and without axiom injection as features.

Cases in unproved sub-goals. Subject or object words can affect the similarity of sentence pairs. Therefore, the number of each case in unproved sub-goals, like $\mathbf{subj}(e_1)$ in Figures 3 and 5, is used as a feature. Here, we count subjective, objective, and dative cases.

Proof steps. In general, complex theorems are difficult to prove and in such cases the sentence pairs are considered to be less similar. We therefore use the number of Coq's proof steps, namely the number of inference rule applications in a given proof, as a feature.

Inference rules. The complexity of a natural deduction proof can be measured in terms of the inference rules used for each proof step. We therefore extract the relative frequency with which each inference rule is used in the proof as a feature. We check seven inference rules for natural deduction using Coq (cf. Figure 2): introduction and elimination rules for conjunction ($\wedge$-INTRO, $\wedge$-ELIM), implication ($\rightarrow$-INTRO, $\rightarrow$-ELIM), and existential quantification ($\exists$-INTRO, $\exists$-ELIM), and the elimination rule for equality ($=$-ELIM).

Predicate overlap. Intuitively, the more predicates overlap between the premise and the conclusion, the more likely it is that the inference can be proved. We therefore use the proportion of predicates that overlap between the premise and the conclusion as a feature.

Semantic type overlap. Each semantic representation in higher-order logic has a semantic type, such as Entity for entities and Prop for propositions. As with predicates, we use the degree of semantic type overlap between the premise and the conclusion as a feature.

Existence of negative clauses. Whether or not the premise or conclusion contain negative clauses is an effective measure of similarity. In semantic representations, negative clauses are represented by the negation operator $\neg$, so we check for negation operators in the premise and the conclusion and set this feature to 1.0 if either contains one.

5.2 Non-logic-based Features

We also use the following eight non-logic-based features.

Noun/verb overlap. We extract and lemmatize all nouns and verbs from the sentence pairs and use the degrees of overlap of the noun and verb lemmas as features.

Part-of-speech overlap. We obtain part-of-speech (POS) tags for all words in the sentence pairs by first tokenizing them with the Penn Treebank Project tokenizer[1] and then POS tagging them with C&C POS tagger (Curran and Clark, 2003). The degree of overlap between the sentences' POS tags is used as a feature.

Synset overlap. For each sentence in the pair, we obtain the set containing all the synonym lemmas (the synset) for the words in the sentence. The degree of overlap between the sentences' synsets is used as a feature.

Synset distance. For each word in the first sentence, we compute the maximum path similarity between its synset and the synset of any other word in the second sentence. Then, we use the average of maximum path similarities as a feature.

Sentence length. If the conclusion sentence is long, there will possibly be many sub-goals in the proof. We therefore use the average of the sentence lengths and the difference in length between the premise and the conclusion sentences as features.

String similarity. We use the similarity of the sequence of characters within the sentence pairs as a feature. The Python *Difflib*[2] function returns the similarity between two sequences as a floating-point value in $[0, 1]$. This measure is given by $2.0 * M/T$, where T is the total number of elements in both sequences and M is the number of matches. This feature is 1.0 if the sequences are identical and 0.0 if they have nothing in common.

Sentence similarity from vector space models. We calculate sentence similarity by using three major vector space models, TF-IDF, latent semantic analysis (LSA) (Deerwester et al., 1990), and latent Dirichlet allocation (LDA) (Blei et al., 2003). We use these cosine similarities as features.

Existence of passive clauses. Passive clauses have an influence on similarity. In CCG trees, passive clauses are represented using the syntactic category $S_{pss}\backslash NP$. We check for the occurrence of passive clauses in the premise and conclusion, and if either of them contains a passive clause then the feature is set to 1.0.

[1] ftp://ftp.cis.upenn.edu/pub/treebank/public_html/tokenization.html

[2] https://docs.python.org/3.5/library/difflib.html

ID	Sentence1	Sentence2	Entailment	Score
23	There is no biker jumping in the air.	A lone biker is jumping in the air	*no*	4.2
1412	Men are sawing logs.	Men are cutting wood.	*yes*	4.5
9963	The animal is grazing on the grass.	The cop is sitting on a police bike.	*unknown*	1

Table 1: Examples in the SICK dataset with different entailment labels and similarity scores.

	γ	ρ	MSE
Mueller et al. (2016)	0.882	0.835	0.229
Our system	0.838	0.796	0.561
SemEval2014 Best Score	0.828	0.769	0.325
The Meaning Factory	0.827	0.772	0.322
UTexas	0.714	0.674	0.499
Baseline	0.653	0.745	0.808

Table 2: Results on the test split of SICK dataset.

6 Experiments and Evaluation

6.1 Experimental Conditions

We evaluated our system[3] using two datasets: the SemEval-2014 version of the SICK dataset (Marelli et al., 2014) and the SemEval-2012 version of the MSR-paraphrase video corpus dataset (MSR-vid) (Agirre et al., 2012). The experimental conditions were as follows.

6.1.1 The SICK dataset

The SICK dataset is a dataset for studying STS as well as for recognizing textual entailment (RTE). It was originally developed for evaluating compositional distributional semantics, so it contains logically challenging expressions such as quantifiers, negations, conjunctions and disjunctions. The dataset contains 9927 sentence pairs with a 5000/4927 training/test split. These sentence pairs are manually annotated with three types of labels *yes* (entailment), *no* (contradiction), or *unknown* (neutral) as well as a semantic relatedness scores in $[1, 5]$ (see Table 1 for a sample).

In this dataset, sentence pairs whose gold entailment labels are *no* tend to be scored a little more highly than the average, whereas those whose labels are *unknown* have a wide range of scores. Thus, we set the baseline of the relatedness score to 5 when the gold entailment label was *yes* and to 3 when the label was *no* or *unknown*.

We compared our system with the following systems: the state-of-the-art neural network-based system (Mueller and Thyagarajan, 2016); the best system (Zhao et al., 2014) from SemEval-2014; and two of the logic-based systems stated in Section 2: namely The Meaning Factory (Bjerva et al., 2014) and UTexas (Beltagy et al., 2014b). The Pearson correlation coefficient γ, Spearman's rank correlation coefficient ρ, and the MSE were used as the evaluation metrics.

6.1.2 The MSR-vid dataset

The MSR-vid dataset is our second dataset for the STS task and contains 1500 sentence pairs with a 750/750 training/test split. All sentence pairs are annotated with semantic relatedness scores in the range [0, 5]. We used this dataset to compare our system with the best system from SemEval-2012 (Bär et al., 2012) and the logic-based UTexas system (Beltagy et al., 2014a). We used the Pearson correlation coefficient γ as the evaluation metric.

6.2 Results

Table 2 shows the results of our experiments with the SICK dataset. Although the state-of-the-art neural network-based system yielded the best results overall, our system achieved higher scores than SemEval-2014 submissions, including the two logic-based systems (The Meaning Factory and UTexas), in terms of Pearson correlation and Spearman's correlation.

The main reason for our system's lower performance in terms of MSE is that some theorems could not be proved because of a lack of lexical knowledge. In the current work, we only consider word-level knowledge (word-for-word paraphrasing); we may expand the knowledge base in the future by using more external resources.

As we mentioned above, the sentence pairs annotated as *unknown* produced a wide range of scores. The Pearson correlation of the *unknown* portion of the SICK dataset was 0.766, which suggests that our logic-based system can also be applied to neutral sentence pairs.

Table 3 shows the results of our experiments with the MSR-vid dataset. These results also indicate that our logic-based system achieved higher accuracy than the other logic-based systems.

[3] Available at https://github.com/mynlp/ccg2lambda.

	γ
SemEval2012 Best Score	0.873
Our system	0.853
Beltagy et al. (2014)	0.830

Table 3: Results on the test split of MSR-vid.

	γ	ρ	MSE
Predicate overlap	**0.691**	0.609	**0.734**
Inference rules	0.632	**0.619**	0.794
Probability of axioms	0.543	0.540	0.865
Proof steps	0.458	0.494	0.915
Proved sub-goals	0.432	0.443	0.926
Logical inference result	0.386	0.399	0.939
Unproved sub-goals' case	0.301	0.307	0.973
Semantic type overlap	0.245	0.219	0.987
Negative clauses	0.163	0.323	1.004
Noun/verb overlap	0.661	0.554	0.763
Vector space model	0.594	0.510	0.857
String similarity	0.414	0.418	0.977
Synset overlap	0.382	0.341	0.978
Synset distance	0.352	0.330	0.999
Part-of-speech overlap	0.349	0.346	0.954
Sentence length	0.231	0.240	0.993
Passive clauses	0.023	0.046	1.017
Only logic-based	0.798	0.760	0.613
Only non logic-based	0.793	0.732	0.621
All	**0.838**	**0.796**	**0.561**

Table 4: Results when training our regressor with each feature group in isolation.

Table 4 shows evaluation results for each feature group in isolation, showing that inference rules and predicate overlaps are the most effective features. Compared with the non-logic-based features, the logic-based features achieved a slightly higher accuracy, a point that will be analyzed in more detail in the next section. Overall, our results show that combining logic-based features with non logic-based ones is an effective method for determining textual similarity.

6.3 Positive examples and error analysis

Table 5 shows some examples for which the prediction score was better when using logic-based features than when using non-logic-based ones.

For IDs 642 and 1360, one sentence contains a passive clause while the other sentence does not. In such cases, the sentence pairs are not superficially similar. By using logical formulas based on event semantics we were able to interpret the sentence containing the passive clause correctly and judge that the passive and non-passive sentences are similar to each other.

In ID 891, one sentence contains a negative clause while the other does not. Using shallow features, the word overlap is small and the prediction score was much lower than the correct score. Our logic-based method, however, interpreted the first sentence as a negative existential formula of the form $\neg\exists x\mathcal{P}(x)$ and the second sentence as an existential formula $\exists x\mathcal{P}'(x)$. Thus, it could easily handle the semantic difference between the positive and negative sentences.

In ID 1158, by contrast, the proportion of word overlap is so high that the prediction score with non-logic-based features was much higher than the correct score. Our method, however, was able to prove the contradiction using an antonym axiom of the form $\forall x(\mathbf{remove}(x) \rightarrow \neg\mathbf{add}(x))$ from WordNet and thus predict the score correctly.

In ID 59, the proportion of word overlap is low, so the prediction score with non-logic-based features was lower than the correct score. Our method, however, was able to prove the partial entailment relations for the sentence pair and thus predict the score correctly. Here the logic-based method captured the common meaning of the sentence pair: both sentences talk about the kids playing in the leaves.

Finally, in ID 71, the prediction score with non-logic-based features was much higher than the correct score. There are two reasons for this phenomenon: negations tend to be omitted in non-logic-based features such as TF-IDF and the proportion of word overlap is high. However, as logical formulas and proofs can handle negative clauses correctly, our method was able to predict the score correctly.

Table 6 shows examples where using only logic-based features produced erroneous results. In ID 3974, the probability of axiom $\forall x(\mathbf{awaken}(x) \rightarrow \mathbf{up}(x))$ was low (0.25) and thus the prediction score was lower than the correct score. Likewise, in ID 4833, the probability of axiom $\forall x(\mathbf{file}(x) \rightarrow \mathbf{do}(x))$ was very low (0.09) and thus the prediction score was negatively affected. In these cases, we need to consider phrase-level axioms such as $\forall x(\mathbf{awaken}(x) \rightarrow \mathbf{wake\_up}(x))$ and $\forall x(\mathbf{file\_nail}(x) \rightarrow \mathbf{do\_manicure}(x))$ using a paraphrase database. This, however, is an issue for future study. In ID 1941, the system wrongly proved the bidirectional entailment relations by

ID	Sentence Pair	Gold	Pred +logic	Pred -logic	Entailment
642	A person is climbing a rock with a rope, which is pink. A rock is being climbed by a person with a rope, which is pink.	5.0	4.9	4.1	Yes
1360	The machine is shaving the end of a pencil. A pencil is being shaved by the machine.	4.7	4.6	3.8	Yes
891	There is no one on the shore. A bunch of people is on the shore.	3.6	3.7	2.6	No
1158	A woman is removing ingredients from a bowl. A woman is adding ingredients to a bowl.	3.3	3.5	4.1	No
59	Kids in red shirts are playing in the leaves. Three kids are jumping in the leaves.	3.9	3.8	3.1	Unknown
71	There is no child lying in the snow and making snow angels. Two people in snowsuits are lying in the snow and making snow angels.	3.3	3.3	4.1	Unknown

Table 5: Examples for which our regressor trained only with logic-based features performs better than when using non-logic features. “Gold”: correct score, “Pred+logic”: prediction score only with logic-based features, “Pred-logic”: prediction score only with non-logic-based features.

ID	Sentence Pair	Gold	System	Axiom
3974	A girl is awakening. A girl is waking up.	4.9	3.6	$\forall x(\mathbf{awaken}(x) \rightarrow \mathbf{wake}(x))$ $\forall x(\mathbf{awaken}(x) \rightarrow \mathbf{up}(x))$
4833	A girl is filing her nails. A girl is doing a manicure.	4.2	1.8	$\forall x(\mathbf{nail}(x) \rightarrow \mathbf{manicure}(x))$ $\forall x(\mathbf{file}(x) \rightarrow \mathbf{do}(x))$
1941	A woman is putting the baby into a trash can. A person is putting meat into a skillet.	1.0	3.3	$\forall x(\mathbf{woman}(x) \rightarrow \mathbf{person}(x))$ $\forall x(\mathbf{trash}(x) \rightarrow \mathbf{skillet}(x))$ $\forall x(\mathbf{baby}(x) \rightarrow \mathbf{meat}(x))$

Table 6: Error examples when training the regressor only with logic-based features.

adding external axioms, so the prediction score was much higher than the correct score. Setting the threshold for the probability of an axiom may be an effective way of improving our axiom-injection method.

7 Conclusion

We have developed a hybrid method for learning textual similarity by combining features based on logical proofs of bidirectional entailment relations with non-logic-based features. The results of our experiments on two datasets show that our system was able to outperform other logic-based systems. In addition, the results show that information about the natural deduction proof process can be used to create effective features for learning textual similarity. Since these logic-based features provide accuracy improvements that are largely additive with those provided by non-logic-based features, neural network-based systems may also benefit from using them.

In future work, we will refine our system so that it can be applied to other tasks such as question answering. Compared with neural network-based systems, our natural deduction-based system can not only assess how similar sentence pairs are, but also explain what the sources of similarity/dissimilarity are by referring to information about sub-goals in the proof. Given this interpretative ability, we believe that our logic-based system may also be of benefit to other natural language processing tasks, such as question answering and text summarization.

Acknowledgments

We thank the three anonymous reviewers for their detailed comments. This work was supported by JST CREST Grant Number JPMJCR1301, Japan.

References

Lasha Abzianidze. 2015. A tableau prover for natural logic and language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP-15)*, pages 2492–2502, Lisbon, Portugal. Association for Computational Linguistics.

Lasha Abzianidze. 2016. Natural solution to FraCaS entailment problems. In *Proceedings of the 5th Joint Conference on Lexical and Computational Semantics*, pages 64–74, Berlin, Germany. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 Task 6: A pilot on semantic textual similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval-2012)*, pages 385–393, Montréal, Canada. Association for Computational Linguistics.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval-2012)*, pages 435–440, Montréal, Canada. Association for Computational Linguistics.

Daisuke Bekki and Koji Mineshima. 2017. Context-passing and underspecification in dependent type semantics. In Stergios Chatzikyriakidis and Zhaohui Luo, editors, *Modern Perspectives in Type Theoretical Semantics*, Studies of Linguistics and Philosophy, pages 11–41. Springer.

Islam Beltagy, Katrin Erk, and Raymond Mooney. 2014a. Probabilistic soft logic for semantic textual similarity. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-2014)*, pages 1210–1219, Baltimore, Maryland. Association for Computational Linguistics.

Islam Beltagy, Stephen Roller, Gemma Boleda, Katrin Erk, and Raymond Mooney. 2014b. UTexas: Natural language semantics using distributional semantics and probabilistic logic. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*, pages 796–801, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Yves Bertot and Pierre Castran. 2010. *Interactive Theorem Proving and Program Development: Coq'Art The Calculus of Inductive Constructions*. Springer Publishing Company, Incorporated, New York, USA.

Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. 2014. The Meaning Factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*, pages 642–646, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning*, 3:993–1022.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

James R Curran and Stephen Clark. 2003. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 91–98. Association for Computational Linguistics.

Scott Deerwester, Susan T. Dumais, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP-2011)*, pages 1394–1404, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1367–1377, San Diego, California. Association for Computational Linguistics.

Dimitri Kartsaklis, Nal Kalchbrenner, and Mehrnoosh Sadrzadeh. 2014. Resolving lexical ambiguity in tensor regression models of meaning. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL-2014)*, pages 212–217, Baltimore, Maryland. Association for Computational Linguistics.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2016. Distributional inclusion hypothesis for tensor-based composition. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers (COLING-2016)*, pages 2849–2860, Osaka, Japan. The COLING 2016 Organizing Committee.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, (ICML-2014)*, pages 1188–1196, Beijing, China.

Mike Lewis and Mark Steedman. 2014. A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-2014)*, pages 990–1000, Doha, Qatar. Association for Computational Linguistics.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC-2014)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association.

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2016. ccg2lambda: A compositional semantics system. In *Proceedings of ACL-2016 System Demonstrations*, pages 85–90, Berlin, Germany. Association for Computational Linguistics.

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2017. On-demand injection of lexical knowledge for recognising textual entailment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2017)*, pages 710–720, Valencia, Spain. Association for Computational Linguistics.

Dale A. Miller and Gopalan Nadathur. 1986. Some uses of higher-order logic in computational linguistics. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, pages 247–256, New York, New York, USA. Association for Computational Linguistics.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP-2015)*, pages 2055–2061, Lisbon, Portugal. Association for Computational Linguistics.

Koji Mineshima, Ribeka Tanaka, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2016. Building compositional semantics and higher-order inference system for a wide-coverage Japanese CCG parser. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2236–2242, Austin, Texas. Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 236–244, Columbus, Ohio. Association for Computational Linguistics.

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

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-2016)*, pages 2786–2792, Arizona, USA. Association for the Advancement of Artificial Intelligence.

Terence Parsons. 1990. *Events in The Semantics of English: a Study in Subatomic Semantics*. MIT Press, Cambridge, USA.

Tamara Polajnar, Laura Rimell, and Stephen Clark. 2015. An exploration of discourse-based sentence spaces for compositional distributional semantics. In *Proceedings of the 1st Workshop on Linking Computational Models of Lexical, Sentential and Discourse-level Semantics*, pages 1–11, Lisbon, Portugal. Association for Computational Linguistics.

Dag Prawitz. 1965. *Natural Deduction – A Proof-Theoretical Study*. Almqvist & Wiksell, Stockholm, Sweden.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, USA.

S. K. M. Wong and Vijay V. Raghavan. 1984. Vector space model of information retrieval: A reevaluation. In *Proceedings of the 7th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 167–185.

Jiang Zhao, Tiantian Zhu, and Man Lan. 2014. ECNU: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval-2014)*, pages 271–277, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Multi-Grained Chinese Word Segmentation

Chen Gong, Zhenghua Li,[*] Min Zhang, Xinzhou Jiang
Soochow University, Suzhou, China
cgong@stu.suda.edu.cn, {zhli13,minzhang}@suda.edu.cn, xzjiang.hw@gmail.com

Abstract

Traditionally, word segmentation (WS) adopts the single-granularity formalism, where a sentence corresponds to a single word sequence. However, Sproat et al. (1996) show that the inter-native-speaker consistency ratio over Chinese word boundaries is only 76%, indicating single-grained WS (SWS) imposes unnecessary challenges on both manual annotation and statistical modeling. Moreover, WS results of different granularities can be complementary and beneficial for high-level applications.

This work proposes and addresses multi-grained WS (MWS). First, we build a large-scale pseudo MWS dataset for model training and tuning by leveraging the annotation heterogeneity of three SWS datasets. Then we manually annotate 1,500 test sentences with true MWS annotations. Finally, we propose three benchmark approaches by casting MWS as constituent parsing and sequence labeling. Experiments and analysis lead to many interesting findings.

1 Introduction

As the first processing step of Chinese language processing, word segmentation (WS) has been extensively studied and made great progress during the past decades, thanks to the annotation of large-scale benchmark datasets, among which the most widely-used are Microsoft Research Corpus (MSR) (Huang et al., 2006), Peking University People Daily Corpus (PPD) (Yu et al., 2003), and Penn Chinese Treebank (CTB) (Xue et al., 2005). Table 1 gives an example sentence segmented in different guidelines. Meanwhile, WS approaches gradually evolve from maximum matching based on lexicon dictionaries (Liu and Liang, 1986), to path searching from segmentation graphs based on language modeling scores and other statistics (Zhang and Liu, 2002), to character-based sequence labeling (Xue, 2003), to shift-reduce incremental parsing (Zhang and Clark, 2007). Recently, neural network models have also achieved success by effectively learning representation of characters and contexts (Zheng et al., 2013; Pei et al., 2014; Ma and Hinrichs, 2015; Chen et al., 2015; Zhang et al., 2016; Cai and Zhao, 2016; Liu et al., 2016).

<table>
<tr><td>MSR</td><td colspan="4">全国各地</td><td>医学</td><td>界</td><td>专家</td><td colspan="2">走出</td><td colspan="2">人民大会堂</td></tr>
<tr><td>PPD</td><td colspan="2">全国</td><td colspan="2">各地</td><td colspan="2">医学界</td><td>专家</td><td>走</td><td>出</td><td>人民</td><td>大会堂</td></tr>
<tr><td>CTB</td><td>全</td><td>国</td><td>各</td><td>地</td><td colspan="2">医学界</td><td>专家</td><td colspan="2">走出</td><td>人民</td><td>大会堂</td></tr>
</table>

Table 1: An example of annotation heterogeneity: 全 (all) 国 (country) 各 (every) 地 (place) 医学 (medical science) 界 (field) 专家 (experts) 走 (walk) 出 (out) 人民 (people) 大会堂 (great hall).

To date, all the labeled datasets adopt the single-granularity formalization, and previous research mainly focuses on single-grained WS (SWS), where one sentence is segmented into a single word sequence. Although different WS guidelines share the same high-level criterion of word boundaries – a character string combined closely and used steadily forms a word, people greatly diverge due to individual differences on knowledge and living environments, etc. An anonymous reviewer kindly points

[*] Correspondence author

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 692–703
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

out that Vladímir Skalička of the Prague School claimed that unlike the “isolating” languages such as French and English, Chinese belongs to the “polysynthetic” type, in which compound words are normally produced from indigenous morphemes (Jernudd and Shapiro, 1989). The vague distinction between morphemes and compounds also contribute to the cognition divergence on the concept of words. Sproat et al. (1996) show that the consensus ratio over word boundaries is only 76% among Chinese native speakers without trained on a common guideline.To fill this gap, WS guidelines need to further group words into many types and provide illustration examples for each type. Nevertheless, it is very challenging even for well-trained annotators to fully grasp the guidelines and to be consistent on uncovered cases. For example, Xiu (2013) (in Tables 1-3) shows that about 3% characters are inconsistently segmented in the PPD training data used in SIGHAN Bakeoff 2005 (Emerson, 2005). We have also observed many inconsistency cases in all MSR/PPD/CTB during this work. In a word, SWS imposes great challenge on data annotation, and as a side effect, enforces statistical models to learn subtleness of annotation guidelines rather than the true WS ambiguities.

From another perspective, WS results of different granularities may be complementary in supporting applications such as information retrieval (IR) (Liu et al., 2008) and machine translation (MT) (Su et al., 2017). On the one hand, coarse-grained words enable statistical models to perform more exact matching and analyzing. On the other hand, fine-grained words are helpful in both reducing data sparseness and supporting deeper understanding of language.[1]

To solve the above two issues for SWS, this paper proposes and addresses multi-grained WS (MWS). Given an input sentence, the goal is to produce a hierarchy structure of all words of different granularities, as illustrated in Figure 1. To tackle the lack of labeled data, we build a large-scale pseudo MWS dataset for model training and tuning by automatically converting annotations of three heterogeneous SWS datasets (i.e. MSR/PPD/CTB) based on the recently proposed coupled sequence labeling approach of Li et al. (2015). In order to fully investigate the problem, we manually annotate 1,500 test sentences with true MWS annotations. Finally, we propose three benchmark approaches by casting MWS as constituent parsing and sequence labeling problems. Experiments and data analysis lead to many interesting findings.

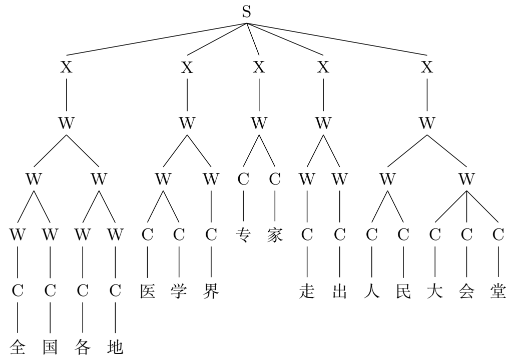

S
X X X X X
W W W W W
W W W W C C W W W W
W W W W C C C 专 家 C C C C C C C
C C C C 医 学 界 走 出 人 民 大 会 堂
全 国 各 地

Figure 1: MWS as a constituent parse tree.

We will release the newly annotated data and the codes of the benchmark approaches at http://hlt.suda.edu.cn/~zhli. However, due to the license issue, we may not directly release all the pseudo MWS datasets. Instead, we will launch a web service for obtaining MWS annotations given a sentence with one of MSR/PPD/CTB annotations.

2 Pseudo MWS Data Conversion

This section introduces the process of gathering pseudo MWS data by making use of the annotation heterogeneity of the three existing datasets, i.e., MSR/PPD/CTB.

2.1 Annotation Heterogeneity

MSR is a manually labeled corpus with word boundaries and named entity tags, and is annotated by Microsoft Research Asia for supporting Chinese text processing (Huang et al., 2006). The key characteristic of MSR is treating named entities as single words. For example, “人民大会堂 (Great Hall of the People)” is a location and forms a word in Table 1. In general, MSR is more coarse-grained than PPD and CTB. PPD is a large-scale corpus with word boundaries, POS tagging, and phonetic notations to facilitate Chi-

[1] Words in CTB are generally more fine-grained than those in PPD and MSR, probably due to the requirement of annotating syntactic structures.

nese information processing, and is annotated by Institute of Computational Linguistics at Peking University (Yu et al., 2003). Based on the Penn Chinese Treebank Project, CTB is built to create a Mandarin Chinese corpus with syntactic bracketing (Xue et al., 2005). We find that CTB is more fine-grained in word boundaries than MSR and PPD, since syntactic annotation tends to require deeper understanding of a sentence. For example, Table 5 reports the averaged number of characters per word in each corpus, and confirms our observations.

For better understanding of annotation heterogeneity, we summarize high-frequency differences among the three datasets observed and gathered during this study in Appendix A. However, it is difficult to obtain a complete list of annotation correspondences among the three datasets, since there are too many low-frequency and irregular cases. Moreover, we also observe a lot of inconsistency annotations of the same word or words with similar structures in all three datasets, as shown in Appendix B.

2.2 Coupled WS for Conversion: MSR/PPD as Example

This section introduces how to automatically produce high-quality PPD-side WS labels for a sentence with MSR-side gold-standard WS labels, by leveraging the two non-overlapping SWS data of MSR and PPD with the coupled sequence labeling approach of Li et al. (2015) and Li et al. (2016). Figure 2 shows the workflow.

Given a sentence $\mathbf{x} = [c_1, ..., c_i, ..., c_n]$, the coupled model aims to produce a sequence of bundled tags $\mathbf{t} = [t_1^a t_1^b, ..., t_i^a t_i^b, ..., t_n^a t_n^b]$, where t_i^a and t_i^b are two labels corresponding to two heterogeneous guidelines respectively. Table 2 gives an example of coupled WS on MSR/PPD. We employ the standard four-tag label set to mark word boundaries of one granularity, among which B, I, E respectively represent that the concerned character situates at the *beginning, inside, end* position of a word, and *S* represents a single-character word. The bottom row shows the gold-standard bundled tag sequence.

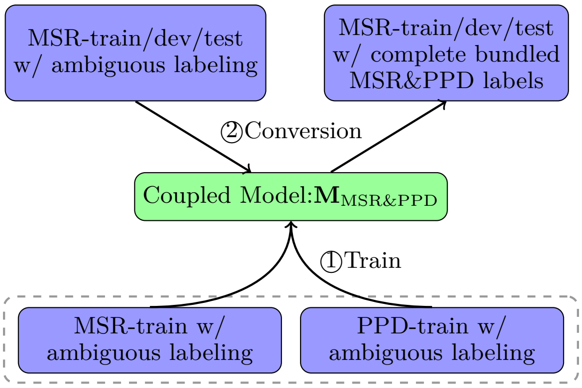


Figure 2: Conversion between MSR/PPD.

Input	全	国	各	地	医	学	界	专	家	...
Ambiguous	BB	IB	IB	EB	BB	EB	SB	BB	EB	...
Labeling for	BI	II	II	EI	BI	EI	SI	BI	EI	...
Training &	BE	IE	IE	EE	BE	EE	SE	BE	EE	...
Conversion	BS	IS	IS	ES	BS	ES	SS	BS	ES	...
Output	BB	IE	IB	EE	BB	EI	SE	BB	EE	...

Table 2: Coupled WS (MSR/PPD as example). Two WS labels are bundled to represent MSR/PPD annotations for a character. Ambiguous labeling is gained supposing this sentence has MSR-side gold-standard annotations.

One key advantage of the coupled model is to directly learn from two *non-overlapping* heterogeneous training datasets, where each dataset only contains single-side gold-standard labels. To deal with this partial (or incomplete) labeling issue, they project each single-side label to a set of bundled labels by considering all labels at the missing side, as shown in the second row in Table 2. Such *ambiguous labelings* are used for model supervision.

Under a traditional CRF, the coupled model defines the score of a bundled tag sequence as

$$
\begin{aligned}
Score(\mathbf{x}, \mathbf{t}; \theta) &= \theta \cdot \mathbf{f}(\mathbf{x}, \mathbf{t}) \\
&= \sum_{i=1}^{n+1} \theta \cdot \begin{bmatrix} \mathbf{f}_{joint}(\mathbf{x}, i, t_{i-1}^a t_{i-1}^b, t_i^a t_i^b) \\ \mathbf{f}_{sep\_a}(\mathbf{x}, i, t_{i-1}^a, t_i^a) \\ \mathbf{f}_{sep\_b}(\mathbf{x}, i, t_{i-1}^b, t_i^b) \end{bmatrix}
\end{aligned}
$$

where $\mathbf{f}_{joint}(.)$ are the *joint features* whereas $\mathbf{f}_{sep\_a/sep\_b}(.)$ are the separate features. Li et al. (2015) demonstrate that the joint features capture the implicit mappings between heterogeneous annotations, while the back-off separate features work as a remedy for the sparseness of the joint features.

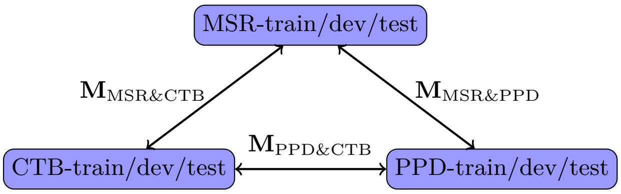

MSR-train/dev/test
$\mathbf{M}_{\text{MSR\&CTB}}$
$\mathbf{M}_{\text{MSR\&PPD}}$
$\mathbf{M}_{\text{PPD\&CTB}}$
CTB-train/dev/test
PPD-train/dev/test

Figure 3: Producing pseudo MWS data.

In their case study of POS tagging, Li et al. (2015) show the coupled model improves tagging accuracy by 95.0 − 94.1 = 0.9% on CTB5-test over the baseline non-coupled model trained on a single training data.

More importantly, they show that the coupled model can be naturally used for the task of annotation conversion, where second-side labels are automatically annotated, given one-side gold-standard labels. The given one-side tags are used to obtain ambiguous labelings, as shown in Table 2, and the coupled model finds the best bundled tag sequence in the *constrained search space*, instead of in the whole bundled tag space, hence greatly reducing the difficulty. Li et al. (2015) report that the coupled model can improve conversion accuracy on POS tagging by 93.9 − 90.6 = 3.3% over the non-coupled model.[2]

2.3 Producing Pseudo MWS Data

Figure 3 shows the workflow of producing pseudo MWS data with three separately trained coupled models. Please note that one coupled model is able to perform conversion between one pair of annotation standards, and thus three coupled models are required for three kinds of annotation standards. Another alternative is that we could directly train one coupled model on MSR/PPD/CTB by extending the approach of Li et al. (2015) from two guidelines into three, which would lead to a much larger bundled tag space. For simplicity, we directly employ their released codes in this work, and leave that for future exploration.

After conversion, we obtain 9 pseudo MWS datasets (i.e., MSR/PPD/CTB-train/dev/test) and represent each sentence in a hierarchy structure as shown in Figure 1. Please kindly note that the guideline-specific information are thrown away, since we do not care which word belongs to which guideline.

In the resulting pseudo MWS data, we find about 0.08% of words overlap with other words, meaning a string "ABC" is segmented into "A/BC" and "AB/C" in two different annotations. We have manually checked these words, and find almost all those cases are caused by conversion errors. This confirms that our treatment of MWS as a hierarchy structure is reasonable.

3 Manual Annotation

In order to fully investigate the MWS problem, we have manually created a true MWS data of 1,500 sentences for final evaluation. From each test dataset in Table 5, we randomly sample 500 sentences with converted pseudo MWS annotations for manual correction. First, two coauthors of this work spent about two hours each day on manual correction of the pseudo MWS annotations for two weeks. During this period, we have summarized a list of high-frequency corresponding patterns among the three guidelines (see Appendix A), and have also written a simple program to automatically detect inconsistent annotations of given words in different training datasets, so that annotators can use the outputs of the program to decide ambiguous cases, which we find is extremely helpful for annotation.

Then, we employ 10 postgraduate students as our annotators who are at different familiarity in WS annotation. Before formal annotation, the annotators are trained for two hours on the basic concepts of MWS, high-frequency correspondences among the three guidelines, and the use of the outputs of the program. We also encourage the annotators to access the three training datasets directly for studying concrete cases under real contexts. Moreover, annotators are asked to recheck their annotations before final submission to improve quality.

To measure the inter-annotator consistency, 150 sentences (10%) are sampled for double annotation, and are grouped into four batches for four pairs of annotators. After annotation, two annotators on the same batch compare

[2] The accuracy seems quite low. The reason is only the 20% most ambiguous words of each sentence are manually labeled and evaluated in their experiments.

	#Words	Granularities Distrbution (%)		
		Single	Two	Three
Before	44,593	74.5	24.0	1.5
After	45,279	71.6	26.8	1.6

Table 3: Data statistics of the MWS test data before and after manual annotation.

their results and produce a consensus submission through discussion.

The annotation process lasts for four days, and each annotator spends about 8 hours in total on completing 160 sentences on average. Table 3 compares data statistics on the $1,500$ sentences before and after manual annotation. The second column reports the number of words, and the last three columns report the distribution of words according to their granularity levels. To illustrate how to gain the distribution, we take Figure 1 as an example, which contains 1 single-grained words, 9 two-grained words, and 7 three-grained words.[3]

Table 3 shows that only 71.6% of all words are single-grained, which is somehow roughly consistent with the inter-native-speaker consistency ratio (76%) in Sproat et al. (1996). Among multi-grained words, $\frac{26.8}{26.8+1.6} = 94.4\%$ are two-grained. It is clear that manual annotation increases both the number of words by $\frac{45{,}279-44{,}593}{44{,}593} = 1.5\%$, and the number of multi-grained words by $74.5 - 71.6 = 2.9\%$. In fact, during annotation, we also feel that multi-granularity phenomena are under-represented in the pseudo MWS data. The reason may be two-fold. First, the conversion models incline to suppress granularity differences, since most words have the same granularity in different datasets. Second, the exist of many inconsistencies in the same dataset also makes the conversion models more reluctant to produce multi-grained words.

The inter-annotator consistency ratio is $\frac{3859}{3935} = 98.07\%$, where the denominator is the word number after merging the submission of all annotator pairs, and the numerator is the consensus word number. We argue that the consistency ratio is not high, considering most words do not need correction in the pseudo MWS annotations. In fact, we find that this annotation task is actually very difficult, since the annotators must consider three guidelines simultaneously. The main inconsistency source of all four annotator pairs are due to the situation where one annotator notices a mistake while another annotator overlooks it. To solve this issue, our long-term plan is to compile a unified MWS guideline by integrating existing SWS guidelines, and gradually improve it by more manual MWS annotation.[4]

4 Benchmark MWS Approaches

There has recently been a surge of interest in applying neural network models to both parsing and sequence labeling tasks. In this work, we propose three simple benchmark approaches for MWS, inspired by recently neural models for constituent parsing (Cross and Huang, 2016) and SWS (Pei et al., 2014).

4.1 MWS as Constituent Parsing

Due to its hierarchy structure shown in Figure 1, we naturally cast MWS as a constituent parsing problem, where characters are leaf nodes; “C” represent a character, “W” represent a word; “X” means that the spanning word cannot be further merged into a more coarse-grained word.

We employ the recently proposed transition-based constituent parser of Cross and Huang (2016) due to its simplicity and competitive performance on different parsing benchmark datasets. In the transition system, a stack S stores processed tokens and partial trees collected so far; a queue Q contains unprocessed tokens; structural[5] and labeling[6] decisions are alternatively made to advance the state until a complete tree forms. The network architecture is composed of two parts: 1) two cascaded

[3] Formally, we call a word s three-grained if there are two other words $s1$ and $s2$ satisfying any one conditions: 1) $s2 \in s1 \in s$ (like “全国各地” in Figure 1); 2) $s2 \in s \in s1$ (like “全国”); 3) $s \in s1 \in s2$ (like “全”), where $\in$ means substring. The definition of two-grained words is analogous; otherwise single-grained.

[4] Although this work has been confined to the three guidelines of MSR/PPD/CTB, we feel that the three guidelines can well capture most multi-granularity phenomena of words. During manual annotation, we have found very few cases where an obvious multi-granularity structure is not covered by the three guidelines.

[5] Shifting the first token in Q into S, or combining the top two items in S

[6] Assigning a non-terminal label or “NULL” to the top item in S

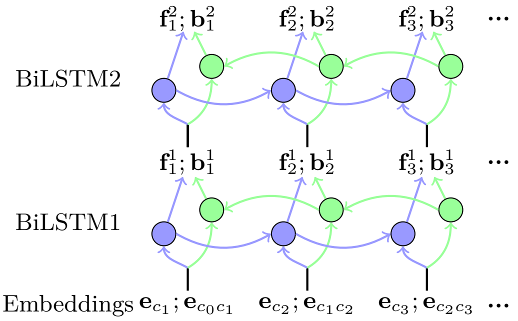

$\mathbf{f}^2_1;\mathbf{b}^2_1$
$\mathbf{f}^2_2;\mathbf{b}^2_2$
$\mathbf{f}^2_3;\mathbf{b}^2_3$
···
BiLSTM2
$\mathbf{f}^1_1;\mathbf{b}^1_1$
$\mathbf{f}^1_2;\mathbf{b}^1_2$
$\mathbf{f}^1_3;\mathbf{b}^1_3$
···
BiLSTM1
Embeddings $\mathbf{e}_{c_1};\mathbf{e}_{c_0c_1}$
$\mathbf{e}_{c_2};\mathbf{e}_{c_1c_2}$
$\mathbf{e}_{c_3};\mathbf{e}_{c_2c_3}$
···

Figure 4: Two-layer BiLSTM architecture.

Chars	全	国	各	地	医	学	界	专	家	...
MWS labels	SBB	SEI	SBI	SEE	BB	EI	SE	B	E	...

Table 4: MWS as sequence labeling. SWS labels for the same character are organized fine-to-coarse.

bidirectional LSTM layers to encode the input token sequence, as shown in Figure 4; 2) two separate multilayer perceptrons (MLPs) to make structural/labeling decisions based on 4/3 simple LSTM span features. A span feature represents a sentence span (i, j) by concatenating the element-wise differences of BiLSTM outputs:

$$\mathbf{r}_{(i,j)} = [\mathbf{f}^1_j - \mathbf{f}^1_{i-1}; \mathbf{b}^1_i - \mathbf{b}^1_{j+1}; \mathbf{f}^2_j - \mathbf{f}^2_{i-1}; \mathbf{b}^2_i - \mathbf{b}^2_{j+1}]$$

To adapt the original parsing model to our MWS task, we concatenate bichar embeddings $\mathbf{e}_{c_{i-1}c_i}$ with single char embedding $\mathbf{e}_{c_i}$ as inputs to the first-layer BiLSTM, inspired by Pei et al. (2014), who show that bichar embeddings are very helpful for SWS.

4.2 MWS as Sequence Labeling

It is also straightforward to model MWS as a sequence labeling task by replacing SWS labels with MWS labels for each character. Table 4 encodes the MWS structure in Figure 1 with a sequence of MWS labels. The idea is to concatenate multiple SWS tags simultaneously for one character to denote the positions of the character under words of different granularities. Please note that each MWS label contains at most three SWS labels since we only consider three SWS datasets in this work. Here, we organize the SWS labels in the order of fine-to-coarse granularities.

For simplicity and fair comparison, we adopt a similar network architecture as the parsing model described in Section 4.1. To decide the MWS label of a character c_i in the input sentence, we feed the outputs of the two-layer BiLSTM outputs $[\mathbf{f}^1_i;\mathbf{b}^1_i,\mathbf{f}^2_i;\mathbf{b}^2_i]$ into a single-hidden-layer MLP.

	Train	Dev	Test	#Char per Word
MSR	78,232	8,692	3,985	1.71
PPD	46,815	2,000	5,000	1.67
CTB	16,091	803	1910	1.63

Table 5: Data statistics (in sentence number). The last column reports the averaged character number of each word.

4.3 MWS as SWS Aggregation

Instead of directly training a MWS model on the three pseudo MWS training datasets, we can also train three separate SWS models on the three SWS training datasets. Given an input sentence, we apply the three SWS models and then merge their outputs as MWS results.

The network architecture is the same with the sequence labeling model in Section 4.2, except the MLP outputs correspond to SWS labels instead of MWS labels.

5 Experiments

Data: for MSR, we adopt the training/test datasets of the SIGHAN Bakeoff 2005 (Emerson, 2005), and cut off 10% random training sentences as the dev data following Zhang et al. (2016); for PPD and CTB, we follow Li et al. (2015) and directly adopt their datasets and data split. Table 5 shows the data statistics.[7]

Evaluation Metrics: the goal of MWS is to precisely produce all words of different granularities given the input sentence. Therefore, to reach a balance of both precision ($P = \frac{\#Word_{gold \cap sys}}{\#Word_{sys}}$) and recall ($R = \frac{\#Word_{gold \cap sys}}{\#Word_{gold}}$), we use the F1 score ($= \frac{2PR}{P+R}$) as in SWS.

Hyper-parameter: we implement all our approaches based on the codes released by Cross and Huang (2016), by making extensions such as adding bichar embeddings and

[7] A DBC-to-SBC (double/single-byte characters) case preprocessing is performed on all datasets to avoid encoding inconsistency.

	Dev (Pseudo)			Test (Manual)							
	P	R	F	P	R	F	#Words	Single	Two	Three	Overlapping
Parsing	96.55	96.40	96.48	97.00	95.16	**96.07**	44,408	74.9%	23.5%	1.6%	–
w/o Bichar Emb	95.58	95.04	95.51	96.37	94.11	95.22	44,434	74.2%	24.1%	1.7%	–
Sequence Labeling	**96.86**	96.26	**96.59**	**97.01**	94.96	95.97	44,323	75.8%	22.7%	1.5%	–
w/o Bichar Emb	95.88	94.94	95.41	96.56	94.18	95.35	44,162	75.7%	22.8%	1.5%	–
SWS Aggregation	90.43	**97.44**	93.80	92.11	**96.59**	94.30	47,478	64.6%	31.4%	4.0%	1.0%

Table 6: Performance of different MWS approaches.

supporting sequence labeling.[8] For simplicity, char and bichar embeddings are randomly initialized following Cross and Huang (2016). The dimensions of char and bichar embeddings are both 50 and other hyper-parameters are the same with Cross and Huang (2016). In our preliminary experiments, we observe that under their neural network framework, the MWS performance is quite stable when rerunning under random initialization or reasonably altering other hyper-parameters. Due to time limitation, we leave the use of pre-trained embeddings and more hyper-parameter tuning for future exploration.

Training/test settings: when training the parsing and sequence labeling based MWS models (not SWS aggregation) on MSR/PPD/CTB-train, we adopt the simple corpus weighting strategy used in Li et al. (2015) to balance the contributions of each training dataset. Before each iteration, we randomly sample 10,000 sentences from each training dataset, and merge and shuffle them for one-iteration training. We use merged MSR/PPD/CTB-dev as the MWS dev data for model selection.[9]

For the SWS aggregation model, three SWS models are separately trained on the three training/dev datasets. For evaluation, three SWS outputs produced independently are merged as one MWS result given a sentence.

In all experiments, training stops when F-score on the dev data does not improve in 20 consecutive iterations, and we choose the model that performs best on the dev data for final evaluation.

Main results: Table 6 reports the performance of different approaches on both the pseudo MWS dev data and the manually annotated MWS test data. The “#Word” column reports the total number of words returned by the corresponding model; the following three columns show the percentages of words of different granularities; the last “Overlapping” column gives the percent of words that overlap with other words, which only happens in the “SWS aggregation” approach, since no constraint can be applied to the three separate SWS models during testing. From the results, we can draw the following findings.

First, the results suggest that using pseudo training and dev datasets to build a MWS model is feasible, based on two evidences: 1) our simple benchmark model can reach a high F-score of 96.07% on the manually annotated test data, which is 1.77% higher than directly aggregating outputs of three SWS models; 2) the P/R/F scores on the pseudo dev data and on the manually labeled test data are quite consistent in general, indicating that it is reliable to use the pseudo dev data for model selection and tuning.

Second, the parsing approach and the sequence labeling approach (with or without bichar embeddings) achieve very similar performance (within 0.15% vibration), More importantly, the parsing approach produces more words and more multi-grained words than the sequence labeling approach, indicating that it is potentially more proper to model MWS as a parsing problem in order to better capture and represent multi-granularity structures. Another possible disadvantage of the sequence labeling approach is that the trained model cannot produce more granularity levels (e.g., four-grained) beyond

[8] https://github.com/jhcross/span-parser. We are very grateful for their helping us solve some code issues at the early stage of this work.

[9] For MSR-dev, only the first 3,000 sentences are used during training due to efficiency concern.

those in the training data. Nevertheless, compared against the manual annotations in Table 3, both the parsing and sequence labeling approaches retrieve much less multi-grained words, which is caused by the under-representation issue of the pseudo training data, as discussed in Section 3.

Third, the SWS aggregation approach achieves the best recall at the price of very low precision on both dev/test data. We believe the reason is that training three SWS models separately on one of the three training datasets has two disadvantages: 1) connections among different guidelines are totally ignored, leading to many overlapping words (1.0%); 2) smaller training data also degrades the performance of each SWS model.

Finally, using bichar embeddings turns out very helpful for MWS, and leads to 0.97 ∼ 1.18% F-score improvement on dev data and 0.62 ∼ 0.85% on test data, which is consistent with the SWS results in Pei et al. (2014).

6 Related Work

As far as we know, this is the first work that formally proposes and addresses the problem of Chinese MWS under the data-driven machine learning framework. It is true that the industrial community, driven by practical demand, has long been interested in retrieving words of different granularities from the engineering perspective, based on lexicon dictionaries and heuristic rules (Zhu and Li, 2008; Hou et al., 2010). We also discover two publicly released toolkits, i.e., IKAnalyzer[10] and PoolWord[11], which consider all substrings in a sentence and return those above a threshold probability as candidate words. In contrast, this paper defines MWS as a strict hierarchy structure, and propose a supervised learning framework for the problem.

To alleviate the high OOV-ratio issue of character-based sequence labeling, Zhang et al. (2006) and Zhao and Kit (2007) propose subword-based sequence labeling for word segmentation by extracting high-frequency subword and treating them as the basic labeling units. Li (2011) and Li and Zhou (2012) propose to jointly parse the internal structures of words and syntactic structure of a sentence. Their definition of internal structures mainly considers prefix or suffix information. They manually annotate the internal structures of words that have high-frequency prefixes or suffixes and left other words with flat structures in CTB. Zhang et al. (2013) further annotate internal structures of all words in CTB and then perform character-level parsing with WS labels. Cheng et al. (2015) propose to cope with the multiple WS standard problem based on internal word structures. After close study of the above works, we find that the MWS annotations automatically built in this work actually capture a lot of subwords and word internal structures in previous works. Most importantly, the main focus of previous works is to improve SWS or parsing performance, whereas this work aims to build a hierarchy structure of multi-grained words. We leave the integration of MWS and parsing for future work.

It has been a long debate whether there exists an optimal WS granularity for MT, which is further complicated by the inevitable mistakes contained in 1-best WS outputs. Dyer et al. (2008) propose an MT model based on source-language word lattices, obtained by merging the outputs of different segmenters. Xiao et al. (2010) propose joint SWS and MT based on word lattices. Recently, Su et al. (2017) propose a word lattice-based neural MT model. They train many segmenters on MSR/PPD/CTB, and merge the outputs to produce word lattices for source-language sentences, which is similar to our SWS aggregation approach. All above works show the usefulness of word lattices instead of a single SWS output. In help IR, Liu et al. (2008) propose a ranking based WS approach for producing words of different granularities. We believe this work can further help both IR and MT by supplying with more accurate MWS results.

7 Conclusion

This work proposes and addresses the problem of MWS, so that all words of different granularities can be captured in a hierarchy structure given a sentence. We can draw the

[10] https://github.com/medcl/elasticsearch-analysis-ik

[11] http://pullword.com/

following interesting findings.

(1) Our annotation conversion approach can gather high-quality pseudo MWS training/dev datasets, and hence it is feasible to use them for model training and tuning.

(2) Manual MWS data annotation tells us that about 28.4% words are multi-grained, and among them 94.4% are two-grained words.

(3) The parsing and sequence labeling approaches achieve very similar performance, and outperform the SWS aggregation approach by a large margin.

We believe there are many exploration directions for this new task, among which we are particularly interested in three in the near future: 1) improving our benchmark approaches by considering task-specific features and neural network architectures, 2) verifying the usefulness of MWS to high-level applications such as MT, 3) integrating MWS with syntactic parsing in some way by exploiting existing treebanks.

Acknowledgments

The authors would like to thank the anonymous reviewers for the helpful comments. We are greatly grateful to all students in LA group for their hard work as annotators. Especially, we thank Jiawei Sun for her extensive participant and support through this work. We also thank Wenliang Chen and Guodong Zhou for the helpful discussions. This work was supported by National Natural Science Foundation of China (Grant No. 61525205, 61373095, 61502325).

References

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for chinese. In *Proceedings of ACL*, pages 409–420.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of EMNLP*, pages 1197–1206.

Fei Cheng, Kevin Duh, and Yuji Matsumoto. 2015. Synthetic word parsing improves chinese word segmentation. In *ACL*, pages 262–267.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of EMNLP*, pages 1–11.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of ACL*, pages 1012–1020.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 123–133.

Lei Hou, Min Chu, Jingming Tang, Jian Sun, Xiaoling Liao, Rengang Peng, Yang Yang, and Bingjing Xu. 2010. Method and device for providing multi-granularity word segmentation result. Chinese Patent (CN102479191A).

Chang-Ning Huang, Yumei Li, and Xiaodan Zhu. 2006. Tokenization Guidelines of Chinese Text (V5.0, in Chinese). Microsoft Research Asia.

Björn H. Jernudd and Michael J. Shapiro, editors. 1989. *The Politics of Language Purism (page 214)*. Mouton de Gruyter.

Zhenghua Li, Jiayuan Chao, Min Zhang, and Wenliang Chen. 2015. Coupled sequence labeling on heterogeneous annotations: POS tagging as a case study. In *Proceedings of ACL*, pages 1783–1792.

Zhenghua Li, Jiayuan Chao, Min Zhang, and Jiwen Yang. 2016. Fast coupled sequence labeling on heterogeneous annotations via context-aware pruning. In *Proceedings of EMNLP*, pages 753–762.

Zhongguo Li. 2011. Parsing the internal structure of words: A new paradigm for chinese word segmentation. In *Proceedings of ACL*, pages 1405–1414.

Zhongguo Li and Guodong Zhou. 2012. Unified dependency parsing of chinese morphological and syntactic structures. In *Proceedings of EMNLP*, pages 1445–1454.

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, and Ting Liu. 2016. Exploring segment representations for neural segmentation models. In *Proceedings of AAAI*, pages 2880–2886.

Yixuan Liu, Bin Wang, Fan Ding, and Sheng Xu. 2008. Information retrieval oriented word segmentation based on character association strength ranking. In *Proceedings of EMNLP*, pages 1061–1069.

Yuan Liu and Nanyuan Liang. 1986. Fundation of Chinese processing: statistics of modern Chinese word frequencies. *Journal of Chinese Information Processing (in Chinese)*, 0(1):17–25.

Jianqiang Ma and Erhard Hinrichs. 2015. Accurate linear-time chinese word segmentation via embedding matching. In *Proceedings of ACL-IJCNLP*, pages 1733–1743.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of ACL*, pages 293–303.

Richard Sproat, William Gales, Chilin Shih, and Nancy Chang. 1996. A Stochastic Finite-State Word-Segmentation Algorithm for Chinese. *Computational Linguistics*, 22(3).

Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. 2017. Lattic-based recurrent neurla network encoders for neural machine translation. In *Proceedings of AAAI*.

Xinyan Xiao, Yang Liu, YoungSook Hwang, Qun Liu, and Shouxun Lin. 2010. Joint tokenization and translation. In *Proceedings of COLING*, pages 1200–1208.

Chi Xiu. 2013. *The research and implementation of method for domain Chinese word segmentation*. Ph.D. thesis, Beijing University of Technology.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Shiwen Yu, Huiming Duan, Xuefeng Zhu, Bin Swen, and Baobao Chang. 2003. Specification for corpus processing at Peking University: Word segmentation, POS tagging and phonetic notation (In Chinese). *Journal of Chinese Language and Computing*, 13(2):121–158.

Hua-Ping Zhang and Qun Liu. 2002. Model of Chinese words rough segmentation based on n-shortest-paths method. *Journal of Chinese Information Processing (in Chinese)*, 16(5):1–7.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2013. Chinese parsing exploiting characters. In *Proceedings of ACL*, pages 125–134.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-based neural word segmentation. In *Proceedings of ACL*, pages 421–431.

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based tagging for confidence-dependent chinese word segmentation. In *Proceedings of COLING/ACL*, pages 961–968.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of ACL*, pages 840–847.

Hai Zhao and Chunyu Kit. 2007. Effective subsequence-based tagging for Chinese word segmentation. *Journal of Chinese Information Processing (in Chinese)*, 21(5):8–13.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of EMNLP*, pages 647–657.

Jian Zhu and Shan Li. 2008. Method and device for large- and small-grained segmentation of Chinese text. Chinese Patent (CN101246472A).

Appendix A: Collected Annotation Inconsistencies

In MSR

(1) “一日 (one day)” is annotated as “一日 (one day)” or “一 (one)/日 (day)”.

(2) “过多 (too much)” is annotated as “过多 (too much)” or “过 (too)/多 (much)”.

(3) “这个 (this one)” and “这项 (this item)” have the same stucture of “这 (this)” + #, however they are annotated as “这个 (this one)” and “这 (this)/项 (item)” respectively.

(4) “无异于 (the same to)” and “归功于 (owe to)” have the same stucture of # + “于 (to)”, however they are annotated as “无异 (the same)/于 (to)” and “归功于 (owe to)” respectively.

(5) “核武器 (nuclear weapon)” and “核技术 (nuclear technology)” have the same stucture of “核 (nuclear)” + #, however they are annotated as “核武器 (nuclear weapon)” and “核 (nuclear)/技术 (technology)” respectively.

(6) “下一步 (the next step)” and “下一场 (the next game)” have the same stucture of “下一 (the next)” + #, however they are annotated as “下一步 (the next step)” and “下 (next)/一 (one)/场 (game)” respectively.

(7) “副主任 (deputy director)” and “副总统 (vice-president)” have the same stucture of “副 (vice)” + #, however they are annotated as “副 (deputy)/主任 (director)” and “副总统 (vice-president)” respectively.

(8) “工作者 (worker)” and “创始者 (creator)” have the same stucture of # + “者 (-er/or)”, however they are annotated as “工作者 (worker)” and “创始 (create)/者 (-or)” respectively.

(9) “跨世纪 (cross century)” and “跨国界 (cross border)” have the same stucture of “跨” + # (cross + #), however they are annotated as “跨世纪 (cross century)” and “跨 (cross)/国界 (border)”

In PPD

(1) “部长级 (ministerial level)” is annotated as “部长级 (ministerial level)” or “部长 (ministerial)/级 (level)”.

(2) “一日 (one day)” is annotated as “一日 (one day)” or “一 (one)/日 (day)”.

(3) “过多 (too much)” is annotated as “过多 (too much)” or “过 (too)/多 (much)”.

(4) “最大 (biggest)” is annotated as “最大 (biggest)” or “最 (most)/大 (big)”.

(5) “还有 (and also)” is annotated as “还有 (and also)” or “还 (also)/有 (have)”.

(6) “重奖 (reward greatly)” is annotated as “重奖 (reward)” or “重 (reward)/奖 (greatly)”.

(7) “借助于 (by means of)” and “归功于 (owe to)” have the same stucture of # + “于 (to)”, however they are annotated as “借助于 (by means of)” and “归功 (owe)/于 (to)” respectively.

(8) “南斯拉夫联盟 (Yugoslavia Union)” and “南联盟 (Yugoslavia Union)” have the same stucture of # + “联盟 (Union)”, however they are annotated as “南斯拉夫/联盟 (Yugoslavia Union)” and “南联盟 (Yugoslavia Union)” respectively.

category	Chinese example	CTB	PPD	MSR
时间词	上午十一时 (11 a.m.)	上午/十一时	上午/十一时	上午十一时
(temporal word)	今年下半年 (the second half of this year)	今年/下半年	今年/下半年	今年下半年
	80 年代中期 (the mid-1980s)	80 年代/中期	80/年代/中期	80 年代中期
	2000 年 1 月 1 日 (January 1, 2000)	2000 年/1 月/1 日	2000 年/1 月/1 日	2000 年 1 月 1 日
数量词	一个 (one)	一/个	一个	一个
(quantifier)	33 亿元 (3.3 billion yuan)	33 亿/元	33 亿/元	33 亿元
	八十二年 (eighty-two years)	八十二/年	八十二/年	八十二年
	十多个 (more than ten)	十多/个	十/多/个	十多个
团体、机构、组织	欧洲联盟 (European Union)	欧洲/联盟	欧洲/联盟	欧洲联盟
(orgnization)	乒乓球队 (table tennis team)	乒乓球队	乒乓球队	乒乓球/队
	中共中央 (the Central Committee of the Communist Party of China)	中共/中央	中共中央	中共中央
	人事部门 (personnel department)	人事/部门	人事部门	人事/部门
地名	森林公园 (forest park)	森林/公园	森林/公园	森林公园
(placename)	塞尔维亚共和国 (The Republic of Serbia)	塞尔维亚/共和国	塞尔维亚/共和国	塞尔维亚共和国
	中华人民共和国 (The People's Republic of China)	中华/人民/共和国	中华人民共和国	中华人民共和国
代词 + 名词	各国 (each country)	各/国	各国	各国
(pronoun + noun)	每人 (everyone)	每/人	每人	每人
	各单位 (each unit)	各/单位	各/单位	各单位
专名 + 名词	东方人 (oriental)	东方人	东方/人	东方/人
(proper noun + noun)	诺贝尔奖 (Nobel Prize)	诺贝尔奖 or 诺贝尔/奖	诺贝尔奖	诺贝尔/奖
令人 + #	令人满意 (satisfactory)	令人满意 or 令/人/满意	令人满意	令人/满意
(make sb. + #)	令人感动 (touching)	令/人/感动	令人感动	令人/感动
	令人瞩目 (eye-catching)	令/人/瞩目	令人瞩目	令人/瞩目
# + 于	有利于 (beneficial to)	有利/于 or 有利于	有利/于 or 有利于	有利于
(# + to/for)	用于 (use for)	用于 or 用/于	用于	用于
	囿于 (confined to)	囿于	囿于 or 囿/于	囿/于
# + 率	使用率 (utilization rate)	使用率	使用率	使用/率
(# + rate)	通胀率 (inflation rate)	通胀率	通胀率	通/胀/率
	通货膨胀率 (inflation rate)	通货膨胀率	通货膨胀率	通货膨胀/率
	市场占有率 (market share)	市场/占有率	市场占有率	市场占有/率
# + 出	看出 (find out)	看出	看/出	看/出
(# + out)	走出 (go out)	走出	走/出	走出
	拨出 (dial out)	拨出	拨/出	拨/出
跨 + #	跨世纪 (cross-century)	跨世纪 or 跨/世纪	跨/世纪	跨世纪
(cross + #)	跨年度 (go beyond the year)	跨/年度	跨年度	跨年度
	跨国界 (cross border)	跨国界	跨/国界	跨/国界
# + 污染	水污染 (water pollution)	水污染 or 水/污染	水污染	水污染 or 水/污染
(# + pollution)	环境污染 (environmental pollution)	环境/污染	环境/污染	环境污染
# + 工业	轻工业 (light industry)	轻工业 or 轻/工业	轻工业	轻工业
(# + industry)	重工业 (heavy industry)	重工业 or 重/工业	重工业	重工业
	化学工业 (chemical industry)	化学/工业	化学工业	化学工业 or 化学/工业
全 + #	全市 (whole city)	全/市	全市	全/市 or 全市
(whole + #)	全天 (whole day)	全/天 or 全天	全天	全/天
	全省 (whole province)	全/省	全省	全省
# + 法	组织法 (constitutive law)	组织/法	组织/法	组织法
(# + law)	刑事诉讼法 (criminal procedure law)	刑事/诉讼法	刑事诉讼法	刑事/诉讼法 or 刑事诉讼/法
	土地管理法 (land administration law)	土地/管理法	土地管理法	土地/管理/法
# + 后接成分	演唱者 (singer)	演唱者	演唱者	演唱/者
(# + subsequent	金融家 (financier)	金融家	金融家	金融/家
component)	投资商 (investor)	投资商	投资商	投资/商
	丰富性 (richness)	丰富性	丰富性	丰富/性
	商业化 (commercialization)	商业化	商业化	商业/化
	知识型 (knowledge-based)	知识型	知识型	知识/型

Table 7: An incomplete collection of annotation heterogeneity.

(9) “中共中央 (The CPC Central Committee)” and “越共中央 (Vietnamese Communist Party)” have the same stucture of # + “共中央 (the central government)”, however they are annotated as “中共中央 (The CPC Central Committee)” and “越共 (Vietnamese Communist Party)/中央 (central)” respectively.

(10) “下一步 (the next step)” and “下一场 (the next game)” have the same stucture of “下一 (the next)” + #, however they are annotated as “下一步 (the next step)” and “下 (next)/一 (one)/场 (game)” respectively.

(11) “跨年度 (go beyond the year)” and “跨国界 (cross border)” have the same stucture of “跨” + # (cross + #), however they are annotated as “跨 (go beyond)/年度 (year)” and “跨国界 (cross border)” respectively.

In CTB

(1) “重量级 (heavyweight)” is annotated as “重量级 (heavyweight)” or “重量 (heavy)/级 (weight)”.

(2) “一日 (one day)” is annotated as “一日 (one day)” or “一 (one)/日 (day)”.

(3) “再就业 (re-employment)” is annotated as “再就业 (re-employment)” or “再 (once again)/就业 (employment)”.

(4) “野牛 (wild cow)” is annotated as “野牛 (wild cow)” or “野 (wild)/牛 (cow)”.

(5) “最大 (biggest)” is annotated as “最大 (biggest)” or “最 (most)/大 (big)”.

(6) “还有 (and also)” is annotated as “还有 (and also)” or “还 (also)/有 (have)”.

(7) “下一步 (the next step)” is annotated as “下一步 (the next step)” or “下 (next)/一 (one)/步 (step)”.

(8) “副总统 (vice-president)” is annotated as “副总统 (vice-president)” or “副 (vice)/总统 (president)”.

(9) “变得 (change into)” is annotated as “变得 (change into)” or “变 (change) 得 (into)”.

(10) “有利于 (beneficial to)” and “归功于 (owe to)” have the same stucture of # + “于 (to)”, however they are annotated as “有利于 (beneficial to)” and “归功 (owe)/于 (to)” respectively.

(11) “跨年度 (go beyond the year)” and “跨国界 (cross border)” have the same stucture of “跨” + # (cross + #), however they are annotated as “跨 (go beyond)/年度 (year)” and “跨国界 (cross border)” respectively.

Appendix B: See Table 7

Don't Throw Those Morphological Analyzers Away Just Yet: Neural Morphological Disambiguation for Arabic

Nasser Zalmout **Nizar Habash**
Computational Approaches to Modeling Language Lab
New York University Abu Dhabi
United Arab Emirates
{nasser.zalmout,nizar.habash}@nyu.edu

Abstract

This paper presents a model for Arabic morphological disambiguation based on Recurrent Neural Networks (RNN). We train Long Short-Term Memory (LSTM) cells in several configurations and embedding levels to model the various morphological features. Our experiments show that these models outperform state-of-the-art systems without explicit use of feature engineering. However, adding learning features from a morphological analyzer to model the space of possible analyses provides additional improvement. We make use of the resulting morphological models for scoring and ranking the analyses of the morphological analyzer for morphological disambiguation. The results show significant gains in accuracy across several evaluation metrics. Our system results in 4.4% absolute increase over the state-of-the-art in full morphological analysis accuracy (30.6% relative error reduction), and 10.6% (31.5% relative error reduction) for out-of-vocabulary words.

1 Introduction

Recurrent Neural Networks (RNN) in general, and Long Short-Term Memory (LSTM) cells in particular, have been proven very successful for various Natural Language Processing (NLP) tasks, especially those involving sequential data tagging. RNN models can produce near or above state-of-the-art performance with minimal language-specific feature engineering. These models have the capacity of capturing syntactic and semantic features through the lexical word-level embeddings, and subword features through character-level embeddings.

Morphologically rich languages pose many challenges to NLP through their high degree of ambiguity and sparsity. These challenges are exacerbated for languages with limited resources. Morphological analyzers help reduce sparsity by providing several out-of-context morpheme-based analyses for words, but they usually introduce ambiguity by returning multiple analyses for the same surface form. Therefore, the model would require a further step of morphological disambiguation to choose the correct analysis in context.

Morphological modeling involves heavy use of sequential tagging, so using an LSTM-based model would be highly advantageous. LSTM models are also optimal for long-sequence tagging in particular, so such systems should be able to outperform other deep learning models with fixed window-based modeling. Morphological disambiguation is a well studied problem in the literature, but LSTM-based contributions are still relatively scarce. In this paper we use Bidirectional-LSTM (Bi-LSTM) models for morphological tagging and language modeling, and use the results of these models in ranking the analyses of the morphological analyzer. We incorporate various subword and morphological features at different linguistic depths in the tagger, along with both word-based and character-based embeddings.

Our results show significant accuracy gains for all the morphological features we study, and across several evaluation metrics. We compare our system against a strong baseline and a state-of-the-art-system. We achieve 4.4% absolute over the state-of-the-art in full morphological analysis accuracy (30.6% relative error reduction). When evaluated for the out-of-vocabulary (OOV) words alone, the system achieves 10.6% absolute increase (31.5% relative error reduction), and shows significant performance boost across all evaluation metrics.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 704–713
Copenhagen, Denmark, September 7–11, 2017.

2 Linguistic Issues

What distinguishes morphologically rich languages (MRL), like Arabic, from other languages is that their words include more morphemes (such as prefixes and suffixes) representing a number of morphological features, e.g., gender, number, person, mood, as well as attachable clitics. Table 1 shows the set of 16 Arabic morphological features we model As a result, MRLs tend to have more fully inflected words (types) than their poor counterparts. For instance when comparing Modern Standard Arabic (an MRL) with English (not an MRL), the total number of Arabic words in a large corpus is 20% less than the English parallel version of the corpus; however the the total number of unique Arabic types is twice that of English (El Kholy and Habash, 2010).

Feature	Definition
diac	Diacratization
lex	Lemma
pos	Basic part-of-speech tags (34 tags)
gen	Gender
num	Number
cas	Case
stt	State
per	Person
asp	Aspect
mod	Mood
vox	Voice
prc0	Proclitic 0, article proclitic
prc1	Proclitic 1, preposition proclitic
prc2	Proclitic 2, conjunction proclitic
prc3	Proclitic 3, question proclitic
enc0	Enclitic

Table 1: The morphological features we use in the various models. The first two groups are lexical features; and the last two groups are inflectional and clitic features respectively, in addition to the part-of-speech tag.

Furthermore, MRLs have a tendency towards a higher degree of ambiguity, stemming from different interpretations of the same surface morphemes. In Modern Standard Arabic (MSA), this ambiguity is exacerbated by the language's diacritzation-optional orthography-leading a word to have about 12 analyses per word on average (Habash, 2010). These two issues, form richness and form ambiguity, are at the heart of why MRLs are challenging to NLP. Richness of form increases model sparsity, and ambiguity makes disambiguation harder. Table 2 shows an example of the various in-context and out-of-context morphological analyses of the word قيمتها *qymtha*[1] ('its value' among other readings).

A potential solution is to build a morphological analyzer, also known as morphological dictionary, that encodes all the word inflections in the language. A good morphological dictionary should cover all the inflected forms of a word lemma (richness); and return all the possible analyses of a surface word (ambiguity). Finally, both richness and ambiguity are more challenging when an MRL has limited data, and when the data is noisy.

3 Background and Related Work

Deep learning models have recently emerged as a viable approach for several morphological modeling tasks in general. Neural approaches are particularly appealing due to their generic modeling capabilities that can be scaled to multiple tasks, and for less reliance on specific feature engineering. Notable contributions include the work of Collobert et al. (2011), where they present a learning model that is applicable to several NLP tasks, like chunking, named entity recognition, and part-of-speech (POS) tagging, by deliberately avoiding task-specific feature engineering. They use a window-based deep neural network. The fixed window size, however, limits access to further parts of the sentence that might be relevant to the target word. Moreover, the analysis is applied on the surface word level only, without considering any subword features. Several other contributions utilize somewhat similar approaches, with various neural architectures (Wang et al., 2015; Huang et al., 2015). Dos Santos and Zadrozny (2014), on the other hand, argue that subword information is useful for certain NLP tasks, like POS tagging. They propose a character-based embedding along with the word embeddings, to be able to capture internal morphemic structures. Character embeddings, capturing subword features, are well studied in other contributions too (Labeau et al., 2015; Rei et al., 2016; Belinkov and Glass, 2015).

Morphological disambiguation, however, has

[1] All Arabic transliterations are provided in the Habash-Soudi-Buckwalter transliteration scheme (Habash et al., 2007).

واشارت الصحيفة الى ان تاريخ رسم اللّوحة ليس معروفا الا ان **قيمتها** تقدر ب ٢٥ مليون دولار.
wAšArt AlSHyfħ Alý An tAryx rsm AllwHħ lys mErwfA AlA An **qymthA** *tqdr b 25 mlywn dwlAr .*
The newspaper pointed out that the date of the painting is unknown, but **its value** is estimated at 25 million dollars.

diac	lex	gloss	pos	prc3	prc2	prc1	prc0	per	asp	vox	mod	gen	num	stt	cas	enc0
qay∼amatohA	qay∼am	evaluate;assess	verb	0	0	0	0	3	p	a	i	f	s	na	na	3fs:dobj
qay∼amotahA	qay∼am	evaluate;assess	verb	0	0	0	0	2	p	a	i	m	s	na	na	3fs:dobj
qay∼amotihA	qay∼am	evaluate;assess	verb	0	0	0	0	2	p	a	i	f	s	na	na	3fs:dobj
qay∼amotuhA	qay∼am	evaluate;assess	verb	0	0	0	0	1	p	a	i	m	s	na	na	3fs:dobj
qay∼imatahA	qay∼im	caretaker	noun	0	0	0	0	na	na	na	na	f	s	c	a	3fs:poss
qay∼imatihA	qay∼im	caretaker	noun	0	0	0	0	na	na	na	na	f	s	c	g	3fs:poss
qay∼imatuhA	qay∼im	caretaker	noun	0	0	0	0	na	na	na	na	f	s	c	n	3fs:poss
qiymatahA	**qiymaħ**	**value;worth**	**noun**	**0**	**0**	**0**	**0**	**na**	**na**	**na**	**na**	**f**	**s**	**c**	**a**	**3fs:poss**
qiymatihA	qiymaħ	value;worth	noun	0	0	0	0	na	na	na	na	f	s	c	g	3fs:poss
qiymatuhA	qiymaħ	value;worth	noun	0	0	0	0	na	na	na	na	f	s	c	n	3fs:poss
qymthA	qymthA	NOAN	noun_prop	0	0	0	0	na	na	na	na	m	s	i	u	0

Table 2: An example highlighting Arabic's rich morphology and ambiguous orthography. The word قيمتها *qiymatahA* 'its value' has a specific analysis in the context of the sentence shown at the top of the table; but it has many other analyses and diacritizations out of context. The correct analysis is bolded (4th from the bottom of the list).

relatively fewer deep learning contributions. Yildiz et al. (2016) presented a disambiguation model for Turkish based on Convolutional Neural Networks (CNN). Their model creates a representation for the surface form of a word from the root along with a set of morphemic features. Then they train a model to predict the optimal analysis of a word given the annotations within a context window. Shen et al. (2016), on the other hand, use a character-based Bi-LSTM model for morphological disambiguation of morphologically complex languages, without using a morphological analyzer. The LSTM cells have the advantage of capturing a longer sequence window than those of the fixed window and CNN approaches.

Arabic morphological analysis and disambiguation have seen a considerable amount of work, spanning both MSA (Habash and Rambow, 2005; Diab et al., 2004; Khalifa et al., 2016; Abdelali et al., 2016), and Dialectal Arabic (Duh and Kirchhoff, 2005; Al-Sabbagh and Girju, 2012; Habash et al., 2013). The current state-of-the-art system is MADAMIRA (Pasha et al., 2014); which uses SVMs to disambiguate among a target word's various morphological analyses provided by a morphological dictionary.

Neural-based contributions for Arabic, however, are also relatively scarce. Among the contributions that utilize morphological structures to enhance the neural models in different NLP tasks, we note Guzmán et al. (2016) for machine translation, and Abandah et al. (2015) for diacritization. Darwish et al. (2017) use Bi-LSTM models to train a POS tagger, and compare it against SVM-based models. The SVM models in their system outperform the neural model, even with incorporating pre-trained embeddings. Heigold et al. (2016) developed character-based neural models for morphological tagging for 14 different languages, including Arabic, using the UD treebank. Most related to our work though is by Shen et al. (2016), who applied their Bi-LSTM morphological disambiguation model on MSA, but did not present any improvements over the state-of-the-art.

Occurring in parallel to our work, Inoue et al. (2017) used multi-task learning to model fine-grained POS tags, using the individual morphosyntactic features. They also use dictionary information concatenated to the word embeddings, similar to the approach we use in this paper, and use the same dataset. Our approach provides slightly higher accuracy scores for the individual features, but the joint features score in their system is higher.

In this paper we study various architectures for neural based morphological tagging. We then use these architectures, along with neural language modeling systems, to train models for various Arabic morphological features. We utilize these models for morphological disambiguation of the optimal analysis for each given word in context.

4 Approach

The morphological disambiguation task involves choosing the correct morphological analysis from the set of potential analyses, obtained from the analyzer. Towards that end, we train several models for the individual morphological features, and use their results to score and rank the different analyses and choose an optimal overall analysis. These features can be grouped into non-lexical features, where a tagger is used to obtain the relevant morphological tag, or **morphological feature tagging**, and lexical features that need a language model (Roth et al., 2008), or **neural language models**. Table 1 shows the set of morphological features we work with. The lexical features are handled with a language model, while the inflectional, clitic, and part-of-speech features are handled with a tagger.

We use Bi-LSTM-based taggers for the morphological feature tagging tasks, with various embedding levels and morphological features. We investigate the different architectures and design options in detail in Section 5. We then use the best design to build 14 different taggers, each specific to an individual feature. We also use LSTM-based neural language models for the lexical features. We discuss the neural language models in more detail in Section 6.

We then use the results for these various models to score the potential morphological analyses from the analyzer for each given word. These scores are used to rank the analyses and return the one with the highest result. The process of scoring is also tuned through tuning weights for the used features. The details of the ranking and disambiguation process are provided in Section 6.

Dataset: We use the Penn Arabic Treebank (PATB parts 1,2 and 3) (Maamouri et al., 2004) for all the experiments in this paper. We follow the data splits recommend by Diab et al. (2013) for training, dev, and testing sets. We use Alif/Ya and Hamza normalization, and we remove all diacritics. The pre-trained word embeddings are trained using the LDC's Gigaword corpus for MSA (Parker et al., 2011). Table 3 shows the overall data sizes.

Dataset	Size (words)
Train	503,015
Dev	63,137
Test	63,172
Gigaword corpus	2,154M

Table 3: Dataset statistics

Evaluation: We use accuracy as the evaluation metric for all experiments reported in the paper.

Baselines: We use the Maximum Likelihood Estimation (MLE) baseline, calculated by counting the frequency scores for each given word/tag out of context, with backoff to the most frequent tag for unknown words. We also use the MADAMIRA (release-2.1) scores as another baseline, designated as the state-of-the-art system. Unless otherwise specified, MADAMIRA was configured in the ADD_PROP backoff mode, which adds a proper noun analysis to all words. We use this configuration to match the analyzer format we used in training the deep learning system, and to match the models in previous contributions.

5 Neural Morphological Feature Tagging Architectures

The task of morphological tagging in general relies on the context for accurate analysis. Such tasks can be modeled as a sequential data tagging problem, with both word and subword embeddings. While word embeddings are used to convey syntactic and semantic features, subword embeddings convey morphological features.

We present our morphological tagging model in this section, and use the POS feature as a test case. We then generalize our findings for all the other features for the morphological disambiguation process in Section 6. The POS tag set we use is the MADAMIRA tag set presented at (Pasha et al., 2014), and covered in detail at the MADAMIRA manual (Pasha et al., 2013), comprised of 34 tags.

5.1 Deep Learning Model

Given a sentence consisting of N words $\{w_1, w_2, ..., w_N\}$, every word w_i is converted into a vector

$$v_i = [r^{wrd}; r^{morph}]$$

which is composed of the word (or character sequence) embedding vector r^{wrd}, and the morphological features embedding vector r^{morph}. The morphological features vector can be constructed through various constructs, representing morphological and/or subword units.

We then use two LSTM layers to model the relevant context for both directions of the target word, where the input is represented by the v_n vectors mentioned above:

$$\overrightarrow{c}_i = g(v_i, \overrightarrow{c}_{i-1})$$

$$\overleftarrow{c}_i = g(v_i, \overleftarrow{c}_{i+1})$$

We join both sides, apply a non-linearity function, and softmax to get a probability distribution. We use two hidden layers of size 800. Each layer is composed of two LSTM layers for each direction, and a dropout wrapper with keep probability of 0.8, and peephole connections. We use Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.003, and cross-entropy cost function. We use Tensorflow as the development environment.

subword and Morphological Features Several features can be used to represent the r^{morph} vectors mentioned before. These features are utilized to convey morphological information that are not represented at the word-level embeddings. We use various features with various linguistic depth:

(a) Fixed-width affixes We represent the prefixes and suffixes through a fixed character length substring from the beginning and the end of every word. This requires no linguistic information. We use a subset of three characters on both ends.

(b) Language-specific affixes (*lightstemmer*) We use regular expressions to maximally match affix patterns at the word's beginning and end. This requires basic linguistic knowledge of the target language, but doesn't require any large-scale lexical resources or annotated corpora.

(c) Potential POS tags from a morphological dictionary We use a high coverage morphological dictionary to obtain all possible POS tags of the target word. This requires advanced resources/annotations of the language. We include the set of potential tags in a vector representation and concatenate it with the word embedding.

The vector representation of these features is made up of the sum of the one-hot vectors for each individual component.

5.2 Word and Character Embeddings

Using character-level embeddings has recently been proven proficient for various NLP problems. In this paper we also study the effect of using word-level vs character-level embeddings on the overall morphological tagging problem, especially in light of the various subword and morphological features that we utilize. For word-level embeddings, we pre-train the word vectors using Word2Vec (Mikolov et al., 2013) on the Gigaword corpus mentioned in Section 4 (and Table 3), and the text of the training dataset. The embedding dimension for the words is 250. For the character-level embeddings, we concatenate the word embeddings with the sequence of character embeddings, initialized with their one-hot representation.

5.3 Results

Model	**Accuracy**
MLE Baseline	92.5
MADAMIRA (no backoff)	95.9
MADAMIRA (with backoff)	97.0

Table 4: Maximum Likelihood Estimation (MLE) and MADAMIRA baselines for POS tagging.

Model	**Embedding**	
	Word	Char
No Morphology	96.4	96.7
Fixed Character Affixes	96.6	NA
Lightstemmer	96.7	96.8
Morphological Dictionary	97.5	97.5
+ Fixed Character Affixes	**97.6**	NA
+ Lightstemmer	**97.6**	**97.6**

Table 5: Results for word embeddings (Word) and character-level embeddings (Char) for POS tagging. We don't provide character-level embeddings results for the Fixed Character Affixes approach, because such features would be redundant with the character embeddings themselves.

Table 4 shows the baseline scores for the systems, including the results for MADAMIRA with and without backoff. Table 5 shows the results for all systems. The results show clear improvement for all systems over the baseline and state-of-the-art without using subword. In fact, our system with no morphology outperforms MADAMIRA without using backoff. While our best result outperforms both MADAMIRA systems. Affixes (fixed length or lightstemmer) in general increase the accuracy across all systems. We notice, however, that the performance doesn't vary much between the fixed-width and lightstemmer affixes. This proves that the Bi-LSTM model is powerful enough to identify relevant features from a character-stream only, without the need for language-specific affixes. Using the morphological dictionary has the largest effect of improvement across all systems (among the three morphology features used): 0.8% over the next best approach (absolute score difference). We obtain the highest accuracy scores when incorporating both the morphological dictionary tags, and affixes, whether for the fixed-width or the lightstemmer approaches. This system shows 20.0% error reduction over MADAMIRA with backoff, and 41% error reduction without backoff.

The character-level embedding system shows a somewhat similar behavior in terms of relative performance. We do not provide the results for the Fixed Character Affixes approach here since the character embeddings would capture these fixed affixes within the overall embedding vector anyway. Hence, it will only provide redundant representation without any additional information.

We observe that the character-based system, without any additional features, outperforms the word-based system. This is expected, since the system has access to subword features, conveyed in the characters stream, that are not available for the word-based system. The same behavior persists with the lightstemmer, but the performance gap is smaller, since the word-based system is now provided with similar subword features that the character stream conveys. These subword features are somewhat redundant for the character-based system, so the performance is only slightly better.

Surprisingly, however, both systems perform exactly the same when using the morphological dictionary features. This indicates that the morphological features are powerful enough to convey and exceed the subword features that a character stream can convey.

5.4 POS Tagging Error Analysis

We analyzed the resulting tag predictions against the gold tags by transforming the POS tag set space into four categories: nominals, verbs, particles, and punctuation, and observed the resulting error patterns. We noticed that the errors' distribution across all developed systems is somewhat similar throughout the four different categories. Nominals dominate almost 80.0% of all errors, even though they constitute 61.5% only of the total tokens. When introducing the morphological dictionary tags as features, all four categories increase in accuracy (except for the punctuation, being tagged almost correctly at all systems). Verbs, however, have the highest accuracy increase, at 1.5%, relative to 1.0% for nominals and 0.8% for particles. This can be the result of verbs being the least common category in the dataset at 8.0% (vs 64.0%, 14.0%, and 12.0% for nominals, particles, and punctuation, respectively). The nominals set is also relatively bigger than the other categories, which makes it internally confusable with errors within the nominals' options, like noun_adj or noun_num/noun_quant, among others.

6 Morphological Disambiguation

In this section we apply the morphological feature tagging architecture we discussed earlier for POS tagging to the remaining morphological features. We use the results of these taggers, along with the language models for *diac* and *lex*, as the input to the scoring and ranking process.

6.1 Morphological Tagging Models

Section 5 shows that the best performing neural architecture for POS tagging, as an example of morphological tagging in general, is using the embeddings (either character-based or word-based) with the relevant morphological tags from the dictionary, along with fixed or lightstemmer affixes. The performance of both word and character embeddings in this architecture was similar, so we opt for the word embeddings due to the excessive computational overhead affiliated with training character-level embeddings.

We apply the same architecture for the 14 morphological, non-lexical, features we study in this paper. Table 6 shows the results for the different taggers, relative to the MLE and MADAMIRA baselines that we used in the previous section. All features show significant performance boost.

Notable features though include case and state, where good tagging requires a relatively wide analysis window surrounding the target word. These features have the biggest performance gap between the baselines and the Bi-LSTM approach among the various other features. This is mainly due to the fact that LSTM cells have the capability of maintaining a longer sequence memory than the other approaches, hence capturing more of the sentence structure when tagging, compared to traditional window-based approaches.

System	pos	cas	num	gen	vox	mod	stt	asp	per	enc0	prc0	prc1	prc2	prc3
MLE	92.5	80.5	98.3	97.5	97.7	97.4	90.2	97.9	97.9	98.3	97.9	98.5	97.9	99.6
MADAMIRA	97.0	91.1	99.5	99.4	99.1	99.1	97.0	99.3	99.2	99.6	99.6	99.6	99.6	**99.9**
Bi-LSTM	97.6	94.5	99.6	99.5	99.2	99.4	97.9	99.4	99.4	99.7	99.7	99.8	**99.7**	**99.9**
Disambiguated Bi-LSTM	**97.9**	**94.8**	**99.7**	**99.7**	**99.4**	**99.6**	**98.3**	**99.6**	**99.6**	**99.8**	**99.8**	**99.9**	**99.7**	**99.9**
Absolute Increase	0.9	3.7	0.2	0.3	0.3	0.5	1.3	0.3	0.4	0.2	0.2	0.3	0.1	0.0
Error Reduction	30.0	42.0	40.0	50.0	33.0	56.0	43.0	43.0	50.0	50.0	50.0	75.0	25.0	0.0

Table 6: Morphological tagging results. The absolute increase and error reduction are of the disambiguated Bi-LSTM against MADAMIRA.

6.2 Neural Language Models

In addition to the morphological taggers for the non-lexical features, we use neural language models for the lemmatization and diacritization features. Lemmas and diacratized forms are lexical in nature, and cannot be modeled directly using a classifier. We use an LSTM-based neural language model (Enarvi and Kurimo, 2016), with class-based input rather than words. Using a class-based approach speeds convergence drastically and improves the overall perplexity, especially for the *diac* (diacritization) language model, which has a relatively large type count.

We use the MKCLS tool (Och, 1999), through GIZA++ (Och and Ney, 2003), to train the word classes. We use two hidden layers of size 500 and input layer of size 300, and use Nesterov Momentum as the optimization algorithm.

We encode the testing set in the HTK Standard Lattice Format (SLF), with a word mesh representation for the various options of each word.

Table 7 shows the accuracy results of the language models for *lex* and *diac* for both MADAMIRA (which uses SRILM (Stolcke, 2002) for language modeling), and the LSTM model we use here. All models are trained on the same ATB training dataset used in the paper. The LSTM results outperform MADAMIRA's vastly, proving the superiority of neural language models.

Feature	*lex*	*diac*
3-gram model	76.7	68.2
3-gram model disambiguated	96.2	87.7
Our system (LSTM)	89.6	73.5
Our system disambiguated	**96.9**	**91.7**

Table 7: The language model accuracy scores for both MADAMIRA and the LSTM models, for the *lex* and *diac* features.

6.3 Disambiguation

We use a similar morphological disambiguation approach to the model proposed by Habash and Rambow (2005) and Roth et al. (2008), where the resulting morphological features are matched and scored against the morphological analyzer options, as a way to rank the different analyses, and tuned using feature weights. If the analysis and the predicted morphological tag for a feature of a given word match, the analysis score for that analysis is incremented by the weight corresponding to that feature.

The morphological analysis with the highest score is chosen as the disambiguated option. Any tie-breaking after the disambiguation is handled through random selection among the reduced options[2]. For feature weight tuning we use the approach presented by Roth et al. (2008), using the Downhill Simplex Method (Nelder and Mead, 1965). A tuning dataset of almost 2K lines (∼63K words) is randomly selected from the training dataset. We retrain all the systems using the remaining training dataset, to be used in the tuning process. We finally use the resulting optimal weights in the original systems, trained on the full training dataset.

6.4 Evaluation

We use the following accuracy metrics to evaluate the disambiguation model, which Pasha et al. (2014) also use in their evaluation:

- EVALFULL: The percentage of correctly analyzed words across all morphological features. This is the strictest possible metric.
- EVALDIAC: The percentage of words where the chosen analysis has the correct fully diacritized form.

[2] This results in %0.02 variation range only in the EVALFULL end result.

Evalustion Metric	All Words			Out-Of-Vocabulary Words		
	MADAMIRA	Our System	Error Reduction	MADAMIRA	Our System	Error Reduction
EVALFULL	85.6	**90.0**	30.6	66.3	**76.9**	31.5
EVALDIAC	87.7	**91.7**	32.5	70.2	**79.8**	32.8
EVALLEX	96.2	**96.8**	15.8	82.9	**87.8**	28.7
EVALPOS	97.0	**97.9**	30.0	89.9	**96.0**	60.4
EVALATBTOK	99.4	**99.6**	33.3	94.2	**97.8**	60.2

Table 8: Accuracy results of the disambiguation system, evaluated using different metrics, for all words and out-of-vacbulary (OOV) words alone. OOV percentage of all words is 7.9%.

- EVALLEX: The percentage of words where the chosen analysis has the correct lemma.
- EVALPOS: The percentage of words where the chosen analysis has the correct part-of-speech.
- EVALATBTOK: The percentage of words that have a correct ATB tokenization.[3]

Deep learning models, through word embeddings, provide an advantage in terms of the analysis of unseen words. So, in addition to calculating the metrics for all the words in the testing set, we also calculate these metrics for the out-of-vocabulary (OOV) words alone.

Table 8 shows the accuracy scores for MADAMIRA and our system. All evaluation metrics indicate the performance boost of our system relative to MADAMIRA, with significant relative error reduction. The same trend stands for the OOV words, with even higher absolute and relative error reduction scores, especially for EVALLEX, EVALPOS, and EVALATBTOK. This increase in OOV analysis accuracy is the result of modeling the data on a semantic level, with the embeddings and neural networks, instead of pure lexical approach.

6.5 Discussion

We conducted additional data analysis over the test set comparing the performance of our system to MADAMIRA.

Comparative Error Patterns When considering full analyses, we observe that our system still makes some errors in words where MADAMIRA is correct. However, the number of times our system is correct and MADAMIRA is not is over twice as the reverse (MADAMIRA is correct and our system is not). From a manual analysis of a sample of 500 words, we observe the majority of the instances where MADAMIRA was correct and our system failed involved the case features. This is not surprising since case is one the features our system still struggles with although we have made major improvements beyond MADAMIRA. Shahrour et al. (2015) used syntax as an additional model to improve the analysis of case. Our model still improves the accuracy beyond theirs, but this highlights the value of using syntax in future work.

Minority Feature-Value Pairs While we show a lot of improvements across the board above in terms of accuracy, we also observe very large improvements in the performance on some minority feature-value pairs. For example, among the values of the *case* feature, the nominative (cas:n) and accusative (cas:a) appear about 7.0% and 11.0% of all words, respectively, of all the values of case. We improve the F-1 score from 70.4% in MADAMIRA to 84.1% in our system for (cas:n); and we similarly improve the F-1 score from 76.7% in MADAMIRA to 85.5% in our system for (cas:a). We also observe similar improvement in the *mood* feature, with the F-1 score for subjunctive mood (occurring 0.55% of all words) increasing from 76.9% in MADAMIRA to 89.5%.

This great increase was not observed across all features. The F1-score of the passive voice feature-value pair (vox:p) occurring 0.6% of all words (and 7.0% of all verbs) only increased from 70.1% in MADAMIRA to 73.4% in our system. Voice in Arabic is harder to model than mood and case since some verbal constructions can be rather ambiguous even for human readers; for example, the noun phrase الكاتبة التي نشرت مقالتها *AlkAtbħ Alty nšrt mqAlthA* has two readings 'the writer who published her article' (active voice) or 'the writer whose article was published' (passive voice). Case and mood are more likely to be de-

[3] ATB scheme tokenizes all clitics except the +ال *Al* 'the' determiner.

terminable from the context using long and short-distance syntactic clues. In the example above the case of the noun مقالتها *mqAlthA* 'her article' is dependent on the voice reading of the verb, which determines if the noun is the subject or object of the verb. For another example, the F1-score of the 2nd person feature-value pair (per:2) occurring 0.05% of all words (and 0.5% of all verbs) only increased from 29.7% in MADAMIRA to 31.7% in our system. The very low performance in the 2nd person makes sense, since the corpus we used is a news corpus where the 2nd person is hardly ever used. We would expect more training data to help such feature-value pairs.

7 Conclusion and Future Work

In this paper we presented an LSTM-based morphological disambiguation system for Arabic. The system significantly outperforms a state-of-the-art system. Our experiments showed that enriching the input word embedding with additional morphological features increases the morphological tagging accuracy drastically, beyond the capabilities of even character-level embeddings. We also showed that using an LSTM based system provides a significant performance boost for syntax based features, which often require wide context window for accurate tagging.

Future directions include exploring additional deep learning architectures for morphological modeling and disambiguation, especially joint and sequence-to-sequence models. We also intend to further investigate the role of syntax features in morphological disambiguation, and explore additional techniques for more accurate tagging. Finally, we aim at applying our models to Arabic dialects and other languages. We expect that character-level embeddings will have a bigger role in scenarios with noisy input, such as non-standard spontaneous orthography used in social media.

Acknowledgment The first author was supported by the New York University Abu Dhabi Global PhD Student Fellowship program.

References

Gheith A. Abandah, Alex Graves, Balkees Al-Shagoor, Alaa Arabiyat, Fuad Jamour, and Majid Al-Taee. 2015. Automatic diacritization of Arabic text using recurrent neural networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(2):183–197.

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16, San Diego, California.

Rania Al-Sabbagh and Roxana Girju. 2012. A supervised pos tagger for written Arabic social networking corpora. In *Proceedings of KONVENS 2012*, pages 39–52. OGAI.

Yonatan Belinkov and James Glass. 2015. Arabic diacritization with recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2281–2285.

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

Kareem Darwish, Hamdy Mubarak, Ahmed Abdelali, and Mohamed Eldesouki. 2017. Arabic POS tagging: Don't abandon feature engineering just yet. In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 130–137, Valencia, Spain.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652*.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic Tagging of Arabic Text: From Raw Text to Base Phrase Chunks. In *Proceedings of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL04)*, pages 149–152, Boston, MA.

Cícero Nogueira Dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*, pages 1818–1826.

Kevin Duh and Katrin Kirchhoff. 2005. POS tagging of dialectal Arabic: a minimally supervised approach. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, Semitic '05, pages 55–62, Ann Arbor, Michigan.

Ahmed El Kholy and Nizar Habash. 2010. Techniques for Arabic Morphological Detokenization and Orthographic Denormalization. In *Proceedings of the seventh International Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta.

Seppo Enarvi and Mikko Kurimo. 2016. Theanolm - an extensible toolkit for neural network language modeling. *CoRR*, abs/1605.00942.

Francisco Guzmán, Houda Bouamor, Ramy Baly, and Nizar Habash. 2016. Machine translation evaluation for Arabic using morphologically-enriched embeddings. In *Proceedings of COLING 2016, the International Conference on Computational Linguistics*, pages 1398–1408, Osaka, Japan.

Nizar Habash and Owen Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 573–580, Ann Arbor, Michigan.

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013. Morphological analysis and disambiguation for dialectal Arabic. In *Proceedings of NAACL-HLT*, pages 426–432, Atlanta, Georgia.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Y Habash. 2010. *Introduction to Arabic natural language processing*, volume 3. Morgan & Claypool Publishers.

Georg Heigold, Josef van Genabith, and Günter Neumann. 2016. Scaling character-based morphological tagging to fourteen languages. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3895–3902.

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

Go Inoue, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Joint prediction of morphosyntactic categories for fine-grained Arabic part-of-speech tagging exploiting tag dictionary information. In *Proceedings of the 21st SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, Vancouver, Canada.

Salam Khalifa, Nasser Zalmout, and Nizar Habash. 2016. Yamama: Yet another multi-dialect Arabic morphological analyzer. In *Proceedings of the International Conference on Computational Linguistics (COLING): System Demonstrations*, pages 223–227, Osaka, Japan.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Matthieu Labeau, Kevin Löser, Alexandre Allauzen, and Rue John von Neumann. 2015. Non-lexical neural architecture for fine-grained pos tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 232–237.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109, Cairo, Egypt.

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

J.A. Nelder and R. Mead. 1965. A simplex method for function minimization. *The Computer Journal*, 7:308–313.

Franz Josef Och. 1999. An efficient method for determining bilingual word classes. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics*, EACL '99, pages 71–76, Stroudsburg, PA, USA.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*.

Robert Parker, David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. 2011. Arabic Gigaword Fifth Edition. LDC catalog number No. LDC2011T11, ISBN 1-58563-595-2.

Arfath Pasha, Mohamed Al-Badrashiny, Ahmed El Kholy, Ramy Eskander, Mona Diab, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *Proceedings of LREC*, Reykjavik, Iceland.

Arfath Pasha, Mohammad Al-Badrashiny, Mona Diab, Nizar Habash, Manoj Pooleery, and Owen Rambow. 2013. *MADAMIRA v1.0 User Manual*. `http://www.nizarhabash.com/publications/MADAMIRA-UserManual.pdf`.

Marek Rei, Gamal KO Crichton, and Sampo Pyysalo. 2016. Attending to characters in neural sequence labeling models. *arXiv preprint arXiv:1611.04361*.

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In *ACL 2008: The Conference of the Association for Computational Linguistics*, Columbus, Ohio.

Anas Shahrour, Salam Khalifa, and Nizar Habash. 2015. Improving Arabic diacritization through syntactic analysis. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1309–1315, Lisbon, Portugal.

Qinlan Shen, Daniel Clothiaux, Emily Tagtow, Patrick Littell, and Chris Dyer. 2016. The role of context in neural morphological disambiguation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 181–191, Osaka, Japan.

Andreas Stolcke. 2002. SRILM - an Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, volume 2, pages 901–904, Denver, CO.

Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *CoRR*, abs/1510.06168.

Eray Yildiz, Caglar Tirkaz, H. Bahadir Sahin, Mustafa Tolga Eren, and Ozan Sonmez. 2016. A morphology-aware network for morphological disambiguation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2863–2869. AAAI Press.

Paradigm Completion for Derivational Morphology

Ryan Cotterell[϶] Ekaterina Vylomova[fi] Huda Khayrallah[϶] Christo Kirov[϶] David Yarowsky[϶]

[϶]Center for Language and Speech Processing, Johns Hopkins University, USA
[fi]Department of Computing and Information Systems, University of Melbourne, Australia
ryan.cotterell@jhu.edu evylomova@student.unimelb.edu.au
{huda,ckirov,yarowsky}@cs.jhu.edu

Abstract

The generation of complex derived word forms has been an overlooked problem in NLP; we fill this gap by applying neural sequence-to-sequence models to the task. We overview the theoretical motivation for a paradigmatic treatment of derivational morphology, and introduce the task of derivational paradigm completion as a parallel to inflectional paradigm completion. State-of-the-art neural models, adapted from the inflection task, are able to learn a range of derivation patterns, and outperform a non-neural baseline by 16.4%. However, due to semantic, historical, and lexical considerations involved in derivational morphology, future work will be needed to achieve performance parity with inflection-generating systems.

1 Introduction

Unlike inflectional morphology, which produces grammatical variants of the same core lexical item (e.g., *take*↦*takes*), derivational morphology is one of the key processes by which new lemmata are created. For example, the English verb *corrode* can evolve into the noun *corrosion*, the adjective *corrodent*, and numerous other complex derived forms such as *anticorrosive*. Derivational morphology is often highly productive, leading to the ready creation of neologisms such as *Rao-Blackwellize* and *Rao-Blackwellization*, both originating from the Rao-Blackwell theorem. Despite the prevalence of productive derivational morphology, however, there has been little work on its generation. Commonly used derivational resources such as NomBank (Meyers et al., 2004) are still finite. Moreover, the complex phonological and historical changes (e.g., the adjectivization *corrode*↦*corrosive*) and affix selection (e.g., choosing between English deverbal suffixes *-ment* and *-tion*) make generation of derived forms an interesting and challenging problem for NLP.

In this work, we show that viewing derivational morphological processes as paradigmatic may be fruitful for generation. This means that there are a number of well-defined form-function pairs associated with a core word. For example, a typical English verb may have five forms in its inflectional paradigm, corresponding to its base (*take*), past tense (*took*), past participle (*taken*), progressive (*taking*) and third-person singular (*takes*) forms. These forms are related by a consistent set of relations, such as affixation. Similarly, a verb may have several slots in its derivational paradigm: The form *take* has the agentive nominalization *taker*, and the abilitative adjectivization *takable*. Note there are also consistent patterns associated with each derivational slot, e.g., the *-er* suffix regularly produces the agentive.

Exploiting this paradigmatic characterization of derivational morphology allows us to create a statistical model capable of generating derivationally complex forms. We apply state-of-the-art models for inflection generation, which learn mappings from fixed paradigm slots to derived forms. Empirically, we compare results for two models on the new task of derivational paradigm completion: a neural sequence-to-sequence model and a standard non-neural baseline. Our best neural model for derivation achieves 71.7% accuracy, beating the non-neural baseline by 16.4 points. Nevertheless, we note this is about 25 points lower than the equivalent model on the English inflection task (and even 20 points lower than the model's performance on the harder Finnish inflection generation). These results point to additional complications in derivation that require more elaborate models or data annotation to overcome. While inflection generation is

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 714–720
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Semantics	POS	Affix
NEGATION	J→J	*un-*, *in-*, *il-*, *ir-*
ORIGIN	N→J	*-an*, *-ian*, *-ish*, *-ese*
RELATION	N→J	*-ous*, *-ious*, *-eous*
DIMINUTIVE	N→N	*-ette*
REPEAT	V→V	*re-*
PATIENT	V→N	*-ee*
RESULT	V→N	*-ment*, *-ion*, *-tion*, *-tion*, *-al*, *-ure*
AGENT	V→N	*-er*, *-or*, *-ant*, *-ee*
POTENTIAL	V→J	*-able*,*-abil*, *ible*

Table 1: A partial list of derivational transformations in English with corresponding POS changes and semantic labels.

becoming a solved problem (Cotterell et al., 2017), derivation generation is still very much open.

2 Derivational Morphology

The generation of derived forms is structurally similar to the generation of inflectional variants, but presents additional challenges for NLP. Here, we provide linguistic background comparing the two types of morphological processes.

Inflection and Derivation. Inflectional morphology primarily marks semantic features that are necessary for syntax, e.g., gender, tense and aspect. Thus, it follows that in most languages inflection never changes the part of speech of the word and often does not change its basic meaning. The set of inflectional forms for a given lexeme is said to form a paradigm, e.g., the full paradigm for the verb *to take* is ⟨*take*, *taking*, *takes*, *took*, *taken*⟩. Each entry in an inflectional paradigm is termed a slot and is indexed by a syntacto-semantic category, e.g., the PAST form of *take* is *took*. We may reasonably expect that all English verbs—including neologisms—have these five forms.[1] Furthermore, there is typically a fairly regular relationship between a paradigm slot and its form (e.g., add *-s* for the third person singular form). Derivational morphology, on the other hand, often changes the core part of speech of a word and makes more radical changes in meaning. In fact, derivational processes are often subcategorized by the part-of-speech change they engender, e.g., *corrode*↦*corrosion* is a deverbal nominalization.

Derivational Paradigms. Much like inflection, derivational processes may be organized into paradigms, with slots corresponding to more abstract lexico-semantic categories for an associated part of speech (Corbin, 1987; Booij, 2008; Štekauer, 2014). Lieber (2004) presents one of the first theoretical frameworks to enumerate a set of derivational paradigm slots, motivated by previous studies of semantic primitives by Wierzbicka (1988). A partial listing of possible derivational paradigm slots for base English adjectives, nouns, and verbs is given in Table 1. The list contains several productive cases. A key difficulty comes from the the fact that the mapping between semantics and suffixes is not always clean; Lieber (2004) points out the category AGENT could be expressed by the suffix *-er* (as in *runner*) or by *-ee* (as in *escapee*). However, both *-er* and *-ee* may have the PATIENT role; consider *burner* ("a cheap phone intended to be disposed of, i.e. burned") and *employee* ("one being employed"), respectively. We flesh out partial derivational paradigms for several English verbs in Table 2.

Unlike in inflectional paradigms, where we expect most cells to be filled for any given base form, derivational paradigms often contain base-slot combinations that are not semantically compatible, leading to the gaps in Table 2.[2] We also observe increased paradigm irregularity due to some derived forms becoming lexicalized at different points in history, differences in the language from which the base word entered the target language (e.g., English roots of Germanic and Latinate origin behave differently (Bauer, 1983)), as well as other factors that are not obvious from the characters in the base word (e.g., gender or number of the resulting noun).

As an example of how difficult these factors can make derivation, consider the wide variety of potential nominalizations corresponding to the RESULT of a verb, e.g., *-ion*, *-al* and *-ment*, (Jackendoff, 1975). While any particular English verb will almost exclusively employ exactly one of these suffixes (e.g., we have *refuse*↦*refusal* and other candidates **refusion* and **refusement* are illicit),[3] the information required to choose the correct suffix may be both arbitrary or not easily available.

[1] Only a handful of English irregulars distinguish between the past tense and the past participle, e.g., *took* and *taken*, and thus have five *unique* forms in their verbal paradigms; most English verbs have four *unique* forms.

[2] For instance, if suffix *-ee* marks a PATIENT it is semantically not compatible with intransitive verbs, i.e., **sneezee* cannot be derived from intransitive *sneeze*.

[3] Note some forms appear to have multiple nominalizations, e.g., *deport*↦{*deportation*,*deportment*}, but closer inspection shows there is one regular semantic transformation per word sense: *deportation* is eviction, but *deportment* is behavior.

Base	-er/-or	-ee	-ment/-tion	-able/-ible
POS	V↦N	V↦N	V↦N	V↦J
Semantic	AGENT	PATIENT	RESULT	POTENTIAL
animate	*animator*	—	*animation*	*animatable*
attract	*attractor*	*attractee*	*attraction*	*attractable*
—	*aggressor*	*aggressee*	*aggression*	—
employ	*employer*	*employee*	*employment*	*employable*
place	*placer*	—	*placement*	*placeable*
repel	*repeller*	*repelee*	*repellence*	*repellable*
escape	*escapee*	—	—	*escapable*
corrode	*corroder*	—	*corrosion*	*corrosible*
derive	*deriver*	*derivee*	*derivation*	*derivable*

Table 2: Partial derivational paradigm for several English verbs; semantic gaps are indicated with —.

Productivity. There is a general agreement in linguistics that frequently used complex words become part of the lexicon as wholes, while most others are likely to be constructed from constituents (Bauer, 2001; Aronoff and Lindsay, 2014); the latter ones typically follow derivational patterns, or rules, such as adding *-able* to express potential or ability or applying *-ly* to convert adjectives into adverbs. These patterns typically present two essential properties: productivity and restrictedness. Productivity relates to the ability of a pattern to be applied to any novel base form to create a new word, potentially on-the-fly. One example of such a productive transformation is adding *-less* (privative construction), which may attach to almost any noun to form an adjective. Moreover, the resulting form's meaning is compositional and predictable. Many derivational suffixes in English are of this type. On the other hand, some patterns are subject to semantic, pragmatic, morphological or phonological restrictions. Consider the English patient suffix *-ee*, which cannot be attached to a base ending in /i(:)/, e.g., it cannot be attached to the verb *free* to form *freeee*. Restrictedness is closely related to productivity, i.e., highly productive rules are less restricted. A parsimonious model of derivational morphology would describe forms using productive rules when possible, but may store forms with highly restricted patterns directly as full lexical items.

A Note On Terminology. We would like to make a subtle, but important point regarding terminology: the phrase *morphologically rich* in the NLP community almost exclusively refers to *inflectional*, rather than *derivational* morphology. For example, English is labeled as morphologically impoverished, whereas German and Russian are considered morphologically rich, e.g., see the introduction of Tsarfaty et al. (2010). As regards derivation, English is quite complex and even similar in richness to German or Russian as it contains productive formations from two substrata: Germanic and Latinate. From this perspective, English is very much a morphologically rich language. Indeed, a corpus study on the Brown Corpus showed that the *majority* of English words are morphologically complex when derivation is considered (Light, 1996). Note that there many languages that have exhibit neither rich inflection, nor rich derivational morphology, e.g., Chinese, which most commonly employs compounding for word word formation (Chung et al., 2014).

3 Task and Models

We discuss our two systems for derivational paradigm completion and the results they achieve.

3.1 Data

We experiment on English derivational triples extracted from NomBank (Meyers et al., 2004).[4] Each triple consists of a base form, the semantics of the derivation and a corresponding derived form e.g., ⟨*ameliorate*, RESULT, *amelioration*⟩. Note that in this task we do not predict whether a slot exists, merely what form it would take given the base and the slot. In terms of current study, we consider the following derivational types: verb nominalization such as RESULT, AGENT and PATIENT, adverbalization and adjective-noun transformations. We intentionally avoid zero-derivations. We also

[4] There are few resources annotated for derivation in non-English languages, making wider experimentation difficult.

	baseline		*1-best seq2seq*		*10-best seq2seq*
	acc	edit	acc	edit	*acc*
all	55.3%	2.01	71.7%	0.97	84.5%
NOMINAL (J↦N)	23.1%	3.45	35.1%	2.67	70.2%
RESULT (V↦N)	40.0%	2.24	52.9%	1.86	72.6%
AGENT (V↦N)	52.2%	0.94	65.6%	0.78	82.2%
ADVERB (J↦R)	90.0%	0.21	93.3%	0.18	96.5%

Table 3: Results under two metrics (accuracy and Levenshtein distance) comparing the non-neural baseline from the 201 SIGMORPHON shared task and the neural sequence-to-sequence model (both for 1-best and 10-best output).

exclude overly orthographically distant pairs by filtering out those for which the Levenshtein distance exceeds half the sum of their lengths, which appear to be misannotations in NomBank. The final dataset includes 6,029 derivational samples, which we split into train (70%), development (15%), and test (15%).[5] We also note that NomBank annotations are often semantically more coarse-grained.

3.2 Evaluation Metrics

We evaluate on 3 metrics: accuracy, average edit distance, and F_1. Accuracy measures how often system output exactly matches the gold string. Edit distance, by comparison, measures the Levenshtein distance between system output and the gold string. Finally, we calculate affix F_1 scores for individual derivational affixes. E.g., for *-ment* precision is the number of words where the model correctly predicted *-ment* (out of total predictions) and recall is the number of words where the model correctly predicted out of the number of true words.

3.3 Baseline Transducer

We train a simple transducer for each base-to-paradigm slot mapping in the training set, identical to the baseline described in Cotterell et al. (2016). This uses an averaged perceptron classifier to greedily apply an output transformation (substitution, deletion, or insertion) to each input character given the surrounding characters and previous decisions.

3.4 RNN Encoder-Decoder

Following Kann and Schütze (2016) on the morphological inflection task, we use an encoder-decoder gated recurrent neural network (Bahdanau et al., 2015). First, an encoder network encodes a sequence: the concatenation of the characters of the input word and a tag describing the desired transformation—both represented by embeddings. This encoder is bidirectional and consists of two gated RNNs (Cho et al., 2014), one encoding the input in the forward direction, the other one encoding in the backward direction. The output of the two RNNs is the resulting hidden vectors $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$. The hidden state is a concatenation of the forward and backward hidden vectors, i.e., $h_i = [\overrightarrow{h_i}\overleftarrow{h_i}]$.

The decoder also consists of an RNN, but is additionally equipped with an attention mechanism. The latter computes a weight for each of the encoder hidden vectors for each character or subtag, which can be roughly understood as giving a certain importance to each of the inputs. The probability of the target sequence $y = (y_1, \ldots, y_{|y|})$ given the input sequence $x = (x_1, \ldots, x_{|x|})$ is modeled by

$$\begin{aligned} p(y \mid x_1, \ldots, x_{|x|}) &= \prod_{t=1}^{|y|} p(y_t \mid \{y_1, \ldots, y_{t-1}\}, c_t) \\ &= \prod_{t=1}^{|y|} g(y_{t-1}, s_t, c_t), \qquad (1) \end{aligned}$$

where g is a multi-layer perceptron, s_t is the hidden state of the decoder and c_t is the sum of the encoder states h_i, scored by attention weights $\alpha_i(s_{t-1})$ that depend on the decoder state: $c_t = \sum_i \alpha_i(s_{t-1})h_i$.

Input Encoding. We model this problem as a character translation problem, with special encodings for the transformation tags that indicate the type of derivation. For example, we treat the triple: ⟨*ameliorate*, RESULT, *amelioration*⟩ as the source string `a m e l i o r a t e RESULT` and target string `a m e l i o r a t i o n`. This is similar to the encoding in Kann and Schütze (2016).

Training. We use the Nematus toolkit (Sennrich et al., 2017).[6] We exactly follow the recipe in Kann and Schütze (2016), the winning submission on the 2016 SIGMORPHON shared task for inflectional morphology. Accordingly, we use a character embedding size of 300, 100 hidden units in both the encoder and decoder, Adadelta (Zeiler, 2012) with a minibatch size of 20, and a beam size of 12. We train for 300 epochs and select the test model based on the performance on the development set.

4 Experimental Results

Table 3 compares the accuracy of our baseline system with the accuracy of our sequence-to-sequence

[5] The dataset is available at `http://github.com/ryancotterell/derviational-paradigms`.

[6] `https://github.com/rsennrich/nematus/`

neural network using the data splits discussed in §3.1. In all cases, the network outperforms the baseline. While 1-best performance is not nearly as high as that expected from a state-of-the-art inflectional generation system, the key point is that performance significantly increases when considering the 10-best outputs. This suggests that the network is indeed learning the correct set of possible nominalization patterns. However, the information needed to correctly choose among these patterns for a given input is not necessarily available to the network. In particular, the network is only aware of important disambiguating historical (e.g., is the input of Latin or Greek origin) and lexical-semantic (e.g., is the input verb transitive or intransitive) factors to the extent that they are implicitly encoded in the input character sequence. We speculate that making these additional pieces of information directly available as input features will significantly improve 1-best accuracy.

Unfortunately, NomBank does not provide the necessary annotations in most cases. For instance, there is no way to differentiate *actor* and *actress* without gender. It also does not distinguish the semantics of some adjective nominalizations, e.g., *activism* and *activity*. Future work will reannotate NomBank to make these finer-grained distinctions.

Error Analysis. We observe mistakes on less frequent suffixes, e.g., *-age*—we predict **draination* instead of *drainage*. Also, there are several cases where NomBank only lists one available form, e.g., *complexity*, and our model predicts *complexness*. We also see mistakes on irregular adverbs, e.g., we generate *advancely* from *advance*, rather than *in-advance*, as well as in PATIENT nominalizations, e.g., the model produces *containee* in place of *content*—this last distinction is unpredictable.

5 Related Work

Previous work in unsupervised morphological *segmentation* and has implicitly incorporated derivational morphology. Such systems attempt to segment words into all constituent morphs, treating inflectional and derivational affixes as equivalent. The popular Morfessor tool (Creutz and Lagus, 2007) is one example of such an unsupervised segmentation system, but many others exist, e.g., Poon et al. (2009), Narasimhan et al. (2015) *inter alia*. Supervised segmentation and analysis models in the literature can also break down derivationally complex forms into their morphs, provided presegmented and labeled data is available for training (Ruokolainen et al., 2013; Cotterell et al., 2015; Cotterell and Schütze, 2017). Our work, however, builds directly upon recent efforts in the *generation* of inflectional morphology (Durrett and DeNero, 2013; Nicolai et al., 2015; Ahlberg et al., 2015; Rastogi et al., 2016; Faruqui et al., 2016). We differ in that we focus on derivational morphology. In another recent line of work, Vylomova et al. (2017) predict derivationally complex forms using sentential context. Our work differs from their approach in that we attempt to generate derivational forms divorced from the context, but the underlying neural sequence-to-sequence architecture is quite similar.

affix	F_1	affix	F_1	affix	F_1
-ly	1.0	*-ity*	0.54	*-ence*	0.32
-er	0.86	*-ment*	0.45	*-ure*	0.22
-ation	0.78	*-ist*	0.43	*-ee*	0.20
-or	0.59	*-ness*	0.40	*-age*	0.20

Table 4: F_1 for various suffix attachments with the sequence-to-sequence model

6 Conclusion

We have presented a statistical model for the generation of derivationally complex forms, a task that has gone essentially unexplored in the literature. Viewing derivational morphology as paradigmatic, where slots refer to semantic categories, e.g., *corrode*+RESULT↦*corrosion*, we draw upon recent advances in the generation of inflectional morphology. Applying this method works well, achieving an overall accuracy of 71.71%, and beating a non-neural baseline. Performance, however, is lower than on the task of paradigm completion for inflectional morphology, indicating that paradigm completion for derivational morphology is more challenging than its inflectional counterpart.

Acknowledgements

The authors would like to thank Jason Eisner and Colin Wilson for helpful discussions about derivation. The first author acknowledges funding from an NDSEG graduate fellowship. This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-15-C-0113. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 1024–1029, Denver, Colorado. Association for Computational Linguistics.

Mark Aronoff and Mark Lindsay. 2014. Productivity, blocking and lexicalization. *The Oxford Handbook of Derivational Morphology*, pages 67–83.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Laurie Bauer. 1983. *English Word-formation*. Cambridge University Press.

Laurie Bauer. 2001. *Morphological Productivity*, volume 95. Cambridge University Press.

Geert Booij. 2008. Paradigmatic morphology. *La Raison Morphologique. Hommage à la Mémoire de Danielle Corbin*, pages 29–38.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Karen Steffen Chung, Nathan W. Hill, and Sun Jackson T.-S. 2014. Sino-Tibetan. In Rochelle Lieber and Pavol Štekauer, editors, *The Oxford Handbook of Derivational Morphology*, chapter 34, pages 609–650. Oxford University Press, Oxford.

Danielle Corbin. 1987. *Morphologie Dérivationnelle et Structuration du Lexique*, volume 193. Walter de Gruyter.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. The CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, Vancouver, Canada. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22, Berlin, Germany. Association for Computational Linguistics.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015. Labeled morphological segmentation with semi-Markov models. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning (CoNLL)*, pages 164–174, Beijing, China. Association for Computational Linguistics.

Ryan Cotterell and Hinrich Schütze. 2017. Joint semantic synthesis and morphological analysis of the derived word. *Transactions of the Association for Computational Linguistics (TACL)*, 5:147–161.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):3.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 1185–1195, Atlanta, Georgia. Association for Computational Linguistics.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 634–643, San Diego, California. Association for Computational Linguistics.

Ray Jackendoff. 1975. Morphological and semantic regularities in the lexicon. *Language*, pages 639–671.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 555–560, Berlin, Germany. Association for Computational Linguistics.

Rochelle Lieber. 2004. *Morphology and Lexical Semantics*, volume 104. Cambridge University Press.

Marc Light. 1996. *Morphological Cues for Lexical Semantics*. Ph.D. thesis, University of Rochester.

A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young, and R. Grishman. 2004. The nombank project: An interim report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, Massachusetts, USA. Association for Computational Linguistics.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2015. An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics (TACL)*, 3:157–167.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 922–931, Denver, Colorado. Association for Computational Linguistics.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 209–217, Boulder, Colorado. Association for Computational Linguistics.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 623–633, San Diego, California. Association for Computational Linguistics.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2013. Supervised morphological segmentation in a low-resource learning setting using conditional random fields. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning (CoNLL)*, pages 29–37, Sofia, Bulgaria. Association for Computational Linguistics.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain. Association for Computational Linguistics.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kuebler, Yannick Versley, Marie Candito, Jennifer Foster, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing of morphologically rich languages (SPMRL) what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12, Los Angeles, CA, USA. Association for Computational Linguistics.

Pavol Štekauer. 2014. Derivational paradigms. In Rochelle Lieber and Pavol Štekauer, editors, *The Oxford Handbook of Derivational Morphology*, chapter 12, pages 354–369. Oxford University Press, Oxford.

Ekaterina Vylomova, Ryan Cotterell, Timothy Baldwin, and Trevor Cohn. 2017. Context-aware prediction of derivational word-forms. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 118–124, Valencia, Spain. Association for Computational Linguistics.

Anna Wierzbicka. 1988. *The Semantics of Grammar*, volume 18. John Benjamins Publishing.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

A Sub-Character Architecture for Korean Language Processing

Karl Stratos
Toyota Technological Institute at Chicago
stratos@ttic.edu

Abstract

We introduce a novel sub-character architecture that exploits a unique compositional structure of the Korean language. Our method decomposes each character into a small set of primitive phonetic units called jamo letters from which character- and word-level representations are induced. The jamo letters divulge syntactic and semantic information that is difficult to access with conventional character-level units. They greatly alleviate the data sparsity problem, reducing the observation space to 1.6% of the original while increasing accuracy in our experiments. We apply our architecture to dependency parsing and achieve dramatic improvement over strong lexical baselines.

1 Introduction

Korean is generally recognized as a language isolate: that is, it has no apparent genealogical relationship with other languages (Song, 2006; Campbell and Mixco, 2007). A unique feature of the language is that each character is composed of a small, fixed set of basic phonetic units called **jamo** letters. Despite the important role jamo plays in encoding syntactic and semantic information of words, it has been neglected in existing modern Korean processing algorithms. In this paper, we bridge this gap by introducing a novel compositional neural architecture that explicitly leverages the sub-character information.

Specifically, we perform Unicode decomposition on each Korean character to recover its underlying jamo letters and construct character- and word-level representations from these letters. See Figure 1 for an illustration of the decomposition. The decomposition is *deterministic*; this is a crucial departure from previous work that uses language-specific sub-character information such as radical (a graphical component of a Chinese character). The radical structure of a Chinese character does not follow any systematic process, requiring an incomplete dictionary mapping between characters and radicals to take advantage of this information (Sun et al., 2014; Yin et al., 2016). In contrast, our Unicode decomposition does not need any supervision and can extract correct jamo letters for all possible Korean characters.

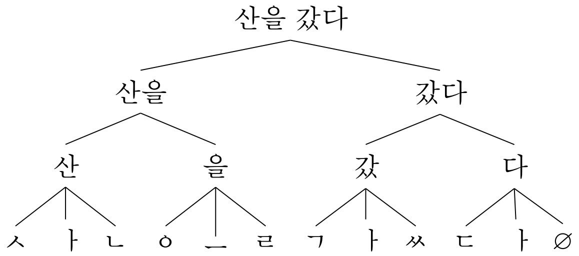


Figure 1: Korean sentence "산을 갔다" (*I went to the mountain*) decomposed to words, characters, and jamos.

Our jamo architecture is fully general and can be plugged in any Korean processing network. For a concrete demonstration of its utility, in this work we focus on dependency parsing. McDonald et al. (2013) note that "Korean emerges as a very clear outlier" in their cross-lingual parsing experiments on the universal treebank, implying a need to tailor a model for this language isolate. Because of the compositional morphology, Korean suffers extreme data sparsity at the word level: 2,703 out of 4,698 word types ($> 57\%$) in the held-out portion of our treebank are OOV. This makes the language challenging for simple lexical parsers even when

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 721–726
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

augmented with a large set of pre-trained word representations.

While such data sparsity can also be alleviated by incorporating more conventional character-level information, we show that incorporating jamo is an effective and economical new approach to combating the sparsity problem for Korean. In experiments, we decisively improve the LAS of the lexical BiLSTM parser of Kiperwasser and Goldberg (2016) from 82.77 to 91.46 while reducing the size of input space by 98.4% when we replace words with jamos. As a point of reference, a strong feature-rich parser using *gold* POS tags obtains 88.61.

To summarize, we make the following contributions.

- To our knowledge, this is the first work that leverages jamo in end-to-end neural Korean processing. To this end, we develop a novel sub-character architecture based on deterministic Unicode decomposition.

- We perform extensive experiments on dependency parsing to verify the utility of the approach. We show clear performance boost with a drastically smaller set of parameters. Our final model outperforms strong baselines by a large margin.

- We release an implementation of our jamo architecture which can be plugged in any Korean processing network.[1]

2 Related Work

We make a few additional remarks on related work to better situate our work. Our work follows the successful line of work on incorporating sub-lexical information to neural models. Various *character*-based architectures have been proposed. For instance, Ma and Hovy (2016) and Kim et al. (2016) use CNNs over characters whereas Lample et al. (2016) and Ballesteros et al. (2015) use bidirectional LSTMs (BiLSTMs). Both approaches have been shown to be profitable; we employ a BiLSTM-based approach.

Many previous works have also considered *morphemes* to augment lexical models (Luong et al., 2013; Botha and Blunsom, 2014; Cotterell et al., 2016). Sub-character models are substantially rarer; an extreme case is considered by Gillick et al. (2016) who process text as a sequence of bytes. We believe that such byte-level models are too general and that there are opportunities to exploit natural sub-character structure for certain languages such as Korean and Chinese.

There exists a line of work on exploiting graphical components of Chinese characters called radicals (Sun et al., 2014; Yin et al., 2016). For instance, 足 (*foot*) is the radical of 跑 (*run*). While related, our work on Korean is distinguished in critical ways and should not be thought of as just an extension to another language. First, as mentioned earlier, the compositional structure is fundamentally different between Chinese and Korean. The mapping between radicals and characters in Chinese is nondeterministic and can only be loosely approximated by an incomplete dictionary. In contrast, the mapping between jamos and Korean characters is deterministic (Section 3.1), allowing for systematic decomposition of all possible Korean characters. Second, the previous work on Chinese radicals was concerned with learning word embeddings. We develop an end-to-end compositional model for a downstream task: parsing.

3 Method

3.1 Jamo Structure of the Korean Language

Let $\mathcal{W}$ denote the set of word types and $\mathcal{C}$ the set of character types. In many languages, $c \in \mathcal{C}$ is the most basic unit that is meaningful. In Korean, each character is further composed of a small fixed set of phonetic units called jamo letters $\mathcal{J}$ where $|\mathcal{J}| = 51$. The jamo letters are categorized as head consonants $\mathcal{J}_h$, vowels $\mathcal{J}_v$, or tail consonants $\mathcal{J}_t$. The composition is completely systematic. Given any character $c \in \mathcal{C}$, there exist $c_h \in \mathcal{J}_h$, $c_v \in \mathcal{J}_v$, and $c_t \in \mathcal{J}_t$ such that their composition yields c. Conversely, any $c_h \in \mathcal{J}_h$, $c_v \in \mathcal{J}_v$, and $c_t \in \mathcal{J}_t$ can be composed to yield a valid character $c \in \mathcal{C}$.

As an example, consider the word 갔다 (`went`). It is composed of two characters, 갔, 다 $\in \mathcal{C}$. Each character is furthermore composed of three jamo letters as follows:

- 갔 $\in \mathcal{C}$ is composed of ㄱ $\in \mathcal{J}_h$, ㅏ $\in \mathcal{J}_v$, and ㅆ $\in \mathcal{J}_t$.

- 다 $\in \mathcal{C}$ is composed of ㄷ $\in \mathcal{J}_h$, ㅏ $\in \mathcal{J}_v$, and an empty letter $\varnothing \in \mathcal{J}_t$.

The tail consonant can be empty; we assume a special symbol $\varnothing \in \mathcal{J}_t$ to denote an empty letter.

[1] https://github.com/karlstratos/koreannet

Figure 1 illustrates the decomposition of a Korean sentence down to jamo letters.

Note that the number of possible characters is combinatorial in the number of jamo letters, loosely upper bounded by $51^3 = 132,651$. This upper bound is loose because certain combinations are invalid. For instance, ㅁ $\in \mathcal{J}_h \cap \mathcal{J}_t$ but ㅁ $\notin \mathcal{J}_v$ whereas ㅏ $\in \mathcal{J}_v$ but ㅏ $\notin \mathcal{J}_h \cup \mathcal{J}_t$.

The combinatorial nature of Korean characters motivates the compositional architecture below. For completeness, we describe the entire forward pass of the transition-based BiLSTM parser of Kiperwasser and Goldberg (2016) that we use in our experiments.

3.2 Jamo Architecture

The parameters associated with the jamo layer are

- Embedding $e^l \in \mathbb{R}^d$ for each letter $l \in \mathcal{J}$
- $U^{\mathcal{J}}, V^{\mathcal{J}}, W^{\mathcal{J}} \in \mathbb{R}^{d \times d}$ and $b^{\mathcal{J}} \in \mathbb{R}^d$

Given a Korean character $c \in \mathcal{C}$, we perform Unicode decomposition (Section 3.3) to recover the underlying jamo letters $c_h, c_v, c_t \in \mathcal{J}$. We compose the letters to induce a representation of c as

$$h^c = \tanh\left(U^{\mathcal{J}} e^{c_h} + V^{\mathcal{J}} e^{c_v} + W^{\mathcal{J}} e^{c_t} + b^{\mathcal{J}}\right)$$

This representation is then concatenated with a character-level lookup embedding, and the result is fed into an LSTM to produce a word representation. We use an LSTM (Hochreiter and Schmidhuber, 1997) simply as a mapping $\phi : \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \to \mathbb{R}^{d_2}$ that takes an input vector x and a state vector h to output a new state vector $h' = \phi(x, h)$. The parameters associated with this layer are

- Embedding $e^c \in \mathbb{R}^{d'}$ for each $c \in \mathcal{C}$
- Forward LSTM $\phi^f : \mathbb{R}^{d+d'} \times \mathbb{R}^d \to \mathbb{R}^d$
- Backward LSTM $\phi^b : \mathbb{R}^{d+d'} \times \mathbb{R}^d \to \mathbb{R}^d$
- $U^{\mathcal{C}} \in \mathbb{R}^{d \times 2d}$ and $b^{\mathcal{C}} \in \mathbb{R}^d$

Given a word $w \in \mathcal{W}$ and its character sequence $c_1 \ldots c_m \in \mathcal{C}$, we compute

$$f_i^c = \phi^f\left(\begin{bmatrix} h^{c_i} \\ e^{c_i} \end{bmatrix}, f_{i-1}^c\right) \qquad \forall i = 1 \ldots m$$
$$b_i^c = \phi^b\left(\begin{bmatrix} h^{c_i} \\ e^{c_i} \end{bmatrix}, b_{i+1}^c\right) \qquad \forall i = m \ldots 1$$

and induce a representation of w as

$$h^w = \tanh\left(U^{\mathcal{C}} \begin{bmatrix} f_m^c \\ b_1^c \end{bmatrix} + b^{\mathcal{C}}\right)$$

Lastly, this representation is concatenated with a word-level lookup embedding (which can be initialized with pre-trained word embeddings), and the result is fed into a BiLSTM network. The parameters associated with this layer are

- Embedding $e^w \in \mathbb{R}^{d_{\mathcal{W}}}$ for each $w \in \mathcal{W}$
- Two-layer BiLSTM Φ that maps $h_1 \ldots h_n \in \mathbb{R}^{d+d_{\mathcal{W}}}$ to $z_1 \ldots z_n \in \mathbb{R}^{d^*}$
- Feedforward for predicting transitions

Given a sentence $w_1 \ldots w_n \in \mathcal{W}$, the final d^*-dimensional word representations are given by

$$(z_1 \ldots z_n) = \Phi\left(\begin{bmatrix} h^{w_1} \\ e^{w_1} \end{bmatrix} \ldots \begin{bmatrix} h^{w_n} \\ e^{w_n} \end{bmatrix}\right)$$

The parser then uses the feedforward network to greedily predict transitions based on words that are active in the system. The model is trained end-to-end by optimizing a max-margin objective. Since this part is not a contribution of this paper, we refer to Kiperwasser and Goldberg (2016) for details.

By setting the embedding dimension of jamos d, characters d', or words $d_{\mathcal{W}}$ to zero, we can configure the network to use any combination of these units. We report these experiments in Section 4.

3.3 Unicode Decomposition

Our architecture requires dynamically extracting jamo letters given any Korean character. This is achieved by simple Unicode manipulation. For any Korean character $c \in C$ with Unicode value $U(c)$, let $\overline{U}(c) = U(c) - 44032$ and $T(c) = \overline{U}(c) \bmod 28$. Then the Unicode values $U(c_h)$, $U(c_v)$, and $U(c_t)$ corresponding to the head consonant, vowel, and tail consonant are obtained by

$$U(c_h) = 1 + \left\lfloor \frac{\overline{U}(c)}{588} \right\rfloor + \texttt{0x10ff}$$
$$U(c_v) = 1 + \left\lfloor \frac{(\overline{U}(c) - T(c)) \bmod 588}{28} \right\rfloor + \texttt{0x1160}$$
$$U(c_t) = 1 + T(c) + \texttt{0x11a7}$$

where c_t is set to $\varnothing$ if $T(c_t) = 0$.

	Training	Development	Test
# projective trees	5,425	603	299
# non-projective trees	12	0	0

	#	# Ko	Examples
word	31,060	–	프로그램보다 갈비 booz
char	1,772	1,315	최 귤 흠 냥 셧 캐 쪽 @ 正 a
jamo	500	48	ㄱ ㄳ ㄼ ㅏ ㅠ ㅢ @ 正 a

Table 1: Treebank statistics. Upper: Number of trees in the split. Lower: Number of unit types in the training portion. For simplicity, we include non-Korean symbols (e.g., @, 正, a) as characters/jamos.

3.4 Why Use Jamo Letters?

The most obvious benefit of using jamo letters is alleviating data sparsity by flattening the combinatorial space of Korean characters. We discuss some additional explicit benefits. First, jamo letters often indicate syntactic properties of words. For example, a tail consonant ㅆ strongly implies that the word is a past tense verb as in 갔다 (went), 왔다 (came), and 했다 (did). Thus a jamo-level model can identify unseen verbs more effectively than word- or character-level models. Second, jamo letters dictate the sound of a character. For example, 갔 is pronounced as got because the head consonant ㄱ is associated with the sound g, the vowel ㅏ with o, and the tail consonant ㅆ with t. This is clearly critical for speech recognition/synthesis and indeed has been investigated in the speech community (Lee et al., 1994; Sakti et al., 2010). While speech processing is not our focus, the phonetic signals can capture useful lexical correlation (e.g., for onomatopoeic words).

4 Experiments

Data We use the publicly available Korean treebank in the universal treebank version 2.0 (McDonald et al., 2013).[2] The dataset comes with a train/development/test split; data statistics are shown in Table 1. Since the test portion is significantly smaller than the dev portion, we report performance on both.

As expected, we observe severe data sparsity with words: 24,814 out of 31,060 elements in the vocabulary appear only *once* in the training data. On the dev set, about 57% word types and 3% character types are OOV. Upon Unicode decomposition, we obtain the following 48 jamo types:

ㄱ ㄳ ㄲ ㄵ ㄴ ㄷ ㄶ ㄹ ㄸ ㄻ ㄺ ㄼ ㅁ
ㅀ ㅃ ㅂ ㅅ ㅄ ㅇ ㅆ ㅉ ㅈ ㅋ ㅊ ㅍ ㅌ
ㅏ ㅎ ㅑ ㅐ ㅓ ㅒ ㅕ ㅔ ㅗ ㅖ ㅙ ㅘ ㅛ ㅚ
ㅝ ㅜ ㅟ ㅞ ㅡ ㅠ ㅣ ㅢ

none of which is OOV in the dev set.

Implementation and baselines We implement our jamo architecture using the DyNet library (Neubig et al., 2017) and plug it into the BiLSTM parser of Kiperwasser and Goldberg (2016).[3] For Korean syllable manipulation, we use the freely available toolkit by Joshua Dong.[4] We train the parser for 30 epochs and use the dev portion for model selection. We compare our approach to the following baselines:

- McDonald13: A cross-lingual parser originally reported in McDonald et al. (2013).
- Yara: A beam-search transition-based parser of Rasooli and Tetreault (2015) based on the rich non-local features in Zhang and Nivre (2011). We use beam width 64. We use 5-fold jackknifing on the training portion to provide POS tag features. We also report on using *gold* POS tags.
- K&G16: The basic BiLSTM parser of Kiperwasser and Goldberg (2016) without the sublexical architecture introduced in this work.
- Stack LSTM: A greedy transition-based parser based on stack LSTM representations. Dyer15 denotes the word-level variant (Dyer et al., 2015). Ballesteros15 denotes the character-level variant (Ballesteros et al., 2015).

For pre-trained word embeddings, we apply the spectral algorithm of Stratos et al. (2015) on a 2015 Korean Wikipedia dump to induce 285,933 embeddings of dimension 100.

Parsing accuracy Table 2 shows the main result. The baseline test LAS of the original cross-lingual parser of McDonald13 is 55.85. Yara achieves 85.17 with predicted POS tags and 88.61 with gold POS tags. The basic BiLSTM model of K&G16 obtains 82.77 with pre-trained word embeddings (78.95 without). The stack LSTM parser is comparable to K&G16 at the word level

[2] https://github.com/ryanmcd/uni-dep-tb

[3] https://github.com/elikip/bist-parser

[4] https://github.com/JDongian/python-jamo

System	Features	Feature Representation	Emb	POS	Dev		Test	
					UAS	LAS	UAS	LAS
McDonald13	cross-lingual features	large sparse matrix	–	PRED	–	–	71.22	55.85
Yara (beam 64)	features in Z&N11	large sparse matrix	–	PRED	76.31	62.83	91.19	85.17
				GOLD	79.08	68.85	92.93	88.61
K&G16	word	31060 × 100 matrix	–	–	68.87	48.25	88.61	78.95
		298115 × 100 matrix	YES		76.30	60.88	90.00	82.77
Dyer15	word, transition	31067 × 100 matrix	–	–	69.40	48.46	88.41	78.22
		298122 × 100 matrix	YES		75.99	59.38	90.73	83.89
Ballesteros15	char, transition	1779 × 100 matrix	–	–	84.22	76.41	91.27	86.25
KoreanNet	char	1772 × 100 matrix	–	–	84.76	76.95	94.75	90.81
		1772 × 200 matrix			84.83	77.29	94.55	91.04
	jamo	500 × 100 matrix	–	–	84.27	76.07	94.59	90.77
		500 × 200 matrix			84.68	77.27	94.86	91.46
	char, jamo	2272 × 100 matrix	–	–	85.35	78.18	94.79	91.19
		2272 × 200 matrix			85.74	78.76	94.55	91.31
	word, char, jamo	302339 × 200 matrix	YES	–	**86.39**	**79.68**	**95.17**	**92.31**

Table 2: Main result. Upper: Accuracy with baseline models. Lower: Accuracy with different configurations of our parser network (word-only is identical to K&G16).

(Dyer15), but it performs significantly better at the character level (Ballesteros15) reaching 86.25 test LAS.

We observe decisive improvement when we incorporate sub-lexical information into the parser of K&G16. In fact, a *strictly* sub-lexical parser using only jamos or characters clearly outperforms its lexical counterpart despite the fact that the model is drastically smaller (e.g., 90.77 with 500 × 100 jamo embeddings vs 82.77 with 298115 × 100 word embeddings). Notably, jamos alone achieve 91.46 which is not far behind the best result 92.31 obtained by using word, character, and jamo units in conjunction. This demonstrates that our compositional architecture learns to build effective representations of Korean characters and words for parsing from a minuscule set of jamo letters.

5 Discussion of Future Work

We have presented a natural sub-character architecture to model the unique compositional orthography of the Korean language. The architecture induces word-/sentence-level representations from a small set of phonetic units called jamo letters. This is enabled by efficient and deterministic Unicode decomposition of characters.

We have focused on dependency parsing to demonstrate the utility of our approach as an economical and effective way to combat data sparsity. However, we believe that the true benefit of this architecture will be more evident in speech processing as jamo letters are definitions of sound in the language. Another potentially interesting application is informal text on the internet. Ill-formed words such as ㅎㅎㅎ (shorthand for 하하하, an onomatopoeic expression of laughter) and ㄴㄴ (shorthand for 노노, a transcription of `no no`) are omnipresent in social media. The jamo architecture can be useful in this scenario, for instance by correlating ㅎㅎㅎ and 하하하 which might otherwise be treated as independent.

Acknowledgments

The author would like to thank Lingpeng Kong for his help with using the DyNet library, Mohammad Rasooli for his help with using the Yara parser, and Karen Livescu for helpful comments.

References

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proc. EMNLP*.

Jan A Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *ICML*, pages 1899–1907.

Lyle Campbell and Mauricio J Mixco. 2007. *A glossary of historical linguistics*. Edinburgh University Press.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1651–1660.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. ACL*.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of NAACL*.

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

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL*.

Geunbae Lee, Jong-Hyeok Lee, and Kyunghee Kim. 1994. Phonemie-level, speech and natural, language integration for agglutinative languages. *GGGGGGGG 0*.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Ryan T McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *ACL (2)*, pages 92–97.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *CoRR*, abs/1503.06733.

Sakriani Sakti, Andrew Finch, Ryosuke Isotani, Hisashi Kawai, and Satoshi Nakamura. 2010. Korean pronunciation variation modeling with probabilistic bayesian networks. In *Universal Communication Symposium (IUCS), 2010 4th International*, pages 52–57. IEEE.

Jae Jung Song. 2006. *The Korean language: Structure, use and context*. Routledge.

Karl Stratos, Michael Collins, and Daniel Hsu. 2015. Model-based word embeddings from decompositions of count matrices. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1282–1291, Beijing, China. Association for Computational Linguistics.

Yaming Sun, Lei Lin, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2014. Radical-enhanced chinese character embedding. In *International Conference on Neural Information Processing*, pages 279–286. Springer.

Rongchao Yin, Quan Wang, Rui Li, Peng Li, and Bin Wang. 2016. Multi-granularity chinese word embedding. In *Proceedings of the Empiricial Methods in Natural Language Processing*.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 188–193. Association for Computational Linguistics.

Do LSTMs really work so well for PoS tagging? – A replication study

Tobias Horsmann and Torsten Zesch
Language Technology Lab
Department of Computer Science and Applied Cognitive Science
University of Duisburg-Essen, Germany
{tobias.horsmann,torsten.zesch}@uni-due.de

Abstract

A recent study by Plank et al. (2016) found that LSTM-based PoS taggers considerably improve over the current state-of-the-art when evaluated on the corpora of the Universal Dependencies project that use a *coarse-grained* tagset. We replicate this study using a fresh collection of 27 corpora of 21 languages that are annotated with *fine-grained* tagsets of varying size. Our replication confirms the result in general, and we additionally find that the advantage of LSTMs is even bigger for larger tagsets. However, we also find that for the very large tagsets of morphologically rich languages, hand-crafted morphological lexicons are still necessary to reach state-of-the-art performance.

1 Introduction

Part-of-Speech (PoS) tagging is an important processing step for many NLP applications. When researchers want to use a PoS tagger, they would ideally choose an off-the-shelf PoS tagger which is optimized for a specific language. If a suited tagger is not available two options remain: a) implementation of your own tagger, which requires technical knowledge and experience, or b) using an existing tagger and hope that the resulting model will be sufficiently accurate. One can assume that many taggers fit more languages than the one for which they have been constructed originally. Ideally, researchers should be able to fall back to a well-evaluated language-independent tagger if no reference implementation for a language is available.

A recent study by Plank et al. (2016) evaluated an LSTM PoS tagger and compared the results to Conditional Random Fields (CRF) (Lafferty et al., 2001) and Hidden-Markov (HMM) implementations on corpora of various languages. Their evaluation concludes that the LSTM tagger reaches better results than the CRF and HMM tagger. The evaluation corpora were all annotated with a *coarse-grained* tagset with 17 tags. Thus, this LSTM tagger seems to be a well-performing, language-independent choice for learning models on coarse-grained tagsets. While for many tasks a coarse-grained tagset might be sufficient some tasks require more fine-grained tagsets.

We, thus, consider it worthwhile to explore if the results are reproducible using corpora with fine-grained tagsets. We use the LSTM tagger provided by Plank et al. (2016) and compare the results likewise to CRF and an off-the-shelf HMM tagger implementation. We compile a fresh set of 27 corpora of 21 languages which uses the commonly used *fine-grained* tagset of the respective language. We suggest these corpora as evaluation set for tasks which require fine-grained PoS tags, as all corpora are freely available for research purposes. Our intention is to replicate the findings of Plank et al. (2016), which have been achieved on a coarse-grained tagset and investigate if they transfer to fine-grained tagsets.

2 PoS Tagger Paradigms

We distinguish two PoS tagger paradigms, which can be used to implement a tagger: The first one is *Feature Engineering*, in which a classifier learns a mapping from human-defined features to a PoS tag. Defining good features is often a non-trivial task, which furthermore requires a lot of experience. For instance a suffix feature which checks a word-ending for “ing” is highly discriminative for English gerunds, but might not provide any useful information for other languages. The details of the feature implementation might render a

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 727–736
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

			Tokens			
Group	**Corpus Id**	**Source**	(10^3)	**# Tags**	**Annotation**	**Reference**
Germanic	Danish	Copenhagen DTB	255	36	manual	(Buch-Kromann and Korzen, 2010)
	Dutch	Alpino	200	20	manual	(Bouma et al., 2000)
	English	Brown	1,100	180	manual	(Nelson Francis and Kuçera, 1964)
	German-1	Hamburg DTB	4,800	54	manual	(Brants et al., 2004)
	German-2	Tiger	880	54	manual	(Telljohann et al., 2004)
	German-3	Tüba-D/Z	1,500	54	manual	(Foth et al., 2014)
	Icelandic	Mim	1,000	703	auto	(Helgadóttir et al., 2012)
	Norwegian	Norwegian DTB	1,300	19	manual	(Solberg et al., 2014)
	Swedish-1	Talbanken	96	25	manual	(Einarsson, 1976)
	Swedish-2	Stockholm-Umea	1,100	153	manual	(Ejerhed and Källgren, 1997)
Romanic	Braz.Portuguese	MAC-Morpho	1,000	82	manual	(Aluísio et al., 2003)
	French-1	Multitag	370	992	manual	(Paroubek, 2000)
	French-2	Sequoia	200	29	manual	(Candito et al., 2014)
	Italian	Turin Parallel	80	15	auto	(Bosco et al., 2012)
	Spanish	IULA DTB	550	241	manual	(Marimon et al., 2014)
Slavic	Croatian-1	Croatian DTB	200	692	manual	(Željko Agić and Ljubešić, 2014)
	Croatian-2	Hr500k	500	769	manual	(Ljubešić et al., 2016)
	Czech	Prague DTB	2,000	1,574	manual	(Bejček et al., 2013)
	Polish	Polish National Corpus	1,000	27	manual	(Przepiórkowski et al., 2008)
	Russian	Russian Open Corpus	1,700	22	manual	(Bocharov et al., 2013)
	Slovak	MULTEXT-East	84	956	manual	(Erjavec, 2010)
	Slovene-1	IJS-ELAN	540	1,181	auto	(Erjavec, 2002)
	Slovene-2	SSJ	590	1,304	manual	(Krek et al., 2013)
Others	Afrikaans	AfriBooms	50	12	manual	(Augustinus et al., 2016)
	Finnish	FinnTreebank	170	1573	manual	(Voutilainen, 2011)
	Hebrew	HaAretz Corpus	11,000	22	auto	(Itai and Wintner, 2008)
	Hungarian	The Szeged Treebank	1,200	1,085	manual	(Csendes et al., 2005)

Table 1: Corpora used in our experiments

tagger unsuited for learning models for other languages or tagsets. We will, thus, experiment with features and their configurations, and investigate how well they perform in combination for learning fine-grained tagsets of various languages. We implement those experiments using CRF which are frequently used for PoS tagging (Remus et al., 2016; Ljubešić et al., 2016).

The second paradigm is *Architecture Engineering*, which relies on methods to learn the input representation by themselves. The challenge lies in finding an architecture that supports this self-learning process. Most recent representatives of this paradigm are neural networks of which we use the LSTM tagger provided by Plank et al. (2016).

In our experiments, we will focus on how to provide word- and character-level information to the classifiers as these two types of information are most relevant and most frequently used for training PoS tagger models. Furthermore, we will evaluate the performance on Out-Of-Vocabulary (OOV) words to learn if the taggers generalize to unseen words.

To provide a reference value to a well-known PoS tagger, we will compare all results to the HMM-based HunPos (Halácsy et al., 2007) tagger, which is a freely available re-implementation of the TNT tagger (Brants, 2000). HunPos has been used before for training models of various languages and tagsets (Seraji, 2011; Attardi et al., 2010; Hládek et al., 2012) which is why we consider this tagger to be a suitable baseline.

3 Evaluation Corpora Dataset

Table 1 shows the fine-grained annotated corpora we collected by screening the literature. We do not claim that this list is complete, but the provided corpora are all reasonably easy to access and can be freely used for research purposes.

Selection To ensure reproducibility, we preferably selected corpora which are directly available via the Internet except *German-3, Hungarian* and *Swedish-2*. We intentionally exclude languages such as Chinese or Japanese, which do not provide whitespace delimiters to mark word boundaries. Tagging those languages requires a morpho-

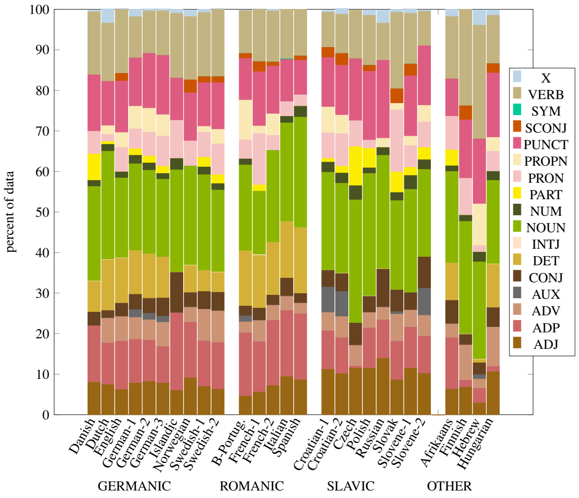


Figure 1: Coarse-grained PoS tag distribution of corpora by language group

logical analysis which is a different task than the tagging task on which we are focusing here. Most corpora are manually annotated or were at least human-verified. There are four exceptions which we decided to add anyway to increase the number of languages represented in our setup. The tagset granularity of the corpora ranges from coarse (12 tags) to morphologically fine (1574 tags) to evaluate all taggers on various stages of granularity.

Language & Corpora Diversity We analyzed the distribution of PoS tags in the corpora by mapping all tags to the 17 coarse-grained PoS tags of the Universal Dependencies (UD) project (Nivre et al., 2015) in Figure 1. The mappings to the UD tagset have been manually created. The partly large differences between the syntactical classes help to better understand the challenge in construction a tagger that is suited for all those languages. For instance, Germanic and Romanic languages have a lot of determiners while they do not occur at all in Slavic languages.

Corpus Size & Tagset The corpora have varying sizes which makes a direct comparison between corpora difficult. To run our experiments under fully controlled conditions, we extract a randomized sub-sample of sentences from each corpus, which accounts for 50k tokens, and run all our experiments with 10fold cross-validation (CV).[1] Results reported use the fine-grained tagset of the respective corpus.

We deliberately do not use the corpora from the UD Treebank project in order to provide results on a fresh dataset. Additionally, UD uses a coarse-grained tagset for all its corpora. While this granularity is sufficient for many tasks, linguistic analysis often requires more fine-grained tagsets, and it is not clear whether results achieved on coarse-grained tagsets transfer well to more fine-grained tagsets. The collected corpora, thus, also represent an alternative dataset, which we suggest in case the UD tagset is too coarse-grained.

4 CRF Experiments

We reviewed the recent literature to determine the most commonly used features for training PoS taggers. As re-occurring features, we found word ngrams, fixed character sequences focusing on either pre-, in-, or suffixes of words and word distributional knowledge for PoS taggers of various languages (Brants, 2000; Horsmann and Zesch, 2016; Ljubešić et al., 2016). Word- and character-ngrams have been used with various parametrizations depending on the language and there is no agreement which parameters are most advisable. We will, hence, run a series of parameter-search experiments over the word- and character-ngram parametrization to determine a configuration applicable to all languages. For this, we evaluate all permutations of the subsequently introduced feature configurations with 10fold cross-validation. The objective is to find a configuration that works well on all corpora, languages, and tagsets.

Word Features We experiment with adding the $1,2,3$ words to the right and left of the current word as lower-cased string features.

Character Features Which character-ngram is discriminative for a PoS tag strongly depends on the language. To avoid a language bias, we use a frequency-based approach in which we select the N most frequently occurring character-ngrams of length $1,2,3,4$ from the training dataset. We experiment with the following frequency cut-off values of $N \ \varepsilon \ \{250, 500, 750, 1000\}$ to select only frequent and potentially informative character-ngrams as features. These N features are boolean and are set to 1 if the respective character-ngram occurs in the current word.

Semantic Features We use Brown clustering (Brown et al., 1992) to create word clusters. The

[1] While randomization prohibits exact reproducibility, it is no barrier to the more interesting replicability. It is also less prone to continued overfitting on the known test set.

		Word		Top 750							
Lang.		Ngrams ±1		Char Ngrams		Clusters		Best CRF		HunPos	
Group	Corpus Id	All	OOV	All	OOV	All	OOV	All	OOV	All	OOV
Germanic	Danish	90.9	53.3	90.3	69.3	89.5	67.6	96.1	82.4	94.9	74.2
	Dutch	86.5	66.9	85.0	71.7	88.0	77.7	90.7	83.7	89.9	80.6
	English	87.5	45.1	90.3	70.1	89.1	64.0	94.6	80.2	93.8	77.7
	German-1	88.5	62.4	90.3	77.7	90.8	73.7	94.6	84.6	94.4	83.7
	German-2	87.2	60.3	90.9	77.7	90.8	76.1	95.2	87.1	94.9	85.4
	German-3	86.3	58.5	91.7	76.8	91.6	77.6	94.4	85.0	94.4	83.9
	Icelandic	67.5	14.2	76.5	45.1	68.3	28.9	80.9	53.6	79.8	51.9
	Norwegian	92.4	77.1	91.6	80.6	92.8	82.7	96.1	89.7	95.5	86.5
	Swedish-1	91.1	70.6	92.9	82.2	92.3	79.9	96.3	90.3	95.6	85.9
	Swedish-2	78.7	29.7	87.2	67.3	81.4	48.8	91.0	74.6	91.4	77.6
Romanic	B-Portug.	86.9	62.8	87.8	73.6	89.7	76.0	92.8	83.8	93.3	84.2
	French-1	81.9	40.1	85.9	66.5	81.6	58.2	89.2	75.7	88.2	71.8
	French-2	95.4	67.3	93.8	74.5	91.9	79.3	97.7	88.2	97.4	82.4
	Italian	93.3	68.6	91.6	74.8	91.7	75.5	96.4	86.5	95.8	80.8
	Spanish	88.5	45.5	94.5	78.2	88.1	58.8	96.4	83.5	96.6	83.6
Slavic	Croatian-1	69.0	18.6	80.6	56.3	75.2	47.2	84.9	65.4	84.7	66.7
	Croatian-2	66.3	15.9	78.5	54.4	73.5	44.8	83.4	63.9	82.6	63.9
	Czech	64.1	14.4	79.2	56.0	75.2	39.2	83.1	62.9	81.7	60.9
	Polish	82.9	58.1	92.5	86.9	86.5	72.5	95.5	91.5	93.6	85.4
	Russian	83.7	53.7	93.0	83.5	88.2	70.9	95.5	87.5	94.6	83.6
	Slovak	67.7	14.9	80.5	57.8	65.6	31.9	83.5	63.8	82.9	61.6
	Slovene-1	72.6	17.4	83.5	55.6	72.4	39.4	86.4	62.5	82.6	59.6
	Slovene-2	65.4	12.1	78.2	50.5	73.0	39.0	83.0	59.4	86.2	59.5
Other	Afrikaans	95.7	75.0	95.3	80.3	95.8	81.9	97.8	89.6	97.3	85.5
	Finnish	62.6	10.0	77.1	48.5	67.8	33.8	82.3	56.7	81.3	55.8
	Hebrew	82.3	41.7	81.3	60.9	76.3	53.3	90.5	68.5	90.3	60.1
	Hungarian	72.7	13.9	86.7	63.3	72.0	31.7	89.9	69.6	89.4	69.5

Table 2: Accuracy of CRF taggers (10fold CV)

unlabelled text is obtained from the Leipzig Corpus Collection (Quasthoff et al., 2006), which provides large text quantities crawled from the web for many languages. We use $15 \cdot 10^6$ tokens to create the clusters from the same amount of text for all languages. We provide the cluster ids in substrings of varying length to the classifier (Owoputi et al., 2013).

Results In Figure 2, we show the results of our parameter search experiment. The triangles mark the results of the various feature configurations. The diamond symbol shows the configuration which works best over all corpora. We refer to this best working configuration as *Best CRF* subsequently, it uses a word-context window of 1 word to the left and right and the 750 most frequent character [1..4] grams with additionally adding word clusters. Especially for morphologically-rich languages, the spread is quite large which is caused by the lower number of character-ngrams in those configurations. For corpora such as *Slovene-1*, we see that more accurate configurations exist than *Best CRF* but more importantly, the selected configuration is always among the best working ones.

We show the results of *Best CRF* and the performance of the individual features for each language in Table 2, and compare the results to HunPos, the highest accuracies are highlighted in grey. When evaluating the features separately, the character-ngrams reach the highest accuracy on OOV words. Especially on the Slavic language family the character-ngrams perform much better than using only word-ngrams or clusters. Furthermore, using only character-ngrams is often competitive to using only word-ngrams. Hence, a rather naïve strategy to achieving a decent performance on almost any language is to just use

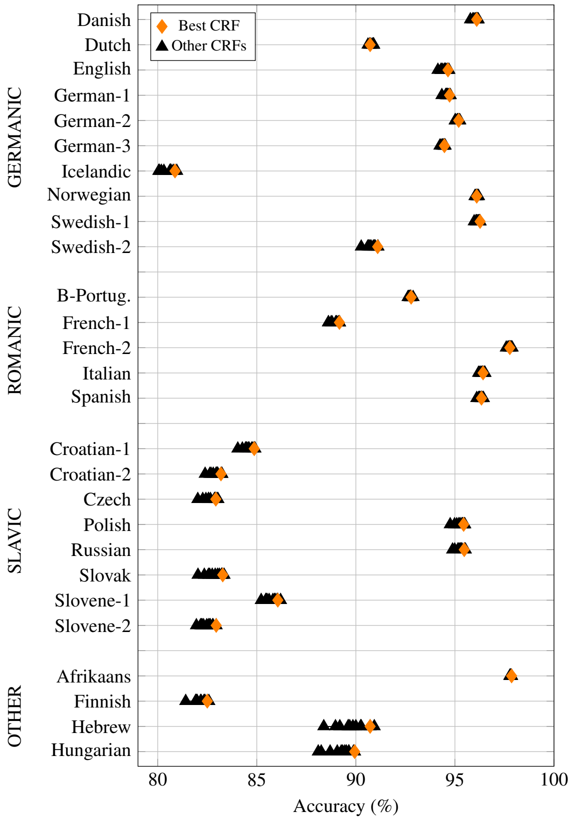


Figure 2: Variance of CRF taggers (10fold CV)

all kinds of character-ngrams. The cluster feature also performs better than the word-ngrams. Considering that we had to limit the amount of data for creating the clusters for comparability, this feature assumedly has more potential when using larger data sizes (Derczynski et al., 2015). The combination of all features in the column *Best CRF* shows that the features address quite different information and add up well, so unsurprisingly, this configuration reaches the overall best accuracies. The difference to HunPos is, with often less than one percent point difference, only small. Off-the-shelf taggers do, hence, not necessary have a disadvantage over constructing an own tagger. In the remainder of this work, we will use the *Best CRF* configuration when discussing CRF tagger results.

5 LSTM Experiments

When using neural networks, the details of how word and character information is provided greatly influences the learning success of the network. We will reproduce network setups which have also been used in Plank et al. (2016) to ensure comparability to the coarse-grained results to which we compare our results:

Word In this setup, we train a network on the word embeddings only and provide them to a bidirectional LSTM. This setup will serve as baseline.

Char The character embeddings of a word are provided to a bidirectional LSTM. The last state of the forward and the backward character LSTM are combined (Ling et al., 2015) and provided to another bidirectional LSTM layer.

Word-Char This architecture is a combination of the previous two architectures. The last state of the character LSTMs is added to the word embedding information before it is provided to the next LSTM layer.

Word-Char+ The architecture by Plank et al. (2016) combines word and character level information and additionally considers the log-frequency of the next word during training. This tagger reported state-of-the-art results and we use the provided reference implementation of this tagger in our setup.

LSTMs have the reputation to require larger amounts of training data. With the 50k tokens we use this is barely fulfilled, however, Plank et al. (2016) find this sensitivity to be less severe and set a corpus size of 60k tokens as lower bound for their coarse-grained tagging experiments. We will come back to this data size issue in Section 7, where we evaluate using all tokens in a corpus (and arriving at the same conclusions as for our 50k token datasets). Furthermore, in many cases only smaller dataset sizes are available, sometimes even less than 50k tokens. It is, thus, important to know if considering neural network taggers makes sense at all (on fine-grained tagsets), thus we will train LSTM models on smaller dataset sizes.

We implement the LSTM taggers in DyNet (Neubig et al., 2017) and use the hyper-parameter settings by Plank et al. (2016), i.e. we train 20 epochs using Statistical-Gradient-Descent with a learning rate of 0.1 and adding Gaussian noise of 0.2 to the embedding layer. We train word embeddings on the data we already used for the *semantic feature* in the CRF experiments by using *fastText* (Bojanowski et al., 2016) . The the character-level embeddings are trained on-the-fly.

Results In Figure 3, we show the results for the LSTM architectures. The *Word-Char+* tagger performs best followed by *Word-Char*, which is not surprising as *Word-Char+* is based on this

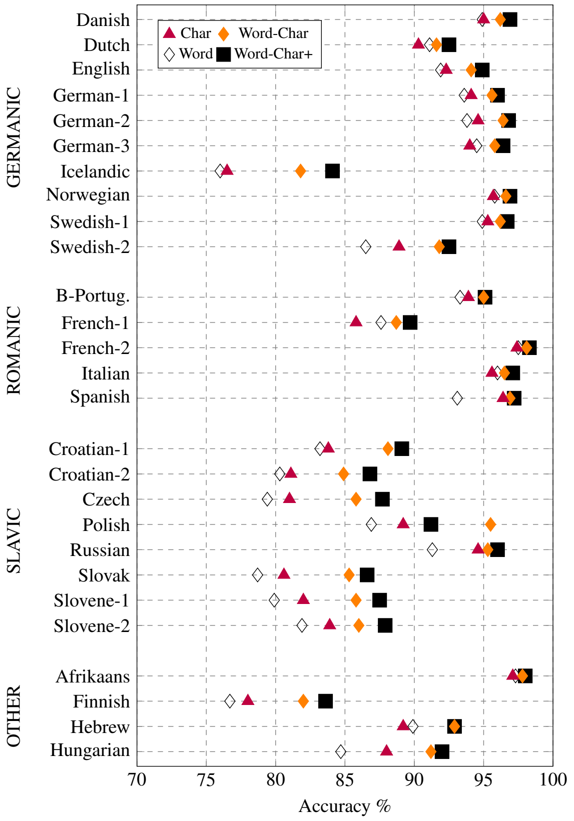


Figure 3: Variance of LSTM taggers (10fold CV)

architecture. For the Germanic and Romanic languages, the accuracy of the various architectures is similar but for Slavic languages, which use much more fine-grained tagsets, the differences are rather large. For instance, the *Char* architecture reaches only small improvements over the *Word* baseline on *Croatian or Czech* while on *Spanish, or Hungarian* the character architecture is clearly better than the baseline. Table 3 shows the detailed results and additionally reports the accuracy values on OOV with best results highlighted in grey. The *Char* architecture is in many cases competitive to the HunPos reference system. This shows that the performance of many off-the-shelf taggers is rather easy to approximate by relying only on character-level information.

The results by the *Char* architecture also explains why the *Word-Char* architecture performs so well although the amount of syntactical information is quite limited with 50k tokens. A large part of the necessary information is already obtained by the character model, which requires a lot less training data than a model on the word level. Thus, the results of Plank et al. (2016) on coarse-tagsets are reproducible for fine-grained tagsets with the *Word-Char* architecture being the essential property to achieving high accuracy.

6 Influence of Tagset Size

A researcher who works with morphologically rich languages will often be interested in additional morphologic details such as case or gender. This drastically complicates the task, as a few hundred instead of a few dozen PoS tag distinctions have to be learned. In this experiment, we will examine the impact of an increasing number of PoS tags on the accuracy of the taggers to provide reference values of how much performance a tagger seems to loose with an increasing tagset size.

Results In Figure 4, we show a comparison of the tagging accuracy in relation to the number of PoS tags. We show the best performing LSTM tagger *Word-Char+*, the *CRF* tagger and *HunPos*. Each data point represents the averaged CV result on one corpus with the respective tagger. We see a certain clustering of the data points for the small tagset sizes, which shows that the taggers tend to perform highly similarly for many languages. This means that the tagset size has a larger effect on the accuracy than the language of the corpus.

For each PoS tagger, a regression trendline is plotted which indicates the average loss in accuracy with an increasing tagset size. For one-hundred additional PoS tags, *Word-Char+* loses 0.35 points in accuracy, while *CRF* and *HunPoS* have a much steeper decay of 0.45 points. Hence, with growing tagset size the tagger choice becomes increasingly more important. Furthermore, the benefit of more sophisticated tagger architectures becomes only apparent on large PoS tagsets.

7 Comparison with Reference Taggers

In this experiment, we compare our results to reference taggers from the literature that are tailored towards certain languages. Our experiments until now were limited to the fixed dataset size that we set at the beginning for comparability. Especially for the morphologically fine-grained tagsets this might have been problematic, as it is doubtful if all PoS tags of a morphological tagset do even occur on 50k tokens. Thus, in order to evaluate the taggers using all available data, we will reproduce setups reported in the literature and compare the performance of the taggers to those results.

This experiment limits the number of comparisons we can make drastically, as we need to have

Lang.		Word		Char		Word-Char		Word-Char+		HunPos	
Group	Corpus Id	All	OOV	All	OOV	All	OOV	All	OOV	All	OOV
Germanic	Danish	94.9	72.7	95.0	79.1	96.4	82.5	96.9	83.4	94.9	74.2
	Dutch	91.1	82.3	90.3	83.6	91.6	85.7	92.5	87.1	89.9	80.6
	English	91.9	65.9	92.3	77.4	94.1	79.6	94.9	80.9	93.8	77.7
	German-1	93.6	78.3	94.1	84.5	95.6	87.6	96.0	88.3	94.4	83.7
	German-2	94.5	82.4	94.6	87.1	96.4	90.1	96.8	91.5	94.4	85.4
	German-3	93.8	80.3	94.0	84.9	95.8	88.6	96.4	89.8	94.4	83.9
	Icelandic	76.0	34.8	76.5	49.3	81.8	56.2	84.1	60.6	79.8	51.9
	Norwegian	95.8	86.2	95.7	88.2	96.6	90.3	96.9	90.3	95.5	86.5
	Swedish-1	94.9	81.4	95.3	86.7	96.2	89.0	96.7	89.8	95.6	85.9
	Swedish-2	86.5	54.3	88.9	74.3	91.8	78.5	92.5	80.4	91.4	77.6
Romanic	B-Portug.	93.3	82.4	93.9	87.4	95.0	90.3	95.1	90.8	93.3	84.2
	French-1	87.6	67.0	85.8	72.0	88.7	77.4	89.7	78.7	88.2	71.8
	French-2	97.5	80.4	97.4	83.4	98.1	87.7	98.3	88.7	97.4	82.4
	Italian	96.0	81.3	95.6	84.2	96.5	85.9	97.1	86.9	95.8	80.8
	Spanish	93.1	63.3	96.4	85.5	96.9	86.1	97.2	87.0	96.6	83.6
Slavic	Croatian-1	83.2	55.5	83.8	67.5	88.1	72.8	89.1	75.2	84.7	66.9
	Croatian-2	80.3	52.4	81.1	63.8	84.9	69.1	86.8	72.4	82.6	63.9
	Czech	79.4	49.1	81.0	62.7	85.8	68.7	87.7	72.4	81.7	60.9
	Polish	86.9	73.6	89.2	84.7	95.5	91.2	91.2	88.0	93.6	85.4
	Russian	91.3	73.2	94.6	85.8	95.3	86.9	96.0	88.4	94.6	83.6
	Slovak	78.7	44.9	80.6	65.0	85.3	69.7	86.6	71.4	82.9	61.6
	Slovene-1	81.9	44.5	83.9	61.1	86.0	62.6	87.9	65.7	82.6	59.6
	Slovene-2	79.9	47.9	82.0	63.4	85.8	67.4	87.5	70.1	86.2	59.5
Other	Afrikaans	97.3	82.8	97.1	85.8	97.8	88.4	98.0	90.0	97.3	85.5
	Finnish	76.7	42.7	78.0	57.6	82.0	58.9	83.6	61.2	81.3	55.8
	Hebrew	89.9	60.2	89.2	66.9	92.2	69.7	92.9	72.1	90.3	60.1
	Hungarian	84.7	53.3	88.0	73.1	91.2	76.9	92.0	79.0	89.4	69.5

Table 3: Accuracy of LSTM taggers (10fold CV)

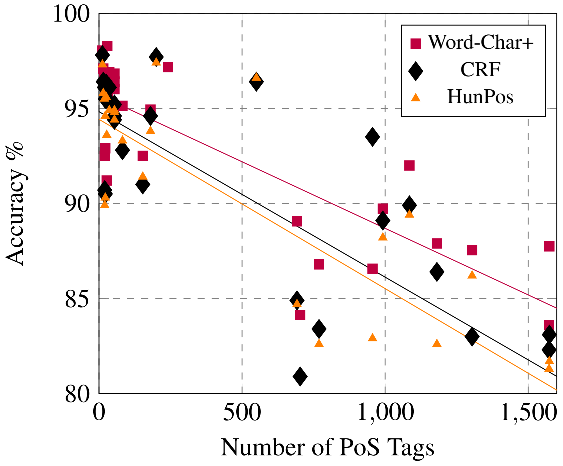


Figure 4: Influence of tagset size on accuracy

the same corpora as used in the literature. We, thus, reproduce for *Czech* the setup by Spoustová et al. (2009) with training on 10^6 and evaluation on $2 \cdot 10^5$ tokens, for *German-2* the setup by Giesbrecht and Evert (2009) and for *Swedish-2* the setup by Östling (2013), which both use 10fold cross-validation over the full corpus size.

Taggers for Slavic languages often make use of additional resources such as morphological dictionaries, which we intentionally do not include to avoid human-crafted resources that are not available for all languages. Thus, we do not expect to reach state-of-the-art performance, but we want to quantify the size of the gap.

Results In Table 4, we show a comparison of our results to the results reported in the literature. On *German-2* and *Swedish-2*, the Word-Char+ tagger is able to reach better results than the reported reference values except for *Czech* which uses a morphologically fine-grained tagset. Thus, language-

			Δ to reference tagger		
Corpus Id	# Tags	Acc (%)	HunPos	CRF	Word-Char+
Czech	1,574	95.9	-4.7	-3.2	-1.5
German-2	54	97.6	-0.1	-0.2	0.9
Swedish-2	153	96.1	0.0	-0.6	0.1

Table 4: Results of reproducing setups in the literature using the *full corpus size*

fitted PoS taggers reach better results than neural networks when training models on corpora with extremely fine-grained PoS tagsets. However, for smaller tagsets sizes the need for using language-fitting is negligible.

8 Conclusion

We replicated a study in which LSTM PoS taggers are compared to CRF and HMM taggers on corpora with a coarse-grained tagset. Our replication focused on whether results reported for coarse-grained tagsets do also hold when training models on fine-grained tagsets. Therefore, we collected a large set of 27 evaluation corpora that are annotated with the commonly used fine-grained tagset of 21 languages. The replication confirmed the superior performance of the LSTM tagger reported by Plank et al. (2016) also on fine-grained tagsets. However, we also found that for smaller tagset sizes the differences between the LSTM, our self-implemented CRF and the HMM tagger are often only small. The advantages of the LSTM tagger over other taggers grow proportionally with the tagsets size of the corpus. On morphologically fine tagsets, even the LSTM tagger fails to reach results reported in the literature when reproducing those setups.

Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under grant No. GRK 2167, Research Training Group "User-Centred Social Media".

References

Željko Agić and Nikola Ljubešić. 2014. The setimes.hr linguistically annotated corpus of croatian. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 1724–1727, Reykjavik, Iceland. ELRA.

Sandra Aluísio, Jorge Pelizzoni, Ana Raquel Marchi, Lucélia de Oliveira, Regiana Manenti, and Vanessa Marquiafável. 2003. *An Account of the Challenge of Tagging a Reference Corpus for Brazilian Portuguese*. Faro, Portugal.

Giuseppe Attardi, Stefano Dei Rossi, Giulia Di Pietro, Alessandro Lenci, Simonetta Montemagni, and Maria Simi. 2010. A Resource and Tool for Supersense Tagging of Italian Texts. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta. ELRA.

Liesbeth Augustinus, Peter Dirix, Daniel Van Niekerk, Ineke Schuurman, Vincent Vandeghinste, Frank Van Eynde, and Gerhard Van Huyssteen. 2016. AfriBooms: An Online Treebank for Afrikaans. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Portorož, Slovenia. ELRA.

Eduard Bejček, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Magda Ševčíková, Jan Štěpánek, and Šárka Zikánová. 2013. Prague Dependency Treebank 3.0.

V. V. Bocharov, S.V. Alexeeva, D.V. Granovsky, E.V. Protopopova, M.E. Stepanova, and A.V. Surikov. 2013. Crowdsourcing morphological annotation. http://opencorpora.org.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*.

Cristina Bosco, Manuela Sanguinetti, and Leonardo Lesmo. 2012. The Parallel-TUT: a multilingual and multiformat treebank. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Istanbul, Turkey. ELRA.

Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2000. Alpino: Wide Coverage Computational Analysis of Dutch. *Computational Linguistics in the Netherlands 2000. Selected Papers from the 11th CLIN Meeting.*, pages 45–59.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic Interpretation of a German Corpus. *Research on Language and Computation*, 2(4):597–620.

Thorsten Brants. 2000. TnT: A Statistical Part-of-speech Tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, pages 224–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peter F Brown, Peter V DeSouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18:467–479.

Matthias Buch-Kromann and Iørn Korzen. 2010. The Unified Annotation of Syntax and Discourse in the Copenhagen Dependency Treebanks. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 127–131, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marie Candito, Guy Perrier, Bruno Guillaume, Corentin Ribeyre, Karén Fort, Djamé Seddah, and Éric Villemonte De La Clergerie. 2014. Deep Syntax Annotation of the Sequoia French Treebank. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Reykjavik, Iceland. ELRA.

Dóra Csendes, János Csirik, Tibor Gyimóthy, and András Kocsor. 2005. *The Szeged Treebank*. Karloy Vary, Czech Republic.

Leon Derczynski, Sean Chester, and Kenneth S.Bøgh. 2015. Tune your brown clustering, please. In *Proceedings of the conference on Recent Advances in Natural Language Processing*, pages 110 – 117, Bulgaria.

Jan Einarsson. 1976. Talbankens skriftspråkskonkordans. Technical report, Lund University: Department of Scandinavian Languages.

Eva Ejerhed and Gunnel Källgren. 1997. Stockholm Umeå Corpus (SUC) version 1.0. *Department of Linguistics, Umeå University*.

Tomaž Erjavec. 2002. Compiling and Using the IJS-ELAN Parallel Corpus. In *Informatica*, pages 299–307.

Tomaž Erjavec. 2010. MULTEXT-East Version 4: Multilingual Morphosyntactic Specifications, Lexicons and Corpora. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta. ELRA.

Kilian A. Foth, Arne Köhn, Niels Beuck, and Wolfgang Menzel. 2014. Because Size Does Matter: The Hamburg Dependency Treebank. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Reykjavik, Iceland. ELRA.

Eugenie Giesbrecht and Stefan Evert. 2009. Is Part-of-Speech Tagging a Solved Task? An Evaluation of POS Taggers for the German Web as Corpus. In *Proceedings of the 5th Web as Corpus Workshop (WAC5)*, San Sebastian, Spain.

Péter Halácsy, András Kornai, and Csaba Oravecz. 2007. HunPos: An open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the ACL*, pages 209–212, Stroudsburg. Association for Computational Linguistics.

Sigrún Helgadóttir, Ásta Svavarsdóttir, Eiríkur Rögnvaldsson, Kristín Bjanadóttir, and Hrafn Loftsson. 2012. The Tagged Icelandic Corpus (MIM). In *Proceedings of the Workshop on Language Technology for Normalisation of Less-Resourced Languages*, Istanbul, Turkey. ELRA.

D. Hládek, J. Staš, and J. Juhár. 2012. Dagger: The Slovak morphological classifier. In *Proceedings ELMAR-2012*, pages 195–198.

Tobias Horsmann and Torsten Zesch. 2016. LTL-UDE @ EmpiriST 2015: Tokenization and PoS Tagging of Social Media Text. In *Proceedings of the 10th Web as Corpus Workshop (WAC-X) and the EmpiriST Shared Task*, pages 120–126, Berlin, Germany.

Alon Itai and Shuly Wintner. 2008. Language resources for Hebrew. *Language Resources and Evaluation*, 42(1):75–98.

Simon Krek, Tomaž Erjavec, Kaja Dobrovoljc, Sara Može, Nina Ledinek, and Nanika Holz. 2013. Training corpus ssj500k 1.3. Slovenian language resource repository CLARIN.SI.

John D Lafferty, Andrew McCallum, and Fernando C N Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernández Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation.

Nikola Ljubešić, Filip Klubička, Željko Agić, and Ivo-Pavao Jazbec. 2016. New Inflectional Lexicons and Training Corpora for Improved Morphosyntactic Annotation of Croatian and Serbian. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Paris, France. ELRA.

Montserrat Marimon, Núria Bel, Beatriz Fisas, Blanca Arias, Silvia Vázquez, Jorge Vivaldi, Carlos Morell, and Mercè Lorente. 2014. The IULA Spanish LSP Treebank. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Reykjavik, Iceland. ELRA.

W. Nelson Francis and Henry Kuçera. 1964. *Manual of Information to Accompany a Standard Corpus of Present-day Edited American English, for use with Digital Computers*. Department of Linguistics, Brown University.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The Dynamic Neural Network Toolkit - Technical Report.

Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, Sam Bowman, Giuseppe G. A. Celano, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Berta Gonzales, Bruno Guillaume, Jan Hajič, Dag Haug, Radu Ion, Elena Irimia, Anders Johannsen, Hiroshi Kanayama, Jenna Kanerva, Simon Krek, Veronika Laippala, Alessandro Lenci, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Shunsuke Mori, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Prokopis Prokopidis, Sampo Pyysalo, Loganathan Ramasamy, Rudolf Rosa, Shadi Saleh, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Jan Štěpánek, Alane Suhr, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Sumire Uematsu, Larraitz Uria, Viktor Varga, Veronika Vincze, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2015. Universal Dependencies 1.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University in Prague.

Robert Östling. 2013. Stagger: An Open-Source Part of Speech Tagger for Swedish. *Northern European Journal of Language Technology (NEJLT)*, 3:1–18.

Olutobi Owoputi, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 NAACL: HLT*. Association for Computational Linguistics.

Patrick Paroubek. 2000. Language Resources as by-Product of Evaluation: The MULTITAG Example. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Athens, Greece. ELRA.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proceedings of ACL2016*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

Adam Przepiórkowski, Rafał L. Górski, Barbara Lewandowska-Tomaszyk, and Marek Łaziński. 2008. Towards the National Corpus of Polish. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Marrakech, Morocco. ELRA.

Uwe Quasthoff, Matthias Richter, and Christian Biemann. 2006. Corpus Portal for Search in Monolingual Corpora. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 1799–1802, Genoa. ELRA.

Steffen Remus, Gerold Hintz, Darina Benikova, Thomas Arnold, Judith Eckle-Kohler, Christian M. Meyer, Margot Mieskes, and Chris Biemann. 2016. EmpiriST: AIPHES Robust Tokenization and POS-Tagging for Different Genres. In *Proceedings of the 10th Web as Corpus Workshop (WAC-X)*, pages 106–114, Berlin, Germany.

Mojgan Seraji. 2011. A Statistical Part-of-Speech Tagger for Persian. In *Proceedings of the 18th Nordic Conference of Computational Linguistics*.

Per Erik Solberg, Arne Skjæholt, Lilja Øvrelid, Kristin Hagen, and Janne Bondi Johannessen. 2014. The Norwegian Dependency Treebank. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, Reykjavik, Iceland. ELRA.

Drahomíra "johanka" Spoustová, Jan Hajič, Jan Raab, and Miroslav Spousta. 2009. Semi-Supervised Training for the Averaged Perceptron POS Tagger. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 763–771, Athens, Greece. Association for Computational Linguistics.

Heike Telljohann, Erhard Hinrichs, Sandra Kübler, Ra Kübler, and Universität Tübingen. 2004. The Tüba-D/Z Treebank: Annotating German with a Context-Free Backbone. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 1175–1178. ELRA.

Atro Voutilainen. 2011. FinnTreeBank: Creating a research resource and service for language researchers with Constraint Grammar. *Constraint Grammar Applications*, page 41.

The Labeled Segmentation of Printed Books

Lara McConnaughey
Computer Science Division
University of California, Berkeley
laram@berkeley.edu

Jennifer Dai
Computer Science Division
University of California, Berkeley
jenniferdai@berkeley.edu

David Bamman
School of Information
University of California, Berkeley
dbamman@berkeley.edu

Abstract

We introduce the task of *book structure labeling*: segmenting and assigning a fixed category (such as TABLE OF CONTENTS, PREFACE, INDEX) to the document structure of printed books. We manually annotate the page-level structural categories for a large dataset totaling 294,816 pages in 1,055 books evenly sampled from 1750–1922, and present empirical results comparing the performance of several classes of models. The best-performing model, a bidirectional LSTM with rich features, achieves an overall accuracy of 95.8 and a class-balanced macro F-score of 71.4.

1 Introduction

The availability of large-scale book corpora (such as those created by Google Books, the Internet Archive and the HathiTrust) has driven a flurry of work in cultural analytics over the past decade, in which the text contained in historical books has provided the raw material for the analysis of genre (Underwood, 2016), literary character (Bamman et al., 2014), geographic attention (Wilkens, 2013), fame (Michel et al., 2010), and much more.

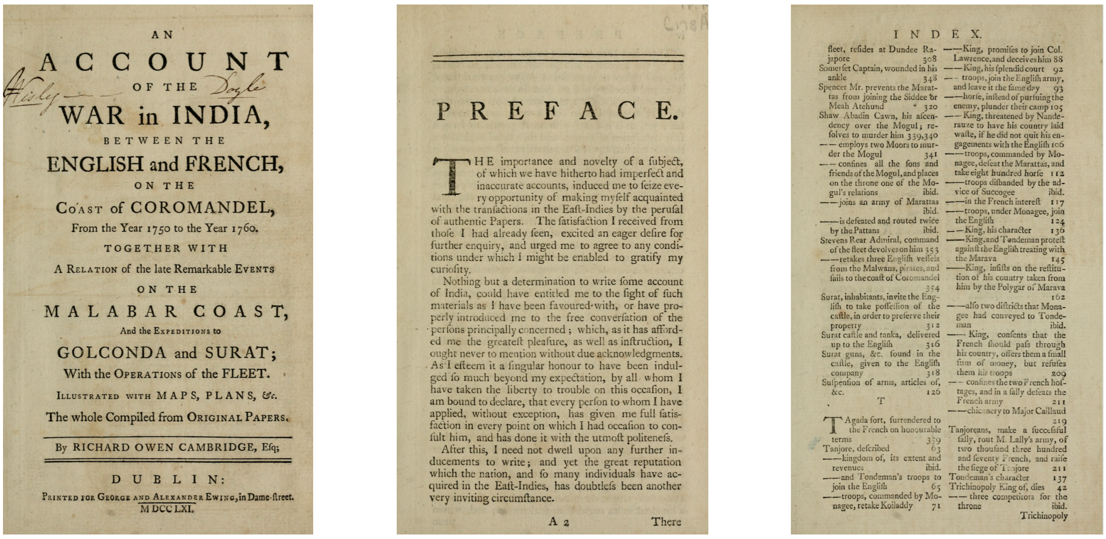
Books, however, are not undifferentiated streams of text in the same way that born-digital documents like tweets or emails are; they are physical objects with materiality (Werner, 2012) and are arranged in a complex structure steeped in a long design tradition, with the core content of the work placed between structured frontmatter (such as a table of contents and introduction) and backmatter (such as an appendix and index). Not all of this content is desirable for all analyses; as we show below, 11% of all pages in books belong to the peritext (Genette, 1987) that surrounds the core content, with wide variability from book to book. For other analyses, such as those addressing questions in book history (Kirschenbaum and Werner, 2014), isolating this structure in a consistent way across historical documents can enable research into how the form of the printed book has, for example, changed over time.

While other work has focused on extracting the idiosyncratic structure inherent in each book, such as recognizing chapter boundaries in order to automatically generate a table of contents, or link a parsed table of contents to positions in a book (Déjean and Meunier, 2005, 2009; Wu et al., 2013; Gao et al., 2009), labeling document segments with a fixed typology has complementary benefits, allowing researchers to identify consistent categories in all books regardless of the names assigned by a specific author or publisher, or popular at a given time.[1]

At the same time, book structure labeling presents real challenges to automatic identification. While large-scale digital collections originate in page scans of the books, the most common form of access by researchers is through the output of OCR; raw image files are prohibitively expensive both in terms of disk space (15.1 million books from the HathiTrust consumes 677 ter-

[1] For example, a section whose function is to outline the structural regions of a book and list the pages on which they begin may be known at different points in history as a "table of contents," "index," or several other terms.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 737–747
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics



AN
ACCOUNT
Hisly —— OF THE Doyle
WAR in INDIA,
BETWEEN THE
ENGLISH and FRENCH,
ON THE
CoAST of COROMANDEL,
From the Year 1750 to the Year 1760.
TOGETHER WITH
A RELATION of the late Remarkable EVENTS
ON THE
MALABAR COAST,
And the EXPEDITIONS to
GOLCONDA and SURAT;
With the OPERATIONS of the FLEET.
ILLUSTRATED WITH MAPS, PLANS, &c.
The whole Compiled from ORIGINAL PAPERS.

By RICHARD OWEN CAMBRIDGE, Eſq;

DUBLIN:
PRINTED FOR GEORGE AND ALEXANDER EWING, in Dame-ſtreet.
M DCC LXI.

(a) Title page.

PREFACE.

THE importance and novelty of a ſubject, of which we have hitherto had imperfect and inaccurate accounts, induced me to ſeize every opportunity of making myſelf acquainted with the tranſactions in the Eaſt-Indies by the peruſal of authentic Papers. The ſatisfaction I received from thoſe I had already ſeen, excited an eager deſire for further enquiry, and urged me to agree to any conditions under which I might be enabled to gratify my curioſity.

Nothing but a determination to write ſome account of India, could have entitled me to the ſight of ſuch materials as I have been favoured with, or have properly introduced me to the free converſation of the perſons principally concerned; which, as it has afforded me the greateſt pleaſure, as well as inſtruction, I ought never to mention without due acknowledgments. As I eſteem it a ſingular honour to have been indulged ſo much beyond my expectation, by all whom I have taken the liberty to trouble on this occaſion, I am bound to declare, that every perſon to whom I have applied, without exception, has given me full ſatisfaction in every point on which I had occaſion to conſult him, and has done it with the utmoſt politeneſs.

After this, I need not dwell upon any further inducements to write; and yet the great reputation which the nation, and ſo many individuals have acquired in the Eaſt-Indies, has doubtleſs been another very inviting circumſtance.

A 2 There

(b) Preface.

INDEX.

fleet, reſides at Dundee Rajapore 308
Somerſet Captain, wounded in his ankle 348
Spencer Mr. prevents the Marattas from joining the Siddee or Meah Atehund 320
Shaw Abadin Cawn, his aſcendency over the Mogul; reſolves to murder him 339,340
—— employs two Moors to murder the Mogul 341
—— confines all the ſons and friends of the Mogul, and places on the throne one of the Mogul's relations ibid.
——joins an army of Marattas ibid.
——is defeated and routed twice by the Pattans ibid.
Stevens Rear Admiral, command of the fleet devolves on him 353
——retakes three Engliſh veſſels from the Malwans, pirates, and ſails to the coaſt of Coromandel 354
Surat, inhabitants, invite the Engliſh to take poſſeſſion of the caſtle, in order to preſerve their property 312
Surat caſtle and tanka, delivered up to the Engliſh 316
Surat guns, &c. found in the caſtle, given to the Engliſh company 318
Suſpenſion of arms, articles of, &c. 126

T

TAgada fort, ſurrendered to the French on honourable terms 359
Tanjore, deſcribed 63
——kingdom of, its extent and revenues ibid.
——and Tondeman's troops to join the Engliſh 65
——troops, commanded by Monagee, retake Koiladdy 71
——King, promiſes to join Col. Lawrence, and deceives him 88
——King, his ſplendid court 92
—— troops, join the Engliſh army, and leave it the ſame day 93
——horſe, inſtead of purſuing the enemy, plunder their camp 105
——King, threatened by Nanderauze to have his country laid waſte, if he did not quit his engagements with the Engliſh 106
——troops, commanded by Monagee, defeat the Marattas, and take eight hundred horſe 112
——troops diſbanded by the advice of Succogee ibid.
——in the French intereſt 117
——troops, under Monagee, join the Engliſh 124
——King, his character 136
——King, and Tondeman proteſt againſt the Engliſh treating with the Marava 145
——King, inſiſts on the reſtitution of his country taken from him by the Polygar of Marava 162
——alſo two diſtricts that Monagee had conveyed to Tondeman ibid.
—— King, conſents that the French ſhould paſs through his country, offers them a ſmall ſum of money, but refuſes them his troops 209
—— confines the two French hoſtages, and in a ſally defeats the French army 211
——chicanery to Major Caillaud 219
Tanjoreans, make a ſucceſsful ſally, rout M. Lally's army, of two thouſand three hundred and ſeventy French, and raiſe the ſiege of Tanjore 211
Tondeman's character 137
Trichinopoly King of, dies 42
—— three competitors for the throne ibid.

Trichinopoly

(c) Index.

Figure 1: *An Account of the War in India, Between the English and French* (1761). From the HathiTrust.

abytes of space[2]) and in the resources required for computational processing. While people are able to distinguish the different sections of a book with ease, the degraded nature of the OCR output (especially for historical books) blurs the clear markers that signal the category to human readers—both in terms of the lexical signals like "Preface" or "Index" that head a page, and its visual structure as well. Figure 1 illustrates an example of three pages from a single book drawn from the HathiTrust; figure 2 displays the corresponding OCR output; the degradation introduced by OCR affects not only the accuracy of character and word identification, but also the structural layout as well.

To address these limitations and enable research that depends on reasoning over fine-grained document structure within books, we introduce the task of labeled segmentation, and make the following contributions:

- We create an human-annotated gold standard of 294,816 pages in 1,055 printed books drawn from the HathiTrust Digital Library.
- We approach this problem in the most common resource-deficient scenario researchers most frequently encounter: with access only to the pre-existing output of OCR.
- We compare several different classes of models, including a fast independent predictor (a random forest), a simple linear sequence labeling model (CRF), and a sequence labeling bidirectional LSTM that can capture non-linearities in the feature space. All data and pre-trained models are openly available to the public at https://github.com/dbamman/book-segmentation.

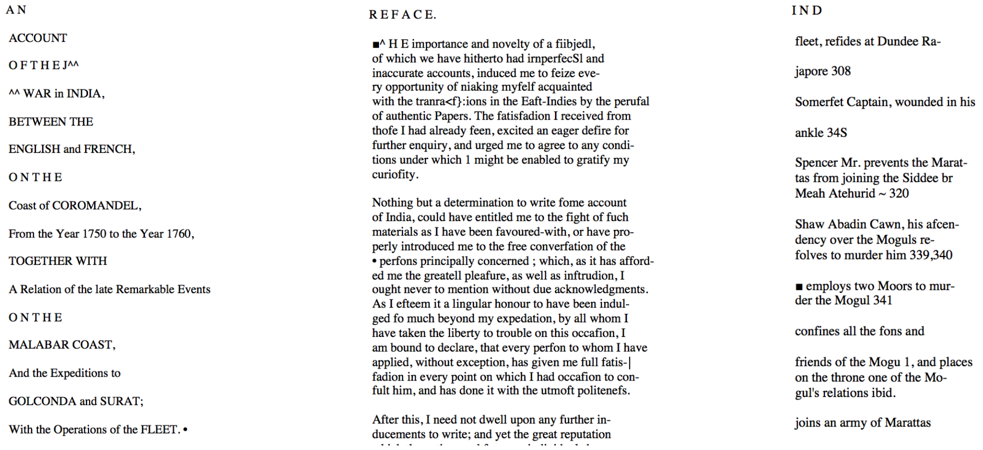
2 Data

In order to support the analysis and prediction of labeled document structure, we present a manually annotated dataset of 1,055 books, where each page has been labeled according to one of 10 categories described in §2.1 below. All books originate in the HathiTrust Digital Library. In order to capture historically representative phenomena, we use the decade-stratified sample of 1,075 books from Bamman et al. (2017), in which each decade from 1750-1922 is roughly equally represented. From this sample of 1,075 apparent books, we exclude all non-book records (including digitized newspaper clippings, unbound pamphlets and reports, opera programs, etc.) to yield a total labeled dataset of 294,816 pages in 1,055 books.

2.1 Categories

While there is no codified form of the standard categories that are present in print books, modern book designers generally adhere to a tradition involving a typical sequence of parts (Wilson, 1993; Lee, 2009). We draw on this tradition to inform our set of the following ten categories; to contextualize its prevalence, each category is listed with its description and the fraction of the 1,055 books

[2] https://www.hathitrust.org/statistics_visualizations



```
A N

ACCOUNT

O F T H E J^^

^^ WAR in INDIA,

BETWEEN THE

ENGLISH and FRENCH,

O N T H E

Coast of COROMANDEL,

From the Year 1750 to the Year 1760,

TOGETHER WITH

A Relation of the late Remarkable Events

O N T H E

MALABAR COAST,

And the Expeditions to

GOLCONDA and SURAT;

With the Operations of the FLEET. •
```

(a) Title page.

```
R E F A C E.

■^ H E importance and novelty of a fiibjedl,
of which we have hitherto had irnperfecSl and
inaccurate accounts, induced me to feize eve-
ry opportunity of niaking myfelf acquainted
with the tranra<f}:ions in the Eaft-Indies by the perufal
of authentic Papers. The fatisfadion I received from
thofe I had already feen, excited an eager defire for
further enquiry, and urged me to agree to any condi-
tions under which 1 might be enabled to gratify my
curiofity.

Nothing but a determination to write fome account
of India, could have entitled me to the fight of fuch
materials as I have been favoured-with, or have pro-
perly introduced me to the free converfation of the
• perfons principally concerned ; which, as it has afford-
ed me the greatell pleafure, as well as inftrudion, I
ought never to mention without due acknowledgments.
As I efteem it a lingular honour to have been indul-
ged fo much beyond my expedation, by all whom I
have taken the liberty to trouble on this occafion, I
am bound to declare, that every perfon to whom I have
applied, without exception, has given me full fatis-|
fadion in every point on which I had occafion to con-
fult him, and has done it with the utmoft politenefs.

After this, I need not dwell upon any further in-
ducements to write; and yet the great reputation
```

(b) Preface.

```
I N D

fleet, refides at Dundee Ra-

japore 308

Somerfet Captain, wounded in his

ankle 34S

Spencer Mr. prevents the Marat-
tas from joining the Siddee br
Meah Atehurid ~ 320

Shaw Abadin Cawn, his afcen-
dency over the Moguls re-
folves to murder him 339,340

■ employs two Moors to mur-
der the Mogul 341

confines all the fons and

friends of the Mogu 1, and places
on the throne one of the Mo-
gul's relations ibid.

joins an army of Marattas
```

(c) Index.

Figure 2: OCR output for the page scans illustrated in fig. 1.

in our dataset in which it appears (for example, 47.8% of books have an annotated preface).

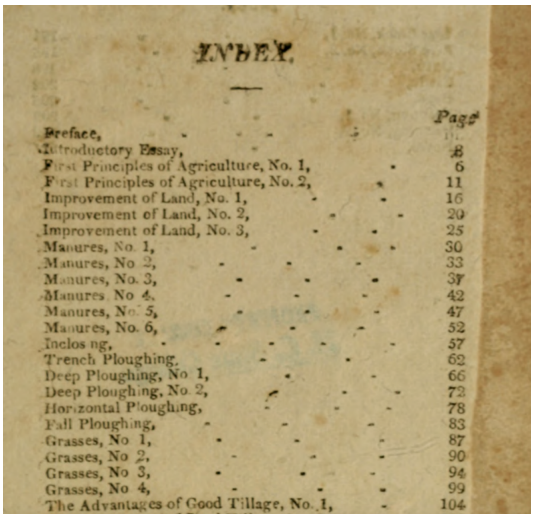
- Title page (93.0%), which lists the title of the work and (optionally) other information including the names of the author, translator, and others involved in its creation. In this category we group the primary title page along with the Half-title (a page that generally only presents the title of the work, often preceding the main title page or first chapter).
- Ad card (18.1%), which lists other works by the author or publisher; or, more generally, any other object that is advertised for sale.
- Publisher (39.9%), which includes the modern Copyright page (typically on the verso side of the title page) and also the Colophon (an imprint often appearing at the end of a work).
- Dedication (17.5%), an inscription by the author dedicating the work to another.
- Preface (47.8%), which includes a Foreword, Preface, and Introduction. While modern designers articulate prescriptive distinctions among these categories primarily in their subject matter and authorial voice,[3] we do not find a strong distinction among these sub-categories evident when labeling the text. We therefore follow Genette (1987) in grouping all together as prefatory material.
- Table of contents (46.8%), which includes "an accurate listing of all textual matter which follows it and the page on which the parts of the book commence" (Wilson, 1993).
- Text (99.3%), which includes the main contents of the book. Text is naturally the most frequent category, but only accounts for 89.4% of pages in all books in our dataset. We also see wide variability from book to book; the average Text ratio in books is 0.82, with a standard deviation of 0.18.
- Appendix (14.4%) includes a heterogeneous mix of other minor categories that appear infrequently in different books. These include: Notes (1.1%) (which "have the character of footnotes which, because of their extent, are placed at the back of the book" (Wilson, 1993)); Bibliography (1.7%), "a listing of the books and periodicals, etc., which the author has used as source material or which he recommends as supplementary reading matter" (Wilson, 1993); Glossary (0.6%), "a list of definitions of terms used in the text" (Wilson, 1993), Errata (4.1%), mistakes corrected in the printing of the book, and Subscribers (1.7%), a list of individuals who have committed to purchasing the work in advance (a historical category not frequently seen in modern texts). We annotate each of these subcategories individually for future work, but collapse them into the single category of Appendix for the work below.
- Index (19.2%), which "serves to catalogue,

[3] "A preface is written by the author and is generally about the writing of the book. A foreword is a comment on the book and/or the author by another person. An introduction, which may by the author or another, may contain such matter, but it's primarily a preparation for, or explanation of, the content" (Lee, 2009)

with page indications, all the references which an author wishes to identify" (Wilson, 1993).

- N/A. For each of the nine categories above, we annotate the beginning and end pages present in a book; any page not contained within a labeled section receives the label N/A.

As Genette (1987) articulates, each of these structural categories mediates the relationship between the text and its audience, and each serves a different illocutionary purpose. The TITLE is addressed to the general public (not necessarily the readers) and is not only informational (informing of the name of the work), but also serves as important marketing material as well; PREFACES are addressed to readers, and may be written either by the author of the core content (*authorial*) or by another (*allographic*) and communicate the intention or interpretation of the work; the illocutionary force of a DEDICATION, in contrast, is performative: its very presence is a speech act that serves to dedicate the work to another.



INDEX.

	Page
Preface,	iii
Introductory Essay,	3
First Principles of Agriculture, No. 1,	6
First Principles of Agriculture, No. 2,	11
Improvement of Land, No. 1,	16
Improvement of Land, No. 2,	20
Improvement of Land, No. 3,	25
Manures, No. 1,	30
Manures, No 2,	33
Manures, No. 3,	37
Manures No 4.	42
Manures, No. 5,	47
Manures, No. 6,	52
Inclosing,	57
Trench Ploughing.	62
Deep Ploughing, No 1,	66
Deep Ploughing, No 2,	72
Horizontal Ploughing,	78
Fall Ploughing,	83
Grasses, No 1,	87
Grasses, No 2,	90
Grasses, No 3,	94
Grasses, No 4,	99
The Advantages of Good Tillage, No. 1,	104

Figure 3: Table of contents page listed as "Index."

For all categories, we label based on the tenor of the category's meaning, and not on the title of the section that may appear on the page. Figure 3 illustrates one such example of this distinction—a page whose function is to serve as a table of contents but is headed as an "Index" (and also appears at the back of the book, like contemporary indices); rather than functioning as an index in providing references to concepts within the text, it outlines the organizational structure of the sections (as a table of contents does).

Human judgments of these ten categories are relatively uncontroversial; to calculate the coherence of the task, we calculated the inter-annotator agreement rate for two annotators on 25 books, and find a chance-corrected Cohen's $\kappa = 0.83$, suggesting a very high level of agreement.[4] All books then receive a single judgment of page-level annotations by a single annotator.

3 Methods

To explore our ability to label book structure automatically, we test three different feature-rich classifiers. All make use of the same set of features.

3.1 Features

Keywords. Most words on a page are not predictive of the category to which it belongs; a word like *Britain* in a biography of Churchill may distinguish that book from other books, but will also equally be found on the title page, table of contents, preface, content, index, or any other category. Some words, however, are discriminative, such as the titles of the categories ("index," "preface," "dedication," etc.). To identify these terms, we train a unigram logistic regression classifier on the training-only partition of the data (described in section 4 below) and manually select keywords with high face validity. We create two sets of features from these keywords: presence of a keyword in the header of the page (the first four lines) and the presence of a keyword anywhere on the page.

Longest increasing subsequence. As figure 3 shows, tables of contents are distinguished from indices in that the page numbers generally increase from the top of the page to the bottom, corresponding to the linear order of the book. To capture this, we create a feature for the longest increasing subsequence (LIS) of numbers on the page. The LIS for any set of n randomly permuted numbers converges to a Tracy-Widom distribution (Baik et al., 1999); to enable feature value comparisons across pages with different total numbers, we conduct a permutation test by shuffling the numbers on the page and recalculating the LIS for that resample; we set the feature value to be 1 only when the observed LIS is greater than 5% of the LIS scores for the permutations (i.e., $p < 0.05$).

[4] Using the non-parametric bootstrap to account for the size of the sample in our confidence of the agreement rate, we find a 95% confidence interval for κ to be within the interval [0.65, 0.94].

Alphabetical sort. Indices, in contrast, are distinguished from tables of contents in that their lines are sorted alphabetically (from the top to the bottom of the page). To capture this, we create a feature measuring the degree to which the lines on a page are sorted, operationalized as the Spearman rank correlation coefficient (ρ) between the set of lines in their original order and the lines in sorted order. Perfectly sorted lines have a $\rho = 1$; inversely sorted lines have $\rho = -1$ and randomly ordered lines have an expected $\rho = 0$. To account for random sorting that take place by chance, we set this feature value to be ρ only when its p value (rejecting $H_0 \equiv \rho = 0) < 0.01$.

Letter distribution. In addition to measuring the degree which the full page is alphabetized, we can also capture important structural qualities of indices by measuring the degree to which initial letters in words are overrepresented on the page. We calculate this by measuring the empirical distribution of initial downcased letters `[a-z]` for all words in the book, and measure the degree to which the empirical distribution on the page overrepresents any individual letter. Rather than comparison the full distributions (using e.g., Jensen-Shannon divergence), we calculate the number of letters whose frequency on the page deviates from the book frequency for that letter by a z-score (accounting for the number of times we observe the letter) corresponding to a critical value $\alpha \leq 0.05$.

Roman numerals. Frontmatter preceding the main content is often paginated with roman numerals, rather than the arabic numerals found in the content. To capture this, we create a binary feature identifying the presence of roman numerals in the first four lines (header) or last four lines (footer) of the page, using the resources of Underwood (2017).

Page density. Content pages are relatively dense with characters (both letters and numbers); title pages and tables of contents are defined by greater volume of whitespace. To capture this differential, we introduce features for the ratio of words and numbers among all (whitespace-delimited) tokens and for the overall number of tokens observed.

Position. We create a set of binary features marking the position of the page within the book (appearance in the first ten pages, last ten pages, and in which quintile it appears), and its real-valued positional ratio within the book (page number divided by the total pages).

Page Sequence. Not all books distinguish frontmatter from the main content with roman numerals; to address this, we identify the page with the first marked page number and create a feature that captures whether a page appears before or after that first marked page.

TextTiling While all words are not indicative of the categories on their own, they can provide a natural segmentation of the book into discrete discourse chunks, in that the language that characterizes a given main content section may differ from that within an introduction (and certainly from more structured sections like indices or tables of contents). To capture this, we create a feature for each page derived from TextTiling (Hearst, 1997): for a given page at position i, we calculate the cosine similarity between the intervals [page$_1$, page$_{i-1}$] and [page$_i$, page$_n$].

The feature classes above total 172 features for each individual page. When representing a page as input to the models below, we also conjoin information about all pages within a window of three pages around the target page; each page is thus represented by a total of 7 × 172 distinct features. All non-binary features are standardized to standard normals, whose means and variances are estimated using the distribution observed in the training-only partition of the data.

3.2 Models

We compare three different model classes: a random forest (Breiman, 2001), which can capture complex nonlinearities in the feature space but is constrained to make independent predictions; an ℓ_2-regularized conditional random field (Lafferty et al., 2001), which can account for temporal dependencies in the predictions but is limited to linear relationships; and a bidirectional sequence labeling LSTM (Graves, 2012; Ma and Hovy, 2016), which can reason over sequential information while also capturing more complex non-linearities. The observed input to all methods for each page x_i is the same feature representation $f(x_i)$; the CRF also includes information about label transition features, decoding the entire sequence using Viterbi decoding; and the bidirectional LSTM captures persistent state information for each page as two H-dimensional hidden layers, one for the forward direction h_f and one for

Method	Accuracy	Macro precision	Macro recall	Macro F
Majority class	0.888	0.089	0.100	0.094
Random Forest	0.959 [0.947, 0.969]	0.866 [0.831, 0.894]	0.593 [0.555, 0.632]	0.677 [0.641, 0.715]
CRF	0.940 [0.915, 0.959]	0.654 [0.615, 0.695]	0.744 [0.683, 0.835]	0.686 [0.644, 0.740]
BiLSTM	0.958 [0.947, 0.968]	0.776 [0.741, 0.807]	0.670 [0.630, 0.709]	0.714 [0.679, 0.747]

Table 1: Full segment labeling, along with 95% bootstrap confidence intervals.

the backward direction h_b (we set $H = 25$ in these experiments). Predictions for each time step i are made using the vector concatenation of $[h^i_f; h^i_b]$.

4 Evaluation

We compare the performance of the three models described above at the task of page-level labeling: both the multiclass classification problem of predicting which of the 10 labels applies to each page, and the binary task of {TEXT, NON-TEXT} prediction, in which the nine front- and backmatter labels are collapsed into the single label NON-TEXT; while the former allows access to fine-grained categories of (e.g.) indices and tables of contents, the latter covers the common scenario where researchers are interested only in isolating where the core text begins and ends.

Experimentally, we divide the full training data into two partitions: a training-only partition of 400 books, on which we experiment with feature and model development, and a test partition of the remaining 655 books. All results presented below are the result of tenfold cross-validation on the test partition. Each fold trains on $\frac{8}{10}$ of the test data, uses $\frac{1}{10}$ of the 655 books as development for model selection (e.g., to optimize the ℓ_2 regularization parameter for the CRF, terminate training for the BiLSTM, and optimize the depth of the random forest), and uses $\frac{1}{10}$ of the 655 for test. We supplement each training fold with the 400 books from the training-only partition, but this data is never used for evaluation below.

In total, we evaluate the performance on 655 books and calculate 95% confidence intervals for each metric using the non-parametric bootstrap, drawing $B = 10,000$ resamples of books (not individual pages) in order to account for the statistical dependence between page-level predictions.

4.1 Full segment labeling

Table 1 presents the results for full multiclass segment labeling. To contextualize these results, we also provide a simple baseline of predicting the majority class (TEXT) for all pages; since most pages in a book are core content, this yields a high absolute accuracy against which to compare, but a low macro precision/recall/F score (which evenly weights the importance of each class).

All three methods achieve relatively similar performance when measured by absolute accuracy (though the room for improvement over the baseline is small). When treating all classes as equally important and measuring by the macro F score, both sequence labeling methods (CRF and bidirectional LSTM) show slight improvements over the independent predictions of a random forest, but not significantly so, suggesting that the feature representation of the book (which they all share as identical input) is perhaps a strong enough signal that mitigates the label dependencies.

Category	Precision	Recall	F	True n
Title	0.782	0.751	0.766	887
Dedication	0.630	0.489	0.551	188
Pubinfo	0.697	0.590	0.639	261
Ad card	0.642	0.516	0.572	717
TOC	0.844	0.842	0.843	1,139
Preface	0.736	0.643	0.686	2,253
Text	0.971	0.991	0.981	160,721
Index	0.894	0.628	0.737	2,586
Appendix	0.688	0.412	0.515	2,460
N/A	0.894	0.801	0.845	9,791

Table 2: Individual category results, BiLSTM.

Table 2 lists the precision, recall and F-score results by category for the best-performing model (bidirectional LSTM). Several categories are worth calling out: the precision and recall for recognizing table of contents is high (≥ 0.84 for both metrics), suggesting that this method may provide a solid foundation for work in book structure extraction that relies on an identified table of contents in order to recognize the idiosyncratic structure of books. Title page and index recognition are also relatively high (0.89 precision/0.63 recall); what these three categories have in common are strong structural features (the distribution of ink and whitespace on the page; regularities in the numbers and the degree of alphabetization).

While dedications and publication information

Method	Accuracy	Macro P	Macro R	Macro F
Majority class	0.888	0.444	0.500	0.470
Chop	0.857	0.804	0.692	0.725
Random Forest	0.966 [0.953, 0.976]	0.947 [0.937, 0.956]	0.877 [0.836, 0.911]	0.908 [0.877, 0.931]
CRF	0.963 [0.949, 0.973]	0.887 [0.843, 0.921]	0.920 [0.896, 0.941]	0.902 [0.872, 0.926]
BiLSTM	0.965 [0.953, 0.974]	0.938 [0.924, 0.951]	0.881 [0.843, 0.913]	0.907 [0.880, 0.928]

Table 3: Content/non-content labeling, along with 95% bootstrap confidence intervals.

are both relatively infrequent (often occupying a single page in a book), the greatest point of confusion is in separating the main content from the structurally similar pages that typically precede it (in the preface) and follow it (in the appendix). While confusion between PREFACE/TEXT and APPENDIX/TEXT account for most of the errors, figure 4 illustrates several difficult cases and exemplary mistakes in the other categories: fig. 4a is a page that blurs the line between an index and table of contents; fig. 4b is an advertisement for a book "in the press and speedily will be published"; and fig. 4c is a dedication that, without strong lexical indicators, is mistaken for a title page.

In order to understand the contribution that individual features make on the predictions, we carry out an ablation test for each feature class, in which we remove a feature class from the model and perform exactly the same training and test procedure as described in section 4: we train a model on the training fold supplemented with the 400 books in the training partition, perform hyperparameter optimization on development data, and report accuracy on the held-out test fold, repeating ten times, once for each fold in cross-validation.

Feature	Δ Macro F-score
–Keywords	-0.15
–Position	-0.03
–Density	-0.02
–Window	-0.01
–Roman	-0.01
–Letter	-0.01
–LIS	0.00
–TextTiling	0.00
–Page sequence	0.00
–Alphabetical	0.01

Table 4: Feature ablation results for the BiLSTM model, illustrating the change in macro F-score that results by removing a given feature class from the full model.

The simplest features are the most informative: the small set of keywords learned from the training partition (which include common section labels like *preface*, *content*, *index*, *advertisement*, other informative markers such as *dedicated*, *copyright*, and currency markers like $, £), the position of the page in the book, and the density of characters (including words and numbers) on the page.

4.2 Content/non-content segment labeling

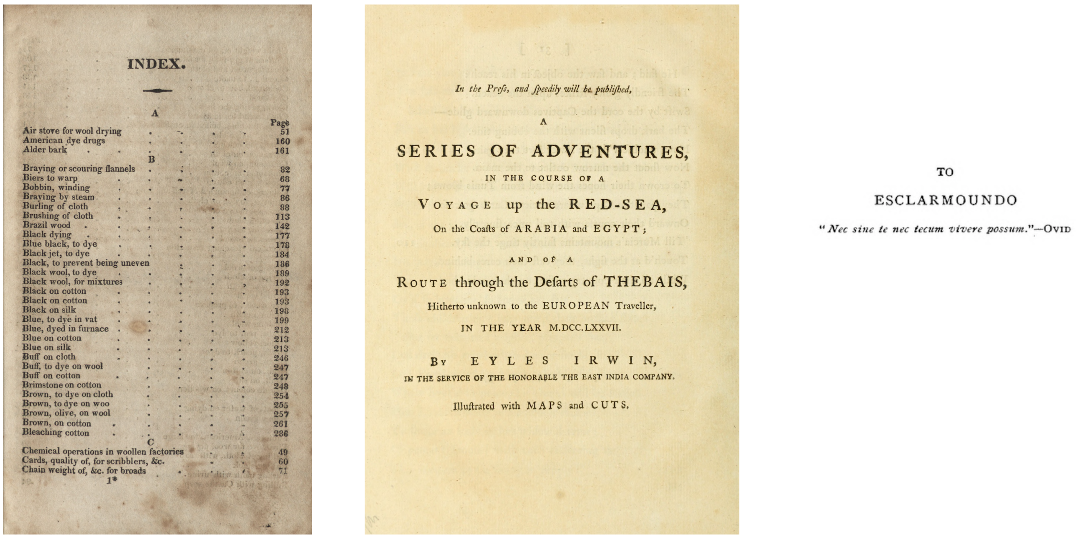
In order to assess the ability of these different features and models to demarcate the core text of a work, we binarize the multiclass label, assigning TEXT to all pages labeled TEXT in the multiclass setting, and NON-TEXT to all other pages. We train all three classifiers again on these binarized labels and repeat the training procedure for each model outlined in section 4.

Table 3 shows the results for the binary task of {TEXT, NOT-TEXT} prediction. Here again we contextualize these results with two simpler baselines: a simple majority class predictor (always predict TEXT), and a model that identifies the average start and end positions in a book for the first and last text page (respectively) within the training data, and predicts TEXT for pages within that range (roughly within the [0.10, 0.94] interval), and NON-TEXT for all pages outside of it. This corresponds to a heuristic that chops off the first 10% of a book and the last 6% as NON-TEXT.

The chop heuristic performs worse than the majority class predictor in terms of absolute accuracy, but improves over the class-balanced macro scores. All three feature-rich models show substantial improvements over all metrics, but are indistinguishable from each other, each achieving nearly identical performance. For this reduced purpose, any of the three classifiers are sufficient for segmenting TEXT from NON-TEXT, even a random forest making independent predictions for each page.

5 Related work

The work described here has points of intersection with several other threads of research. The most direct originates in work that grows out of the INEX and ICDAR book structure extraction com-



INDEX.

1*

(a) Index/TOC.

In the Preſs, and ſpeedily will be publiſhed,

A

SERIES OF ADVENTURES,

IN THE COURSE OF A

VOYAGE up the RED-SEA,

On the Coaſts of ARABIA and EGYPT;

AND OF A

ROUTE through the Deſarts of THEBAIS,

Hitherto unknown to the EUROPEAN Traveller,

IN THE YEAR M.DCC.LXXVII.

BY EYLES IRWIN,

IN THE SERVICE OF THE HONORABLE THE EAST INDIA COMPANY.

Illuſtrated with MAPS and CUTS.

(b) Ad predicted as text.

TO

ESCLARMOUNDO

"*Nec sine te nec tecum vivere possum.*"—OVID

(c) Dedication predicted as title.

Figure 4: Exemplary mistakes in prediction.

petitions (Kazai et al., 2009, 2010; Doucet et al., 2011, 2013), in which participants are challenged to recognize the fine-grained structure present in documents (recognizing, for example, that the current article has sections entitled "Abstract," "Introduction," "Data, "References," etc.). The most successful systems recognize structure by parsing the table of contents (Déjean and Meunier, 2005, 2009; Wu et al., 2013; Gao et al., 2009) rather than relying on the content of the book itself. Our work primarily differs in the fundamental design choice of prescribing a fix set of categories into which we classify pages (in order to enable comparison across documents) rather than prioritizing the idiosynractic structure of a book (which is useful for generating new tables of contents).

Given the relatively constant page-level categories that printers use to describe book design, we formulate our problem as a classification task into a set of pre-established categories. An alternative is to take an unsupervised approach, and learn the set of categories empirically from the data; this general problem of book segmentation in its unlabeled form shares functional similarity with other work in general unsupervised topic or discourse segmentation (Hearst, 1997; Utiyama and Isahara, 2001; Chen et al., 2009)—most notably, the work of Eisenstein and Barzilay (2008) (for whom the section labels may be considered a form of "cue words" akin to discourse markers). Given the amount of data in large-scale book collections, we see this as an interesting path forward (either in a fully unsupervised or semi-supervised setting); an unsupervised approach that includes aspects of metadata such as country of publication or publisher may also be fruitful in accommodating variation in printer's rules as a function of time and geographical location (books by French publishers, for example, often place the table of contents at the back of the book).

As figure 2 illustrates, one of the primary challenges that we face with the labeled segmentation of books is the degraded nature of the input; unlike contemporary business documents for which OCR is largely a solved problem, historical books present several challenges due to their binding, age, and significant variation in font and printing. Much work has focused on overcoming these limitations from several perspectives, including creating ground truth for historical books (Papadopoulos et al., 2013), bootstrapping their alignment with existing resources (Feng and Manmatha, 2006; Yalniz and Manmatha, 2011), exploiting the fact that books often have multiple scans or printings that could be leveraged (Smith et al., 2011; Wemhoener et al., 2013) or developing methods that account for variation in the printing process (Berg-Kirkpatrick et al., 2013; Berg-Kirkpatrick and Klein, 2014).

Large-scale book corpora are increasingly being used as the raw material for linguistic analysis, especially those focused on measuring historical change (Hamilton et al., 2016a,b; Kulkarni et al., 2015; Mitra et al., 2014; Mihalcea and Nastase, 2012; Kim et al., 2014). These studies use not only the observed word frequencies provided by the Google Ngram dataset, but also increasingly structured representations of language as well (Lin et al., 2012; Goldberg and Orwant, 2013). The task of labeled book segmentation may be helpful in reducing the noise inherent in the use of statistics aggregated from these large datasets—both in terms of filtering out the 11% of all pages that are not the core content (e.g., such as indices), and also in grounding the text at the appropriate date for historical analysis (such as deriving statistics only from the core content, and not from an introduction written years afterward).

6 Conclusion

We introduce in this work the task of *book structure labeling*, the problem of assigning to each page in a printed book its membership in one of a set of predetermined categories. In annotating a large dataset of books, we are able to empirically assess the ability to accurately segment and label books from a range of historical time periods.

The ten categories that form our typology are drawn from printers' guides and informed by contemporary criticism, but still reflect our historical present; while we have in part let our categories be shaped by our experience labeling texts (so that we have preserved in our annotations historical categories not in contemporary use, such as SUBSCRIBERS), we recognize that the act of categorization glosses over meaningful distinctions—for example, while we have grouped sections marked ADVERTISEMENT, TO THE READER, PREFACE, INTRODUCTION, FOREWORD and others into the single category of PREFACE, such labels may have historically significant differences that may be worth preserving for some analyses. Nevertheless, we expect the coarse distinctions we outline here to occasion research that requires access to those broad categories. Potential uses of this work include using the categories directly to answer questions in book history (e.g., charting the historical prevalence of advertisements and their variation across time), improving the task of idiosyncratic structure detection by identifying tables of contents, and identifying the fine-grained topics of books by parsing recognized indices.

In this work, we deliberately focus on the resource-deficient scenario most commonly encountered by researchers working with large book corpora, in which books are represented as the output of errorful OCR. In providing a labeled dataset for others to use, we hope to encourage other work that reasons about the structure present in alternative representations (such as images) as well.

Acknowledgments

Many thanks to the anonymous reviewers and Hannah Alpert-Abrams and for their valuable feedback, and to the HathiTrust Research Center for their assistance in enabling this work. The research reported in this article was supported by a grant from the Digital Humanities at Berkeley initiative and resources provided by NVIDIA.

References

Jinho Baik, Percy Deift, and Kurt Johansson. 1999. On the distribution of the length of the longest increasing subsequence of random permutations. *J. Amer. Math. Soc.*, 4.

David Bamman, Michelle Carney, Jon Gillick, Cody Hennesy, and Vijitha Sridhar. 2017. Estimating the date of first publication in a large-scale digital library. In *Proceedings of the ACM/IEEE Annual Joint Conference on Digital Libraries*.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A Bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 370–379, Baltimore, Maryland. Association for Computational Linguistics.

Taylor Berg-Kirkpatrick, Greg Durrett, and Dan Klein. 2013. Unsupervised transcription of historical documents. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 207–217, Sofia, Bulgaria. Association for Computational Linguistics.

Taylor Berg-Kirkpatrick and Dan Klein. 2014. Improved typesetting models for historical OCR. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 118–123, Baltimore, Maryland. Association for Computational Linguistics.

Leo Breiman. 2001. Random forests. *Mach. Learn.*, 45(1):5–32.

Harr Chen, S. R. K. Branavan, Regina Barzilay, and David R. Karger. 2009. Global models of document structure using latent permutations. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 371–379, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hervé Déjean and Jean-Luc Meunier. 2005. Structuring documents according to their table of contents. In *Proceedings of the 2005 ACM Symposium on Document Engineering*, DocEng '05, pages 2–9, New York, NY, USA. ACM.

Hervé Déjean and Jean-Luc Meunier. 2009. On tables of contents and how to recognize them. *International Journal of Document Analysis and Recognition (IJDAR)*, 12(1):1–20.

Antoine Doucet, Gabriella Kazai, and Jean-Luc Meunier. 2011. ICDAR 2011 book structure extraction competition. In *ICDAR*.

Antoine Doucet, Gabriella Kazai, and Jean-Luc Meunier. 2013. Overview of the ICDAR 2013 competition on book structure extraction.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 334–343, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shaolei Feng and R Manmatha. 2006. A hierarchical, HMM-based automatic evaluation of OCR accuracy for a digital library of books. In *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pages 109–118. ACM.

L. Gao, Z. Tang, X. Lin, X. Tao, and Y. Chu. 2009. Analysis of book documents' table of content based on clustering. In *2009 10th International Conference on Document Analysis and Recognition*, pages 911–915.

Gérard Genette. 1987. *Paratexts: Thresholds of Interpretation*. Cambridge University Press.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of English books. In *Second Joint Conference on Lexical and Computational Semantics (*SEM)*, volume 1, pages 241–247.

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016a. Cultural shift or linguistic drift? comparing two computational models of semantic change. In *Proceedings of EMNLP*.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016b. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501, Berlin, Germany. Association for Computational Linguistics.

Marti A. Hearst. 1997. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.*, 23(1):33–64.

Gabriella Kazai, Antoine Doucet, Marijn Koolen, and Monica Landoni. 2009. Overview of the inex 2009 book track. In *INEX*.

Gabriella Kazai, Antoine Doucet, Marijn Koolen, and Monica Landoni. 2010. *Overview of the INEX 2009 Book Track*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. *arXiv preprint arXiv:1405.3515*.

Matthew Kirschenbaum and Sarah Werner. 2014. Digital scholarship and digital studies: The state of the discipline. *Book History*, 17.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 625–635, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Marshall Lee. 2009. *Bookmaking: Editing, Design, Production (Third Edition)*. W. W. Norton & Company.

Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic annotations for the Google Books ngram corpus. In *Proceedings of the ACL 2012 system demonstrations*, pages 169–174. Association for Computational Linguistics.

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

Jean-Baptiste Michel, Yuan Kui Shen, Aviva P. Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2010. Quantitative Analysis of Culture Using Millions of Digitized Books. *Science*.

Rada Mihalcea and Vivi Nastase. 2012. Word epoch disambiguation: Finding how words change over time. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 259–263. Association for Computational Linguistics.

Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee, and Pawan Goyal. 2014. That's sick dude!: Automatic identification of word sense change across different timescales. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1020–1029, Baltimore, Maryland. Association for Computational Linguistics.

Christos Papadopoulos, Stefan Pletschacher, Christian Clausner, and Apostolos Antonacopoulos. 2013. The IMPACT dataset of historical document images. In *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing*, HIP '13, pages 123–130, New York, NY, USA. ACM.

David A. Smith, R. Manmatha, and James Allan. 2011. Mining relational structure from millions of books: Position paper. In *Proceedings of the 4th ACM Workshop on Online Books, Complementary Social Media and Crowdsourcing*, BooksOnline '11, pages 49–54, New York, NY, USA. ACM.

Ted Underwood. 2016. The life cycles of genres. *Cultural Analytics*.

Ted Underwood. 2017. Datamunging Github repository. `https://github.com/tedunderwood/DataMunging`.

Masao Utiyama and Hitoshi Isahara. 2001. A statistical model for domain-independent text segmentation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 499–506, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Wemhoener, Ismet Zeki Yalniz, and R Manmatha. 2013. Creating an improved version using noisy OCR from multiple editions. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 160–164. IEEE.

Sarah Werner. 2012. Where material book culture meets digital humanities.

Matthew Wilkens. 2013. The geographic imagination of Civil War-era American fiction. *American Literary History*, 25(4):803–840.

Adrian Wilson. 1993. *Design of Books*. Chronicle Books.

Z. Wu, P. Mitra, and C. L. Giles. 2013. Table of contents recognition and extraction for heterogeneous book documents. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1205–1209.

Ismet Zeki Yalniz and Raghavan Manmatha. 2011. A fast alignment scheme for automatic OCR evaluation of books. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 754–758. IEEE.

Cross-lingual, Character-Level Neural Morphological Tagging

Ryan Cotterell[∂] and **Georg Heigold**[ʁ]

[∂]Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD, USA
[ʁ]German Research Center for Artificial Intelligence, Saarbrücken, SL, GER
ryan.cotterell@jhu.edu georg.heigold@gmail.com

Abstract

Even for common NLP tasks, sufficient supervision is not available in many languages—morphological tagging is no exception. In the work presented here, we explore a transfer learning scheme, whereby we train character-level recurrent neural taggers to predict morphological taggings for high-resource languages and low-resource languages together. Learning joint character representations among multiple related languages successfully enables knowledge transfer from the high-resource languages to the low-resource ones, improving accuracy by up to 30%.

1 Introduction

State-of-the-art morphological taggers require thousands of annotated sentences to train. For the majority of the world's languages, however, sufficient large-scale annotation is not available and obtaining it would often be infeasible. Accordingly, an important road forward in low-resource NLP is the development of methods that allow for the training of high-quality tools from smaller amounts of data. In this work, we focus on transfer learning—we train a recurrent neural tagger for a low-resource language jointly with a tagger for a related high-resource language. Forcing the models to share character-level features among the languages allows large gains in accuracy when tagging the low-resource languages, while maintaining (or even improving) accuracy on the high-resource language.

Recurrent neural networks constitute the state of the art for a myriad of tasks in NLP, e.g., multilingual part-of-speech tagging (Plank et al., 2016), syntactic parsing (Dyer et al., 2015; Zeman et al., 2017), morphological paradigm completion (Cotterell et al., 2016, 2017) and language modeling (Sundermeyer et al., 2012; Melis et al., 2017); recently, such models have also improved morphological tagging (Heigold et al., 2016, 2017). In addition to increased performance over classical approaches, neural networks also offer a second advantage: they admit a clean paradigm for multitask learning. If the learned representations for all of the tasks are embedded jointly into a shared vector space, the various tasks reap benefits from each other and often performance improves for all (Collobert et al., 2011b). We exploit this idea for language-to-language transfer to develop an approach for cross-lingual morphological tagging.

We experiment on 18 languages taken from four different language families. Using the Universal Dependencies treebanks, we emulate a low-resource setting for our experiments, e.g., we attempt to train a morphological tagger for Catalan using primarily data from a related language like Spanish. Our results demonstrate the successful transfer of morphological knowledge from the high-resource languages to the low-resource languages without relying on an externally acquired bilingual lexicon or bitext. We consider both the single- and multi-source transfer case and explore how similar two languages must be in order to enable high-quality transfer of morphological taggers.[1]

2 Morphological Tagging

Many languages in the world exhibit rich inflectional morphology: the form of individual words mutates to reflect the syntactic function. For example, the Spanish verb *soñar* will appear as *sueño* in the first person present singular, but *soñáis* in the second person present plural, depending on the bundle of syntaco-semantic attributes associated with

[1] While we only experiment with languages in the same family, we show that closer languages within that family are better candidates for transfer. We remark that future work should consider the viability of more distant language pairs.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 748–759
Copenhagen, Denmark, September 7–11, 2017.

POS=D CASE=NOM NUM=PL	POS=N CASE=NOM NUM=PL	POS=N CASE=NOM NUM=PL GEN=FEM	POS=A CASE=NOM NUM=PL	POS=N CASE=NOM NUM=SG	POS=P	POS=N CASE=ACC NUM=SG
Все	счастливые	семьи	похожи	друг	на	друга...
All	happy	families	are similar	to each other		

Figure 1: Example of a morphologically-tagged sentence in Russian using the annotation scheme provided in the UD dataset.

	PRESENT INDICATIVE		PAST INDICATIVE	
	SINGULAR	PLURAL	SINGULAR	PLURAL
1	*sueño*	*soñamos*	*soñé*	*soñamos*
2	*sueñas*	*soñáis*	*soñaste*	*soñasteis*
3	*sueña*	*sueñan*	*soñó*	*soñaron*

Table 1: Partial inflection table for the Spanish verb *soñar*

the given form (in a sentential context). For concreteness, we list a more complete table of Spanish verbal inflections in Table 1. Note that some languages, e.g., Archi, Northeast Caucasian language, display a veritable cornucopia of potential forms with the size of the verbal paradigm exceeding 10,000 (Kibrik, 1998).

Standard NLP annotation, e.g., the scheme in Sylak-Glassman et al. (2015), marks forms in terms of *universal* key-attribute pairs, e.g., the first person present singular is represented as [pos=V, per=1, num=SG, tns=PRES]. This bundle of key-attributes pairs is typically termed a morphological tag and we may view the goal of morphological tagging to label each word in its sentential context with the appropriate tag (Oflazer and Kuruöz, 1994; Hajič and Hladká, 1998). As the part-of-speech (POS) is a component of the tag, we may view morphological tagging as a strict generalization of POS tagging, where we have significantly refined the set of available tags. All of the experiments in this paper make use of the universal morphological tag set available in the Universal Dependencies (UD) (Nivre et al., 2016). As an example, we have provided a Russian sentence with its UD tagging in Figure 1.

Transferring Morphology. The transfer of morphology is arguably more dependent on the relatedness of the languages in question than other annotations in NLP, such as POS and named entity recognition (NER). POS lends itself nicely to a universal annotation scheme (Petrov et al., 2012) and traditional NER is limited to a small number of cross-linguistically compliant categories, e.g., PERSON and PLACE. Even universal dependency arcs employ cross-lingual labels (Nivre et al., 2016).

Morphology, on the other hand, typically requires more fine-grained annotation, e.g., grammatical case and tense. It is often the case that one language will make a semantic distinction in the form (or at all) that another does not. For example, the Hungarian noun overtly marks 17 grammatical cases and Slavic verbs typically distinguish two aspects through morphology, while English marks none of these distinctions. If the word form in the source language does not overtly mark a grammatical category in the target language, it is nigh-impossible to expect a successful transfer. For this reason, much of our work focuses on the transfer of related languages—specifically exploring *how* close two languages must be for a successful transfer. Note that the language-specific nature of morphology does not contradict the universality of the annotation; each language may mark a different subset of categories, i.e., use a different set of the universal keys and attributes, but there is a single, universal set, from which the key-attribute pairs are drawn. See Newmeyer (2007) for a linguistic treatment of cross-lingual annotation.

Notation. We will discuss morphological tagging in terms of the following notation. We will consider two (related) languages: a high-resource *source* language ℓ_s and a low-resource *target* language ℓ_t. Each of these languages will have its own (potentially overlapping) set of morphological tags, denoted $\mathcal{T}_s$ and $\mathcal{T}_t$, respectively. We will work with the union of both sets $\mathcal{T} = \mathcal{T}_s \cup \mathcal{T}_t$. An individual tag $m_i = [k_1{=}v_1, \ldots, k_M{=}v_M] \in \mathcal{T}$ is comprised of universal keys and attributes, i.e., the pairs (k_i, v_i) are completely language-agnostic. In the case where a language does not mark a distinction, e.g., case on English nouns, the corresponding keys are excluded from the tag. Typically, $|\mathcal{T}|$ is large (see Table 3). We denote the set of training sentences for the high-resource source language as $\mathcal{D}_s$ and the set of training sentences for the low-resource target language as $\mathcal{D}_t$. In the experimental section, we will also consider a multi-source setting where we have multiple high-resource languages, but, for ease of explication, we stick to the single-source case in the development of the model.

3 Character-Level Neural Transfer

Our formulation of transfer learning builds on work in multi-task learning (Caruana, 1997; Collobert et al., 2011b). We treat each individual language as a task and train a joint model for all the tasks. We first discuss the current state of the art in morphological tagging: a character-level recurrent neural network. After that, we explore three augmentations to the architecture that allow for the transfer learning scenario. All of our proposals force the embedding of the characters for both the source and the target language to share the same vector space, but involve different mechanisms, by which the model may learn language-specific features.

3.1 Character-Level Neural Networks

Character-level neural networks currently constitute the state of the art in morphological tagging (Heigold et al., 2017). We draw on previous work in defining a conditional distribution over taggings $\boldsymbol{t}$ for a sentence $\boldsymbol{w}$ of length $|\boldsymbol{w}| = N$ as

$$p_{\boldsymbol{\theta}}(\boldsymbol{t} \mid \boldsymbol{w}) = \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(t_i \mid \boldsymbol{w}), \tag{1}$$

which may be seen as a 0^{th} order conditional random field (CRF) (Lafferty et al., 2001) with parameter vector $\boldsymbol{\theta}$.[2] Importantly, this factorization of the distribution $p_{\boldsymbol{\theta}}(\boldsymbol{t} \mid \boldsymbol{w})$ also allows for efficient exact decoding and marginal inference in $\mathcal{O}(|\mathcal{T}| \cdot N)$-time, but at the cost of not admitting any explicit interactions in the output structure, i.e., between adjacent tags.[3] We parameterize the distribution over tags at each time step as

$$p_{\boldsymbol{\theta}}(t_i \mid \boldsymbol{w}) = \text{softmax}\left(W\boldsymbol{e}_i + \boldsymbol{b}\right), \tag{2}$$

where $W \in \mathbb{R}^{|\mathcal{T}| \times n}$ is an embedding matrix, $\boldsymbol{b} \in \mathbb{R}^{|\mathcal{T}|}$ is a bias vector and positional embeddings $\boldsymbol{e}_i$[4] are taken from a concatenation of the output of two long short-term memory recurrent neural networks (LSTMs) (Hochreiter and Schmidhuber, 1997), folded forward and backward, respectively, over a sequence of input vectors. This constitutes a bidirectional LSTM (Graves and Schmidhuber, 2005). We define the positional embedding vector as follows

$$\boldsymbol{e}_i = \left[\text{LSTM}(\boldsymbol{v}_{1:i}); \text{LSTM}(\boldsymbol{v}_{N,i+1})\right], \tag{3}$$

where each $\boldsymbol{v}_i \in \mathbb{R}^n$ is, itself, a word embedding. Note that the function LSTM returns the *last* final hidden state vector of the network. This architecture is the *context* bidirectional recurrent neural network of Plank et al. (2016). Finally, we derive each word embedding vector $\boldsymbol{v}_i$ from a character-level bidirectional LSTM embedder. Namely, we define each word embedding as the concatenation

$$\begin{aligned}\boldsymbol{v}_i = \Big[&\text{LSTM}\left(\langle c_{i_1}, \ldots, c_{i_{M_i}}\rangle\right); \\ &\text{LSTM}\left(\langle c_{i_{M_i}}, \ldots, c_{i_1}\rangle\right)\Big].\end{aligned} \tag{4}$$

In other words, we run a bidirectional LSTM over the character stream. This bidirectional LSTM is the *sequence* bidirectional recurrent neural network of Plank et al. (2016). Note a concatenation of the sequence of character symbols $\langle c_{i_1}, \ldots, c_{i_{M_i}}\rangle$ results in the word string w_i. Each of the M_i characters c_{i_k} is a member of the set Σ. We take Σ to be the union of sets of characters in the languages considered.

We direct the reader to Heigold et al. (2017) for a more in-depth discussion of this and various additional architectures for the computation of $\boldsymbol{v}_i$; the architecture we have presented in Equation (5) is competitive with the best performing setting in Heigold et al.'s study.

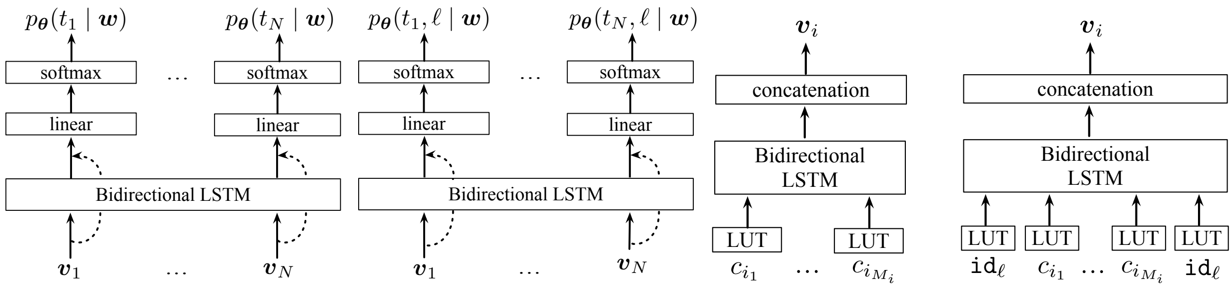
3.2 Cross-Lingual Morphological Transfer as Multi-Task Learning

Cross-lingual morphological tagging may be formulated as a multi-task learning problem. We seek to learn a set of shared character embeddings for taggers in both languages together through optimization of a joint loss function that combines the high-resource tagger and the low-resource one. The first loss function we consider is the following:

[2] The parameter vector $\boldsymbol{\theta}$ is a vectorization of all the parameters discussed below.

[3] As an aside, it is quite interesting that a model with the factorization in Equation (1) outperforms the MARMOT model (Müller et al., 2013), which focused on modeling higher-order interactions between the morphological tags, e.g., they employ up to a (pruned) 3rd order CRF. That such a model achieves state-of-the-art performance indicates, however, that richer source-side features, e.g., those extracted by our character-level neural architecture, are more important for morphological tagging than higher-order tag interactions, which come with the added unpleasantness of exponential (in the order) decoding.

[4] Note that $|\boldsymbol{e}_i| = n$; see §4.4 for the exact values used in the experimentation.



(a) Vanilla architecture for neural morphological tagging.

(b) Joint morphological tagging and language identification.

(c) Character-level biLSTM embedder.

(d) Language-specific biLSTM embedder.

Figure 2: We depict four subarchitectures used in the models we develop in this work. Combining (a) with the character embeddings in (c) gives the vanilla morphological tagging architecture of Heigold et al. (2017). Combining (a) with (d) yields the language-universal softmax architecture and (b) and (c) yields our joint model for language identification and tagging.

$$\mathcal{L}_{multi}(\boldsymbol{\theta}) = \sum_{(\boldsymbol{t},\boldsymbol{w})\in\mathcal{D}_s} \log p_{\boldsymbol{\theta}}(\boldsymbol{t} \mid \boldsymbol{w}, \ell_s) + \sum_{(\boldsymbol{t},\boldsymbol{w})\in\mathcal{D}_t} \log p_{\boldsymbol{\theta}}\left(\boldsymbol{t} \mid \boldsymbol{w}, \ell_t\right). \quad (5)$$

Crucially, our cross-lingual objective forces both taggers to share part of the parameter vector $\boldsymbol{\theta}$, which allows it to represent morphological regularities between the two languages in a common embedding space and, thus, enables transfer of knowledge. This is no different from monolingual multi-task settings, e.g., jointly training a chunker and a tagger for the transfer of syntactic information (Collobert et al., 2011b). We point out that, in contrast to our approach, almost all multi-task transfer learning, e.g., for dependency parsing (Guo et al., 2016), has shared word-level embeddings rather than character-level embeddings. See §6 for a more complete discussion.

We consider two parameterizations of this distribution $p_{\boldsymbol{\theta}}(t_i \mid \boldsymbol{w}, \ell)$. First, we modify the initial character-level LSTM embedding such that it also encodes the identity of the language. Second, we modify the softmax layer, creating a language-specific softmax.

Language-Universal Softmax. Our first architecture has one softmax, as in Equation (2), over all morphological tags in $\mathcal{T}$ (shared among all the languages). To allow the architecture to encode morphological features specific to one language, e.g., the third person present plural ending in Spanish is *-an*, but *-ão* in Portuguese, we modify the creation of the character-level embeddings. Specifically, we augment the character alphabet Σ with a distinguished symbol that indicates the language: $\mathtt{id}_\ell$. We then pre- and postpend this symbol to the character stream for every word before feeding the characters into the bidirectional LSTM Thus, we arrive at the new *language-specific* word embeddings,

$$\boldsymbol{v}_i^\ell = \Big[\text{LSTM}\left(\langle \mathtt{id}_\ell, c_{i_1}, \ldots, c_{i_{M_i}}, \mathtt{id}_\ell \rangle\right); \text{LSTM}\left(\langle \mathtt{id}_\ell, c_{i_{M_i}}, \ldots, c_{i_1}, \mathtt{id}_\ell \rangle\right)\Big]. \quad (6)$$

This model creates a language-specific embedding vector $\boldsymbol{v}_i$, but the individual embeddings for a given character are shared among the languages jointly trained on. The remainder of the architecture is held constant.

Language-Specific Softmax. Next, inspired by the architecture of Heigold et al. (2013), we consider a language-specific softmax layer, i.e., we define a new output layer for every language:

$$p_{\boldsymbol{\theta}}\left(t_i \mid \boldsymbol{w}, \ell\right) = \text{softmax}\left(W_\ell \boldsymbol{e}_i + \boldsymbol{b}_\ell\right), \quad (7)$$

where $W_\ell \in \mathbb{R}^{|\mathcal{T}|\times n}$ and $\boldsymbol{b}_\ell \in \mathbb{R}^{|\mathcal{T}|}$ are now *language-specific*. In this architecture, the embeddings $\boldsymbol{e}_i$ are the same for all languages—the model has to learn language-specific behavior exclusively through the output softmax of the tagging LSTM.

Joint Morphological Tagging and Language Identification. The third model we exhibit is a joint architecture for tagging and language identification. We consider the following loss function:

$$\mathcal{L}_{joint}(\boldsymbol{\theta}) = \sum_{(\boldsymbol{t},\boldsymbol{w})\in\mathcal{D}_s} \log p_{\boldsymbol{\theta}}(\ell_s, \boldsymbol{t} \mid \boldsymbol{w}) + \sum_{(\boldsymbol{t},\boldsymbol{w})\in\mathcal{D}_t} \log p_{\boldsymbol{\theta}}\left(\ell_t, \boldsymbol{t} \mid \boldsymbol{w}\right), \quad (8)$$

where we factor the joint distribution as

$$p_{\boldsymbol{\theta}}\left(\ell, \boldsymbol{t} \mid \boldsymbol{w}\right) = p_{\boldsymbol{\theta}}\left(\ell \mid \boldsymbol{w}\right) \cdot p_{\boldsymbol{\theta}}\left(\boldsymbol{t} \mid \boldsymbol{w}, \ell\right). \quad (9)$$

Romance				Slavic			
lang	train	dev	test	lang	train	dev	test
(ca)	13123	1709	1846	(bg)	8907	1115	1116
(es)	14187	1552	274	(cs)	61677	9270	10148
(fr)	14554	1596	298	(pl)	6800	7000	727
(it)	12837	489	489	(ru)	4029	502	499
(pt)	8800	271	288	(sk)	8483	1060	1061
(ro)	7141	1191	1191	(uk)	200	30	25
Germanic				Uralic			
lang	train	dev	test	lang	train	dev	test
(da)	4868	322	322	(et)	14510	1793	1806
(no)	15696	2410	1939	(fi)	12217	716	648
(sv)	4303	504	1219	(hu)	1433	179	188

Table 2: Number of tokens in each of the train, development and test splits (organized by language family).

Just as before, we define $p_{\boldsymbol{\theta}}(\boldsymbol{t} \mid \boldsymbol{w}, \ell)$ above as in Equation (7) and we define

$$p_{\boldsymbol{\theta}}(\ell \mid \boldsymbol{w}) = \text{softmax}\left(\boldsymbol{u}^\top \tanh(V\boldsymbol{e}_i)\right), \qquad (10)$$

which is a multi-layer perceptron with a binary softmax (over the two languages) as an output layer; we have added the additional parameters $V \in \mathbb{R}^{|\mathcal{T}|\times n}$ and $\boldsymbol{u} \in \mathbb{R}^{|\mathcal{T}|}$. In the case of multi-source transfer, this is a softmax over the set of languages.

Comparative Discussion. The first two architectures discussed in §3.2 represent two possibilities for a multi-task objective, where we condition on the language of the sentence. The first integrates this knowledge at a lower level and the second at a higher level. The third architecture discussed in §3.2 takes a different tack—rather than conditioning on the language, it predicts it. The joint model offers one interesting advantage over the two architectures proposed. Namely, it allows us to perform a morphological analysis on a sentence where the language is unknown. This effectively alleviates an early step in the NLP pipeline, where language id is performed and is useful in conditions where the language to be tagged may not be known *a-priori*, e.g., when tagging social media data.

While there are certainly more complex architectures one could engineer for the task, we believe we have found a relatively diverse sampling, enabling an interesting experimental comparison. Indeed, it is an important empirical question which architectures are most appropriate for transfer learning. Since transfer learning affords the opportunity to reduce the sample complexity of the "data-hungry" neural networks that currently dominate NLP research, finding a good solution for cross-lingual transfer in state-of-the-art neural models will likely be a boon for low-resource NLP in general.

Romance		Slavic		Germanic		Uralic	
lang	$\lvert\mathcal{T}\rvert$	lang	$\lvert\mathcal{T}\rvert$	lang	$\lvert\mathcal{T}\rvert$	lang	$\lvert\mathcal{T}\rvert$
(ca)	172	(bg)	380	(da)	124	(et)	654
(es)	232	(cs)	2282	(no)	169	(fi)	1440
(fr)	142	(pl)	774	(sv)	155	(hu)	634
(it)	179	(ru)	520				
(pt)	375	(sk)	597				
(ro)	367	(uk)	220				

Table 3: Number of unique morphological tags for each of the experimental languages (organized by language family).

4 Experiments

Empirically, we ask three questions of our architectures. i) How well can we transfer morphological tagging models from high-resource languages to low-resource languages in each architecture? (Does one of the three outperform the others?) ii) How many annotated data in the low-resource language do we need? iii) How closely related do the languages need to be to get good transfer?

4.1 Experimental Languages

We experiment with the language families: Romance (Indo-European), Northern Germanic (Indo-European), Slavic (Indo-European) and Uralic. In the Romance sub-grouping of the wider Indo-European family, we experiment on Catalan (ca), French (fr), Italian (it), Portuguese (pt), Romanian (ro) and Spanish (es). In the Northern Germanic family, we experiment on Danish (da), Norwegian (no) and Swedish (sv). In the Slavic family, we experiment on Bulgarian (bg), Czech (bg), Polish (pl), Russian (ru), Slovak (sk) and Ukrainian (uk). Finally, in the Uralic family we experiment on Estonian (et), Finnish (fi) and Hungarian (hu).

4.2 Datasets

We use the morphological tagging datasets provided by the Universal Dependencies (UD) treebanks (the concatenation of the 4th and 6th columns of the file format) (Nivre et al., 2016). We list the size of the training, development and test splits of the UD treebanks we used in Table 2. Also, we list the number of unique morphological tags in each language in Table 3, which serves as an approximate measure of the morphological complexity each language exhibits. Crucially, the data are annotated in a cross-linguistically consistent manner, such that words in the different languages that have the same syntacto-semantic function have the same bundle of tags (see §2 for a discussion). Potentially, further gains would be possible by using a more universal scheme, e.g., the UNIMORPH scheme.

4.3 Baselines

We consider two baselines in our work. First, we consider the MARMOT tagger (Müller et al., 2013), which is currently the best performing non-neural model. The source code for MARMOT is freely available online,[5] which allows us to perform fully controlled experiments with this model. Second, we consider the alignment-based projection approach of Buys and Botha (2016).[6] We discuss each of the two baselines in turn.

4.3.1 Higher-Order CRF Tagger

The MARMOT tagger is the leading non-neural approach to morphological tagging. This baseline is important since non-neural, feature-based approaches have been found empirically to be more efficient, in the sense that their learning curves tend to be steeper. Thus, in the low-resource setting we would be remiss to not consider a feature-based approach. Note that this is not a transfer approach, but rather only uses the low-resource data.

4.3.2 Alignment-based Projection

The projection approach of Buys and Botha (2016) provides an alternative method for transfer learning. The idea is to construct pseudo-annotations for bitext given cross-lingual alignments (Och and Ney, 2003). Then, one trains a standard tagger using the projected annotations. The specific tagger employed is the WSABIE model of Weston et al. (2011), which—like our approach— is a 0th-order discriminative neural model. In contrast to ours, however, their network is shallow. We compare the two methods in more detail in §6.

4.3.3 Architecture Study

Additionally, we perform a thorough study of the neural transfer learner, considering all three architectures. A primary goal of our experiments is to determine which of our three proposed neural transfer techniques is superior. Even though our experiments focus on morphological tagging, these architectures are more general in that they may be easily applied to other tasks, e.g., parsing or machine translation. We additionally explore the viability of multi-source transfer, i.e., the case where we have multiple source languages. All of our architectures generalize to the multi-source case without any complications.

4.4 Experimental Details

We train our models with the following conditions.

Evaluation Metrics. We evaluate using average per token accuracy, as is standard for both POS tagging and morphological tagging, and per feature F_1 as employed in Buys and Botha (2016). The per feature F_1 calculates a key F_1^k for each key in the target language's tags by asking if the key-attribute pair $k_i{=}v_i$ is in the predicted tag. Then, the key-specific F_1^k values are averaged equally. Note that F_1 is a more flexible metric as it gives partial credit for getting some of the attributes in the bundle correct, where accuracy does not.

Hyperparameters. Our networks are four layers deep (two LSTM layers for the character embedder, i.e., to compute $\boldsymbol{v}_i$ and two LSTM layers for the tagger, i.e., to compute $\boldsymbol{e}_i$) and we use an embedding size of 128 for the character input vector size and hidden layers of 256 nodes in all other cases. All networks are trained with the stochastic gradient method RMSProp (Tieleman and Hinton, 2012), with a fixed initial learning rate and a learning rate decay that is adjusted for the other languages according to the amount of training data. The batch size is always 16. Furthermore, we use dropout (Srivastava et al., 2014). The dropout probability is set to 0.2. We used Torch 7 (Collobert et al., 2011a) to configure the computation graphs implementing the network architectures.

5 Results and Discussion

We report our results in three tables. First, we report a detailed cross-lingual evaluation in Table 4. Secondly, we report a comparison against two baselines in Table 5 (accuracy) and Table 6 (F_1). We see two general trends of the data. First, we find that genetically closer languages yield better source languages. Second, we find that the multi-softmax architecture is the best in terms of transfer ability, as evinced by the results in Table 4. We find a wider gap between our model and the baselines under the accuracy than under F_1. We attribute this

[5] http://cistern.cis.lmu.de/marmot/

[6] We do not have access to the code as the model was developed in industry, so we compare to the numbers reported in the original paper, as well as additional numbers provided to us by the first author in a personal communication. The numbers will not be, strictly speaking, comparable. However, we hope they provide insight into the relative performance of the tagger.

		target language											
		$|\mathcal{D}_t| = 100$						$|\mathcal{D}_t| = 1000$					
		(ca)	(es)	(fr)	(it)	(pt)	(ro)	(ca)	(es)	(fr)	(it)	(pt)	(ro)
source language	(ca)	—	87.9%	84.2%	84.6%	81.1%	67.4%	—	94.1%	**93.5%**	93.1%	**89.0%**	89.8%
	(es)	88.9%	—	85.5%	85.6%	81.8%	69.5%	**95.5%**	—	**93.5%**	93.5%	88.9%	89.7%
	(fr)	88.3%	87.0%	—	83.6%	79.5%	**69.9%**	95.4%	93.8%	—	93.3%	88.6%	89.7%
	(it)	88.4%	87.8%	84.2%	—	80.6%	69.1%	95.4%	94.0%	93.3%	—	88.7%	**90.3%**
	(pt)	88.4%	88.9%	85.1%	84.7%	—	69.6%	95.3%	**94.2%**	**93.5%**	93.6%	—	89.8%
	(ro)	87.6%	87.2%	85.0%	84.4%	79.9%	—	95.3%	93.6%	93.4%	93.2%	88.5%	—
	multi-source	**89.8%**	**90.9%**	**86.6%**	**86.8%**	**83.4%**	67.5%	95.4%	**94.2%**	93.4%	**93.8%**	88.7%	88.9%

(a) Results for the Romance languages.

		target language											
		$|\mathcal{D}_t| = 100$						$|\mathcal{D}_t| = 1000$					
		(bg)	(cs)	(pl)	(ru)	(sk)	(uk)	(bg)	(cs)	(pl)	(ru)	(sk)	(uk)
source language	(bg)	—	47.4%	44.7%	67.3%	39.7%	57.3%	—	73.7%	75.0%	84.1%	70.9%	72.0%
	(cs)	57.8%	—	56.5%	62.6%	62.6%	54.0%	80.9%	—	**80.0%**	84.1%	78.1%	64.7%
	(pl)	54.3%	54.0%	—	59.3%	57.8%	48.0%	78.3%	74.9%	—	**84.2%**	75.9%	57.3%
	(ru)	**68.8%**	48.6%	47.4%	—	46.5%	**60.7%**	**83.1%**	73.6%	76.0%	—	71.4%	**72.7%**
	(sk)	55.2%	57.4%	54.8%	61.2%	—	49.3%	77.6%	**76.3%**	78.4%	83.9%	—	60.7%
	(uk)	44.1%	36.0%	34.4%	43.2%	30.0%	—	67.3%	64.8%	66.9%	76.1%	56.0%	—
	multi-source	64.5%	**57.9%**	**57.0%**	**64.4%**	**64.8%**	58.7%	81.6%	74.8%	78.1%	83.1%	**79.6%**	69.3%

(b) Results for the Slavic languages.

		target language					
		$|\mathcal{D}_t| = 100$			$|\mathcal{D}_t| = 1000$		
		(da)	(no)	(sv)	(da)	(no)	(sv)
source	(da)	—	77.6%	73.1%	—	90.1%	90.0%
	(no)	83.1%	—	75.7%	93.1%	—	90.5%
	(sv)	81.4%	76.5%	—	92.6%	90.2%	—
	multi-source	**87.8%**	**82.3%**	**77.2%**	**93.9%**	**91.2%**	**90.9%**

(c) Results for the Northern Germanic languages.

		target language					
		$|\mathcal{D}_t| = 100$			$|\mathcal{D}_t| = 1000$		
		(et)	(fi)	(hu)	(et)	(fi)	(hu)
source	(et)	—	**60.9%**	**60.4%**	—	**85.1%**	74.8%
	(fi)	**60.1%**	—	60.3%	**82.3%**	—	**75.2%**
	(hu)	47.1%	48.3%	—	76.9%	81.2%	—
	multi-source	54.7%	55.3%	55.4%	78.7%	81.8%	73.3%

(d) Results for the Uralic languages.

Table 4: Results for transfer learning with our joint model. The tables highlight that the best source languages are often genetically and typologically closest. Also, we see that multi-source often helps, albeit more often in the $|\mathcal{D}_t| = 100$ case.

to the fact that F_1 is a softer metric in that it assigns credit to partially correct guesses.

Source Language. As discussed in §2, the transfer of morphology is language-dependent. This intuition is borne out in the results from our study (see Table 4). We see that in the closer grouping of the Western Romance languages, i.e., Catalan, French, Italian, Portuguese, and Spanish, it is easier to transfer than with Romanian, an Eastern Romance language. Within the Western grouping, we see that the close pairs, e.g., Spanish and Portuguese, are amenable to transfer. We find a similar pattern in the other language families, e.g., Russian is the best source language for Ukrainian, Czech is the best language source for Slovak and Finnish is the best source language for Estonian.

Multi-Source Transfer. In many cases, we find that multiple sources noticeably improve the results over the single-source case. For instance, when we have multiple Romance languages as a source language, we see gains of up to 2%. We also see gains in the Northern Germanic languages when using multiple source languages. From a linguistic point of view, this is logical as different source languages may be similar to the target language along different dimensions, e.g., when transferring among the Slavic languages, we note that Russian retains the complex nominal case system of Serbian, but south Slavic Bulgarian is lexically more similar.

Performance Against the Two Baselines. As shown in Table 5 and Table 6, our model outperforms the projection tagger of Buys and Botha (2016) even though our approach does not utilize bitext, large-scale alignment or monolingual corpora—rather, all transfer between languages happens through the forced sharing of character-level features.[7] Our model, does, however, require

[7] We would like to highlight some issues of comparability with the results in Buys and Botha (2016). Strictly speaking, the results are not comparable and our improvement over their method should be taken with a grain of salt. As the source code is not publicly available and developed in industry, we resorted to numbers in their published work and additional numbers

	Accuracy												
	B&B (2016)			MarMoT		Ours (Mono)		Ours (Universal)		Ours (Joint)		Ours (Specific)	
	en (int)	best (non)	best (int)	100	1000	100	1000	100	1000	100	1000	100	1000
🇧🇬 (bg)	36.3	38.2	50.0	56.5	78.8	40.2	66.6	57.8	80.9	64.5	**81.6**	63.5	80.8
🇨🇿 (cs)	24.4	49.3	53.4	49.2	69.2	32.1	66.1	57.4	77.6	57.9	**74.8**	56.1	74.2
🇩🇰 (da)	36.6	46.9	46.9	75.9	90.9	45.3	86.6	77.6	90.1	87.8	93.9	89.2	**94.3**
🇪🇸 (es)	39.9	75.3	75.5	85.9	93.1	64.7	92.5	85.1	60.9	90.9	**94.2**	90.7	**94.2**
🇫🇮 (fi)	27.4	51.8	56.0	50.0	77.5	28.0	74.2	48.3	81.2	55.3	81.8	55.4	80.7
🇮🇹 (it)	38.1	75.5	75.9	81.7	92.3	67.0	88.9	84.7	93.1	86.8	**93.8**	86.1	93.3
🇵🇱 (pl)	25.3	47.4	51.3	51.7	71.1	32.1	60.9	47.4	**78.4**	57.0	78.1	56.1	76.4
🇵🇹 (pt)	36.6	71.9	72.2	77.0	86.3	61.7	85.6	80.6	88.7	83.4	88.7	82.4	**89.1**
🇸🇪 (sv)	29.3	44.5	44.5	69.5	88.3	46.1	84.2	75.7	90.0	77.2	**90.9**	78.3	90.7

Table 5: Comparison of our approach to various baselines for low-resource tagging under token-level accuracy. We compare on only those languages in Buys and Botha (2016). Note that tag-level accuracy was not reported in the original B&B paper, but was acquired through personal communication with the first author. All architectures presented in this work are used in their multi-source setting. The B&B and MarMoT models are single-source.

annotation of a small number of sentences in the target language for training. We note, however, that this does not necessitate a large number of human annotation hours (Garrette and Baldridge, 2013).

Reducing Sample Complexity. Another interesting a point about our model that is best evinced in Figure 3 is the feature-based CRF approach seems to be a better choice for the low-resource setting, i.e., the neural model has greater sample complexity. However, in the multi-task scenario, we find that the neural tagger's learning curve is even steeper. In other words, if we have to train a tagger on very little data, we are better off using a neural multi-task approach than a feature-based approach; preliminary attempts to develop a multi-task version of MarMoT failed (see Figure 3).

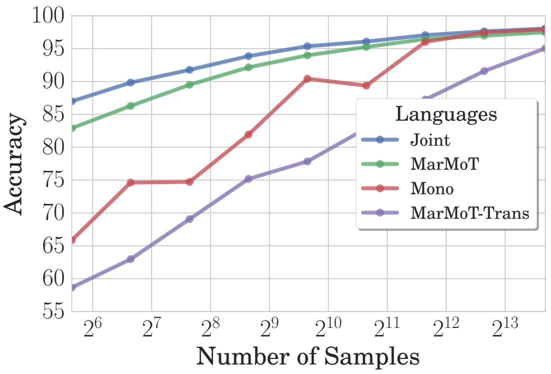


Figure 3: Learning Curve for Spanish and Catalan comparing our monolingual model, our joint model and two MarMoT models. The first MarMoT model is identical to those trained in the rest of the paper and the second attempts a multi-task approach, which failed so no further experimentation was performed with this model.

6 Related Work

We divide the discussion of related work topically into three parts for ease of intellectual digestion.

6.1 Alignment-Based Distant Supervision.

Most cross-lingual work in NLP—focusing on morphology or otherwise—has concentrated on indirect supervision, rather than transfer learning. The goal in such a regime is to provide noisy labels for training the tagger in the low-resource language through annotations projected over aligned bitext with a high-resource language. This method of projection was first introduced by Yarowsky and Ngai (2001) for the projection of POS annotation. While follow-up work (Fossum and Abney, 2005; Das and Petrov, 2011; Täckström et al., 2012) has continually demonstrated the efficacy of projecting simple part-of-speech annotations, Buys and Botha (2016) were the first to show the use of bitext-based projection for the training of a *morphological* tagger for low-resource languages.

As we also discuss the training of a morphological tagger, our work is most closely related to Buys and Botha (2016) in terms of the task itself. We contrast the approaches. The main difference lies therein, that our approach is not projection-based and, thus, does not require the construction of a bilingual lexicon for projection based on bitext.

obtained through direct communication with the authors. First, we used a slightly newer version of UD to incorporate more languages: we used v2 whereas they used v1.2. There are minor differences in the morphological tagset used between these versions. Also, in the $|\mathcal{D}_t| = 1000$ setting, we are training on significantly more data than the models in Buys and Botha (2016). A much fairer comparison is to our models with $|\mathcal{D}_t| = 100$. Also, we compare to their method using their standard (non) setup. This method is fair in so far as we evaluate in the same manner, but it disadvantages their approach, which cannot predict tags that are not in the source language.

	F_1													
	B&B (2016)			MarMoT		Ours (Mono)		Ours (Universal)		Ours (Joint)		Ours (Specific)	
	en (int)	best (non)	best (int)	100	1000	100	1000	100	1000	100	1000	100	1000
🇧🇬 (bg)	51.6	61.9	65.0	53.7	74.7	26.0	68.0	55.1	77.3	56.6	77.8	55.1	**78.6**
🇨🇿 (cs)	55.7	61.6	64.0	60.8	80.5	30.9	65.3	54.5	66.3	54.7	66.5	54.6	**67.0**
🇩🇰 (da)	65.4	70.7	73.1	69.7	92.9	35.3	90.1	85.9	93.2	86.9	**93.5**	83.2	93.2
🇪🇸 (es)	60.7	74.0	74.6	82.4	92.6	55.9	91.4	88.4	93.6	89.2	**94.1**	87.6	93.8
🇫🇮 (fi)	59.1	57.2	59.1	44.6	78.3	17.5	61.7	48.6	73.6	49.3	**74.4**	46.2	73.9
🇮🇹 (it)	66.1	74.4	75.3	78.7	90.0	56.4	87.0	83.1	90.5	83.3	**91.9**	82.7	91.7
🇵🇱 (pl)	47.3	56.8	60.4	57.8	81.8	31.6	69.7	61.9	83.9	62.5	**84.7**	62.6	83.2
🇵🇹 (pt)	60.2	69.2	73.1	67.6	80.0	42.9	82.0	77.9	86.3	78.1	**86.5**	71.8	85.7
🇸🇪 (sv)	55.1	72.1	74.6	69.7	90.2	44.1	86.4	82.5	93.2	83.5	**93.7**	82.8	93.4

Table 6: Comparison of our approach to various baselines for low-resource tagging under F_1 to allow for a more complete comparison to the model of Buys and Botha (2016). All architectures presented in this work are used in their multi-source setting. The B&B and MarMoT models are single-source. We only compare on those languages used in B&B.

Rather, our method jointly learns multiple taggers and forces them to share features—a true transfer learning scenario. In contrast to projection-based methods, our procedure always requires a minimal amount of annotated data in the low-resource target language—in practice, however, this distinction is non-critical as projection-based methods without a small mount of seed target language data perform poorly (Buys and Botha, 2016).

6.2 Character-level NLP.

Our work also follows a recent trend in NLP, whereby traditional word-level neural representations are being replaced by character-level representations for a myriad tasks, e.g., POS tagging dos Santos and Zadrozny (2014), parsing (Ballesteros et al., 2015), language modeling (Ling et al., 2015), sentiment analysis (Zhang et al., 2015) as well as the tagger of Heigold et al. (2017), whose work we build upon. Our work is also related to recent work on character-level morphological generation using neural architectures (Faruqui et al., 2016; Rastogi et al., 2016).

6.3 Neural Cross-lingual Transfer in NLP.

In terms of methodology, however, our proposal bears similarity to recent work in speech and machine translation–we discuss each in turn. In speech recognition, Heigold et al. (2013) train a cross-lingual neural acoustic model on five Romance languages. The architecture bears similarity to our multi-language softmax approach. Dependency parsing benefits from cross-lingual learning in a similar fashion (Guo et al., 2015, 2016).

In neural machine translation (Sutskever et al., 2014; Bahdanau et al., 2015), recent work (Firat et al., 2016; Zoph and Knight, 2016; Johnson et al., 2016) has explored the possibility of jointly train translation models for a wide variety of languages. Our work addresses a different task, but the undergirding philosophical motivation is similar, i.e., attack low-resource NLP through multi-task transfer learning. Kann et al. (2017) offer a similar method for cross-lingual transfer in morphological inflection generation.

7 Conclusion

We have presented three character-level recurrent neural network architectures for multi-task cross-lingual transfer of morphological taggers. We provided an empirical evaluation of the technique on 18 languages from four different language families, showing wide-spread applicability of the method. We found that the transfer of morphological taggers is an eminently viable endeavor among related language and, in general, the closer the languages, the easier the transfer of morphology becomes. Our technique outperforms two strong baselines proposed in previous work. Moreover, we define standard low-resource training splits in UD for future research in low-resource morphological tagging. Future work should focus on extending the neural morphological tagger to a joint lemmatizer (Müller et al., 2015) and evaluate its functionality in the low-resource setting.

Acknowledgements

RC acknowledges the support of an NDSEG fellowship. Also, we would like to thank Jan Buys and Jan Botha who help us compare to the numbers reported in their paper. We are grateful to Schloss Dagstuhl Leibniz Center for Informatics for hosting the event at which work on this paper started.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 349–359, Lisbon, Portugal. Association for Computational Linguistics.

Jan Buys and Jan A. Botha. 2016. Cross-lingual morphological tagging for low-resource languages. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1954–1964, Berlin, Germany. Association for Computational Linguistics.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. 2011a. Torch7: A Matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*.

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

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. The CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, Vancouver, Canada. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22, Berlin, Germany. Association for Computational Linguistics.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)*, pages 600–609, Portland, Oregon, USA. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 634–643, San Diego, California. Association for Computational Linguistics.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 866–875, San Diego, California. Association for Computational Linguistics.

Victoria Fossum and Steven P. Abney. 2005. Automatically inducing a part-of-speech tagger by projecting from multiple source languages across aligned corpora. In *Second International Joint Conference on Natural Language Processing (IJCNLP)*, pages 862–873.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 138–147, Atlanta, Georgia. Association for Computational Linguistics.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL)*, pages 1234–1244, Beijing, China. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pages 2734–2740.

Jan Hajič and Barbora Hladká. 1998. Tagging inflective languages: Prediction of morphological categories for a rich structured tagset. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*

(ACL-COLING), pages 483–490, Montreal, Quebec, Canada. Association for Computational Linguistics.

Georg Heigold, Guenter Neumann, and Josef van Genabith. 2017. An extensive empirical evaluation of character-based morphological tagging for 14 languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 505–513, Valencia, Spain. Association for Computational Linguistics.

Georg Heigold, Günter Neumann, and Josef van Genabith. 2016. Neural morphological tagging from characters for morphologically rich languages. *CoRR*, abs/1606.06640.

Georg Heigold, Vincent Vanhoucke, Andrew Senior, Patrick Nguyen, Marc'Aurelio Ranzato, Matthieu Devin, and Jeffrey Dean. 2013. Multilingual acoustic models using distributed deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8619–8623. IEEE.

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

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR*, abs/1611.04558.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. One-shot neural cross-lingual transfer for paradigm completion. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada. Association for Computational Linguistics.

Aleksandr E. Kibrik. 1998. Archi (Caucasian – Daghestanian). In *The Handbook of Morphology*, pages 455–476. Blackwell Oxford.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 282–289.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1520–1530, Lisbon, Portugal. Association for Computational Linguistics.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with Lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2268–2274, Lisbon, Portugal. Association for Computational Linguistics.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 322–332, Seattle, Washington, USA. Association for Computational Linguistics.

Frederick J. Newmeyer. 2007. Linguistic typology requires crosslinguistic formal categories. *Linguistic Typology*, 11(1):133–157.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*, pages 1659–1666.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kemal Oflazer and Ìlker Kuruöz. 1994. Tagging and morphological disambiguation of Turkish text. In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, pages 144–149. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan T. McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*, pages 2089–2096.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 623–633, San Diego, California. Association for Computational Linguistics.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31th International Conference on Machine Learning (ICML)*, pages 1818–1826.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *13th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 194–197.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015. A language-independent feature schema for inflectional morphology. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL)*, pages 674–680, Beijing, China. Association for Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 477–487, Montréal, Canada. Association for Computational Linguistics.

Tijmen Tieleman and Geoff Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. WSABIE: scaling up to large vocabulary image annotation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2764–2770.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Language Technologies 2001: The Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinkova, Jan Hajic jr., Jaroslava Hlavacova, Václava Kettnerová, Zdenka Uresova, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria dePaiva, Kira Droganova, Héctor Martínez Alonso, Çağr Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonca, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. Conll 2017 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NIPS)*, pages 649–657.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 30–34, San Diego, California. Association for Computational Linguistics.

Word-Context Character Embeddings for Chinese Word Segmentation

Hao Zhou[*]
Nanjing University
& Toutiao AI Lab
zhouhao.nlp@bytedance.com

Zhenting Yu[*]
Nanjing University
yuzt@nlp.nju.edu.cn

Yue Zhang
Singapore University
of Technology and Design
yue_zhang@sutd.edu.sg

Shujian Huang
Nanjing University
huangsj@nlp.nju.edu.cn

Xinyu Dai
Nanjing University
daixinyu@nju.edu.cn

Jiajun Chen
Nanjing University
chenjj@nlp.nju.edu.cn

Abstract

Neural parsers have benefited from automatically labeled data via dependency-context word embeddings. We investigate training character embeddings on a word-based context in a similar way, showing that the simple method significantly improves state-of-the-art neural word segmentation models, beating tri-training baselines for leveraging auto-segmented data.

1 Introduction

Neural network Chinese word segmentation (CWS) models (Zhang et al., 2016; Liu et al., 2016; Cai and Zhao, 2016) appeal for their strong ability of feature representation, employing unigram and bigram character embeddings as input features (Zheng et al., 2013; Pei et al., 2014; Ma and Hinrichs, 2015; Chen et al., 2015a). They give state-of-the-art performances. We investigate leveraging automatically segmented texts for enhancing their accuracies.

Such semi-supervised methods can be divided into two main categories. The first one is *bootstrapping*, which includes *self-training* and *tri-training*. The idea is to generate more training instances by automatically labeling large-scale data. Self-training (Yarowsky, 1995; McClosky et al., 2006; Huang et al., 2010; Liu and Zhang, 2012) labels additional data by using the base classifier itself, and tri-training (Zhou and Li, 2005; Li et al., 2014) uses two extra classifiers, taking the instances with the same labels for additional training data. A second semi-supervised learning method in NLP is *knowledge distillation*, which extracts knowledge from large-scale auto-labeled data as features.

Tri-training has been used in neural parsing, giving considerable improvements for both of dependency (Weiss et al., 2015) and constituent parsing (Vinyals et al., 2015; Choe and Charniak, 2016). Knowledge from auto-labeled data has also been used for parsing (Bansal et al., 2014; Melamud et al., 2016), where word embeddings are trained on automatic dependency tree context. Such knowledge has also been proved effective in conventional discrete CWS models, such as *label distribution information* (Wang et al., 2011; Zhang et al., 2013). However, it has not been investigated for neural CWS.

We propose *word-context character embeddings* (WCC), using segmentation label information in the pre-training of unigram and bigram character embeddings. The method packs the label distribution information into the embeddings, which could be regarded as a way for knowledge parameterization. Our idea follows Levy and Goldberg (2014), who use *dependency contexts* to train *word embeddings*. Additionally, motivated by co-training, we propose *multi-view word-context character embeddings* for cross-domain segmentation, which pre-trains two types of embedding for in-domain and out-of-domain data, respectively. In-domain embeddings are used for solving data sparseness, and out-of-domain embeddings are used for domain adaptation.

Our proposed model is simple, efficient and effective, giving average 1% accuracy improvement on in-domain data and 3.5% on out-of-domain data, respectively, significantly out-performing self-training and tri-training methods for leveraging auto-segmented data.

2 Baseline Segmentation Model

Chinese word segmentation can be regarded as a character sequence labeling task, where each char-

[*] Equal contributions

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 760–766
Copenhagen, Denmark, September 7–11, 2017.

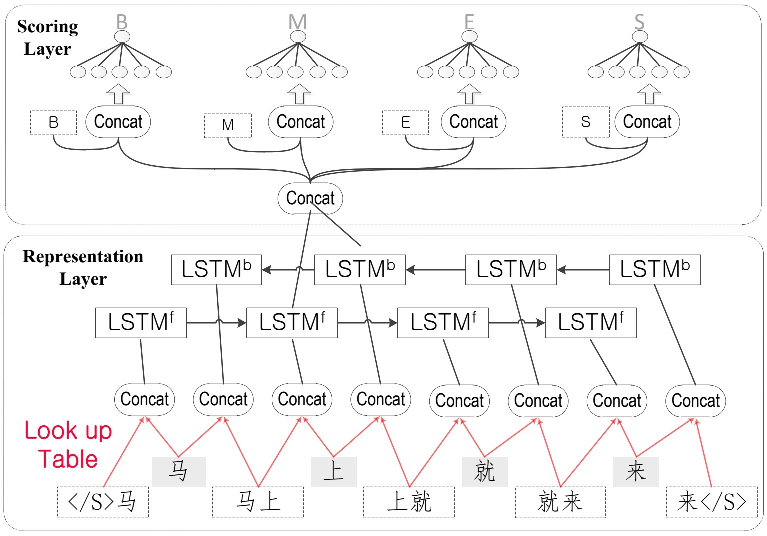


Figure 1: Baseline model architecture.

acter in the sentence is assigned a *segment label* from left to right, including {*B, M, E, S*}, to indicate the segmentation (Xue, 2003; Low et al., 2005; Zhao et al., 2006). *B, M, E* represent the character is the beginning, middle or end of a multi-character word, respectively. *S* represents that the current character is a single character word.

Following Chen et al. (2015b), a standard bi-LSTM model (Graves, 2008) is used to assign segmentation label for each character. As shown in Figure 1, our model consists of a *representation layer* and a *scoring layer*. The representation layer utilizes a bi-LSTM to capture the context of each character in the sentence. Given a sentence $\{w_1, w_2, w_3, \cdots, w_N\}$, where w_i is the i_{th} character in the sentence, and N is the sentence length, we have a corresponding embedding e_{w_i} and $e_{w_{i-1}w_i}$ for each character unigram w_i and character bigram $w_{i-1}w_i$, respectively. A forward word representation e_i^f is calculated as follows:

$$\begin{aligned} e_i^f &= concat_1(e_{w_i}, e_{w_{i-1}w_i}), \\ &= \tanh(W_1[e_{w_i}; e_{w_{i-1}w_i}]) \end{aligned}$$

A backward representation e_i^b can be obtained in the same way. Then e_i^f and e_i^b are fed into forward and backward LSTM units at current position, obtaining the corresponding forward and backward LSTM representations r_i^{lstm-f} and r_i^{lstm-b}, respectively.

In the *scoring layer*, we first obtain a linear combination of r_i^{lstm-f} and r_i^{lstm-b}, which is the final representation at the i_{th} position.

$$\begin{aligned} r_i &= concat_2(r_i^{lstm-f}, r_i^{lstm-b}) \\ &= \tanh(W_2[r_i^{lstm-f}; r_i^{lstm-b}]) \end{aligned}$$

Given the representation r_i, we use a scoring unit to score for each potential segment label. Given r_i, the score of segment label *M* is:

$$f_M^i = W_M h,$$

where

$$\begin{aligned} h &= concat_3(r_i, e_M), \\ &= \tanh(W_3[r_i; e_M]) \end{aligned}$$

W_M is the score matrix for label *M*, and e_M is the label embedding for label *M*.

3 Word-Context Character Embeddings

Our model structure is a derivation from the skip-gram model (Mikolov et al., 2013), similar to Levy and Goldberg (2014). Given a sentence with length n: $\{w_1, w_2, w_3, \cdots w_n\}$ and its corresponding segment labels: $\{l_1, l_2, l_3, \cdots l_n\}$, the pre-training context of current character w_t is the around characters in the windows with size c, together with their corresponding segment labels (Figure 2). Characters w_i and labels l_i in the context are represented by vectors $e^c_{w_i} \in \mathbb{R}^d$ and $e^c_{l_i} \in \mathbb{R}^d$, respectively, where d is the embedding dimensionality.

The word-context embedding of character w_t is represented as $e_{w_t} \in \mathbb{R}^d$, which is trained by predicting the surrounding context representations $e^c_{w'}$, and $e^c_{l_i}$, parameterizing the labeled segmentation information in the embedding parameters. To capture order information (Ling et al., 2015), we use different embedding matrices for context embedding in different context positions, training different embeddings for the same word when they reside on different locations as the context word. In particular, our context window size is five. As a result, each word has four different versions of e^c, namely e^c_{-1}, e^c_{-2}, e^c_{+1}, and e^c_{+2}, each taking a distinct embedding matrix. Given the context window $[w_{-2}, w_{-1}, w, w_{+1}, w_{+2}]$, w_{-1} is the left first context word of the focus word w, e^c_{-1,w_i} will be selected from embedding matrix E_{-1}, and w_{+1} is the right first word of w, e^c_{+1,w_i} will be selected from embedding matrix E_{+1}.

Note that each character has two types of embeddings, where e_{w_i} is the embedding form of w_i when w_i is the focus word, and $e^c_{w_i}$ is the embedding form of w_i when w_i is used as a surrounding context word. We do not have e_{l_i} because l_i only acts as the surrounding context. After pre-training, e_{w_i} will be used as the WCC embeddings.

The objective of our model is to maximize the average log probability of the context:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)+\log p(t_{t+j}|w_t)$$

Negative sampling (Mikolov et al., 2013) is used, where $\log p(w_{t+j}|w_t)$ and $\log p(t_{t+j}|w_t)$ are computed as:

$$\begin{aligned}p(w_{t+j}|w_t) &= \log\sigma({e^c_{w_{t+j}}}^{\top} e_{w_t}) \\ &\quad + \sum_{i=1}^{k}\mathbb{E}_{w_i\sim P_n(w)}[\log\sigma(-{e^c_{w_i}}^{\top} e_{w_t})]\end{aligned}$$

and

$$\begin{aligned}p(t_{t+j}|w_t) &= \log\sigma({e^c_{l_{t+j}}}^{\top} e_{w_t}) \\ &\quad + \sum_{i=1}^{k}\mathbb{E}_{l_i\sim P_n(l)}[\log\sigma(-{e^c_{l_i}}^{\top} e_{w_t})],\end{aligned}$$

respectively, where $P_n(w)$ and $P_n(l)$ is the noise distributions and k is the size of negative samples for each data sample.

Bigram embeddings are trained in the same way as unigram character embeddings. For out-of-domain segmentation, we pre-train two embeddings for each token, extracting knowledge from the two domains, respectively.

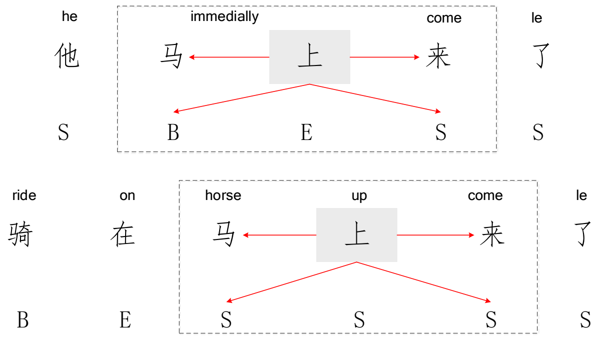


Figure 2: Word-context for the character ’ 上’ in two different sentences. The windows size $c = 3$.

4 Experiments

4.1 Set-up

We perform experiments on three standard datasets for Chinese word segmentation: PKU and MSR from the second SIGHAN bakeoff shared task, and Chinese Treebank 6.0 (CTB6). For PKU and MSR, 10% of the training data are randomly selected as development data. We follow Zhang et al. (2016) to split the CTB6 corpus into training, development and testing sections. For evaluating cross-domain performance, we also experiment on Chinese novel data. Following Zhang et al. (2014), the training set of CTB5 is selected for training, and the manually annotated sentences of free Internet novel ’Zhuxian’ (ZX) are selected as the development and test data (Liu and Zhang, 2012)[1].

Chinese Gigaword (LDC2011T13, 4M) is used for in-domain unlabeled data. For out-of-domain data, 20K raw sentences of Zhuxian is used. We take self-training and tri-training as baselines, which also use large-scale auto-segmented data. For self-training, skip-gram pre-training and word-context character embedding, unlabeled corpus is segmented automatically by our baseline model. For tri-training, we additionally use the ZPar (Zhang and Clark, 2007) and ICTCLAS[2] as our base classifiers .

We use F1 to evaluate segmentation accuracy. The recalls of in-vocabulary (IV) and out-of-vocabulary (OOV) are also measured.

4.2 Hyper-Parameters

The hyper-parameters used in this work are listed in Table 1. The values are selected according

[1] http://zhangmeishan.github.io/eacl14mszhang.zip

[2] http://ictclas.nlpir.org/

unigram dimension	50
bigram dimesion	50
label embedding dimention	32
LSTM hidden size	100
LSTM input size	100
learning rate	0.1
windows size	5

Table 1: Hyper-parameters.

System	CTB6	PKU	MSR	Speed
Greedy	94.9	95.0	97.2	14.7
CRF	95.0	95.1	97.2	3.6

Table 2: Comparisons between greedy and CRF segmentation. Speed: tokens per millisecond.

Systems	+4X	+8X	+16X	+32X
baseline	94.9			
self-training	95.0	94.9	94.9	94.8
tri-training	95.2	95.3	95.4	95.4

Table 3: Results of self-training and tri-training on CTB6 with varying scaled training data.

Type	System	CTB6	PKU	MSR
non-nn	Tseng et al. (2005)	-	95.0	96.4
	Sun et al. (2009)	-	95.2	97.3
	Wang et al. (2011)	95.8	-	-
	Zhang et al. (2013)	-	**96.1**	97.5
nn	Zheng et al. (2013)	-	92.4	93.3
	Pei et al. (2014)	-	95.2	97.2
	Kong et al. (2015)	-	90.6	90.7
	Ma and Hinrichs (2015)	-	95.1	96.6
	Chen et al. (2015c)†	-	94.8	95.6
	Xu and Sun (2016)	95.8	**96.1**	96.3
	Liu et al. (2016)	95.5	95.7	97.6
	Zhang et al. (2016)	95.4	95.1	97.0
	Cai and Zhao (2016)	-	95.5	96.5
comb	Zhang et al. (2016)	96.0	95.7	97.7
Ours	*baseline*	94.9	95.0	97.2
	+ *self-training*	95.0	94.8	97.0
	+ *tri-training*	95.5	95.5	97.4
	+ *skip-gram embeddings*	95.3	95.5	97.4
	+ *WCC embeddings*	**96.2**	**96.0**	**97.8**

Table 4: Comparison with other models.

to the development set of CTB6. Many previous character-based CWS models use a transition matrix to model the tag dependency and CRF for structured inference (Pei et al., 2014; Chen et al., 2015a). However, we find that, the greedy model obtains comparable segmentation accuracies across CTB6, PKU and MSR, yet giving much fast speed (Table 2). Hence we adopt the greedy model as our baseline segmentation model.

4.3 Utilizing Varying-Scale Data

The results of self-training and tri-training with varying-scale training data are list in Table 3, where +4X means adding 4 times the size of supervised training data into the training set. We find that self-training does not work well, and tri-training with 16X gives a 0.5% accuracy improvement. We adopt this setting for our baseline in the remaining experiments[3].

We also try to choose more effective examples for self-training and tri-training, by selecting training instances according to the base segmentation model score. However, the segmentation performances do not get improved. A possible reason is that the training instances with higher confidence are always shorter than the original sampled sentences, which may not be very helpful for semi-spervised segmentation.

4.4 In-Domain Results

As shown in Table 4, pre-training with conventional skip-gram embeddings gives only small improvements, which is consistent as findings of previous work (Chen et al., 2015a; Ma and Hinrichs, 2015; Cai and Zhao, 2016). Segmentation with self-training even shows accuracy drops on PKU and MSR. We speculate that the self-training by the neural CWS baseline is sensitive to the segmentation errors of the auto-labeled data. On average, our method obtains an absolute 1% accuracy improvement over the baseline, outperforming other semi-supervised method significantly[4].

We compare our model with other state-of-the-art segmentation models[5], which are grouped into 3 classes, namely traditional segmentation models (*non-nn*), neural segmentation models (*nn*), and the combination of both neural and traditional discrete features (*comb*). Our simple model gives top accuracies compared with related work. Liu et al. (2016), Cai and Zhao (2016) and Zhang et al. (2016) propose to incorporate word embedding features in the neural CWS, pre-training the word embeddings in the large-scale labeled data. Different to them, we employ a simpler character level model containing word information, yet obtaining higher F1 scores.

[3] For out-of-domain experiments, we include both the +16X and the 20K out-of-domain data for self-training and tri-training.

[4] The p-values are below 0.01 using pairwise t-test.

[5] Results with † are obtained from Cai and Zhao (2016), because results in the original paper use dictionary resources.

ours:	若在 (if)_ 鬼王 (guiwang)_ 手上 (hand) 夺下 (wrest)_ 七星剑 (qixin sword)，_ 我 (I) 必 (must)_ 器重 (think highly of)_ 于 (at) 你 (you)
baseline:	若在 (if)_ 鬼 (gui)_ 王 (king)_ 手上 (hand) 夺下 (wrest)_ 七 (seven)_ 星 (star)_ 剑 (sword)， 我 (I)_ 必 (must)_ 器 (ware)_ 重 (heavy) 于 (at)_ 你 (you)

Figure 3: Case studies.

System		F1	ROOV	RIV
Zhang et al. (2014b))	*baseline*	87.7	-	-
	+ *self-training*	88.7	-	-
Liu et al. (2014)	*baseline*	87.5	-	-
	+*Chinese Wikipedia*	**90.6**	-	-
Ours	*baseline*†	86.6	60.8	91.7
	+ *skip-gram*‡	87.6	-	-
	+ *self-training*‡	87.8	70.3	91.5
	+ *tri-training*‡	88.1	68.1	91.5
	+ *WCC embeddings*‡	89.1	70.4	93.7
	+ *multi-view WCC embeddings*$^{\sharp}$	**90.1**	**74.1**	**93.3**

Table 5: Results on the out-of-domain data. Models with † do not use large-scale data, models with ‡ use in-domain large-scale data, and models with $^{\sharp}$ use both in-domain, and out-of-domain large-scale data.

4.5 Out-of-Domain Results

We test out-of-domain performance of our model on the ZX dataset. We also use the multi-view word-context character embeddings (WCC) for cross domain segmentation, which uses two types of embeddings by simple vector concatenation. One type of embeddings is pre-trained on in-domain data, and the other type is pre-trained on out-of-domain data. In such case, the multi-view embeddings includes cross-domain information, which may enhance the cross-domain segmentation performance (Mou et al., 2016).

As shown in Table 5, using word-context character (*WCC*) embeddings and multi-view word-context character embeddings both give significantly higher accuracy improvements compared with other semi-supervised methods. Additionally, we find that multi-view WCC embeddings give an extra 1% F1 score improvement over WCC embeddings. Our proposed model also significantly improves the OOV recall (ROOV) and IV recall (RIV). By studying the cases of segmented output (Figure 3), we find that our model can recognize OOV words such as '鬼王', '七星剑' and the IV word '器重', which are incorrectly labeled by the baseline. This confirms that our proposed model is helpful for the data sparseness problem on closed domain and domain adaptation on across domain.

We also list the results of Zhang et al. (2014) and Liu et al. (2014) on this dataset. Liu et al. (2014) obtains better out-of-domain performance than our model. However, their results cannot be compared directly with ours because they use partial labeled URL link data from Chinese Wikipedia data for training.

5 Conclusion

We proposed word-context character embeddings for semi-supervised neural CWS, which makes the segmentation model more accurate on in-domain data, and more robust on the out-of-domain data. Our segmentation model is simple yet effective, achieving state-of-the-art segmentation accuracies on standard benchmarks. It can also be useful for other NLP tasks with small labeled training data, but a large unlabeled data. Our code could be downloaded at `https://github.com/zhouh/WCC-Segmentation`.

Acknowledge

We would like to thank the anonymous reviewers for their insightful comments. We also thank Ji Ma and Meishan Zhang for their helpful discussions. This work was partially founded by the Natural Science Foundation of China (61672277, 71503124) and the China National 973 project 2014CB340301.

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *ACL (2)*, pages 809–815.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–420. Association for Computational Linguistics.

Wenliang Chen, Min Zhang, and Yue Zhang. 2013. Semi-supervised feature transformation for dependency parsing. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1303–1313. Association for Computational Linguistics.

Wenliang Chen, Yue Zhang, and Min Zhang. 2014. Feature embedding for dependency parsing. In *COLING*, pages 816–826.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for chinese word segmentation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015b. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015c. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1197–1206. Association for Computational Linguistics.

Kook Do Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336. Association for Computational Linguistics.

Alex Graves. 2008. *Supervised Sequence Labelling with Recurrent Neural Networks*. Ph.D. thesis, Technical University Munich.

Zhongqiang Huang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 12–22. Association for Computational Linguistics.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2015. Segmental recurrent neural networks. *ICLR*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *ACL (2)*, pages 302–308. Citeseer.

Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Ambiguity-aware ensemble training for semi-supervised dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 457–467. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Yang Liu and Yue Zhang. 2012. Unsupervised domain adaptation for joint segmentation and pos-tagging. In *COLING (Posters)*, pages 745–754.

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, and Ting Liu. 2016. Exploring segment representations for neural segmentation models. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*.

Yijia Liu, Yue Zhang, Wanxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for crf-based chinese word segmentation using free annotations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 864–874. Association for Computational Linguistics.

Jin Kiat Low, Hwee Tou Ng, and Wenyuan Guo. 2005. A maximum entropy approach to chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 1612164, pages 448–455.

Jianqiang Ma and Erhard Hinrichs. 2015. Accurate linear-time chinese word segmentation via embedding matching. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1733–1743. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the main conference on human language technology conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159. Association for Computational Linguistics.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 479–489, Austin, Texas. Association for Computational Linguistics.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *ACL (1)*, pages 293–303.

Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2009. A discriminative latent variable chinese segmenter with hybrid word/character information. In *Proceedings of Human Language Technologies: The 2009 Annual*

Conference of the North American Chapter of the Association for Computational Linguistics, pages 56–64. Association for Computational Linguistics.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, volume 171.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In *NIPS*.

Yiou Wang, Yoshimasa Tsuruoka Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving chinese word segmentation and pos tagging with semi-supervised methods using large auto-analyzed data. In *IJCNLP*, pages 309–317.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. *arXiv preprint arXiv:1506.06158*.

Jingjing Xu and Xu Sun. 2016. Dependency-based gated recursive neural network for chinese word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–572. Association for Computational Linguistics.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013. Exploring representations from unlabeled data with co-training for chinese word segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 311–321. Association for Computational Linguistics.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Type-supervised domain adaptation for joint segmentation and pos-tagging. In *EACL*, pages 588–597.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-based neural word segmentation. In *Proceedings of the 54nd Annual Meeting of the Association for Computational Linguistics*.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 840–847, Prague, Czech Republic. Association for Computational Linguistics.

Hai Zhao, Chang-Ning Huang, and Mu Li. 2006. An improved chinese word segmentation system with conditional random field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, volume 1082117. Sydney: July.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *EMNLP*, pages 647–657.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering*, 17(11):1529–1541.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *51st Annual Meeting of the Association for Computational Linguistics*.

Segmentation-Free Word Embedding for Unsegmented Languages[*]

Takamasa Oshikiri
Graduate School of Engineering Science, Osaka University
RIKEN Center for Advanced Intelligence Project
CyberAgent, Inc.
`mail@oshikiri.org`

Abstract

In this paper, we propose a new pipeline of word embedding for unsegmented languages, called *segmentation-free word embedding*, which does not require word segmentation as a preprocessing step. Unlike space-delimited languages, unsegmented languages, such as Chinese and Japanese, require word segmentation as a preprocessing step. However, word segmentation, that often requires manually annotated resources, is difficult and expensive, and unavoidable errors in word segmentation affect downstream tasks. To avoid these problems in learning word vectors of unsegmented languages, we consider word co-occurrence statistics over all possible candidates of segmentations based on frequent character n-grams instead of segmented sentences provided by conventional word segmenters. Our experiments of noun category prediction tasks on raw Twitter, Weibo, and Wikipedia corpora show that the proposed method outperforms the conventional approaches that require word segmenters.

1 Introduction

Word embedding, which learns dense vector representation of words from large text corpora, has received much attention in the natural language processing (NLP) community in recent years. It is reported that the representation of words well captures semantic and syntactic properties of words (Bengio et al., 2003; Mikolov et al., 2013; Pennington et al., 2014), and is useful for many downstream NLP tasks, including part-of-speech tagging, syntactic parsing, and machine translation (Huang et al., 2011; Socher et al., 2013; Sutskever et al., 2014).

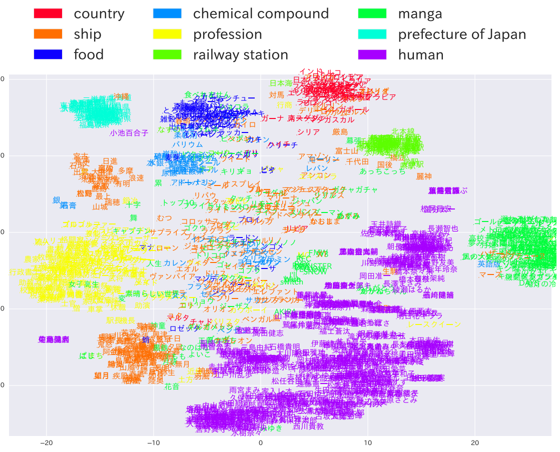


Figure 1: t-SNE projections of vector representation of Japanese nouns that generated by our proposed method without word dictionary. These proper nouns are color-coded according to its categories which extracted from Wikidata.

In order to train word embedding models on a raw text corpus, we have to do word segmentation as a preprocessing step. In space-delimited languages such as English and Spanish, simple rule-based and co-occurrence-based approaches offer reasonable segmentations. On the other hands, these approaches are impractical for unsegmented languages such as Chinese, Japanese, and Thai. Therefore, machine learning-based approaches are widely used in NLP for unsegmented languages. Conditional random field (CRF)-based supervised word segmentation (Kudo et al., 2004; Tseng et al., 2005) is still the most used one in Japanese and Chinese NLP (Prettenhofer and Stein, 2010; Funaki and Nakayama, 2015; Ishiwatari et al.,

[*] This work was done while the author was at Shimodaira laboratory, Division of Mathematical Science, Graduate School of Engineering Science, Osaka University, and Mathematical Statistics Team, RIKEN Center for Advanced Intelligence Project.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 767–772
Copenhagen, Denmark, September 7–11, 2017.

2015; Nakazawa et al., 2016).

However, there are some problems for these supervised word segmentation as a preprocessing step of a word embedding pipeline. First, they require language-specific manually annotated resources such as word dictionaries and segmented corpora. Since these manually annotated resources are typically unavailable for domain-specific corpora (e.g. Twitter or Weibo corpora that contain many neologisms and informal words), we have to create manually annotated resources if we need. Second, they cannot take advantage of word occurrence frequencies in a corpus. Even though a certain proper noun (e.g. "老人と海" (*The Old Man and the Sea*)) occurs frequently in a corpus, word segmenters will continue to split the proper noun erroneously (e.g. "老人/と/海" (a old man / and / a sea)) if it is not registered in the word dictionary. Because of segmentation errors incurred by these problems, the downstream word embedding model cannot learn vector representation of proper nouns, neologisms, and informal words.

In this paper, in order to learn word vectors from a raw text corpus while avoiding the above problems, we propose a new word segmentation-free pipeline for word embedding, referred to as *segmentation-free word embedding (sembei)*. Our framework first enumerates all possible segmentations (referred to as a *frequent n-gram lattice*) based on character n-grams that frequently occurred in the raw corpus, and then learns n-gram vectors from co-occurrence frequencies over the frequent n-gram lattice. Using the general idea of segmentation-free word embedding, we can extend existing word embedding models. Specifically, in this paper, we propose a segmentation-free version of the widely used skip-gram model with negative sampling (SGNS) (Mikolov et al., 2013), which we refer to as *SGNS-sembei*.

Although the frequent character n-grams necessarily include many non-words (i.e. n-grams that are not words), remarkably, our results show that nearest neighbor search works well for frequent words and even proper nouns (e.g. nearest neighbors of n-gram "ドイツ" (Germany) are "中国" (China), "イギリス" (United Kingdom), etc.). This observation suggests that we can use the proposed method for automatic acquisition of synonyms from large raw text corpora.

We conduct experiments on a noun category prediction task on several corpora and observe that our method outperforms the conventional approaches that use word segmenters. Fig. 1 shows a t-SNE projection of vector representation of Japanese nouns which is learned from only a raw Twitter corpus. We can see that the proposed method can learn vector representation of these nouns, and the learnt representation achieves good separation based on their categories.

2 Related Work

There are some representation models that do not rely on any segmenters. Dhingra et al. (2016) proposed a character-based RNN model for vector representation of tweets, and Schütze (2017) proposed a new text embedding method that learns n-gram vectors from the corpus that segmented randomly and then constructs text embeddings by summing up the n-gram vectors. In the field of representation learning for biological sequences (e.g. DNA and RNA), Asgari and Mofrad (2015) applied the skip-gram model (Mikolov et al., 2013) to fixed length fragments of biological sequences. These methods mainly aim at learning vector representation of texts or biological sequences instead of words or fragments of sequences. On the other hand, in this paper, we focus on learning vector representation of words from a raw corpus of unsegmented languages.

3 Conventional Approaches to Word Embeddings

Word embedding is also commonly used in NLP for unsegmented languages (Prettenhofer and Stein, 2010; Funaki and Nakayama, 2015; Ishiwatari et al., 2015). In these studies, they usually segment a raw corpus into words using a word segmenter or a morphological analyzer, and then feed the segmented corpus to word embedding models (e.g. the skip-gram model (Mikolov et al., 2013) or the GloVe (Pennington et al., 2014)) as in the case of space-delimited languages. The flowchart of the above process is shown in the left part of Fig. 2.

3.1 The original SGNS

The original skip-gram model with negative sampling (Mikolov et al., 2013) (we refer to it as *the original SGNS*) learns vector representation of words $\boldsymbol{v}_w$ and their contexts $\tilde{\boldsymbol{v}}_c$ that minimize the

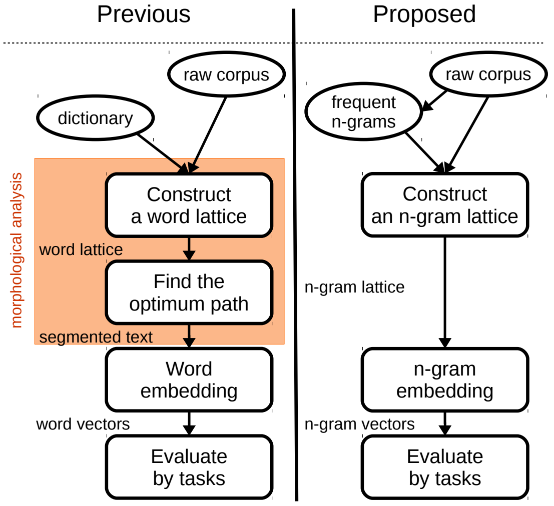


Figure 2: Flowcharts of previous and proposed pipelines. Morphological analyzers are in charge of the shaded part. Our main idea is to replace a word dictionary with a set of frequent character n-grams, and omit the indentification of the optimum path.

following objective function:

$$\underset{\{\boldsymbol{v}_w\}\cup\{\tilde{\boldsymbol{v}}_c\}}{\text{maximize}} \sum_{(w,c)\in\mathcal{D}} \log\sigma(\boldsymbol{v}_w^\top\tilde{\boldsymbol{v}}_c) + \sum_{(w,c)\in\mathcal{D}'} \log\sigma(-\boldsymbol{v}_w^\top\tilde{\boldsymbol{v}}_c) \tag{1}$$

where $\sigma(x) := (1+e^{-x})^{-1}$, $\mathcal{D}$ is a multiset (bag) of positive samples (i.e. co-occurred pairs in the corpus), and $\mathcal{D}'$ is a multiset of negative samples. This objective function is maximized using stochastic gradient descent (SGD).

4 Segmentation-Free Word Embeddings

In this section, we first introduce the general idea of *segmentation-free word embeddings* (*sembei*), and then propose a segmentation-free version of the SGNS.

While conventional word embedding approaches learn word vectors from segmented corpora that provided by word segmenters, our approach learns n-gram vectors from raw corpora, as in the right part of Fig. 2. In order to learn n-gram vectors from a raw corpus of unsegmented languages, we first construct a *frequent n-gram lattice*, which represents all possible segmentations based on frequent character n-grams of the corpus, in the same way as the construction of word lattices used in morphological analysis. Then, we learn n-gram vectors using co-occurrence statistics over the frequent n-gram lattice instead of segmented corpora as in conventional approaches.

4.1 Segmentation-Free Version of the SGNS

Here, we introduce a segmentation-free version of SGNS, referred to as *SGNS-sembei*, as an application of the idea of segmentation-free word embedding. Our method simply optimizes the original SGNS's objective function (1) with the slight modification: changing the definition of the multiset of positive samples $\mathcal{D}$.

In SGNS-sembei, $\mathcal{D}$ is redefined as the multiset of character n-gram pairs (w, c) where w and c occur adjacently in the corpus (i.e. w and c are connected in the frequent n-gram lattice). In addition, to discriminate co-occurrence with different order in the frequent n-gram lattice, we define contextual words with their relative positions to the center word as the same way as Ling et al. (2015) did.

We also redefine the multiset of negative samples $\mathcal{D}'$ using $\mathcal{D}$ in the same way as the original SGNS, and then optimize the objective function (1) using SGD.

Table 1: Examples of labels of entities (in Japanese, and in English for reference) and its categories extracted from Wikidata.

label (ja)	label (en)	category
ドイツ	Germany	country
二酸化炭素	carbon dioxide	chemical compound
消防士	firefighter	profession
アップルパイ	apple pie	food
長友佑都	Yuto Nagatomo	human

5 Experiment

In this section, we evaluate our method by the noun category prediction task on Twitter, Weibo, and Wikipedia corpora.

The C++ implementation of the proposed method is available on GitHub[1].

5.1 Settings

We used four raw text corpora: Wikipedia (Japanese), Wikipedia (Chinese), Twitter (Japanese), and Weibo (Chinese). The Wikipedia corpora consist of only a part of the Wikipedia

[1] https://github.com/oshikiri/w2v-sembei

Table 2: Micro F-scores (higher is better) and coverages [%] (in parentheses, higher is better).

	dictionary		Japanese		Chinese	
	default	Wikidata	Wikipedia	Twitter	Wikipedia	Weibo
SGNS	✓		0.896 (34)	0.761 (46)	0.889 (86)	0.766 (88)
SGNS	✓	✓	0.945 (98)	0.867 (96)	**0.891** (94)	0.765 (93)
SGNS-sembei			**0.949** (**100**)	**0.870** (**100**)	**0.891** (**100**)	**0.811** (**100**)

dumps[2] (dated on February 20th, 2017), whose HTML tags are removed. The Weiboscope corpus (Chau et al., 2013) consists of 226,841,122 posts mainly in Chinese, and we use only a part of it. The Twitter corpus consists of 17,316,968 Japanese tweets that were collected from October 26th, 2016 until November 22nd, 2016 via the Twitter Streaming API. We removed hashtags, users' id, and URL from Twitter and Weibo corpora. We extracted about 1,460k frequent n-grams[3] as the *frequent character n-grams* for our proposed method.

We extracted the noun-category pairs from the Wikidata (Vrandečić and Krötzsch, 2014) (We used the dump dated January 9th, 2017) as follows. We first extracted Wikidata entities whose headwords are also in the 1,460k frequent n-grams, and then extracted the Wikidata entities whose "`instance of`" properties are any of the predetermined category set[4], and then collected names and their categories of the entities. Examples of the extracted noun-category pairs are shown in Table 1.

We randomly split the noun-category pairs into a train (60%) and a test (40%) set. We trained linear C-SVM classifiers (Hastie et al., 2009) with the train set to predict categories from vector representation of the nouns. We performed a grid search over $(C, \text{classifier}) \in \{0.5, 1, 5, 10, 50, 100\} \times \{\text{one-vs-one}, \text{one-vs-rest}\}$ of linear SVM using the train set for each vector representation, and reported the best scores on the test set.

5.2 Baseline Systems

We compared SGNS-sembei with the conventional approaches that use the original SGNS and word segmenters. To segment the raw corpora, we used the MeCab (Kudo et al., 2004) for Japanese corpora and the Stanford Word Segmenter (Tseng et al., 2005) for Chinese corpora with their default dictionaries[5]. And we ignored the words that occur less than 5 times. We also ran these baseline systems in an ideal setting: running the word segmenters with the default dictionaries and additional dictionaries that consist of the nouns extracted in § 5.1.

We performed a grid search over $(h, t, n_{\text{neg}}) \in \{5, 8, 10\} \times \{10^{-5}, 10^{-4}, 10^{-3}\} \times \{3, 10, 25\}$ where h is the size of context window, t is the sampling threshold, and n_{neg} is the number of negative samples.

5.3 Results

In both the original SGNS and SGNS-sembei, we fixed the dimensionality of vector representation to 200 and the number of iterations to 5 in both baseline and our method. In SGNS-sembei, we used the number of negative samples $n_{\text{neg}} = 10$, size of context window $h = 1$, initial learning rate $\alpha_{\text{init}} = 0.01$.

The resulting micro F-scores and the coverages (i.e. the percentages of the noun-category pairs whose nouns' vector representation exists) are shown in Table 2, and the t-SNE (Maaten and Hinton, 2008) projections of Japanese nouns vectors learned from the Twitter corpus are shown in Fig. 1. We observed that our proposed method outperforms the conventional approaches that use word segmenters. Furthermore, the coverages of our method were higher than those of the SGNS with the default dictionary (especially in Japanese) and competitive to those of the SGNS with the default dictionary and Wikidata (which is an ideal

[2] We used {ja,zh}wiki-20170220-pages-articles1.xml in `https://dumps.wikimedia.org`

[3] In this experiment, we defined the frequent n-grams as the union of the top-k_n frequent n-grams, where n and k_n are the pre-specified numbers. And we used $n = 8$, $(k_1, \ldots, k_8) = (10000, 300000, 300000, 300000, 200000, 200000, 100000, 50000)$ for Japanese corpora, and $n = 7$, $(k_1, \ldots, k_7) = (10000, 400000, 400000, 300000, 200000, 100000, 50000)$ for Chinese corpora

[4] {country, profession, ship, railway station, food, chemical compound, prefecture of Japan, manga, human } for Japanese, and {country, profession, television series, business enterprise, city, chemical compound, taxon, human} for Chinese

[5] We use `mecab-ipadic v2.7.0` for the MeCab and `dict-chris6.ser.gz` for the Stanford Word Segmenter.

setting) even though our method does not require any manually annotated resources. We can also see that the learnt representation achieves good separation based on their categories as in Fig. 1. Nearest neighbor search using Twitter and Weibo corpora was also performed as preliminary experiments, and surprisingly, it worked well for frequent words as in Table. 3.

Table 3: Results of nearest neighbor search for frequent words

Language	Query	3-Nearest Neighbors
Japanese	ドイツ (Germany)	中国 (China), イギリス (UK), ポーランド (Poland)
	酸素 (oxygen)	水素 (hydrogen), 鉄分 (iron), 二酸化炭素 (carbon dioxide)
Chinese	德国 (Germany)	美国 (USA), 英国 (UK), 法国 (France)
	羽毛球 (badminton)	台球 (billiards), 网球 (tennis), 乒乓球 (pingpong)

6 Conclusion

We proposed segmentation-free word embedding for unsegmented languages. Although our method does not rely on any manually annotated resources, experimental results of the noun category prediction task on several corpora showed that our method outperforms conventional approaches that rely on manually annotated resources.

As an anonymous reviewer suggested, a possible direction of future work is to leverage another word segmentation approach which uses linguistic features, such as the Stanford Word Segmenter (Tseng et al., 2005) with k-best segmentations.

Acknowledgments

I would like to thank Hidetoshi Shimodaira, Thong Pham, Kazuki Fukui, and anonymous reviewers for their helpful suggestions. This work was partially supported by grants from Japan Society for the Promotion of Science KAKENHI (JP16H02789) to HS.

References

Ehsaneddin Asgari and Mohammad R. K. Mofrad. 2015. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLOS ONE*, 10(11):1–15.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Michael Chau, Chung hong Chan, and King wa Fu. 2013. Assessing censorship on microblogs in china: Discriminatory keyword analysis and the real-name registration policy. *IEEE Internet Computing*, 17:42–50.

Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William Cohen. 2016. Tweet2vec: Character-based distributed representations for social media. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 269–274, Berlin, Germany. Association for Computational Linguistics.

Ruka Funaki and Hideki Nakayama. 2015. Image-mediated learning for zero-shot cross-lingual document retrieval. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 585–590, Lisbon, Portugal. Association for Computational Linguistics.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*, 2 edition. Springer New York.

Fei Huang, Alexander Yates, Arun Ahuja, and Doug Downey. 2011. Language models as representations for weakly supervised nlp tasks. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 125–134, Portland, Oregon, USA. Association for Computational Linguistics.

Shonosuke Ishiwatari, Nobuhiro Kaji, Naoki Yoshinaga, Masashi Toyoda, and Masaru Kitsuregawa. 2015. Accurate cross-lingual projection between count-based word vectors by exploiting translatable context pairs. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 300–304, Beijing, China. Association for Computational Linguistics.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237, Barcelona, Spain. Association for Computational Linguistics. `http://taku910.github.io/mecab/`.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, Denver, Colorado. Association for Computational Linguistics.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Toshiaki Nakazawa, Chenchen Ding, Hideya Mino, Isao Goto, Graham Neubig, and Sadao Kurohashi. 2016. Overview of the 3rd workshop on asian translation. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, pages 1–46, Osaka, Japan. The COLING 2016 Organizing Committee.

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

Peter Prettenhofer and Benno Stein. 2010. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1118–1127, Uppsala, Sweden. Association for Computational Linguistics.

Hinrich Schütze. 2017. Nonsymbolic text representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 785–796, Valencia, Spain. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A Conditional Random Field Word Segmenter for SIGHAN bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 171. Jeju Island, Korea. `http://nlp.stanford.edu/software/segmenter.shtml`.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

From Textbooks to Knowledge: A Case Study in Harvesting Axiomatic Knowledge from Textbooks to Solve Geometry Problems

Mrinmaya Sachan **Avinava Dubey** **Eric P. Xing**
School of Computer Science
Carnegie Mellon University
{mrinmays, akdubey, epxing}@cs.cmu.edu

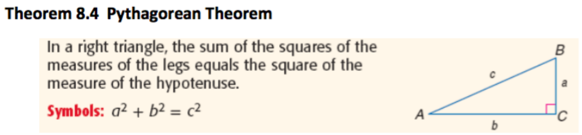
Abstract

Textbooks are rich sources of knowledge. Harvesting knowledge from textbooks is a key challenge in many educational applications. In this paper, we present an approach to obtain axiomatic knowledge of geometry in the form of horn-clause rules from math textbooks. The approach uses rich contextual and typographical features extracted from the textbooks. It also leverages the redundancy and shared ordering of axioms across multiple textbooks to accurately harvest axioms. These axioms are then parsed into horn-clause rules that are used to improve the state-of-the-art in solving geometry problems.

1 Introduction

Recently, researchers have proposed standardized tests as "drivers for progress in AI" (Clark and Etzioni, 2016). There is a growing body of work in solving standardized tests such as reading comprehensions (Richardson et al., 2013; Sachan et al., 2015, inter alia), science question answering (Schoenick et al., 2016; Sachan et al., 2016, inter alia), algebra word problems (Kushman et al., 2014, inter alia), geometry problems (Seo et al., 2015), pre-university entrance exams (Fujita et al., 2014), etc. A major challenge in building these solvers is the lack of subject knowledge. For example, geometry tests require knowledge of geometry axioms and pre-university exams require knowledge of laws of physics, chemistry, etc.

In this paper, we present an automatic approach that can (a) harvest such subject knowledge from textbooks, and (b) parse the extracted knowledge to structured programs that the solvers can use. Unlike information extraction systems trained on domains such as web documents (Chang et al., 2003; Etzioni et al., 2004, inter alia), learning an information extraction system that can extract axiomatic knowledge from textbooks is challenging because of the small amount of in-domain labeled data available for these tasks. We tackle this challenge by (a) leveraging the *redundancy* and *shared ordering* of axiom mentions across multiple textbooks[1], and (b) utilizing rich contextual and typographical features[2] from textbooks to effectively extract and parse axioms. Finally, we also provide an approach to parse the extracted axiom mentions from various textbooks and reconcile them to achieve the best program for each axiom.



Theorem 8.4 Pythagorean Theorem

In a right triangle, the sum of the squares of the measures of the legs equals the square of the measure of the hypotenuse.

Symbols: $a^2 + b^2 = c^2$

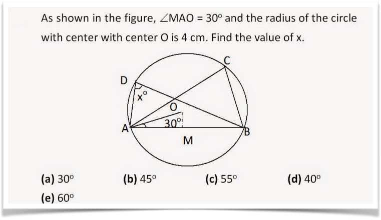
Figure 1: An excerpt of a textbook from our dataset that introduces the Pythagoras theorem. The textbook has a lot of typographical features that can be used to harvest this theorem: The textbook explicitly labels it as a "theorem"; there is a colored bounding box around it; an equation writes down the rule and there is a supporting figure. Our models leverages such rich contextual and typographical information (when available) to accurately harvest axioms and then parses them to horn-clause rules. The horn-clause rule derived by our approach for the Pythagoras theorem is: $isTriangle(ABC) \wedge perpendicular(AC, BC) \implies BC^2 + AC^2 = AB^2$.

As a case study, we use our approach to harvest axiomatic knowledge of geometry from math textbooks, and use this knowledge to improve the state-of-the-art system for solving SAT style geometry problems. Seo et al. (2015) recently presented *GEOS*, an automated end-to-end system that solves SAT style geometry questions such as the one shown in Figure 2. *GEOS* derives a logical expression that represents the meaning of the

[1] The same axiom can be potentially mentioned in a number of textbooks in different ways. All textbooks typically introduce axioms in roughly the same order – for example, pythagorous theorem would typically be introduced after introducing the notion of a right angled triangle.

[2] Textbooks contain rich context and typographical information (see Figure 1 for an illustrative example). We use this rich information as features in our model.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 773–784
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics



As shown in the figure, ∠MAO = 30° and the radius of the circle with center with center O is 4 cm. Find the value of x.

(a) 30° (b) 45° (c) 55° (d) 40°
(e) 60°

Text Description:	Diagram:
measure(∠MAO, 30°)	liesOn(A, circle O), liesOn(B, circle O),
isCircle(O)	liesOn(C, circle O), liesOn(D, circle O)
radius(O, 4 cm)	isLine(AB), isLine(BC), isLine(CA), isLine(BD), isLine(DA)
?x	isTriangle(ABC), isTriangle(ABD), isTriangle(AOM)
	measure(∠ADB, x), measure(∠MAO, 30°)
	measure(∠AMO, 90°)
	…

Figure 2: An example SAT style geometry problem with the question text, corresponding diagram and (optionally) answer candidates. Below: A logical expression that represents the meaning of the text description and the diagram in the problem. *GEOS* derives a weighted logical expression where each predicates also carries a weighted score but we do not show them here for clarity.

text description and the diagram (also shown in Figure 2), and then solves the geometry question by checking the satisfiablity of the derived logical expression. While this solver has its basis in coordinate geometry and indeed works, it has some key issues: *GEOS* requires an explicit mapping of each predicate into a set of constraints over point coordinates[3]. These constraints can be non-trivial to write, requiring significant manual engineering. As a result, *GEOS*'s constraint set is incomplete and it cannot solve a number of SAT style geometry questions. Furthermore, this solver is not interpretable. As our user studies show, it is not natural for a student to understand the solution of these geometry questions in terms of satisfiability of constraints over coordinates. A more natural way for students to understand and reason about these questions is through deductive reasoning using axioms of geometry[4].

We use our model to extract and parse axiomatic knowledge from a novel dataset of 20 publicly available math textbooks. We use this structured axiomatic knowledge to build a new axiomatic solver that performs logical inference to solve geometry problems. Our axiomatic solver outperforms *GEOS* on all existing test sets introduced in Seo et al. (2015) as well as a new test set of geometry questions collected from these textbooks. We also performed user studies on a number of school students studying geometry who found that our axiomatic solver is more *interpretable* and *useful* compared to *GEOS*.

2 Background: GEOS

Our work reuses *GEOS* to parse the question text and diagram into its formal problem description as shown in Figure 2. *GEOS* parses the question text and the diagram to a formal problem description. *GEOS* uses a logical formula, a first-order logic expression that includes known numbers or geometrical entities (e.g. 4 cm) as constants, unknown numbers or geometrical entities (e.g. O) as variables, geometric or arithmetic relations (e.g. *isLine*, *isTriangle*) as predicates and properties of geometrical entities (e.g. *measure*, *liesOn*) as functions.

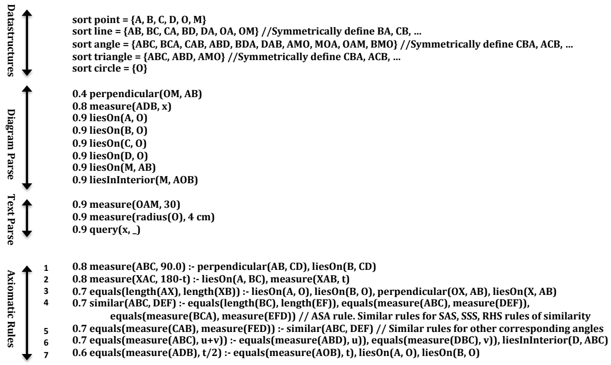
This is done by learning a set of relations that potentially correspond to the question text (or the diagram) along with a confidence score. For diagram parsing, *GEOS* uses a publicly available diagram parser for geometry problems (Seo et al., 2014). For text parsing, *GEOS* takes a multi-stage approach, which maps words or phrases in the text to their corresponding concepts, and then identifies relations between identified concepts. Given this formal problem description, *GEOS* use a numerical method to check the satisfiablity of literals by defining a relaxed indicator function for each literal. These indicator functions are manually engineered for every predicate. Since this is a cumbersome process, *GEOS* has an incomplete mapping of literals to indicator functions.

3 Set up for the Axiomatic Solver

In this work, we replace the numerical solver of *GEOS* with an axiomatic solver. We extract axiomatic knowledge from textbooks and parse them into horn clause rules. Then we build an axiomatic solver that performs logical inference with these horn clause rules and the formal problem description. A sample logical program (in prolog notation) that solves the problem in Figure 2 is given in Figure 3. The logical program has a set of declarations from the *GEOS* text and diagram parsers which describe the problem specification

[3] For example, the predicate *isPerpendicular*(AB, CD) is mapped to the constraint $\frac{y_B - y_A}{x_B - x_A} \times \frac{y_D - y_C}{x_D - x_C} = -1$.

[4] For example, the deductive reasoning required to solve the question in Figure 2 is: (1) Use the axiom that the sum of interior angles of a triangle is 180°and the fact that ∠AMO is 90°to conclude that ∠MOA is 60°. (2) △MOA ∼ △MOB (using a similar triangle axiom) and then, ∠MOB = ∠MOA = 60°(using the axiom that corresponding angles of similar triangles are equal). (3) Use angle sum rule to conclude that ∠AOB = ∠MOB + ∠MOA = 120°. (4) Use the axiom that the angle subtended by an arc of a circle at the centre is double the angle subtended by it at any point on the circle to conclude that ∠ADB = 0.5×∠AOB = 60°.


Datastructures
Diagram Parse
Text Parse
Axiomatic Rules

```
  sort point = {A, B, C, D, O, M}
  sort line = {AB, BC, CA, BD, DA, OA, OM} //Symmetrically define BA, CB, ...
  sort angle = {ABC, BCA, CAB, ABD, BDA, DAB, AMO, MOA, OAM, BMO} //Symmetrically define CBA, ACB, ...
  sort triangle = {ABC, ABD, AMO} //Symmetrically define CBA, ACB, ...
  sort circle = {O}

  0.4 perpendicular(OM, AB)
  0.8 measure(ADB, x)
  0.9 liesOn(A, O)
  0.9 liesOn(B, O)
  0.9 liesOn(C, O)
  0.9 liesOn(D, O)
  0.9 liesOn(M, AB)
  0.9 liesInInterior(M, AOB)

  0.9 measure(OAM, 30)
  0.9 measure(radius(O), 4 cm)
  0.9 query(x, _)

1 0.8 measure(ABC, 90.0) :- perpendicular(AB, CD), liesOn(B, CD)
2 0.8 measure(XAC, 180-t) :- liesOn(A, BC), measure(XAB, t)
3 0.7 equals(length(AX), length(XB)) :- liesOn(A, O), liesOn(B, O), perpendicular(OX, AB), liesOn(X, AB)
4 0.7 similar(ABC, DEF) :- equals(length(BC), length(EF)), equals(measure(ABC), measure(DEF)),
        equals(measure(BCA), measure(EFD)) // ASA rule. Similar rules for SAS, SSS, RHS rules of similarity
5 0.7 equals(measure(CAB), measure(FED)) :- similar(ABC, DEF) // Similar rules for other corresponding angles
6 0.7 equals(measure(ABC), u+v)) :- equals(measure(ABD), u)), equals(measure(DBC), v)), liesInInterior(D, ABC)
7 0.6 equals(measure(ADB), t/2) :- equals(measure(AOB), t), liesOn(A, O), liesOn(B, O)
```

Figure 3: A sample logical program (in prolog style) that solves the problem in Figure 2. The program consists of a set of data structure declarations that correspond to types in the prolog program, a set of declarations from the diagram and text parse and a subset of the geometry axioms written as horn clause rules. The axioms are used as the underlying theory with the aforementioned declarations to yield the solution upon logical inference. Normalized confidence weights from the diagram, text and axiom parses are used as probabilities. For readers understanding, we list the axioms in the order (1 to 7) they are used to solve the problem. However, this ordering is not required. Other (less probable) declarations and axiom rules are not shown here for clarity but they can be assumed to be present.

and the parsed horn clause rules describe the underlying theory. Normalized confidence scores from question text, diagram and axiom parsing models are used as probabilities in the program. Next, we describe how we harvest structured axiomatic knowledge from textbooks.

4 Harvesting Axiomatic Knowledge

We present a structured prediction model that identifies axioms in textbooks and then parses them. Since harvesting axioms from a single textbook is a very hard problem, we use multiple textbooks and leverage the redundancy of information to accurately extract and parse axioms. We first define a joint model that identifies axiom mentions in each textbook and aligns repeated mentions of the same axiom across textbooks. Then, given a set of axioms (with possibly, multiple mentions of each axiom), we define a parsing model that maps each axiom to a horn clause rule by utilizing the various mentions of the axiom.

Given a set of textbooks $\mathcal{B}$ in machine readable form (XML in our experiments), we extract chapters relevant for geometry in each of them to obtain a sequence of sentences (with associated typographical information) from each textbook. Let $\mathbf{S}_b = \{s_0^{(b)}, s_1^{(b)}, \ldots s_{|\mathbf{S}_b|}^{(b)}\}$ denote the sequence of sentences in textbook b. $|\mathbf{S}_b|$ denotes the number of sentences in textbook b.

4.1 Axiom Identification and Alignment

We decompose the problem of extracting axioms from textbooks into two tractable sub-problems: (a) identification of axiom mentions in each textbook using a sequence labeling approach, and (b) aligning repeated mentions of the same axiom across textbooks. Then, we combine the learned models for these sub-problems into a joint optimization framework that simultaneously learns to identify and align axiom mentions. Joint modeling of the axiom identification and alignment is necessary as both sub-problems can help each other.

4.1.1 Axiom Identification

Linear-chain CRF formulation (Lafferty et al., 2001) can be used for the subproblem of axiom identification. Given $\{\mathbf{S}_b | b \in \mathcal{B}\}$, the model labels each sentence $s_i^{(b)}$ as **B**efore, **I**nside or **O**utside an axiom. Hereon, a contiguous block of sentences labeled $\mathbf{B}$ or $\mathbf{I}$ will be considered as an axiom mention. Let $\mathcal{T} = \{\mathbf{B}, \mathbf{I}, \mathbf{O}\}$ denote the tag set. Let $y_i^{(b)}$ be the tag assigned to $s_i^{(b)}$ and $\mathbf{Y}_b$ be the tag sequence assigned to $\mathbf{S}_b$. The CRF defines:

$$p(\mathbf{Y}_b|\mathbf{S}_b;\boldsymbol{\theta}) \propto \prod_{k=1}^{|\mathbf{S}_b|} \exp\left(\sum_{i,j\in\mathcal{T}} \boldsymbol{\theta}_{ij}^T \mathbf{f}_{ij}(y_{k-1}^{(b)}, y_k^{(b)}, \mathcal{S}_b)\right)$$

We find the parameters $\boldsymbol{\theta}$ using maximum-likelihood estimation with L2 regularization:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_{b\in\mathcal{B}} \log p(\mathbf{Y}_b|\mathbf{S}_b;\boldsymbol{\theta}) - \lambda||\boldsymbol{\theta}||_2^2$$

We use L-BFGS to optimize the objective and Viterbi decoding for inference.

Features: Features f look at a pair of adjacent tags $y_{k-1}^{(b)}$, $y_k^{(b)}$, the input sequence $\mathbf{S}_b$, and where we are in the sequence. The features (listed in Table 1) include various content based features encoding various notions of similarity between pairs of sentences as well as various typographical features such as whether the sentences are annotated as an axiom (or theorem or corollary) in the textbook, contain equations, diagrams, text that is bold or italicized, are in the same node of the xml hierarchy, are contained in a bounding box, etc.

Some extracted axiom mentions contain pointers to a diagram eg. “Figure 2.1”. We consider the diagram to be a part of the axiom mention.

4.1.2 Axiom Alignment

Next, we leverage the redundancy of information and the relatively fixed ordering of axioms in various textbooks by aligning various mentions of the same axiom across textbooks and introducing structural constraints on the alignment.

Content	Sentence Overlap	Semantic Textual Similarity between the current and next sentence. We include features that compute the proportion of common unigrams and geometry entities (constants, predicates and functions) across the two sentences. This feature is conjoined with the tag assigned to the current and next sentence.
	Geometry entities	No. of geometry entities (normalized by the number of tokens) in this sentence. This feature is conjoined with the tag assigned to the current sentence.
	Intra-sentence semantics	Indicator that the current sentence contains any one of the following words: *hence*, *if*, *equal*, *twice*, *proportion*, *ratio*, *product*. This feature is conjoined with the tag assigned to the current sentence.
Typography	Axiom, Theorem, Corollary Mention	(a) The current (or previous) sentence is mentioned as an Axiom, Theorem or Corollary e.g. *Similar Triangle Theorem* or *Corollary 2.1*. (b) The section or subsection in the textbook containing the current (or previous) sentence mentions an Axiom, Theorem or Corollary. This feature is conjoined with the tag assigned to the current (and previous) sentence.
	Eqn. Template	The current (or next) sentence contains an equation eg. $PA \times PB = PT^2$. This feature is conjoined with the tag assigned to the current (and next) sentence.
	Assoc. Diagram	The current sentence contains a pointer to a figure eg. “Figure 2.1”. This feature is conjoined with the tag assigned to the current sentence.
	RST edge	Indicator for the RST relation between the current and next sentence. This feature is conjoined with the tag assigned to the current and next sentence.
	Bold/Underline	The sentence (or previous) sentence contains text that is in bold font or underlined. Conjoined with the tag assigned to the current (and previous) sentence.
	XML structure	Indicator that the current and previous sentence are in the same node of the XML hierarchy. Conjoined with the tag assigned to the current and previous sentence.
	Bounding box	Indicator that the current and previous sentence are bounded by a bounding box in the textbook. Conjoined with the tag assigned to the current and previous sentence.

Table 1: Feature set for our axiom identification model. The features are based on content and typography.

Let $\mathbf{A}_b = \left(A_1^{(b)}, A_2^{(b)}, \ldots, A_{|\mathbf{A}_b|}^{(b)}\right)$ be the axiom mentions extracted from textbook b. Let $\mathbf{A}$ denote the collection of axiom mentions extracted from all textbooks. We assume a global ordering of axioms $\mathbf{A}^* = (A_1^*, A_2^*, \ldots, A_U^*)$ where U is some pre-defined upper bound on the total number of axioms in geometry. Then, we emphasize that the axiom mentions extracted from each textbooks (roughly) follow this ordering. Let $Z_{ij}^{(b)}$ be a random variable that denotes if axiom $A_i^{(b)}$ extracted from book b refers to the global axiom A_j^*. We introduce a log-linear model that factorizes over alignment pairs:

$$P(\mathbf{Z}|\mathbf{A};\boldsymbol{\phi}) = \frac{1}{Z(\mathbf{A};\boldsymbol{\phi})} \times$$
$$\exp\left(\sum_{\substack{b_1,b_2\in\mathcal{B}\\ b_1\neq b_2}} \sum_{1\leq k\leq U} \sum_{\substack{1\leq i\leq|\mathbf{A}_{b_1}|\\ 1\leq j\leq|\mathbf{A}_{b_2}|}} Z_{ik}^{(b_1)} Z_{jk}^{(b_2)} \boldsymbol{\phi}^T \mathbf{g}(A_i^{(b_1)}, A_j^{(b_2)})\right)$$

Here, $Z(\mathbf{A};\boldsymbol{\phi})$ is the partition function of the log-linear model. $\mathbf{g}$ denotes the feature function described later. We introduce the following constraints on the alignment structure:

C1: An axiom appears in one book at-most once

C2: An axiom refers to exactly one theorem in the global ordering

C3: Ordering Constraint: If i^{th} axiom in a book refers to the j^{th} axiom in the global ordering then no axiom succeeding the i^{th} axiom can refer to a global axiom preceding j.

Learning with Hard Constraints: We find the optimal parameters $\boldsymbol{\phi}$ using maximum-likelihood estimation with L2 regularization:

$$\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi}} \log P(\mathbf{Z}|\mathbf{A};\boldsymbol{\phi}) - \mu||\boldsymbol{\phi}||_2^2$$

We use L-BFGS to optimize the objective. To compute feature expectations appearing in the gradient of the objective, we use a Gibbs sampler. The sampling equations for Z_{ik}^b are:

$$P(Z_{ik}^{(b)}|rest) \propto \exp\left(T_b(i,k)\right) \quad (1)$$
$$T_b(i,k) = Z_{ik}^{(b)} \sum_{\substack{b'\in\mathcal{B}\\ b'\neq b}} \sum_{1\leq j\leq|A_{b'}|} Z_{jk}^{(b')} \boldsymbol{\phi}^T \mathbf{g}(A_i^{(b)}, A_j^{(b')})$$

Note that the constraints $C1\ldots3$ define the feasible space of alignments. Our sampler always samples the next $Z_{ik}^{(b)}$ in this feasible space.

Learning with Soft Constraints: We might want to treat some constraints, in particular, the ordering constraints $C3$ as soft constraints. We can write down the constraint $C3$ using the alignment variables:

$$Z_{ij}^{(b)} \leq 1 - Z_{kl}^{(b)}$$
$$\forall\ 1 \leq i < k \leq |\mathbf{A}_b|, 1 \leq l < j \leq U$$
$$\forall\ b \in \mathcal{B}$$

To model these constraints as soft constraints, we penalize the model for violating these constraints. Let the penalty for violating the above constraint be $\exp\left(\nu \max\left(0, 1 - Z_{ij}^{(b)} - Z_{kl}^{(b)}\right)\right)$. We introduce a new regularization term: $\mathbf{R}(\mathbf{Z}) = \sum_{\substack{1\leq i<k\leq|\mathbf{A}_b|\\ 1\leq l<j\leq U\\ b\in\mathcal{B}}} \exp\left(\nu \max\left(0, 1 - Z_{ij}^{(b)} - Z_{kl}^{(b)}\right)\right)$. Here ν is a hyper-parameter to tune the cost of violating a constraint. We write down the following regularized objective:

$$\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi}} \log P(\mathbf{Z}|\mathbf{A};\boldsymbol{\phi}) - \mathbf{R}(\mathbf{Z}) - \mu||\boldsymbol{\phi}||_2^2$$

We use L-BFGS to find the optimal parameters $\boldsymbol{\phi}*$. We perform Gibbs sampling to compute feature expectations. The sampling equation for $Z_{ik}^{(b)}$ is similar (eq 1), but:

$$T_b(i,k) = \sum_{\substack{b'\in\mathcal{B}\\ b'\neq b}} \sum_{1\leq j\leq|A_{b'}|} Z_{ik}^{(b)} Z_{jk}^{(b')} \boldsymbol{\phi}^T \mathbf{g}(A_i^{(b)}, A_j^{(b')})$$

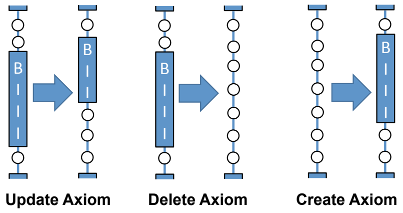


Figure 4: An illustration of the three operations to sample axiom blocks.

$$+\nu \sum_{\substack{b'\in\mathcal{B}\\ b'\neq b}} \sum_{i<j\leq|A_{b'}|} \sum_{1\leq l<k} \left(1 - Z_{ik}^{(b)} - Z_{jl}^{(b')}\right)$$
$$+\nu \sum_{\substack{b'\in\mathcal{B}\\ b'\neq b}} \sum_{1\leq j<i|} \sum_{k<l\leq U} \left(1 - Z_{ik}^{(b)} - Z_{jl}^{(b')}\right)$$

Features: Now, we describe the features g. These too include content based features encoding various notions of similarity between pairs of axiom mentions as well as various typographical features. The features are listed in Table 2.

4.1.3 Joint Identification and Alignment

Joint modeling of axiom identification and alignment components is useful as both problems potentially help each other. Let $Y_{ij}^{(b)}$ denote that the sentence $s_i^{(b)}$ from book b has tag j. We reuse the definitions of the alignment variables $Z_{ij}^{(b)}$ as before. We further define $Z_{i0}^{(b)}$ such that it denotes that the i^{th} axiom in textbook b is not aligned to any global axiom. We again define a log-linear model with factors that score axiom identification and axiom alignments.

$$p(\mathbf{Y},\mathbf{Z}|\{\mathbf{S}_b\};\boldsymbol{\theta},\boldsymbol{\phi}) \propto f_{AI}(\mathbf{Y}|\{\mathbf{S}_b\};\boldsymbol{\theta}) \times f_{AA}(\mathbf{Z}|\mathbf{Y},\{\mathbf{S}_b\};\boldsymbol{\phi})$$

Here, the factors:

$$f_{AI} = \exp(\sum_{b\in\mathcal{B}} \sum_{k=1}^{|\mathbf{S}_b|} \sum_{i,j\in\mathcal{T}} Y_{k-1i}^{(b)} Y_{kj}^{(b)} \boldsymbol{\theta}_{ij}^T \mathbf{f}_{ij}(i,j,\mathcal{S}_b))$$
$$f_{AA} = \exp(\sum_{\substack{b_1,b_2\in\mathcal{B}\\ b_1\neq b_2}} \sum_{1\leq k\leq U} \sum_{\substack{1\leq i\leq|\mathbf{A}_{b_1}|\\ 1\leq j\leq|\mathbf{A}_{b_2}|}} Z_{ik}^{(b_1)} Z_{jk}^{(b_2)} \boldsymbol{\phi}^T \mathbf{g}(A_i^{(b_1)}, A_j^{(b_2)}))$$

We write down the model constraints below:
C1': Every sentence has a unique label
C2' Tag O cannot be followed by tag I
C3' Consistency between Y's and Z's i.e. axiom boundaries defined by Y's and Z's must agree.
C4' = C3.

We use L-BFGS for learning. To compute feature expectations, we use a Metropolis Hastings sampler that samples $\mathbf{Y}'s$ and $\mathbf{Z}'s$ alternatively. Sampling for $\mathbf{Z}'s$ reduces to Gibbs sampling and the sampling equations are as same as before (Section 4.1.2). For better mixing, we sample $\mathbf{Y}$ in blocks. Consider blocks of $\mathbf{Y}$'s which denote axiom boundaries at time stamp t , we define three operations to sample axiom blocks at the next time stamp. The operations (shown in Figure 4) are:

Update axiom: The axiom boundary can be shrunk, expanded or moved. The new axiom, however, cannot overlap with other axioms.
Delete axiom: The axiom can be deleted by labeling all its sentences as O.
Introduce axiom: Given a contiguous sequence of sentences labeled O, a new axiom can be introduced.

Note that these three operations define an ergodic Markov chain. We use the axiom identification part of the model as the proposal:

$$Q(\bar{\mathbf{Y}}|\mathbf{Y}) \propto \exp\left(\sum_{b\in\mathcal{B}} \sum_{k=1}^{|\mathbf{S}_b|} \sum_{i,j\in\mathcal{T}} \bar{Y}_{k-1i}^{(b)} \bar{Y}_{kj}^{(b)} \boldsymbol{\theta}_{ij}^T \mathbf{f}_{ij}(i,j,\mathcal{S}_b)\right)$$

Hence, the acceptance ratio only depends on the alignment part of the model: $R(\bar{\mathbf{Y}}|\mathbf{Y}) = \min\left(1, \frac{U(\bar{\mathbf{Y}})}{U(\mathbf{Y})}\right)$ where $U(\mathbf{Y}) = f_{AA}$. We again have two variants, where we model the ordering constraints (C4$'$) as soft or hard constraints.

4.2 Axiom Parsing

After harvesting axioms, we build a parser for these axioms that maps raw axioms to horn clause rules. The axiom harvesting step provides us a multi-set of axiom extractions. Let $\mathcal{A} = \{\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_{|\mathcal{A}|}\}$ represent the multi-set where each axiom $\mathbf{A}_i$ is mentioned at least once.

First, we describe a base parser that parses axiom mentions to horn clause rules. Then, we utilize the redundancy of axiom extractions from various sources (textbooks) to improve our parser.

4.2.1 Base Axiomatic Parser

Our base parser identifies the *premise* and *conclusion* portions of each axiom and then uses *GEOS*'s text parser to parse the two portions into a logical formula. Then, the two logical formulas are put together to form horn clause rules.

Axiom mentions (for example, the Pythagoras theorem mention in Figure 1) are often accompanied by equations or diagrams. When the mention has an equation, we simply treat the equation as the *conclusion* and the rest of the mention as the *premise*. When the axiom has an associated diagram, we always include the diagram in the *premise*. We learn a model to predict the split of the axiom text into two parts forming the *premise* and the *conclusion* spans. Then, the *GEOS* parser maps the *premise* and *conclusion* spans to *premise* and *conclusion* logical formulas, respectively.

Let Z_s represent the split that demarcates the *premise* and *conclusion* spans. We score the ax-

Unigram, Bigram, Dependency and Entity Overlap	Real valued features that compute the proportion of common unigrams, bigrams, dependencies and geometry entities (constants, predicates and functions) across the two axioms. When comparing geometric entities, we include geometric entities derived from the associated diagrams when available.
Longest Common Subsequence	Real valued feature that computes the length of longest common sub-sequence of words between two axiom mentions normalized by the total number of words in the two mentions.
Number of sentences	Real valued feature that computes the absolute difference in the number of sentences in the two mentions.
Alignment Scores	We use an off-the-shelf monolingual word aligner – *JACANA* (Yao et al., 2013) pretrained on PPDB – and compute alignment score between axiom mentions as the feature.
MT Metrics	We use two common MT evaluation metrics *METEOR* (Denkowski and Lavie, 2010) and *MAXSIM* (Chan and Ng, 2008), and use the evaluation scores as features. While *METEOR* computes n-gram overlaps controlling on precision and recall, *MAXSIM* performs bipartite graph matching and maps each word in one axiom to at most one word in the other.
Summarization Metrics	We also use *Rouge-S* (Lin, 2004), a text summarization metric, and use the evaluation score as a feature. *Rouge-S* is based on skip-grams.
Equation Template	Indicator feature that matches templates of equations detected in the axiom mentions.
Image Caption	Proportion of common unigrams in the image captions of the diagrams associated with the axiom mentions. If both mentions do not have associated diagrams, this feature doesn't fire.
XML structure	Indicator matching the current (and parent) node of axiom mentions in respective XML hierarchies.

Table 2: Feature set for our axiom alignment model. The features are based on content, structure and typography.

iom split as a log-linear model: $p(Z_s|a;\mathbf{w}) \propto \exp\left(\mathbf{w}^T\mathbf{h}(a, Z_s)\right)$. Here, $\mathbf{h}$ are feature functions described later. We found that in most cases (>95%), the premise and conclusion are contiguous spans in the axiom mention where the left span corresponds to the *premise* and the right span corresponds to the *conclusion*. Hence, we search over the space of contiguous spans to infer Z_s. We use L-BGFGS for learning.

Features: We list the features $\mathbf{h}$ in Table 3. The features are defined over candidate spans forming the text split, are strongly inspired from rhetorical structure theory (Mann and Thompson, 1988) and previous works on discourse parsing (Marcu, 2000; Soricut and Marcu, 2003). Given a beam of *Premise* and *Conclusion* splits, we use the *GEOS* parser to get *Premise* and *Conclusion* logical formulas for each split in the beam and obtain a beam of axiom parses for each axiom in each textbook.

4.2.2 Multi-source Axiomatic Parser

Now, we describe a multi-source parser that utilizes the redundancy of axiom extractions from various sources (textbooks). Given a beam of 10-best parses for each axiom from each source, we use a number of heuristics to determine the best parse for the axiom:

1. Majority Voting: For each axiom, pick the parse that occurs most frequently across beams.

2. Average Score: Pick the parse that has the highest average parse score (only counting top 5 parses for each source), for each axiom.

3. Learn Source Confidence: Learn a set of weights $\{\mu_1, \mu_2, \ldots, \mu_S\}$, one for each source and then picks the parse that has the highest average weighted parse score for each axiom.

4. Predicate Score: Instead of selecting from one of the top parses across various sources, treat each axiom parse as a bag of premise predicates and a bag of conclusion predicates. Then, pick a subset of premise and conclusion predicates for the final parse using average scoring with thresholding.

5 Experiments

Datasets: We use a collection of grade 6-10 Indian high school math textbooks by four publishers/authors – *NCERT*, *R S Aggarwal*, *R D Sharma* and *M L Aggarwal* – a total of $5 \times 4 = 20$ textbooks to validate our model. Millions of students in India study geometry from these books every year and these books are readily available online. We manually marked chapters relevant for geometry in these books and then parsed them using Adobe Acrobat's *pdf2xml* parser. Then, we annotated geometry axioms, alignments and parses for grade 6, 7 and 8 textbooks by the four publishers/authors. We use grade 6, 7 and 8 textbook annotations for development, training, and testing, respectively. All the hyper-parameters in all the models are tuned on the development set using grid search.

GEOS used 13 types of entities and 94 functions and predicates. We add some more entities, functions and predicates to cover other more complex concepts in geometry not covered in *GEOS*. Thus, we obtain a final set of 19 entity types and 115 functions and predicates for our parsing model. We use Stanford CoreNLP (Manning et al., 2014) for feature generation. We use two datasets for evaluating our system: (a) practice and official SAT style geometry questions used in *GEOS*, and (b) an additional dataset of geometry questions collected from the aforementioned textbooks. This dataset consists of a total of 1406 SAT style questions across grades 6-10, and is approximately 7.5 times the size of the dataset used in *GEOS*. We split the dataset into training (350 questions),

Discourse Markers	Discourse markers (connectives, cue-words or cue-phrases, etc) have been shown to give good indications on discourse structure (Marcu, 2000). We build a list of discourse markers using the training set, considering the first and last tokens of each span, culled to top 100 by frequency. We use these 100 discourse markers as features. We repeat the same procedure by using part-of-speech (POS) instead of words and use them as features.
Punctuation	Punctuation at the segment border is an excellent cue. We include indicator features whether there is a punctuation at the segment border.
Text Organization	Indicator that the two text spans are part of the same (a) sentence, (b) paragraph.
XML Structure	Indicator that the two spans are in the same node in the XML hierarchy. Conjoined with the indicator feature that the two spans are part of the same paragraph.
RST Parse	We use an off-the-shelf RST parser (Feng and Hirst, 2014) and include an indicator feature that the segmentation matches the parse segmentation. We also include the RST label as a feature.
Span Lengths	The distribution of the two text spans is typically dependent on their lengths. We use the ratio of the length of the two spans as an additional feature.
Soricut and Marcu Segmenter	Soricut and Marcu (2003) (section 3.1) presented a statistical model for deciding elementary discourse unit boundaries. We use the probability given by this model retrained on our training set as feature. This feature uses both lexical and syntactic information.
Head / Common Ancestor/ Attachment Node	Head node is the word with the highest occurrence as a lexical head in the lexicalized tree among all the words in the text span. The attachment node is the parent of the head node. We have features for the head words of the left and right spans, the common ancestor (if any), the attachment node and the conjunction of the two head node words. We repeat these features with part-of-speech (POS) instead of words.
Syntax	Distance to (a) root (b) common ancestor for the nodes spanning the respective spans. We use these distances, and the difference in the distances as features.
Dominance	*Dominance* (Soricut and Marcu, 2003) is a key idea in discourse which looks at syntax trees and studies sub-trees for each span to infer a logical nesting order between the two. We use the dominance relationship is a feature. See Soricut and Marcu (2003) for details.
Span Similarity	Proportion of (a) words (b) geometry relations (c) relation-arguments shared by the two spans.
No. of Relations	Number of geometry relations represented in the two spans. We use the Lexicon Map from GEOS to compute the number of expressed geometry relations.
Relative Position	Relative position of the two lexical heads and the text split in sentence.

Table 3: Feature set for our axiom parsing model.

	Strict Comp.			Relaxed Comp.		
	P	**R**	**F**	**P**	**R**	**F**
Identification	64.3	69.3	66.7	84.3	87.9	86.1
Joint-Hard	68.0	68.1	68.0	85.4	87.1	86.2
Joint-Soft	69.7	71.1	**70.4**	86.9	88.4	**87.6**

Table 4: Test set Precision, Recall and F-measure scores for axiom identification when performed alone and when performed jointly with axiom alignment. We show results for both strict as well as relaxed comparison modes. For the joint model, we show results when we model ordering constraints as hard or soft constraints.

	P	**R**	**F**	**NMI**
Alignment	71.8	74.8	73.3	0.60
Joint-Hard	75.0	76.4	75.7	0.65
Joint-Soft	79.3	81.4	**80.3**	**0.69**

Table 5: Test set Precision, Recall, F-measure and NMI scores for axiom alignment when performed alone and when performed jointly with axiom identification. For the joint model, we show results when we model ordering constraints as hard or soft constraints.

development (150 questions) and test (906 questions) with equal proportion of grade 6-10 questions. We annotated the 500 training and development questions with ground-truth logical forms. We use the training set to train another version of *GEOS* with expanded set of entity types, functions and predicates. We call this system *GEOS++*.

Results: We first evaluate the axiom identification, alignment and parsing models individually.

For axiom identification, we compare the results of automatic identification with gold axiom identifications and compute the precision, recall and F-measure on the test set. We use strict as well as relaxed comparison. In strict comparison mode the automatically identified mentions and gold mentions must match exactly to get credit, whereas, in the relaxed comparison mode only a majority (>50%) of sentences in the automatically identified mentions and gold mentions must match to get credit. Table 4 shows the results of axiom identification where we clearly see improvements in performance when we jointly model axiom identification and alignment. This is due to the fact that both the components reinforce each other. We also observe that modeling the ordering constraints as soft constraints leads to better performance than modeling them as hard constraints. This is because the ordering of presentation of axioms is generally (yet not always) consistent across textbooks.

To evaluate axiom alignment, we first view it as a series of decisions, one for each pair of axiom mentions and compute precision, recall and F-score by comparing automatic decisions with gold decisions. Then, we also use a standard clustering metric, Normalized Mutual Information (NMI) (Strehl and Ghosh, 2002) to measure the quality of axiom mention clustering. Table 5 shows the results on the test set when gold axiom identifications are used. We observe improvements in axiom alignment performance too when we jointly model axiom identification and alignment jointly both in terms of F-score as well as NMI. Modeling ordering constraints as soft constraints again leads to better performance than modeling them as hard constraints in terms of both metrics.

To evaluate axiom parsing, we compute precision, recall and F-score in (a) deriving literals in axiom parses, as well as for (b) the final axiom parses on our test set. Table 6 shows the re-

		Literals			Full Parse		
		P	R	F	P	R	F
	GEOS	86.7	70.9	78.0	64.2	56.6	60.2
GEOS++	Single Src.	91.6	75.3	82.6	68.8	60.4	64.3
	Maj. Voting	90.2	78.5	83.9	70.0	63.3	66.5
	Avg. Score	90.8	79.6	84.9	71.7	66.4	69.0
	Src. Confid.	91.0	79.9	85.1	73.3	68.1	70.6
	Pred. Score	92.8	82.8	**87.5**	76.6	70.1	**73.2**

Table 6: Test set Precision, Recall and F-measure scores for axiom parsing. These scores are computed over literals derived in axiom parses or full axiom parses. We show results for the old *GEOS* system, for the improved *GEOS++* system with expanded entity types, functions and predicates, and for the multi-source parsers presented in this paper.

	Practice	Official	Textbook
GEOS	61	49	32
Our System	**64**	**55**	**51**
Oracle	80	78	72

Table 7: Scores for solving geometry questions on the SAT practice and official datasets and a dataset of questions from the 20 textbooks. We use SATs grading scheme that rewards a correct answer with a score of 1.0 and penalizes a wrong answer with a negative score of 0.25. *Oracle* uses gold axioms but automatic text and diagram interpretation in our logical solver. All differences between *GEOS* and our system are significant (p¡0.05 using the two-tailed paired t-test).

	Interpretability		Usefulness	
	GEOS	*O.S.*	*GEOS*	*O.S.*
Grade 6	2.7	**2.9**	2.9	**3.2**
Grade 7	3.0	**3.7**	3.3	**3.6**
Grade 8	2.7	**3.5**	3.1	**3.5**
Grade 9	2.4	**3.3**	3.0	**3.7**
Grade 10	2.8	**3.1**	3.2	**3.8**
Overall	2.7	**3.3**	3.1	**3.6**

Table 8: User study ratings for *GEOS* and our system (O.S.) by students in grade 6-10. Ten students in each grade were asked to rate the two systems on a scale of 1-5 on two facets: 'interpretability' and 'usefulness'. Each cell shows the mean rating computed over ten students in that grade for that facet.

sults of axiom parsing for *GEOS* (trained on the training set) as well as various versions of our best performing system (*GEOS*++ with our axiomatic solver) with various heuristics for multi-source parsing. The results show that our system (single source) performs better than *GEOS* as it is trained with the expanded set of entity types, functions and predicates. The results also show that the choice of heuristic is important for the multi-source parser – though all the heuristics lead to improvements over the single source parser. The average score heuristic that chooses the parse with the highest average score across sources performs better than majority voting which chooses the best parse based on a voting heuristic. Learning the confidence of every source and using a weighted average is an even better heuristic. Finally, predicate scoring which chooses the parse by scoring predicates on the premise and conclusion sides performs the best leading to 87.5 F1 score (when computed over parse literals) and 73.2 F1 score (when computed on the full parse). The high F1 score for axiom parsing on the test set shows that our approach works well and we can accurately harvest axiomatic knowledge from textbooks.

Finally, we use the extracted horn clause rules in our axiomatic solver for solving geometry problems. For this, we over-generate a set of horn clause rules by generating 3 horn clause parses for each axiom and use them as the underlying theory in prolog programs such as the one shown in Figure 3. We use weighted logical expressions for the question description and the diagram derived from *GEOS*++ as declarations, and the (normalized) score of the parsing model multiplied by the score of the joint axiom identification and alignment model as weights for the rules. Table 7 shows the results for our best end-to-end system and compares it to *GEOS* on the practice and official SAT dataset from Seo et al. (2015) as well as questions from the 20 textbooks. On all the three datasets, our system outperforms *GEOS*. Especially on the dataset from the 20 textbooks (which is indeed a harder dataset and includes more problems which require complex reasoning based on geometry), *GEOS* doesn't perform very well whereas our system still achieves a good score. *Oracle* shows the performance of our system when gold axioms (written down by an expert) are used along with automatic text and diagram interpretations in *GEOS*++. This shows that there is scope for further improvement in our approach.

Interpretability: Students around the world solve geometry problems through rigorous deduction whereas the numerical solver in *GEOS* does not provide such interpretability. One of the key benefits of our axiomatic solver is that it provides an easy-to-understand student-friendly deductive solution to geometry problems.

To test the interpretability of our axiomatic solver, we asked 50 grade 6-10 students (10 students in each grade) to use *GEOS* and our system (*GEOS*++ with our axiomatic solver) as a web-based assistive tool while learning geometry. They were each asked to rate how 'interpretable' and 'useful' the two systems were on a scale of 1-5. Table 8 shows the mean rating by students in each grade on the two facets. We can observe that students of each grade found our system to be more interpretable as well as more useful to them than *GEOS*. This study lends support to our claims about the need of an interpretable deductive solver for geometry problems.

6 Related Work

Solving Geometry Problems: While the problem of using computers to solve geometry questions is old (Feigenbaum and Feldman, 1963; Schattschneider and King, 1997; Davis, 2006), NLP and computer vision techniques were first used to solve geometry problems in Seo et al. (2015). While Seo et al. (2014) only aligned geometric shapes with their textual mentions, Seo et al. (2015) also extracted geometric relations and built *GEOS*, the first automated system to solve SAT style geometry questions. *GEOS* used a coordinate geometry based solution by translating each predicate into a set of manually written constraints. A boolean satisfiability problem posed with these constraints was used to solve the multiple-choice question. *GEOS* had two key issues: (a) it needed access to answer choices which may not always be available for such problems, and (b) it lacked the deductive geometric reasoning used by students to solve these problems. Our axiomatic solver mitigates these issues by performing deductive reasoning using axiomatic knowledge extracted from textbooks.

Information Extraction from Textbooks: Our model builds upon ideas from Information extraction (IE), which is the task of automatically extracting structured information from unstructured and/or semi-structured documents. While there has been a lot of work in IE on domains such as web documents (Chang et al., 2003; Etzioni et al., 2004; Cafarella et al., 2005; Chang et al., 2006; Banko et al., 2007; Etzioni et al., 2008; Mitchell et al., 2015) and scientific publication data (Shah et al., 2003; Peng and McCallum, 2006; Saleem and Latif, 2012), work on IE from educational material is much more sparse. Most of the research in IE from educational material deals with extracting simple educational concepts (Shah et al., 2003; Canisius and Sporleder, 2007; Yang et al., 2015; Wang et al., 2015; Liang et al., 2015; Wu et al., 2015; Liu et al., 2016b; Wang et al., 2016) or binary relational tuples (Balasubramanian et al., 2002; Clark et al., 2012; Dalvi et al., 2016) using existing IE techniques. On the other hand, our approach extracts axioms and parses them to horn clause rules. This is much more challenging. Raw application of rule mining or sequence labeling techniques used to extract information from web documents and scientific publications to educational material usually leads to poor results as the amount of redundancy in educational material is lower and the amount of labeled data is sparse. Our approach tackles these issues by making judicious use of typographical information, the redundancy of information and ordering constraints to improve the harvesting and parsing of axioms. This has not been attempted in previous work.

Language to Programs: After harvesting axioms from textbooks, we also present an approach to parse the axiom mentions to horn clause rules. This work is related to a large body of work on semantic parsing (Zelle and Mooney, 1993, 1996; Kate et al., 2005; Zettlemoyer and Collins, 2012, inter alia). Semantic parsers typically map natural language to formal programs such as database queries (Liang et al., 2011; Berant et al., 2013; Yaghmazadeh et al., 2017, inter alia), commands to robots (Shimizu and Haas, 2009; Matuszek et al., 2010; Chen and Mooney, 2011, inter alia), or even general purpose programs (Lei et al., 2013; Ling et al., 2016; Yin and Neubig, 2017; Ling et al., 2017). More specifically, Liu et al. (2016a) and Quirk et al. (2015) learn “If-Then” and “If-This-Then-That” rules, respectively. In theory, these works can be adapted to parse axiom mentions to horn-clause rules. However, this would require a large amount of supervision which would be expensive to obtain. We mitigated this issue by using redundant axiom mention extractions from multiple textbooks and then combining the parses obtained from various textbooks to achieve a better final parse for each axiom.

7 Conclusion

We presented an approach to harvest structured axiomatic knowledge from math textbooks. Our approach uses rich features based on context and typography, the redundancy of axiomatic knowledge and shared ordering constraints across multiple textbooks to accurately extract and parse axiomatic knowledge to horn clause rules. We used the parsed axiomatic knowledge to improve the best previously published automatic approach to solve geometry problems. A user-study conducted on a number of school students studying geometry found our approach to be more interpretable and useful than its predecessor. While this paper focused on harvesting geometry axioms from textbooks as a case study, it can be extended to obtain valuable structured knowledge from textbooks in areas such as science, engineering and finance.

References

Niranjan Balasubramanian, Stephen Soderland, Oren Etzioni Mausam, and Robert Bart. 2002. out of the box information extraction: a case study using bio-medical texts. Technical report.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2670–2676.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544.

Michael J. Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni. 2005. Knowitnow: Fast, scalable information extraction from the web. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*, pages 563–570.

Sander Canisius and Caroline Sporleder. 2007. Bootstrapping information extraction from field books. In *EMNLP-CoNLL*, pages 827–836.

Yee Seng Chan and Hwee Tou Ng. 2008. Maxsim: A maximum similarity metric for machine translation evaluation. In *The 2008 Annual Conference of the Association for Computational Linguistics (ACL)*.

Chia-Hui Chang, Chun-Nan Hsu, and Shao-Cheng Lui. 2003. Automatic information extraction from semi-structured web pages by pattern discovery. *Decision Support Systems*, 35(1):129–147.

Chia-Hui Chang, Mohammed Kayed, Moheb R Girgis, and Khaled F Shaalan. 2006. A survey of web information extraction systems. *IEEE transactions on knowledge and data engineering*, 18(10):1411–1428.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011)*, pages 859–865.

Peter Clark and Oren Etzioni. 2016. My computer is an honor student - but how intelligent is it? standardized tests as a measure of ai. In *Proceedings of AI Magazine*.

Peter Clark, Phil Harrison, Niranjan Balasubramanian, and Oren Etzioni. 2012. Constructing a textual kb from a biology textbook. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 74–78. Association for Computational Linguistics.

Bhavana Dalvi, Sumithra Bhakthavatsalam, Chris Clark, Peter Clark, Oren Etzioni, Anthony Fader, and Dirk Groeneveld. 2016. IKE - an interactive tool for knowledge extraction. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction, AKBC@NAACL-HLT 2016, San Diego, CA, USA, June 17, 2016*, pages 12–17.

Tom Davis. 2006. Geometry with computers. Technical report.

Michael Denkowski and Alon Lavie. 2010. Extending the meteor machine translation evaluation metric to the phrase level. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 250–253. Association for Computational Linguistics.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. 2008. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74.

Oren Etzioni, Michael J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Methods for domain-independent information extraction from the web: An experimental comparison. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pages 391–398.

Edward A Feigenbaum and Julian Feldman. 1963. *Computers and thought*. The AAAI Press.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521.

Akira Fujita, Akihiro Kameda, Ai Kawazoe, and Yusuke Miyao. 2014. Overview of todai robot project and evaluation framework of its nlp-based problem solving. *World History*, 36:36.

Rohit J Kate, Yuk Wah, Wong Raymond, and J Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of AAAI-05*. Citeseer.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of ACL*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

Tao Lei, Fan Long, Regina Barzilay, and Martin C Rinard. 2013. From natural language specifications to program input parsers. Association for Computational Linguistics (ACL).

Chen Liang, Zhaohui Wu, Wenyi Huang, and C Lee Giles. 2015. Measuring prerequisite relations among concepts. In *EMNLP*, pages 1668–1674.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 590–599. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. *arXiv preprint arXiv:1603.06744*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction for rationale generation: Learning to solve and explain algebraic word problems. Association for Computational Linguistics (ACL) – To appear.

Chang Liu, Xinyun Chen, Eui Chul Shin, Mingcheng Chen, and Dawn Song. 2016a. Latent attention for if-then program synthesis. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4574–4582. Curran Associates, Inc.

Hanxiao Liu, Wanli Ma, Yiming Yang, and Jaime Carbonell. 2016b. Learning concept graphs from online educational data. *Journal of Artificial Intelligence Research*, 55:1059–1090.

William C Mann and Sandra A Thompson. 1988. {Rhetorical Structure Theory: Toward a functional theory of text organisation}. *Text*, 3(8):234–281.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Daniel Marcu. 2000. *The theory and practice of discourse parsing and summarization*.

Cynthia Matuszek, Dieter Fox, and Karl Koscher. 2010. Following directions using statistical machine translation. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 251–258. IEEE.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*.

Fuchun Peng and Andrew McCallum. 2006. Information extraction from research papers using conditional random fields. *Information processing & management*, 42(4):963–979.

Chris Quirk, Raymond J. Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 878–888.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

Mrinmaya Sachan, Avinava Dubey, Eric P Xing, and Matthew Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Mrinmaya Sachan, Kumar Avinava Dubey, and Eric P. Xing. 2016. Science question answering using instructional materials. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Mrinmaya Sachan and Eric P. Xing. 2016. Easy questions first? A case study on curriculum learning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Ozair Saleem and Seemab Latif. 2012. Information extraction from research papers by data integration and data validation from multiple header extraction sources. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, pages 177–180.

Doris Schattschneider and James King. 1997. *Geometry Turned On: Dynamic Software in Learning, Teaching, and Research*. Mathematical Association of America Notes.

Carissa Schoenick, Peter Clark, Oyvind Tafjord, Peter D. Turney, and Oren Etzioni. 2016. Moving beyond the turing test with the allen AI science challenge. *CoRR*, abs/1604.04315.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. 2014. Diagram understanding in geometry questions. In *Proceedings of AAAI*.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: combining text and diagram interpretation. In *Proceedings of EMNLP*.

Parantu K Shah, Carolina Perez-Iratxeta, Peer Bork, and Miguel A Andrade. 2003. Information extraction from full text scientific articles: Where are the keywords? *BMC bioinformatics*, 4(1):20.

Nobuyuki Shimizu and Andrew R. Haas. 2009. Learning to follow navigational route instructions. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 1488–1493.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics.

Alexander Strehl and Joydeep Ghosh. 2002. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617.

Shuting Wang, Chen Liang, Zhaohui Wu, Kyle Williams, Bart Pursel, Benjamin Brautigam, Sherwyn Saul, Hannah Williams, Kyle Bowen, and C Lee Giles. 2015. Concept hierarchy extraction from textbooks. In *Proceedings of the 2015 ACM Symposium on Document Engineering*, pages 147–156. ACM.

Shuting Wang, Alexander Ororbia, Zhaohui Wu, Kyle Williams, Chen Liang, Bart Pursel, and C Lee Giles. 2016. Using prerequisites to extract concept maps from textbooks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 317–326. ACM.

Jian Wu, Jason Killian, Huaiyu Yang, Kyle Williams, Sagnik Ray Choudhury, Suppawong Tuarob, Cornelia Caragea, and C. Lee Giles. 2015. Pdfmef: A multi-entity knowledge extraction framework for scholarly documents and semantic search. In *Proceedings of the 8th International Conference on Knowledge Capture*, K-CAP 2015.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Type- and content-driven synthesis of SQL queries from natural language. *CoRR*, abs/1702.01168.

Yiming Yang, Hanxiao Liu, Jaime G. Carbonell, and Wanli Ma. 2015. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*, pages 159–168.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. A lightweight and high performance monolingual word aligner. In *ACL (2)*, pages 702–707.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *The 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada.

John M. Zelle and Raymond J. Mooney. 1993. Learning semantic grammars with constructive inductive logic programming. In *Proceedings of the 11th National Conference on Artificial Intelligence. Washington, DC, USA, July 11-15, 1993.*, pages 817–822.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*.

Luke S Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*.

RACE: Large-scale ReAding Comprehension Dataset From Examinations

Guokun Lai[*] and **Qizhe Xie**[*] and **Hanxiao Liu** and **Yiming Yang** and **Eduard Hovy**
{guokun, qzxie, hanxiaol, yiming, hovy}@cs.cmu.edu
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We present RACE, a new dataset for benchmark evaluation of methods in the reading comprehension task. Collected from the English exams for middle and high school Chinese students in the age range between 12 to 18, RACE consists of near 28,000 passages and near 100,000 questions generated by human experts (English instructors), and covers a variety of topics which are carefully designed for evaluating the students' ability in understanding and reasoning. In particular, the proportion of questions that requires reasoning is much larger in RACE than that in other benchmark datasets for reading comprehension, and there is a significant gap between the performance of the state-of-the-art models (43%) and the ceiling human performance (95%). We hope this new dataset can serve as a valuable resource for research and evaluation in machine comprehension. The dataset is freely available at http://www.cs.cmu.edu/~glai1/data/race/ and the code is available at https://github.com/qizhex/RACE_AR_baselines

1 Introduction

Constructing an intelligence agent capable of understanding text as people is the major challenge of NLP research. With recent advances in deep learning techniques, it seems possible to achieve human-level performance in certain language understanding tasks, and a surge of effort has been devoted to the machine comprehension task where people aim to construct a system with the ability to answer questions related to a document that it has to comprehend (Chen et al., 2016; Kadlec et al., 2016; Dhingra et al., 2016; Yang et al., 2017).

Towards this goal, several large-scale datasets (Rajpurkar et al., 2016; Onishi et al., 2016; Hill et al., 2015; Trischler et al., 2016; Hermann et al., 2015) have been proposed, which allow researchers to train deep learning systems and obtain results comparable to the human performance. While having a suitable dataset is crucial for evaluating the system's true ability in reading comprehension, the existing datasets suffer several critical limitations. Firstly, in all datasets, the candidate options are directly extracted from the context (as a single entity or a text span), which leads to the fact that lots of questions can be solved trivially via word-based search and context-matching without deeper reasoning; this constrains the types of questions as well. Secondly, answers and questions of most datasets are either crowd-sourced or automatically-generated, bringing a significant amount of noises in the datasets and limits the ceiling performance by domain experts, such as 82% for Childrens Book Test and 84% for Who-did-What. Yet another issue in existing datasets is that the topic coverages are often biased due to the specific ways that the data were initially collected, making it hard to evaluate the ability of systems in text comprehension over a broader range of topics.

To address the aforementioned limitations, we constructed a new dataset by collecting a large set of questions, answers and associated passages in the English exams for middle-school and high-school Chinese students within the 12–18 age range. Those exams were designed by domain experts (instructors) for evaluating the reading comprehension ability of students, with ensured quality and broad topic coverage. Furthermore, the answers by machines or by humans can be objectively graded for evaluation

[*] indicates equal contribution

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 785–794
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

and comparison using the same evaluation metrics. Although efforts have been made with a similar motivation, including the MCTest dataset created by (Richardson et al., 2013) (containing 500 passages and 2000 questions) and several others (Peñas et al., 2014; Rodrigo et al., 2015; Khashabi et al., 2016; Shibuki et al., 2014), the usefulness of those datasets is significantly restricted due to their small sizes, especially not suitable for training powerful deep neural networks whose success relies on the availability of relatively large training sets.

Our new dataset, namely RACE, consists of 27,933 passages and 97,687 questions. After reading each passage, each student is asked to answer several questions where each question is provided with four candidate answers – only one of them is correct . Unlike existing datasets, both the questions and candidate answers in RACE are not restricted to be the text spans in the original passage; instead, they can be described in any words. A sample from our dataset is presented in Table 1.

Our latter analysis shows that correctly answering a large portion of questions in RACE requires the ability of reasoning, the most important feature as a machine comprehension dataset (Chen et al., 2016). RACE also offers two important subdivisions of the reasoning types in its questions, namely passage summarization and attitude analysis, which have not been introduced by the any of the existing large-scale datasets to our knowledge.

In addition, compared to other existing datasets where passages are either domain-specific or of a single fixed style (namely news stories for CNN/-Daily Mail, NEWSQA and Who-did-What, fiction stories for Children's Book Test and Book Test, and Wikipedia articles for SQUAD), passages in RACE almost cover all types of human articles, such as news, stories, ads, biography, philosophy, etc., in a variety of styles. This comprehensiveness of topic/style coverage makes RACE a desirable resource for evaluating the reading comprehension ability of machine learning systems in general.

The advantages of our proposed dataset over existing large datasets in machine reading comprehension can be summarized as follows:

- All questions and candidate options are generated by human experts, which are intentionally designed to test human agent's ability in reading comprehension. This makes RACE a relatively accurate indicator for reflecting the text comprehension ability of machine learning systems under human judge.
- The questions are substantially more difficult than those in existing datasets, in terms of the large portion of questions involving reasoning. At the meantime, it is also sufficiently large to support the training of deep learning models.
- Unlike existing large-scale datasets, candidate options in RACE are human generated sentences which may not appear in the original passage. This makes the task more challenging and allows a rich type of questions such as passage summarization and attitude analysis.
- Broad coverage in various domains and writing styles: a desirable property for evaluating generic (in contrast to domain/style-specific) comprehension ability of learning models.

2 Related Work

In this section, we briefly outline existing datasets for the machine reading comprehension task, including their strengths and weaknesses.

2.1 MCTest

MCTest (Richardson et al., 2013) is a popular dataset for question answering in the same format as RACE, where each question is associated with four candidate answers with a single correct answer. Although questions in MCTest are of high-quality ensured by careful examinations through crowdsourcing, it contains only 500 stores and 2000 questions, which substantially restricts its usage in training advanced machine comprehension models. Moreover, while MCTest is designed for 7 years old children, RACE is constructed for middle and high school students at 12–18 years old hence is more complicated and requires stronger reasoning skills. In other words, RACE can be viewed as a larger and more difficult version of the MCTest dataset.

2.2 Cloze-style datasets

The past few years have witnessed several large-scale cloze-style datasets (Hermann et al., 2015; Hill et al., 2015; Bajgar et al., 2016; Onishi et al., 2016), whose questions are formulated by obliterating a word or an entity in a sentence.

Passage:

In a small village in England about 150 years ago, a mail coach was standing on the street. It didn't come to that village often. People had to pay a lot to get a letter. The person who sent the letter didn't have to pay the postage, while the receiver had to.
"Here's a letter for Miss Alice Brown," said the mailman.
" I'm Alice Brown," a girl of about 18 said in a low voice.
Alice looked at the envelope for a minute, and then handed it back to the mailman.
"I'm sorry I can't take it, I don't have enough money to pay it", she said.
A gentleman standing around were very sorry for her. Then he came up and paid the postage for her.
When the gentleman gave the letter to her, she said with a smile, " Thank you very much, This letter is from Tom. I'm going to marry him. He went to London to look for work. I've waited a long time for this letter, but now I don't need it, there is nothing in it."
"Really? How do you know that?" the gentleman said in surprise.
"He told me that he would put some signs on the envelope. Look, sir, this cross in the corner means that he is well and this circle means he has found work. That's good news."
The gentleman was Sir Rowland Hill. He didn't forgot Alice and her letter.
"The postage to be paid by the receiver has to be changed," he said to himself and had a good plan.
"The postage has to be much lower, what about a penny? And the person who sends the letter pays the postage. He has to buy a stamp and put it on the envelope." he said . The government accepted his plan. Then the first stamp was put out in 1840. It was called the "Penny Black". It had a picture of the Queen on it.

Questions:

1): The first postage stamp was made _.
A. in England B. in America C. by Alice D. in 1910

2): The girl handed the letter back to the mailman because _ .
A. she didn't know whose letter it was
B. she had no money to pay the postage
C. she received the letter but she didn't want to open it
D. she had already known what was written in the letter

3): We can know from Alice's words that _ .
A. Tom had told her what the signs meant before leaving
B. Alice was clever and could guess the meaning of the signs
C. Alice had put the signs on the envelope herself
D. Tom had put the signs as Alice had told him to

4): The idea of using stamps was thought of by _ .
A. the government
B. Sir Rowland Hill
C. Alice Brown
D. Tom

5): From the passage we know the high postage made _ .
A. people never send each other letters
B. lovers almost lose every touch with each other
C. people try their best to avoid paying it
D. receivers refuse to pay the coming letters

Answer: ADABC

Table 1: Sample reading comprehension problems from our dataset.

CNN/Daily Mail (Hermann et al., 2015) are the largest machine comprehension datasets with 1.4M questions. However, both require limited reasoning ability (Chen et al., 2016). In fact, the best machine performance obtained by researchers (Chen et al., 2016; Dhingra et al., 2016) is close to human's performance on CNN/Daily Mail.

Childrens Book Test (CBT) (Hill et al., 2015) and Book Test (BT) (Bajgar et al., 2016) are constructed in a similar manner. Each passage in CBT consist of 20 contiguous sentences extracted from children's books and the next (21st) sentence is used to make the question. The main difference between the two datasets is the size of BT being 60 times larger. Machine comprehension models have also matched human performance on CBT (Bajgar et al., 2016).

Who Did What (WDW) (Onishi et al., 2016) is yet another cloze-style dataset constructed from the LDC English Gigaword newswire corpus. The authors generate passages and questions by picking two news articles describing the same event, using one as the passage and the other as the question.

High noise is inevitable in cloze-style datasets due to their automatic generation process, which is reflected in the human performance on these datasets: 82% for CBT and 84% for WDW.

2.3 Datasets with Span-based Answers

In datasets such as SQUAD (Rajpurkar et al., 2016), NEWSQA (Trischler et al., 2016) and MS MARCO (Nguyen et al., 2016), the answer to each question is in the form of a text span in the article. Articles of SQUAD, NEWSQA and MS MARCO come from Wikipedia, CNN news and the Bing search engine respectively. The answer to a certain question may not be unique and could be multiple spans. Instead of evaluating the accuracy, researchers need to use F1 score, BLEU (Papineni et al., 2002) or ROUGE (Lin and Hovy, 2003) as metrics, which measure the overlap between the prediction and ground truth answers since the questions come without candidate spans.

Datasets with span-based answers are challenging as the space of possible spans is usually large. However, restricting answers to be text spans in the context passage may be unrealistic and more importantly, may not be intuitive even for humans, indicated by the suffered human performance of 80.3% on SQUAD (or 65% claimed by Trischler et al. (2016)) and 46.5% on NEWSQA. In other words, the format of span-based answers may not necessarily be a good examination of reading comprehension of machines whose aim is to approach the comprehension ability of *humans*.

2.4 Datasets from Examinations

There have been several datasets extracted from examinations, aiming at evaluating systems under the same conditions as how humans are evaluated in schools. E.g., the AI2 Elementary School Science Questions dataset (Khashabi et al., 2016) contains 1080 questions for students in elementary schools; NTCIR QA Lab (Shibuki et al., 2014) evaluates systems by the task of solving real-world university entrance exam questions; The Entrance Exams task at CLEF QA Track (Peñas et al., 2014; Rodrigo et al., 2015) evaluates the system's reading comprehension ability. However, data provided in these existing tasks are far from sufficient for the training of advanced data-driven machine reading models, partially due to the expensive data generation process by human experts.

To the best of our knowledge, RACE is the first *large-scale* dataset of this type, where questions are created based on exams designed to evaluate human performance in reading comprehension.

3 Data Analysis

In this section, we study the nature of questions covered in RACE at a detailed level. Specifically, we present the dataset statistics in Section 3.1, and then analyze different reasoning/question types in RACE in the remaining subsections.

3.1 Dataset Statistics

As mentioned in section 1, RACE is collected from English examinations designed for 12–15 year-old middle school students, and 15–18 year-old high school students in China. To distinguish the two subgroups with drastic difficulty gap, RACE-M denotes the middle school examinations and RACE-H denotes high school examinations. We split 5% data as the development set and 5% as the test set for RACE-M and RACE-H respectively. The number of samples in each set is shown in Table 2. The statistics for RACE-M and RACE-H is summarized in Table 3. We can find that the length of the passages and the vocabulary size in the RACE-H are much larger than that of the RACE-M, an evidence of the higher difficulty of high school examinations.

However, notice that since the articles and questions are selected and designed to test Chinese students learning English as a foreign language, the vocabulary size and the complexity of the language constructs are simpler than news articles and Wikipedia articles in other QA datasets.

3.2 Reasoning Types of the Questions

To get a comprehensive picture about the reasoning difficulty requirement of RACE, we conduct human annotations of questions types. Following Chen et al. (2016); Trischler et al. (2016), we stratify the questions into five classes as follows with ascending order of difficulty:

- Word matching: The question exactly matches a span in the article. The answer is self-evident.

- Paraphrasing: The question is entailed or paraphrased by exactly one sentence in the passage. The answer can be extracted within the sentence.

- Single-sentence reasoning: The answer could be inferred from a single sentence of the article by recognizing incomplete information or conceptual overlap.

- Multi-sentence reasoning: The answer must be inferred from synthesizing information distributed across multiple sentences.

- Insufficient/Ambiguous: The question has no answer or the answer is not unique based on the given passage.

We refer readers to (Chen et al., 2016; Trischler et al., 2016) for examples of each category.

To obtain the proportion of different question types, we sample 100 passages from RACE (50 from RACE-M and 50 from RACE-H), all of which have 5 questions hence there are 500 questions in total. We put the passages on Amazon Mechanical Turk[1], and a Hit is generated by a passage

[1] https://www.mturk.com/mturk/welcome

Dataset	RACE-M			RACE-H			RACE			
Subset	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test	All
# passages	6,409	368	362	18,728	1,021	1,045	25,137	1,389	1,407	27,933
# questions	25,421	1,436	1,436	62,445	3,451	3,498	87,866	4,887	4,934	97,687

Table 2: The separation of the training, development and test sets of RACE-M,RACE-H and RACE

Dataset	RACE-M	RACE-H	RACE
Passage Len	231.1	353.1	321.9
Question Len	9.0	10.4	10.0
Option Len	3.9	5.8	5.3
Vocab size	32,811	125,120	136,629

Table 3: Statistics of RACE where Len denotes length and Vocab denotes Vocabulary.

with 5 questions. Each question is labeled by two crowdworkers. We require the turkers to both answer the questions and label the reasoning type. We pay \$0.70 and \$1.00 per passage in RACE-M and RACE-H respectively, and restrict the access to master turkers only. Finally, we get 1000 labels for the 500 questions.

The statistics about the reasoning type is summarized in Table 4. The higher difficulty level of RACE is justified by its higher ratio of reasoning questions in comparison to CNN, SQUAD and NEWSQA. Specifically, 59.2% questions of RACE are either in the category of single-sentence reasoning or in the category of multi-sentence reasoning, while the ratio is 21%, 20.5% and 33.9% for CNN, SQUAD and NEWSQA respectively. Also notice that the ratio of word matching questions on RACE is only 15.8%, the lowest among several categories. In addition, questions in RACE-H are more complex than questions in RACE-M since RACE-M has more word matching questions and fewer reasoning questions.

3.3 Subdividing Reasoning Types

To better understand our dataset and facilitate future research, we list the subdivisions of questions under the reasoning category. We find the most frequent reasoning subdivisions include: detail reasoning, whole-picture understanding, passage summarization, attitude analysis and world knowledge. One question may fall into multiple divisions. Definition of these subdivisions and their associated examples are as follows:

1. Detail reasoning: to answer the question, the agent should be clear about the details of the passage. The answer appears in the passage but it cannot be found by simply matching the question with the passage. For example, Question 1 in the sample passage falls into this category.

2. Whole-picture reasoning: the agent needs to understand the whole picture of the story to obtain the correct answer. For example, to answer the Question 2 in the sample passage, the agent is required to comprehend the entire story.

3. Passage summarization: The question requires the agent to select the best summarization of the passage among four candidate summarizations. A typical question of this type is "The main idea of this passage is __.". An example question can be found in Appendix A.1.

4. Attitude analysis: The question asks about the opinions/attitudes of the author or a character in the story towards somebody or something, e.g.,

- *Evidence*: "...Many people optimistically thought industry awards for better equipment would stimulate the production of quieter appliances. It was even suggested that noise from building sites could be alleviated ..."
- *Question*: What was the author's attitude towards the industry awards for quieter?
- *Options*: A.suspicious B.positive C.enthusiastic D.indifferent

5. World knowledge: Certain external knowledge is needed. Most frequent questions under this category involve simple arithmetic.

- *Evidence*: "The park is open from 8 am to 5 pm."
- *Question*: The park is open for __ hours a day.
- *Options*: A.eight B.nine C.ten D.eleven

To the best of our knowledge, questions like passage summarization and attitude analysis have not been introduced by any of the existing large-scale machine comprehension datasets. Both are crucial components in evaluating humans' reading comprehension abilities.

Dataset	RACE-M	RACE-H	RACE	CNN	SQUAD	NEWSQA
Word Matching	29.4%	11.3%	15.8%	13.0%†	39.8%*	32.7%*
Paraphrasing	14.8%	20.6%	19.2%	41.0%†	34.3%*	27.0%*
Single-Sentence Reasoning	31.3%	34.1%	33.4%	19.0%†	8.6%*	13.2%*
Multi-Sentence Reasoning	22.6%	26.9%	25.8%	2.0%†	11.9%*	20.7%*
Ambiguous/Insufficient	1.8%	7.1%	5.8%	25.0%†	5.4%*	6.4%*

Table 4: Statistic information about Reasoning type in different datasets. * denotes the numbers coming from (Trischler et al., 2016) based on 1000 samples per dataset, and numbers with † come from (Chen et al., 2016).

4 Collection Methodology

We collected the raw data from three large free public websites[2][3][4] in China[5], where the reading comprehension problems are extracted from English examinations designed by teachers in China. The data before cleaning contains 137,918 passages and 519,878 questions in total, where there are 38,159 passages with 156,782 questions in the middle school group, and 99,759 passages with 363,096 questions in the high school group.

The following filtering steps are conducted to clean the raw data. Firstly, we remove all problems and questions that do not have the same format as our problem setting, e.g., a question would be removed if the number of its options is not four. Secondly, we filter all articles and questions that are not self-contained based on the text information, i.e. we remove the articles and questions containing images or tables. We also remove all questions containing keywords "underlined" or "paragraph", since it is difficult to reproduce the effect of underlines and the paragraph segment information. Thirdly, we remove all duplicated articles.

On one of the websites (xkw.com), the answers are stored as images. We used two standard OCR programs tesseract [6] and ABBYY FineReader [7] to process the images. We remove all the answers that two software disagree. The OCR task is easy since we only need to recognize printed alphabet A, B, C, D with a standard font. Finally, we get the cleaned dataset RACE, with 27,933 passages and 97,687 questions.

5 Experiments

In this section, we compare the performance of several state-of-the-art reading comprehension models with human performance. We use accuracy as the metric to evaluate different models.

5.1 Methods for Comparison

Sliding Window Algorithm Firstly, we build the rule-based baseline introduced by Richardson et al. (2013). It chooses the answer having the highest matching score. Specifically, it first concatenates the question and the answer and then calculates the TF-IDF style matching score between the concatenated sentence with every window (a span of text) of the article. The window size is decided by the model performance in the training and dev sets.

Stanford Attentive Reader Stanford Attentive Reader (Stanford AR) (Chen et al., 2016) is a strong model that achieves state-of-the-art results on CNN/Daily Mail. Moreover, the authors claim that their model has nearly reached the ceiling performance on these two datasets.

Suppose that the triple of passage, question and options is denoted by $(p, q, o_{1,\cdots,4})$. We first employ bidirectional GRUs to encode p and q respectively into $h_1^p, h_2^p, \dots, h_n^p$ and h^q. Then we summarize the most relevant part of the passage into s^p with an attention model. Following Chen et al. (2016), we adopt a bilinear attention form. Specifically,

$$\begin{aligned} \alpha_i &= \text{Softmax}_i((h_i^p)^T W_1 h^q) \\ s^p &= \sum_i \alpha_i h_i^p \end{aligned} \tag{1}$$

Similarly, we use bidirectional GRUs to encode option o_i into a vector h^{o_i}. Finally, we compute the matching score between the i-th option $(i = 1, \cdots, 4)$ and the summarized passage using

[2] http://www.21cnjy.com/

[3] http://5utk.ks5u.com/

[4] http://zujuan.xkw.com/

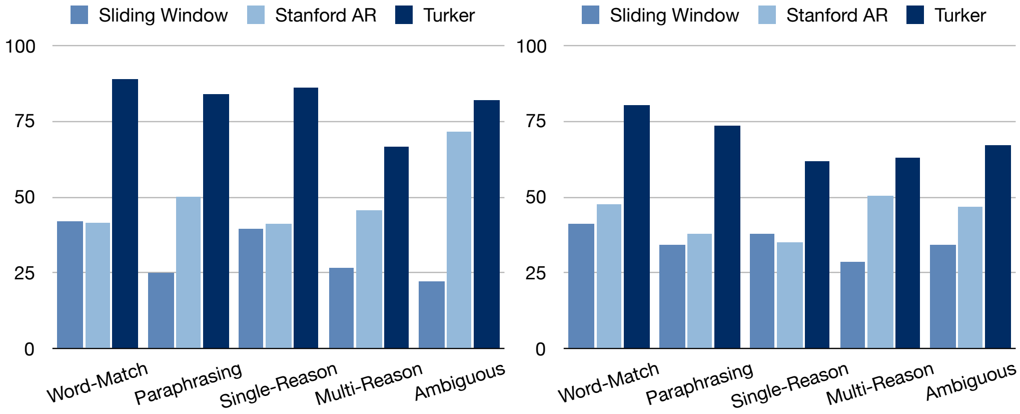
[5] We checked that our dataset does not include example questions of exams with copyright, such as SSAT, SAT, TOEFL and GRE.

[6] https://github.com/tesseract-ocr

[7] https://www.abbyy.com/FineReader

	RACE-M	RACE-H	RACE	MCTest	CNN	DM	CBT-N	CBT-C	WDW
Random	24.6	25.0	24.9	24.8	0.06	0.06	10.6	10.2	32.0†
Sliding Window	37.3	30.4	32.2	51.5†	24.8	30.8	16.8†	19.6†	48.0†
Stanford AR	44.2	43.0	43.3	–	73.6†	76.6†	–	–	64.0†
GA	43.7	44.2	44.1	–	77.9†	80.9†	70.1†	67.3†	71.2†
Turkers	85.1	69.4	73.3	–	–	–	–	–	–
Ceiling Performance	95.4	94.2	94.5	–	–	–	81.6†	81.6†	84†

Table 5: Accuracy of models and human on the each dataset, where † denotes the results coming from previous publications. DM denotes Daily Mail and WDW denotes Who-Did-What .



(a) RACE-M

(b) RACE-H

Figure 1: Test accuracy of different baselines on each reasoning type category introduced in Section 3.2, where Word-Match, Single-Reason, Multi-Reason and Ambiguous are the abbreviations for Word matching, Single-sentence Reasoning, Multi-sentence Reasoning and Insufficient/Ambiguous respectively.

a bilinear attention. We pass the scores through softmax to get a probability distribution. Specifically, the probability of option i being the right answer is calculated as

$$p_i = \text{Softmax}_i(h^{o_i} W_2 s^d) \tag{2}$$

Gated-Attention Reader Gated AR (Dhingra et al., 2016) is the state-of-the-art model on multiple datasets. To build query-specific representations of tokens in the document, it employs an attention mechanism to model multiplicative interactions between the query embedding and the document representation. With a multi-hop architecture, GA also enables a model to scan the document and the question iteratively for multiple passes. In other words, the multi-hop structure makes it possible for the reader to refine token representations iteratively and the attention mechanism find the most relevant part of the document. We refer readers to (Dhingra et al., 2016) for more details.

After obtaining a query specific document representation s^d, we use the same method as bilinear operation listed in Equation 2 to get the output.

Note that our implementation slightly differs from the original GA reader. Specifically, the Attention Sum layer is not applied at the final layer and no character-level embeddings are used.

Implementation Details We follow Chen et al. (2016) in our experiment settings. The vocabulary size is set to $50k$. We choose word embedding size $d = 100$ and use the 100-dimensional Glove word embedding (Pennington et al., 2014) as embedding initialization. GRU weights are initialized from Gaussian distribution $\mathcal{N}(0, 0.1)$. Other parameters are initialized from a uniform distribution on $(-0.01, 0.01)$. The hidden dimensionality is set to 128 and the number of layers is set to one for both Stanford AR and GA. We use vanilla stochastic gradient descent (SGD) to train our models. We apply dropout on word embeddings and the gradient is clipped when the norm of the gradient is larger than 10. We use a grid search on validation set to choose the learning rate within $\{0.05, 0.1, 0.3, 0.5\}$ and dropout rate within $\{0.2, 0.5, 0.7\}$. The highest accuracy on validation set is obtained by setting learning rate to 0.1 for Stanford AR and 0.3 for GA and dropout rate to 0.5. The data of RACE-M and RACE-H is used together to train our model and testing is performed separately.

5.2 Human Evaluation

As described in section 3.2, a randomly sampled subset of test set has been labeled by Amazon Turkers, which contains 500 questions with half from RACE-H and with the other half from RACE-M. The turkers' performance is 85% for RACE-M and 70% for RACE-H. However, it is hard to guarantee that every turker performs the survey carefully, given the difficult and long passages of high school problems. Therefore, to obtain the ceiling human performance on RACE, we manually labeled the proportion of valid questions. A question is valid if it is unambiguous and has a correct answer. We found that 94.5% of the data is valid, which sets the ceiling human performance. Similarly, the ceiling performance on RACE-M and RACE-H is 95.4% and 94.2% respectively.

5.3 Main Results

We compare models' and human ceiling performance on datasets which have the same evaluation metric with RACE. The compared datasets include RACE, MCTest, CNN/Daily Mail (CNN and DM), CBT and WDW. On CBT, we report performance on two subsets where the missing token is either a common noun (CBT-C) or name entity (CBT-N) since the language models have already reached human-level performance on other types (Hill et al., 2015). The comparison is shown in Table 5.

Performance of Sliding Window We first compare MCTest with RACE using Sliding Window, where it is unable to train Stanford AR and Gated AR on MCTest's limited training data. Sliding Window achieves an accuracy of 51.5% on MCTest while only 37.3% on RACE, meaning that to answer the questions of RACE requires more reasoning than MCTest.

The performance of sliding window on RACE is not directly comparable with CBT and WDW since CBT has ten candidate answers for each question and WDW has an average of three. Instead, we evaluate the performance improvement of sliding window on the random baseline. Larger improvement indicates more questions solvable by simple matching. On RACE, Sliding Window is 28.6% better than the random baseline, while the improvement is 58.5%, 92.2% and 50% for CBT-N, CBT-C and WDW.

The accuracy on RACE-M (37.3%) and RACE-H (30.4%) indicates that the middle school questions are simpler based on the matching algorithm.

Performance of Neural Models We further compare the difficulty of different datasets by state-of-the-art neural models' performance. A lower performance means that more problems are unsolvable by machines. The Stanford AR and Gated AR achieve an accuracy of only 43.3% and 44.1% on RACE while their accuracy is much higher on CNN/Daily Mail, Childrens Book Test and Who-Did-What. It justifies the fact that, among current large-scale machine comprehension datasets, RACE is the most challenging one.

Human Ceiling Performance The human performance is 94.5% which shows our data is quite clean compared to other large-scale machine comprehension datasets. Since we cannot enforce every turker do the test cautiously, the result shows a gap between turkers' performance and human performance. Reasonably, problems in the high school group with longer passages and more complex questions lead to more significant divergence. Nevertheless, the start-of-the-art models still have a large room to be improved to reach turkers' performance. The performance gap is 41% for the middle school problems and 25% for the high school problems. What's more, The performance of Stanford AR and GA is only less than a half of the ceiling human performance, which indicates that to match the humans' reading comprehension ability, we still have a long way to go.

5.4 Reason Types Analysis

We evaluate human and models on different types of questions, shown in Figure 1. Turkers do the best on word matching problems while doing the worst on reasoning problems. Sliding window performs better on word matching than problems needing reasoning or paraphrasing. Surprisingly, Stanford AR does not have a stronger performance on the word matching category than reasoning categories. A possible reason is that the proportion of data in reasoning categories is larger than that of data. Also, the candidate answers of simple matching questions may share similar word embeddings. For example, if the question is about color, it is difficult to distinguish candidate answers, "green", "red", "blue" and "yellow", in the embedding vector space. The similar performance on different categories also explains the reason that the performance of the neural models is close in the middle and high school groups in Table 5.

6 Conclusion

We introduce a large, high-quality dataset for reading comprehension that is carefully designed to examine human ability on this task. Some desirable properties of RACE include the broad coverage of domains/styles and the richness in the question format. Most importantly, it requires substantially more reasoning to do well on RACE than on other datasets, as there is a significant gap between the performance of state-of-the-art machine comprehension models and that of the human. We hope this dataset will stimulate the development of more advanced machine comprehension models.

Acknowledgement

We would like to thank Graham Neubig for suggestions on the draft and Diyi Yang's help on obtaining the crowdsourced labels.

This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the DEFT program.

References

Ondrej Bajgar, Rudolf Kadlec, and Jan Kleindienst. 2016. Embracing data abundance: Booktest dataset for reading comprehension. *arXiv preprint arXiv:1610.00956* .

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858* .

Bhuwan Dhingra, Hanxiao Liu, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549* .

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301* .

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547* .

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Peter Clark, Oren Etzioni, and Dan Roth. 2016. Question answering via integer programming over semi-structured knowledge. *arXiv preprint arXiv:1604.06076* .

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 71–78.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* .

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. *arXiv preprint arXiv:1608.05457* .

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Anselmo Peñas, Yusuke Miyao, Álvaro Rodrigo, Eduard H Hovy, and Noriko Kando. 2014. Overview of clef qa entrance exams task 2014. In *CLEF (Working Notes)*. pages 1194–1200.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* .

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*. volume 3, page 4.

Álvaro Rodrigo, Anselmo Peñas, Yusuke Miyao, Eduard H Hovy, and Noriko Kando. 2015. Overview of clef qa entrance exams task 2015. In *CLEF (Working Notes)*.

Hideyuki Shibuki, Kotaro Sakamoto, Yoshinobu Kano, Teruko Mitamura, Madoka Ishioroshi, Kelly Y Itakura, Di Wang, Tatsunori Mori, and Noriko Kando. 2014. Overview of the ntcir-11 qa-lab task. In *NTCIR*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830* .

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W Cohen. 2017. Semi-supervised qa with generative domain-adaptive nets. *arXiv preprint arXiv:1702.02206* .

Beyond Sentential Semantic Parsing: Tackling the Math SAT with a Cascade of Tree Transducers

Mark Hopkins, Cristian Petrescu-Prahova, Roie Levin,
Ronan Le Bras, Alvaro Herrasti, and **Vidur Joshi**
Allen Institute for Artificial Intelligence
Seattle, WA
{markh,cristipp,roiel,rlebras,alvaroh,vidurj}@allenai.org

Abstract

We present an approach for answering questions that span multiple sentences and exhibit sophisticated cross-sentence anaphoric phenomena, evaluating on a rich source of such questions – the math portion of the Scholastic Aptitude Test (SAT). By using a tree transducer cascade as its basic architecture, our system (called EUCLID) propagates uncertainty from multiple sources (e.g. coreference resolution or verb interpretation) until it can be confidently resolved. Experiments show the first-ever results (43% recall and 91% precision) on SAT algebra word problems. We also apply EUCLID to the public Dolphin algebra question set, and improve the state-of-the-art F_1-score from 73.9% to 77.0%.

1 Introduction

Math word problems pose questions that are challenging for current question answering (QA) systems to solve. Consider the following question originating from a study guide for the Math SAT[1]:

> **Example 1:** Suppose $3x + y = 15$, where x is a positive integer. What is the difference between the largest possible value of y and the smallest possible value of x, assuming that y is also a positive integer?

The correct response is 11; however its relationship with the other numbers in the question (3 and 15) is oblique and not easily mapped to an operator tree or equation template. This encourages us to build a semantic parser that produces an explicit representation of what the question is asking, if we want to make quantitative progress on the question set. However, while it is not hard to formalize the semantics:

$$X \times Y = \{(x, y) \mid 3x + y = 15, x, y \in \mathbb{Z}^+\}$$
$$X = \{x \mid (x, y) \in X \times Y\}$$
$$Y = \{y \mid (x, y) \in X \times Y\}$$
$$\textbf{solve: } \max Y - \min X$$

it is not clear how to devise a compositional transformation from the original question to the formal semantics, since the meaning is dispersed throughout the discourse, such that neither the maximization nor the minimization can be locally derived from some subtree of the syntactic structure.

Moreover, SAT questions quickly reach the limits of preprocessing tools like anaphora resolution:

> **Example 2:** $\langle r, s, t\rangle$ In the sequence above, if each term after the first is x more than the previous term, what is the average of r, s, and t in terms of r and x?

Understanding this question requires a nuanced resolution of *each term after the first* to the subsequence $\langle s, t\rangle$, a coreference resolution beyond the grasp of the current state of the art.

Generally speaking, question discourse (with its complex cross-sentence semantics and anaphora) has not been a major focus of QA research. In this paper, we use math SAT questions to develop an approach to handling question discourse. Our parser uses an intermediate semantic language that allows complex semantics (like those of Example 1) to be compositionally constructed from a multi-sentence question passage (Section 5.1). By architecting our semantic parser as a cascade of nondeterministic tree transducers (Gécseg and Steinby, 1997), we can propagate uncertainty until it can be

[1] The Math SAT is a standardized exam administered to college-bound high school students in the United States.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 795–804
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

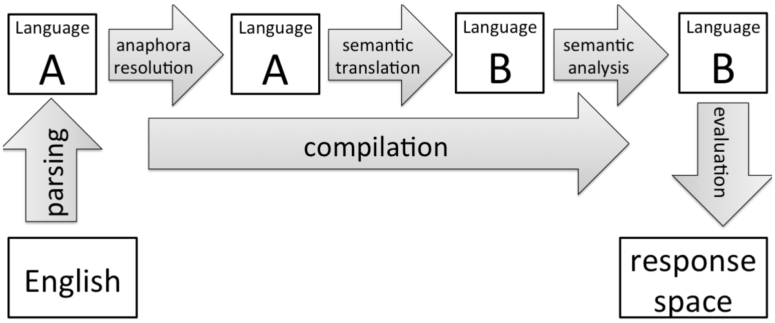


Figure 1: High-level view of EUCLID's architecture.

confidently resolved – sometimes as late as during program interpretation (Section 6). The integrated approach also allows us to handle novel classes of anaphoric phenomena by framing anaphora resolution as an operation on a parse forest decorated with implicits (Section 5.2).

Ultimately we produce an end-to-end system (called EUCLID) that achieves 43% recall and 91% precision on SAT closed-vocabulary algebra questions, a subset (described in more detail in the next section) that constitutes approximately 45% of a typical math SAT. We also achieve state-of-the-art results on the publicly released Dolphin question set (Shi et al., 2015), a set of more than 1500 algebra questions released by Microsoft Research. Finally, we provide a look at our early progress on extending the system to tackle the math SAT in its entirety.

2 Anatomy of a Math SAT

To assess our semantic parser, we compiled three question sets. Two question sets were created from sample SAT exams found in study guides (published by Kaplan and McGraw-Hill). We used the Kaplan set (12 exams, 648 total questions) for training/development and the McGraw-Hill set (13 exams[2], 686 total questions) for devtest. We reserved official practice exams (8 exams, 396 total questions) released by the College Board for final testing. We did not subselect questions from the exams, rather we used them in their entirety.[3] We encoded mathematical formatting using LATEX.

During the compilation of these questions, they were split into 4 broad categories:

1. **Algebra (closed vocabulary)** (e.g. Examples 1 and 2) **:** Algebra questions drawn from a limited mathematical vocabulary.
2. **Algebra (open vocabulary)** (e.g. "At a basketball tournament involving 8 teams, each team played 4 games with each of the other teams. How many games were played at this tournament?") **:** Algebra questions drawn from an open-ended vocabulary.
3. **Geometry:** Geometry questions, typically involving a diagram.
4. **Other** A catch-all for questions that do not fall neatly into the above categories.

In this paper, we focus our attention on closed-vocabulary algebra, which constitutes approximately 45% of the questions.

3 Related Work

Most of the recent work on math questions has focused on open-vocabulary algebra problems, also known as math story problems. Benchmark datasets include Alg514 (Kushman et al., 2014), AI2 (Hosseini et al., 2014), Illinois and Commoncore (Roy and Roth, 2015), and DRAW (Upadhyay and Chang, 2016). A common property of these datasets is that they have been curated such that any given question can be solved by a limited-depth operator tree (AI2, Illinois, Commoncore) or a limited set of equation templates (Alg514 and DRAW). Because of this, it is feasible to use discriminative approaches (Kushman et al., 2014; Hosseini et al., 2014; Roy and Roth, 2015; Zhou et al., 2015; Koncel-Kedziorski et al., 2015; Mitra and Baral, 2016) that extract the quantities, featurize the question, and then perform a weighted search over the space of instantiated operator trees or equation templates. However it is not clear how one can extend these discriminative techniques to

[2] 12 full exams + 1 PSAT

[3] One exception: we exclude the "comparison"-style questions (discontinued in 2005) from pre-2005 exams.

handle the complex semantics found in Examples 1 and 2.

Very recently, (Matsuzaki et al., 2017) published a paper about their semantic parsing approach to pre-university math problems (harvested from Japanese exams rather than the Math SAT). It is challenging to do a direct comparison, since they report results only on the Japanese-language exams. They report end-to-end system results of 11% recall and 50% precision.

(Shi et al., 2015) harvested a fairly diverse set of closed-vocabulary algebra problems (called Dolphin) from the web and provided the first results on that dataset. Here, we demonstrate how to handle the more complex discourse semantics and anaphoric phenomena found in Math SAT questions, and establish a new state-of-the-art result on the Dolphin benchmark.

4 System Overview

Figure 1 shows a high-level view of our QA system. We will give a general overview in this section, and then explore more advanced concepts and examples in the subsequent section.

4.1 Intermediate Languages

Our QA system has two basic languages that mediate the transformation from the question passage to the answer: a syntactic language $\mathcal{A}$ and a semantic language $\mathcal{B}$.

Syntactic language $\mathcal{A}$ has a constituent-style syntax convenient[4] for the tree transducers in our cascade. In Figure 2, we show an example. We have three basic node types: *clauses*, *entities* (these correspond to noun phrases), and *details* (these correspond to adjectival and adverbial modifiers). Each node has a table of *fields* (key-value pairs) that store child relationships and auxiliary information like tense and number. For brevity, this additional structure is omitted from Figure 2, but a more explicit visualization can be found in Figure 4 (top).

A program in semantic language $\mathcal{B}$ is a set of constraint declarations. For instance, the question from Figure 2 ("Let $m+3 < 15$. If m is a positive integer, what is the sum of all values of m?") compiles to the semantic program in Figure 3. When

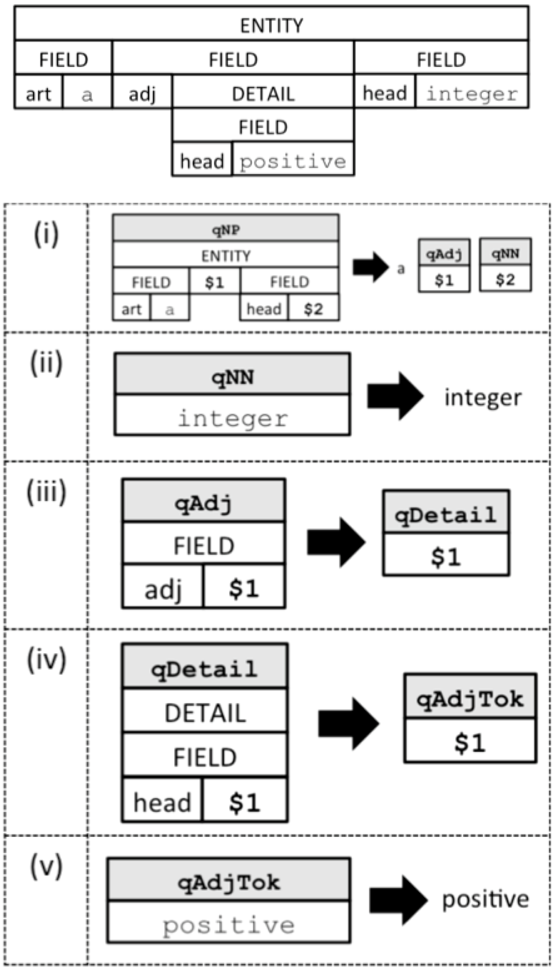


Figure 4: Example XTOPs transducer rules (bottom) used to derive a syntactic parse from the noun phrase "a positive integer" (via backward application of the transducer).

the form of the tree is unimportant, it will be convenient to use a more legible LISP-style format, e.g.

$(< (+\ m\ 3)\ 15)$
$(>\ m\ 0)$
$(\text{int}\ m)$
$(\text{proto}\ m\ M)$
$(=\ ?q\ (\text{sum}\ M))$

Every constraint in this program should be easily understandable, except for (proto m M), which loosely means that M is the set of all possible values of m. In Section 5.1, we discuss the proto directive in more detail.

[4] We experimented with adopting an existing syntax, like the Penn Treebank Syntax or the Stanford Dependency Syntax, but it turned out to be easier to develop the syntax in parallel with the needs of our system. Having said that, it is not intended to be wildly different from those formalisms.

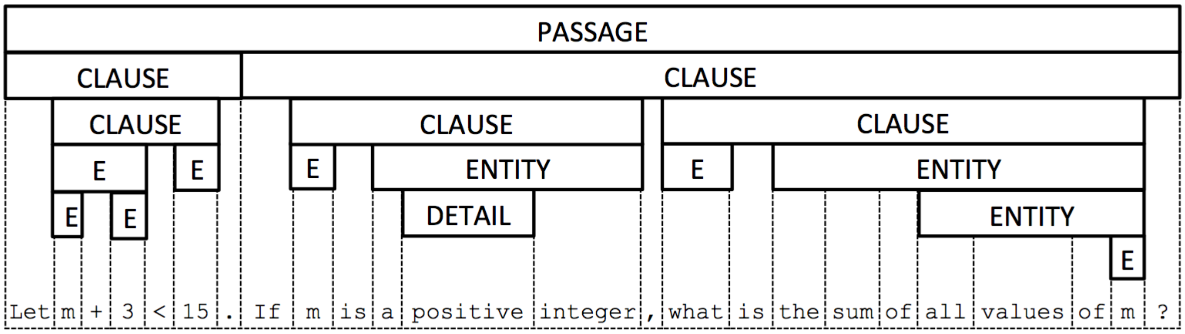

PASSAGE
CLAUSE
CLAUSE
CLAUSE
CLAUSE
CLAUSE
E
E
E
ENTITY
E
ENTITY
E
E
DETAIL
ENTITY
E
Let m + 3 < 15 . If m is a positive integer , what is the sum of all values of m ?

Figure 2: Example syntactic parse. For convenience, we show the correspondence of the nodes of our syntactic parse (top) to the original question passage (bottom). In the parse tree, "E" stands for "ENTITY".

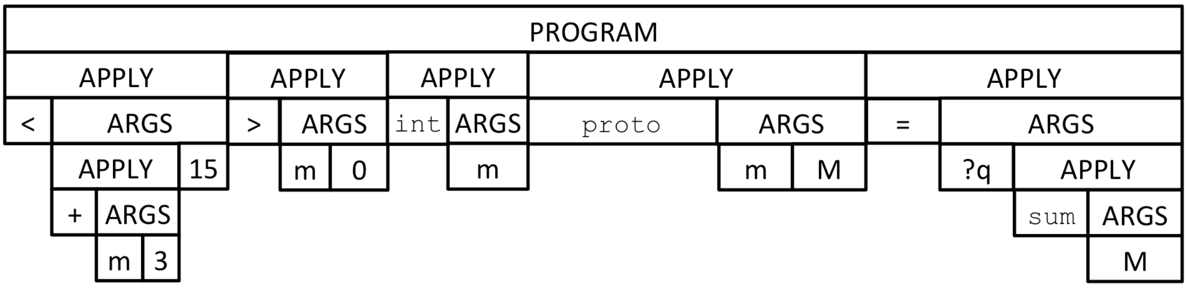


Figure 3: Example semantic program for the question "Let $m + 3 < 15$. If m is a positive integer, what is the sum of all values of m?"

4.2 Syntactic Parsing

The first stage of our QA system parses the question passage into language $\mathcal{A}$[5]. We implemented the parser as the backward application of an extended top-down tree-to-string (XTOPs) transducer[6].

We refer the reader to (Maletti et al., 2009) for a theoretical presentation of XTOPs, and instead give a brief intuitive presentation of the device. An XTOPs transducer defines a top-down transformation from a tree language to a string language, via a set of stateful rewrite rules. For instance, rules (i) through (v) of Figure 4 can generate the string "a positive integer" from the $\mathcal{A}$-tree pictured at the top of the figure, given start state **qNP**.

Given an XTOPs transducer M, we can parse string s through *backward application* of the transducer, i.e. compute the set of trees $M^{-1}(s)$ that could have generated string s from the start state. Efficient backward application of XTOPs transducers is supported by packages like Tiburon (May and Knight, 2006).

Our XTOPs transducer has approximately 140 states and 550 engineered rules (approximately 200 of these rules are used for parsing formal mathematics and a subset of LaTeX). Most lexical rules are automatically generated on-the-fly from WordNet (Miller, 1995).

4.3 Compilation

We then compile the parses of the question passage, by running them forward through a cascade of bottom-up tree transducers (Engelfriet, 1975). Again we refer the reader to the literature (Maletti, 2011, 2014) for a theoretical presentation of bottom-up tree transducers, and use Figure 5 to provide intuition about the device. A bottom-up tree transducer defines a transformation from a tree language to a (possibly different) tree language, via a set of stateful bottom-up rewrite rules.

In Figure 5, we show how this transformation works in the context of the semantic translation step, which uses a multi bottom-up transducer (MBOT) to map our syntactic language $\mathcal{A}$ into our

[5] Recall: language $\mathcal{A}$ is the syntactic language described in the previous section. An example is shown in Figure 2.

[6] We chose to implement the parsing step by engineering a transducer rather than using an off-the-shelf statistical parser. While we tried to retrofit a parser – e.g. as done by (Seo et al., 2015) – to serve our needs, it turned out to be somewhat more robust (and relatively simple) to engineer our own.

semantic language $\mathcal{B}$. There is a single state (indicated by a gray shaded rectangle) that has two children: (i) a *return value*, and (ii) a set of *side-effect* statements.

The first rule application transforms "all values of m" into a *return value* of M (a new variable introduced to indicate the set of all values of variable m) and a *side-effect* (proto m M), indicating that M equals the set of all possible values of m. The second rule application transforms "the sum of M" into a *return value* of (sum M), and propagates upward the accumulated side-effects.

We implemented all three compilation steps from Figure 1 (anaphora resolution, semantic translation, and semantic analysis) as the forward application of a bottom-up tree transducer. Anaphora resolution resolves any nodes that refer to other nodes in the tree. Semantic translation translates syntactic language $\mathcal{A}$ into semantic language $\mathcal{B}$. Semantic analysis type-checks the trees for internal consistency.

4.4 Interpretation

Finally, each derived $\mathcal{B}$-tree is sent to an evaluator to obtain an answer. Our main evaluator is a wrapped version of Z3 (de Moura and Bjorner, 2008), a widely used Satisfiability Modulo Theories (SMT) solver. If it does not find an answer, we fall back to a numeric optimization solver similar to one used by (Seo et al., 2015).

5 Spotlights

Having provided a bird's eye view in the last section, we now spotlight some key details of our QA system.

5.1 Spotlight: Complex Aggregations

A core challenge of semantic parsing is how best to read complex semantic phenomena from a syntactic representation. Two such phenomena are superlatives and counting. GeoQuery (Zelle and Mooney, 1996) has examples[7] of these, as does[8] WebQuestions (Berant et al., 2013). Unfortunately, it is not clear how existing strategies for dealing with aggregative constructs (e.g. (Liang et al., 2011)) can be extended to the more complex multi-sentence questions found on the SATs. For instance, the basic semantics of Example 1 (enumerated in Section 1) is dispersed throughout the question passage, such that neither the maximization nor the minimization can be locally derived from some subtree of the dependency structure.

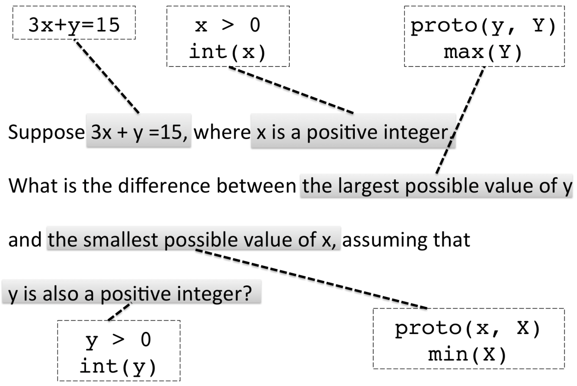

3x+y=15
x > 0
int(x)
proto(y, Y)
max(Y)
Suppose 3x + y =15, where x is a positive integer.
What is the difference between the largest possible value of y
and the smallest possible value of x, assuming that
y is also a positive integer?
y > 0
int(y)
proto(x, X)
min(X)

Figure 6: Understanding complex aggregations by decomposing them into order-independent atoms.

To deal with this challenge, we designed our semantic language $\mathcal{B}$ to decompose the semantics of aggregative constructs into order-independent atoms. Consider the following restatement of the semantics of Example 1:

$$\begin{gathered}\mathsf{proto}(\dot{x}, X)\\ \mathsf{proto}(\dot{y}, Y)\\ 3\dot{x} + \dot{y} = 15\\ \dot{x} > 0\\ \dot{y} > 0\\ \dot{x} \in \mathbb{Z}\\ \dot{y} \in \mathbb{Z}\\ \textbf{solve:}\ \max Y - \min X\end{gathered}$$

where $\mathsf{proto}(\dot{z}, Z)$ designates that a variable $\dot{z}$ should be treated as the prototype variable of a statement in set-builder notation, i.e. $Z = \{\dot{z} \mid ...\}$. We treat any other statement featuring prototype variable $\dot{z}$ as a constraint appearing on the right side of the set-builder statement. If there are multiple prototype statements, they are grouped into a single set-builder statement (as occurs with $\dot{x}$ and $\dot{y}$ in our example).

The power of this decomposition is that it can be reconstructed piecemeal from an arbitrarily complex passage. The atomic statements can be interpreted locally in an arbitrary order, as in Figure 6, then synthesized into set-builder notation during evaluation.

[7] e.g. "What is the capital of the state that borders the most states?"

[8] e.g. "How many pets did John F. Kennedy own?"

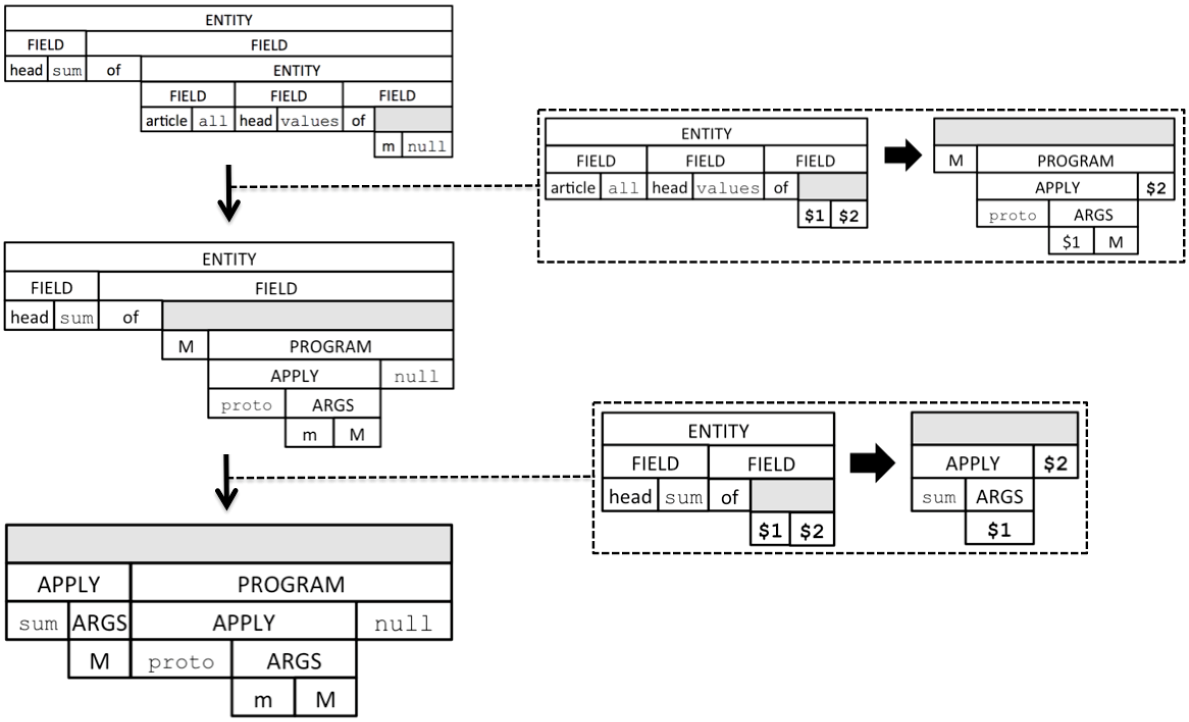


Figure 5: Example semantic translation using an MBOT.

5.2 Spotlight: Complex Anaphoric Phenomena

The anaphora resolution task (Ge et al., 1998) is typically defined at the lexical level. For instance, in the sentence "c is equal to its square," a traditional evaluation like the CoNLL-2011 Shared Task (Pradhan et al., 2011) would ask whether the string "its" is aligned to the string "c". These evaluations also assume that both the reference and the referent (a.k.a. antecedent) are contiguous strings in the text.

Math SAT problems exhibit a host of new challenges that fall outside traditionally studied definitions of anaphora resolution:

- **One-to-many coreference**[9] *(One integer is 5 more than another. What is the sum of the numbers?)*: "The numbers" refers to two discontiguous strings: "one integer" and "another".

- **Implicit set reference** *(Two numbers sum to 5. If the first is 2, what is the second?)*: "The second" implies a latent set that needs to be resolved (to "two numbers") in order to understand the sentence. This phenomenon is shown in Figure 7.

- **Implicit clausal reference** *(If 7 is divided by 3, what is the remainder?)*: "The remainder" implies a latent clause that needs to be resolved (to "7 is divided by 3") in order to understand the sentence.

We address this broader class of anaphora by a two-pass process:

1. First, we introduce implicit sets and clauses when appropriate. For instance, implicit sets are introduced for superlative and ordinal constructions, while implicit clauses are introduced for functional nouns like "remainder." In Figure 7, these implicits are depicted as bracketed phrases (i.e. [of a set]).

2. Anaphora resolution then proceeds as a bottom-up tree-labeling process, shown in Figure 7. For each subtree, a resolution function ρ partially maps subtree entities to subtree nodes. Note that ancestors can overwrite the resolutions of their descendants. This occurs in the second example of Figure 7, where the implicit set E7 is initially resolved to implicit set E4, but is later resolved to the entity E1 ("two numbers") once it comes into scope.

In our initial system, the resolution function ρ was

[9] A recent ACL paper (Vala et al., 2016) has provided a preliminary treatment of this phenomenon.

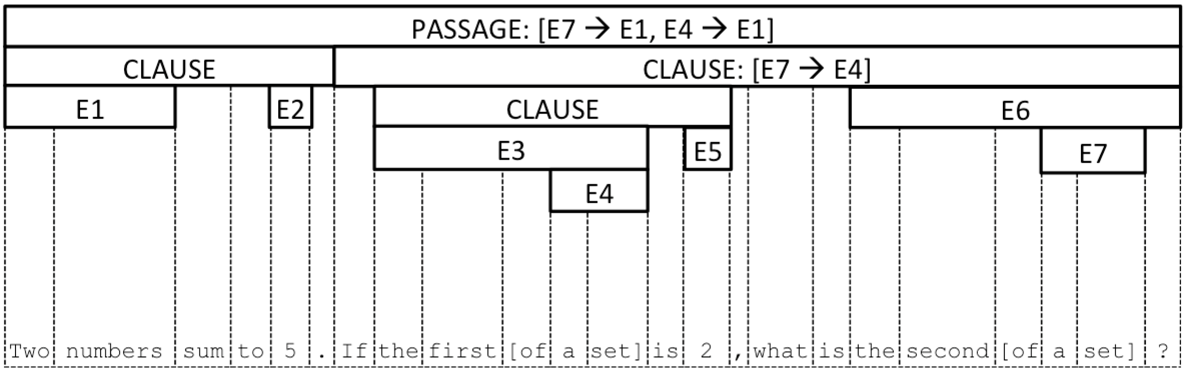

PASSAGE: [E7 → E1, E4 → E1]
CLAUSE
CLAUSE: [E7 → E4]
E1
E2
CLAUSE
E6
E3
E5
E7
E4
Two numbers sum to 5 . If the first [of a set] is 2 , what is the second [of a set] ?

Figure 7: Bottom-up anaphora resolution in our QA system. For convenience, we show the correspondence of the nodes of our syntactic parse (top) to the original question passage (bottom). In the parse tree, "E" is an abbreviation for ENTITY.

engineered heuristically. We later replaced this with a learned version (by using our system to generate training data). Due to space considerations, details are omitted.

6 Results with the Unweighted Nondeterministic Cascade

In the basic cascade from Section 4, the number of trees passed from module to module can expand, but it can also contract (for instance, in the semantic translation step, there can be multiple ways of translating a parse, or none at all). This allows the QA system to disambiguate question passages by eliminating parses for which there is no consistent semantics. On the subset of the Kaplan questions for which at least one parse exists, the average number of trees after the parsing step is 7.5. The average number of trees after the semantic analysis step goes down to only 2.4. At that point, obviously we need to choose some priority in which to feed these finalized programs to the evaluation module. Using a simple heuristic (process smaller programs first), we obtain 70.2% recall and 95.8% precision on the Kaplan closed-vocabulary algebra questions[10].

This high precision can be partially attributed to the fact that most SAT questions are multiple-choice (thus we can sequentially evaluate the finalized programs until we find a viable answer). We do not have that luxury on the Dolphin dataset, a set of direct-answer algebra questions curated by Microsoft Research (split into a development set of 374 questions and a test set of 1504 questions). On the subset of the development questions for which at least one parse exists (90.3% of the questions), the average number of trees after the parsing step is 4.3. The average number of trees after the semantic analysis step goes down to 1.5. Our basic system obtains 66.3% recall on the development questions. Naturally the precision is not as high as on the multiple choice questions, but surprisingly we still obtain 85.5% precision, even with an unweighted cascade.

7 Introducing a Parse Ranker

Most of this precision loss is due to legitimate parse ambiguity that cannot be resolved through semantic interpretability alone. Rather, the disambiguation requires some additional pragmatic convention. Consider the example: "When the reciprocal of three times a number is subtracted from the reciprocal of the number, the result is one sixth. Find the number." By interpreting "the reciprocal of three" as $\frac{1}{3}$, the meaning of this question becomes "When $\frac{1}{3}$ times a number is subtracted from the reciprocal of the number, the result is one sixth. Find the number." This is not however the most human-intuitive interpretation of the question. Somehow the system must identify the pragmatic cues that cause humans to disprefer this interpretation.

To identify these cues, we insert a *parse ranking module* between the parsing module and the anaphora resolution module (see Figure 1 for a reminder of the system components). The goal of the parse ranker is to associate a lower cost to "more intuitive" interpretations when there are multiple plausible syntactic interpretations. The

[10] Recall and precision numbers are computed over the entire set of questions, regardless of whether they have a valid parse.

	recall	prec.	F1
Kaplan (non-blind)	70.2	95.8	81.0
McGraw-Hill (blind)	41.0	91.8	56.7
Official (blind)	43.1	90.8	58.5

Table 1: Results on the closed-vocabulary algebra subsets of our Math SAT question sets.

	EUCLID			(Shi et al., 2015)		
	rec	**prec**	**F1**	**rec**	**prec**	**F1**
dev	78.1	97.0	86.5	-	-	-
test	65.1	94.1	77.0	60.3	95.4	73.9

Table 2: Results on the Dolphin question sets. The increase in recall is statistically significant with a P-value < 0.01.

rest of the cascade propagates these costs. Similar to existing work, e.g. (Charniak and Johnson, 2005), we implement the cost function as an L_1-regularized logistic regression model.

Adding the trained parse ranker module improves performance on the Dolphin development set to 75.7% recall and 97.3% precision (from 66.3% recall and 85.5% precision).

8 Final Results

Results from our final system are shown in Table 1 (for the closed-vocabulary algebra subsets of our math SAT question sets) and Table 2 (for the Dolphin question sets). EUCLID generalizes reasonably well to the blind SAT questions, achieving approximately 60% of the system's recall on the training questions, at a precision of approximately 91%. To give a sense of the extent of the generalization from training to test, Table 3 offers a couple of correctly answered questions from the blind[11] McGraw-Hill set, plus their closest analog in the training questions (by edit distance). The performance on the blind test sets (including all questions, not just closed-vocabulary algebra) corresponds to an SAT score of approximately 350 (out of 800). A random-guessing baseline has an expected score of 200.

Table 4 provides a failure analysis on the McGraw-Hill data, categorizing a sample of 50 questions. Half of the questions failed to have a valid parse. Roughly a quarter of the questions had at least one valid parse, but none of these resulted in a semantic program. 18% of the questions compiled into at least one semantic program, but none of these produced an answer when fed to the interpreter. 8% of the questions compiled into a semantic form that produced an incorrect answer.

development (blind)	**training**
Set M consists of the consecutive integers from -15 to y, inclusive. If the sum of all of the integers in set M is 70, how many numbers are in the set?	If the sum of the consecutive integers from -15 to x, inclusive, is 51, what is the value of x?
If a, b, and c are positive even integers such that $a < b < c$ and $a + b + c = 60$, then the greatest possible value of c is	If x and y are different positive integers and $3x + y = 17$, the difference between the largest possible value of y and the smallest possible value of x is

Table 3: Some correctly answered questions on the blind McGraw-Hill set, and their closest parallel (by edit distance) in the training set (Kaplan).

development (blind)	**training**
failed to parse	50%
failed to map parse into a semantic program	24%
failed to produce an answer from any semantic program	18%
produced an incorrect answer	8%

Table 4: Error analysis on the blind McGraw-Hill set, surveying the first point of failure for a sample of 50 incorrectly answered questions.

Besides the Math SAT datasets, EUCLID also has state-of-the-art performance on the public Dolphin question set, achieving an absolute recall improvement[12] of nearly 5% with a small loss in precision. This raises the state-of-the-art F_1-score on this data set from 73.9% to 77.0%.

[11] Apart from harvesting a sample of correctly answered questions for this analysis, the McGraw-Hill set was kept completely blind. The official set was left completely untouched.

[12] This improvement is statistically significant with a P-value < 0.01.

genre	dataset	blind?	recall	precision	F1-score
closed algebra	Kaplan	no	70.2	95.8	81.0
	McGraw-Hill	yes	41.0	91.8	56.7
	official	yes	43.1	90.8	58.5
geometry	Kaplan	no	11.7	95.0	20.8
	McGraw-Hill	yes	5.6	76.9	10.4
open algebra	hosseini-ma2	no	34.7	57.5	43.3
	hosseini-ma1	yes	29.9	45.4	36.1

Table 5: Snapshot of early progress across several subgenres of the Math SAT. In our early stages, we are hillclimbing on the hosseini datasets from (Hosseini et al., 2014), which are simpler than the open-vocabulary algebra questions from the Math SAT.

9 Towards a Broad-Coverage SAT solver

This paper reports on the first steps of a longer-term initiative to build a unified system that can pass the math SAT. We have made some preliminary forays into extending the system to handle the more complex subdomains described in Section 2, namely open-vocabulary algebra and geometry. Key research challenges presented by these new domains are:

- **Mapping into richer semantic languages:** The math story problems of open-vocabulary algebra require languages that reason about state change and can introduce assumptions not explicitly represented in the text.
- **Robustly synthesizing diagram and text information:** For geometry questions, we are building on key early work in this area performed by (Seo et al., 2014, 2015).
- **Extending the system in a scalable way:** Writing new transducer rules for each new domain is not a sustainable way to extend our system. We are exploring how to use natural language to "program" our system, e.g. by automatically inducing transducer rules for paraphrase text.

A snapshot of our current progress is shown in Table 5.

10 Discussion

In the process of creating a QA system for math SAT questions, this project has yielded several general strategies for beyond-sentential semantic parsing. For instance:

- One can modularize the parser as a cascade of tree transducers, allowing uncertainty about anaphora resolution and lexical interpretation to be propagated until it can be confidently resolved, sometimes as late as program interpretation (see Section 6).
- One can atomize complex semantic phenomena (e.g. aggregrative constructs) into small order-independent pieces. This allows a simpler transformation from a syntactic form, because these atoms can be locally recognized, incrementally composed, and globally reconstituted into a structured semantics (see Section 5.1).
- One can reframe bread-and-butter NLP tasks (e.g. anaphora resolution) to fit better within (and take advantage of) the context of the transducer cascade (see Section 5.2)

An important focus of this paper has been issues of representation, namely how to develop intermediate structured languages that facilitate the automatic transformation of question discourse into a response. Because we can use the resulting QA system to generate gold intermediate trees for any correctly answered question in our dataset, one way to view this work is as a data annotation project. One distinguishing advantage is that our intermediate languages come with a "proof of usefulness." They are not designed based on speculative utility – rather, they have already proven useful in the context of a functioning QA system.

Acknowledgments

The authors would like to thank Luke Zettlemoyer, Jayant Krishnamurthy, Oren Etzioni, and the anonymous reviewers for valuable feedback on earlier drafts of the paper.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Joost Engelfriet. 1975. Bottom-up and top-down tree transformationsa comparison. *Mathematical systems theory*, 9(2):198–231.

Niyu Ge, John Hale, and Eugene Charniak. 1998. A statistical approach to anaphora resolution.

Ferenc Gécseg and Magnus Steinby. 1997. Tree languages. In *Handbook of formal languages*, pages 1–68. Springer.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *TACL*, 3:585–597.

Nate Kushman, Luke S. Zettlemoyer, Regina Barzilay, and Yoav Artzi. 2014. Learning to automatically solve algebra word problems. In *ACL*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *ACL*.

Andreas Maletti. 2011. How to train your multi bottom-up tree transducer. In *ACL*.

Andreas Maletti. 2014. The power of regularity-preserving multi bottom-up tree transducers. In *CIAA*.

Andreas Maletti, Jonathan Graehl, Mark Hopkins, and Kevin Knight. 2009. The power of extended top-down tree transducers. *SIAM J. Comput.*, 39:410–430.

Takuya Matsuzaki, Takumi Ito, Hidenao Iwane, Hirokazu Anai, and Noriko H. Arai. 2017. Semantic parsing of pre-university math problems. In *ACL*.

Jonathan May and Kevin Knight. 2006. Tiburon: A weighted tree automata toolkit. In *CIAA*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Arindam Mitra and Chitta Baral. 2016. Addressing a question answering challenge by combining statistical methods with inductive rule learning and reasoning. In *AAAI*.

Leonardo Mendona de Moura and Nikolaj Bjorner. 2008. Z3: An efficient smt solver. In *TACAS*.

Sameer Pradhan, Lance A. Ramshaw, Mitchell P. Marcus, Martha Palmer, Ralph M. Weischedel, and Nianwen Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *CoNLL Shared Task*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *EMNLP*.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. 2014. Diagram understanding in geometry questions. In *AAAI*.

Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *EMNLP*.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *EMNLP*.

Shyam Upadhyay and Ming-Wei Chang. 2016. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. *CoRR*, abs/1609.07197.

Hardik Vala, Andrew Piper, and Derek Ruths. 2016. The more antecedents, the merrier: Resolving multi-antecedent anaphors. In *ACL*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI, Vol. 2*.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *EMNLP*.

Learning Fine-Grained Expressions to Solve Math Word Problems

Danqing Huang[1*], Shuming Shi[2], Jian Yin[1], and Chin-Yew Lin[3]
{huangdq2@mail2,issjyin@mail}.sysu.edu.cn
shumingshi@tencent.com
cyl@microsoft.com
[1] Guangdong Key Laboratory of Big Data Analysis and Processing, Sun Yat-sen University
[2]Tencent AI Lab [3] Microsoft Research

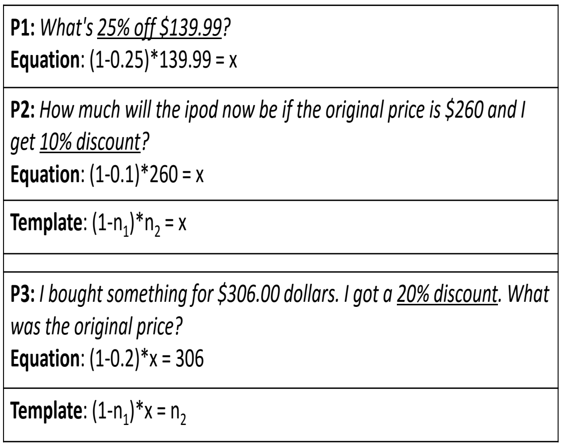
Abstract

This paper presents a novel template-based method to solve math word problems. This method learns the mappings between math concept phrases in math word problems and their math expressions from training data. For each equation template, we automatically construct a rich template sketch by aggregating information from various problems with the same template. Our approach is implemented in a two-stage system. It first retrieves a few relevant equation system templates and aligns numbers in math word problems to those templates for candidate equation generation. It then does a fine-grained inference to obtain the final answer. Experiment results show that our method achieves an accuracy of 28.4% on the linear Dolphin18K benchmark, which is 10% (54% relative) higher than previous state-of-the-art systems while achieving an accuracy increase of 12% (59% relative) on the TS6 benchmark subset.

1 Introduction

The research topic of automatically solving math word problems dates back to the 1960s (Bobrow, 1964a,b; Charniak, 1968). Recently many systems have been proposed to these types of problems (Kushman et al., 2014; Hosseini et al., 2014; Koncel-Kedziorski et al., 2015; Zhou et al., 2015; Roy and Roth, 2015; Shi et al., 2015; Upadhyay et al., 2016; Mitra and Baral, 2016). On a recent evaluation conducted by Huang et al. (2016), current state-of-the-art systems only achieved an 18.3% accuracy on their published dataset Dolphin18K. Their results indicate that math word problem solving is a very challenging task.

To solve a math word problem, a system needs to understand natural language text to extract information from the problem as local context. Also, it should provide an external knowledge base, including commonsense knowledge (e.g. *"a chicken has two legs"*) and mathematical knowledge (e.g. *"the perimeter of a rectangle = 2 * length + 2 * width"*). The system can then perform reasoning based on the above two resources to generate an answer.



P1: *What's 25% off $139.99?*
Equation: (1-0.25)*139.99 = x

P2: *How much will the ipod now be if the original price is $260 and I get 10% discount?*
Equation: (1-0.1)*260 = x

Template: $(1-n_1)*n_2 = x$

P3: *I bought something for $306.00 dollars. I got a 20% discount. What was the original price?*
Equation: (1-0.2)*x = 306

Template: $(1-n_1)*x = n_2$

Figure 1: Math Word Problem Examples.

In this paper, we focus on the acquisition of mathematical knowledge, or deriving math concepts from natural language. Consider the first two problems $P1$ and $P2$ in Figure 1. The math concept in the problems tells you to take away a percentage from one and get the resulting percentage of a total. Using mathematical language, it can be formulated as $(1-n_1)*n_2$, where n_1, n_2 are quantities. In this example, we can derive the concept of subtraction from the text "*[NUM] % off*" and "*[NUM] % discount*".

[*] Work done while this author was an intern at Microsoft Research.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 805–814
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Acquisition of mathematical knowledge is non-trivial. Initial statistical approaches (Hosseini et al., 2014; Roy and Roth, 2015; Koncel-Kedziorski et al., 2015) derive math concepts based on observations from their dataset of specific types of problems, e.g. problems with one single equation. For example, Hosseini et al. (2014) assumes verbs and only verbs embed math concepts and map them to addition/subtraction. Roy and Roth (2015); Koncel-Kedziorski et al. (2015) assume there is only one unknown variable in the problem and cannot derive math concepts involving constants or more than one unknown variables, such as "*the product of two unknown numbers*".

Template-based approaches (Kushman et al., 2014; Zhou et al., 2015; Upadhyay et al., 2016), on the other hand, leverage the built-in composition structure of equation system templates to formulate all types of math concepts seen in training data, such as $(1 - n_1) * n_2 = x$ in Figure 1. However, they suffer from two major shortcomings. First, the math concepts they learned, which is expressed as an entire template, fails to capture a lot of useful information with sparse training instances. We argue that it would be more expressive if the math concept is learned in a finer granularity. Second, their learning processes rely heavily on lexical and syntactic features, such as the dependency path between two slots in a template. When applied to a large-scale dataset, they create a huge and sparse feature space and it is unclear how these template-related features would contribute.

To alleviate the sparseness problem of math concept learning and better utilize templates, we propose a novel approach to capture rich information contained in templates, including textual expressions that imply math concepts. We parse the template into a tree structure and define "template fragment" as any subtree with at least one operator and two operands. We learn fine-grained mappings between textual expressions and template fragments, based on longest common substring. For example, given the three problems in Figure 1, we can map "*[NUM] %* off" and "*[NUM] %* discount" to $1 - n_1$, and "*[NUM] %* off *[NUM]*" to $(1 - n_1) * n_2 = x$. In this way, we can decompose the templates and learn math concepts in a finer grain. Furthermore, we observe that problems of the same template share some common properties. By aggregating problems of the same template and capturing these properties, we automatically construct a sketch for each template in the training data.

Our approach is implemented in a two-stage system. We first retrieve a few relevant templates in the training data. This narrows our search space to focus only on those templates that are likely to be relevant. Then we align numbers in the problem to those few returned templates, and do fine-grained inference to obtain the final answer. We show that the textural expressions and template sketch we propose are effective for both stages. In addition, our system significantly reduces the hypothesis space of candidate equations compared to previous systems, which benefits the learning process and inference at scale.

We evaluate our system on the benchmark dataset provided by Huang et al. (2016). Experiments show that our system outperforms two state-of-the-art baselines with a more than 10% absolute (54% relative) accuracy increase in the linear benchmark and a more than 20% absolute (71% relative) accuracy increase for the dataset with a template size greater than or equal to 6.

In the remaining parts of this paper, we introduce related work in Section 2, describe template sketch and textual expression learning in Section 3, present our two-stage system in Section 4, summarize experiment setup and results in Section 5, and conclude this paper in Section 6.

2 Related Work

Automatic math word problem solving methods (Bobrow, 1964a,b; Charniak, 1968, 1969; Briars and Larkin, 1984; Fletcher, 1985; Dellarosa, 1986; Bakman, 2007; Yuhui et al., 2010) developed before 2008 are mostly rule-based. They accept limited well-format input sentences and map them into certain structures by pattern matching. They usually focus on problems with simple math operations such as addition or subtraction. Please see Mukherjee and Garain (2008) for a summary.

In recent years, symbolic and statistical methods have been explored by various researchers. In the symbolic approach, systems transform math word problems to structured representations. Bakman (2007) maps math problems to predefined schema with a table of textual formulas and changing verbs. Liguda and Pfeiffer (2012) uses augmented semantic networks to represent math problems. Shi et al. (2015) parses math problems to their pre-defined semantic language. However, these methods are only effective in their designated math problem categories and are not scalable to other categories. For example, the method used by Shi et al. (2015) works extremely well for solving number word problems but not others.

In the statistical machine learning approach, Hosseini et al. (2014) solves addition and subtraction problems by extracting quantities as states and derive math concepts from verbs in the training data. Kushman et al. (2014) and Zhou et al. (2015) generalize equations attached to problems with variable slots and number slots. They learn a probabilistic model for finding the best solution equation. Upadhyay et al. (2016) follows their approach and leverage math word problems without equation annotation as external resources. Seo et al. (2015) solves a set of SAT geometry questions with text and diagram provided. Koncel-Kedziorski et al. (2015) and Roy and Roth (2015) target math problems that can be solved by one single linear equation. They map quantities and words to candidate equation trees and select the best tree using a statistical learning model. Mitra and Baral (2016) considers addition and subtraction problems in three basic problem types: “Change”, “Part Whole” and “Comparison”. They manually design different features for each type, which is difficult to expand to more types.

In summary, previous methods can achieve high accuracy in limited math problem categories, (i.e. (Kushman et al., 2014; Shi et al., 2015)), but do not scale or perform well in datasets containing various math problem types as in Huang et al. (2016), as their designed features are becoming sparse. Their process of acquiring mathematical knowledge is either sparse or based on certain assumptions of specific problem types. To alleviate this problem, we introduce our template sketch construction and fine-grained expressions learning in the next section.

3 Template Sketch Construction

A template sketch contains template information. We define three categories of information for the sketch shown in this section. Next we describe how we construct a template sketch, via aggregation of rich information from training problems. We group problems of the same template in training set as one cluster and collect information. See Figure 2 for the outline of our template sketch construction.

3.1 Definition

Template: It is first introduced in Kushman et al. (2014). It is a unique form of an equation system. For example, given an equation system as follows:

$$\begin{gathered} 2 \cdot x_1 + 4 \cdot x_2 = 34 \\ x_1 + 4 = x_2 \end{gathered}$$

This equation system is a solution for a specific math word problem. We replace the numbers with four number slots $\{n_1, n_2, n_3, n_4\}$ and generalize the equations to the following template:

$$\begin{gathered} n_1 \cdot x_1 + n_2 \cdot x_2 = n_3 \\ x_1 + n_4 = x_2 \end{gathered}$$

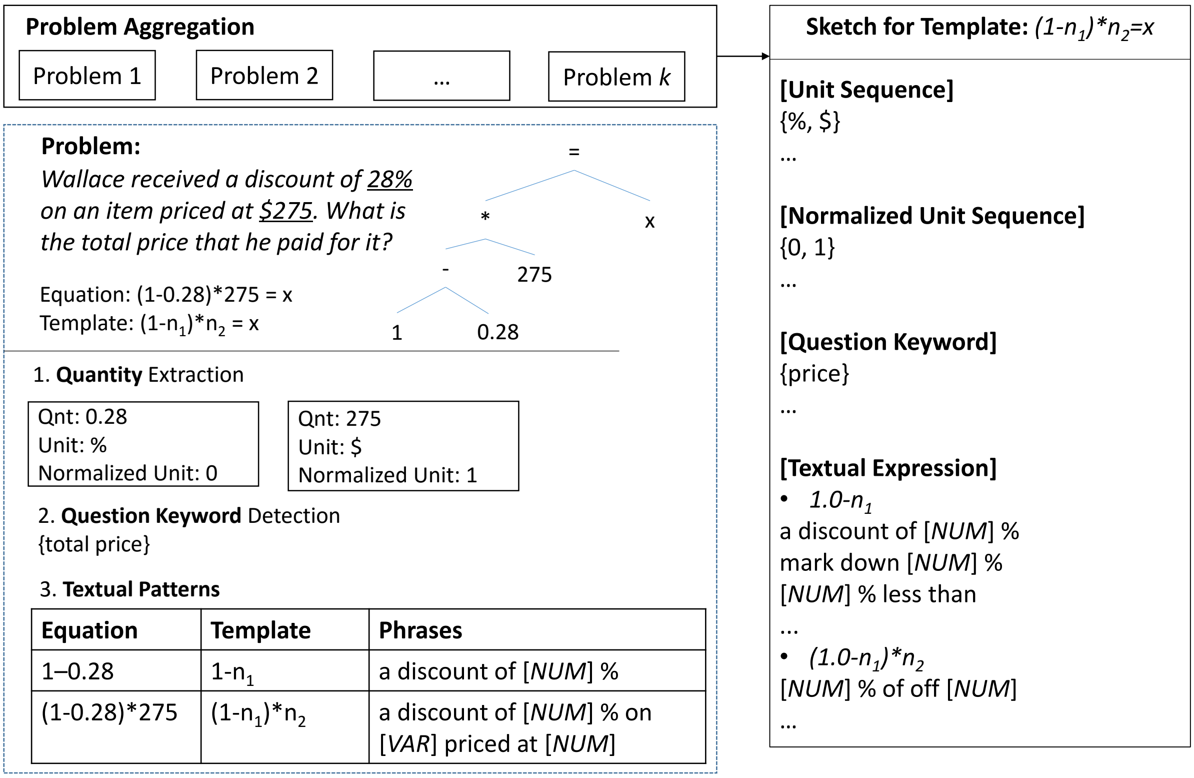
Alignment: We align numbers in the math problem with the number slots of a template. For the first math problem in Figure 1 with its corresponding template $(1 - n_1) * n_2 = x$, there are two numbers 0.25 and 139.99 to align with two number slots n_1 and n_2, which results in two different alignments.

Kushman et al. (2014) aligns nouns to variable slots $\{x_1, x_2, ...\}$ which leads to a huge hypothesis space and does not perform as well as the number slot alignment only method proposed later by (Zhou et al., 2015). Therefore, we only consider number slot alignment in this paper.

3.2 Textual Expressions

For template fragments, there are usually some textual expressions. For example, “n_1 % off” and “n_1 % discount” are both mapped to the template fragment $1 - n_1$.

We employ a statistical framework to automatically mine textual expressions for template fragments from a training dataset. First we parse the equation to a hierarchical tree. In a bottom-up approach, we obtain each possible subtree as a template fragment t_k, which associates with at least one number slot. For each t_k, we use the numbers to anchor the number-related phrases in the problem, replace numbers with“*[NUM]*” and noun phrases with “*[VAR]*”, and cluster the phrases $P = \{p_1, p_2, \cdots\}$ with the same t_k across all data given all training problems. Then we compute the longest common substring lcs^k_{ij} between pairs p_i and p_j and calculate tf-idf score of lcs^k_{ij}. We keep the lcs^k_{ij} with scores above certain empirically determined threshold as the textual expressions.



Equation	Template	Phrases
1–0.28	$1\text{-}n_1$	a discount of [*NUM*] %
(1-0.28)*275	$(1\text{-}n_1)*n_2$	a discount of [*NUM*] % on [*VAR*] priced at [*NUM*]

Figure 2: Template Sketch Construction.

3.3 Slot Type

Number slots in templates have their own type of constraints. For example, in the template $(1-n_1)*n_2 = x$, usually n_1 represents a percentage quantity and n_2 is the quantity of an object.

We model slot types with quantity units, and find the direct governing noun phrase as its 'owner'. For the problem in Figure 2, we extract quantity unit sequence as $\{\%, \$\}$, normalized unit sequence as $\{0, 1\}$ (because % and $ are of different quantity types), and quantity owners as {discount, item}. The slot type information provides important clues to choose the correct template and alignment.

3.4 Question Keyword

Question keyword decides which template we use. Given the following problem setting: "A rectangle has a width of 5cm and a length of 10cm.", we can ask either Q1:"What is the *area* of the rectangle?" or Q2: "What is the *difference* between width and length?". The question keywords *area* and *difference* help our system to decide if is should apply template $n_1 * n_2 = x$ for Q1 and apply template $n_1 - n_2 = x$ for Q2.

We first detect the question sentence (containing keywords "what","how","figure out"...). Then we extract the question keyword on the dependency tree with simple rules that we observed in the dev set (e.g. retrieving nouns with "$attr - nsubj$" dependency relation with keyword "what"). Please note that we favor recall over precision of our detected question keywords since they are used as features instead of hard constrains on template decision. Simple rule-based extraction can already satisfy our need for detecting question keywords in math problems.

4 Two-Stage System

In this section, we describe our two-stage system for solving math problems, including template retrieval and alignment ranking. We show how to apply textual expressions and template sketch to our system.

4.1 Template Retrieval

We use an efficient retrieval module to first narrow our search space and focus only on templates that are likely to be relevant. Let χ denote the set of test problems, and $T = \{t_1, t_2, \ldots, t_j\}$ as the template set in the training data. For each test problem x_i, our goal is to select the correct template

t_j. We define the conditional probability of selecting a template given a problem as follows:

$$p(t_j|x_i;\nu_t) = \frac{\exp(\nu_t \cdot f(x_i, t_j))}{\sum_{t'_j \in T} \exp(\nu_t \cdot f(x_i, t'_j))}$$

where ν_t is the model parameter and $f(x_i, t_j)$ is the feature vector. We apply the Ranking SVM (Herbrich et al., 2000) to minimize a regularized margin-based pairwise loss. We then have the following objective function:

$$\frac{1}{2}\|\nu_t\|^2 + C\sum_i l(\nu_t^T f(x_i, t_j)^+ - \nu_t^T f(x_i, t_l)^-)$$

where superscript "+" indicates the correct instance and "-" indicates the false ones. We use the loss function $l(t) = max(0, 1-t)^2$.

To construct the vector $f(x_i, t_j)$ for template t_j, we use the three categories in the template sketch shown in Table 1. Let $Q(t_j)$ represent the cluster of training problems with template t_j.

Textual Features
Contains textual expressions in each template fragments?
Average Word Overlap with $Q(t_j)$
Max Word Overlap with $Q(t_j)$
Quantity Features
Unit sequence in $Q(t_j)$
Normalized unit sequence in $Q(t_j)$
Question Features
Is Question keyword in $Q(t_j)$

Table 1: Features for template retrieval.

At the phrase level, as we have mined different expressions in 3.2 for slots in templates, we can extract the phrases related to each number or number pair in a test problem and match them with expressions. For example, given a test problem to match template $(1 - n_1) * n_2 = x$ in Figure 2, we have two groups of patterns to match, corresponding to $1.0 - n_1$ and $(1.0 - n_1) * n_2$ respectively.

Quantity types in a problem are important. We use the unit type sequences and normalized unit type sequence for describing number slot types in a template. In addition, if a number unit type cannot differentiate each number slot, we will make use of number "owner" as defined in subsection 3.3. For example, in the sentence "The width is 3cm and the length is 5cm", we extract two quantities with unit type sequence {cm, cm}; and owner {width, length}.

In addition, we consider question keywords for templates. For example, if the question keyword is "difference", then $x + n_1 = n_2$ will have a higher probability of being selected than $x = n_1 * n_2$.

We observe that in some cases, one word difference can lead to two different templates. To consider cases in which some templates are very similar (e.g. $x + n_1 = n_2$ and $n_1 + n_2 = x$, part/whole unknown), we retrieve the top ranked N (N=3) templates as candidates for alignment in the next stage.

4.2 Alignment Ranking

For each top N templates from the previous stage, we generate possible alignments $A = \{a_1, a_2, \ldots, a_m\}$ as the candidate equation system for the test problem x_i. We train a ranking model to choose the alignment with the highest probability $p(a_k|x_i, t_j; \nu_a)$, where ν_a is the model parameter vector.

$$p(a_k) = \frac{\exp(\nu_a \cdot f(x_i, a_k))}{\sum_{a'_k \in A} \exp(\nu_{a'} \cdot f(x_i, a'_k))}$$

We use the same ranking model as in template selection stage and the objective function is changed to:

$$\frac{1}{2}\|\nu_a\|^2 + C\sum_i l(\nu_a^T f(x_i, a_k)^+ - \nu_a^T f(x_i, a_l)^-)$$

We design more fine-grained features for each number slot to formulate the alignment feature vector $f(x_i, a_k)$. It contains the following features in Table 2.

Textual Features
Match textual expressions in template fragment aligned to each number slot (pair)
Quantity Features
Aligned unit sequence in $Q(t_j)$
Aligned normalized unit sequence in $Q(t_j)$
Relationship with noun phrase
Optimal number 1 or 2 is used?
Solution Features
Is integer solution?
Is positive solution?

Table 2: Features for alignment ranking.

At the textual level, we want to capture textual expressions describing each number slot. For example, in the template $(1 - n_1) * n_2 = x$, we have mined patterns of $1 - n_1$ in 3.2, such as "a discount of n_1 %", "mark down n_1 %", etc. Given the problem in Figure 2 as the test problem, alignment (**1-0.28**) $* 275 = x$ matches textual expressions, while (**1-275**) $* 0.28 = x$ does not.

For quantity features, we use the alignment-ordered unit sequence. For the problem in Figure 2 mapping to template $(1 - n_1) * n_2 = x$, we have two different alignments: $\{n_1$:0.28, n_2:275$\}$, $\{n_1$:275,n_2:0.28$\}$. Their aligned unit sequences are $\{$%, \$$\}$ and $\{$\$,%$\}$ respectively. We also use the relations of quantities with noun phrases to differentiate number slot interaction with unknown variable slots and number slots, such as $n_1 * x$ and $n_1 * n_2$.

Some templates have numerical solution properties while others do not. For example, template $x_1 = (n_1 - n_2)/(n_3 - n_4)$ would be less likely to have any strong indication of integer solution properties. We count the percentage of integer/positive solutions from the corresponding problems as the probability that this template prefers an integer/positive solution.

4.3 Model Discussion

Our method has two main differences from previous template-based methods (Kushman et al., 2014; Zhou et al., 2015; Upadhyay et al., 2016).

First, previous methods implicitly model mapping from problem text to templates. We learn fine-grained textual expressions mapped to template fragments; and explicitly model the property of templates with template sketches. Second, previous methods align numbers for all templates in a training set, while we only examine the N most probable templates. This significantly reduces the equation candidate search space. Given a problem in which m numbers align with a template of n number slots, the number of possible equation candidates would be A_m^n. The search space grows linearly with the number of templates in the training data. Suppose $m = 5$, $n = 4$ and we have 1000 templates, the total space would be $(5 * 4 * 3 * 2) * 1000 = 120,000$ for one problem in Zhou et al. (2015), and will be much larger if it considers unknown variable alignment as in (Kushman et al., 2014).

5 Experiments

Settings As demonstrated in Huang et al. (2016), previous datasets for math problems are limited in both scale and diversity. We conduct our experiment on their dataset Dolphin18K. We use the linear subset, containing 10,644 problems in total. We use two baseline systems for comparison: (1) ZDC (Zhou et al., 2015) is a statistical learning method that is an improved version of KAZB (Kushman et al., 2014)[1]. (2) SIM (Huang et al., 2016) is a simple similarity based method. We do not compare other systems because they only solve one specific type of problem, e.g. Hosseini et al. (2014) only handle addition/subtraction problems and Koncel-Kedziorski et al. (2015) aim to solve problems with one single linear equation. Experiments are conducted using 5-fold cross-validation with 80% problems randomly selected as training data and the remaining 20% for testing. We report the solution accuracy.

5.1 Overall Evaluation Results

Table 3 shows the overall performance of different systems. In the table, the size of a template is the number of problems corresponding to a template. For example, for templates with a size 100 or larger, their problem counts add up to 1,807.

Template Size	problems	ZDC (%)	SIM (%)	Ours (%)
>=100	1807	34.2	29.7	64.5
>=50	4281	31.1	27.2	39.3
>=20	5392	29.4	25.8	36.9
>=10	6216	25.3	24.6	35.7
>=6	6827	21.7	20.2	**34.6**
>=5	7081	21.6	20.1	34.3
>=4	7262	21.1	19.8	33.8
>=3	7466	20.7	19.7	33.2
>=2	8229	20.6	20.3	32.2
>=1	10644	17.9	18.4	**28.4**

Table 3: Overall evaluation results.

From the table, we observe that our model consistently achieves better performance than the baselines on all template sizes. As the template size becomes larger, all three systems achieve better performance. When template size equals 6 (TS6, as a de-facto template size constrain adopted in ZDC), our model achieve an absolute increase of over 12% (59% relative). This demonstrates the effectiveness of our proposed method.

[1] We ignore KAZB because it does not complete running on the dataset in three days

When including long tail problems with a template size less than 2, performance of all three systems drop significantly. This is because the templates of these problems are not seen in the training set, and thus are difficult to solve using these template-based methods. Still, we have at least 10% absolute (54% relative) accuracy increase on the whole test set compared to the two baselines. Previous template-based methods require templates size larger than 6 in the data as constraints. From the result, we can see that our method relaxes the template size constraint and matches more problems with less frequent templates.

5.2 Accuracy per Template

Here we investigate the performance of different templates. In Table 4, we sample some dominant templates and report their accuracies. For our model, we report both template retrieval accuracy and final solution accuracy.

As we can see, our method performs better than the baselines for most dominant templates. Performance of the dominant templates can reach an accuracy level of 60%. This proves that our template sketch and textual expressions are effective in capturing rich template information.

To our surprise, some templates tend to perform better than others even with smaller template sizes. For example, $x_1 = n_1 - n_2$, which represents the subtraction problem, has 63 problems but performs not as well as $x_1 = (n_1 - n_2)/(n_3 - n_4)$ which has 48 problems. We look into their corresponding problems and find out that $x_1 = n_1 - n_2$ are applied to more themes in natural language than $x_1 = (n_1 - n_2)/(n_3 - n_4)$, which are almost about the theme of "coordinate slope".

In our model, there is a gap between template retrieval accuracy and final solution accuracy, which means that although we select the correct template candidates for the problem, the alignment model cannot rank the equations correctly.

5.3 Two-Stage Evaluation

Next, we evaluate the performance of our two-stage system. Accuracy of template retrieval and alignment ranking is shown in Table 5.

For template retrieval accuracy, Hit@N means the correct template for a problem is included in the top N list returned by our model. We estimate the best achievable performance by using oracle template retrieval. The result is **47.1%** (Hit@ALL), which means 47.1% of the templates exist more than once in the problem set. Please note that our template retrieval evaluation may be underestimated, since in some cases, a test problem can be solved by different templates.

We then use the top N templates as input for both our alignment ranking and ZDC. From the table, we have the following observations: (1) Hit@3 performs better compared to Hit@1 for both systems. This confirms our claim that some templates are similar and we need to incorporate more fine-grained features to differentiate in the alignment step; (2) It obtains the highest accuracy when $N = 3$ and decreases when N gets larger. Both systems get benefits from our template retrieval which helps retrieve relevant templates and reduce the hypothesis space of equations; (3) Given the same N templates input, our alignment ranking achieves better performance than ZDC. This implies that our features are more indicative.

5.4 Feature Ablation

This section describes our feature ablation study.

Template Retrieval In Table 6, we conduct three configurations against our model (FULL). Each ablated configuration corresponds to one category of our template sketch. From the table, we can see that all three categories of features contribute to system performance. We remove QUANTITY results in the worse performance comparing to the FULL model.

Alignment Ranking In Table 7, N means to select the top N templates in the previous stage for alignment. The column "Correct Template" represents how well the alignment model can perform given the correct template input for alignment. Our alignment model (FULL) performs the best compared to the three ablated settings, which confirms the effectiveness of template properties.

5.5 Error Analysis

We have observed that template-based methods have difficulty solving problems with small template sizes, especially for cases that have a single problem instance (i.e. template size = 1). We sample 100 problems in which our system makes mistakes in the dev set of Dolphin18K and summarize them in Table 8.

Quantity Type The types of quantities are difficult to determined. For the example problem in

				Ours	
Template	problems	ZDC (%)	SIM (%)	Template retrieval Acc (%)	Final Acc (%)
$n_1 * x_1 = n_2$	548	26.3	23.9	87.0	58.7
$n_1/x_1 = n_2/n_3$	453	21.4	29.8	94.1	61.5
$x_1 = n_1 * n_2$	403	23.6	28.0	78.9	63.4
$n_1 * x_1 + n_2 * x_2 = n_3;$ $x_1 + x_2 = n_4$	300	86.3	69.7	94.9	85.8
$x_1 = n_1 * n_2 * n_3$	103	22.3	32.0	67.0	55.0
$x_1 + x2 = n_1$ $x1 - x_2 = n2$	80	39.7	48.8	79.4	65.1
$x_1 = n_1 - n_2$	63	11.7	15.9	50.7	23.4
$x_1 = (n_1 - n_2)/(n_3 - n_4)$	48	14.9	18.8	95.7	89.4

Table 4: Accuracy Per Template. Template retrieval acc reports percent of templates appears in one of the top 3 templates returned by our method.

Hit@N	1	2	3	4	5	10	20	50	ALL
Template retrieval Acc (%)	17.5	22.4	26.3	27.2	28.0	30.2	32.7	35.2	47.1
Final Acc (%)	24.9	27.6	**28.4**	27.9	27.4	25.3	22.3	22.1	20.1
ZDC (%)	19.5	20.1	20.1	19.9	19.8	19.1	18.9	18.6	17.9

Table 5: Results of template retrieval and final accuracy with different top N templates retrieved.

Model	Hit @1 (%)	Hit @3 (%)	Hit @10 (%)
FULL	17.5	26.3	30.2
-TEXTUAL	14.1	24.7	28.4
-QUANTITY	11.4	23.4	25.9
-QUESTION	16.9	25.4	29.8

Table 6: Feature ablation of template retrieval.

Model	Correct Template (%)	N= 1 (%)	N= 3 (%)
FULL	34.5	24.9	28.4
-TEXTUAL	31.9	22.2	25.1
-QUANTITY	29.2	20.9	23.3
-SOLUTION	26.3	18.7	21.2

Table 7: Feature ablation of alignment ranking.

the table, if we can detect “24 male” is the same as “men”, the problem can be solved.

Relation/State Detection If we can identify the changed states or mathematical relations between variables, we can solve this category of problems. In the example problem, it is important to understand that “commission is taken out” is my money state.

External Knowledge This requires specific mathematical models, such as permutation and combination, or commonsense knowledge, e.g. a dice has 6 sides.

Equation Decomposition The limitation of template-based approaches is that they require test problems belonging to one of the templates seen in training. Thus, for problems corresponding to template sizes less than 2, we can decompose templates into smaller units and conduct learning more precisely. We then need to generate the equations, which is also a challenge.

6 Conclusion and Future Work

In this paper, we propose a novel approach to solving math word problems with rich information of templates. We learn mappings between textual expressions and template fragments. Furthermore, we automatically construct sketches for each template. We implement a two-stage system, including template retrieval and alignment ranking. Experiments show that our method performs signifi-

Category	Math Problem
Quantity Type (10%)	The ratio of **women** to **men** in a certain club is 3 to 2. If there are 24 **male** club members, then how many female club members are there?
Relation/State Detection (12%)	If I am selling something for $25,000 and **a 7% commission is taken out**, how much money will I **be left** with?
External Knowledge (23%)	Find the probability that total score is 10 or more given at least one dice show 6 if 2 dice red & blue thrown?
Equation Decomposition (55%)	The average weight of A, B and C is 45 kg. If the average weight of A and B is 40 kg and that of B and C is 43 kg, the weight of B is?

Table 8: Error Categorization.

cantly better than two state-of-the-art systems.

Based on our error analysis, we plan to improve our model by detecting quantity types more accurately, learning relations and incorporating commonsense knowledge. For long tail problems with a template size less 2, we want to utilize the fine-grained expressions we have learned and decompose equations for learning. Then we can reason with small equation units to generate a final equation in testing. We would like to leverage semantic parsing and transform math problems to a more structured representation. Additionally, we plan to apply our findings to generating math problem.

7 Acknowledgments

This work is supported by the National Natural Science Foundation of China (61472453, U1401256, U1501252, U1611264). Thanks to the anonymous reviewers for their helpful comments and suggestions.

References

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. Http://arxiv.org/abs/math/0701393.

Daniel G. Bobrow. 1964a. Natural language input for a computer problem solving system. Technical report, Cambridge, MA, USA.

Daniel G. Bobrow. 1964b. Natural language input for a computer problem solving system. Ph.D. Thesis.

Diane J. Briars and Jill H. Larkin. 1984. An integrated model of skill in solving elementary word problems. *Cognition and Instruction*, 1(3):245–296.

Eugene Charniak. 1968. Carps, a program which solves calculus word problems. Technical report.

Eugene Charniak. 1969. Computer solution of calculus word problems. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, pages 303–316.

Denise Dellarosa. 1986. A computer simulation of children's arithmetic word-problem solving. *Behavior Research Methods, Instruments, & Computers*, 18(2):147–154.

Charles R. Fletcher. 1985. Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods, Instruments, & Computers*, 17(5):565–571.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 2000. *Large Margin Rank Boundaries for Ordinal Regression*, chapter 7.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Christian Liguda and Thies Pfeiffer. 2012. Modeling math word problems with augmented semantic networks. In *Natural Language Processing and Information Systems. International Conference on Applications of Natural Language to Information Systems (NLDB-2012)*, pages 247–252.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29(2):93–122.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752. The Association for Computational Linguistics.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen tau Yih. 2016. Learning from explicit and implicit supervision jointly for algebra word problems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Ma Yuhui, Zhou Ying, Cui Guangzuo, Ren Yun, and Huang Ronghuai. 2010. Frame-based calculus of solving arithmetic multistep addition and subtraction word problems. *Education Technology and Computer Science, International Workshop*, 2:476–479.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Structural Embedding of Syntactic Trees for Machine Comprehension

Rui Liu[*] **Junjie Hu**[*] **Wei Wei**[*] **Zi Yang**[*] **Eric Nyberg**
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh PA 15213, USA
{ruil, junjieh, weiwei, ziy, ehn}@cs.cmu.edu

Abstract

Deep neural networks for machine comprehension typically utilizes only word or character embeddings without explicitly taking advantage of structured linguistic information such as constituency trees and dependency trees. In this paper, we propose *structural embedding of syntactic trees* (SEST), an algorithm framework to utilize structured information and encode them into vector representations that can boost the performance of algorithms for the machine comprehension. We evaluate our approach using a state-of-the-art neural attention model on the SQuAD dataset. Experimental results demonstrate that our model can accurately identify the syntactic boundaries of the sentences and extract answers that are syntactically coherent over the baseline methods.

1 Introduction

Reading comprehension such as SQuAD (Rajpurkar et al., 2016) or NewsQA (Trischler et al., 2016) requires identifying a span from a given context, which is an extension to the traditional question answering task, aiming at responding questions posed by human with natural language (Nyberg et al., 2002; Ferrucci et al., 2010; Liu, 2017; Yang, 2017). Many works have been proposed to leverage deep neural networks for such question answering tasks, most of which involve learning the query-aware context representations (Dhingra et al., 2016; Seo et al., 2017; Wang and Jiang, 2016; Xiong et al., 2017). Although deep learning based methods demonstrated great potentials for question answering, none them take syntactic information of the sentences such as *constituency tree* and *dependency tree* into consideration. Such techniques have been proven to be useful in many natural language understanding tasks in the past and illustrated noticeable improvements such as the work by (Rajpurkar et al., 2016). In this paper, we adopt similar ideas but apply them to a neural attention model for question answering.

The constituency tree (Manning et al., 1999) of a sentence defines internal nodes and terminal nodes to represent phrase structure grammars and the actual words. Figure 1 illustrates the constituency tree of the sentence "the architect or engineer acts as the project coordinator". Here, "the architect or engineer" and "the project coordinator" are labeled as noun phrases ("NP"), which is critical for answering the question below. Here, the question asks for the name of certain occupation that can be best answered using an noun phrase. Utilizing the know ledge of a constituency relations, we can reduce the size of the candidate space and help the algorithm to identify the correct answer.

> *Whose role is to design the works, prepare the specifications and produce construction drawings, administer the contract, tender the works, and manage the works from inception to completion?*

On the other hand, a dependency tree (Manning et al., 1999) is constructed based on the dependency structure of a sentence. Figure 2 displays the dependency tree for sentence

> *The Annual Conference, roughly the equivalent of a diocese in the Anglican Communion and the Roman Catholic Church or a synod in some Lutheran denominations such as the Evangelical Lutheran Church in America, is the basic unit of organization within the UMC.*

[*] Authors contributed equally to this work.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 815–824
Copenhagen, Denmark, September 7–11, 2017.

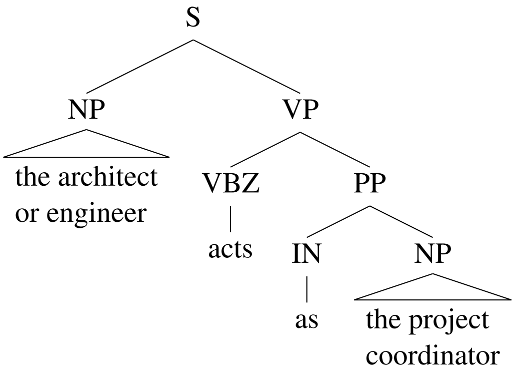


Figure 1: The constituency tree of context "the architect or engineer acts as the project coordinator"

"The Annual Conference" being the subject of "the basic unit of organization within the UMC" provides a critical clue for the model to skip over a large chunk of the text when answering the question "What is the basic unit of organization within the UMC". As we show in the analysis section, adding dependency information dramatically helps identify dependency structures within the sentence, which is otherwise difficult to learn.

In this paper, we propose Structural Embedding of Syntactic Trees (SEST) that encode syntactic information structured by constituency tree and dependency tree into neural attention models for the question answering task. Experimental results on SQuAD dataset illustrates that the syntactic information helps the model to choose the answers that are both succinct and grammatically coherent, which boosted the performance on both qualitative studies and numerical results. Our focus is to show adding structural embedding can improve baseline models, rather than directly compare to published SQuAD results. Although the methods proposed in the paper are demonstrated using syntactic trees, we note that similar approaches can be used to encode other types of tree structured information such as knowledge graphs and ontology relations.

2 Methodology

The general framework of our model is illustrated in Figure 3. Here the input of the model is the embedding of the context and question while the output is two indices *begin* and *end* which indicate the begin and end indices of the answer in the context space.

The input of the model contains two parts: the word/character model and the syntactic model. The shaded portion of our model in Figure 3 represents the encoded syntactic information of both context and question that are fed into the model. To gain an insight of how the encoding works, consider a sentence which syntactic tree consists of four nodes $(o1, o2, o3, o4)$. A specific word is represented to be a sequence of nodes from its leave all the way to the root. We cover how this process work in detail in Section 3.1.1 and 3.1.2. Another input that will be fed into deep learning model is the embedding information for words and characters respectively. There are many ways to convert words in a sentence into a high-dimensional embedding. We choose GloVe (Pennington et al., 2014b) to obtain a pre-trained and fixed vector for each word. Instead of using a fixed embedding, we use Convolutional Neural Networks (CNN) to model character level embedding, which values can be changed during training (Kim, 2014). To integrate both embeddings into the deep neural model, we feed the concatenation of them for the question and the context to be the input of the model.

The inputs are processed in the embedding layer to form more abstract representations. Here we choose a multi-layer bi-directional Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) to obtain more abstract representations for words in the contexts and questions.

After that, we employ an attention layer to fuse information from both the contexts and the questions. Various matching mechanisms using attentions have been extensively studied for machine comprehension tasks (Xiong et al., 2017; Seo et al., 2017; Wang et al., 2016; Wang and Jiang, 2016). We use the Bi-directional Attention flow model (Seo et al., 2017) which performs context-to-question and question-to-context attentions in both directions. The context-to-question attention signifies which question words are most relevant to each context word. For each context word, the attention weight is first computed by a softmax function with question words, and the attention vector of each context word is then computed by a weighted sum of the question words' embeddings obtained from the embedding layer. The question-to-context attention summarizes a context vector by performing a soft attention with context words given the question. We then represent each context word as the concatenation of the embedding vector obtained from the embedding layer, the atten-

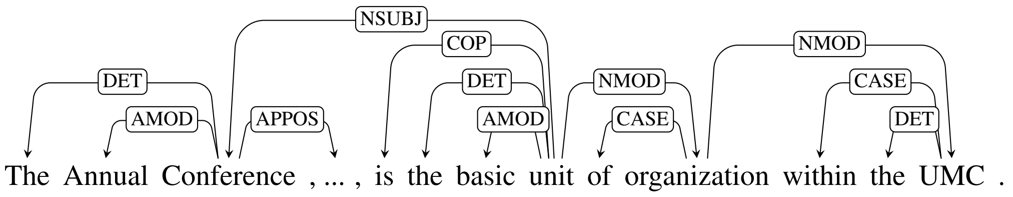

NSUBJ
COP
DET
DET
NMOD
NMOD
CASE
AMOD
APPOS
AMOD
CASE
DET
The Annual Conference , ... , is the basic unit of organization within the UMC .

Figure 2: Partial dependency parse tree of an example context "The Annual Conference, roughly the equivalent of a diocese in the Anglican Communion and the Roman Catholic Church or a synod in some Lutheran denominations such as the Evangelical Lutheran Church in America, is the basic unit of organization within the UMC."

tion vector obtained from the context-to-question attention and the context vector obtained from the question-to-context attention. We then feed the concatenated vectors to a stacked bi-directional LSTM with two layers to obtain the final representations of context words. We note that our proposed structural embedding of syntactic trees can be easily applied to any attention approaches mentioned above.

For the machine comprehension task in this paper, the answer to the question is a phrase in the context. In the output layer, we use two softmax functions over the output of the attention layer to predict the begin and end indices of the phrase in the context.

3 Structural Embedding of Syntactic Tree

We detail the procedures of two alternative implementation of our methods: the Structural Embedding of Constituency Trees model (SECT) and the Structural Embedding of Dependency Trees model (SEDT). We assume that the syntactic information has already been generated in the preprocessing step using tools such as the Stanford CoreNLP (Manning et al., 2014).

3.1 Syntactic Sequence Extraction

We first extract a *syntactic collection* $\mathcal{C}(p)$ for each word p, which consists of a set of nodes $\{o_1, o_2, \ldots, o_{d-1}, o_d\}$ in the syntactic parse tree $\mathcal{T}$. Each node o_i can be a word, a grammatical category (e.g., part-of-speech tagging), or a dependency link label, depending on the type of syntactic tree we use. To construct syntactic embeddings, the first thing we need to do is to define a specific *processing order* $\mathcal{A}$ over the syntactic collection $\mathcal{C}(p)$, in which way we can extract a *syntactic sequence* $\mathcal{S}(p)$ for the word p.

3.1.1 Structural Embedding of Constituency Trees (SECT)

The constituency tree is a syntactic parse tree constructed by phrase structure grammars (Manning et al., 1999), which defines the way to hierarchically construct a sentence from words in a bottom-up manner based on constituency relations. Words in the contexts or the questions are represented by leaf nodes in the constituency tree while the non-terminal nodes are labeled by categories of the grammar. Non-terminal nodes summarize the grammatical function of the sub-tree. Figure 1 shows an example of the constituency tree with "the architect or engineer" being annotated as a noun phrase (NP).

A path originating from the leaf node to the root node captures the syntactic information in the constituency tree in a hierarchical way. The higher the node is, the longer span of words the sub-tree of this node covers. Hence, to extract the syntactic sequence $\mathcal{S}(p)$ for a leaf node p, it is reasonable to define the processing order $\mathcal{A}(p)$ from the leaf p all the way to its root. For example, the syntactic sequence for the phrase "the project coordinator" in Figure 1 is detected as (NP, PP, VP, S). In practice, we usually take the last hidden units of Bi-directional encoding mechanisms such as Bi-directional LSTM to represent the sequence state, as is indicated in Figure 4 (a). We set a window size to limit the amount of information that is used in our models. For example, if we choose the window size as 2, then the syntactic sequence becomes (NP, PP). This process is introduced for both performance and memory utilization consideration, which is discussed in detail in Section 4.5.

In addition, a non-terminal node at a particular position in the syntactic sequence defines the begin and end indices of a phrase in the context. By measuring the similarity between syntactic se-

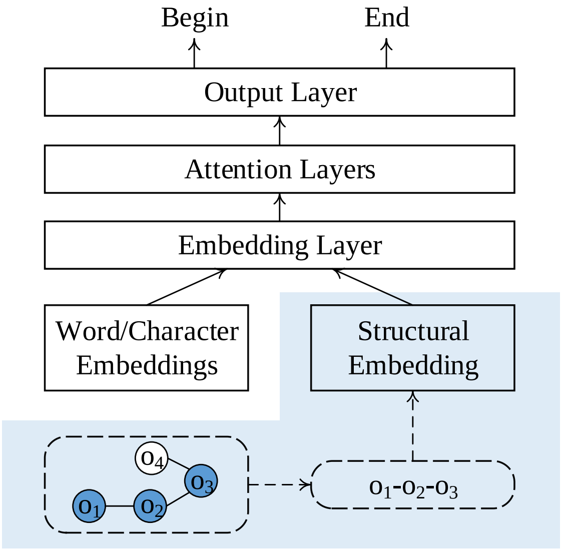


Figure 3: Model Framework. The neural network for training and testing is built by components with solid lines, which includes the embedding layer, attention layer, and output layer. The shaded area highlights the part of the framework that involves syntactic information. Components with dashed lines is an example to illustrate how syntactic information is decoded. Here a sentence is decomposed into a syntactic tree with four nodes and the syntactic information for a specific word is recorded as the path from its position in the syntactic tree to the root, i.e. (o_1, o_2, o_3) in this case.

quences $\mathcal{S}(p)$ extracted for each word p of both the question and the context, we are able to locate the boundaries of the answer span. This is done in the attention layer shown in Figure 3.

3.1.2 Structural Embedding of Dependency Trees (SEDT)

The dependency tree is a syntactic tree constructed by dependency grammars (Manning et al., 1999), which defines the way to connect words by directed links that represent dependencies. A dependency link is able to capture both long and short distance dependencies of words. Relations on links vary in their functions and are labeled with different categories. For example, in the dependency tree plotted in Figure 2, the link from "unit" to "Conference" indicates that the target node is a nominal subject (i.e. NSUBJ) of the source node.

The syntactic collection $\mathcal{C}(p)$ for dependency tree is defined as p's children, each represented by its word embedding concatenated with a vector that uniquely identifies the dependency label. The processing order $\mathcal{A}(p)$ for dependency tree is then defined to be the dependent's original order in the sentence.

Take the word "unit" as an example, we encode the dependency sub-tree using a Bi-directional LSTM, as indicated in Figure 4 (b). In such as a sub-tree, since children are directly linked to the root, they are position according to the original sequence in the sentence. Similar to the syntactic encoding of C-Tree, we take the last hidden states as its embedding.

Similar to SECT, we use a window of size l to limit the amount of syntactic information for the learning models by choosing only the l-nearest dependents, which is again reported in Section 4.5.

3.2 Syntactic Sequence Encoding

Similar to previous work (Cho et al., 2014; Kim, 2014), we use a neural network to encode a variable-length syntactic sequence into a fixed-length vector representation. The encoder can be a Recurrent Neural Network (RNN) or a Convolutional Neural Network (CNN) that learns a structural embedding for each node such that embedding of nodes under similar syntactic trees are close in their embedding space.

We can use a Bi-directional LSTM as our RNN encoder, where the hidden state $\mathbf{v}_t^p$ is updated according to Eq. 1. Here $\mathbf{x}_t^p$ is the t^{th} node in the syntactic sequence of the word p, which is a vector that uniquely identifies each syntactic node. We obtain the structural embedding of the given word p, $\mathbf{v}_{\text{Bi-LSTM}}^p = \mathbf{v}_T^p$ to be the final hidden state.

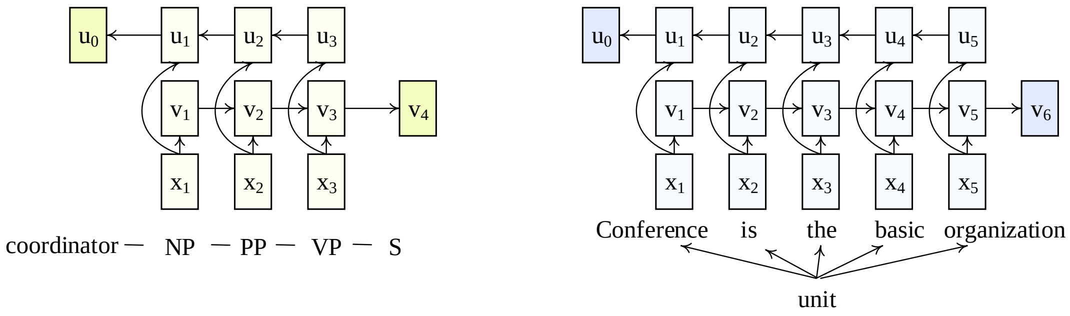
$$\mathbf{v}_t^p = \text{Bi-LSTM}(\mathbf{v}_{t-1}^p, \mathbf{x}_t^p) \tag{1}$$

Alternatively, we can also use CNN to obtain embeddings from a sequence of syntactic nodes. We denote l as the length of the filter of the CNN encoder. We define $\mathbf{x}_{i:i+l}^p$ as the concatenation of the vectors from $\mathbf{x}_i^p$ to $\mathbf{x}_{i+l-1}^p$ within the filter. The i^{th} element in the j^{th} feature map can be obtained in Eq. 2. Finally we obtain the structural embedding of the given word p by $\mathbf{v}_{\text{CNN}}^p$ in Eq. 3.

$$\mathbf{c}_{i,j}^p = f(\mathbf{w}_j \cdot \mathbf{x}_{i:i+l-1} + b_j) \tag{2}$$

$$\mathbf{v}_{\text{CNN}}^p = \max_{row}(\mathbf{c}^p) \tag{3}$$

where $\mathbf{w}_j$ and b_j are the weight and bias of the j^{th} filter respectively, f is a non-linear activation function and $\max_{row}(\cdot)$ takes the maximum value along rows in a matrix.



(a) A SECT example

(b) A SEDT example

Figure 4: Two examples are used to illustrate how the syntactic information is encoded for SECT and SEDT respectively. Take Bi-directional LSTM as examples, where $\mathbf{x}$ is a vector such as word embedding, $\mathbf{v}$ and $\mathbf{u}$ are the outputs of the forward and backward LSTMs respectively. For SECT, we encode the syntactic sequence (NP, PP, VP) for the word "coordinator" in Figure 1. We use fixed vectors for syntactic tags (e.g., NP, PP and VP), initialized with multivariate normal distribution. The final representation for the target word "coordinator" can be represented as the concatenation $[\mathbf{Ew}; \mathbf{u}_0; \mathbf{v}_4]$, where $\mathbf{Ew}$ is the word embedding for "coordinator" that is 100 dimensions in our experiments and each of the encoded vector $\mathbf{u}$ and $\mathbf{v}$ can be 30 dimensional. For SEDT, we encode the word "unit" in Figure 2 with its dependent nodes including "Conference", "is", "the", "basic", "organization", ordered by their positions in the original sentence. Each word is represented with its word embedding. Similar to SECT, the final representation is the concatenation $[\mathbf{Ew}; \mathbf{u}_0; \mathbf{v}_6]$, which will be sent to the input layer of a neural network.

4 Experiments

We conducted systematic experiments on the SQuAD dataset (Rajpurkar et al., 2016). We compared our methods against Bi-Directional Attention Flow (BiDAF), as well as the SEST models described in Section 3.

4.1 Preprocessing

A couple of preprocessing steps is performed to ensure that the deep neural models get the correct input. We segmented context and questions into sentences by using NLTK's Punkt sentence segmenter[1]. Words in the sentences were then converted into symbols by using PTB Tokenizer[2]. Syntactic information including POS tags and syntactic trees were acquired by Stanford CoreNLP utilities (Manning et al., 2014). For the parser, we collected constituent relations and dependency relations for each word by using tree annotation and enhanced dependencies annotation respectively. To generate syntactic sequence, we removed sequences whose first node is a punctuation ("$", ":", "#", ".", " " ", " " ", ","). To use dependency labels, we removed all the subcategories (e.g., "nmod:poss" ⇒ "nmod").

4.2 Experiment Setting

We run our experiments on a machine that contains a single GTX 1080 GPU with 8GB VRAM. All of the models being compared have the same settings on character embedding and word embedding. As introduced in Section 2, we use a variable character embedding with a fixed pre-trained word embedding to serve as part of the input into the model. The character embedding is implemented using CNN with a one-dimensional layer consists of 100 units with a channel size of 5. It has an input depth of 8. The max length of SQuAD is 16 which means there are a maximum 16 words in a sentence. The fixed word embedding has a dimension of 100, which is provided by the GloVe data set (Pennington et al., 2014a). The settings for syntactic embedding are slightly different for each model. The BiDAF model does not deal with syntactic information. The POS model contains syntactic information with 39 different POS tags that serve as both input and output. For SECT and SEDT the input of the model has a size of 8 with 30 units to be output. Both of them has a maximum length size that is set to be 10 and 20 respectively, which values will be further discussed in Section 4.5. They also have two different ways to encode the syntactic information as indi-

[1] http://www.nltk.org/api/nltk.tokenize.html

[2] http://nlp.stanford.edu/software/tokenizer.shtml

cated in Section 3: LSTM and CNN. We apply the same sets of parameters when we experiment them with the two models. We report the results on the SQuAD development set and the blind test set.

4.3 Predictive Performance

We first compared the performance of single models between the baseline approach BiDAF and the proposed SEST approaches, including SEPOS, SECT-LSTM, SECT-CNN, SEDT-LSTM, and SEDT-CNN, on the development dataset of SQuAD. For each model, we conducted 5 different single experiments and evaluated them using two metrics: "Exact match" (EM), which calculates the ratio of questions that are answered correctly by strict string comparison, and the F1 score, which calculates the harmonic mean of the precision and recall between predicted answers and ground true answers at the character level. As shown in Table 1, we reported the maximum, the mean, and the standard deviation of EM and F1 scores across all single runs for each approach, and highlighted the best model using bold font. SECT-LSTM is the second best method, which confirms the predictive powers of different types of syntactic information. We could see that SEDT-LSTM model outperforms the baseline method and other proposed methods in terms of both EM and F1. Another observation is that our propose models achieve higher relative improvements in EM scores than F1 scores over the baseline methods, providing the evidence that syntactic information can accurately locate the boundaries of the answer.

Moreover, we found that both SECT-LSTM and SEDT-LSTM have better performance than their CNN counterparts, which suggests that LSTM can more effectively preserve the syntactic information. As a result, we conducted further analysis of only SECT-LSTM and SEDT-LSTM models in the subsequent subsections and drop the suffix "-LSTM" for abbreviation. We built an ensemble model from the 5 single models for the baseline method BiDAF and our proposed methods SEPOS, SECT-LSTM, and SEDT-LSTM. The ensemble model choose the answer with the highest sum of confidence scores among the 5 single models for each question. We compared these models on both the development set and official test set and reported the results in Table 2. We found that the models have higher performance on the test set than the development set, which coincides with the previous results on the same data set (Seo et al., 2017; Xiong et al., 2017).

4.4 Contribution of Syntactic Sequence

To take a closer look at how syntactic sequences affect the performance, we removed the character/word embedding from our model seen in Figure 3 and conducted experiments based on the syntactic input alone. In particular, we are interested in two aspects related to syntactic sequences: First, the ability to predict answers of questions of syntactic sequences compared to complete random sequences. Second, the amount of impacts brought by our proposed ordering introduced in Section 3.1.1 and Section 3.1.2 compared to random ordering.

We compared the performance of the models using syntactic information along in their original order (i.e. SECT-Only and SEDT-Only) against their counterparts with the same syntactic tree nodes but with randomly shuffled order (i.e. SECT-Random-Order and SEDT-Random-Order) as well as the baselines with randomly generated tree nodes (i.e. SECT-Random and SEDT-Random). Here we choose the length of window size to be 10. The predictive results in terms of EM and F1 metrics are reported in Table 3. From the table we see that both the ordering and the contents of the syntactic tree are important for the models to work properly: constituency and dependency trees achieved over 20% boost on performance compared to the randomly generated ones and our proposed ordering also out-performed the random ordering. It also worth mentioning that the ordering of dependency trees seems to have less impact on the performance compared to that of the constituency trees. This is because sequences extracted from constituency trees contain hierarchical information, which ordering will affect the output of the model significantly. However, sequences extracted from dependency trees are all children nodes, which are often interchangeable and don't seem to be affected by ordering much.

4.5 Window Size Analysis

As we have mentioned in the earlier sections, limiting the window size is an important technique to prevent excessive usage on VRAM. In practice, we found that limiting the window size also benefits the performance of our models. In Table 4 we compared the predictive performance of SECT

Method	Single				Ensemble	
	EM		F1		EM	F1
	Max	Mean (±SD)	Max	Mean (±SD)		
BiDAF	67.10	66.92 (±0.23)	76.92	76.79 (±0.08)	70.97	79.53
SEPOS	67.65	66.05 (±2.94)	77.25	75.80 (±2.65)	71.46	79.70
SECT-LSTM	67.91	67.65 (±0.31)•	77.47	77.19 (±0.21) •	71.76	80.09
SEDT-LSTM	**68.13**	**67.89 (±0.10)•**	**77.58**	**77.42 (±0.19) •**	**72.03**	**80.28**
SECT-CNN	67.29	64.04 (±4.28)	76.91	73.99 (±3.89)	69.70	78.49
SEDT-CNN	67.88	66.53 (±1.91)	77.27	76.21 (±1.67)	71.58	79.80

Table 1: Performance comparison on the development set. Each setting contains five runs trained consecutively. Standard deviations across five runs are shown in the parenthesis for single models. Dots indicate the level of significance.

Method	Single		Ensemble	
	EM	F1	EM	F1
BiDAF	67.69	77.07	72.33	80.33
SECT-LSTM	68.12	77.21	72.83	80.58
SEDT-LSTM	**68.48**	**77.97**	**73.02**	**80.84**

Table 2: Performance comparison on the official blind test set. Ensemble models are trained over the five single runs with the identical network and hyper-parameters.

Method	EM	F1
SECT-Random	5.64	12.85
SECT-Random-Order	30.04	39.98
SECT-Only	34.21	44.53
SEDT-Random	0.92	8.82
SEDT-Random-Order	31.82	43.65
SEDT-Only	32.96	44.37

Table 3: Performance comparisons of models with only syntactic information against their counterparts with randomly shuffled node sequences and randomly generated tree nodes using the SQuAD Dev set

Method	Len	EM	F1
SECT	1	65.58 (± 2.58)	75.31 (± 2.39)
	5	65.74 (± 3.77)	75.48 (± 3.39)
	10	**67.51 (± 0.34)**	**77.14 (± 0.39)**
	Max	67.48 (± 0.33)	77.09 (± 0.45)
SEDT	1	66.23 (± 2.5)	73.85 (± 2.22)
	10	**67.39 (± 0.09)**	**76.93 (± 0.21)**

Table 4: Performance means and standard deviations of different window sizes on the development set.

and SEDT models by varying the length of their window sizes from 1 to maximum on the development set. In general the results illustrate that performances of the models increase with the length of the window. However, we found that for SECT model, its mean performance reached the peak while standard deviations narrowed when window size reaches 10. We also observed that larger window size does not generate predictive results that is as good as the one with window size set to 10. This suggests that there exists an optimal window size for the constituency tree. One possible explanation is increasing the window size leads to the increase in the number of syntactic nodes in the extracted syntactic sequence. Although subtrees might be similar between context and question, it is very unlikely that the complete trees are the same. Because of that, allowing the syntactic sequence to extend beyond the certain heights will introduce unnecessary noise into the learned representation, which will compromise the performance of the models. Similar conclusion holds for the SEDT model, which has an improved performance and decreased variance with the window size is set to 10. We did not perform experiments with window size beyond 10 for SEDT since it will consume VRAM that exceeds the capacity of our computing device.

4.6 Overlapping Analysis

To further understand the performance benefits of incorporating syntactic information into the question answering problem, we can take a look at the questions on which models disagree. Figure 5 is the Venn Diagram on the questions that have been corrected identified by SECT, SEDT and the baseline BiDAF model. Here we see that the vast

Question	BiDAF	SECT
Whose role is to design the works, prepare the specifications and produce construction drawings, administer the contract, tender the works, and manage the works from inception to completion?	[the architect or engineer]$_{\text{NP}}$ [acts as [the project coordinator]$_{\text{NP}}$]$_{\text{VP}}$	[the architect or engineer]$_{\text{NP}}$
What did Luther think the study of law meant?	[represented [uncertainty]$_{\text{NP}}$]$_{\text{VP}}$	[uncertainty]$_{\text{NP}}$
What caused the dynamos to be burnt out?	[the powerful high frequency currents]$_{\text{NP}}$ [set up [in [them]$_{\text{NP}}$]$_{\text{PP}}$]$_{\text{VP}}$	[powerful high frequency currents]$_{\text{NP}}$

Table 5: Questions that are correctly answered by SECT but not BiDAF

Question	Context	BiDAF	SEDT
In the layered model of the Earth, the mantle has two layers below it. What are they?	These advances led to the development of a layered model of the Earth, with a crust and lithosphere on top, the mantle below (separated within itself by seismic discontinuities at 410 and 660 kilometers), and the outer core and inner core below that.	seismic discontinuities at 410 and 660 kilometers), and the outer core and inner core	the outer core and inner core
What percentage of farmland grows wheat?	More than 50% of this area is sown for wheat, 33% for barley and 7% for oats.	33%	50%
What is the basic unit of organization within the UMC?	The Annual Conference, roughly the equivalent of a diocese in the Anglican Communion and the Roman Catholic Church or a synod in some Lutheran denominations such as the Evangelical Lutheran Church in America, is the basic unit of organization within the UMC.	Evangelical Lutheran Church in America	The Annual Conference

Table 6: Questions that are correctly answered by SEDT but not BiDAF

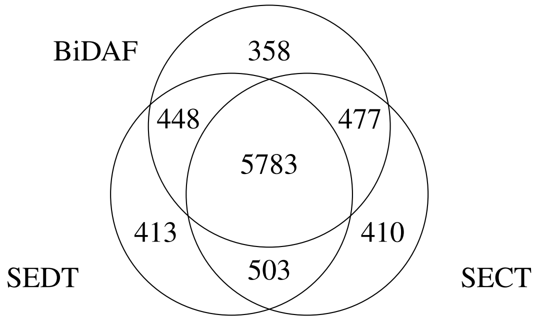


Figure 5: Venn Diagram on the number of correct answers predicted by BiDAF, SECT and SEDT

majority of the correctly answered questions are shared across all three models. The rest of them indicates questions that models disagree and are distributed fairly evenly.

To understand the types of the questions that syntactic models can do better, we extracted three questions that were correctly answered by SECT and SEDT but not the baseline model. In Table 5, all of the three questions are "Wh-questions" and expect the answer of a noun phrase (NP). Without knowing the syntactic information, BiDAF answered questions with unnecessary structures such as verb phrases (vp) (e.g. "acts as $\cdots$ ", "represented $\cdots$ ") or prepositional phrases (pp) (e.g. "in $\cdots$ ") in addition to NPs (e.g. "the architect engineer", "uncertainty" and "powerful high frequency currents") that normal human would answer. For that reason, answers provided by BiDAF failed the exact match although its answers are semantically equivalent to the ones provided by SECT. Having incorporated constituency information provided an huge advantage in inferring the answers that are most natural for a human.

The advantages of using the dependency tree in our model can be illustrated using the questions in Table 6. Here again we listed the ones that are correctly identified by SEDT but not BiDAF. As we can see that the answer provided by BiDAF for first question broke the parenthesis incorrectly, this problem that can be easily solved by utilizing dependency information. In the second example, BiDAF failed to identify the dependency structures between "50%" and the keyword being asked "wheat", which resulted in an incorrect answer that has nothing to do with the question. SEDT, on the other hand, answered the question correctly. In the third question, the key to the answer is to correctly identify the subject of question phrase "is the basic unit of organization". Using the dependency tree as illustrated in Figure 2, SEDT is able to identify the subject phrase correctly, namely "The Annual Conference". However, BiDAF failed to anwer the question correctly and selected a noun phrase as the answer.

5 Related Work

Reading Comprehension. Reading comprehension is a challenging task in NLP research. Since the release of the SQuAD data set, many works have been done to construct models on this massive question answering data set. Rajpurkar et. al. are among the first authors to explore the SQuAD. They used logistic regression with pos tagging information (Rajpurkar et al., 2016) and provided a strong baseline for all subsequent models. A steep improvement was given by the RaSoR model (Lee et al., 2016) which utilized recurrent neural networks to consider all possible subphrases of the context and evaluated them one by one. To avoid comparing all possible candidates and to improve the performance, Match-LSTM (Wang and Jiang, 2016) was proposed by using a pointer network (Vinyals et al., 2015) to extract the answer span from the context. The same idea was taken to the BiDAF (Seo et al., 2017) model by introducing a bi-directional attention mechanism. Despite the above-mentioned strong models for the machine comprehension task, none of them considers syntactic information into their prediction models.

Representations of Texts and Words. One of the main issues in reading comprehension is to identify the latent representations of texts and words (Cui et al., 2016; Lee et al., 2016; Wang et al., 2016; Xiong et al., 2017; Yu et al., 2016). Many pre-trained libraries such as word2vec (Mikolov et al., 2013) and Glove (Pennington et al., 2014a) have been widely used to map words into a high dimensional embedding space. Another approach is to generate embeddings by using neural networks models such as Character Embedding (Kim, 2014) and Tree-LSTM (Tai et al., 2015). One thing that worth mentioning is that although Tree-LSTM does utilize syntactic information, it targets at the phrases or sentences level embedding other than the word level embedding we have discussed in this paper. Many machine comprehension models include both pre-trained embeddings and variable embeddings that can be changed through a training stage (Seo et al., 2017).

6 Conclusion

In this paper, we proposed methods to embed syntactic information into the deep neural models to improve the accuracy of our model in the machine comprehension task. We formally defined our SEST framework and proposed two instances to it: the structural embedding of constituency trees (SECT) and the structural embedding of dependency trees (SEDT). Experimental results on SQuAD data set showed that our proposed approaches outperform the state-of-the-art BiDAF model, proving that the proposed embeddings play a significant part in correctly identifying answers for the machine comprehension task. In particular, we found that our model can perform especially well on exact match metrics, which requires syntactic information to accurately locate the boundaries of the answers. Similar approaches can be used to encode other tree structures such as knowledge graphs and ontology relations.

This work opened several potential new lines of research: 1) In the experiments of our paper we utilized the BiDAF model to retrieve answers from the context. Since there are no structures in the BiDAF models to specifically optimize for syntactic information, an attention mechanism that is designed for to utilize syntactic information should be studied. 2) Another direction of research is to incorporate SEST with deeper neural networks such as VD-CNN (Conneau et al., 2017) to improve learning capacity for syntactic embedding. 3) Tree structured information such as knowledge graphs and ontology structure should be studied and improve question answering tasks using similar techniques to the ones proposed in the paper.

References

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*. pages 1724–1734.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *EACL*.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423* .

Bhuwan Dhingra, Hanxiao Liu, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549* .

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building watson: An overview of the deepqa project. *AI magazine* 31(3):59–79.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* .

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*. pages 1746–1751.

Kenton Lee, Tom Kwiatkowski, Ankur Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436* .

Rui Liu. 2017. *A Phased Ranking Model for Information Systems*. Ph.D. thesis, Carnegie Mellon University.

Christopher D Manning, Hinrich Schütze, et al. 1999. *Foundations of statistical natural language processing*, volume 999. MIT Press.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

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

Eric Nyberg, Teruko Mitamura, Jaime G Carbonell, Jamie Callan, Kevyn Collins-Thompson, Krzysztof Czuba, Michael Duggan, Laurie Hiyakumoto, N Hu, Yifen Huang, et al. 2002. The javelin question-answering system at trec 2002. In *TREC*.

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

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014b. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830* .

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS*. pages 2692–2700.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905* .

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211* .

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *ICLR*.

Zi Yang. 2017. *Analytics Meta Learning*. Ph.D. thesis, Carnegie Mellon University.

Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. 2016. End-to-end answer chunk extraction and ranking for reading comprehension. *arXiv preprint arXiv:1610.09996* .

World Knowledge for Reading Comprehension: Rare Entity Prediction with Hierarchical LSTMs Using External Descriptions

Teng Long, Emmanuel Bengio, Ryan Lowe
Jackie Chi Kit Cheung, Doina Precup
{teng.long,emmanuel.bengio,ryan.lowe}@mail.mcgill.ca
{jcheung,dprecup}@cs.mcgill.ca
School of Computer Science
McGill University

Abstract

Humans interpret texts with respect to some background information, or *world knowledge*, and we would like to develop automatic reading comprehension systems that can do the same. In this paper, we introduce a task and several models to drive progress towards this goal. In particular, we propose the task of *rare entity prediction*: given a web document with several entities removed, models are tasked with predicting the correct missing entities conditioned on the document context and the lexical resources. This task is challenging due to the diversity of language styles and the extremely large number of rare entities. We propose two recurrent neural network architectures which make use of external knowledge in the form of entity descriptions. Our experiments show that our hierarchical LSTM model performs significantly better at the rare entity prediction task than those that do not make use of external resources.

1 Introduction

Reading comprehension is the ability to process some text and understand its contents, in order to form some beliefs about the world. The starting point of this paper is the fact that world knowledge plays a crucial role in human reading comprehension and language understanding. Work in the psychology of reading literature has demonstrated this point, for example by showing that readers are better able to recall the contents of a story when it describes a counter-intuitive but plausible sequence of events, rather than a bizarre or a highly predictable one (Barrett and Nyhof, 2001). This point is also central to work in the Schankian tradition of scripts (Schank and Abelson, 1977).

Despite the importance of world knowledge, previous data sets and tasks for reading comprehension have targeted other aspects of the reading comprehension problem, at times explicitly attempting to factor out its influence. In the Daily Mail/CNN dataset (Hermann et al., 2015), named entities such *Clarkson* and *Top Gear* are replaced by anonymized entity tokens like *ent212*. The Children's Book Test focuses on the role of context and memory (Hill et al., 2016a), and the fictional genre makes it difficult to connect the entities in the stories to real-world knowledge about those entities.

As a result, language models have proved to be a highly competitive solution to these tasks. Chen et al. (2016) showed that their attention-based LSTM model achieves state-of-the-art results on the Daily Mail/CNN data set. In fact, their analysis shows that more than half of the questions can be answered by exact word matching and sentence-level paraphrase detection, and that many of the remaining errors are difficult to solve exactly because the entity anonymization procedure removes necessary world knowledge.

In this paper, we propose a novel task called *rare entity prediction*, which places the use of external knowledge at its core, with the following key features. First, our task is similar in flavour to the Children's Book and other language modeling tasks, in that the goal of the models is to predict missing elements in text. However, our task involves predicting missing named entities, rather than missing words. Second, the number of unique named entities in the data set is very large, roughly on par with the number of documents. As such, there are very few instances per named entity for systems to train on. Instead, they must rely on external knowledge sources such as Freebase (Bollacker et al., 2008) in order to make inferences

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 825–834
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Context
[...] ______, who lived from 1757 to 1827, was admired by a small group of intellectuals and artists in his day, but never gained general recognition as either a poet or painter. [...]
Candidate Entities
Peter Ackroyd: Peter Ackroyd is an English biographer, novelist and critic with a particular interest in the history and culture of London. [...] William Blake: William Blake was an English poet, painter, and printmaker. [...] Emanuel Swedenborg: Emanuel Swedenborg was a Swedish scientist, philosopher, theologian, revelator, and mystic. [...]

Table 1: An abbreviated example from the Wikilinks Rare Entity Prediction dataset. Shown is an excerpt from the text (context), with a missing entity that must be predicted from a list of candidate entities. Each candidate entity is also provided with its description from Freebase.

about the likely entities that fit the context.

For our task, we use a significantly enhanced version of the Wikilinks dataset (Singh et al., 2012), with entity descriptions extracted from Freebase serving as the lexical resources, which we call the *Wikilinks Rare Entity Prediction* dataset. An example from the Wikilinks Entity Prediction dataset is shown in Table 1.

We also introduce several recurrent neural network-based models for this task which take in entity descriptions of candidate entities. Our first model, DOUBENC, combines information derived from two encoders: one for the text passage being read, and one for the entity description. Our second model, HIERENC, is an extension which considers information from a document-level context, in addition to the local sentential context. We show that language modeling baselines that do not consider entity descriptions are unable to achieve good performance on the task. RNN-based models that are trained to leverage external knowledge perform much better; in particular, HIERENC achieves a 17% increase in accuracy over the language model baseline.

2 Related Work

Related to our work is the task of *entity prediction*, also called *link prediction* or *knowledge base completion*, in the context of multi-relational data. Multi-relational datasets like WordNet (Miller, 1995) and Freebase (Bollacker et al., 2008) consist of entity-relation *triples* of the form (head, relation, tail). In entity prediction, either the head or tail entity is removed, and the model has to predict the missing entity. Recent efforts have integrated different sources of knowledge, for example combining distributional and relational semantics for building word embeddings (Fried and Duh, 2015; Long et al., 2016). While this task requires understanding and predicting associations between entities, it does not require contextual reasoning with text passages, which is crucial in rare entity prediction.

Rare entity prediction is also clearly distinct from tasks such as entity tagging and recognition (Ritter et al., 2011), as models are provided with the actual name of the entity in question, and only have to match the entity with related concepts and tags. It is more closely related to the machine reading literature from e.g. Etzioni et al. (2006); however, the authors define machine reading as primarily unsupervised, whereas our task is supervised.

A similar supervised reading comprehension task was proposed by Hermann et al. (2015) using news articles from CNN and the Daily Mail. Given an article, models are tasked with filling in blanks of one-sentence summaries of the article. The original dataset was found to have a low ceiling for machine improvement (Chen et al., 2016); thus, alternative datasets have been proposed that consist of more difficult questions (Trischler et al., 2016; Rajpurkar et al., 2016). A dataset with a similar task was also proposed by Hill et al. (2016a), where models must answer questions about short children's stories. While these tasks require the understanding of unstructured natural language, they do not require integration with external knowledge sources.

Hill et al. (2016b) proposed a method of combining distributional semantics with an external knowledge source in the form of dictionary definitions. The purpose of their model is to obtain more accurate word and phrase embeddings by combining lexical and phrasal semantics, and they achieve fairly good performance on reverse dictionaries and crossword puzzle solving tasks.

Perhaps the most related approach to our work is the one developed by Ahn et al. (2016). The authors propose a WikiFacts dataset where Wikipedia descriptions are aligned with Freebase facts. While they also aim to integrate external knowledge with unstructured natural language, their task differs from ours in that it is primarily a language modeling problem.

More recently, Bahdanau et al. (2017) investigated a similar approach to generate embeddings for out-of-vocabulary words from their definitions and applied it to a number of different tasks. However, their method mainly focuses on modeling generic concepts and is evaluated on tasks that do not require the understanding of world knowledge specifically. Our work, on the other hand, shows the effectiveness of incorporating external descriptions for modeling real-world named entities and is evaluated on a task that explicitly requires the understanding of such external knowledge.

3 Rare Entity Prediction

3.1 The *Wikilinks* Dataset

The *Wikilinks* dataset (Singh et al., 2012) is a large dataset originally designed for cross-document coreference resolution, the task of grouping entity mentions from a set of documents into clusters that represent a single entity. The dataset consists of a list of non-Wikipedia web pages (discovered using the Google search index) that contain hyperlinks to Wikipedia, such as random blog posts or news articles. Every token with a hyperlink to Wikipedia is then marked and considered an entity mention in the dataset. Each entity mention is also linked back to a knowledge base through their corresponding Freebase IDs

In order to ensure the hyperlinks refer to the correct Wikipedia pages, additional filtering is conducted to ensure that either (1) at least one token in the hyperlink (or *anchor*) matches a token in the title of the Wikipedia page, or (2) the anchor text matches exactly an anchor from the Wikipedia page text, which can be considered an alias of the page. As many near-duplicate copies of Wikipedia pages can be found online, any web pages where more than 70% of the sentences match those from their linked Wikipedia pages are discarded.

3.2 The *Wikilinks Rare Entity Prediction* Dataset

We use a significantly pre-processed and augmented version of the *Wikilinks* dataset for the purpose of entity prediction, which we call the *Wikilinks Rare Entity Prediction* dataset. In particular, we parse the HTML texts of the web pages and extract their page contents to form our corpus. Entity mentions with hyperlinks to Wikipedia are marked and replaced by a special token (**blank**), serving as the placeholder for missing entities that we would like the models to predict. The correct missing entity $\tilde{e}$ is preserved as a target. Additionally, we extract the lexical definitions of all entities that are marked in the corpus from Freebase using their Freebase IDs, which are available for all entities in the *Wikilinks* dataset. These lexical definitions will serve as the external knowledge to our models.

Number of documents	269,469
Average # blanks *per* doc	3.69
Average # candidates *per* doc	3.35
Number of unique entities	245,116
# entities with $n <= 5$	207,435 (**84.6%**)
# entities with $n <= 10$	227,481 (**92.8%**)
# entities with $n <= 20$	238,025 (**97.1%**)

Table 2: Statistics for the augmented version of the *Wikilinks* dataset, where n represents the entity frequency in the corpus. Web documents with more than 10 blanks to fill are filtered out for computational reasons.

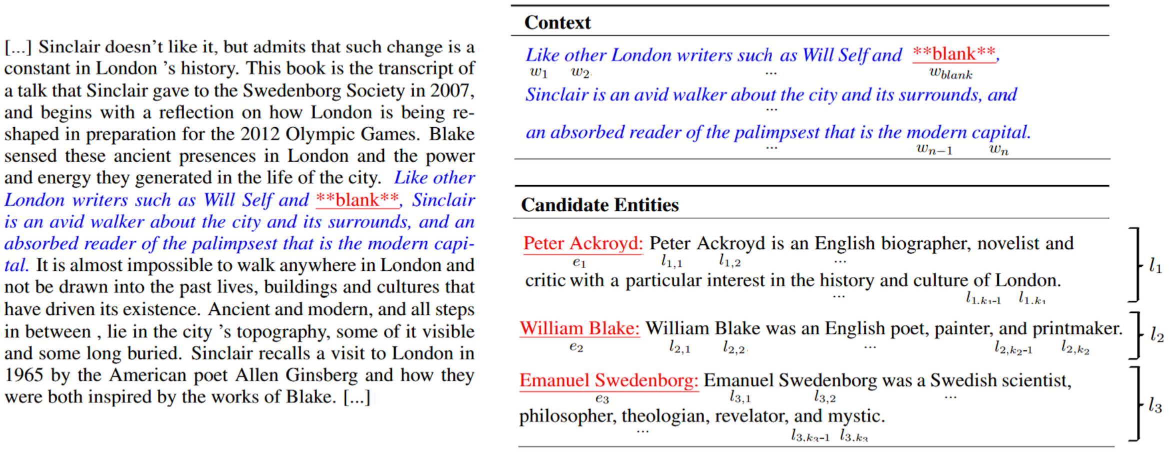
Table 2 shows some basic statistics of a subset of the corpus used in our experiments. As we can see, unlike the Children's Book dataset, which has 50k candidate entities for almost 700k context and query pairs (Hill et al., 2016a), the number of unique entities found in our dataset has the same order of magnitude as the number of documents available.

Moreover, the majority of entities appears a relatively small number of times, with 92.8% observed less than or equal to 10 times across the entire corpus. This suggests that models that only rely on the surrounding context information may not be able to correctly predict the missing entities. This further motivates us to incorporate additional information into the decision process to improve the performance. In the experiments section, we show that the external entity descriptions are indeed necessary to achieve better results.

3.3 Task Definition[1]

Here we formalize the task definition of the entity prediction problem. Given a document $\mathcal{D}$ in

[1] On notation: we use A to denote sequences, $\mathcal{A}$ to denote sets, a to denote words / entities, $\mathbf{a}$ to denote vectors, A to denote matrices.



[...] Sinclair doesn't like it, but admits that such change is a constant in London 's history. This book is the transcript of a talk that Sinclair gave to the Swedenborg Society in 2007, and begins with a reflection on how London is being reshaped in preparation for the 2012 Olympic Games. Blake sensed these ancient presences in London and the power and energy they generated in the life of the city. *Like other London writers such as Will Self and* **blank**, *Sinclair is an avid walker about the city and its surrounds, and an absorbed reader of the palimpsest that is the modern capital.* It is almost impossible to walk anywhere in London and not be drawn into the past lives, buildings and cultures that have driven its existence. Ancient and modern, and all steps in between , lie in the city 's topography, some of it visible and some long buried. Sinclair recalls a visit to London in 1965 by the American poet Allen Ginsberg and how they were both inspired by the works of Blake. [...]

Context

Like other London writers such as Will Self and **blank**, *Sinclair is an avid walker about the city and its surrounds, and an absorbed reader of the palimpsest that is the modern capital.*

Candidate Entities

Peter Ackroyd: Peter Ackroyd is an English biographer, novelist and critic with a particular interest in the history and culture of London.

William Blake: William Blake was an English poet, painter, and printmaker.

Emanuel Swedenborg: Emanuel Swedenborg was a Swedish scientist, philosopher, theologian, revelator, and mystic.

Figure 1: An example from the *Wikilinks Rare Entity Prediction* dataset. Shown is a paragraph from the dataset, along with the context (in blue italics) and the missing entity (in red underline). We also visually show the notation that we use for the remainder of this paper. The correct answer here is Peter Ackroyd.

the corpus, we split it into an ordered list of contexts $\mathcal{C} = \{\mathsf{C}_1, ..., \mathsf{C}_n\}$ where each context C_i $(1 \leq i \leq n)$ is a word sequence $(w_1, ..., w_m)$ where the special token **blank** is found. Let $\mathcal{E}$ be the set of candidate entities. For each missing entity, we want the model to select the correct entity $\tilde{e} \in \mathcal{E}$ to fill the blank slot. In our problem setting, the model also has access to the lexical resource $\mathcal{L} = \{\mathsf{L}_e \mid e \in \mathcal{E}\}$ where $\mathsf{L}_e = (l_{e_1}, ..., l_{e_k})$ is the lexical definition of entity e extracted from the knowledge base. Thus, the task of the model is to predict the correct missing entities for each empty slot in $\mathcal{D}$.

There are several possible ways to specify the candidate set $\mathcal{E}$. For instance, we could define $\mathcal{E}$ so that it includes all entities found in the corpus. However, given the extremely large amount of unique entities found in the dataset, this would render the task difficult to solve from both a practical and computational perspective. We present a simpler version of the task where $\mathcal{E}$ is the set of entities that are present in the document $\mathcal{D}$. Note that we can make the task arbitrarily more difficult by randomly sampling other entities from the entity vocabulary and adding them to candidate set.

We show an example from the *Wikilinks Entity Prediction* dataset in Figure 1, along with a visual guide to the notation from this section.

4 Model Architectures

In this section, we present two models that use the lexical definitions of entities to solve the proposed rare entity prediction problem. The basic building blocks of our models are recurrent neural networks (RNN) with long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997). An RNN is a neural network with feedback connections that allows information from the past to be encoded in the hidden layer representation, thus is ideal for modeling sequential data (Dietterich, 2002) and most language related problems.

LSTMs are an extension of RNNs which include a memory cell $\mathbf{c}_t$ alongside their hidden state representation $\mathbf{h}_t$. Reads and writes to the memory cell are controlled by a set of three gates that allow the model to either keep or discard information from the past and update their state with the current input. This allows LSTMs to model potentially longer dependencies and at the same time mitigate the vanishing and exploding gradient problems, which are quite common among regular RNNs (Bengio et al., 1994). In our experiments, we use LSTMs augmented with peephole connections (Gers et al., 2002).

We denote the output (i.e. the last hidden state) of an RNN f operating on a sequence S as $f(\mathsf{S})$, and subscript the t-th hidden state as $f_t(\mathsf{S})$.

4.1 Double Encoder (DoubEnc)

This model uses two jointly trained recurrent models, a lexical encoder $g(.)$ and a context encoder $f(.)$, and a logistic predictor P (see Figure 2).

The lexical encoder converts the definition of an entity into a vector embedding, while the context encoder repeats the same process for a given context to obtain its context embedding. These two embeddings are then used by P to predict if the

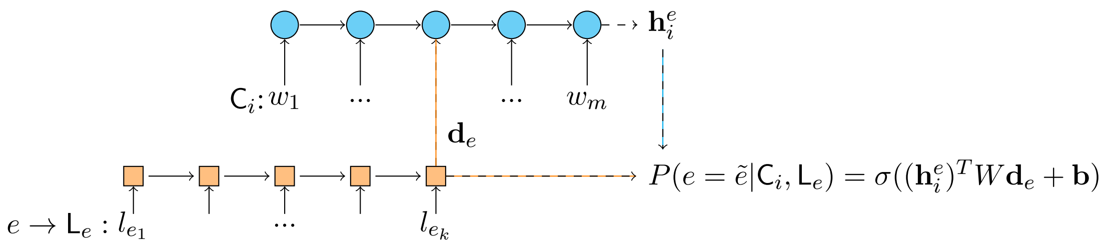

$\mathbf{h}_i^e$
C_i: w_1
...
...
w_m
$\mathbf{d}_e$
$P(e = \tilde{e}|\mathsf{C}_i, \mathsf{L}_e) = \sigma((\mathbf{h}_i^e)^T W \mathbf{d}_e + \mathbf{b})$
$e \rightarrow \mathsf{L}_e : l_{e_1}$
...
l_{e_k}

Figure 2: Our double encoder architecture. Each entity e has an associated lexical definition $\mathsf{L}_e = (l_{e_1}, l_{e_2}, ..., l_{e_k})$, which is fed through the lexical encoder g (orange squares) to provide an encoding $\mathbf{d}_e$. This definition embedding is then fed as the *blank* input token of context C_i to the context encoder f (blue circles), which provides a context embedding $\mathbf{h}_i^e$.

given entity-context pair is correct. Additionally, the blank in the context sentence is replaced by the encoded definition embedding to provide more information to f.

For an entity e in the candidate set $\mathcal{E}$ of document $\mathcal{D}$, we retrieve its corresponding lexical definition L_e, itself a sequence of words, to compute its encoding $g(\mathsf{L}_e) \equiv \mathbf{d}_e$.

For a given context C_i, we replace the embedding of the blank token with $\mathbf{d}_e$. Thus $\mathsf{C}_i = (w_1, ..., w_{blank}, ..., w_m)$ becomes $\mathsf{C}_i^e = (w_1, ..., \mathbf{d}_e, ..., w_m)$[2]. We then compute the context embedding of the new C_i^e, $f(\mathsf{C}_i^e) \equiv \mathbf{h}_i^e$.

After getting $\mathbf{h}_i^e$ and $\mathbf{d}_e$, we wish to compute the probability of candidate e being the correct entity $\tilde{e}$ missing in context C_i. This probability is the output of the predictor:

$$P(e = \tilde{e}|\mathsf{C}_i, \mathsf{L}_e) = \sigma((\mathbf{h}_i^e)^T W \mathbf{d}_e + \mathbf{b})$$

where σ is the sigmoid function, W and $\mathbf{b}$ are model parameters.

The cross term $(\mathbf{h}_i^e)^T W \mathbf{d}_e$ is a dot product between $\mathbf{h}_i^e$ and $\mathbf{d}_e$ that weighs the dimensions differently based on the learned parameters W. Similar prediction methods have been used successfully for question answering (Bordes et al., 2014; Yu et al., 2014) and dialogue response retrieval (Lowe et al., 2015).

We also experimented with only feeding $\mathbf{h}_i^e$ to the predictor, without the cross term, and found this slows down training the lexical encoder. While $\mathbf{h}_i^e$ is a function of $\mathbf{d}_e$, using $\mathbf{d}_e$ in the cross term $(\mathbf{h}_i^e)^T W \mathbf{d}_e$ provides a much shorter gradient path from the loss to the lexical encoder through $\mathbf{d}_e$, thus allowing both modules to learn at the same pace.

Given a context, the model outputs a probability for each entity $e \in \mathcal{E}$. Entities in the candidate set are then ranked against each other according to their predicted probabilities. The entity with the highest probability is considered as the most plausible answer for the missing entity in the current context. We consider the model to make an error if that entity is not $\tilde{e}$.

4.2 Hierarchical Double Encoder (HierEnc)

The double encoder architecture mentioned above considers each context independently. However, since each document consists of a sequence of contexts, the knowledge carried by other contexts in $\mathcal{C}$ could also provide useful information for the decision process of C_i. To that end, we propose a hierarchical model structure by adding a LSTM network r, which we call the temporal network (see Figure 3), on top of the double encoder architecture. Since a document is a sequence of C_is, each *time step* of this network consists of one such context, and thus is indexed with i.

Since we already have a context encoder f, we reuse the output of $f(\mathsf{C}_i^e)$ as the input of r at time step i. More specifically, we combine the embeddings generated by f into a single one via averaging: $\bar{\mathbf{h}}_i = \frac{1}{|\mathcal{E}|} \sum_{e' \in \mathcal{E}} \mathbf{h}_i^{e'}$, which then serves as the input to the temporal network for context C_i. Note that alternatively, one could aggregate information about the past predictions through other means like policies or soft attention. However, this would introduce extra complexities to the learning process. As such, we use averaging to that end.

Finally, at each time step i, the temporal network outputs an embedding $r_i(\mathsf{C}_1, ..., \mathsf{C}_n) \equiv \mathbf{r}_i$. We use this temporal embedding to predict the probability of the context-entity pair with a

[2] We mix the word / vector notation here since each word w is replaced by its corresponding word embedding vector.

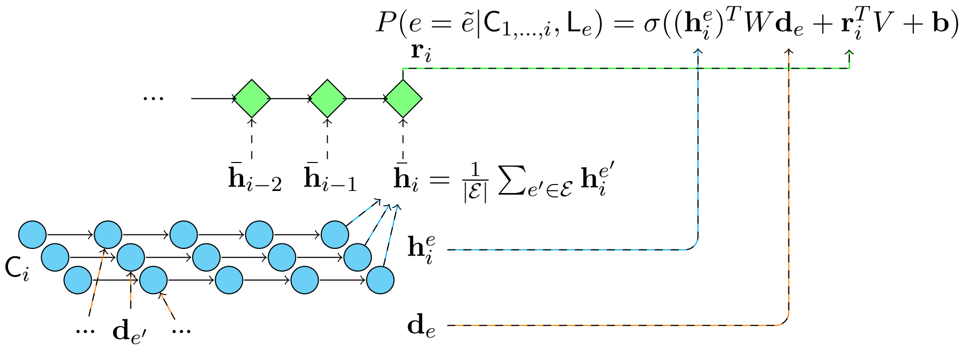

$P(e = \tilde{e}|\mathsf{C}_{1,\ldots,i}, \mathsf{L}_e) = \sigma((\mathbf{h}_i^e)^T W \mathbf{d}_e + \mathbf{r}_i^T V + \mathbf{b})$
$\mathbf{r}_i$
...
$\bar{\mathbf{h}}_{i-2}$
$\bar{\mathbf{h}}_{i-1}$
$\bar{\mathbf{h}}_i = \frac{1}{|\mathcal{E}|} \sum_{e' \in \mathcal{E}} \mathbf{h}_i^{e'}$
$\mathbf{h}_i^e$
C_i
... $\mathbf{d}_{e'}$ ...
$\mathbf{d}_e$

Figure 3: Our hierarchical encoder architecture. Each entity e is encoded as $\mathbf{d}_e$, at each time step, $\mathbf{h}_i^e$ is computed for each e. $\bar{\mathbf{h}}_i$ is the average encoding, which is fed as input to the temporal network r (green diamonds). The temporal network produces $\mathbf{r}_i$, which is used to compute $P(e = \tilde{e}|\mathsf{C}_{1,\ldots,i}, \mathsf{L}_e)$.

slightly altered logistic predictor:

$$P(e = \tilde{e}|\mathsf{C}_{1,\ldots,i}, \mathsf{L}_e) = \sigma((\mathbf{h}_i^e)^T W \mathbf{d}_e + \mathbf{r}_i^T V + \mathbf{b})$$

where W, V and $\mathbf{b}$ are model parameters. The entities in candidate set are again ranked against each other based on their probabilities.

5 Experiments

5.1 Setup

We randomly partition the data into training, validation and test sets. The training set consists of approximately 80% of the total documents, the validation and test sets comprise about 10% each.

In our experiments, the context windows are defined as the sentences where the special ***blank*** tokens are found; the lexical definitions for each entity are the first sentences of their Freebase descriptions. We experimented with different configurations of defining contexts and entity definitions, such as expanding the context windows by including sentences that come before and after the one where blank is found, as well as taking more than one sentence out of the entity description. However, results on validation set show that increasing the context window size and the definition size had very little impact on accuracies, but drastically increased the training time of all models. We thus chose to use only the immediate sentence of the context and the first sentence of the entity description.

To train our models, we use the correct missing entity for each blank as the positive example and all other entities in the candidate set as the negative examples, which we found to be more beneficial empirically than using only a subset of rest of the candidate set. During the testing phase, we present each entity in the candidate set to our models and record the probabilities output by the models. The entity with the highest probability is chosen as the model prediction. For all gradient-based methods, including both baseline models and our proposed models, the learning objective is to minimize the binary cross-entropy of the training data.

We measure the performance on our entity prediction task using the *accuracy*; that is, the number of correct entity predictions made by the model divided by the total number of predictions. This is equivalent to the metric of Recall@1 that is often used in information retrieval.

5.2 Baselines

In order to demonstrate the effects of using lexical resources as external knowledge for solving the task, we present three sets of baselines: one set of simple baselines (RANDOM and FREQENT), one LSTM-based model that only relies on the contexts and does not utilize the definitions (CONTENC), and another set of models that do make use of the entity definitions but in a simplistic fashion (TF-IDF + COS and AVGEMB + COS).

RANDOM For each context in a given document, this baseline simply selects an entity from the candidate set uniformly at random as its prediction.

FREQENT Under this baseline, we rank all entities in the candidate set by the number of times that they appear in the document. For each blank in the document, we always choose the entity with the highest frequency in that document as the prediction. Note that this baseline has access to extra information compared to the other models, in particular the total number of times each entity appears in the document.

ContEnc Instead of using their definitions, entities are treated as regular tokens in vocabulary. Thus for a particular entity e, the context sequence $\mathsf{C}_i = (w_1, ..., w_{blank}, ..., w_m)$ becomes $(w_1, ..., w_e, ..., w_m)$. We feed the sequence C_i into the context encoder and as usual take the last hidden state as the context embedding $\mathbf{h}_i^e$. Thus given C_i and $e \in E$, the probability of e being the correct missing entity is:

$$P(e = \tilde{e}|\mathsf{C}_i) = \sigma((h_i^e)^T W + \mathbf{b})$$

where again σ is the sigmoid function, W and $\mathbf{b}$ are model parameters. This model is essentially a language model baseline, that does not make use of the external a priori knowledge.

Tf-Idf + cos This method takes the term frequency-inverse document frequency (TF-IDF) vectors of the context and the entity definition as their corresponding embeddings. The aggregations of contexts and definitions are treated as their own corpora, and two separate TF-IDF transformers are fitted. Candidate entities are ranked by the *cosine similarity* between their definition vectors and the context vector. The entity with the highest cosine similarity score is chosen as the prediction.

AvgEmb + cos This baseline computes the context embedding by taking the *average* of some pre-trained word embeddings. The entities' embeddings are computed in the same way. In our experiments, we choose to use the published *GloVe* (Pennington et al., 2014) pre-trained word embeddings. Same as above, the prediction is made by considering the cosine similarity between the context embedding and the entity embeddings.

5.3 Hyperparameters

For the ContEnc baseline, we choose 300 as the size of hidden state for the encoder. For the DoubEnc and the HierEnc models, the size of hidden state for both the context encoder and the lexical encoder is set to 300. An RNN with 200 LSTM units is used as the temporal network in the hierarchical case. All three models are trained with stochastic gradient descent with Adam (Kingma and Ba, 2015) as our optimizer, with learning rates of 10^{-3} used for ContEnc and 10^{-4} used for DoubEnc as well as HierEnc. Models with the best performance on validation set are saved and used to test on test set.

Model	Accuracy Valid	Accuracy Test
Fixed Baselines		
Random	29.4%	30.1%
FreqEnt	32.9%	33.1%
Without External Knowledge		
ContEnc	39.3%	39.6%
With External Knowledge		
Tf-Idf + cos	29.2%	30.0%
AvgEmb + cos	35.5%	35.9%
DoubEnc	54.7%	54.0%
HierEnc	**57.3%**	**56.6%**

Table 3: Empirical results on *Wikilinks Entity Prediction* dataset for proposed baselines and models.

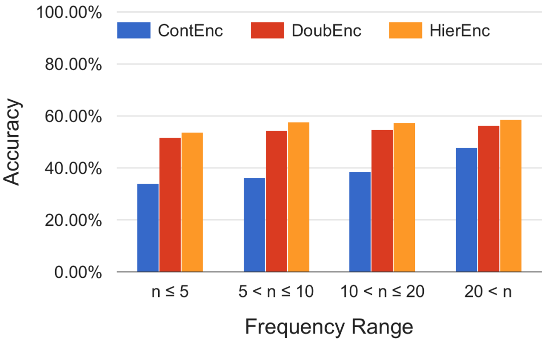


Figure 4: Accuracies of ContEnc, DoubEnc, and HierEnc on test set, for different frequency ranges; n is entity frequency in the entire corpus.

5.4 Results

Empirical results are shown in Table 3. We test our proposed model architectures (detailed in Section 4), along with baselines described in Section 5.2.

It is clear from Table 3 that models that only use contextual knowledge give relatively poor performance compared to the ones that utilize lexical resources. The large discrepancy between the context encoder and the double encoder shows that lexical resources play a crucial role in solving the task. The best result is achieved by the hierarchical double encoder, which confirms that knowing about previous contexts is indeed beneficial to the prediction at the current time step.

We performed statistical significance tests on the predictions from ContEnc, compared to the predictions made by DoubEnc and HierEnc respectively. Both tests yielded $p < 10^{-5}$. We also computed the p-value between DoubEnc and Hi-

Context & Prediction
[...] We heard from Audrey Bomse, who is with the Free Gaza movement. She was in ______, Cyprus. [...]
ContEnc: Istanbul   HierEnc: Larnaca
Candidate Set
Istanbul: Istanbul is the most populous city in Turkey, and the country's economic, cultural, and historical center.
Larnaca: Larnaca is a city on the southern coast of Cyprus and the capital of eponymous district.
Ben Macintyre: Ben Macintyre is a British author, historian, reviewer and columnist writing for The Times newspaper.
(Other candidate entities......)

Table 4: An example from the test set, with the predictions made by ContEnc and HierEnc; HierEnc was able to successfully predict the correct missing entity, *Larnaca*.

erEnc, with $p \approx 0.003$. This suggests that the performance improvement achieved by the hierarchical model is statistically significant.

6 Discussion

Figure 4 provides a performance breakdown of test accuracies over various entity frequencies for ContEnc, DoubEnc, and HierEnc. As we can see, the biggest performance gap between the baseline and our two proposed models occurs when $n \leq 5$; as entity frequencies increase, the accuracy of ContEnc also increases. This confirms the value and necessity of lexical resources, especially when entities appear extremely infrequently. We also see that HierEnc outperforms DoubEnc consistently over all frequency ranges. This suggests that by propagating information from the past through temporal network, the hierarchical model is able to reason beyond the local context, thus achieve higher accuracies.

Table 4 shows an example found in the test set, along with the predictions from ContEnc and HierEnc. Even though the context encoder baseline was able to identify that the missing entity should be a city, it incorrectly predicted *Istanbul*. This is likely because *Istanbul* appears 86 times in the dataset, whereas *Larnaca* appears only twice in the test set, *and not at all in the training set*. It seems that, although the context encoder was able to derive a strong association between Istanbul and Middle Eastern geolocations, such knowledge was not learned for Larnaca because of the lack of examples. Conversely, the hierarchical double encoder was able to take both the context and the external knowledge into account and successfully predicted the correct missing city.

One interesting observation is the margin of difference in accuracy between the context encoder and the embedding average baseline. The context encoder, which is a relatively sophisticated context-only model, only slightly outperforms the simple embedding average baseline that has no learning component. This suggests that the entity definitions are valuable in solving such tasks even when it is used in a rather simplistic way.

As we discussed in Section 5.1, we found in initial experiments that using a large context window size (including sentences before and after the sentence where blank token is found) does not have any significant positive impact on the results. This may imply that words that are most informative about the missing entity in the blank are generally found in vicinity of the blank. It is also likely that more sophisticated models will be able to use the surrounding context information more effectively, leading to greater performance increases.

7 Conclusions

In this paper, we examined the use of external knowledge in the form of lexical resources to solve reading comprehension problems. Specifically, we propose the problem of rare entity prediction. In our *Wikilinks Rare Entity Prediction* dataset, the majority of the entities have very low frequencies across the entire corpus; thus, models that solely rely on co-occurrence statistics tend to under-perform. We show that models leveraging the Freebase descriptions achieve large performance gains, particularly when this information is incorporated intelligently as in our double encoder-based models.

For future work, we plan to examine the effects of other knowledge sources. In this paper, we use entity definitions as the source of external knowledge. However, Freebase also contains other types of valuable information, such as relational information between entities. Thus, one potential direction for future work would be to incorporate both relational information and lexical definitions.

We have demonstrated the crucial role that external knowledge plays in solving tasks with many rare entities. We believe that incorporating external knowledge into other systems, such as dialogue agents, should also see similar positive results. We plan to explore the idea of external knowledge integration further in future research.

Acknowledgements

This work is supported by Samsung Advanced Institute of Technology (SAIT). We would like to thank the anonymous reviewers for their comments and suggestions. We would also like to thank Dzmitry Bahdanau, Tom Bosc, and Pascal Vincent for their discussions on related work, as well as Timothy O'Donnell for his feedback during the writing of this paper.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318* .

Dzmitry Bahdanau, Tom Bosc, Stanislaw Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. 2017. Learning to compute word embeddings on the fly. *arXiv preprint arXiv:1706.00286* .

Justin L. Barrett and Melanie A. Nyhof. 2001. Spreading non-natural concepts: the role of intuitive conceptual structures in memory and transmission of cultural materials. *Journal of Cognition and Culture* 1(1):69–100.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5(2):157–166.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. ACM, pages 1247–1250.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pages 165–180.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Association for Computational Linguistics (ACL)*.

Thomas G. Dietterich. 2002. Machine learning for sequential data: a review. In *Structural, syntactic, and statistical pattern recognition*, Springer, pages 15–30.

Oren Etzioni, Michele Banko, and Michael J. Cafarella. 2006. Machine reading. In *AAAI*. volume 6, pages 1517–1519.

Daniel Fried and Kevin Duh. 2015. Incorporating both distributional and relational semantics in word representations. In *ICLR'15: International Conference on Learning Representations (workshop)*.

Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. 2002. Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research* 3(August):115–143.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1684–1692.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016a. The goldilocks principle: reading children's books with explicit memory representations. In *ICLR 2016*.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2016b. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics (TACL)* .

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

Diederik Kingma and Jimmy Ba. 2015. Adam: a method for stochastic optimization. In *ICLR 2015*.

Teng Long, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. 2016. Leveraging lexical resources for learning entity embeddings in multi-relational data. In *ACL 2016*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu dialogue corpus: a large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of SIGDIAL, 2015*.

George A Miller. 1995. Wordnet: a lexical database for English. *Communications of the ACM* 38(11):39–41.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP), 2016*.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1524–1534.

Roger C. Schank and Robert Abelson. 1977. *Scripts, goals, plans, and understanding*. Erlbaum.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: a large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015, University of Massachusetts, Amherst.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. NewsQA: a machine comprehension dataset. *arXiv preprint arXiv:1611.09830* .

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. In *NIPS Deep Learning and Representation Learning Workshop, Montreal*.

Two-Stage Synthesis Networks for Transfer Learning in Machine Comprehension

David Golub[*]
Stanford University
golubd@cs.stanford.edu

Po-Sen Huang and **Xiaodong He**
Microsoft Research
{pshuang, xiaohe}@microsoft.com

Li Deng[†]
Citadel Securities, LLC
l.deng@ieee.org

Abstract

We develop a technique for transfer learning in machine comprehension (MC) using a novel two-stage synthesis network (SynNet). Given a high-performing MC model in one domain, our technique aims to answer questions about documents in another domain, where we use no labeled data of question-answer pairs. Using the proposed SynNet with a pretrained model on the SQuAD dataset, we achieve an F1 measure of 46.6% on the challenging NewsQA dataset, approaching performance of in-domain models (F1 measure of 50.0%) and outperforming the out-of-domain baseline by 7.6%, without use of provided annotations.[1]

1 Introduction

Machine comprehension (MC), the ability to answer questions over a provided context paragraph, is a key task in natural language processing. The rise of high-quality, large-scale human-annotated datasets for this task (Rajpurkar et al., 2016; Trischler et al., 2016) has allowed for the training of data-intensive but expressive models such as deep neural networks (Wang et al., 2016; Xiong et al., 2017; Seo et al., 2017). Moreover, these datasets have the attractive quality that the answer is a short snippet of text within the paragraph, which narrows the search space of possible answer spans.

However, many of these models rely on large amounts of human-labeled data for training. Yet data collection is a time-consuming and expensive task. Moreover, direct application of a MC model trained on one domain to answer questions over paragraphs from another domain may suffer performance degradation.

While understudied, the ability to transfer a MC model to multiple domains is of great practical importance. For instance, the ability to quickly use a MC model trained on Wikipedia to bootstrap a question-answering system over customer support manuals or news articles, where there is no labeled data, can unlock a great number of practical applications.

In this paper, we address this problem in MC through a two-stage synthesis network (SynNet). The SynNet generates synthetic question-answer pairs over paragraphs in a new domain that are then used in place of human-generated annotations to finetune a MC model trained on the original domain.

The idea of generating synthetic data to augment insufficient training data has been explored before. For example, for the target task of translation, Sennrich et al. (2016) present a method to generate synthetic translations given real sentences to refine an existing machine translation system.

However, unlike machine translation, for tasks like MC, we need to synthesize both the question and answers given the context paragraph. Moreover, while the question is a syntactically fluent natural language sentence, the answer is mostly a salient semantic concept in the paragraph, e.g., a named entity, an action, or a number, which is often a single word or short phrase.[2] Since the answer has a very different linguistic structure compared to the question, it may be more appropri-

[*] Work performed while interning at Microsoft Research.

[†] Work performed when the author was at Microsoft Research.

[1] Code will be available at https://github.com/davidgolub/QuestionGeneration

[2] This assumption holds for MC datasets such as SQuAD and NewsQA, but there are exceptions in certain subdomains of MSMARCO.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 835–844
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

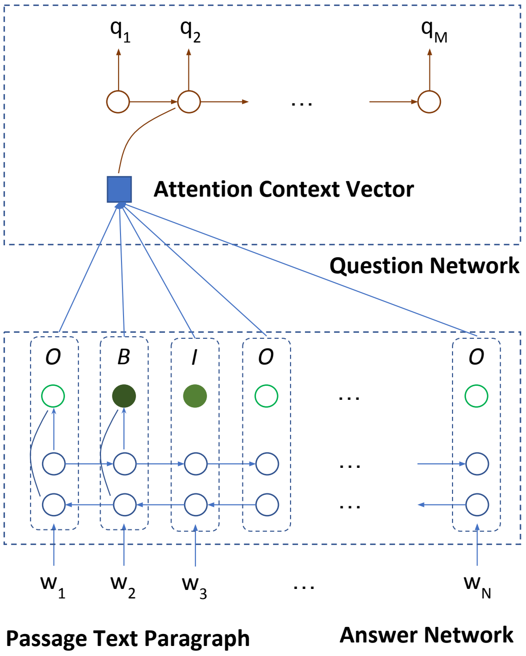


Figure 1: Illustration of the two-stage SynNet. The SynNet is trained to synthesize the answer and the question, given the paragraph. The first stage of the model, an answer synthesis module, uses a bi-directional LSTM to predict IOB tags on the input paragraph, which mark out key semantic concepts that are likely answers. The second stage, a question synthesis module, uses a uni-directional LSTM to generate the question, while attending on embeddings of the words in the paragraph and IOB ids. Although multiple spans in the paragraph could be identified as potential answers, we pick one span when generating the question.

ate to view answers and questions as two different types of data. Hence, the synthesis of a (question, answer) tuple is needed.

In our approach, we decompose the process of generating question-answer pairs into two steps, answer generation conditioned on the paragraph, and question generation conditioned on the paragraph and answer. We generate the answer first because answers are usually key semantic concepts, while questions can be viewed as a full sentence composed to inquire the concept.

Using the proposed SynNet, we are able to outperform a strong baseline of directly applying a high-performing MC model trained on another domain. For example, when we apply our algorithm using a pretrained model on the Stanford Question-Answering Dataset (SQuAD) (Rajpurkar et al., 2016), which consists of Wikipedia articles, to answer questions on the NewsQA dataset (Trischler et al., 2016), which consists of CNN/Daily Mail articles, we improve the performance of the SQuAD baseline from 39.0% to 46.6% F1 and approach results of previously published work of Trischler et al. (2016) (50.0% F1), without use of labeled data in the new domain. Moreover, an error analysis reveals that we achieve higher accuracy over the baseline on all common question types.

2 Related Work

2.1 Question Answering

Question answering is an active area in natural language processing with ongoing research in many directions (Berant et al., 2013; Hill et al., 2015; Golub and He, 2016; Chen et al., 2016; Hermann et al., 2015). Machine comprehension, a form of extractive question answering where the answer is a snippet or multiple snippets of text within a context paragraph, has recently attracted a lot of attention in the community. The rise of large-scale human annotated datasets with over 100,000 realistic question-answer pairs such as SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2016), and MSMARCO (Nguyen et al., 2016), has led to a large number of successful deep learning models (Lee et al., 2016; Seo et al., 2017; Xiong et al., 2017; Dhingra et al., 2017; Wang and Jiang, 2016).

2.2 Semi-Supervised Learning

Semi-supervised learning has a long history (c.f. Chapelle et al. (2009) for an overview), and has been applied to many tasks in natural language processing such as dependency parsing (Koo et al., 2008), sentiment analysis (Yang et al., 2015),machine translation (Sennrich et al., 2016), and semantic parsing (Berant and Liang, 2014; Wang et al., 2015; Jia and Liang, 2016). Recent work generated synthetic annotations on unsupervised data to boost the performance of both reading comprehension and visual question answering models (Yang et al., 2017; Ren et al., 2015), but on domains with some form of annotated data. There has also been work on generating high-quality questions (Yuan et al., 2017; Serban et al., 2016; Labutov et al., 2015), but not how to best use them to train a model. In contrast, we use the two-stage SynNet to generate data tuples to directly boost performance of a model on a domain with no annotations.

2.3 Transfer Learning

Transfer learning (Pan and Yang, 2010) has been successfully applied to numerous domains in machine learning, such as machine translation (Zoph et al., 2016), computer vision, (Sharif Razavian et al., 2014), and speech recognition (Doulaty et al., 2015). Specifically, object recognition models trained on the large-scale ImageNet challenge (Russakovsky et al., 2015) have proven to be excellent feature extractors for diverse tasks such as image captioning (i.e., Lu et al. (2017); Fang et al. (2015); Karpathy and Fei-Fei (2015)) and visual question answering (i.e., Zhou et al. (2015); Xu and Saenko (2016); Fukui et al. (2016); Yang et al. (2016)), among others. In a similar fashion, we use a model pretrained on the SQuAD dataset as a generic feature extractor to bootstrap a QA system on NewsQA.

3 The Transfer Learning Task for MC

We formalize the task of machine comprehension below. Our MC model takes as input a tokenized question $q = \{q_0, q_1, ...q_n\}$, a context paragraph $p = \{p_0, p_1, ...p_n\}$, where q_i, p_i are words, and learns a function $f(p, q) \mapsto \{a_{start}, a_{end}\}$ where a_{start} and a_{end} are pointer indices into paragraph p, i.e., the answer $a = p_{a_{start}}...p_{a_{end}}$.

Given a collection of labeled paragraph, question, answer triples $\{p, q, a\}_{i=1}^{n}$ from a particular domain s, i.e., Wikipedia articles, we can learn a MC model $f_s(p, q)$ that is able to answer questions in that domain.

However, when applying the model trained in one domain to answer questions in another, the performance may degrade. On the other hand, labeling data to train a model in the new domain is expensive and time-consuming.

In this paper, we propose the task of transferring a MC system $f_s(p, q)$ that is trained in a source domain to answer questions over another target domain, t. In the target domain t, we are given an unlabeled set $p_t = \{p\}_{i=1}^{k}$ of k paragraphs. During test time, we are given an unseen set of paragraphs, p^*, in the target domain, over which we would like to answer questions.

4 The Model

4.1 Two-Stage SynNet

To bootstrap our model f_s we use a SynNet (Figure 1), which consists of answer synthesis and question synthesis modules, to generate data on p_t. Our SynNet learns the conditional probability of generating answer $a = \{a_{start}, a_{end}\}$ and question $q = \{q_1, ...q_n\}$ given paragraph p, $P(q, a|p)$. We decompose the joint probability distribution $P(q, a|p)$ into a conditional probability distribution $P(q|p, a)P(a|p)$, where we first generate the answer a, followed by generating the question q conditioned on the answer and paragraph.

4.1.1 Answer Synthesis Module

In our answer synthesis module we train a simple IOB tagger to predict whether each word in the paragraph is part of an answer or not.

More formally, given a set of words in a paragraph $p = \{p_1...p_n\}$, our IOB tagging model learns the conditional probability of labels $y_1...y_n$, where $y_1 \in \text{IOB}_{\text{START}}, \text{IOB}_{\text{MID}}, \text{IOB}_{\text{END}}$ if a word p_i is marked as an answer by the annotator in our train set, NONE otherwise.

We use a bi-directional Long-Short Term Memory Network (Bi-LSTM) (Hochreiter and Schmidhuber, 1997) for tagging. Specifically, we project each word $p_i \mapsto p_i^*$ into a continuous vector space via pretrained GloVe embeddings (Pennington et al., 2014). We then run a Bi-LSTM over the word embeddings $p_1^*, ...p_n^*$ to produce a context-dependent word representation $h_1, ...h_n$, which we feed into two fully connected layers followed by a softmax to produce our tag likelihoods for each word.

We select all consecutive spans where $y \neq$ NONE produced by the tagger as our candidate answer chunks, which we feed into our question synthesis module for question generation.

4.1.2 Question Synthesis Module

Our question synthesis module learns the conditional probability of generating question $q = \{q_1, ...q_n\}$ given answer $a = a_{start}, a_{end}$ and paragraph $p = p_1...p_n$, $P(q_1, ...q_n|p_1...p_n, a_{start}, a_{end})$. We decompose the joint probability distribution of generating all the question words $q_1, ...q_n$ into generating the question one word at a time, i.e. $\prod_{i=1}^{n} P(q_i|p, a, q_{1...i-1})$.

The model is similar to an encoder-decoder network with attention (Bahdanau et al., 2014), which computes the conditional probability $P(q_i|p_1...p_n, a_{start}, a_{end}, q_{1...i-1})$. We run a Bi-LSTM over the paragraph to produce context-dependent word representations $h = \{h_1, ...h_n\}$. To model where the answer is in the paragraph, similar to Yang et al. (2017), we insert answer information by appending a zero/one feature to the paragraph word embeddings. Then, at each time step i, a decoder network attends to both h and the previously generated question token q_{i-1} to produce a hidden representation r_i. Since paragraphs may often have named entities and rare words not present during training, we incorporate a copy mechanism into our models (Gu et al., 2016).

We use an architecture motivated by latent predictor networks (Ling et al., 2016) to force the model to learn when to copy vs. directly predict the word, without direct supervision of what action to choose. Specifically, at every time step i, two latent predictors generate the probability of generating word w_i, a pointer network C_p (Vinyals et al., 2015) which can copy a word from the context paragraph, and a vocabulary predictor V_p which directly generates a probability distribution of choosing a word w_i from a predefined vocabulary. The likelihood of choosing predictor k at time step i is proportional to $w_k r_i$, and the likelihood of predicting question token q_i is given by $q_i^* = p^v l^v(w_i) + (1 - p^v) l^c(w_i)$, where v represents the vocabulary predictor and c represents the copy predictor, and $l(w_i)$ is the likelihood of the word given by the predictor.[3] For training, since no direct supervision is given as to which predictor to choose, we minimize the cross entropy loss of producing the correct question tokens $\sum_{j=1}^{n} -log(q_j^*)$ by marginalizing out latent variables using a variant of the forward-backward algorithm (see Ling et al. (2016) for full details).

During inference, to generate a question $q_1 ... q_n$, we use greedy decoding in the following manner. At time step i, we select the most likely predictor (C_p or V_p), followed by the most likely word q_i given the predictor. We feed the predicted word as input at the next timestep back into the decoder until we predict the end symbol, END, after which we stop decoding.

4.2 Machine Comprehension Model

Our machine comprehension model $f(p, q) \mapsto a$ learns the conditional likelihood of predicting answer pointers $a = \{a_{start}, a_{end}\}$ given paragraph p and question q, $P(a|p, q)$. In our experiments we use the open-source Bi-directional Attention Flow (BiDAF) network (Seo et al., 2017)[4] since it is one of the best-performing models on the SQuAD dataset,[5] although we note that our algorithm for data synthesis can be used with any MC model.

Algorithm 1: Training Algorithm

```
Input  : $x_s = \{p_s, q_s, a_s\}_{i=1}^{n}$ triplets from
         source domain $s$; pretrained MC
         model on $s$,
         $f_s(p, q) \mapsto \{a_{start}, a_{end}\}$;
         paragraphs from target domain $t$,
         $p_{j=1}^{m}$
Output: MC model on target domain,
         $f_t(p, q) \mapsto \{a_{start}, a_{end}\}$
Train SynNet $g$ to maximize $P(q, a|p)$ on
   source $s$;
Generate samples $x_t = (q, a|p)_{i=1}^{k}$ on text in
   target domain $t$;
Use $x_s \cup x_t$ to finetune MC model $f_s$ on
   domain $t$. For every batch sampled from $x_t$,
   sample $k$ batches from $x_s$;
```

4.3 Algorithm Overview

Having given an overview of our SynNet and a brief overview of the MC model we describe our training procedure, which is illustrated in Algorithm 1.

4.4 Training

Our approach for transfer learning consists of several training steps. First, given a series of labeled examples $x_s = \{p_s, q_s, a_s\}_{i=1}^{n}$ from domain s, paragraphs $p_{j=1}^{m}$ from domain t, and pretrained MC model $f_s(p, q)$, we train the SynNet g_s to maximize the likelihood of the question-answer pairs in s.

Second, we fix our SynNet g_s and we sample $x_t = \{p_t, q_t, a_t\}_{i=1}^{k}$ question-answer pairs on the paragraphs in domain t. Several examples of generated questions can be found in Table 1.

We then transfer the MC model originally learned on the source domain to the target domain t using SGD on the synthetic data. However, since the synthetic data is usually noisy, we alternatively train the MC model with mini-batches from x_s and x_t, which we call *data-regularization*. Every k batches from x, we sample 1 batch of synthetic data from x', where k is a hyper-parameter, which we set to 4. Letting the model encounter many examples from source domain s serves to regularize

[3] Since we only have two predictors, $p^c = 1 - p^v$

[4] See https://github.com/allenai/bi-att-flow

[5] See https://rajpurkar.github.io/SQuAD-explorer/ for latest results

Snippet of context paragraph (answer in bold)	Generated questions (bold) vs. human questions
...At this point, some of these used-luxe models have been around so long that they almost qualify as vintage throwback editions. Recently, **Consumer Report** magazine issued its list of best and worst used cars, and divvied them up by price range ...	**What magazine made best used cars in the USAF?** Who released a list of best and worst used cars
...A high court in northern India on Friday acquitted a wealthy businessman facing the death sentence for the killing of a teen in a case dubbed ”the house of horrors.“ Moninder Singh Pandher was sentenced to death by a lower court in February. The teen was **one of 19** victims – children and ...	**How many victims were in India ?** What was the amount of children murdered ?
Joe Pantoliano has met **with the Obama and McCain camps** to promote mental health and recovery. Pantoliano, founder and president of the eight-month-old advocacy organization No Kidding, Me Too, released a teaser of his new film about various forms of mental illness...	**Which two groups did Joe Pantoliano meet with?** Who did he meet with to discuss the issue?
...Former boxing champion Vernon Forrest , 38 , was shot and killed in **southwest Atlanta , Georgia** , on July 25 . A grand jury indicted the three suspects – Charman Sinkfield , 30 ; Demario Ware , 20 ; and Jquante...	**Where was the first person to be shot ?** Where was Forrest killed?

Table 1: Randomly sampled paragraphs and corresponding synthetic vs. human questions from the NewsQA train set. Human-selected answers from the train set were used as input.

the distribution of the synthetic data in the target domain with real data from s. We checkpoint finetuned model f_s^* every i mini-batches, $i = 1000$ in our experiments, and save a copy of the model at each checkpoint.

At test time, to generate an answer, we feed paragraph $p = \{p_0, p_1, ...p_n\}$ and question q through our finetuned MC model $f^*(p, q)$ to get $P(p_i = a_{start})$, $P(p_i = a_{end})$ for all $i \in 1...n$. We then use dynamic programming (Seo et al., 2017) to find the optimal answer span $\{a_{start}, a_{end}\}$. To improve the stability of using our model for inference, we average the predicted answer likelihoods from model copies at different checkpoints prior to running the dynamic programming algorithm.

5 Experimental Setup

We summarize the datasets we use in our experiments, parameters for our model architectures, and training details.

The SQuAD dataset consists of approximately 100,000 question-answer pairs on Wikipedia, 87,600 of which are used for training, 10,570 for development, and an unknown number in a hidden test set. The NewsQA dataset consists of 92,549 train, 5,166 development and 5,165 test questions on CNN/Daily Mail news articles. Both the domain type (i.e., news) and question types differ between the two datasets. For example, an analysis of a randomly generated sample of 1,000 questions from both NewsQA and SQuAD (Trischler et al., 2016) reveals that approximately 74.1% of questions in SQuAD require word matching or paraphrasing to retrieve the answer, as opposed to 59.7% in NewsQA. As our test metrics, we report two numbers, exact match (EM) and F1 score.

We train a BIDAF model on the SQuAD train dataset and use a two-stage SynNet to finetune it on the NewsQA train dataset.

We initialize word-embeddings for the BIDAF model, answer synthesis module, and question synthesis module with 300-dimensional-GloVe vectors (Pennington et al., 2014) trained on the 840 Billion Words Common Crawl corpus. We set all embeddings of unknown word tokens to zero.

For both the answer synthesis and question synthesis module, we use a vocabulary of size 110,179. We use LSTMs with hidden states of size 150 for the answer module vs. those of size 100 for the question module since the answer module is less memory intensive than the question module.

We train both the answer and question module with Adam (Kingma and Ba, 2015) and a learning rate of 1e-2. We train a BIDAF model with the default hyperparameters provided in the open-source repository. To stop training of the question synthesis module, after each epoch, we monitor both the loss as well as the quality of questions generated on the SQuAD development set. To stop training of the answer synthesis module, we similarly monitor predictions on the SQuAD development set.

To train the question synthesis module, we only use the questions provided in the SQuAD train set. However, to train the answer synthesis module, we further augment the human-annotated labels of each paragraph with tags from a simple NER system[6] because labels of answers provided in the train set are underspecified, i.e., many words in the paragraph that could be potential answers are not labeled. Therefore, we assume any named entities could also be potential answers of certain questions, in addition to the answers explicitly labeled by annotators.

To generate question-answer pairs on the NewsQA train set using the SynNet, we first run every paragraph through our answer synthesis module. We then randomly sample up to 30 candidate answers extracted by our module, which we feed into the question synthesis module. This results in 250,000 synthetic question-answer pairs that we can use to finetune our MC model.

6 Experimental Results

We report the main results on the NewsQA test set (Table 2), report brief results on SQuAD (Table 3), conduct ablation studies (Table 4), and conduct an error analysis.

6.1 Results

We compare to the best previously published work, which trains BARB (Trischler et al., 2016) and Match-LSTM (Wang and Jiang, 2016) architectures, and a BIDAF model we train on NewsQA. Directly applying a BIDAF model trained on SQuAD to predict on NewsQA leads to poor performance with an F1 measure of 39.0%, 13.2% lower than one trained on labeled NewsQA data. Using the 2-stage SynNet already leads to a slight boost in performance (F1 measure of 44.3%), which implies that having exposure to the new domain via question-answer pairs provides important signal for the model during training. When we augment the answers from our answer synthesis module with those from a generic NER system to produce questions, we have an additional 2.3% performance boost. Finally, when we ensemble with the original model, we boost the EM further by 0.2%. Our final system achieves an F1 measure of 46.6%, approaching previously published results of 50.0%. The results demonstrate that using the proposed architecture and training procedure, we can transfer a MC model from one domain to another, without use of annotated data.

We also evaluate the SynNet on the NewsQA-to-SQuAD direction. We directly apply the best setting from the other direction and report the result in Table 3. The SynNet improves over the baseline by 1.6% in EM and 0.7% in F1. Limited by space, we leave out ablation studies in this direction.

6.2 Ablation Studies

To better understand how various components in our training procedure and model impact overall performance we conduct several ablation studies, as summarized in Table 4.

6.2.1 Answer Synthesis

We experiment with using the answer chunks given in the train set, A_{oracle}, to generate synthetic questions, versus those from an NER system, A_{ner}. Results in Table 4(A) show that using human-annotated answers to generate questions leads to a significant performance boost over using answers from an answer generation module. This supports the hypothesis that the answers humans choose to generate questions for provide important linguistic cues for finetuning the machine comprehension model.

6.2.2 Question Synthesis

To see how copying impacts performance, we explore using the entire paragraph to generate the question vs. only the two sentences before and one sentence after the answer span and report results in Table 4(B). On the NewsQA train set, synthetic questions that use 2 sentences contain an average of 3.0 context words within 10 words to the left and right of the answer chunk, those that use the entire context have 2.1 context words, and human generated questions only have 1.7 words. Training with generated questions that have a large amount of overlap with words close to the answer span (i.e., those that use 2-sentences vs. entire context for generation) leads to models that perform worse, especially with synthetic answer spans and no data regularization (35.6% F1 vs. 34.3% F1). One possible reason is that, according to analysis in Trischler et al. (2016), significantly more questions in the NewsQA dataset re-

[6] https://spacy.io/

Method	System	EM	F1
Transfer Learning	M_{sq} (baseline)	24.9	39.0
	M_{sq} + A_{gen} + Q_{gen}	30.6	44.3
	M_{sq} + A_{gen} + A_{ner} + Q_{gen}	32.8	46.6
	M_{sq} + A_{gen} + A_{ner} + Q_{gen} + M^*_{sq}	**33.0**	**46.6**
Supervised Learning	Barb-LSTM on NewsQA (Trischler et al., 2016)	34.9	50.0
	Match-LSTM on NewsQA (Trischler et al., 2016)	34.1	48.2
	BIDAF on NewsQA	37.1	52.3
	BIDAF on SQuAD finetuned on NewsQA	37.3	52.2

Table 2: **Main Results**. Exact match (EM) and span F1 scores on the NewsQA test set of a BIDAF model finetuned with our SynNet. M_{sq} refers to a baseline BIDAF model trained on SQuAD, A_{gen}, Q_{gen} refers to using answers generated from our SynNet respectively to finetune the model on NewsQA, A_{ner} refers to using answers extracted from a standard NER system to generate questions. M^*_{sq} refers to using the baseline SQUAD model in the ensemble.

System	EM	F1
M_{newsqa}	46.3	60.8
M_{newsqa} + S_{net}	47.9	61.5

Table 3: **NewsQA to SQuAD**. Exact match (EM) and span F1 results on SQuAD development set of a NewsQA BIDAF model baseline vs. one finetuned on SQuAD using the data generated by a 2-stage SynNet (S_{net}).

quire paraphrase, inference, and synthesis as opposed to word-matching.

6.2.3 Model Finetuning

To see how the quantity of synthetic questions encountered during training impacts performance, we use $k = \{0, 2, 4\}$ mini-batches from SQuAD for every synthetic mini-batch from NewsQA to finetune our model, and average the prediction of 4 checkpointed models during testing. As we see from the results, letting the model to encounter data from human annotations, although from another domain, serves as a key form of data-regularization, yielding consistent improvement as k increases. We hypothesize this is because the data distribution of machine-generated questions is different than human-annotated ones; our batching scheme provides a simple way to prevent overfitting to this distribution.

A)	EM	F1	B)	EM	F1
k=0	27.2	40.5	2s + A_{ner}	22.8	36.1
k=2	29.8	43.9	all + A_{ner}	27.2	40.5
k=4	30.4	44.3	2s + A_{oracle}	31.3	45.2
			all + A_{oracle}	32.5	46.8

Table 4: **Ablation Studies**. Exact match (EM) and span F1 results on NewsQA test set of a BIDAF model finetuned with a 2-stage SynNet. In study A, we vary k, the number of mini-batches from SQuAD for every batch in NewsQA. In study B, we set $k = 0$, and vary the answer type and how much of the paragraph we use for question synthesis. $2 - sent$ refers to using two sentences before answer span, while all refers to using the entire paragraph. A_{ner} refers to using an NER system and A_{or} refers to using the human-annotated answers to generate questions.

6.3 Error Analysis

In this section we provide a qualitative analysis of some of our components to help guide further research in this task.

6.3.1 Answer Synthesis

We randomly sample and present a paragraph with answers extracted by our answer synthesis module (Tables 5 and 6). Although the module appears to have high precision, i.e., it picks up entities such as the "Atlantic Paranormal Society", it misses clear entities such as "David Schrader", which suggests training a system with full NER/POS tags as la-

They are ghost hunters , or , as they prefer to be called , paranormal investigators . “ Ghost-Hunters ”, which airs a special live show at **7 p.m. Halloween night ,** is helping lift the stigma once attached to paranormal investigators . The show has become so popular that the group featured in each episode – **Atlantic Paranormal Society** - has spawned imitators across **United States** and affiliates in **countries** . TAPS , as the “ **Hunters”** group is informally known , even has its own “ **Reality Radio”** show , magazine , lecture tours , T-shirts – and groupies . “ **Hunters”** has made creepy cool , says David Schrader , a paranormal investigator and co-host of “ **Radio** ”, a radio show that investigates paranormal activity.

Table 5: Sample predictions from our answer synthesis module.

What is Oklahoma’s unemployment rate until Oklahoma City ?
What was the manager of the Oklahoma City agency ?
How many companies are in Oklahoma City ?
How many workers may Oklahoma have as fair hold ?
Who said the bureau has already hired civilians to choose
What was the average hour manager of Oklahoma City ?
How much would Oklahoma have a year to be held
What year did Oklahoma ’s census build job industry ?

Table 6: Predictions from the question synthesis module on a subset of a paragraph.

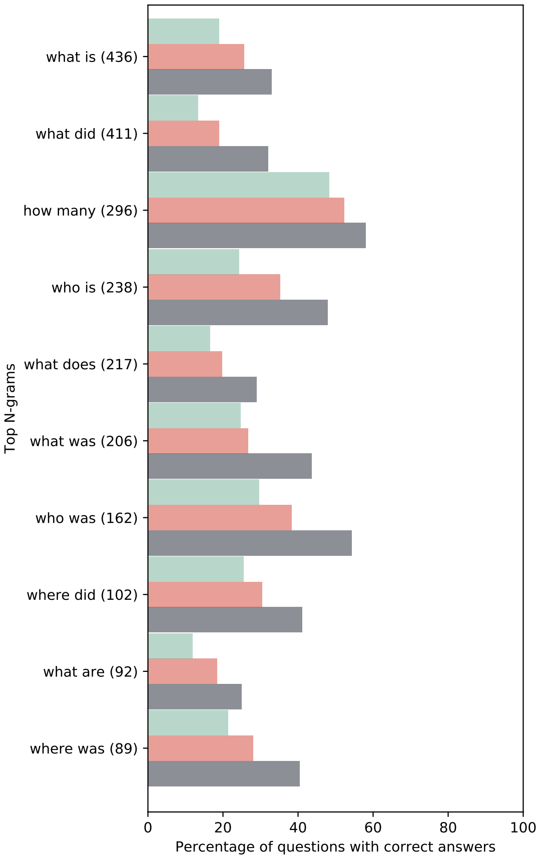


Figure 2: NewsQA accuracy of baseline BIDAF model trained on SQuAD (light green), vs. model finetuned with our method (red) vs. one trained from scratch on NewsQA (dark grey).

bels would yield better results, and also explains why augmenting synthetic data generated by SynNet with such tags leads to improved performance.

6.3.2 Question Synthesis

We randomly sample synthetic questions generated by our module and present our results in Table 6. Due to the copy mechanism, our module has the tendency to directly use many words from the paragraph, especially common entities, such as “Oklahoma” in the example. Thus, one way to generate higher-quality questions may be to introduce a cost function that promotes diversity during decoding, especially within a single paragraph. In turn, this would expose the RC model to a larger variety of training examples in the new domain, which can lead to better performance.

6.3.3 Machine Comprehension Model

We examine the performance over various question types of a finetuned BIDAF on NewsQA vs. one trained on NewsQA vs. one trained on SQuAD (Figure 2). Finetuning with SynNet improves performance over all question types given, with the largest performance boost on location and person-identification questions. Similarly, models trained on synthetic questions tend to approach in-domain performance on numeric and person-identification questions, but still struggle with questions that require higher-order reasoning, i.e. those starting with “what was” or “what did”. Designing a question generator that explicitly requires such reasoning may be one way to further bridge the gap in performance.

7 Conclusion

We introduce a two-stage SynNet for the task of transfer learning for machine comprehension, a task which is both challenging and of practical importance. With our network and a simple training algorithm where we generate synthetic question-answer pairs on the target domain, we are able to generalize a MC model from one domain to another with no annotated data. We present strong results on the NewsQA test set, improving performance of a baseline BIDAF model by over 7.6% F1. Through ablation studies and error analysis, we provide insights into our methodology on the

SynNet and MC models that can help guide further research in this task.

Acknowledgments

We would like to thank Yejin Choi and Luke Zettlemoyer for helpful discussions concerning this work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods for Natural Language Processing (EMNLP)*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning. *IEEE Transactions on Neural Networks* .

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Association for Computational Linguistics (ACL)*.

Bhuwan Dhingra, Hanxiao Liu, William W Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Association for Computational Linguistics (ACL)*.

Mortaza Doulaty, Oscar Saz, and Thomas Hain. 2015. Data-selective transfer learning for multi-domain speech recognition. *arXiv preprint arXiv:1509.02409* .

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, and John C Platt. 2015. From captions to visual concepts and back. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847* .

David Golub and Xiaodong He. 2016. Character-level question answering with attention. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Association for Computational Linguistics (ACL)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The Goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301* .

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* .

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Terry Koo, Xavier Carreras Pérez, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Association for Computational Linguistics (ACL)*.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *Association for Computational Linguistics (ACL)*.

Kenton Lee, Tom Kwiatkowski, Ankur Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436* .

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. In *Association for Computational Linguistics (ACL)*.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* .

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* .

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods for Natural Language Processing (EMNLP)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods for Natural Language Processing (EMNLP)*.

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems (NIPS)*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. In *International Journal of Computer Vision (IJCV)*. Springer.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Association for Computational Linguistics (ACL)*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR)*.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus. In *Association for Computational Linguistics (ACL)*.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. NewsQA: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830* .

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-LSTM and answer pointer. *arXiv preprint arXiv:1608.07905* .

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL)*.

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. In *Association for Computational Linguistics (ACL)*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *International Conference on Learning Representations (ICLR)*.

Huijuan Xu and Kate Saenko. 2016. Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *European Conference on Computer Vision (ECCV)*.

Min Yang, Wenting Tu, Ziyu Lu, Wenpeng Yin, and Kam-Pui Chow. 2015. LCCT: a semisupervised model for sentiment classification. In *North American Chapter of the Association for Computational Linguistics (NAACL)*.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W Cohen. 2017. Semi-supervised QA with generative domain-adaptive nets. In *Association for Computational Linguistics (ACL)*.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2016. Stacked attention networks for image question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. *arXiv preprint arXiv:1705.02012* .

Bolei Zhou, Yuandong Tian, Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. 2015. Simple baseline for visual question answering. *arXiv preprint arXiv:1512.02167* .

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Association for Computational Linguistics (ACL)*.

Deep Neural Solver for Math Word Problems

Yan Wang   Xiaojiang Liu   Shuming Shi
Tencent AI Lab
{brandenwang, kieranliu, shumingshi}@tencent.com

Abstract

This paper presents a deep neural solver to automatically solve math word problems. In contrast to previous statistical learning approaches, we directly translate math word problems to equation templates using a recurrent neural network (RNN) model, without sophisticated feature engineering. We further design a hybrid model that combines the RNN model and a similarity-based retrieval model to achieve additional performance improvement. Experiments conducted on a large dataset show that the RNN model and the hybrid model significantly outperform state-of-the-art statistical learning methods for math word problem solving.

1 Introduction

Developing computer models to automatically solve math word problems has been an interest of NLP researchers since 1963 Feigenbaum et al. (1963); Bobrow (1964); Briars and Larkin (1984); Fletcher (1985). Recently, machine learning techniques Kushman et al. (2014); Amnueypornsakul and Bhat (2014); Zhou et al. (2015); Mitra and Baral (2016) and semantic parsing methods Shi et al. (2015); Koncel-Kedziorski et al. (2015) are proposed to tackle this problem and promising results are reported on some datasets. Although progress has been made in this task, performance of state-of-the-art techniques is still quite low on large datasets having diverse problem types Huang et al. (2016).

Problem: Dan have 2 pens, Jessica have 4 pens. How many pens do they have in total ? **Equation**: x = 4+2
Solution: 6

Table 1: A math word problem

A typical math word problems are shown in Table 1. The reader is asked to infer how many pens Dan and Jessica have, based on constraints provided. Given the success of deep neural networks (DNN) on many NLP tasks (like POS tagging, syntactic parsing, and machine translation), it may be interesting to study whether DNN could also help math word problem solving. In this paper, we propose a recurrent neural network (RNN) model for automatic math word problem solving. It is a sequence to sequence (seq2seq) model that transforms natural language sentences in math word problems to mathematical equations. Experiments conducted on a large dataset show that the RNN model significantly outperforms state-of-the-art statistical learning approaches.

Since it has been demonstrated Huang et al. (2016) that a simple similarity based method performs as well as more sophisticated statistical learning approaches on large datasets, we implement a similarity-based retrieval model and compare with our seq2seq model. We observe that although seq2seq performs better on average, the retrieval model is able to correctly solve many problems for which RNN generates wrong results. We also find that the accuracy of the retrieval model positively correlate with the maximal similarity score between the target problem and the problems in training data: the larger the similarity score, the higher the average accuracy is.

Inspired by these observations, we design a hybrid model which combines the seq2seq model and the retrieval model. In the hybrid model, the retrieval model is chosen if the maximal similarity score returned by the retrieval model is larger than a threshold, otherwise the seq2seq model is selected to solve the problem. Experiments on our

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 845–854
Copenhagen, Denmark, September 7–11, 2017.

dataset show that, by introducing the hybrid model, the accuracy increases from 58.1% to 64.7%.

Our contributions are as follows:

1) To the best of our knowledge, this is the first work of using DNN technology for automatic math word problem solving.

2) We propose a hybrid model where a seq2seq model and a similarity-based retrieval model are combined to achieve further performance improvement.

3) A large dataset is constructed for facilitating the study of automatic math problem solving.[1]

The remaining part of this paper is organized as follows: After analyzing related work in Section 2, we formalize the problem and introduce our dataset in Section 3. We present our RNN-based seq2seq model in Section 4, and the hybrid model in Section 5. Then experimental results are shown and analyzed in Section 6. Finally we conclude the paper in Section 7.

2 Related work

2.1 Math Word Problems Solving

Previous work on automatic math word problem solving falls into two categories: symbolic approaches and statistical learning approaches.

In 1964, STUDENT Bobrow (1964) handles algebraic problems by two steps: first, they transform natural language sentences into kernel sentences using a small set of transformation patterns. Then the kernel sentences are transformed to mathematical expressions by pattern matching. A similar approach is also used to solve English rate problems Charniak (1968, 1969). Liguda and Pfeiffer Liguda and Pfeiffer (2012) propose modeling math word problems with augmented semantic networks. In addition, Addition/subtraction problems are studied most Briars and Larkin (1984); Dellarosa (1986); Bakman (2007); Yuhui et al. (2010); Roy et al. (2015).

In 2015, Shi et.al Shi et al. (2015) propose a system SigmaDolphin which automatically solves math word problems by semantic parsing and reasoning. In the same year, Koncel et.al Koncel-Kedziorski et al. (2015) also formalizes the problem of solving multi-sentence algebraic word problems as that of generating and scoring equation trees.

Since 2014, statistical learning based approaches are proposed to solve the math word problems. Hosseini et al. Hosseini et al. (2014) deal with the open-domain aspect of algebraic word problems by learning verb categorization from training data. Kushman et al. Kushman et al. (2014) proposed a equation template system to solve a wide range of algebra word problems. Zhou et al. Zhou et al. (2015) further extends this method by adopting the max-margin objective, which results in higher accuracy and lower time cost. In addition, Roy and Roth Roy et al. (2015); Roy and Roth (2016) tries to handle arithmetic problems with multiple steps and operations without depending on additional annotations or predefined templates. Mitra et al. Mitra and Baral (2016) presents a novel method to learn to use formulas to solve simple addition-subtraction arithmetic problems.

As reported in 2016 Huang et al. (2016), state-of-the-art approaches have extremely low performance on a big and highly diverse data set (18,000+ problems). In contrast to these approaches, we study the feasibility of applying deep learning to the task of math word problem solving.

2.2 Sequence to Sequence (seq2seq) Learning

With the framework of seq2seq learning Sutskever et al. (2014); Wiseman and Rush (2016), recent advances in neural machine translation (NMT) Bahdanau et al. (2014); Cho et al. (2014) and neural responding machine (NRM) Shang et al. (2015) have demonstrated the power of recurrent neural networks (RNNs) at capturing and translating natural language semantics. The NMT and NRM models are purely data-driven and directly learn to converse from end-to-end conversational corpora.

Recently, the task of translating natural language queries into regular expressions is explored by using a seq2seq model Locascio et al. (2016), which achieves a performance gain of 19.6% over previous state-of-the-art models. To our knowledge, we are the first to apply seq2seq model to the task of math word problem solving.

3 Problem Formulation and Dataset

3.1 Problem Formulation

A math word problem P is a word sequence W_p and contains a set of variables $V_p = \{v_1, \dots, v_m, x_1, \dots, x_k\}$ where $v_1, \dots, v_m$ are known numbers in P and $x_1, \dots, x_k$ are variables

[1] We plan to make the dataset publicly available when the paper is published

Problem: Dan have 5 pens and 3 pencils, Jessica have 4 more pens and 2 less pencils than him. How many pens and pencils do Jessica have in total?
Equation: x = 5 + 4 +3 -2
Solution: 10

Table 2: A math word problem

whose values are unknown. A problem P can be solved by a mathematical equation E_p formed by V_p and mathematical operators.

In math word problems, different equations may belong to a same equation template. For example, equation $x = (9 * 3) + 7$ and equation $x = (4 * 5) + 2$ share the same equation template $x = (n_1 * n_2) + n_3$. To decrease the diversity of equations, we map each equation to an equation template T_p through a number mapping M_p. The number mapping process can be defined as:

Definition 1 *Number mapping: For a problem P with m known numbers, a number mapping M_p maps the numbers in problem P to a list of number tokens $\{n_1, \ldots, n_m\}$ by their order in the problem text.*

Definition 2 *Equation template: A general form of equations. For a problem P with equation E_p and number mapping M_p, its equation template is obtained by mapping numbers in E_p to a list of number tokens $\{n_1, \ldots, n_m\}$ according to M_p.*

Take the problem in Table 2 as an example, first we can obtain a number mapping from the problem:

$$M : \{n_1 = 5; \quad n_2 = 3; \quad n_3 = 4; \quad n_4 = 2; \}$$

and then the given equation can be expressed as an equation template:

$$x = n_1 + n_3 + n_2 - n_4$$

After number mapping, the problem in Table 2 can be mapped to:

"Dan have n_1 pens and n_2 pencils, Jessica have n_3 more pens and n_4 less pencils than him. How many pens and pencils do Jessica have in total?"

We solve math word problems by generating equation templates through a seq2seq model. The input of the seq2seq model is the sequence W_P after number mapping, and the output is an equation template T_P. The equation E_P can be obtained by applying the corresponding number mapping M_P to T_P.

3.2 Constructing a Large Dataset

Most public datasets for automatic math word problem solving are quite small and contains limited types of problems. The most frequently used Alg514 (Kushman et al., 2014) dataset contains only 514 linear algebra problems with 28 equation templates. There are 1,000 problems in the newly constructed DRAW-1K (Shyam and Ming-Wei, 2017) dataset. Dophin1878 (Shi et al., 2015) includes 1,878 number word problems. An exception is the Dolphin18K dataset (Huang et al., 2016) which contains 18,000+ problems. However, this dataset has not been made publicly available so far.

Since DNN-based approaches typically need large training data, we have to build a large dataset of labeled math word problems. We crawl over 60,000 Chinese math word problems from a couple of online education web sites. All of them are real math word problems for elementary school students. We focus on one-unknown-variable linear math word problems in this paper. For other problem types, we would like to leave as future work. Please pay attention that the solutions to the problems are in natural language, and we have to extract equation systems and structured answers from the solution text. We implement a rule-based extraction method for this purpose, which achieves very high precision and medium recall. That is, most equations and structured answers extracted by our method are correct, and many problems are dropped from the dataset. As a result, we get dataset Math23k which contains 23,161 problems labeled with structured equations and answers. Please refer to Table 3 for some statistics of the dataset and a comparison with other public datasets.

4 Deep Neural Solver

In this section, we propose a *RNN-based seq2seq model* to translate problem text to math equations. Since not all numbers in problem text may be useful for solving the problem, we propose, in Section 4.2, a *significant number identification* model to distinguish whether a number in a problem should appear in the corresponding equations.

4.1 RNN based Seq2seq Model

Figure 1 shows our RNN-based seq2seq model for transforming problem text to a math equation, using the problem in Table 2 as an example. The in-

dataset	# problems	# templates	# sentences	# words	problem types
Alg514	514	28	1.62k	19.3k	algebra, linear
Dolphin1878	1,878	1,183	3.30k	41.4k	number word problems
DRAW-1K	1,000	Unknown	6.23k	81.5k	algebra, linear, one-VAR
Math23K	23,161	2,187	70.1k	822k	algebra, linear, one-VAR

Table 3: Statistics of our dataset and several publicly available datasets

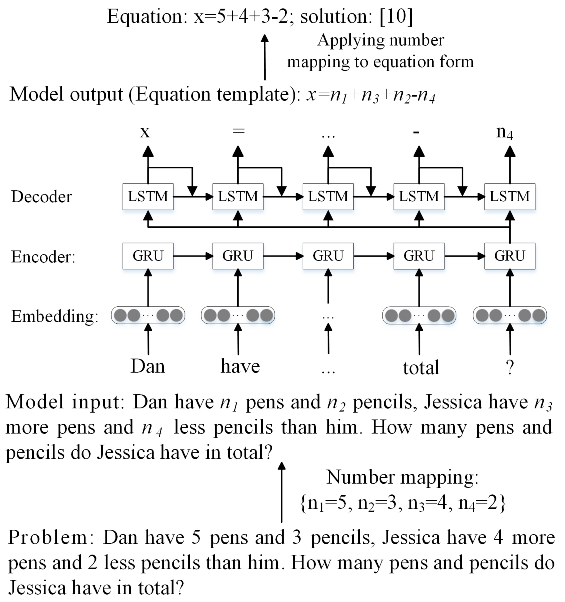

Equation: x=5+4+3-2; solution: [10]
Applying number mapping to equation form
Model output (Equation template): x=n1+n3+n2-n4
x
=
...
-
n4
Decoder
LSTM
LSTM
LSTM
LSTM
LSTM
Encoder:
GRU
GRU
GRU
GRU
GRU
Embedding:
...
Dan
have
...
total
?
Model input: Dan have n1 pens and n2 pencils, Jessica have n3 more pens and n4 less pencils than him. How many pens and pencils do Jessica have in total?
Number mapping: {n1=5, n2=3, n3=4, n4=2}
Problem: Dan have 5 pens and 3 pencils, Jessica have 4 more pens and 2 less pencils than him. How many pens and pencils do Jessica have in total?

Figure 1: The seq2seq model

put sequence W is the problem after number mapping:

"Dan have n_1 pens and n_2 pencils, Jessica have n_3 more pens and n_4 less pencils than him. How many pens and pencils do Jessica have in total?"

The output sequence $R = \{r_1, \ldots, r_s\}$ is the equation template:

$$x = n_1 + n_3 + n_2 - n_4$$

The gated recurrent units (GRU) (Chung et al., 2014) and long short-memory (LSTM) (Hochreiter and Schmidhuber, 1997) cells are used for encoding and decoding, respectively. The reason why we use GRU as the encoder instead of LSTM is that the GRU has less parameters and less likely to be overfitted on small dataset. Four fundamental operational stages of GRU are as follows:

$$\begin{aligned}
z_t &= \sigma(W^{(z)} x_t + U^z h_{t-1}) && \text{(Update gate)}\\
r_t &= \sigma(W^{(r)} x_t + U^r h_{t-1}) && \text{(Reset gate)}\\
\hat{h_t} &= tanh(r_t \odot U h_{t-1} + W x_t) && \text{(New memory)}\\
h_t &= (1 - z_t) \odot \hat{h_t} + z_t \odot h_{t-1} && \text{(Hidden state)}
\end{aligned} \tag{1}$$

where σ represents the sigmoid function and $\odot$ is an element-wise multiplication. The input x_t is a word w_t along with previously generated character r_{t-1} . The variables U and W are weight matrices for each gate.

The fundamental operational stages of LSTM are as follows:

$$\begin{aligned}
i_t &= \sigma(W^{(i)} x_t + U^i h_{t-1}) && \text{(Input gate)}\\
f_t &= \sigma(W^{(f)} x_t + U^f h_{t-1}) && \text{(Forget gate)}\\
o_t &= \sigma(W^{(o)} x_t + U^o h_{t-1}) && \text{(Output gate)}\\
\tilde{c}_t &= tanh(W^{(c)} x_t + U^{(c)} h_{t-1}) && \text{(New memory)}\\
c_t &= f_t \odot \tilde{c}_{t-1} + i_t \odot \tilde{c}_t && \text{(Final memory)}\\
h_t &= o_t \odot tanh(c_t) && \text{(Hidden state)}
\end{aligned} \tag{2}$$

where the input x_t is a word w_t along with previously generated character r_{t-1} .

Then, we redesigned the activation function of the seq2seq model, which is different from vanilla seq2seq models. If we directly generate equation templates by a softmax function, some incorrect equations may be generated, such as: "$x = n_1 + + * n_2$" and "$x = (n_1 * n_2$". To ensure that the output equations are mathematically correct, we need to find out which characters are illegal according to previously generated characters. This is done by five predefined rules like:

- Rule 1: If r_{t-1} in $\{+, -, *, /\}$, then r_t will not in $\{+, -, *, /,), =\}$;
- Rule 2: If r_{t-1} is a number, then r_t will not be a number and not in $\{(, =\}$;

- Rule 3: If r_{t-1} is "=", then r_t will not in $\{+, -, *, /, =,)\}$;
- Rule 4: If r_{t-1} is "(", then r_t will not in $\{(,), +, -, *, /, =\}$;
- Rule 5: If r_{t-1} is ")", then r_t will not be a number and not in $\{(,)\}$;

A binary vector ρ_t can be generated depends on r_{t-1} and these rules. Each position in ρ_t is corresponding to a character in the output vocabulary, where "1" represents that the character is mathematically correct, and "0" indicates mathematically incorrect. Thus, the output probability distribution at each time-step t can be calculated as:

$$P(\hat{r_t}|h_t) = \frac{\rho_t \odot e^{h_t^T W^s}}{\sum \rho_t \odot e^{h_t^T W^s}} \tag{3}$$

where h_t is the output of LSTM decoder, and W^s is the weight matrix. The probability of mathematically incorrect characters will be 0.

Our model is five layers deep, with a word embedding layer, a two-layer GRU as encoder and a two-layer LSTM as decoder. Both the encoder and decoder contain 512 nodes. We perform standard dropout during training (Srivastava et al., 2014) after GRU and LSTM layer with dropout probability equal to 0.5. We train for 80 epochs, utilizing a mini-batch size of 256 and a learning-rate of 0.01.

4.2 Significant Number Identification (SNI)

In a math word problem, not all numbers appear in the equation for solving the problem. An example is shown in Table 4, where the number *"1"* in *"1 day, 1 girl"* and number *"2"* in *"She has 2 types of"* should not be used in equation construction. We say a number is *significant* if the number should be included in the equation to the problem; otherwise it is *insignificant*. For the problem in Table 4, significant numbers are 9, 3, and 5, while 1 and 2 are insignificant numbers. Identifying significant and insignificant numbers are important for constructing correct equations. For this purpose, we build a LSTM-based binary classification model to determine whether a number in a piece of problem text is significant.

The training data for SNI model are extracted from the math word problems. Each number and its context in problems is a training instance of SNI. An instance will be labelled"True" if the number is *significant*, otherwise it will be labelled "False". The structure of SNI model is shown in Figure 2. By using single layer LSTMs with 128 nodes and a symmetric window of length 3, our model achieves 99.1% accuracy. Table 4 is an example of number mapping with and without SNI.

Problem: 1 day, 1 girl was organizing her book case making sure each of the shelves had exactly 9 books on it. She has 2 types of books - mystery books and picture books. If she had 3 shelves of mystery books and 5 shelves of picture books, how many books did she have in total?
Number mapping: $n_1 = 1; n_2 = 1; n_3 = 9;$ $n_4 = 2; n_5 = 3; n_6 = 5$
Equation template: $x = n_5 * n_3 + n_6 * n_3$
Number mapping with SNI: $n_1 = 9; n_2 = 3; n_3 = 5$
Equation template with SNI: $x = n_2 * n_1 + n_3 * n_1$
Problem after number mapping and SNI: 1 day, 1 girl was organizing her book case making sure each of the shelves had exactly n_1 books on it. She has 2 types of books -mystery books and picture books. If she had n_2 shelves of mystery books and n_3 shelves of picture books, how many books did she have in total?

Table 4: Significant number identification (SNI) example

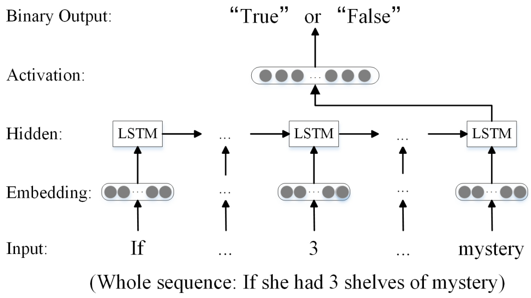


Figure 2: The significant number identification model

5 Hybrid Model

To compare the performance of our deep neural solver and traditional statistical learning methods, we implement a similarity-based retrieval model (refer to Section 5.1 for more details).

The Venn diagram in Figure 3 shows the relationship between the problems solved by the re-

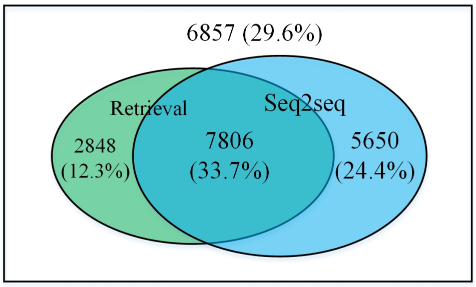


Figure 3: **Green area**: problems correctly solved by the retrieval model; **Blue area**: problems correctly solved by the seq2seq model; **Overlapped area**: problems correctly solved by both models; **White area**: problems that both models fail to solve

trieval model and those solved by the seq2seq model. We can see that although seq2seq performs better on average, the retrieval model is able to correctly solve many problems that seq2seq cannot solve. If we can combine the two models properly to build a hybrid model, more problems may get solved.

In this section, we first give some details about the retrieval model in Section 5.1, then the hybrid model is introduced in Section 5.2.

5.1 Retrieval Model

The retrieval model solves problems by calculating the lexical similarity between the testing problem and each problem in the training data, and then the equation template of the most similar problem is applied to the testing problem. Each problem is modeled as a vector of word TF-IDF scores $W = [w_{1,d}, w_{2,d}, \ldots, w_{N,d}]^T$, where

$$w_{t,d} = tf_{t,d} * \frac{|D|}{|d \in D | t \in d|} \tag{4}$$

and $tf_{t,d}$ is the word frequency of word t in problem d; $|D|$ is the total number of problems in dataset; $|d \in D | t \in d|$ is the number of documents containing the word t.

The similarity between the testing problem P_T and another problem Q can be calculated by the Jaccard similarity between their corresponding vectors:

$$J(P_T, Q) = \frac{|P_T \cap Q|}{|P_T \cup Q|} = \frac{|P_T \cap Q|}{|P_T| + |Q| - |P_T \cap Q|} \tag{5}$$

The retrieval model will choose training problem Q_1 that have the maximal similarity with P_T and use the equation template T of Q_1 as the template of problem P_T.

An important and interesting observation about the retrieval model is the relation between the maximal similarity and solution accuracy. Figure 4 shows the results of only considering the problems for those the maximal similarity returned by retrieval model is above a threshold θ (in other words, we skip a problem if its corresponding maximal similarity is below the threshold). It is clear that the larger the similarity score, the higher the average accuracy is. In our hybrid model, we make use of this property to combine the seq2seq model and the retrieval model.

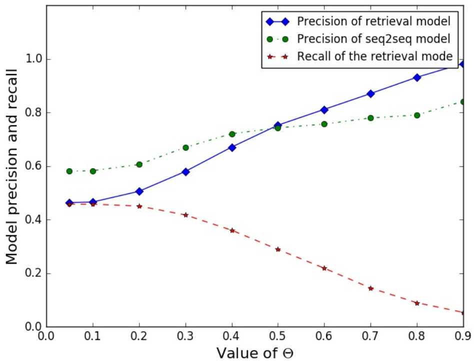


Figure 4: Precision and recall of the retrieval model, and the precision of the seq2seq model w.r.t. different similarity threshold (θ) values

5.2 Hybrid Model

Our hybrid model combines the retrieval model and the seq2seq model by setting a hyper-parameter θ as the threshold of similarity. In algorithm 1, if the Jaccard similarity between testing problem P_T and the retrieved problem Q_1 is higher than θ, the model will choose the equation template T of Q_1 as the equation template of problem P_T. Otherwise an equation template will be generated by a seq2seq model. As shown in Figure 4, the retrieval model has a higher precision than the seq2seq model when we set a high threshold.

6 Experiments

In this section, we conduct experiments on two datasets to examine the performance of the proposed models. Our main experimental result is to show a significant improvement over the baseline

Algorithm 1 Hybrid model

```
Input: $Q$: problems in training data;
       $P_T$: testing problem;
       $\theta$: pre-defined threshold of similarity
Output: Problem solution
Get equation templates and number mappings for training problems $Q$ and testing problem $P_T$.
Number identification: identify significant numbers
Retrieval:
choose problem $Q_1$ from $Q$ that has the maximal Jaccard similarity with $P_T$
if $J(P_T, Q_1) > \theta$ then
    Apply the retrieval model: select equation template $T$ of $Q_1$
else
    Apply the seq2seq model: $T = seq2seq(P_T)$
end if
Applying number mappings of $P_T$ to $T$ and calculating final solution
```

	Math23K	Alg514
ZDC	42.1%	**79.7%**
Retrieval model w/o SNI	46.0%	70.1%
Retrieval model w/ SNI	47.2%	70.1%
Seq2seq model w/o SNI	53.7%	17.2%
Seq2seq model w/ SNI	58.1%	16.1%
Hybrid model w/o SNI	61.1%	70.1%
Hybrid model w/ SNI	**64.7%**	70.1%

Table 5: Model comparison (average accuracy of 5-fold cross validation)

	ZDC	R	R(S)	Seq	Seq(S)	H
R(S)	≫	>				
Seq	≫	≫	≫			
Seq(S)	≫	≫	≫	≫		
H	≫	≫	≫	≫	≫	
H(S)	≫	≫	≫	≫	≫	≫

Table 6: Result of significance test. The meaning of abbreviations in this table is as follows: R: retrieval model w/o SNI; R(S): retrieval model w/ SNI; Seq: seq2seq model w/o SNI; Seq(S): seq2seq model w/ SNI; H: hybrid model w/o SNI; H(S): hybrid model w/ SNI

method on the proposed Math23K dataset. We further show that the baseline method cannot solve problems with new equation templates. In contrast, the proposed seq2seq model is quite robust on problems with new equation templates (refer to Table 7).

6.1 Experimental Setup

Datasets: As introduced in Section 3.2, we collected a dataset called Math23K which contains 23161 math word problems labeled with equation templates and answers. All these problems are linear algebra questions with only one variable. There are 2187 equation templates in the dataset. In addition, we also evaluate our method on a public dataset Alg514 (Kushman et al., 2014).

Baseline: We compare our proposed methods with two baselines. The first baseline is the retrieval model introduced in Section 5.1. The second one is ZDC (Zhou et al., 2015), which is an improved version of KAZB (Kushman et al., 2014). It maps a problem to one equation template defined in the training set by reasoning across problem sentences. It reports an accuracy of 79.7% on the Alg514 dataset. The Stanford parser is adopted in ZDC to parse all math word problems to Stanford coreNLP output formats. [2]

6.2 Experimental Results

Each approach is evaluated on each dataset via 5-fold cross-validation: In each run, 4 folds are used for training and 1 fold is used for testing. Evaluation results are summarized in Table 5. First, to test the effectiveness of *significant number identification* (SNI), model performance before and after the application of SNI are compared. Then, the performance of the hybrid model, seq2seq model, and retrieval model are examined on two datasets respectively.

To check whether the performance improvements are significant enough, we conduct statistical significance study upon pairs of methods. Table 6 shows the results of sign test, where the symbol $>$ indicates that the method in the row significantly (with p value < 0.05) improves the performance of the method in the column, and the symbol $\gg$ indicates that the performance improvement is extremely significant (with p value < 0.01).

Several observations can be made from the re-

[2] We also try to run KAZB on our dataset, but fail on our workstation (2 12-core E5-2650 CPU, 128G RAM, 4 K80 GPUs) due to large memory consumption.

sults. First, the seq2seq model significantly outperforms state-of-the-art statistical learning methods (ZDC and the retrieval model). Second, by combining the retrieval model and the seq2seq model using a simple mechanism, our hybrid model achieves significant performance gain with respect to the seq2seq model. Third, the SNI module can effectively improve model accuracy. The accuracy of the hybrid model and seq2seq model gains approximately 4% increase after number identification. Please pay attention that on the small dataset of Alg514, the seq2seq model behaves much worse than others. This is not surprising, because deep neural networks typically need large training data.

Figure 5 shows the performance of different models on various scales of training data. As expected, the seq2seq model performs very well on big datasets, but poorly on small datasets.

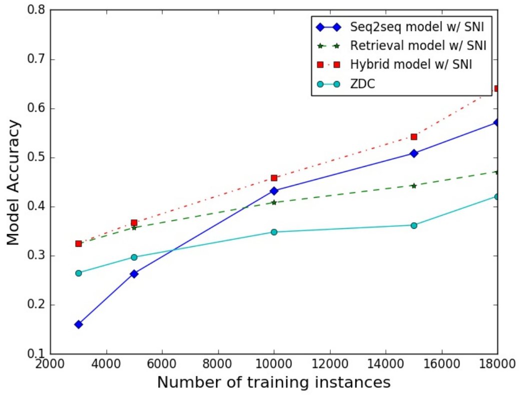


Figure 5: Performance of different models versus the size of training set

Ability to Generate New Equation Templates: please note that many problems in Math23K can be solved using the same equation template. For example, a problem which corresponds to the equation $x = (9 * 3) + 7$ and a different problem that maps to $x = (4 * 5) + 2$ share the same equation template.

One nice property of the seq2seq model is its ability of generating new equation templates. Most previous statistical learning methods (with a few exceptions) for math word problem solving are only able to select an equation template from those in the training data. In other words, they cannot generate new templates. To test the performance of the seq2seq model in generating new templates, we make a new split of our dataset between training data and test data, to ensure that the training data and the test data do not share overlapped templates. As a result, we get a training set with 19, 024 problems and 1, 802 equation templates, and a testing set with 4, 137 problems and 315 equation templates.

	Math23K
ZDC	15.1%
Retrieval model w/o SNI	26.1%
Retrieval model w/ NI	29.2%
Seq2seq model w/o SNI	40.3%
Seq2seq model w/ SNI	**47.5%**
Hybrid model w/o SNI	40.3%
Hybrid model w/ SNI	**47.7%**

Table 7: Experimental results of non-overlapping templates between training data and test data

Experimental results on the new training set and test set are shown is shown in Table 7. By comparing Table 5 and Table 7, it is clear that the gap between the seq2seq model and the baselines becomes larger in the new settings. It is because the seq2seq model can effectively generate new equation templates for new problems, instead of selecting equation templates from the training set.

Although ZDC and the retrieval model cannot generate new templates, their accuracy is not zero in the new settings. That is because one problem can be solved by multiple equation templates: Although one problem is labeled with template T_1 in the test set, it may also be solved by another template T_2 in the training set.

6.3 Discussion

Compare to most previous statistical learning methods for math problem solving, our proposed seq2seq model and hybrid model have the following advantages: 1) They have higher accuracy on large training data. On the Math23K dataset, the hybrid model achieves at least 22% higher accuracy than the baselines. 2) They have the ability of generating new templates (i.e., templates that are not in the training data. 3) They do not rely on sophisticated feature engineering.

7 Conclusion

We have proposed an RNN-based seq2seq model to automatically solve math word problems. This model directly transforms problem text to a math equation template. This is the first work of applying deep learning technologies to math word problem solving. In addition, we have designed a hybrid model which combines the seq2seq model and a retrieval model to further improve performance. A large dataset has been constructed for model training and empirical evaluation. Experimental results show that both the seq2seq model and the hybrid model significantly outperform state-of-the-art statistical learning methods in math word problem solving.

The output of our seq2seq model is a single equation containing one unknown variable. Therefore our approach is only applicable to the problems whose solution involves one linear equation of one unknown variable. As future work, we plan to extend our model to be able to generate equation systems and nonlinear equations.

References

Bussaba Amnueypornsakul and Suma Bhat. 2014. Machine-guided solution to mathematical word problems. In *PACLIC*. pages 111–119.

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

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. *arXiv preprint math/0701393* .

Daniel G Bobrow. 1964. Natural language input for a computer problem solving system .

Diane J Briars and Jill H Larkin. 1984. An integrated model of skill in solving elementary word problems. *Cognition and instruction* 1(3):245–296.

Eugene Charniak. 1968. *CALCULUS WORD PROBLEMS*. Ph.D. thesis, Massachusetts Institute of Technology.

Eugene Charniak. 1969. Computer solution of calculus word problems. In *Proceedings of the 1st international joint conference on Artificial intelligence*. Morgan Kaufmann Publishers Inc., pages 303–316.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Denise Dellarosa. 1986. A computer simulation of childrens arithmetic word-problem solving. *Behavior Research Methods, Instruments, & Computers* 18(2):147–154.

Edward A Feigenbaum, Julian Feldman, et al. 1963. *Computers and thought*. New York.

Charles R Fletcher. 1985. Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods, Instruments, & Computers* 17(5):565–571.

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

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*. pages 523–533.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word probl ems? large-scale dataset construction and evaluation. *Proceedings of the 2016 North American Chapter of the ACL (NAACL HLT)* .

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *TACL* 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. Association for Computational Linguistics.

Christian Liguda and Thies Pfeiffer. 2012. Modeling math word problems with augmented semantic networks. In *International Conference on Application of Natural Language to Information Systems*. Springer, pages 247–252.

Nicholas Locascio, Karthik Narasimhan, Eduardo DeLeon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. *arXiv preprint arXiv:1608.03000* .

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. ACL.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413* .

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics* 3:1–13.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364* .

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *EMNLP*. pages 1132–1142.

Upadhyay Shyam and Chang Ming-Wei. 2017. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. In *EACL*. pages 494–504.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

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

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960* .

Ma Yuhui, Zhou Ying, Cui Guangzuo, Ren Yun, and Huang Ronghuai. 2010. Frame-based calculus of solving arithmetic multi-step addition and subtraction word problems. In *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*. IEEE, volume 2, pages 476–479.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *EMNLP*. pages 817–822.

Latent Space Embedding for Retrieval in Question-Answer Archives

Deepak P[1] **Dinesh Garg**[2] **Shirish Shevade**[3]
[1]Queen's University Belfast, Northern Ireland, UK deepaksp@acm.org
[2]Indian Institute of Technology Gandhinagar, India dgarg@iitgn.ac.in
[3]Indian Institute of Science, Bangalore, India shirish@csa.iisc.ernet.in

Abstract

Community-driven Question Answering (CQA) systems such as Yahoo! Answers have become valuable sources of reusable information. CQA retrieval enables usage of historical CQA archives to solve new questions posed by users. This task has received much recent attention, with methods building upon literature from translation models, topic models, and deep learning. In this paper, we devise a CQA retrieval technique, *LASER-QA*, that embeds question-answer pairs within a unified latent space preserving the local neighborhood structure of question and answer spaces. The idea is that such a space mirrors semantic similarity among questions as well as answers, thereby enabling high quality retrieval. Through an empirical analysis on various real-world QA datasets, we illustrate the improved effectiveness of LASER-QA over state-of-the-art methods.

1 Introduction

Community-based Question Answering (CQA) services such as Yahoo! Answers[1], Quora[2], StackOverflow[3], and Baidu Zhidao[4] have become a dependable source of knowledge to solve common user problems. These allow a user to post queries such as *how* and *why* questions that seek descriptive solutions and opinions as answers. Over time, these services build up a large archive of question-answer knowledge that may be leveraged to solve new user questions. The CQA retrieval problem, that has received much recent attention, is about addressing this opportunity. CQA retrieval methods focus on finding historical archived knowledge (questions, answers or QA pairs) that are relevant to a newly posed user question. The central technical challenge that differentiates CQA retrieval from other general purpose IR tasks is that of the need to address the lexical gap (aka *lexical chasm*) in QA archives. Lexical chasm means that text fragments in questions (e.g., *disk full*) may lead to semantically correlated content in answers (e.g.,*format*). This QA-correlation is different from semantic relatedness such as *synonymy* and *antonymy*; in the above example, the correlation is due to *disk full* issues often leading to solution involving disk *format*ing. Explicit correlation modelling, using statistical translation models, have met with much success in CQA retrieval.

Table 1: Example CQA Pairs

#	QA	Cause
1	**Q:** My internet connection is not working, my router shows the "Internet" led blinking in red.	Router Authentication Issue
	A: Please go to the router login page and re-login with broadband credentials; click "connect" and you should be on the internet.	
2	**Q:** My internet connection is not working, only the power led is lit in the router.	Router Loose Connection
	A: Can you check whether the broadband cable is plugged in. Maybe, the broadband cable is not connected properly.	

In this paper, we take a neighborhood preserving learning approach, and learn a unified representation for QA pairs in an abstract *latent space*. Consider two example CQA pairs from a technical support forum presented in Table 1; the intuitive causes listed alongside are external to the dataset. Though the questions are reasonably similar lexically, they pertain to very different issues

[1] https://answers.yahoo.com/
[2] https://www.quora.com/
[3] http://stackoverflow.com/
[4] https://en.wikipedia.org/wiki/Baidu_Knows

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 855–865
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

as illustrated by the wide disparity in the answers posed to them. We model QA-pairs in a unified space that preserves the similarity neighborhood in question and answer spaces. In this example, the wide divergence in answer-space similarity neighborhoods between the two QAs would pull them apart, so they live in different parts of the latent space, reflecting the dissimilarity between their causes. Thus, our contribution in this paper is a neighborhood-preserving method for CQA retrieval, LASER-QA, expanding to **LA**tent-**S**pace **E**mbedding for **R**retrieval in **QA** archives.

2 Related Work

The three main CQA retrieval tasks target retrieving (a) related past questions (Zhou et al., 2015), (b) potentially usable past answers (Shtok et al., 2012), and (c) past question-answer pairs (Xue et al., 2008). Techniques for CQA typically use one of: (i) statistical translation models, (ii) topic models and (iii) neural networks. A fourth class target exploiting metadata such as question categories and author data, or domain-specific syntactic information, and are not as applicable in the absence of such information.

In the interest of keeping this section focused on retrieval, we do not cover other tasks that have been addressed for CQA, such as QA-pair discovery (Deepak and Visweswariah, 2014), clustering (Deepak, 2016) and auxiliary IR tasks such as query suggestions (Deepak et al., 2013).

2.1 Translation Model based Techniques

Translation models (Brown et al., 1990) take parallel corpora, collections of document pairs expressing the same thing in different natural languages, and learn correlations between words/phrases; for example, $p(f|e)$ quantifies the probability of an english word *e* getting translated to a french word *f* in an English-French translation system. Though question-answer pairs do not semantically qualify as parallel corpora, usage of translation models treating them so(Xue et al., 2008) have led to retrieval accuracy improvements. Simplistically, a high probability for $p(format|disk)$ leads to retrieval models boosting the score of a answer containing the word *format* to respond to a user query involving a *disk* problem. Later methods have improved upon them by phrase-level (Zhou et al., 2011) and entity-level (Singh, 2012) modelling as well as by unimportant word removal (Lee et al., 2008) and differential treatment of concepts (Park and Croft, 2015). Recent work has even explored using a different language (e.g., Chinese) to enrich questions (Zhang et al., 2015).

2.2 Topic Model based Techniques

Topic models (Blei et al., 2003) have been used to retrieve topically similar questions (Cai et al., 2011) with usage of the solution side leading to further improvements (Ji et al., 2012). They have been combined with language modeling whereby question and answer parts are modeled to have been generated from *paired* latent topics, but in ”question and answer languages” (Zhang et al., 2014). We will use such paired topic modelling, called TBLM, as a baseline in our experimental study.

2.3 Topic+Translation Models

Hybrid methods build upon topic and translation models by interpolating the separate scorings. Due to the usage of a combination of multiple types of parameterized models, the results of such ”pipeline methods” have been observed to be hard to reproduce (Qiu et al., 2013).We use a recent hybrid scoring method, called TopicTRLM (Zhou et al., 2015), as a baseline in our experimental study.

2.4 Deep Learning Methods

Neural networks such as DBNs (Wang et al., 2011; Hu et al., 2013) and more sophisticated neural pipelines (Shen et al., 2015) have been explored for CQA retrieval. A recent work (Nakov et al., 2016a) trains a neural network to discriminate between good and bad comments for a question. Using neural networks for retrieval within question datasets (not involving answers) has also been a subject of recent interest (e.g., (Bogdanova et al., 2015; Das et al., 2016)). The most recent method for generic QA-pair processing, which we will call as AENN (Zhou et al., 2016), trains separate autoencoders for question and answer corpora, and induces correlatedness of intermediate representations in a fine-tuning step. In our empirical analysis, we will use the AENN approach from (Zhou et al., 2016) as a baseline.

2.5 Auxiliary-information based Methods

This category of methods target to exploit specific kinds of auxiliary information that are potentially available with CQA data. Techniques have considered usage of question categories (Cao et al., 2009; Zhou et al., 2014), the split between question title and description (Qiu et al., 2013), and assumptions of the question syntax (Duan et al., 2008). While such information is available in many systems, QA information from systems such as forums and chat-based customer support sometimes have very little information other than just QA-pairs. We target a general scenario where such metadata is not expected as a pre-requisite, as in the case of most techniques from other categories.

3 Problem Statement

Let $\mathcal{D} = \{(\boldsymbol{q}_1, \boldsymbol{a}_1), \dots, (\boldsymbol{q}_n, \boldsymbol{a}_n)\}$ be QA pairs from a CQA archive where answer $\boldsymbol{a}_i$ is associated with question $\boldsymbol{q}_i$; for cases involving multiple answers for a question, the question would be replicated for each answer. For a new question $\boldsymbol{q}$, the CQA retrieval problem is about devising a scoring function $f(\boldsymbol{q}, (\boldsymbol{q}_i, \boldsymbol{a}_i))$ that quantifies the relevance of each $(\boldsymbol{q}_i, \boldsymbol{a}_i)$ pair from $\mathcal{D}$ to the new question $\boldsymbol{q}$. Having devised a scoring function, retrieval is trivially accomplished by choosing an ordered set of top-t QA pairs from $\mathcal{D}$ in accordance with their $f(\cdot, \cdot)$ scores.

3.1 Evaluation

In the datasets that we use, we have labels indicating which QAs are related/relevant to a particular question. Thus, the quality of the scoring function can be evaluated using traditional information retrieval metrics (Robertson and Zaragoza, 2007) such as Precision, MAP, MRR, and NDCG when measured against such labellings. In addition, we will use one more metric, namely **Success Rate**, *the fraction of questions for which at least one related question is ranked among the top-t*, in evaluation.

4 LASER-QA: The Proposed Technique

Our method, LASER-QA, embeds QA pairs in $\mathcal{D}$, within a unified space of desired dimensionality.

$$\{(\boldsymbol{q}_1, \boldsymbol{a}_1), \dots, (\boldsymbol{q}_n, \boldsymbol{a}_n)\} \xrightarrow{LASER-QA} \{\boldsymbol{u}_1, \dots, \boldsymbol{u}_n\}$$

where, $\boldsymbol{u}_i \in \mathbb{R}^d$ is a vector space embedding in the latent space $\mathbb{R}^d$. As we will illustrate, LASER-QA targets to preserve the local similarity structures in the question and answer spaces within the unified embedding. Having built the embedding of QA pairs, cosine similarity between vectors in $\mathbb{R}^d$ is used for scoring:

$$f(\boldsymbol{q}, (\boldsymbol{q}_i, \boldsymbol{a}_i)) = \frac{\boldsymbol{u}^\top \boldsymbol{u}_i}{\|\boldsymbol{u}\| \|\boldsymbol{u}_i\|}, \tag{1}$$

where, $\boldsymbol{u} \in \mathbb{R}^d$ is the embedding of the new question $\boldsymbol{q}$; we will outline the embedding of single questions into $\mathbb{R}^d$ in a later section.

Our motivation behind LASER-QA stems from the idea of Local Linear Embedding (LLE) (Saul and Roweis, 2000); further, the choice of local neighborhood preservation is motivated by pervasive usage of local neighbors (i.e., k-NN retrieval) in case-based reasoning frameworks (De Mantaras et al., 2005) that seek to reuse structured problem-solution data.

4.1 Data Representation

We use the *tf-idf* vector representation for each question (denoted as $\boldsymbol{x}_i$) and each answer ($\boldsymbol{y}_i$) in $\mathcal{D}$. The *tf-idf* vectors are in $\mathbb{R}^D$ where D denotes the size of the vocabulary. The question and answer tf-idf vectors are arranged as columns to form matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$, both of size $D \times n$. Recall, the *latent space* would be a Euclidean space of dimension d, and typically, we have $d < D$. Our method is intentionally designed to not rely on the specifics of the representation used, and thus can make use of any vector representation of text data. Note that our latent space embeddings in $\mathbb{R}^d$ are evidently unrelated to distributional text embeddings (e.g., (Mikolov et al., 2013)) and are complementary in that such embeddings could be used as an alternative input representation for $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$.

4.2 Regularized Reconstruction Coefficients

For any question $\boldsymbol{x}_i$, let $\mathcal{N}_k(\boldsymbol{x}_i)$ denote the set of top-k nearest questions to the question $\boldsymbol{x}_i$, proximity assessed using cosine similarity of vectors in $\mathbb{R}^D$; analogously, $\mathcal{N}_k(\boldsymbol{y}_i)$ denotes the top-k nearest answers to $\boldsymbol{y}_i$. Much like the representation, the similarity measure may also be replaced as appropriate. Inspired by LLE (Saul and Roweis, 2000), we model the local neighborhood geometry around $\boldsymbol{x}_i$ using *reconstruction coefficient* w_{ij}^q for each question $\boldsymbol{x}_j \in \mathcal{N}_k(\boldsymbol{x}_i)$. We intend to learn the co-efficients such that $\boldsymbol{x}_i$ may be reconstructed well as a linear combination of the neighbors using the co-efficients. Thus, these co-efficients are computed by minimizing, for every question

$\boldsymbol{x}_i$, the *regularized reconstruction penalty (RRP)* given below:

$$\text{RRP}(\boldsymbol{x}_i) = \frac{1}{2}\left\|\boldsymbol{x}_i - \sum_{\boldsymbol{x}_j \in \mathcal{N}_k(\boldsymbol{x}_i)} w_{ij}^q \boldsymbol{x}_j\right\|^2 + \frac{\lambda}{2}\sum_{\boldsymbol{x}_j \in \mathcal{N}_k(\boldsymbol{x}_i)} \left(w_{ij}^q\right)^2 \tag{2}$$

The first term denotes the approximation error in reconstructing $\boldsymbol{x}_i$ as a linear combination of its k nearest neighbors using weights w_{ij}^q. The second term is an L2 regularization term weighted with a non-negative hyperparameter λ, which we set to 0.01 in our experiments. We replaced the sum-to-one constraint in (Saul and Roweis, 2000) by L2 regularization since the former produces large swings in magnitude on either sides of 0.0 (note co-efficients are not constrained to be non-negative) on high-dimensional spaces such as our tf-idf space, leading to stability concerns.

By explicitly assigning $w_{ij}^q = 0 \; \forall \boldsymbol{x}_j \notin \mathcal{N}_k(\boldsymbol{x}_i)$, we rewrite the above problem as:

$$\begin{aligned} \min_{\boldsymbol{w}_i^q} \; & \frac{1}{2}\boldsymbol{w}_i^{q\top}\left(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I}\right)\boldsymbol{w}_i^q - \\ & \boldsymbol{w}_i^{q\top}\boldsymbol{X}^\top\boldsymbol{x}_i + \frac{1}{2}\boldsymbol{x}_i^\top\boldsymbol{x}_i \\ & \text{subject to } \; w_{ij}^q = 0 \; \forall j \notin \mathcal{N}_k(\boldsymbol{x}_i) \end{aligned} \tag{3}$$

where $\boldsymbol{I}$ is an $n \times n$ identity matrix and $\boldsymbol{w}_i^q$ is a column vector of size n comprising reconstruction coefficients vector for $\boldsymbol{x}_i$. It can be shown that the nonzero entries of the optimal coefficient vector is:

$$(\boldsymbol{X}_i^\top\boldsymbol{X}_i + \lambda\boldsymbol{I}_k)^{-1}\boldsymbol{X}_i^\top\boldsymbol{x}_i \tag{4}$$

where $\boldsymbol{I}_k$ is an identity matrix of the size k and matrix $\boldsymbol{X}_i$ is a $D \times k$ matrix obtained from the matrix $\boldsymbol{X}$ by retaining only those columns which are neighbors of $\boldsymbol{x}_i$. Note, the above matrix inverse is well-defined since the matrix is *positive definite* by construction. Once we find these optimal coefficient vectors for all questions (answers), we stack them together column-wise and obtain a matrix, $\boldsymbol{W}^q$ ($\boldsymbol{W}^a$) of size $n \times n$, called the reconstruction coefficient matrix for questions (answers). These two matrices $\boldsymbol{W}^q$ and $\boldsymbol{W}^a$ capture the local geometry of the questions and answers in the QA-archive $\mathcal{D}$.

4.3 Embedding into Latent Space $\mathbb{R}^d$

In this step, we use the $\boldsymbol{W}^q$ and $\boldsymbol{W}^a$ matrices to do the transformation of the QA pairs, the $(\boldsymbol{x}_i, \boldsymbol{y}_i)$s to $\boldsymbol{u}_i$s. Building upon LLE, we develop a scheme to preserve the local neighborhood structure around $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ in learning the $\boldsymbol{u}_i$.

$$\begin{aligned} \min_{\boldsymbol{U}} \; & \alpha\sum_{i=1}^{n}\|\boldsymbol{u}_i\text{-}\boldsymbol{U}\boldsymbol{w}_i^q\|^2 + (1\text{-}\alpha)\sum_{i=1}^{n}\|\boldsymbol{u}_i\text{-}\boldsymbol{U}\boldsymbol{w}_i^a\|^2 \\ & \text{subject to: } \sum_{i=1}^{n}\boldsymbol{u}_i = 0 \\ & \qquad\qquad \boldsymbol{U}\boldsymbol{U}^\top = (n\text{-}1)\boldsymbol{I}_d \end{aligned} \tag{5}$$

where, $\boldsymbol{U}$ is a $d \times n$ matrix whose i^{th} column is equal to $\boldsymbol{u}_i$. $\alpha \in [0, 1]$ is a weighting parameter that allows to trade-off between question and answer spaces. At $\alpha = 1$, the embedding $\boldsymbol{u}_i$ will try to maximally align with question $\boldsymbol{x}_i$ and vice versa. Our constraints, like the analogous ones in LLE, ensure origin-centered mean solutions and avoid degenerate solutions, respectively (Pang et al., 2005). The first constraint is soft in that any optimal solution disregarding the constraint can be shifted to ensure origin-centering.

Towards capturing the optimal solution for Eq. 5, we define three $n \times n$ symmetric matrices,

$$\boldsymbol{Q} = (\boldsymbol{I} - \boldsymbol{W}^q)(\boldsymbol{I} - \boldsymbol{W}^q)^\top \tag{6}$$

$$\boldsymbol{A} = (\boldsymbol{I} - \boldsymbol{W}^a)(\boldsymbol{I} - \boldsymbol{W}^a)^\top \tag{7}$$

$$\boldsymbol{Z} = \alpha\boldsymbol{Q} + (1 - \alpha)\boldsymbol{P}, \; \alpha \in [0, 1] \tag{8}$$

Theorem 1. *If the eigenvalues of the matrix $\boldsymbol{Z}$ are arranged in the descending order and the eigenvectors corresponding to the last d eigenvalues are denoted by $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_d\}$, then, the optimal solution for Eq. (5), denoted by $\boldsymbol{U}^*$ is:*

$$\boldsymbol{U}^* = \begin{pmatrix} \boldsymbol{v}_1^\top \\ \boldsymbol{v}_2^\top \\ \cdots \\ \boldsymbol{v}_d^\top \end{pmatrix} \tag{9}$$

Further, origin centering is achieved by the following transformation:

$$\boldsymbol{U}^*_{centered} = \boldsymbol{U}^* - \boldsymbol{U}^*\boldsymbol{e}\boldsymbol{e}^\top \tag{10}$$

where $\mathbf{e}$ *is a a vector of all* $1'$*s.*

Proof: First, observe that the objective function of Eq. (5) can be rewritten in a compact form:

$$\alpha \left\| \boldsymbol{U} - \boldsymbol{U}\boldsymbol{W}^q \right\|_F^2 + (1-\alpha) \left\| \boldsymbol{U} - \boldsymbol{U}\boldsymbol{W}^a \right\|_F^2 \quad (11)$$

where, $\| \cdot \|_F$ denotes the Frobenius norm. Now, keeping the first constraint aside, we fold in the second constraint using Lagrange multipliers yielding the following Lagrangian $L(\boldsymbol{U}, \boldsymbol{\Lambda})$.

$$L(\boldsymbol{U}, \boldsymbol{\Lambda}) = \alpha \left\| \boldsymbol{U}\text{-}\boldsymbol{U}\boldsymbol{W}^q \right\|_F^2 + (1\text{-}\alpha) \left\| \boldsymbol{U}\text{-}\boldsymbol{U}\boldsymbol{W}^a \right\|_F^2 \\ + \boldsymbol{e}^\top \left(\boldsymbol{\Lambda} \circ \left(\boldsymbol{U}\boldsymbol{U}^\top\text{-}(n\text{-}1)\boldsymbol{I}_d \right) \right) \boldsymbol{e} \quad (12)$$

where $\boldsymbol{e}$ is an all 1's vector. In this Lagrangian,

- Matrix $\boldsymbol{\Lambda}$ is a $d \times d$ symmetric matrix denoting the Lagrange multipliers for the second constraint. Note, the last term is a compact representation of $d^2/2$ equality constraints.
- The symbol $\circ$ denotes the Hadamard products (element wise product) of two matrices.

For any matrix $\boldsymbol{M}$, we have $\|\boldsymbol{M}\|_F^2 = \text{Tr}(\boldsymbol{M}\boldsymbol{M}^\top)$ where Tr(.) is the trace. Thus, we can rewrite the first two terms of Eq.(12) as:

$$\alpha \,\text{Tr}\left((\boldsymbol{U} - \boldsymbol{U}\boldsymbol{W}^q)(\boldsymbol{U} - \boldsymbol{U}\boldsymbol{W}^q)^\top \right) + \\ (1-\alpha)\,\text{Tr}\left((\boldsymbol{U} - \boldsymbol{U}\boldsymbol{W}^a)(\boldsymbol{U} - \boldsymbol{U}\boldsymbol{W}^a)^\top \right)$$

A slight re-arrangement yields:

$$\alpha \,\text{Tr}\left(\boldsymbol{U}(\boldsymbol{I} - \boldsymbol{W}^q)(\boldsymbol{I} - \boldsymbol{W}^q)^\top \boldsymbol{U}^\top \right) + \\ (1-\alpha)\,\text{Tr}\left(\boldsymbol{U}(\boldsymbol{I} - \boldsymbol{W}^a)(\boldsymbol{I} - \boldsymbol{W}^a)^\top \boldsymbol{U}^\top \right)$$

The Q and A space components are now separated out into the first and second terms. We now simplify the notation using Eq. (6) and (7) to:

$$L(\boldsymbol{U}, \boldsymbol{\Lambda}) = \alpha \,\text{Tr}\left(\boldsymbol{U}\boldsymbol{Q}\boldsymbol{U}^\top \right) + (1\text{-}\alpha)\,\text{Tr}\left(\boldsymbol{U}\boldsymbol{A}\boldsymbol{U}^\top \right) \\ + \boldsymbol{e}^\top \left(\boldsymbol{\Lambda} \circ \left(\boldsymbol{U}\boldsymbol{U}^\top\text{-}(n\text{-}1)\boldsymbol{I}_d \right) \right) \boldsymbol{e}$$

Recall the following for any matrices $\boldsymbol{A}, \boldsymbol{B}$, & $\boldsymbol{C}$.

1. $\boldsymbol{A} \circ (\boldsymbol{B} - \boldsymbol{C}) = (\boldsymbol{A} \circ \boldsymbol{B}) - (\boldsymbol{A} \circ \boldsymbol{C})$
2. $\boldsymbol{e}^\top (\boldsymbol{A} \circ \boldsymbol{B})\, \boldsymbol{e} = \text{Tr}(\boldsymbol{A}\boldsymbol{B}^\top) = \text{Tr}(\boldsymbol{A}^\top \boldsymbol{B})$

This allows us to rewrite Eq. (13) as:

$$L(\boldsymbol{U}, \boldsymbol{\Lambda}) = \text{Tr}\left(\alpha \boldsymbol{U}\boldsymbol{Q}\boldsymbol{U}^\top + (1-\alpha)\boldsymbol{U}\boldsymbol{A}\boldsymbol{U}^\top \right. \\ \left. + \boldsymbol{\Lambda}\boldsymbol{U}\boldsymbol{U}^\top - (n-1)\boldsymbol{\Lambda} \right) \quad (13)$$

To find an optimal $\boldsymbol{U}$, we differentiate $L(\boldsymbol{U}, \boldsymbol{\Lambda})$ w.r.t $\boldsymbol{U}$ and equate to zero. This leads to:

$$\frac{\partial L(\boldsymbol{U}, \boldsymbol{\Lambda})}{\partial \boldsymbol{U}} = 2\alpha \boldsymbol{U}\boldsymbol{Q} + 2(1-\alpha)\boldsymbol{U}\boldsymbol{A} + 2\boldsymbol{\Lambda}\boldsymbol{U} = 0$$

The above follows from standard matrix properties (Petersen and Pedersen, 2012). Re-arranging:

$$(\alpha \boldsymbol{Q} + (1-\alpha)\boldsymbol{A})\, \boldsymbol{U}^\top = -\boldsymbol{U}^\top \boldsymbol{\Lambda} \quad (14)$$

One possible solution of the above equation could be constructed in the following manner.

1. Let $\boldsymbol{Z} = \alpha \boldsymbol{Q} + (1-\alpha)\boldsymbol{A}$
2. Compute the Eigen decomposition of $\boldsymbol{Z}$
3. Find the lowest (i.e., bottom) d Eigen values, and take the corresponding Eigen vectors.
4. Form a matrix $\boldsymbol{U}$ by stacking the selected Eigen vectors row-wise.

While any subset of d eigenvectors (and their eigenvalues) would be a solution for Eq. (14), we would take the bottom d eigenvectors for minimizing the objective; this is so since the objective becomes $\text{Tr}(-\boldsymbol{\Lambda})$ when Eq. (14) holds. The matrix constructed above is the optimal $\boldsymbol{U}^*$ in Eq. (9). This completes the proof. ■

The first constraint in Eq.(5) is then applied to centre the vectors around the origin using Eq.(10).

4.4 Embedding a new Question in $\mathbb{R}^d$

To use the historical $\boldsymbol{u}_i$ vectors to retrieve historical QAs against a new question (vector) $\boldsymbol{q}$, we need to embed the latter in the same space $\mathbb{R}^d$. This is achieved using the same structure as applied in forming the embedding; we start with identifying, from $\mathcal{D}$, the k-nearest questions to $\boldsymbol{q}$. The reconstruction co-efficient vector $\boldsymbol{w}^q$ is then learnt using Eq. (2). Finally, we obtain the embedding $\boldsymbol{u}$ for $\boldsymbol{x}$ as a $\boldsymbol{w}^q$-weighted linear combination of the $\mathbb{R}^d$ embeddings corresponding to the k-nearest neighbors. This is captured in steps 9-11 in Algorithm 1 given in the next section.

```
Algorithm 1: LASER-QA Algorithm
input   : $\mathcal{D} = \{(\boldsymbol{q}_1, \boldsymbol{a}_1), \ldots, (\boldsymbol{q}_n, \boldsymbol{a}_n)\}$
          (CQA corpus) & query $\boldsymbol{q}$
output  : Top-$t$ relevant QA pairs from $\mathcal{D}$
Offline Phase
1  Use appropriate data representation to form
     vector-pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ for every QA $(\boldsymbol{q}_i, \boldsymbol{a}_i)$;
2  Compute the reconstruction coefficient
     matrices $\boldsymbol{W}^q$ and $\boldsymbol{W}^a$ (Ref. Section 4.2);
3  $\boldsymbol{Q} \leftarrow (\boldsymbol{I} - \boldsymbol{W}^q)(\boldsymbol{I} - \boldsymbol{W}^q)^\top$;
4  $\boldsymbol{A} \leftarrow (\boldsymbol{I} - \boldsymbol{W}^a)(\boldsymbol{I} - \boldsymbol{W}^a)^\top$;
5  $\boldsymbol{Z} \leftarrow \alpha \boldsymbol{Q} + (1 - \alpha)\boldsymbol{A}$;
6  $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_d\} \leftarrow$ Bottom $d$ eigenvectors of
     matrix $\boldsymbol{Z}$;
7  $\boldsymbol{U}^* \leftarrow \begin{pmatrix} \boldsymbol{v}_1^\top \\ \boldsymbol{v}_2^\top \\ \cdots \\ \boldsymbol{v}_d^\top \end{pmatrix}$;
8  $\boldsymbol{U}^*_{\text{centered}} = \boldsymbol{U}^* - \boldsymbol{U}^* \boldsymbol{e}\boldsymbol{e}^\top$;
Query-time (Online) Phase
9  $\boldsymbol{x} \leftarrow$ Vector representation of the query $\boldsymbol{q}$;
10 $\boldsymbol{w}_x \leftarrow$ Vector of size $n$ capturing the
     reconstruction coefficients for $\boldsymbol{x}$;
11 $\boldsymbol{u} \leftarrow \boldsymbol{U}^*_{\text{centered}} \boldsymbol{w}_x$;
12 Output top-$t$ QA pairs based by computing
     the following scores
     $$f(\boldsymbol{q}, (\boldsymbol{q}_i, \boldsymbol{a}_i)) = \frac{\boldsymbol{u}^\top \boldsymbol{u}_i}{\|\boldsymbol{u}\| \|\boldsymbol{u}_i\|}$$
```

4.5 LASER-QA Algorithm

The details of the LASER-QA technique from the previous sections are summarized in Algorithm 1, with the offline (Steps 1-8) and query-time phase (Steps 9-12) clearly demarcated. It may be noted that, LASER-QA, being an optimization-based method, preserves Q/A-space local neighborhoods on a best-effort basis and does not offer guarantees on the fraction of local neighbors preserved from either spaces in the $\mathbb{R}^d$ embedding.

4.5.1 Generalizability of LASER-QA

LASER-QA can be easily extended to incorporate other kinds of information that might be available along with QA pairs such as images, votes (e.g., Blurtit[5], Quora and Yahoo! Answers) tags (Quora), categories (answers.com[6] and Yahoo! Answers) or comments (Quora and Blurtit). Consider data in the form of triplets $(\boldsymbol{q}_i, \boldsymbol{a}_i, \boldsymbol{m}_i)$ where $\boldsymbol{m}_i$ represents the extra information. The $\boldsymbol{m}_i$ vectors are subjected to the same form of processing as $\boldsymbol{q}_i$ and $\boldsymbol{a}_i$ vectors, leading to the $\boldsymbol{W}^m$ and $\boldsymbol{M}$ matrices. Line 5 in Algorithm 1 would then change to:

$$\boldsymbol{Z} \leftarrow \alpha_q \boldsymbol{Q} + \alpha_a \boldsymbol{A} + \alpha_m \boldsymbol{M} \tag{15}$$

where the different αs denote interpolation weights that need to be set appropriately. The remainder of the LASER-QA steps remain identical to Algorithm 1. It may be noted that α_m could be set to a low value if the utility of the extra information is deemed to be low.

4.5.2 Scalability of LASER-QA

We now analyze the scalability of LASER-QA, separately analysing the (a) one-time offline phase, and (b) query-time phase.

Offline Phase: This is a one-time operation at the system design time, involving matrix multiplications followed by eigen-decomposition. Our matrices being sparse, multiplications are fast and worst-case quadratic[7] in n. The Eigendecomposition is $\mathcal{O}(n^3)$, but being a fundamental matrix operation, very efficient implementations exist (especially for symmetric matrices such as ours) with sub-second response times for n of the order of thousands (in packages such as Eigen[8] and LAPACK[9]), trendlines illustrating that Eigendecompositions with even n of the order of millions are easy. The embeddings of all vectors are then indexed using conventional multi-dimensional indexes and/or locality sensitive hashing to aid querying.

Online/Query-time Phase: This encompasses (a) an IR query to find the k most similar questions, (b) solving for the k reconstruction co-efficients in Eq. 4 and forming the embedding, and (c) simply querying for top-t nearest neighbors over indexes built at design-time. The main query-time overhead (vis-a-vis conventional information retrieval) is the additional query over the multi-dimensional index; this construction ensures fast sub-linear response times for the online phase.

Scalability against other methods: In contrast to LASER-QA, it is notable that the baselines

[5] http://www.blurtit.com/

[6] http://www.answers.com/

[7] https://goo.gl/RQ1m0V

[8] https://goo.gl/phMJv9

[9] https://goo.gl/rJjBY6 (Fig 3.1)

employ expensive query-time operations; specifically, it is unclear as to how query-time retrieval using Eq.4 in the TBLM paper (Zhang et al., 2014) and Eq.1 in Topic-TRLM paper (Zhou et al., 2015) could be completed in sub-linear time.

5 Experimental Evaluation

5.1 Datasets and Experimental Setup

Datasets: We use two recent datasets in our evaluation, CQADupStack (Hoogeveen et al., 2015) and SemEval2016-Task3 (Nakov et al., 2016b). The former has a manually labelled set of *related* questions to every question, whereas the latter has relevance labels associated with answers (appearing as *comments*); these labellings make automated evaluation possible. Among the 12 subsets in CQADupStack, owing to scalability issues of the AENN baseline, we choose the three smaller subsets from CQADupStack, namely *webmasters* (1299 QAs), *android* (2193), and *gis* (3726) for a full comparative evaluation. Each of these are split into two halves, with one portion used for the training (that is, learning the statistical model such as LASER-QA, translation model, etc.) and the other one used for the testing (the 50:50 split ensures a sizeable test set). The *related* labellings are used only for evaluation purposes; however, since only training pairs are retrieved within this setup, *related* labellings across QAs in the testing set would be missed, artificially lowering the recall of all the methods in our evaluation. In a recent analysis (Hoogeveen et al., 2016), CQADupStack authors quantify the incompleteness of labeling in the dataset. Such issues further artificially reduce retrieval accuracies as estimated from our automated evaluation. The SemEval2016 dataset, on the other hand, has an implicit test-train split. We use the subset of the data categorized under *Qatar Living Lounge*, the largest category (which is 27% of the full dataset), for our experiments. All 'comments' that are labelled relevant to the associated question are paired together as QA-pairs to form a training set of 1366 pairs, with the test questions from the dataset used as is.

Baselines: As detailed in Section 2, we compare against three baselines (a) TBLM (Zhang et al., 2014) (topic model approach), (b) Topic-TRLM (Zhou et al., 2015) (topic+translation models), and (b) AENN (Zhou et al., 2016) (deep learning). TBLM requires an *answer quality signal* that we set to unity. We use author-recommended parameter settings for TBLM and TopicTRLM. Since AENN learns a latent space representation (though a separate one for questions and answers unlike LASER-QA), the evaluation w.r.t LASER-QA is a direct comparison of the quality of the respective latent spaces. The AENN method requires training triplets, i.e., [question, answer, other answer]; we populate the *other answer* part using the answer of a related question. This gives AENN an advantage as it uses relations among training pairs that are unavailable to other methods. For AENN, quality measures peaked around 2000 (for *webmasters* and *gis*) and 3000 (for *android* and *SemEval2016*) for latent space dimensionality; our results are from such settings.

LASER-QA Parameters: We set $k = 15$ and $\alpha = 0.8$, the latter ensuring that the question space is given more importance. We always set d to the number of eigen vectors in $\boldsymbol{Z}$, equalling $|\mathcal{D}|$. We will separately study LASER-QA trends against parameter variations as well.

Evaluation Metrics: We use Precision, Success Rate (SR) (Ref. Sec 3), MAP and NDCG (Robertson and Zaragoza, 2007) for our evaluation. Precision simply measures the fraction of related documents among the top-t that were retrieved. Due to this rank-agnostic construction, precision is unable to incentivize for putting the relevant results at the top of the result instead of deeper down. In contrast, MAP and NDCG are rank-aware metrics. MAP[10] computes the average of precisions computed at rank positions where a relevant result is returned. NDCG is another rank-aware metric[11] that discounts the appearance of the revelant result based on it's rank in the result set. We assess statistical significance using randomization tests (Smucker et al., 2007).

5.2 Evaluation Results and Insights

Table 2 summarizes the comparative evaluation across varying t (best results boldfaced). The following observations are notable:

- LASER-QA outperforms the other methods across datasets. This is followed by Topic-TRLM, TBLM and then AENN.

- LASER-QA's margin is highest at (small) values of t that are typical of scenarios involving human perusal of results. As t in-

[10] https://goo.gl/xr7NnD

[11] https://goo.gl/26Pcct

Table 2: Retrieval Results (● & ○ denote statistical significance at p-value < 0.01 & < 0.05 respectively)

Dataset →	**webmasters (#QAs=1299)**				**android (#QAs=2193)**				**gis (#QAs=3726)**				**SemEval2016 (#QAs=1366)**			
	t=5				**t=5**				**t=5**				**t=5**			
Method	Prec	SR	MAP	NDCG	Prec	SR	MAP	NDCG	Prec	SR	MAP	NDCG	Prec	SR	MAP	NDCG
LASER-QA	**0.022**$^\bullet$	**0.101**$^\bullet$	**0.079**$^\bullet$	**0.082**$^\bullet$	**0.026**$^\circ$	**0.127**$^\bullet$	**0.094**$^\bullet$	**0.102**$^\bullet$	**0.023**$^\bullet$	**0.111**$^\bullet$	**0.085**$^\bullet$	**0.089**$^\bullet$	**0.141**$^\circ$	**0.407**$^\bullet$	**0.266**$^\circ$	**0.265**$^\circ$
TBLM	0.009	0.043	0.031	0.034	0.017	0.080	0.059	0.061	0.008	0.041	0.034	0.035	0.084	0.259	0.158	0.158
TTRLM	0.012	0.056	0.030	0.033	0.022	0.101	0.061	0.067	0.016	0.079	0.048	0.052	0.067	0.201	0.134	0.139
AENN	0.008	0.035	0.023	0.026	0.007	0.033	0.012	0.012	0.005	0.026	0.016	0.017	0.030	0.148	0.060	0.059
	t=10				**t=10**				**t=10**				**t=10**			
Method	Prec	SR	MAP	NDCG	Prec	SR	MAP	NDCG	Prec	SR	MAP	NDCG	Prec	SR	MAP	NDCG
LASER-QA	**0.013**$^\bullet$	**0.116**$^\bullet$	**0.080**$^\bullet$	**0.088**$^\bullet$	**0.015**$^\circ$	**0.148**$^\bullet$	**0.097**$^\bullet$	**0.110**$^\bullet$	**0.013**$^\bullet$	**0.123**$^\bullet$	**0.086**$^\bullet$	**0.095**$^\bullet$	**0.125**$^\bullet$	**0.556**$^\bullet$	**0.291**$^\bullet$	**0.292**$^\bullet$
TBLM	0.006	0.050	0.031	0.036	0.010	0.089	0.060	0.066	0.005	0.046	0.034	0.037	0.070	0.296	0.156	0.163
TTRLM	0.010	0.093	0.036	0.043	0.014	0.124	0.065	0.076	0.010	0.098	0.050	0.060	0.069	0.383	0.152	0.160
AENN	0.005	0.043	0.024	0.029	0.005	0.047	0.014	0.018	0.003	0.030	0.016	0.019	0.036	0.259	0.069	0.078
	t=20				**t=20**				**t=20**				**t=20**			
Method	Prec	SR	MAP	NDCG	Prec	SR	MAP	NDCG	Prec	SR	MAP	NDCG	Prec	SR	MAP	NDCG
LASER-QA	**0.007**	**0.121**	**0.080**$^\bullet$	**0.092**$^\bullet$	**0.008**	**0.157**	**0.097**$^\bullet$	**0.116**$^\bullet$	**0.007**	**0.126**	**0.087**$^\bullet$	**0.100**$^\bullet$	**0.079**	**0.605**	**0.296**$^\bullet$	**0.327**$^\bullet$
TBLM	0.003	0.050	0.031	0.038	0.006	0.100	0.060	0.070	0.003	0.052	0.034	0.038	0.051	0.333	0.157	0.175
TTRLM	0.006	0.118	0.037	0.052	**0.008**	0.149	0.066	0.085	0.006	0.119	0.052	0.067	0.059	0.519	0.149	0.182
AENN	0.003	0.053	0.025	0.032	0.003	0.066	0.015	0.023	0.002	0.039	0.017	0.021	0.023	0.321	0.074	0.104
	t=50				**t=50**				**t=50**				**t=50**			
Method	Prec	SR	MAP	NDCG	Prec	SR	MAP	NDCG	Prec	SR	MAP	NDCG	Prec	SR	MAP	NDCG
LASER-QA	**0.003**	0.125	**0.081**$^\bullet$	**0.096**$^\bullet$	0.003	0.166	**0.097**$^\bullet$	**0.121**$^\bullet$	0.002	0.129	**0.087**$^\bullet$	**0.103**$^\bullet$	0.033	0.630	**0.295**$^\bullet$	**0.359**$^\bullet$
TBLM	0.001	0.056	0.031	0.040	0.003	0.112	0.060	0.074	0.001	0.060	0.035	0.040	0.026	0.346	0.151	0.187
TTRLM	**0.003**	**0.145**$^\circ$	0.038	0.061	**0.004**$^\circ$	**0.177**	0.066	0.093	**0.003**$^\bullet$	**0.152**$^\bullet$	0.052	0.074	**0.039**	**0.667**	0.140	0.214
AENN	0.002	0.070	0.025	0.035	0.002	0.088	0.015	0.028	0.001	0.065	0.017	0.025	0.016	0.407	0.066	0.107

		t=5				**t=50**			
Dataset	#QAs	Prec	SR	MAP	NDCG	Prec	SR	MAP	NDCG
stats	4004	**0.016**$^\bullet$	**0.076**$^\bullet$	**0.057**$^\bullet$	**0.060**$^\bullet$	0.002	0.096	**0.058**$^\bullet$	**0.071**$^\bullet$
programmers	4107	**0.020**$^\bullet$	**0.096**$^\bullet$	**0.068**$^\bullet$	**0.075**$^\bullet$	0.002	0.115	**0.069**$^\bullet$	**0.088**$^\bullet$
wordpress	4744	**0.019**$^\bullet$	**0.091**$^\bullet$	**0.069**$^\bullet$	**0.074**$^\bullet$	0.002	0.112	**0.070**$^\bullet$	**0.085**$^\bullet$
physics	5044	**0.025**$^\bullet$	**0.120**$^\bullet$	**0.088**$^\bullet$	**0.094**$^\bullet$	0.003	0.148	**0.090**$^\bullet$	**0.111**$^\bullet$
mathematica	5084	**0.018**$^\bullet$	**0.087**$^\bullet$	**0.067**$^\bullet$	**0.072**$^\bullet$	0.002	0.116	**0.069**$^\bullet$	**0.084**$^\bullet$
unix	5330	**0.023**$^\bullet$	**0.115**$^\bullet$	**0.089**$^\bullet$	**0.094**$^\bullet$	0.003	0.137	**0.091**$^\bullet$	**0.107**$^\bullet$
gaming	6398	**0.034**$^\bullet$	**0.166**$^\bullet$	**0.130**$^\bullet$	**0.137**$^\bullet$	0.004	0.189	**0.132**$^\bullet$	**0.155**$^\bullet$
english	6668	**0.024**$^\bullet$	**0.115**$^\bullet$	**0.090**$^\bullet$	**0.095**$^\bullet$	**0.003**	**0.130**	**0.092**$^\bullet$	**0.107**$^\bullet$

Table 3: LASER-QA Results (Boldfacing and Statistical Significance indications from comparison with TopicTRLM and TBLM) over Larger Categories in CQADupStack

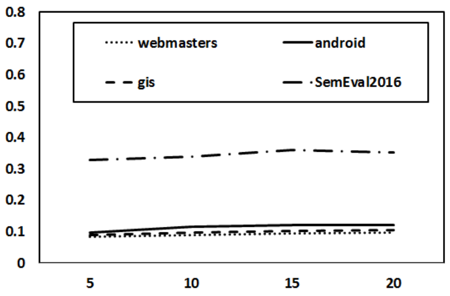


Figure 1: NDCG (Y-axis) v/s. k

creases way beyond the training neighborhood size (i.e., 15), LASER-QA is seen to deteriorate gracefully (as expected).

- LASER-QA performance peaks on rank-aware metrics such as MAP and NDCG (even at $t = 50$), indicating it's high effectiveness in producing relevant results at the top.

These observations underline the effectiveness of LASER-QA as a CQA retrieval method. It may be noted that LASER-QA uses compact representations ($d < 2000$), as compared to vocabulary space representations that are typically ≥ 5000.

Trends at High t: The performance trends at high values of t are explained by the usage of the local neighborhood (of size k) in LASER-QA latent space learning. Going down the result list much beyond k reveals expected, but graceful, decline in accuracy. For automated processing scenarios that necessitate large t, a correspondingly high k may be used in learning. It is notable that LASER-QA's focus on local neighborhood manifests as enhanced accuracy at the top of the result set.

LASER-QA Analysis on Larger CQADupStack Datasets: Owing to scalability issues of AENN that disallows a full evaluation over larger categories in CQADupStack, we present LASER-QA results over them in Table 3 to illustrate the consistency in trends. Boldfacing and statistical significance have the same semantics as earlier, with the comparison performed against only TopicTRLM and TBLM.

5.3 LASER-QA Parameter Analysis

We now analyze the NDCG trends (NDCG being the most popular IR metric) across LASER-QA parameters, i.e., k, α and d, varying each one separately keeping the default choice for others.

- **Varying k:** Figure 1 plots NDCG against values of k from $\{5, 10, 15, 20\}$. As may be seen, the accuracy is seen to improve with increasing k in the lower ranges, while sat-

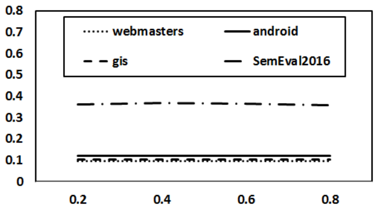


Figure 2: NDCG (Y-axis) v/s varying values of α

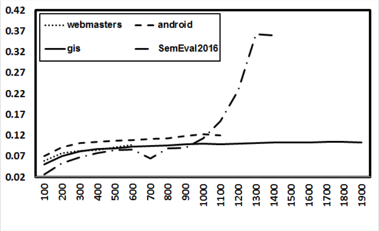


Figure 3: NDCG (Y-axis) v/s varying values of d

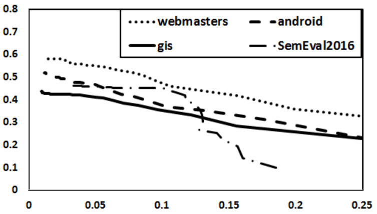


Figure 4: Precision (Y-axis) v/s Recall (X-axis)

urating beyond 15. The trends are seen to be similar across datasets.

- **Varying α:** The retrieval accuracies were seen to be stable across a wide range of values of α such as illustrated in Figure 2. This shows that LASER-QA is not very sensitive to α.

- **Varying d:** The size of the latent space, d, forms a critical parameter for LASER-QA. Given the LASER-QA construction, this space is limited by the number of eigenvectors in the matrix $\boldsymbol{Z}$ which is $n \times n$. This means, d is limited above by n, the size of the training dataset. Table 3 plots the accuracies with varying values of d, with the upper end different for different datasets due to the dependence on the training dataset size. The plots indicate that the performance improves steadily with increasing values of d. The performance saturates beyond 400 for the topically coherent CQADupStack datasets. The *Qatar Living Lounge* category in SemEval2016, unlike the CQADupStack categories, is more diverse discussing issues ranging from massage centres to immigration. Thus, LASER-QA is able to make use of much more dimensions to model the complexity involved.

To summarize, LASER-QA is not very sensitive to α and is best run with $k \geq 15$ and values of $d \approx n$.

Finally, the precision-recall curve with varying values of t is presented in Figure 4. As may be observed, LASER-QA exhibits a gradual degradation of precision with increasing t correlated with a corresponding improvement in recall. The diversity in the SemEval2016 dataset manifests as a sharper precision drop at high t, as the result set starts to transcend sub-topic boundaries.

6 Conclusions

We considered the problem of CQA retrieval – the task of retrieving relevant historical QA pairs in response to a new question. We formulated a method that builds upon the ideas from local linear embedding to use collective corpus level information across historical QA pairs to embed them in a latent space. In contrast to the mainstream paradigm in literature, we do not explicitly model lexical correlations; instead, we learn an embedding of QA pairs in a way that the local neighborhood in question and answer spaces are preserved. LASER-QA provides a single-model based solution in lieu of learning separate models (e.g., topic and translation models) which are then interpolated to a final scoring; the latter approach has been observed to have reproducilibility issues (Qiu et al., 2013). We analyzed our method empirically against state of the art methods from across families of CQA retrieval methods that use topic models, translation models and deep learning. Our empirical results confirm that the LASER-QA method significantly outperforms the baselines on all IR metrics of interest, underlining the effectiveness of our modelling.
Future Work: A study on the correlation between the kNNs in the LASER-QA embedded space and the original Question and Answer spaces would provide insights into the extent of correlation between manifolds in the original spaces. Further, we would like to see how LASER-QA generalizes to beyond text; one immediate scenario of interest is to explore how pictures and multimedia within QAs may be leveraged within LASER-QA.

Acknowledgments

A part of this work for Dinesh Garg was supported by the SERB-DST Early Career Research Grant No. ECR/2016/002035.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Dasha Bogdanova, Cícero Nogueira dos Santos, Luciano Barbosa, and Bianca Zadrozny. 2015. Detecting semantically equivalent questions in online user forums. In *CoNLL*, volume 123, page 2015.

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85.

Li Cai, Guangyou Zhou, Kang Liu, and Jun Zhao. 2011. Learning the latent topics for question retrieval in community QA. In *IJCNLP*, volume 11, pages 273–281.

Xin Cao, Gao Cong, Bin Cui, Christian Søndergaard Jensen, and Ce Zhang. 2009. The use of categorization information in language models for question retrieval. In *CIKM*, pages 265–274. ACM.

Arpita Das, Harish Yenala, Manoj Kumar Chinnakotla, and Manish Shrivastava. 2016. Together we stand: Siamese networks for similar question retrieval. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Ramon Lopez De Mantaras, David McSherry, Derek Bridge, David Leake, Barry Smyth, Susan Craw, Boi Faltings, Mary Lou Maher, MICHAEL T COX, Kenneth Forbus, et al. 2005. Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, 20(03):215–240.

P Deepak. 2016. Mixkmeans: Clustering question-answer archives. In *EMNLP*, pages 1576–1585.

P Deepak, Sutanu Chakraborti, and Deepak Khemani. 2013. Query suggestions for textual problem solution repositories. In *ECIR*, pages 569–581. Springer.

P Deepak and Karthik Visweswariah. 2014. Unsupervised solution post identification from discussion forums. In *ACL (1)*, pages 155–164.

Huizhong Duan, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. Searching questions by identifying question topic and question focus. In *ACL*, pages 156–164.

Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. 2015. CQADupStack: A benchmark data set for community question-answering research. In *Proceedings of the 20th Australasian Document Computing Symposium*, page 3. ACM.

Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. 2016. Cqadupstack: Gold or silver?

Haifeng Hu, Bingquan Liu, Baoxun Wang, Ming Liu, and Xiaolong Wang. 2013. Multimodal DBN for predicting high-quality answers in cQA portals. In *ACL (2)*, pages 843–847.

Zongcheng Ji, Fei Xu, Bin Wang, and Ben He. 2012. Question-answer topic model for question retrieval in community question answering. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2471–2474. ACM.

Jung-Tae Lee, Sang-Bum Kim, Young-In Song, and Hae-Chang Rim. 2008. Bridging lexical gaps between queries and questions on large online Q&A collections with compact translation models. In *EMNLP*, pages 410–418.

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

Preslav Nakov, Lluís Màrquez, and Francisco Guzmán. 2016a. It takes three to tango: Triangulation approach to answer ranking in community question answering. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1586–1597.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016b. Semeval-2016 task 3: Community question answering. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 525–545.

Yanwei Pang, Lei Zhang, Zhengkai Liu, Nenghai Yu, and Houqiang Li. 2005. Neighborhood preserving projections (NPP): A novel linear dimension reduction method. In *ICIC*, pages 117–125.

Jae Hyun Park and W Bruce Croft. 2015. Using key concepts in a translation model for retrieval. In *SIGIR*, pages 927–930. ACM.

K.B. Petersen and M. S. Pedersen. 2012. The matrix cookbook.

Xipeng Qiu, Le Tian, and Xuanjing Huang. 2013. Latent semantic tensor indexing for community-based question answering. In *ACL (2)*, pages 434–439.

Stephen Robertson and Hugo Zaragoza. 2007. On rank-based effectiveness measures and optimization. *Information Retrieval*, 10(3):321–339.

Lawrence K Saul and Sam T Roweis. 2000. An introduction to locally linear embedding. *unpublished. Available at: http://www. cs. toronto. edu/˜ roweis/lle/publications. html*.

Yikang Shen, Wenge Rong, Zhiwei Sun, Yuanxin Ouyang, and Zhang Xiong. 2015. Question/answer matching for CQA system via combining lexical and sequential information. In *AAAI*.

Anna Shtok, Gideon Dror, Yoelle Maarek, and Idan Szpektor. 2012. Learning from the past: answering new questions with past answers. In *WWW*, pages 759–768. ACM.

Amit Singh. 2012. Entity based Q&A retrieval. In *EMNLP-CoNLL*, pages 1266–1277. Association for Computational Linguistics.

Mark D. Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 623–632, New York, NY, USA. ACM.

Baoxun Wang, Bingquan Liu, Xiaolong Wang, Chengjie Sun, and Deyuan Zhang. 2011. Deep learning approaches to semantic relevance modeling for chinese question-answer pairs. *ACM Transactions on Asian Language Information Processing (TALIP)*, 10(4):21.

Xiaobing Xue, Jiwoon Jeon, and W Bruce Croft. 2008. Retrieval models for question and answer archives. In *SIGIR*, pages 475–482. ACM.

Kai Zhang, Wei Wu, Haocheng Wu, Zhoujun Li, and Ming Zhou. 2014. Question retrieval with high quality answers in community question answering. In *CIKM*, pages 371–380.

Weinan Zhang, Zhaoyan Ming, Yu Zhang, Ting Liu, and Tat-Seng Chua. 2015. Exploring key concept paraphrasing based on pivot language translation for question retrieval. In *AAAI*, pages 410–416.

Guangyou Zhou, Li Cai, Jun Zhao, and Kang Liu. 2011. Phrase-based translation model for question retrieval in community question answer archives. In *ACL(1)*, pages 653–662.

Guangyou Zhou, Yubo Chen, Daojian Zeng, and Jun Zhao. 2014. Group non-negative matrix factorization with natural categories for question retrieval in community question answer archives. In *COLING*, pages 89–98.

Guangyou Zhou, Yin Zhou, Tingting He, and Wensheng Wu. 2016. Learning semantic representation with neural networks for community question answering retrieval. *Knowledge-Based Systems*, 93:75–83.

Tom Chao Zhou, Michael Rung-Tsong Lyu, Irwin King, and Jie Lou. 2015. Learning to suggest questions in social media. *Knowledge and Information Systems*, 43(2):389–416.

Question Generation for Question Answering

Nan Duan[†], Duyu Tang[†], Peng Chen[§], Ming Zhou[†]
Microsoft Research Asia[†]
Microsoft Research and AI Group[§]
{nanduan,dutang,peche,mingzhou}@microsoft.com

Abstract

This paper presents how to generate questions from given passages using neural networks, where large scale QA pairs are automatically crawled and processed from Community-QA website, and used as training data. The contribution of the paper is 2-fold: First, two types of question generation approaches are proposed, one is a retrieval-based method using convolution neural network (CNN), the other is a generation-based method using recurrent neural network (RNN); Second, we show how to leverage the generated questions to improve existing question answering systems. We evaluate our question generation method for the answer sentence selection task on three benchmark datasets, including SQuAD, MS MARCO, and WikiQA. Experimental results show that, by using generated questions as an extra signal, significant QA improvement can be achieved.

1 Introduction

Question Answering (or QA) is one of the core problems for AI, and consists of several typical tasks, i.e. community-based QA (Qiu and Huang, 2015), knowledge-based QA (Berant et al., 2013), text-based QA (Yu et al., 2014), and reading comprehension (Rajpurkar et al., 2016). Most of current QA systems, e.g. (Berant and Liang, 2014), (Qiu and Huang, 2015), (Xiong et al., 2017), (Yin and Schtze, 2017), need labeled QA pairs as training data. Although labeling efforts have been made, such as WebQuestions dataset (Berant et al., 2013) and SimpleQuestions dataset (Bordes et al., 2015) for knowledge-based QA, WikiQA dataset (Yang et al., 2015) for text-based QA, SQuAD dataset (Rajpurkar et al., 2016) and MS MARCO dataset (Nguyen et al., 2016) for reading comprehension, these datasets are still with limited sizes, as labeling is very expensive.

Motivated by this, we explore how to generate questions from given passages using neural networks, with three expected goals: (1) the training data should need few or no human efforts and reflect commonly-asked question intentions; (2) the questions are generated based on natural language passages, and should have good quality; (3) the generated questions should be helpful to QA tasks.

To achieve the 1^{st} goal, we propose to acquire large scale high-quality training data from Community-QA (CQA) website. The motivation of using CQA website for training data collection is that, such websites (e.g., YahooAnswers, Quora, etc.) contain large scale QA pairs generated by real users, and these questions reflect the most common user intentions, and therefore are useful to search, QA, and chatbot scenarios.

To achieve the 2^{nd} goal, we explore two ways to generate questions for a given passage, one is a retrieval-based method using convolution neural network (CNN), the other is a generation-based method using recurrent neural network (RNN). We evaluate the generation quality by BLEU score (Papineni et al., 2002) and human annotations, and discuss their pros and cons in Section 9.

To achieve the 3^{rd} goal, we integrate our question generation approach into an end-to-end QA task, i.e., answer sentence selection, and evaluate its impact on three popular benchmark datasets, SQuAD, MS MARCO, and WikiQA. Experimental results show that, the generated questions can improve the QA quality on all these three datasets.

2 Question Generation

Formally, given a passage S, question generation (**QG**) engine generates a set of questions $\{Q_i\}$,

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 866–874
Copenhagen, Denmark, September 7–11, 2017.

where each generated $\mathcal{Q}_i$ can be answered by $\mathcal{S}$. There are four components in our QG engine:

1. *Question Pattern Mining*, which extracts the frequently-asked question patterns from large scale CQA questions, without any human annotation effort;
2. *Question Pattern Prediction*, which predicts top-N question patterns $\mathcal{Q}_p^1, ..., \mathcal{Q}_p^N$ given $\mathcal{S}$, by a retrieval-based method or a generation-based method. Therefore, "*Prediction*" has two different meanings here: in retrieval-based method, it means to rank existing question patterns and select the highest ranked ones, while in generation-based method, it means to generate question patterns based on $\mathcal{S}$ in a sequence-to-sequence manner, each of which could be a totally new question pattern beyond the existing question pattern set;
3. *Question Topic Selection*, which selects a phrase $\mathcal{Q}_t$ from $\mathcal{S}$ as the question topic, based on a predicted question pattern $\mathcal{Q}_p$. $\mathcal{Q}_t$ will be filled into $\mathcal{Q}_p$ to form a complete question;
4. *Question Ranking*, which ranks all generated questions by a set of features. Here, multiple questions with different intentions could be generated, as $\mathcal{S}$ could contain multiple facts.

3 Question Pattern Mining

A question pattern (or **QP**) is defined as a word sequence $\mathcal{Q}_p = \{w_1, w_2, ..., w_L\}$. Each QP should contain one and only one special word "#" as the placeholder, and all the other $L-1$ words are terminal words. For example, "*who founded # ?*" is a question pattern, and "#" denotes the placeholder, which could be an organization name. We generate questions only using *frequently-asked* question patterns, where "frequently-asked" denotes that each question pattern should be extracted from a large scale question set more than T times, where T denotes a pre-defined threshold.

In this paper, a question cluster (or **QC**) based approach is proposed to mine frequently-asked question patterns from large scale CQA questions.

First, a set of question clusters is collected from CQA webpages, and each question cluster consists of questions that are grouped as related questions[1] by the CQA website. For example, when the query "*what is the population of nyc*" is issued to YahooAnswers[2], the returned page contains a list of related questions including "*population of nyc*", "*nyc population*", "*nyc census*", and etc.

Second, for each question cluster $\mathcal{QC} = \{\mathcal{Q}_1, ..., \mathcal{Q}_K\}$, we enumerate all valid continuous n-grams, each of which should contain at least one content word and its order should be equal or less then 7, as question topic candidates $\{\mathcal{Q}_t^1, ..., \mathcal{Q}_t^M\}$. We then assign an importance score $Impts(\cdot)$ to each question topic candidate $\mathcal{Q}_t^m$:

$$Impts(\mathcal{Q}_t^m) = \sum_{\mathcal{Q}_k \in \mathcal{QC}} \delta(\mathcal{Q}_t^m, \mathcal{Q}_k) \cdot |\mathcal{Q}_t^m|$$

$\mathcal{Q}_t^m$ denotes the m^{th} question topic candidate, $\delta(\mathcal{Q}_t^m, \mathcal{Q}_k)$ equals to 1 when $\mathcal{Q}_t^m$ occurs in $\mathcal{Q}_k$, and 0 otherwise, $|\mathcal{Q}_t^m|$ denotes $\mathcal{Q}_t^m$'s word count, which boosts longer question topic candidates.

For each $\mathcal{QC}$, we select $\mathcal{Q}_t$ with the highest importance score as the question topic, and remove it from each question to form a question pattern. We call each removed question topic as a **historical question topic** of its corresponding question pattern. If $\mathcal{Q}_t$ doesn't exist in a question, ignore it.

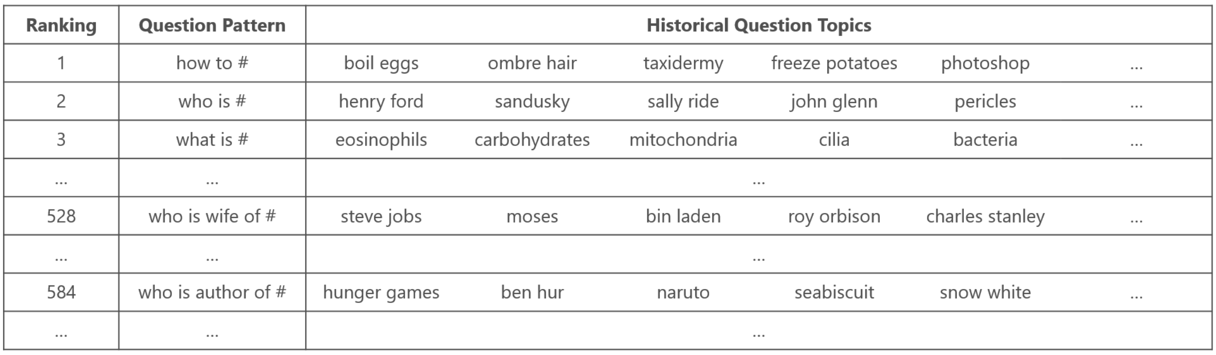
Mining question patterns based on question clusters is motivated by the observation that, all questions within a QC tend to ask questions about an identical "*question topic*" (e.g., *nyc* in the above example) from different aspects, or the same aspect but using different expressions. Thus, we can leverage the consensus information among questions to detect the boundary of the question topic: *the more times an n-gram occurs in different questions within a question cluster, the more likely it is the question topic of the current question cluster*.

Although the question pattern mining approach described above is simple, it works surprisingly well. Table 1 shows statistics of question patterns mined from YahooAnswers, and Figure 1 gives examples of frequently-asked question patterns with their corresponding historical question topics. We have two interesting observations:

1. Most frequently-asked question patterns (frequency >=10,000) are with high quality, and reflect the most common user intentions;
2. Most historical question topics extracted are entities. This is achieved without using any prior semantic knowledge base or dictionary.

[1] Usually, each question page of a CQA website contains a field that shows a list of related questions.

[2] https://answers.yahoo.com/



Ranking	Question Pattern	Historical Question Topics					
1	how to #	boil eggs	ombre hair	taxidermy	freeze potatoes	photoshop	...
2	who is #	henry ford	sandusky	sally ride	john glenn	pericles	...
3	what is #	eosinophils	carbohydrates	mitochondria	cilia	bacteria	...
...	...	...					
528	who is wife of #	steve jobs	moses	bin laden	roy orbison	charles stanley	...
...	...	...					
584	who is author of #	hunger games	ben hur	naruto	seabiscuit	snow white	...
...	...	...					

Figure 1: Examples of frequently-asked question patterns with corresponding historical question topics.

# of QCs	67,330,506
# of QPs	20,818,569
# of QPs (freq. >= 10,000)	105,623

Table 1: Statistics of question patterns mined from YahooAnswers, where QC denotes Question Cluster and QP denotes Question Pattern.

4 Question Pattern Prediction

Given a passage $\mathcal{S}$, question pattern prediction predicts $\mathcal{S}$'s most related question patterns, and then use them to generate questions. For example, given $\mathcal{S}$ as "*Tesla Motors is an American automaker and energy storage company co-founded by Elon Musk, Martin Eberhard, Marc Tarpenning, JB Straubel and Ian Wright, and is based in Palo Alto.*", two question patterns can be derived from $\mathcal{S}$, including: (1) *who founded # ?*, which can be inferred by the context around "*co-founded by*", and (2) *where is # located ?*, which can be inferred by the context around "*is based in*". Based on these two question patterns, we can generate two questions, "*who founded Tesla Motors ?*" and "*where is Tesla Motors located ?*" respectively.

4.1 Training Data Construction

We collect QA pairs from YahooAnswers. For each QA pair $< \mathcal{Q}, A >$, if (1) $\mathcal{Q}$ can be matched by a frequently-asked question pattern $\mathcal{Q}_p$, and (2) the matched question topic $\mathcal{Q}_t$ of $\mathcal{Q}$ based on $\mathcal{Q}_p$ exists in A, then we create a training instance as $< A, \mathcal{Q}_p, \mathcal{Q}_t >$. $\mathcal{Q}_t \in A$ makes sure that the question topic occurs in both $\mathcal{Q}$ and A. If the matched question topic only exists in $\mathcal{Q}$, we just discard it. By doing so, we collect a total of 1,984,401 training instances as training data for QP prediction.

Two neural network-based question pattern prediction approaches are explored in this paper:

- *Retrieval-based QP Prediction*, which considers QP prediction as a ranking task;
- *Generation-based QP Prediction*, which considers QP prediction as a generation task.

4.2 Retrieval-based QP Prediction

The retrieval-based QP prediction is done based on an attention-based convolution neural network. It takes a passage and a question pattern as input, and outputs their corresponding vector representations. We denote each input pair as $\langle \mathcal{S}, \mathcal{Q}_p \rangle$, where $\mathcal{S}$ is a passage, and $\mathcal{Q}_p$ is a question pattern.

In the input layer, given an input pair $\langle \mathcal{S}, \mathcal{Q}_p \rangle$, an attention matrix $\mathcal{A}tt \in \Re^{|\mathcal{S}| \times |\mathcal{Q}_p|}$ is generated by pre-trained word embeddings of $\mathcal{S}$ and $\mathcal{Q}_p$, where each element $\mathcal{A}tt_{i,j} \in \mathcal{A}$tt is computed as:

$$\mathcal{A}tt_{i,j} = cosine(v_i^{\mathcal{S}}, v_j^{\mathcal{Q}_p})$$

where $v_i^{\mathcal{S}}$ (or $v_j^{\mathcal{Q}_p}$) denotes the embedding vector of the i^{th} (or j^{th}) word in $\mathcal{S}$ (or $\mathcal{Q}_p$).

Then, column-wise and row-wise max-pooling are applied to $\mathcal{A}tt$ to generate two attention vectors $V^{\mathcal{S}} \in \Re^{|\mathcal{S}|}$ and $V^{\mathcal{Q}_p} \in \Re^{|\mathcal{Q}_p|}$, where the k^{th} elements of $V^{\mathcal{S}}$ and $V^{\mathcal{Q}_p}$ are computed as:

$$V_k^{\mathcal{S}} = \max_{1<l<|\mathcal{Q}_p|} \{\mathcal{A}tt_{k,l}\} \; and \; V_k^{\mathcal{Q}_p} = \max_{1<l<|\mathcal{S}|} \{\mathcal{A}tt_{l,k}\}$$

$V_k^{\mathcal{S}}$ (or $V_k^{\mathcal{Q}_p}$) can be interpreted as the *attention score* of the k^{th} word in $\mathcal{S}$ (or $\mathcal{Q}_p$) with regard to all words in $\mathcal{Q}_p$ (or $\mathcal{S}$).

Next, two *attention distributions* $D^{\mathcal{S}} \in \Re^{|\mathcal{S}|}$ and $D^{\mathcal{Q}_p} \in \Re^{|\mathcal{Q}_p|}$ are generated for $\mathcal{S}$ and $\mathcal{Q}_p$ based on $V^{\mathcal{S}}$ and $V^{\mathcal{Q}_p}$ respectively, where the k^{th} elements of $D^{\mathcal{S}}$ and $D^{\mathcal{Q}_p}$ are computed as:

$$D_k^{\mathcal{S}} = \frac{e^{V_k^{\mathcal{S}}}}{\sum_{l=1}^{|\mathcal{S}|} e^{V_l^{\mathcal{S}}}} \; and \; D_k^{\mathcal{S}\mathcal{Y}} = \frac{e^{V_k^{\mathcal{Q}_p}}}{\sum_{l=1}^{|\mathcal{Q}_p|} e^{V_l^{\mathcal{Q}_p}}}$$

$D_k^{\mathcal{S}x}$ (or $D_k^{\mathcal{S}y}$) can be interpreted as the *normalized attention score* of the k^{th} word in $\mathcal{S}$ (or $\mathcal{Q}_p$) with regard to all words in $\mathcal{Q}_p$ (or $\mathcal{S}$).

Last, we update each pre-trained word embedding $v_k^{\mathcal{S}}$ (or $v_k^{\mathcal{Q}_p}$) to $\hat{v}_k^{\mathcal{S}}$ (or $\hat{v}_k^{\mathcal{Q}_p}$), by multiplying every value in $v_k^{\mathcal{S}}$ (or $v_k^{\mathcal{Q}_p}$) with $D_k^{\mathcal{S}}$ (or $D_k^{\mathcal{Q}_p}$). The underlying intuition of updating pre-trained word embeddings is to *re-weight the importance of each word in $\mathcal{S}$ (or $\mathcal{Q}_p$) based on $\mathcal{Q}_p$ (or $\mathcal{S}$), instead of treating them in an equal manner*.

In the convolution layer, we first derive an input matrix $Z_{\mathcal{S}} = \{l_1, ..., l_{|\mathcal{S}|}\}$, where l_t is the concatenation of a sequence of $m = 2d - 1$[3] updated word embeddings $[\hat{v}_{t-d}^{\mathcal{S}}, ..., \hat{v}_t^{\mathcal{S}}, ..., \hat{v}_{t+d}^{\mathcal{S}}]$, centralized in the t^{th} word in $\mathcal{S}$. Then, the convolution layer performs sliding window-based feature extraction to project each vector representation $l_t \in Z_{\mathcal{S}}$ to a contextual feature vector $h_t^{\mathcal{S}}$:

$$h_t^{\mathcal{S}} = tanh(W_c \cdot l_t)$$

where W_c is the convolution matrix, $tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$ is the activation function. The same operation is performed to $\mathcal{Q}_p$ as well.

In the pooling layer, we aggregate local features extracted by the convolution layer from $\mathcal{S}$, and form a sentence-level global feature vector with a fixed size independent of the length of the input sentence. Here, *max-pooling* is used to force the network to retain the most useful local features by $l_p^{\mathcal{S}} = [v_1^{\mathcal{S}}, ..., v_K^{\mathcal{S}}]$, where:

$$v_i^{\mathcal{S}} = \max_{t=1,...,|\mathcal{S}|} \{h_t^{\mathcal{S}}(i)\}$$

$h_t^{\mathcal{S}}(i)$ denotes the i^{th} value in the vector $h_t^{\mathcal{S}}$. The same operation are performed to $\mathcal{Q}_p$ as well.

In the output layer, one more non-linear transformation is applied to $l_p^{\mathcal{S}}$:

$$y(\mathcal{S}) = tanh(W_s \cdot l_p^{\mathcal{S}})$$

W_s is the semantic projection matrix, $y(\mathcal{S})$ is the final sentence embedding of $\mathcal{S}$. The same operation is performed to $\mathcal{Q}_p$ to obtain $y(\mathcal{Q}_p)$.

We train model parameters W_c and W_s by minimizing the following ranking loss function:

$$\mathcal{L} = \max\{0, M - cosine(y(\mathcal{S}), y(\mathcal{Q}_p)) \\ + cosine(y(\mathcal{S}), y(\mathcal{Q}_p^-))\}$$

where M is a constant, $\mathcal{Q}_p^-$ is a negative instance.

4.3 Generation-based QP Prediction

The generation-based QP prediction is done based on an sequence-to-sequence BiGRU (Bahdanau et al., 2015) that is commonly used in the neural machine translation field.

The encoder reads a word sequence of an input passage $\mathcal{S} = (x_1, ..., x_{|\mathcal{S}|})$, and the decoder predicts a word sequence of an output question pattern $\mathcal{Q}_p = (y_1, ..., y_{|\mathcal{Q}t|})$. The probability of generating a question pattern $\mathcal{Q}_p$ is computed as:

$$p(\mathcal{Q}_p) = \prod_{i=1}^{|\mathcal{Q}_p|} p(y_i | y_1, ..., y_{i-1}, c_i)$$

where each conditional probability is defined as:

$$p(y_i | y_1, ..., y_{i-1}, c_i) = g(y_{i-1}, s_i, c_i)$$

$g(\cdot)$ denotes a nonlinear function that outputs the probability of generating y_i. s_i denotes the hidden state of time t in decoder, which is computed as:

$$s_i = (1 - z_i) \circ s_{i-1} + z_i \circ \tilde{s}_i$$

where

$$\tilde{s}_i = \tanh(WE_{y_{i-1}} + U[r_i \circ s_{i-1}] + Cc_i)$$

$$z_i = \delta(W_z E_{y_{i-1}} + U_z s_{i-1} + C_z c_i)$$

$$r_i = \delta(W_r E_{y_{i-1}} + U_r s_{i-1} + C_r c_i)$$

$\delta(\cdot)$ is the sigmoid function, $\circ$ represents element-wise multiplication, $E_w \in \Re^{m \times 1}$ denotes the word embedding of a word w, W, W_z, $W_r \in \Re^{n \times m}$, U, U_z, $U_r \in \Re^{n \times n}$, and C, C_z, $C_r \in \Re^{n \times 2n}$ are weights. c_i denotes the context vector, which is computed as:

$$c_i = \sum_{j=1}^{|\mathcal{S}|} \alpha_{ij} h_j$$

where

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{|\mathcal{S}|} exp(e_{ik})}$$

$$e_{ij} = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$

$v_a \in \Re^{n'}$, $W_a \in \Re^{n' \times n}$, and $U_a \in \Re^{n' \times 2n}$ are weights. h_j denotes the j^{th} hidden state from encoder, which is the concatenation of the forward hidden state $\overrightarrow{h}_j$ and the back forward state $\overleftarrow{h}_j$.

[3] In this paper, m is set to 3.

For training, stochastic gradient descent (SGD) algorithm is used, and Adadelta (Zeiler, 2012) is used to adapt the learning rate of each parameter. Given a batch of $D = \{< \mathcal{S}, \mathcal{Q}_p >\}_{i=1}^{M}$ pairs with size M instances, the objective function is to minimize the negative log-likelihood:

$$\mathcal{L} = -\frac{1}{M} \sum_{<\mathcal{S},\mathcal{Q}_p>} \sum_{t=1}^{T} log(p(y_t|y_{<t}, \mathcal{S}))$$

For prediction, beam search is used to output the top-N question pattern predictions.

Retrieval-based approach can only find existing question patterns for each passage, but it makes sure that each question pattern comes from real questions and is in a good grammatical form; Generation-based approach, on the other hand, can generate totally new question patterns beyond existing question pattern set. We will compare both of them in the experimental part (Section 8).

5 Question Topic Selection

Given a passage $\mathcal{S}$ and a predicted question pattern $\mathcal{Q}_p$, question topic selection selects an n-gram (or a phrase) $\mathcal{Q}_t$ from $\mathcal{S}$, which can be then filled into $\mathcal{Q}_p$ to form a complete question. Since we have two question pattern prediction methods, we have two ways to select the question topic $\mathcal{Q}_t$ as well.

For $\mathcal{Q}_p$ from retrieval-based method, two types of prior knowledge are used to extract question topic candidates from $\mathcal{S}$, including:

- Entities as question topic candidates, which are detected based on Freebase[4] entities;
- Noun phrases as question topic candidates, which are detected based on the Stanford parser (Klein and Manning, 2003).

Once a question topic candidate $\mathcal{Q}_t$ is extracted from $\mathcal{S}$, we then measure how $\mathcal{Q}_t$ can fit $\mathcal{Q}_p$ by:

$$s(\mathcal{Q}_t, \mathcal{Q}_p) = \frac{1}{N} \cdot \sum_k \#(\mathcal{Q}_p^{t_k}) \cdot dist(v_{\mathcal{Q}_t}, v_{\mathcal{Q}_p^{t_k}})$$

$s(\mathcal{Q}_t, \mathcal{Q}_p)$ denotes the confidence that $\mathcal{Q}_t$ can be filled into $\mathcal{Q}_p$ to generate a reasonable question. $\mathcal{Q}_p^{t_k}$ denotes the k^{th} historical question topic of $\mathcal{Q}_p$. $\#(\mathcal{Q}_p^{t_k})$ denotes the number of times that $\mathcal{Q}_p^{t_k}$ is extracted from different question clusters to generate $\mathcal{Q}_p$. v_p denotes the question topic embedding of p, which is computed as the average of word embeddings in p. $dist(\cdot)$ denotes the cosine distance between two question topic embeddings. $N = \sum_k \#(\mathcal{Q}_p^{t_k})$ denotes the total number of historical question topics of $\mathcal{Q}_p$. The basic principle of the above equation is that, *the historical question topics of a given question pattern can help on measuring how possible a question topic candidate can be filled into this question pattern to form a reasonable question.* For example, as most historical question topics of "*who founded # ?*" are organization names, then it is very unlikely a date or a film name is suitable for this question pattern to generate a reasonable question.

For $\mathcal{Q}_p$ from generation-based method, suppose the placeholder # is the i^{th} word in $\mathcal{Q}_p$, then we select the j^{th} word $w_j \in \mathcal{S}$ as the question topic, which satisfies the following constraint:

$$w_j = \arg\max_{w_{j'} \in \mathcal{S}} \alpha_{ij'} = \arg\max_{w_{j'} \in \mathcal{S}} \frac{exp(e_{ij'})}{\sum_{k=1}^{|\mathcal{S}|} exp(e_{ik})}$$

This question topic selection strategy leverages the attention scores between $\mathcal{S}$ and $\mathcal{Q}_p$, and can be considered as a *COPY* mechanism.

6 Question Ranking

Given a predicted question pattern $\mathcal{Q}_p$ and a selected question topic $\mathcal{Q}_t$ of an input passage $\mathcal{S}$, a complete question $\mathcal{Q}$ can be simply generated by replacing $\#$ in $\mathcal{Q}_p$ with $\mathcal{Q}_t$. We use a set of features to rank generated question candidates:

- *question pattern prediction score*, which is the prediction score by either retrieval-based approach or generation-based approach;
- *question topic selection score*, for retrieval-based approach, this score is computed as $s(\mathcal{Q}_t, \mathcal{Q}_p)$, while for generation-based approach, this score is the attention score;
- *QA matching score*, which measures relevance between generated question $\mathcal{Q}$ and $\mathcal{S}$.
- *word overlap between* $\mathcal{Q}$ and $\mathcal{S}$, which counts number of words that co-occur in $\mathcal{Q}$ and $\mathcal{S}$;
- *question pattern frequency*, which equals to the extraction frequency of $\mathcal{Q}_p$, if $\mathcal{Q}$ is generated from or matched by $\mathcal{Q}_p$, and 0 otherwise.

All features are combined by a linear model as:

$$p(\mathcal{Q}|\mathcal{S}) = \sum_i \lambda_i \cdot h_i(\mathcal{Q}, \mathcal{S}, \mathcal{Q}_p, \mathcal{Q}_t)$$

[4] https://developers.google.com/freebase/

where $h_i(\mathcal{Q}, \mathcal{S}, \mathcal{Q}_p, \mathcal{Q}_t)$ is one of the features described above, and λ_i is the corresponding weight.

7 Question Generation for QA

This section describes how question generation can improve existing QA systems. There are several types of QA systems, i.e. knowledge-based QA, community-based QA, text-based QA, etc, and in this paper, we focus on text-based QA task (a.k.a. answer sentence selection), which aims to select one or multiple answer sentences from a text given an input question. We select this task as it can be considered as a dual task of QG.

A typical answer sentence selection method, such as (Yin et al., 2016; Santos et al., 2016; Miller et al., 2016; Tymoshenko et al., 2016), computes the relevance score between input question $\mathcal{Q}$ and each answer candidate $\mathcal{A}$, and selects the one with the highest relevance score as the final answer:

$$\widehat{\mathcal{A}} = \arg\max_{\mathcal{A}} P(\mathcal{A}|\mathcal{Q})$$

Motivated by Dual Learning (He et al., 2016), we integrate question generation into answer ranking procedure, by changing the above formula to:

$$\widehat{\mathcal{A}} = \arg\max_{\mathcal{A}}\{P(\mathcal{A}|\mathcal{Q}) + \lambda \cdot QQ(\mathcal{Q}, \mathcal{Q}^{gen}_{max})\}$$

λ is hyper-parameter, and in order to compute $QQ(\mathcal{Q}, \mathcal{Q}^{gen}_{max})$, we generate top-10 questions $\{\mathcal{Q}^{gen}_1, ..., \mathcal{Q}^{gen}_{10}\}$ for current answer candidate $\mathcal{A}$, and then compute the *question-to-generated question* matching score $QQ(\mathcal{Q}, \mathcal{Q}^{gen}_{max})$, by computing the similarity between input question $\mathcal{Q}$ and generated questions $\{\mathcal{Q}^{gen}_1, ..., \mathcal{Q}^{gen}_{10}\}$ as:

$$QQ(\mathcal{Q}, \mathcal{Q}^{gen}_{max}) = \arg\max_{i=1,...,10} sim(\mathcal{Q}, \mathcal{Q}^{gen}_i) \cdot p(\mathcal{Q}^{gen}_i)$$

$sim(\mathcal{Q}, \mathcal{Q}^{gen}_i)$ is the similarity between the input question $\mathcal{Q}$ and the i^{th} generated question $\mathcal{Q}^{gen}_i$, and computed as the cosine distance between averaged word embedding of $\mathcal{Q}$ and averaged word embedding of $\mathcal{Q}^{gen}_i$, $p(\mathcal{Q}^{gen}_i)$ denotes the posterior probability that is computed based on the generation score of each generated question:

$$p(\mathcal{Q}^{gen}_i) = \frac{p(\mathcal{Q}^{gen}_i|\mathcal{A})}{\sum_{i'=1}^{10} p(\mathcal{Q}^{gen}_{i'}|\mathcal{A})}$$

$p(\mathcal{Q}^{gen}_i|\mathcal{A})$ is output by the question generation model described in Section 6. The underlying motivation is that, *the questions generated from correct answers are more likely to be similar to labeled questions than questions generated from wrong answers*.

8 Related Work

Yao et al. (2012) proposed a semantic-based question generation approach, which first parses the input sentence into its corresponding Minimal Recursion Semantics (MRS) representation, and then generates a question guided by the English Resource Grammar that includes a large scale hand-crafted lexicon and grammar rules.

Labutov et al. (2015) proposed an 'ontology-crowd-relevance' method for question generation. First, Freebase types and Wikipedia session names are used as semantic tags to understand texts. Question are then generated based on question templates that are aligned with types/session names and labeled by crowdsourcing. All generated questions are ranked by a relevance model.

Chali and Hasan (2015) proposed a topic-to-question method, which uses about 350 general-purpose rules to transform the semantic-role labeled sentences into corresponding questions.

Serban et al. (2016) used the encoder-decoder framework to generate 30M QA pairs, but their inputs are knowledge triples, instead of passages.

Song and Zhao (2016) proposed a question generation method using question template seeds and used search engine to do question expansion.

Du et al. (2017) proposed a neural question generation method using a vanilla sequence-to-sequence RNN model, which is most-related to our work. But this method is still based on labeled dataset, and tried RNN only.

Comparing to all these related work mentioned above, our question generation approach has two uniqueness: (1) all question patterns, that are used as training data for question generation, are automatically extracted from a large scale CQA question set without any crowdsourcing effort. Such question patterns reflect the most common user intentions, and therefore are useful to search, QA, and chatbot engines; (2) it is also the first time question generation is integrated and evaluated in an end-to-end QA task directly, and shows significant improvements.

9 Experiment

9.1 Dataset

As described in Section 4.1, we collect $1,984,401$ $< A, \mathcal{Q}_p, \mathcal{Q}_t >$ pairs from YahooAnswers, and use them as the training set of the question pattern prediction model. We re-use the dev sets and test sets of SQuAD, MS MARCO, and WikiQA, to evaluate the quality of generated questions. The dataset statistics are in Table 2.

	# of QA Pairs
training set	1,984,401
dev set (SQuAD)	26,442
test set (SQuAD)	26,604
dev set (MS MARCO)	39,510
test set (MS MARCO)	42,850
dev set (WikiQA)	2,733
test set (WikiQA)	6,165

Table 2: Dataset Statistics.

Besides, an answer sentence selection model (Yin et al., 2016) is trained based on the 1,984,401 QA pairs from the training set as well, and used to compute the QA matching score for question ranking, as we described in Section 6. Feature weights for question ranking are optimized on dev set.

9.2 Evaluation on Question Generation

We first perform a vanilla sequence-to-sequence method (Du et al., 2017) using the original training sets of these three datasets, and show QG results in Table 3, where BLEU 4 score is used as the metric.

BLEU 4	Seq2Seq-QG
SQuAD	12.28
MS MARCO	10.46
WikiQA	2.04

Table 3: QG results using original training sets.

We then evaluate the quality of the generated questions based on auto-extracted training set. For each passage in the test set, we generate two top-1 questions based on retrieval-based method and generation-based method respectively, and then compare them with labeled questions using *BLEU 4* as the metric. Results are listed in Table 4.

BLEU 4	R-QG	G-QG
SQuAD (crawled)	9.87	12.39
MS MARCO (crawled)	9.86	11.46
WikiQA (crawled)	11.38	13.57

Table 4: QG results using auto-extracted training set, where **R-QG** denotes results from Retrieval-based QG method, **G-QG** denotes results from Generation-based QG method.

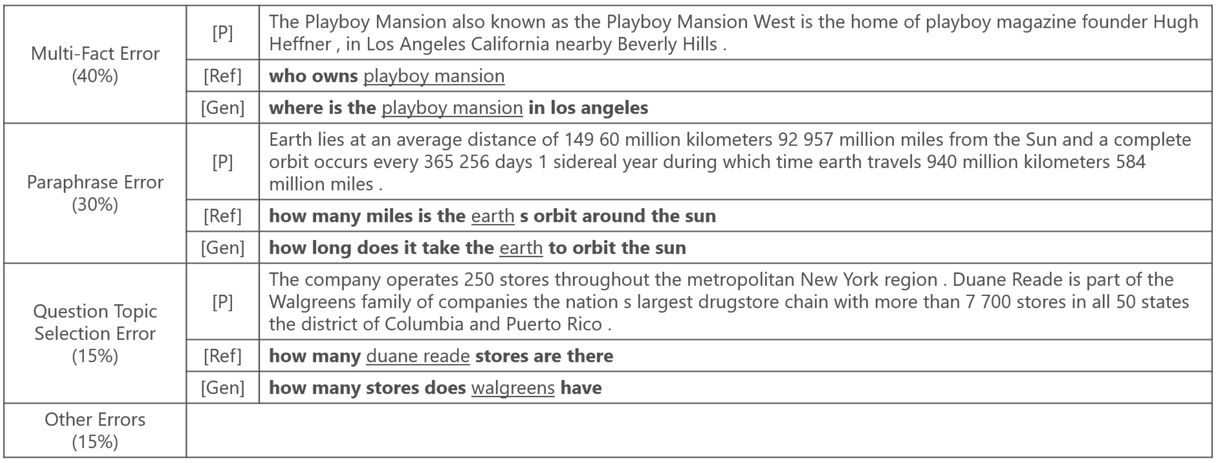
From Table 3 and 4 we can see two findings: (1) Comparing to QG results based on original labeled training sets, G-QG achieves comparable or better results. We think this is due to two facts: first, the size of the automatically constructed training set is much larger than the labeled training sets, and second, as the QA pairs from CQA websites are generated by real users, the quality is good. (2) Generation-based QG performs better than Retrieval-based QG. By analyzing outputs we find that, for question pattern prediction, both retrieval-based and generation-based methods perform similarly. However, Generation-based QG performs better than Retrieval-based QG on question topic selection. This could be caused by the fact that, in Generation-based QG, question topic selection is based on the attention mechanism, which is optimized together with question pattern prediction in an end-to-end way; while in Retrieval-based QG, question topic selection is a separate task, and based on the similarity between each question topic candidate and historical question topics of a given question pattern. The embedding of each question topic is pre-trained, which is not directly related to the question generation task. So such method cannot handle unseen question topics very well. Another disadvantage of Retrieval-based QG is that, each time, we have to compute the similarity between the input passage and each question pattern. When question pattern size is large, the computation is very expensive.

In order to better understand the question generation quality, we manually check a set of sampled outputs, and list the main errors in Figure 2:

- *Multi-Fact Error* (40%). Most input passages include more than one fact. For such a question, it is reasonable to generate different questions from different aspects, all of which can be answered by the input passage. For each passage in QAGen, we only label one question as ground truth. In the future, we will extend QAGen to be a more comprehensive dataset, by labeling multiple questions to each passage for more reasonable evaluation;

- *Paraphrase Error* (30%). The same question can be expressed by different ways. Labeling more paraphrased questions for a passage can alleviate this issue as well;

- *Question Topic Selection Error* (15%). This error is caused by selecting either a total-



Multi-Fact Error (40%)	[P]	The Playboy Mansion also known as the Playboy Mansion West is the home of playboy magazine founder Hugh Heffner , in Los Angeles California nearby Beverly Hills .
	[Ref]	**who owns** playboy mansion
	[Gen]	**where is the** playboy mansion **in los angeles**
Paraphrase Error (30%)	[P]	Earth lies at an average distance of 149 60 million kilometers 92 957 million miles from the Sun and a complete orbit occurs every 365 256 days 1 sidereal year during which time earth travels 940 million kilometers 584 million miles .
	[Ref]	**how many miles is the** earth **s orbit around the sun**
	[Gen]	**how long does it take the** earth **to orbit the sun**
Question Topic Selection Error (15%)	[P]	The company operates 250 stores throughout the metropolitan New York region . Duane Reade is part of the Walgreens family of companies the nation s largest drugstore chain with more than 7 700 stores in all 50 states the district of Columbia and Puerto Rico .
	[Ref]	**how many** duane reade **stores are there**
	[Gen]	**how many stores does** walgreens **have**
Other Errors (15%)		

Figure 2: List of error analysis examples. **[P]** denotes a passage, **[Ref]** denotes the labeled question of the passage, and **[Gen]** denotes the generated question of the passage.

ly wrong question topic, or a partially right question topic. In the future, we plan to develop an independent question topic selection model for the question generation task.

9.3 Evaluation on QA

As described in Section 7, we combine question generation into QA system for answer sentence selection task, and do evaluation on SQuAD, MS MARCO, and WikiQA. Evaluation results are shown in Table 5, 6, and 7, where **QA** denotes the result of our in-house implementation of a retrieval-based answer selection approach (DocChat) proposed by (Yan et al., 2016), **QA+QG** denotes result by combining question-to-generated question matching score with DocChat score.

SQuAD	MAP	MRR	ACC@1
QA	0.8843	0.8915	0.8160
QA+QG	0.8887	0.8963	0.8232

Table 5: Impact of QG on SQuAD.

MS MARCO	MAP	MRR	ACC@1
QA	0.5131	0.5195	0.3029
QA+QG	0.5230	0.5291	0.3153

Table 6: Impact of QG on MS MARCO.

WikiQA	MAP	MRR	ACC@1
QA	0.7703	0.7851	0.6540
QA+QG	0.7742	0.7893	0.6624

Table 7: Impact of QG on WikiQA.

The improvement on MS MARCO dataset is most significant. We think it due to the fact that, the questions from MS MARCO dataset are from Bing search log, which are generated naturally by real users. This is similar to the questions coming for CQA websites; while questions from the other datasets are labeled by crowd-sourcing.

In order to explain these improvements, two datasets, *WikiQG+* and *WikiQG-*, are built from WikiQA test set: given each document and its labeled question, we pair the question with its CORRECT answer sentence as a QA pair and add it to WikiQG+; we also pair the same question with a randomly selected WRONG answer sentence as a QA pair and add it to WikiQG-. Then, we generate questions for passages in WikiQG+ and WikiQG- respectively, and compare them with labeled questions. The BLEU 4 score is **0.2031** on WikiQG+, and 0.1301 on WikiQG-, which indicates that the questions generated from correct answers are more likely to be similar to labeled questions than questions generated from wrong answers.

10 Conslusion

This paper presents a neural question generation method that is based on training data collected from CQA questions. We integrate the QA pair generation task into an end-to-end QA task, and show significant improvements, which indicates that, QA task and QG are dual tasks that can boost each other. In the future, we will explore more ways to leverage QG for QA task.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *ACL*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. In *arXiv*.

Yllias Chali and Sadid Hasan. 2015. Towards topic-to-question generation. In *CL*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Association for Computational Linguistics (ACL)*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *NIPS*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *ACL*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *EMNLP*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jiangfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. In *NIPS*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community based question answering. In *IJCAI*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. In *arXiv*.

Iulian Vlad Serban, Alberto Garca-Durn, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questionswith recurrent neural networks: The 30m factoid question-answer corpus. In *ACL*.

Linfeng Song and Lin Zhao. 2016. Domain-specific question generation from a knowledge base. In *arXiv*.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitt. 2016. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *NAACL*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *ICLR*.

Zhao Yan, Nan Duan, Junwei Bao, Peng Chen, Ming Zhou, Zhoujun Li, and Jianshe Zhou. 2016. Docchat: An information retrieval approach for chatbot engines using unstructured documents. In *ACL*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*.

Xuchen Yao, Gosse Bouma, and Yi Zhang. 2012. Semantics-based question generation and implementation. In *Dialogue and Discourse*.

Wenpeng Yin and Hinrich Schtze. 2017. Task-specific attentive pooling of phrase alignments contributes to sentence matching. In *EACL*.

Wenpeng Yin, Hinrich Schtze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. In *TACL*.

Lei Yu, Karl Moriztt Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. In *NIPS Workshop*.

Matthew Zeiler. 2012. Adadelta: an adaptive learning rate method. In *CoRR abs*.

Learning to Paraphrase for Question Answering

Li Dong† and **Jonathan Mallinson**† and **Siva Reddy**‡ and **Mirella Lapata**†
† ILCC, School of Informatics, University of Edinburgh
‡ Computer Science Department, Stanford University
li.dong@ed.ac.uk, J.Mallinson@ed.ac.uk
sivar@stanford.edu, mlap@inf.ed.ac.uk

Abstract

Question answering (QA) systems are sensitive to the many different ways natural language expresses the same information need. In this paper we turn to paraphrases as a means of capturing this knowledge and present a general framework which learns felicitous paraphrases for various QA tasks. Our method is trained end-to-end using question-answer pairs as a supervision signal. A question and its paraphrases serve as input to a neural scoring model which assigns higher weights to linguistic expressions most likely to yield correct answers. We evaluate our approach on QA over Freebase and answer sentence selection. Experimental results on three datasets show that our framework consistently improves performance, achieving competitive results despite the use of simple QA models.

1 Introduction

Enabling computers to automatically answer questions posed in natural language on any domain or topic has been the focus of much research in recent years. Question answering (QA) is challenging due to the many different ways natural language expresses the same information need. As a result, small variations in semantically equivalent questions, may yield different answers. For example, a hypothetical QA system must recognize that the questions "*who created microsoft*" and "*who started microsoft*" have the same meaning and that they both convey the `founder` relation in order to retrieve the correct answer from a knowledge base.

Given the great variety of surface forms for semantically equivalent expressions, it should come as no surprise that previous work has investigated the use of paraphrases in relation to question answering. There have been three main strands of research. The first one applies paraphrasing to match natural language and logical forms in the context of semantic parsing. Berant and Liang (2014) use a template-based method to heuristically generate canonical text descriptions for candidate logical forms, and then compute paraphrase scores between the generated texts and input questions in order to rank the logical forms. Another strand of work uses paraphrases in the context of neural question answering models (Bordes et al., 2014a,b; Dong et al., 2015). These models are typically trained on question-answer pairs, and employ question paraphrases in a multi-task learning framework in an attempt to encourage the neural networks to output similar vector representations for the paraphrases.

The third strand of research uses paraphrases more directly. The idea is to paraphrase the question and then submit the rewritten version to a QA module. Various resources have been used to produce question paraphrases, such as rule-based machine translation (Duboue and Chu-Carroll, 2006), lexical and phrasal rules from the Paraphrase Database (Narayan et al., 2016), as well as rules mined from Wiktionary (Chen et al., 2016) and large-scale paraphrase corpora (Fader et al., 2013). A common problem with the generated paraphrases is that they often contain inappropriate candidates. Hence, treating all paraphrases as equally felicitous and using them to answer the question could degrade performance. To remedy this, a scoring model is often employed, however independently of the QA system used to find the answer (Duboue and Chu-Carroll, 2006; Narayan et al., 2016). Problematically, the separate paraphrase models used in previous work do not fully utilize the supervision signal from the training data, and as such cannot be properly tuned

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 875–886
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

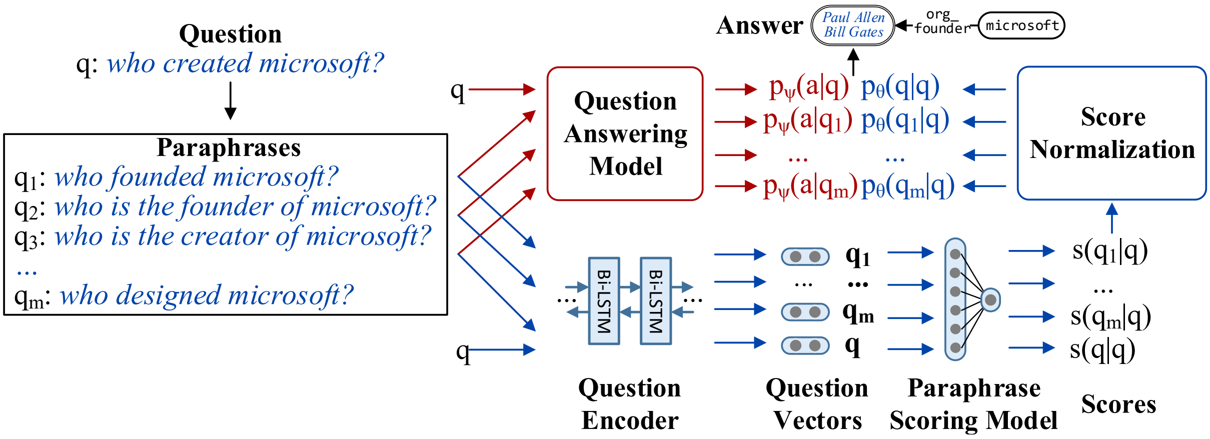


Figure 1: We use three different methods to generate candidate paraphrases for input q. The question and its paraphrases are fed into a neural model which scores how suitable they are. The scores are normalized and used to weight the results of the question answering model. The entire system is trained end-to-end using question-answer pairs as a supervision signal.

to the question answering tasks at hand. Based on the large variety of possible transformations that can generate paraphrases, it seems likely that the kinds of paraphrases that are useful would depend on the QA application of interest (Bhagat and Hovy, 2013). Fader et al. (2014) use features that are defined over the original question and its rewrites to score paraphrases. Examples include the pointwise mutual information of the rewrite rule, the paraphrase's score according to a language model, and POS tag features. In the context of semantic parsing, Chen et al. (2016) also use the ID of the rewrite rule as a feature. However, most of these features are not informative enough to model the quality of question paraphrases, or cannot easily generalize to unseen rewrite rules.

In this paper, we present a general framework for learning paraphrases for question answering tasks. Given a natural language question, our model estimates a probability distribution over candidate answers. We first generate paraphrases for the question, which can be obtained by one or several paraphrasing systems. A neural scoring model predicts the quality of the generated paraphrases, while learning to assign higher weights to those which are more likely to yield correct answers. The paraphrases and the original question are fed into a QA model that predicts a distribution over answers given the question. The entire system is trained end-to-end using question-answer pairs as a supervision signal. The framework is flexible, it does not rely on specific paraphrase or QA models. In fact, this plug-and-play functionality allows to learn specific paraphrases for different QA tasks and to explore the merits of different paraphrasing models for different applications.

We evaluate our approach on QA over Freebase and text-based answer sentence selection. We employ a range of paraphrase models based on the Paraphrase Database (PPDB; Pavlick et al. 2015), neural machine translation (Mallinson et al., 2016), and rules mined from the WikiAnswers corpus (Fader et al., 2014). Results on three datasets show that our framework consistently improves performance; it achieves state-of-the-art results on GraphQuestions and competitive performance on two additional benchmark datasets using simple QA models.

2 Problem Formulation

Let q denote a natural language question, and a its answer. Our aim is to estimate $p\,(a|q)$, the conditional probability of candidate answers given the question. We decompose $p\,(a|q)$ as:

$$p\,(a|q) = \sum_{q' \in H_q \cup \{q\}} \underbrace{p_\psi\,(a|q')}_{\text{QA Model}} \; \underbrace{p_\theta\,(q'|q)}_{\text{Paraphrase Model}} \qquad (1)$$

where H_q is the set of paraphrases for question q, ψ are the parameters of a QA model, and θ are the parameters of a paraphrase scoring model.

As shown in Figure 1, we first generate candidate paraphrases H_q for question q. Then, a neural scoring model predicts the quality of the generated paraphrases, and assigns higher weights to the paraphrases which are more likely to obtain

Input: what be the zip code of the largest car manufacturer	
what be the zip code of the largest vehicle manufacturer	PPDB
what be the zip code of the largest car producer	PPDB
what be the postal code of the biggest automobile manufacturer	NMT
what be the postcode of the biggest car manufacturer	NMT
what be the largest car manufacturer 's postal code	Rule
zip code of the largest car manufacturer	Rule

Table 1: Paraphrases obtained for an input question from different models (PPDB, NMT, Rule). Words are lowercased and stemmed.

the correct answers. These paraphrases and the original question simultaneously serve as input to a QA model that predicts a distribution over answers for a given question. Finally, the results of these two models are fused to predict the answer. In the following we will explain how $p\left(q'|q\right)$ and $p\left(a|q'\right)$ are estimated.

2.1 Paraphrase Generation

As shown in Equation (1), the term $p\left(a|q\right)$ is the sum over q and its paraphrases H_q. Ideally, we would generate all the paraphrases of q. However, since this set could quickly become intractable, we restrict the number of candidate paraphrases to a manageable size. In order to increase the coverage and diversity of paraphrases, we employ three methods based on: (1) lexical and phrasal rules from the Paraphrase Database (Pavlick et al., 2015); (2) neural machine translation models (Sutskever et al., 2014; Bahdanau et al., 2015); and (3) paraphrase rules mined from clusters of related questions (Fader et al., 2014). We briefly describe these models below, however, there is nothing inherent in our framework that is specific to these, any other paraphrase generator could be used instead.

2.1.1 PPDB-based Generation

Bilingual pivoting (Bannard and Callison-Burch, 2005) is one of the most well-known approaches to paraphrasing; it uses bilingual parallel corpora to learn paraphrases based on techniques from phrase-based statistical machine translation (SMT, Koehn et al. 2003). The intuition is that two English strings that translate to the same foreign string can be assumed to have the same meaning. The method first extracts a bilingual phrase table and then obtains English paraphrases by pivoting through foreign language phrases.

Drawing inspiration from syntax-based SMT, Callison-Burch (2008) and Ganitkevitch et al. (2011) extended this idea to syntactic paraphrases, leading to the creation of PPDB (Ganitkevitch et al., 2013), a large-scale paraphrase database containing over a billion of paraphrase pairs in 24 different languages. Pavlick et al. (2015) further used a supervised model to automatically label paraphrase pairs with entailment relationships based on natural logic (MacCartney, 2009). In our work, we employ bidirectionally entailing rules from PPDB. Specifically, we focus on lexical (single word) and phrasal (multiword) rules which we use to paraphrase questions by replacing words and phrases in them. An example is shown in Table 1 where we substitute *car* with *vehicle* and *manufacturer* with *producer*.

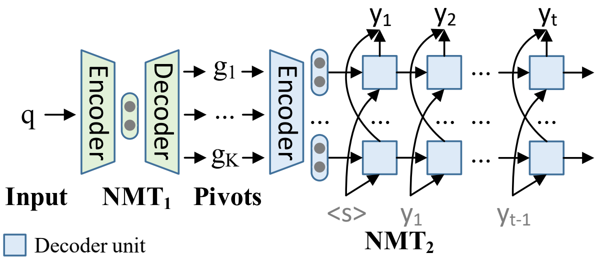


Figure 2: Overview of NMT-based paraphrase generation. NMT$_1$ (green) translates question q into pivots $g_1 \ldots g_K$ which are then back-translated by NMT$_2$ (blue) where K decoders jointly predict tokens at each time step, rather than only conditioning on one pivot and independently predicting outputs.

2.1.2 NMT-based Generation

Mallinson et al. (2016) revisit bilingual pivoting in the context of neural machine translation (NMT, Sutskever et al. 2014; Bahdanau et al. 2015) and present a paraphrasing model based on neural networks. At its core, NMT is trained end-to-end to maximize the conditional probability of a correct translation given a source sentence, using a bilingual corpus. Paraphrases can be obtained by translating an English string into a foreign language and then back-translating it into English. NMT-based pivoting models offer advantages over conventional methods such as the ability to learn continuous representations and to consider wider context while paraphrasing.

In our work, we select German as our pivot following Mallinson et al. (2016) who show that it outperforms other languages in a wide range of paraphrasing experiments, and pretrain two NMT systems, English-to-German (EN-DE) and

Source	Target
the average size of __	what be __ average size
__ be locate on which continent	what continent be __ a part of
language speak in __	what be the official language of __
what be the money in __	what currency do __ use

Table 2: Examples of rules used in the rule-based paraphrase generator.

German-to-English (DE-EN). A naive implementation would translate a question to a German string and then back-translate it to English. However, using only one pivot can lead to inaccuracies as it places too much faith on a single translation which may be wrong. Instead, we translate from multiple pivot sentences (Mallinson et al., 2016). As shown in Figure 2, question q is translated to K-best German pivots, $\mathcal{G}_q = \{g_1, \ldots, g_K\}$. The probability of generating paraphrase $q' = y_1 \ldots y_{|q'|}$ is decomposed as:

$$\begin{aligned} p\left(q' | \mathcal{G}_q\right) &= \prod_{t=1}^{|q'|} p\left(y_t | y_{<t}, \mathcal{G}_q\right) \\ &= \prod_{t=1}^{|q'|} \sum_{k=1}^{K} p\left(g_k | q\right) p\left(y_t | y_{<t}, g_k\right) \end{aligned} \tag{2}$$

where $y_{<t} = y_1, \ldots, y_{t-1}$, and $|q'|$ is the length of q'. Probabilities $p\left(g_k|q\right)$ and $p\left(y_t|y_{<t}, g_k\right)$ are computed by the EN-DE and DE-EN models, respectively. We use beam search to decode tokens by conditioning on multiple pivoting sentences. The results with the best decoding scores are considered candidate paraphrases. Examples of NMT paraphrases are shown in Table 1.

Compared to PPDB, NMT-based paraphrases are syntax-agnostic, operating on the surface level without knowledge of any underlying grammar. Furthermore, paraphrase rules are captured implicitly and cannot be easily extracted, e.g., from a phrase table. As mentioned earlier, the NMT-based approach has the potential of performing major rewrites as paraphrases are generated while considering wider contextual information, whereas PPDB paraphrases are more local, and mainly handle lexical variation.

2.1.3 Rule-Based Generation

Our third paraphrase generation approach uses rules mined from the WikiAnswers corpus (Fader et al., 2014) which contains more than 30 million question clusters labeled as paraphrases by WikiAnswers[1] users. This corpus is a large resource (the average cluster size is 25), but is relatively noisy due to its collaborative nature – 45% of question pairs are merely related rather than genuine paraphrases. We therefore followed the method proposed in (Fader et al., 2013) to harvest paraphrase rules from the corpus. We first extracted question templates (i.e., questions with at most one wild-card) that appear in at least ten clusters. Any two templates co-occurring (more than five times) in the same cluster and with the same arguments were deemed paraphrases. Table 2 shows examples of rules extracted from the corpus. During paraphrase generation, we consider substrings of the input question as arguments, and match them with the mined template pairs. For example, the stemmed input question in Table 1 can be paraphrased using the rules ("*what be the zip code of __*", "*what be __ 's postal code*") and ("*what be the zip code of __*", "*zip code of __*"). If no exact match is found, we perform fuzzy matching by ignoring stop words in the question and templates.

2.2 Paraphrase Scoring

Recall from Equation (1) that $p_\theta\left(q'|q\right)$ scores the generated paraphrases $q' \in H_q \cup \{q\}$. We estimate $p_\theta\left(q'|q\right)$ using neural networks given their successful application to paraphrase identification tasks (Socher et al., 2011; Yin and Schütze, 2015; He et al., 2015). Specifically, the input question and its paraphrases are encoded as vectors. Then, we employ a neural network to obtain the score $s\left(q'|q\right)$ which after normalization becomes the probability $p_\theta\left(q'|q\right)$.

Encoding Let $q = q_1 \ldots q_{|q|}$ denote an input question. Every word is initially mapped to a d-dimensional vector. In other words, vector $\mathbf{q}_t$ is computed via $\mathbf{q}_t = \mathbf{W}_q \mathbf{e}\left(q_t\right)$, where $\mathbf{W}_q \in \mathbb{R}^{d \times |\mathcal{V}|}$ is a word embedding matrix, $|\mathcal{V}|$ is the vocabulary size, and $\mathbf{e}\left(q_t\right)$ is a one-hot vector. Next, we use a bi-directional recurrent neural network with long short-term memory units (LSTM, Hochreiter and Schmidhuber 1997) as the question encoder, which is shared by the input questions and their paraphrases. The encoder recursively processes tokens one by one, and uses the encoded vectors to represent questions. We compute the hidden vectors at the t-th time step via:

[1] wiki.answers.com

$$\begin{aligned}\overrightarrow{\mathbf{h}}_t &= \text{LSTM}\left(\overrightarrow{\mathbf{h}}_{t-1}, \mathbf{q}_t\right), t = 1, \ldots, |q| \\ \overleftarrow{\mathbf{h}}_t &= \text{LSTM}\left(\overleftarrow{\mathbf{h}}_{t+1}, \mathbf{q}_t\right), t = |q|, \ldots, 1\end{aligned} \tag{3}$$

where $\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t \in \mathbb{R}^n$. In this work we follow the LSTM function described in Pham et al. (2014). The representation of q is obtained by:

$$\mathbf{q} = \left[\overrightarrow{\mathbf{h}}_{|q|}, \overleftarrow{\mathbf{h}}_1\right] \tag{4}$$

where $[\cdot, \cdot]$ denotes concatenation, and $\mathbf{q} \in \mathbb{R}^{2n}$.

Scoring After obtaining vector representations for q and q', we compute the score $s(q'|q)$ via:

$$s(q'|q) = \mathbf{w}_s \cdot [\mathbf{q}, \mathbf{q}', \mathbf{q} \odot \mathbf{q}'] + b_s \tag{5}$$

where $\mathbf{w}_s \in \mathbb{R}^{6n}$ is a parameter vector, $[\cdot, \cdot, \cdot]$ denotes concatenation, $\odot$ is element-wise multiplication, and b_s is the bias. Alternative ways to compute $s(q'|q)$ such as dot product or with a bilinear term were not empirically better than Equation (5) and we omit them from further discussion.

Normalization For paraphrases $q' \in H_q \cup \{q\}$, the probability $p_\theta(q'|q)$ is computed via:

$$p_\theta(q'|q) = \frac{\exp\{s(q'|q)\}}{\sum_{r \in H_q \cup \{q\}} \exp\{s(r|q)\}} \tag{6}$$

where the paraphrase scores are normalized over the set $H_q \cup \{q\}$.

2.3 QA Models

The framework defined in Equation (1) is relatively flexible with respect to the QA model being employed as long as it can predict $p_\psi(a|q')$. We illustrate this by performing experiments across different tasks and describe below the models used for these tasks.

Knowledge Base QA In our first task we use the Freebase knowledge base to answer questions. Query graphs for the questions typically contain more than one predicate. For example, to answer the question "*who is the ceo of microsoft in 2008*", we need to use one relation to query "*ceo of microsoft*" and another relation for the constraint "*in 2008*". For this task, we employ the SIMPLEGRAPH model described in Reddy et al. (2016, 2017), and follow their training protocol and feature design. In brief, their method uses rules to convert questions to ungrounded logical forms, which are subsequently matched against Freebase subgraphs. The QA model learns from question-answer pairs: it extracts features for pairs of questions and Freebase subgraphs, and uses a logistic regression classifier to predict the probability that a candidate answer is correct. We perform entity linking using the Freebasee/KG API on the original question (Reddy et al., 2016, 2017), and generate candidate Freebase subgraphs. The QA model estimates how likely it is for a subgraph to yield the correct answer.

Answer Sentence Selection Given a question and a collection of relevant sentences, the goal of this task is to select sentences which contain an answer to the question. The assumption is that correct answer sentences have high semantic similarity to the questions (Yu et al., 2014; Yang et al., 2015; Miao et al., 2016). We use two bidirectional recurrent neural networks (BILSTM) to separately encode questions and answer sentences to vectors (Equation (4)). Similarity scores are computed as shown in Equation (5), and then squashed to $(0, 1)$ by a sigmoid function in order to predict $p_\psi(a|q')$.

2.4 Training and Inference

We use a log-likelihood objective for training, which maximizes the likelihood of the correct answer given a question:

$$\text{maximize} \sum_{(q,a) \in \mathcal{D}} \log p(a|q) \tag{7}$$

where $\mathcal{D}$ is the set of all question-answer training pairs, and $p(a|q)$ is computed as shown in Equation (1). For the knowledge base QA task, we predict how likely it is that a subgraph obtains the correct answer, and the answers of some candidate subgraphs are partially correct. So, we use the binary cross entropy between the candidate subgraph's F1 score and the prediction as the objective function. The RMSProp algorithm (Tieleman and Hinton, 2012) is employed to solve this non-convex optimization problem. Moreover, dropout is used for regularizing the recurrent neural networks (Pham et al., 2014).

At test time, we generate paraphrases for the question q, and then predict the answer by:

$$\hat{a} = \arg\max_{a' \in \mathcal{C}_q} p(a'|q) \tag{8}$$

where $\mathcal{C}_q$ is the set of candidate answers, and $p(a'|q)$ is computed as shown in Equation (1).

3 Experiments

We compared our model which we call PARA4QA (as shorthand for learning to paraphrase for question answering) against multiple previous systems on three datasets. In the following we introduce these datasets, provide implementation details for our model, describe the systems used for comparison, and present our results.

3.1 Datasets

Our model was trained on three datasets, representative of different types of QA tasks. The first two datasets focus on question answering over a structured knowledge base, whereas the third one is specific to answer sentence selection.

WEBQUESTIONS This dataset (Berant et al., 2013) contains $3,778$ training instances and $2,032$ test instances. Questions were collected by querying the Google Suggest API. A breadth-first search beginning with *wh-* was conducted and the answers were crowd-sourced using Freebase as the backend knowledge base.

GRAPHQUESTIONS The dataset (Su et al., 2016) contains $5,166$ question-answer pairs (evenly split into a training and a test set). It was created by asking crowd workers to paraphrase 500 Freebase graph queries in natural language.

WIKIQA This dataset (Yang et al., 2015) has $3,047$ questions sampled from Bing query logs. The questions are associated with $29,258$ candidate answer sentences, $1,473$ of which contain the correct answers to the questions.

3.2 Implementation Details

Paraphrase Generation Candidate paraphrases were stemmed (Minnen et al., 2001) and lowercased. We discarded duplicate or trivial paraphrases which only rewrite stop words or punctuation. For the NMT model, we followed the implementation[2] and settings described in Mallinson et al. (2016), and used English↔German as the language pair. The system was trained on data released as part of the WMT15 shared translation task (4.2 million sentence pairs). We also had access to back-translated monolingual training data (Sennrich et al., 2016a). Rare words were split into subword units (Sennrich et al., 2016b) to handle out-of-vocabulary words in questions. We used the top 15 decoding results as candidate paraphrases. We used the S size package of PPDB 2.0 (Pavlick et al., 2015) for high precision. At most 10 candidate paraphrases were considered. We mined paraphrase rules from WikiAnswers (Fader et al., 2014) as described in Section 2.1.3. The extracted rules were ranked using the pointwise mutual information between template pairs in the WikiAnswers corpus. The top 10 candidate paraphrases were used.

Training For the paraphrase scoring model, we used GloVe (Pennington et al., 2014) vectors[3] pretrained on Wikipedia 2014 and Gigaword 5 to initialize the word embedding matrix. We kept this matrix fixed across datasets. Out-of-vocabulary words were replaced with a special unknown symbol. We also augmented questions with start-of- and end-of-sequence symbols. Word vectors for these special symbols were updated during training. Model hyperparameters were validated on the development set. The dimensions of hidden vectors and word embeddings were selected from $\{50, 100, 200\}$ and $\{100, 200\}$, respectively. The dropout rate was selected from $\{0.2, 0.3, 0.4\}$. The BILSTM for the answer sentence selection QA model used the same hyperparameters. Parameters were randomly initialized from a uniform distribution $\mathcal{U}(-0.08, 0.08)$. The learning rate and decay rate of RMSProp were 0.01 and 0.95, respectively. The batch size was set to 150. To alleviate the exploding gradient problem (Pascanu et al., 2013), the gradient norm was clipped to 5. Early stopping was used to determine the number of epochs.

3.3 Paraphrase Statistics

Table 3 presents descriptive statistics on the paraphrases generated by the various systems across datasets (training set). As can be seen, the average paraphrase length is similar to the average length of the original questions. The NMT method generates more paraphrases and has wider coverage, while the average number and coverage of the other two methods varies per dataset. As a way of quantifying the extent to which rewriting takes place, we report BLEU (Papineni et al., 2002) and TER (Snover et al., 2006) scores between the original questions and their paraphrases. The NMT

[2] github.com/sebastien-j/LV_groundhog

[3] nlp.stanford.edu/projects/glove

Metric	GraphQ			WebQ			WikiQA		
	NMT	PPDB	Rule	NMT	PPDB	Rule	NMT	PPDB	Rule
avg($\|q\|$)	10.87			7.71			6.47		
avg($\|q'\|$)	10.87	12.40	10.51	8.13	8.55	7.54	6.60	7.85	7.15
avg($\#q'$)	13.85	3.02	2.50	13.76	0.71	7.74	13.95	0.62	5.64
Coverage (%)	99.67	73.52	31.16	99.87	35.15	83.61	99.89	31.04	63.12
BLEU (%)	42.33	67.92	54.23	35.14	56.62	42.37	32.40	54.24	40.62
TER (%)	39.18	14.87	38.59	45.38	19.94	43.44	46.10	17.20	48.59

Table 3: Statistics of generated paraphrases across datasets (training set). avg($|q|$): average question length; avg($|q'|$): average paraphrase length; avg($\#q'$): average number of paraphrases; coverage: the proportion of questions that have at least one candidate paraphrase.

method and the rules extracted from WikiAnswers tend to paraphrase more (i.e., have lower BLEU and higher TER scores) compared to PPDB.

3.4 Comparison Systems

We compared our framework to previous work and several ablation models which either do not use paraphrases or paraphrase scoring, or are not jointly trained.

The first baseline only uses the base QA models described in Section 2.3 (SimpleGraph and BiLSTM). The second baseline (AvgPara) does not take advantage of paraphrase scoring. The paraphrases for a given question are used while the QA model's results are directly averaged to predict the answers. The third baseline (DataAugment) employs paraphrases for data augmentation during training. Specifically, we use the question, its paraphrases, and the correct answer to automatically generate new training samples.

In the fourth baseline (SepPara), the paraphrase scoring model is separately trained on paraphrase classification data, without taking question-answer pairs into account. In the experiments, we used the Quora question paraphrase dataset[4] which contains question pairs and labels indicating whether they constitute paraphrases or not. We removed questions with more than 25 tokens and sub-sampled to balance the dataset. We used 90% of the resulting 275K examples for training, and the remaining for development. The paraphrase score $s(q'|q)$ (Equation (5)) was wrapped by a sigmoid function to predict the probability of a question pair being a paraphrase. A binary cross-entropy loss was used as the objective. The classification accuracy on the dev set was 80.6%.

Method	Average F1 (%)	
	GraphQ	WebQ
Sempre (Berant et al., 2013)	10.8	35.7
Jacana (Yao and Van Durme, 2014)	5.1	33.0
ParaSemp (Berant and Liang, 2014)	12.8	39.9
SubGraph (Bordes et al., 2014a)	-	40.4
MCCNN (Dong et al., 2015)	-	40.8
Yao15 (Yao, 2015)	-	44.3
AgendaIL (Berant and Liang, 2015)	-	49.7
STAGG (Yih et al., 2015)	-	48.4 (52.5)
MCNN (Xu et al., 2016)	-	47.0 (53.3)
TypeRerank (Yavuz et al., 2016)	-	**51.6** (52.6)
BiLayered (Narayan et al., 2016)	-	47.2
UDepLambda (Reddy et al., 2017)	**17.6**	49.5
SimpleGraph (baseline)	15.9	48.5
AvgPara	16.1	48.8
SepPara	18.4	49.6
DataAugment	16.3	48.7
Para4QA	**20.4**	**50.7**
−NMT	18.5	49.5
−PPDB	19.5	50.4
−Rule	19.4	49.1

Table 4: Model performance on GraphQuestions and WebQuestions. Results with additional task-specific resources are shown in parentheses. The base QA model is SimpleGraph. Best results in each group are shown in **bold**.

Finally, in order to assess the individual contribution of different paraphrasing resources, we compared the Para4QA model against versions of itself with one paraphrase generator removed (−NMT/−PPDB/−Rule).

3.5 Results

We first discuss the performance of Para4QA on GraphQuestions and WebQuestions. The first block in Table 4 shows a variety of systems previously described in the literature using average F1 as the evaluation metric (Berant et al., 2013). Among these, ParaSemp, SubGraph, MCCNN, and BiLayered utilize paraphrasing resources. The second block compares Para4QA against various related baselines (see Section 3.4). SimpleGraph results on WebQuestions and GraphQuestions are taken from Reddy et al. (2016) and Reddy et al. (2017), respectively.

Overall, we observe that Para4QA outperforms baselines which either do not employ paraphrases (SimpleGraph) or paraphrase scoring (AvgPara, DataAugment), or are not jointly trained (SepPara). On GraphQuestions, our model Para4QA outperforms the previous state of the art by a wide margin. Ablation experiments with one of the paraphrase generators removed

[4] goo.gl/kMP46n

Method	MAP	MRR
BIGRAMCNN (Yu et al., 2014)	0.6190	0.6281
BIGRAMCNN+CNT (Yu et al., 2014)	0.6520	0.6652
PARAVEC (Le and Mikolov, 2014)	0.5110	0.5160
PARAVEC+CNT (Le and Mikolov, 2014)	0.5976	0.6058
LSTM (Miao et al., 2016)	0.6552	0.6747
LSTM+CNT (Miao et al., 2016)	0.6820	0.6988
NASM (Miao et al., 2016)	0.6705	0.6914
NASM+CNT (Miao et al., 2016)	0.6886	0.7069
KVMEMNET+CNT (Miller et al., 2016)	**0.7069**	**0.7265**
BILSTM (baseline)	0.6456	0.6608
AVGPARA	0.6587	0.6753
SEPPARA	0.6613	0.6765
DATAAUGMENT	0.6578	0.6736
PARA4QA	**0.6759**	**0.6918**
−NMT	0.6528	0.6680
−PPDB	0.6613	0.6767
−RULE	0.6553	0.6756
BILSTM+CNT (baseline)	0.6722	0.6877
PARA4QA+CNT	**0.6978**	**0.7131**

Table 5: Model performance on WIKIQA. +CNT: word matching features introduced in Yang et al. (2015). The base QA model is BILSTM. Best results in each group are shown in **bold**.

show that performance drops most when the NMT paraphrases are not used on GRAPHQUESTIONS, whereas on WEBQUESTIONS removal of the rule-based generator hurts performance most. One reason is that the rule-based method has higher coverage on WEBQUESTIONS than on GRAPHQUESTIONS (see Table 3).

Results on WIKIQA are shown in Table 5. We report MAP and MMR which evaluate the relative ranks of correct answers among the candidate sentences for a question. Again, we observe that PARA4QA outperforms related baselines (see BILSTM, DATAAUGMENT, AVGPARA, and SEPPARA). Ablation experiments show that performance drops most when NMT paraphrases are removed. When word matching features are used (see +CNT in the third block), PARA4QA reaches state of the art performance.

Examples of paraphrases and their probabilities $p_\theta(q'|q)$ (see Equation (6)) learned by PARA4QA are shown in Table 6. The two examples are taken from the development set of GRAPHQUESTIONS and WEBQUESTIONS, respectively. We also show the Freebase relations used to query the correct answers. In the first example, the original question cannot yield the correct answer because of the mismatch between the question and the knowledge base. The paraphrase contains "*role*" in place of "*sort of part*", increasing the chance of overlap between the question and the predicate words. The second question contains an informal expression "*play 4*", which confuses the QA model. The paraphrase model generates "*play for*" and predicts a high paraphrase score for it. More generally, we observe that the model tends to give higher probabilities $p_\theta(q'|q)$ to paraphrases biased towards delivering appropriate answers.

Examples	$p_\theta(q'\|q)$
(music.concert_performance.performance_role)	
what sort of part do queen play in concert	0.0659
what role do queen play in concert	0.0847
what be the role play by the queen in concert	0.0687
what role do queen play during concert	0.0670
what part do queen play in concert	0.0664
which role do queen play in concert concert	0.0652
(sports.sports_team_roster.team)	
what team do shaq play 4	0.2687
what team do shaq play for	0.2783
which team do shaq play with	0.0671
which team do shaq play out	0.0655
which team have you play shaq	0.0650
what team have we play shaq	0.0497

Table 6: Questions and their top-five paraphrases with probabilities learned by the model. The Freebase relations used to query the correct answers are shown in brackets. The original question is underlined. Questions with incorrect predictions are in *red*.

We also analyzed which structures were mostly paraphrased within a question. We manually inspected 50 (randomly sampled) questions from the development portion of each dataset, and their three top scoring paraphrases (Equation (5)). We grouped the most commonly paraphrased structures into the following categories: a) question words, i.e., wh-words and and "*how*"; b) question focus structures, i.e., cue words or cue phrases for an answer with a specific entity type (Yao and Van Durme, 2014); c) verbs or noun phrases indicating the relation between the question topic entity and the answer; and d) structures requiring aggregation or imposing additional constraints the answer must satisfy (Yih et al., 2015). In the example "*which year did Avatar release in UK*", the question word is "*which*", the question focus is "*year*", the verb is "*release*", and "*in UK*" constrains the answer to a specific location.

Figure 3 shows the degree to which different types of structures are paraphrased. As can be seen, most rewrites affect Relation Verb, especially on WEBQUESTIONS. Question Focus, Relation NP, and Constraint & Aggregation are more

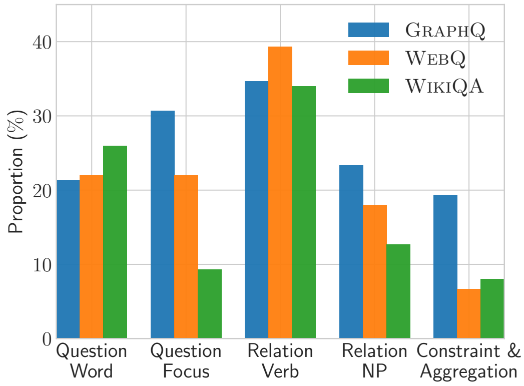


Figure 3: Proportion of linguistic phenomena subject to paraphrasing within a question.

Method	Average F1 (%)	
	Simple	Complex
SimpleGraph	20.9	12.2
Para4QA	**27.4** (+6.5)	**16.0** (+3.8)

Table 7: We group GraphQuestions into simple and complex questions and report model performance in each split. Best results in each group are shown in **bold**. The values in brackets are absolute improvements of average F1 scores.

often rewritten in GraphQuestions compared to the other datasets.

Finally, we examined how our method fares on simple versus complex questions. We performed this analysis on GraphQuestions as it contains a larger proportion of complex questions. We consider questions that contain a single relation as simple. Complex questions have multiple relations or require aggregation. Table 7 shows how our model performs in each group. We observe improvements for both types of questions, with the impact on simple questions being more pronounced. This is not entirely surprising as it is easier to generate paraphrases and predict the paraphrase scores for simpler questions.

4 Conclusions

In this work we proposed a general framework for learning paraphrases for question answering. Paraphrase scoring and QA models are trained end-to-end on question-answer pairs, which results in learning paraphrases with a purpose. The framework is not tied to a specific paraphrase generator or QA system. In fact it allows to incorporate several paraphrasing modules, and can serve as a testbed for exploring their coverage and rewriting capabilities. Experimental results on three datasets show that our method improves performance across tasks. There are several directions for future work. The framework can be used for other natural language processing tasks which are sensitive to the variation of input (e.g., textual entailment or summarization). We would also like to explore more advanced paraphrase scoring models (Parikh et al., 2016; Wang and Jiang, 2016) as well as additional paraphrase generators since improvements in the diversity and the quality of paraphrases could also enhance QA performance.

Acknowledgments The authors gratefully acknowledge the support of the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1; Mallinson) and the European Research Council (award number 681760; Lapata).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 597–604, Ann Arbor, Michigan. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Rahul Bhagat and Eduard Hovy. 2013. What is a paraphrase? *Computational Linguistics*, 39(3):463–472.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar. Association for Computational Linguistics.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases - Volume 8724*, ECML PKDD 2014, pages 165–180, New York, NY, USA. Springer-Verlag New York, Inc.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 196–205, Honolulu, Hawaii. Association for Computational Linguistics.

Bo Chen, Le Sun, Xianpei Han, and Bo An. 2016. Sentence rewriting for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 766–777, Berlin, Germany. Association for Computational Linguistics.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 260–269, Beijing, China. Association for Computational Linguistics.

Pablo Duboue and Jennifer Chu-Carroll. 2006. Answering the question you wish they had asked: The impact of paraphrasing for question answering. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 33–36, New York City, USA. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618, Sofia, Bulgaria. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1156–1165, New York, NY, USA. ACM.

Juri Ganitkevitch, Chris Callison-Burch, Courtney Napoles, and Benjamin Van Durme. 2011. Learning sentential paraphrases from bilingual parallel corpora for text-to-text generation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1168–1179, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: the paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586, Lisbon, Portugal. Association for Computational Linguistics.

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

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54, Sapporo, Japan.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1188–1196.

Bill MacCartney. 2009. *Natural Language Inference*. Ph.D. thesis, Stanford University.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2016. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, Valencia, Spain.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1727–1736.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of english. *Natural Language Engineering*, 7(3):207–223.

Shashi Narayan, Siva Reddy, and Shay B Cohen. 2016. Paraphrase generation from Latent-Variable PCFGs for semantic parsing. In *Proceedings of the 9th International Natural Language Generation Conference*, pages 153–162, Edinburgh, Scotland.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of*

40th Annual Meeting of the Association for Computational Linguistics, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1310–1318, Atlanta, Georgia.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition*, pages 285–290.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. *arXiv preprint arXiv:1702.03196*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for qa evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572, Austin, Texas. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. Technical report.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1442–1451, San Diego, California. Association for Computational Linguistics.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2326–2336, Berlin, Germany. Association for Computational Linguistics.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 66–70, Denver, Colorado. Association for Computational Linguistics.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966, Baltimore, Maryland. Association for Computational Linguistics.

Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. 2016. Improving semantic parsing via answer type inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 149–159, Austin, Texas. Association for Computational Linguistics.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911, Denver, Colorado. Association for Computational Linguistics.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. In *NIPS Deep Learning Workshop*.

Temporal Information Extraction for Question Answering Using Syntactic Dependencies in an LSTM-based Architecture

Yuanliang Meng, Anna Rumshisky, Alexey Romanov
{ymeng,arum,aromanov}@cs.uml.edu
Department of Computer Science
University of Massachusetts Lowell
Lowell, MA 01854

Abstract

In this paper, we propose to use a set of simple, uniform in architecture LSTM-based models to recover different kinds of temporal relations from text. Using the shortest dependency path between entities as input, the same architecture is implemented to extract intra-sentence, cross-sentence, and document creation time relations. A "double-checking" technique reverses entity pairs in classification, boosting the recall of positive cases and reducing misclassifications between opposite classes. An efficient pruning algorithm resolves conflicts globally. Evaluated on QA-TempEval (SemEval2015 Task 5), our proposed technique outperforms state-of-the-art methods by a large margin. We also conduct intrinsic evaluation and post state-of-the-art results on Timebank-Dense.

1 Introduction

Recovering temporal information from text is essential to many text processing tasks that require deep language understanding, such as answering questions about the timeline of events or automatically producing text summaries. This work presents intermediate results of an effort to build a temporal reasoning framework with contemporary deep learning techniques.

Until recently, there has been remarkably few attempts to evaluate temporal information extraction (TemporalIE) methods in context of downstream applications that require reasoning over the temporal representation. One recent effort to conduct such evaluation was SemEval2015 Task 5, a.k.a. QA-TempEval (Llorens et al., 2015a), which used question answering (QA) as the target application. QA-TempEval evaluated systems producing TimeML (Pustejovsky et al., 2003) annotation based on how well their output could be used in QA. We believe that application-based evaluation of TemporalIE should eventually completely replace the intrinsic evaluation if we are to make progress, and therefore we evaluated our techniques mainly using QA-TempEval setup.

Despite the recent advances produced by multi-layer neural network architectures in a variety of areas, the research community is still struggling to make neural architectures work for linguistic tasks that require long-distance dependencies (such as discourse parsing or coreference resolution). Our goal was to see if a relatively simple architecture with minimal capacity for retaining information was able to incorporate the information required to identify temporal relations in text.

Specifically, we use several simple LSTM-based components to recover ordering relations between temporally relevant entities (events and temporal expressions). These components are fairly uniform in their architecture, relying on dependency relations recovered with a very small number of mature, widely available processing tools, and require minimal engineering otherwise. To our knowledge, this is the first attempt to apply such simplified techniques to the TemporalIE task, and we demonstrate this streamlined architecture is able to outperform state-of-the-art results on a temporal QA task with a large margin.

In order to demonstrate generalizability of our proposed architecture, we also evaluate it intrinsically using TimeBank-Dense[1] (Chambers et al., 2014). TimeBank-Dense annotation aims to approximate a complete temporal relation graph by including all intra-sentential relations, all relations between adjacent sentences, and all relations with document creation time. Although our system

[1] https://www.usna.edu/Users/cs/nchamber/caevo/#corpus

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 887–896
Copenhagen, Denmark, September 7–11, 2017.

was not optimized for such a paradigm, and this data is quite different in terms of both the annotation scheme and the evaluation method, we obtain state-of-the-art results on this corpus as well.

2 Related Work

A multitude of TemporalIE systems have been developed over the past decade both in response to the series of shared tasks organized by the community (Verhagen et al., 2007, 2010; UzZaman et al., 2012; Sun et al., 2013; Bethard et al., 2015; Llorens et al., 2015b; Minard et al., 2015) and in standalone efforts (Chambers et al., 2014; Mirza, 2016).

The best methods used by TemporalIE systems to date tend to rely on highly engineered task-specific models using traditional statistical learning, typically used in succession (Sun et al., 2013; Chambers et al., 2014). For example, in a recent QA-TempEval shared task, the participants routinely used a series of classifiers (such as support vector machine (SVM) or hidden Markov chain SVM) or hybrid methods combining hand crafted rules and SVM, as was used by the top system in that challenge (Mirza and Minard, 2015). While our method also relies on decomposing the temporal relation extraction task into subtasks, we use essentially the same simple LSTM-based architecture for different components, that consume a highly simplified representation of the input.

Although there has not been much work applying deep learning techniques to TemporalIE, some relevant work has been done on a similar (but typically more local) task of relation extraction. Convolutional neural networks (Zeng et al., 2014) and recurrent neural networks both have been used for argument relation classification and similar tasks (Zhang and Wang, 2015; Xu et al., 2015; Vu et al., 2016). We take inspiration from some of this work, including specifically the approach proposed by Xu et al. (2015) which uses syntactic dependencies.

3 Dataset

We used QA-TempEval (SemEval 2015 Task 5)[2] data and evaluation methods in our experiments. The training set contains 276 annotated TimeML files, mostly news articles from major agencies or Wikinews from late 1990s to early 2000s. This data contains annotations for events, temporal expressions (referred to as TIMEXes), and temporal relations (referred to as TLINKs). The test set contains unannotated files in three genres: 10 news articles composed in 2014, 10 Wikipedia articles about world history, and 8 blogs entries from early 2000s.

In QA-TempEval, evaluation is done via a QA toolkit which contains yes/no questions about temporal relations between two events or an event and a temporal expression. QA evaluation is not available for most of the training data except for 25 files, for which 79 questions are available. We used used this subset of the training data for validation. The test set contains unannotated files, so QA is the only way to measure the performance. The total of 294 questions is available for the test data, see Table 6.

We also use TimeBank-Dense dataset, which contains a subset of the documents in QA-TempEval. In TimeBank-Dense, all entity pairs in the same sentence or in consecutive sentences are labeled. If there is no information about the relation between two entities, it is labeled as “vague”. We follow the experimental setup in (Chambers et al., 2014), which splits the corpus into training/validation/test sets of 22, 5, and 9 documents, respectively.

4 TIMEX and Event Extraction

The first task in our TemporalIE pipeline (TEA) is to identify time expressions (TIMEXes) and events in text. We utilized the HeidelTime package (Strötgen and Gertz, 2013) to identify TIMEXes. We trained a neural network model to identify event mentions. Contrary to common practice in TemporalIE, our models do not rely on event attributes, and thus we did not attempt to identify them.

Feature	Explanation
is_main_verb	whether the token is the main verb of a sentence
is_predicate	whether the token is the predicate of a phrase
is_verb	whether the token is a verb
is_noun	whether the token is a noun

Table 1: Token features for event extraction

We perform tokenization, part-of-speech tagging, and dependency parsing using NewsReader (Agerri et al., 2014). Every token is represented with a set of features derived from preprocessing. Syntactic dependencies are not used for event extraction, but are used later in the pipeline for

[2] http://alt.qcri.org/semeval2015/task5/

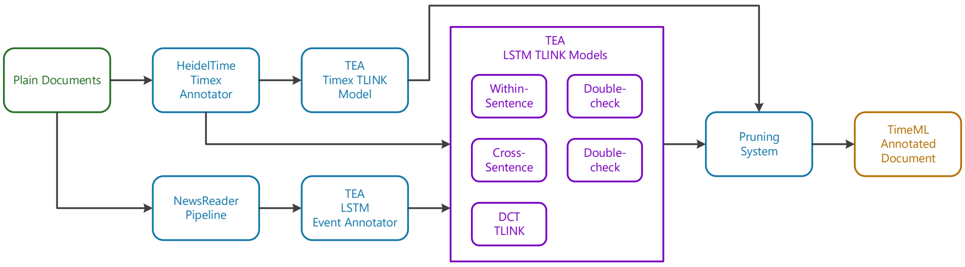


Figure 1: System overview for our temporal extraction annotator (TEA) system

TLINK classification. The features used to identify events are listed in Table 1.

The event extraction model uses long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), an RNN architecture well-suited for sequential data. The extraction model has two components, as shown on the right of Figure 2. One component is an LSTM layer which takes word embeddings as input. The other component takes 4 token-level features as input. These components produce hidden representations which are concatenated, and fed into an output layer which performs binary classification. For each token, we use four tokens on each side to represent the surrounding context. The resulting sequence of nine word embeddings is then used as input to an LSTM layer. If a word is near the edge of a sentence, zero padding is applied. We only use the token-level features of the target token, and ignore those from the context words. The 4 features are all binary, as shown in Table 1. Since the vast majority of event mentions in the training data are single words, we only mark single words as event mentions.

5 TLINK Classification

Our temporal relation (TLINK) classifier consists of four components: an LSTM-based model for intra-sentence entity relations, an LSTM-based model for cross-sentence relations, another LSTM-based model for relations with document creation time, and a rule-based component for TIMEX pairs. The four models perform TLINK classifications independently, and the combined results are fed into a pruning module to remove the conflicting TLINKs. The three LSTM-based components use the same streamlined architecture over token sequences recovered from shortest dependency paths between entity pairs.

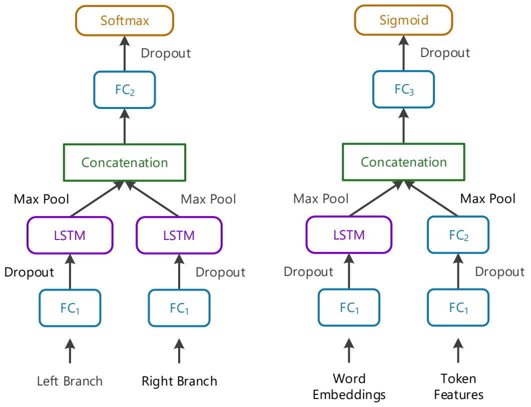


Figure 2: Model architecture. Left: intra-sentence and cross-sentence model. Right: Event extraction model.

5.1 Intra-Sentence Model

A TLINK extraction model should be able to learn the patterns that correspond to specific temporal relations, such as specific temporal prepositional phrases and clauses with temporal conjunctions. This suggests such models may benefit from encoding syntactic relations, rather than linear sequences of lexical items.

We use the shortest path between entities in a dependency tree to capture the essential context. Using the NewsReader pipeline, we identify the shortest path, and use the word embeddings for all tokens in the path as input to a neural network. Similar to previous work in relation extraction (Xu et al., 2015), we use two branches, where the left branch processes the path from the source entity to the least common ancestor (LCA), and the right branch processes the path from the target entity to the LCA. However, our TLINK extraction model uses only word embeddings as input, not POS tags, grammatical relations themselves, or WordNet hypernyms.

For example, for the sentence “Their marriage ended before the war”, given an event pair (*marriage*, *war*), the left branch of the model will receive the sequence (*marriage, ended*), while the right branch will receive (*war, before, ended*). The LSTM layer processes the appropriate sequence of word embeddings in each branch. This is followed by a separate max pooling layer for each branch, so for each LSTM unit, the maximum value over the time steps is used, not the final step value. During the early stages of model design, we observed that this max pooling approach (also used in Xu et al. (2015)) resulted in a slight improvement in performance. Finally, the results from the max pooling layers of both branches are concatenated and fed to a hidden layer, followed by a softmax to yield a probability distribution over the classes. The model architecture is shown in Figure 2 (left). We also augment the training data by flipping every pair, i.e. if $(e_1, e_2) \rightarrow$ BEFORE, $(e_2, e_1) \rightarrow$ AFTER is also included.

5.2 Cross-Sentence Model

TLINKs between the entities in consecutive sentences can often be identified without any external context or prior knowledge. For example, the order of events may be indicated by discourse connectives, or the events may follow natural order, potentially encoded in their word embeddings.

To recover such relations, we use a model similar to the one used for intra-sentence relations, as described in Section5.1. Since there is no common root between entities in different sentences, we use the path between an entity and the sentence root to construct input data. A sentence root is often the main verb, or a conjunction.

5.3 Relations to DCT

The document creation time (DCT) naturally serves as the "current time". In this section, we discuss how to identify temporal relations between an event and DCT. The assumption here is that an event mention and its local context can often suffice for DCT TLINKs. For example, English has inflected verbs for tense in finite clauses, and uses auxiliaries to express aspects.

The model we use is again similar to the one in Section5.2. Although one branch would suffice in this case, we use two branches in our implementation. One branch processes the path from a given entity to the sentence root, and the other branch processes the same path in reverse, from the root to the entity.

5.4 Relations between TIMEXes

Time expressions explicitly signify a time point or an interval of time. Without the TIMEX entities serving as "hubs", many events would be isolated from each other. We use rule-based techniques to identify temporal relations between TIMEX pairs that have been identified and normalized by HeidelTime. The relation between the DCT and other time expressions is just a special case of TIMEX-to-TIMEX TLINK and is handled with rules as well.

DATE value	**Calculation**	**Representation**

2017-08-04	START = $2017 + 7/12 + 3/365$
	$= 2017.591$
	END = START
		(2017.591, 2017.591)

2017-SU
(Summer 2017)
	START = $2017 + 5/12 = 2017.416$
	END = $2017 + 8/12 = 2017.666$
		(2017.416, 2017.666)

Table 2: Examples of DATE values and their tuple representations

In the present implementation, we focus on the DATE class of TIMEX tags, which is prevalent in the newswire text. The TIME class tags which contain more information are converted to DATE. Every DATE value is mapped to a tuple of real values (*start*, *end*). The "value" attribute of TIMEX tags follows the ISO-8601 standard, so the mapping is straightforward. Table 2 provides some examples. We set the minimum time interval to be a day. Practically, such a treatment suffices for our data. After mapping DATE values to tuples of real numbers, we can define 5 relations between TIMEX entities $T_1 = (start_1, end_1)$ and $T_2 = (start_2, end_2)$ as follows:

$$T_1 \times T_2 \rightarrow \begin{cases} \text{BEFORE} & \text{if } end_1 < start_2 \\ \text{AFTER} & \text{if } start_1 > end_2 \\ \text{INCLUDES} & \text{if } start_1 < start_2 \\ & \text{ and } end_1 > end_2 \\ \text{IS\_INCLUDED} & \text{if } start_1 > start_2 \\ & \text{ and } end_1 < end_2 \\ \text{SIMULTANEOUS} & \text{if } start_1 = start_2 \\ & \text{ and } end_1 = end_2 \end{cases} \tag{1}$$

The TLINKs from training data contain more types of relations than the five described in Equation 1. However relations such as IBEFORE ("immediately before"), IAFTER("immediately after") and IDENTITY are only used on event pairs, not TIMEX pairs. The QA system also does not target questions on TIMEX pairs. The purpose here is to use the TIMEX relations to link the otherwise isolated events.

6 Double-checking

A major difficulty we have is that the TLINKs for intra-sentence, cross-sentence, and DCT relations in the training data are not comprehensive. Often, the temporal relation between two entities is clear, but the training data provides no TLINK annotation. We downsampled the NO-LINK class in training in order to address both the class imbalance and the fact that TimeML-style annotation is de-facto sparse, with only a fraction of positive instances annotated.

In addition to that, we introduce a technique to boost the recall of positive classes (not NO-LINK) and to reduce the misclassification between the opposite classes. Since entity pairs are always classified in both orders, if both orders produce a TLINK relation, rather than NO-LINK, we adopt the label with a higher probability score, as assigned by the softmax classifier. We call this technique "double-checking". It serves to reduce the errors that are fundamentally harmful (e.g. BEFORE misclassified as AFTER, and vice versa). We also allow a positive class to have the "veto power" against NO-LINK class. For instance, if our model predicts $(e_1, e_2) \rightarrow$ AFTER but NO-LINK reversely, we adopt the former.

NO-LINK ratio	Recall	Recall	BEFORE	AFTER
	BEFORE	AFTER	as AFTER	as BEFORE
0.5	0.451	0.445	0.075	0.092
0.1	0.643	0.666	0.145	0.159
0.1 + double-check	0.721	0.721	0.125	0.125

Table 3: Effects of downsampling and double-checking on intra-sentence results. 0.5 NO-LINK ratio means that NO-LINKs are downsampled to a half of the number of all positive instances combined. BEFORE as AFTER shows the fraction of BEFORE misclassified as AFTER, and vice versa.

Table 3 shows the effects of double-checking and downsampling the NO-LINK cases on the intra-sentence model. Double-checking technique not only further boosts recall, but also reduces the misclassification between the opposite classes.

7 Pruning TLINKs

The four TLINK classification models in Section 5 deal with different kinds of TLINKs, so their output does not overlap. Nevertheless temporal relations are transitive in nature, so the deduced relations from given TLINKs can be in conflict.

Most conflicts arise from two types of relations, namely BEFORE/AFTER and INCLUDES/IS_INCLUDED. Naturally, we can convert TLINKs of opposite relations and put them all together. If we use a directed graph to represent the BEFORE relations between all entities, it should be acyclic. Sun (2014) proposed a strategy that "prefers the edges that can be inferred by other edges in the graph and remove the ones that are least so". Another strategy is to use the results from separate classifiers or "sieves" to rank TLINKs according to their confidence (Mani et al., 2007; Chambers et al., 2014). High-ranking results overwrite low-ranking ones.

We follow the same idea of purging the weak TLINKs. Given a directed graph, our approach is to remove the edges to break cycles, so that the sum of weights from the removed edges is minimal. This problem is actually an extension of the minimum feedback arc set problem and is NP-hard (Karp, 1972). We therefore adopt a heuristic-based approach, applied separately to the graphs induced by BEFORE/AFTER and INCLUDES/IS_INCLUDED relations.[3] The softmax layer provides a probability score for each relation class, which represents the strength of a link. TLINKs between TIMEX pairs are generated by rules, so we assume them to be reliable and assign them a score of 1. Although INCLUDES/IS_INCLUDED edges can generate conflicts in a BEFORE/AFTER graph as well, we currently do not resolve such conflicts because they are relatively rare. We also do not use SIMULTANEOUS/IDENTITY relations to merge nodes, because we found that it leads to very unstable results.

For a given relation (e.g., BEFORE), we incrementally build a directed graph with all edges representing that relation. We first initialize the graph with TIMEX-to-TIMEX relations. Event vertices are then added to this graph in a random order. For each event, we add all edges associated with it. If this creates a cycle, the edges are removed one by one until there is no cycle, keeping track of the sum of the scores associated with removed edges. We choose the order in which the edges are removed to minimize that value.[4] The algorithm is shown above.

In practice, the vertices do not have a high de-

[3] We found that ENDS and BEGINS TLINKs are too infrequent to warrant a separate treatment.

[4] By removing an edge, we mean resetting the relation to NO-LINK. Another possibility may be to set the relation associated with the edge to the one with the second highest probability score, however this may create additional cycles.

```
$X \leftarrow$ EVENTS;
$V \leftarrow$ TIMEXes;
$E \leftarrow$ TIMEX pairs;
Initialize $G \leftarrow < V, E >$;
for $x \in X$ do
    $V' \leftarrow V + \{x\}$;
    $C \leftarrow \{(x,v) \cup (v,x) | v \in V\}$ ;
    $E' \leftarrow E \cup C$ ;
    $G' \leftarrow < V', E' >$ ;
    if $cycle\_exists(G')$ then
        for $C_i \in \pi(C)$ do
            $score_i = 0$;
            while $C_i \neq \phi$ & $cycle\_exists(G \cup C_i)$
              do
                $c \leftarrow C_i.pop()$;
                $score_i += weight(c)$;
            end
        end
    end
    $G \leftarrow G \cup C_i$ s.t. $i = argmin(score_i)$;
end
```

Algorithm 1: Algorithm to prune edges. $\pi(C)$ denotes some permutations of C, where C is a list of weighted edges.

gree for a given relation, so permuting the candidates $N \times (N-1)$ times (i.e., not fully), where N is the number of candidates, produces only a negligible slowdown. We also make sure to try the greedy approach, dropping the edges with the smallest weights first.

8 Model Settings

In this section, we describe the model settings used in our experiments. All models requiring word embeddings use 300-dimensional word2vec vectors trained on Google News corpus (3 billion running words).[5] Our models are written in Keras on top of Theano.

TIMEX and Event Annotation The LSTM layer of the event extraction model contains 128 LSTM units. The hidden layer on top of that has 30 neurons. The input layer corresponding to the 4 token features is connected with a hidden layer with 3 neurons. The combined hidden layer is then connected with a single-neuron output layer. We set a dropout rate 0.5 on input layer, and another drop out rate 0.5 on the hidden layer before output.

As mentioned earlier, we do not attempt to tag event attributes. Since the vast majority of tokens are outside of event mention boundaries, we set higher weights for the positive class. In order to answer questions about temporal relations, it is not particularly harmful to introduce spurious events, but missing an event makes it impossible to answer any question related to it. Therefore we intentionally boost the recall while sacrificing precision. Table 4 shows the performance of our event extraction, as well as the performance of HeidelTime TIMEX tagging. For events, partial overlap of mention boundaries is considered an error.

Annotation	Prec	Rec	F1
TIMEX	0.838	0.850	0.844
Event	0.729	0.963	0.830

Table 4: TIMEX and event evaluation on validation set.

Intra-Sentence Model We identify 12 classes of temporal relations, plus a NO-LINK class. For training, we downsampled NO-LINK class to 10% of the number of positive instances. Our system does not attempt to resolve coreference. For the purpose of identifying temporal relations, SIMULTANEOUS and IDENTITY links capture the same relation of simultaneity, which allowed us to combine them. The LSTM layer of the intra-sentence model contains 256 LSTM units on each branch. The hidden layer on top of that has 100 neurons. We set a dropout rate 0.6 on input layer, and another drop out rate 0.5 on the hidden layer before output.

Cross-Sentence Model The training and evaluation procedures are very similar to what we did for intra-sentence models, and the hyperparameters for the neural networks are the same. Now the vast majority of entity pairs have no TLINKs explicitly marked in training data. Unlike the intra-sentence scenario, however, a NO-LINK label is truly adequate in most cases. We found that downsampling NO-LINK instances to match the number of all positive instances (ratio=1) yields desirable results. Since positive instances are very sparse in both the training and validation data, the ratio should not be too low, so as not to risk overfitting.

DCT Model We use the same hyperparameters for the DCT model as for the intra-sentence and cross-sentence models. Again, the training files do not sufficiently annotate TLINKs with DCT even if the relations are clear, so there are many false negatives. We downsample the NO-LINK instances so that they are 4 times the number of positive instances.

[5] https://github.com/mmihaltz/word2vec-GoogleNews-vectors

system	coverage	prec	rec	f1
human-fold1-original	0.43	0.91	0.38	0.54
human-fold1-timlinks	0.52	0.93	0.47	**0.62**
TIPSem-fold1-original	0.35	0.57	0.22	0.32
TIPSem-fold1-timex	0.53	0.69	0.38	0.50
orig. validation data	0.37	**0.93**	0.34	0.50
orig. tags TEA tlinks	0.81	0.58	0.47	0.52
TEA-initial	0.78	0.60	0.47	0.52
TEA-double-check	**0.89**	0.60	**0.53**	0.56
TEA-prune	0.82	0.58	0.48	0.53
TEA-flat	0.81	0.44	0.35	0.39
TEA-Dense	0.68	0.70	0.48	0.57
TEA-final	0.84	0.64	**0.53**	**0.58**

Table 5: QA results on validation data. There are **79** questions in total. The 4 systems on the top of the table are provided with the toolkit. The systems starting with "human-" are annotated by human experts. TEA-final utilizes both double-check and pruning. TEA-flat uses the flat context. TEA-Dense is trained on TimeBank-Dense.

9 Experiments

In this section, we first describe the model selection experiments on QA-TempEval validation data, selectively highlighting results of interest. We then present the results obtained with the optimized model on the QA-TempEval task and on TimeBank-Dense.

9.1 Model Selection Experiments

As mentioned before, "gold" TLINKs are sparse, so we cannot merely rely on the F1 scores on validation data to do model selection. Instead, we used the QA toolkit. The toolkit contains 79 yes-no questions about temporal relations between entities in the validation data. Originally, only 6 questions have "no" as the correct answer, and 1 question is listed as "unknown". After investigating the questions and answers, however, we found some errors and typos[6]. After fixing the errors, there are 7 no-questions and 72 yes-questions in total. All evaluations are performed on the fixed data.

The evaluation tool draws answers from the annotations only. If an entity (event or TIMEX) involved in a question is not annotated, or the TLINK cannot be found, the question will then be counted as not answered. There is no way for participants to give an answer directly, other than delivering the annotations. The program generates Timegraphs to infer relations from the annotated TLINKs. As a result, relations without explicit TLINK labels can still be used if they can be inferred from the annotations. The QA toolkit uses the following evaluation measures:

$$\text{coverage} = \frac{\#\text{answered}}{\#\text{questions}},\ \text{precision} = \frac{\#\text{correct}}{\#\text{answered}}$$

$$\text{recall} = \frac{\#\text{correct}}{\#\text{questions}},\ \text{f1} = \frac{2\times\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$$

Table 5 shows the results produced by different models on the validation data. The results of the four systems above the first horizontal line are provided by the task organizer. Among them, the top two use annotations provided by human experts. As we can see, the precision is very high, both above 0.90. Our models cannot reach that precision. In spite of the lower precision, automated systems can have much higher coverages i.e. answer a lot more questions.

As a starting point, we evaluated the validation files in their original form, and the results are shown as "orig. validation data" of Table 5. The precision was good, but with very low coverage. This supports our claim that the TLINKs provided by the training/validation files are not complete. We also tried using the event and TIMEX tags from the validation data, but performing TLINK classification with our system. As shown with "orig. tags TEA tlinks" in the table, now the coverage rises to 64 (or 0.81), and the overall F1 score reaches 0.52. The TEA-initial system uses our own annotators. The performance is similar, with a slight improvement in precision. This result shows our event and TIMEX tags work well, and are not inferior to the ones provided by the training data.

The double-checking technique boosts the coverage a lot, probably because we allow positive results to veto NO-LINKs. Combining double-checking with the pruning technique yields the best results, with F1 score 0.58, answering 42 out of 79 questions correctly.

In order to validate the choice of the dependency path-based context, we also experimented with a conventional flat context window, using the same hyperparameters. Every entity is represented by a 11-word window, with the entity mention in the middle. If two entities are near each other, their windows are cut short before reaching the other entity. Using the flat context instead of dependency paths yields a much weaker performance.

[6] Question 24 from XIE19980821.0077.tml should be answered with "yes", but the answer key contains a typo "is". Question 34 from APW19980219.0476.tml has BEFORE that should be replaced with AFTER. Question 29 from XIE19980821.0077.tml has "unknown" in the answer key, but after reading the article, we believe the correct answer is "no".

This confirms our hypothesis that syntactic dependencies represent temporal relations better than word windows. However, it should be noted that we did not separately optimize the models for the flat context setting. The large performance drop we saw from switching to flat context did not warrant performing a separate parameter search.

We also wanted to check whether a comprehensive annotation of TLINKs in the training data can improve model performance on the QA task. We therefore trained our model on TimeBank-Dense data and evaluated it with QA (see the TEA-Dense line in Table 5). Interestingly, the performance is nearly as good as our top model, although TimeBank-Dense only uses five major classes of relations. For one thing, it shows that our system may perform equally after being trained on sparsely labeled data and on densely labeled data, judged from the QA evaluation tool. If this is true, excessively annotated data may not be necessary in some tasks.

	doc	**words**	**quest**	**yes**	**no**	**dist-**	**dist+**
news	10	6920	99	93	6	40	59
wiki	10	14842	130	117	13	58	72
blogs	8	2053	65	65	0	30	35
total	28	23815	294	275	19	128	166

Table 6: Test data statistics. Adapted from Table 1 in Llorens et al. (2015a).

9.2 QA-TempEval Experiments

We use the QA toolkit provided by the QA-TempEval organizers to evaluate our system on the test data. The documents in test data are not annotated at all, so the event tags, TIMEX tags, and TLINKs are all created by our system.

Table 6 shows the the statistics of test data. As we can see, the vast majority of the questions in the test set should be answered with *yes*. Generally speaking, it is much more difficult to validate a specific relation (answer *yes*) than to reject it (answer *no*) when we have as many as 12 types of relations in addition to the vague NO-LINK class. **dist-** means questions involving entities that are in the same sentence or in consecutive sentences. **dist+** means the entities are farther away.

The QA-TempEval task organizers used two evaluation methods. The first method is exactly the same as the one we used on validation data. The second method used a so-called Time Expression Reasoner (TREFL) to add relations between TIMEXes, and evaluated the augmented results. The goal of such an extra run is to "analyze how a general time expression reasoner could improve results". Our model already includes a component to handle TIMEX relations, so we will compare our results with other systems' in both methods.

News Genre (99 questions)

system	**prec**	**rec**	**f1**	**% answd**	**# correct**
hlt-fbk-ev1-trel1	0.59	0.17	0.27	29	17
hlt-fbk-ev1-trel2	0.43	0.23	0.30	55	23
hlt-fbk-ev2-trel1	0.56	0.20	0.30	36	20
hlt-fbk-ev2-trel2	0.43	0.29	0.35	69	29
ClearTK	0.60	0.06	0.11	10	6
CAEVO	0.59	0.17	0.27	29	17
TIPSemB	0.50	0.16	0.24	32	16
TIPSem	0.52	0.11	0.18	21	11
TEA	**0.61**	**0.44**	**0.51**	**73**	**44**

Wikipedia Genre (130 questions)

system	**prec**	**rec**	**f1**	**% answd**	**# correct**
hlt-fbk-ev1-trel1	0.55	0.16	0.25	29	21
hlt-fbk-ev1-trel2	0.52	0.22	0.35	50	34
hlt-fbk-ev2-trel1	0.58	0.17	0.26	29	22
hlt-fbk-ev2-trel2	0.62	0.36	0.46	58	47
ClearTK	0.60	0.05	0.09	8	6
CAEVO	0.59	0.17	0.26	28	22
TIPSemB	0.52	0.13	0.21	25	17
TIPSem	**0.74**	0.19	0.30	26	25
TEA	0.62	**0.44**	**0.51**	**71**	**57**

Blog Genre (65 questions)

system	**prec**	**rec**	**f1**	**% answd**	**# correct**
hlt-fbk-ev1-trel1	**0.57**	0.18	**0.28**	32	12
hlt-fbk-ev1-trel2	0.43	0.18	0.26	43	12
hlt-fbk-ev2-trel1	0.47	0.14	0.21	29	9
hlt-fbk-ev2-trel2	0.34	**0.20**	0.25	**58**	**13**
ClearTK	0.56	0.08	0.14	14	5
CAEVO	0.48	0.18	0.27	38	12
TIPSemB	0.31	0.08	0.12	25	5
TIPSem	0.45	0.14	0.21	31	9
TEA	0.43	**0.20**	0.27	46	**13**

Table 7: QA evaluation on test data without TREFL

The results are shown in Table 7. We give the results for the hlt-fbk systems that were submitted by the top team. Among them, hlt-fbk-ev2-trel2 was the overall winner of TempEval task in 2015. ClearTK, CAEVO, TIPSEMB and TIPSem were some off-the-shelf systems provided by the task organizers for reference. These systems were not optimized for the task (Llorens et al., 2015a).

For news and Wikipedia genres, our system outperforms all other systems by a large margin. For blogs genre, however, the advantage of our system is unclear. Recall that our training set contains news articles only. While the trained model works well on Wikipedia dataset too, blog dataset is fundamentally different in the following ways: (1) each blog article is very short, (2) the style of writing in blogs is much more informal, with non-standard spelling and punctuation, and (3) blogs

All Genres (294 questions)

system	prec	rec	f1	% awd	# corr
hlt-fbk-ev2-trel2	0.49	0.30	0.37	62	89
hlt-fbk-ev2-trel2-TREFL	0.51	0.34	0.40	**67**	99
TEA	**0.59**	**0.39**	**0.47**	66	**114**
TEA-TREFL	0.58	0.38	0.46	66	111

Table 8: Test results over all genres.

are written in first person, and the content is usually personal stories and feelings.

Interestingly, the comparison between different hlt-fbk submissions suggests that resolving event coreference (implemented by hlt-fbk-ev2-trel2) substantially improves system performance for the news and Wikipedia genres. However, although our system does not attempt to handle event coreference explicitly, it easily outperforms the hlt-fbk-ev2-trel2 system in the genres where coreference seems to matter the most.

Evaluation with TREFL The extra evaluation with TREFL has a post-processing step that adds TLINKs between TIMEX entities. Our model already employs such a strategy, so this post-processing does not help. In fact, it drags down the scores a little. Table 8 summarizes the results over all genres before and after applying TREFL. For comparison, we include the top 2015 system, hlt-fbk-ev2-trel2. As we can see, TEA generally shows substantially higher scores.

9.3 TimeBank-Dense Experiments

We trained and evaluated the same system on TimeBank-Dense to see how it performs on a similar task with a different set of labels and another method of evaluation. In this experiment, we used the event and TIMEX tags from test data, as Mirza and Tonelli (2016).

Since all the NO-LINK (vague) relations are labeled, downsampling was not necessary. We did use double-checking in the final conflict resolution, but without giving positive cases the veto power over NO-LINK. Because NO-LINK relations dominate, especially for cross-sentence pairs, we set class weights to be inversely proportional to the class frequencies during training. We also reduced input batch size to counteract class imbalance.

We ran two sets of experiments. One used the uniform configurations for all the neural network models, similar to our experiments with QA-TempEval. The other tuned the hyperparameters for each component model (number of neurons, dropout rates, and early stop) separately.

system	ClearTK	NavyT	CAEVO	CATENA	TEA-Dense	
					uniform	tuned
F1	0.447	0.453	0.507	0.511	0.505	0.519

Table 9: TEA results on TimeBank-Dense. ClearTK, NavyT, and CAEVO are systems from Chambers et al. (2014). CATENA is from Mirza and Tonelli (2016)

The results from TimeBank-Dense are shown in Talble 9. Even though TimeBank-Dense has a very different methodology for both annotation and evaluation, our “out-of-the-box” model which uses uniform configurations across different components obtains F1 0.505, compared to the best F1 of 0.511 in previous work. Our best result of 0.519 is obtained by tuning hyperparameters on intra-sentence, cross-sentence, and DCT models independently.

For the QA-TempEval task, we intentionally tagged a lot of events, and let the pruning algorithm resolve potential conflicts. In the TimeBank-Dense experiment, however, we only used the provided event tags, which are sparser than what we have in QA-TempEval. The system may have lost some leverage that way.

10 Conclusion

We have proposed a new method for extraction of temporal relations which takes a relatively simple LSTM-based architecture, using shortest dependency paths as input, and re-deploys it in a set of subtasks needed for extraction of temporal relations from text. We also introduce two techniques that leverage confidence scores produced by different system components to substantially improve the results of TLINK classification: (1) a “double-checking” technique which reverses pairs in classification, thus boosting the recall of positives and reducing misclassifications among opposite classes and (2) an efficient pruning algorithm to resolve TLINK conflicts. In a QA-based evaluation, our proposed method outperforms state-of-the-art methods by a large margin. We also obtain state-of-the art results in an intrinsic evaluation on a very different TimeBank-Dense dataset, proving generalizability of the proposed model.

Acknowledgments

This project is funded in part by an NSF CAREER Award to A. Rumshisky (IIS-1652742). We would like to thank Connor Cooper and Kevin Wacome for their contributions to the early stages of this work.

References

Rodrigo Agerri, Josu Bermudez, and German Rigau. 2014. Ixa pipeline: Efficient and ready to use multilingual nlp tools. In *Proc. of the 9th Language Resources and Evaluation Conference (LREC2014)*, pages 26–31.

Steven Bethard, Leon Derczynski, James Pustejovsky, and Marc Verhagen. 2015. Semeval-2015 task 6: Clinical tempeval. In *Proc. of the 9th International Workshop on Semantic Evaluation (SemEval 2015). Association for Computational Linguistics*.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

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

Richard Karp. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations, Proc. Sympos.*, pages 85–103.

Hector Llorens, Nathanael Chambers, Naushad UzZaman, Nasrin Mostafazadeh, James Allen, and James Pustejovsky. 2015a. Semeval-2015 task 5: Qa tempeval - evaluating temporal information understanding with question answering. In *Proc. of the 9th International Workshop on Semantic Evaluation*, pages 46–54. Association for Computational Linguistics.

Hector Llorens, Nathanael Chambers, Naushad UzZaman, Nasrin Mostafazadeh, James Allen, and James Pustejovsky. 2015b. Semeval-2015 task 5: Qa tempeval-evaluating temporal information understanding with question answering. In *Proc. of the International Workshop on Semantic Evaluation (SemEval-2015)*.

Inderjeet Mani, Ben Wellner, Marc Verhagen, and James Pustejovsky. 2007. Three approaches to learning tlinks in timeml. *Technical Report CS-07–268, Computer Science Department*.

Anne-Lyse Minard, Manuela Speranza, Eneko Agirre, Itziar Aldabe, Marieke van Erp, Bernardo Magnini, German Rigau, Rubén Urizar, and Fondazione Bruno Kessler. 2015. Semeval-2015 task 4: Timeline: Cross-document event ordering. In *Proc. of the International Workshop on Semantic Evaluation (SemEval-2015)*.

P Mirza and S Tonelli. 2016. Catena: Causal and temporal relation extraction from natural language texts. In *The 26th International Conference on Computational Linguistics*, pages 64–75. Association for Computational Linguistics.

Paramita Mirza. 2016. Extracting temporal and causal relations between events. *CoRR*, abs/1604.08120.

Paramita Mirza and Anne-Lyse Minard. 2015. Hlt-fbk: a complete temporal processing system for qa tempeval. In *Proc. of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 801–805. Association for Computational Linguistics.

James Pustejovsky, José Castaño, Robert Ingria, Roser Saurì, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003. TimeML: Robust specification of event and temporal expressions in text. In *in Fifth International Workshop on Computational Semantics (IWCS-5)*.

Jannik Strötgen and Michael Gertz. 2013. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298.

Weiyi Sun. 2014. *Time Well Tell: Temporal Reasoning in Clinical Narratives*. PhD dissertation. Department of Informatics, University at Albany, SUNY.

Weiyi Sun, Anna Rumshisky, and Ozlem Uzuner. 2013. Evaluating temporal relations in clinical text: 2012 i2b2 challenge. *Journal of the American Medical Informatics Association*, 20(5):806–813.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2012. Tempeval-3: Evaluating events, time expressions, and temporal relations. *arXiv preprint arXiv:1206.5333*.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proc. of the 4th International Workshop on Semantic Evaluations*, pages 75–80. Association for Computational Linguistics.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proc. of the 5th international workshop on semantic evaluation*, pages 57–62. Association for Computational Linguistics.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. *CoRR*, abs/1605.07333.

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

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

Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *CoRR*, abs/1508.01006.

Ranking Kernels for Structures and Embeddings: A Hybrid Preference and Classification Model

Kateryna Tymoshenko[†] and **Daniele Bonadiman**[†] and **Alessandro Moschitti**
[†]DISI, University of Trento, 38123 Povo (TN), Italy
Qatar Computing Research Institute, HBKU, 34110, Doha, Qatar
{kateryna.tymoshenko,d.bonadiman}@unitn.it
amoschitti@gmail.com

Abstract

Recent work has shown that Tree Kernels (TKs) and Convolutional Neural Networks (CNNs) obtain the state of the art in answer sentence reranking. Additionally, their combination used in Support Vector Machines (SVMs) is promising as it can exploit both the syntactic patterns captured by TKs and the embeddings learned by CNNs. However, the embeddings are constructed according to a classification function, which is not directly exploitable in the preference ranking algorithm of SVMs. In this work, we propose a new hybrid approach combining preference ranking applied to TKs and pointwise ranking applied to CNNs. We show that our approach produces better results on two well-known and rather different datasets: WikiQA for answer sentence selection and SemEval cQA for comment selection in Community Question Answering.

1 Introduction

Recent work on learning to rank (L2R) has shown that deep learning and kernel methods are two very effective approaches, given their ability of engineering features. In particular, in question answering (QA), Convolutional Neural Networks (CNN), e.g., (Severyn and Moschitti, 2015; Miao et al., 2016; Yin et al., 2016) can automatically learn the representation of question and answer passage (Q/AP) in terms of word embeddings and their non-linear transformations. These are then used by the other layers of the network to measure Q/AP relatedness. In contrast, Convolution Tree Kernels (CTK) can be applied to relational structures built on top of syntactic/semantic structures derived from Q/AP text (Tymoshenko et al., 2016a). CNNs as well as CTKs can achieve the state of the art in ranking APs or also questions. Considering their complementary approach for generating features, studying ways to combine them is very promising. In (Tymoshenko et al., 2016a), we investigated the idea of extracting layers from CNNs and using them in a kernel function to be further combined with CTKs in a composite reranking kernel. This was used in an SVM^{Rank} (Joachims, 2002) model, which obtained a significant improvement over the individual methods. However, the simple use of CNN layers as vectors in a preference ranking approach is intutively not optimal since such layers are basically learnt in a classification model, thus they are not optimized for SVM^{Rank}.

In this work, we further compare and investigate different ways of combining CTKs and CNNs in reranking settings. In particular, we follow the intuition that as CNNs learn the embeddings in a classification setting they should be used in the same way for building the reranking kernel, i.e., we need to use the embeddings in a pointwise reranking fashion. Therefore, we propose a hybrid preference-pointwise kernel, which consists in (i) a standard reranking kernel based on CTKs applied to the Q/AP structural representations; and (ii) a classification kernel based on the embeddings learned by neural networks. The intuition about the hybrid models is to add CNN layer vectors, not their difference, to the preference CTK. That is, CNN layers are still used as they were used in a classification setting whereas CTKs follow the standard SVM^{Rank} approach.

We tested our proposed models on the answer sentence selection benchmark, WikiQA (Yang et al., 2015), and the benchmark from cQA SemEval-2016 Task 3.A[1] corpus. We show that

[1] http://alt.qcri.org/semeval2016/

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 897–902
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

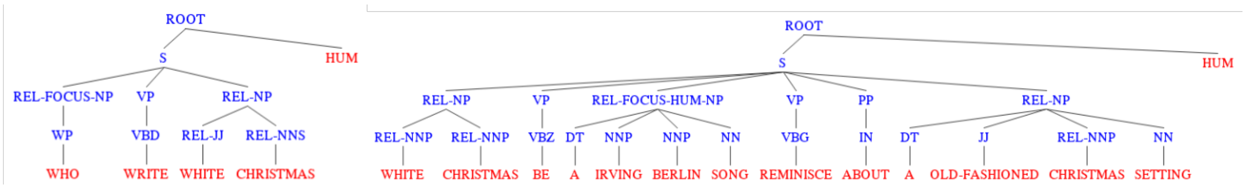


Figure 1: Shallow chunk-based tree representation of a question in the Q/AP pair: Q: *"Who wrote white Christmas?"*, AP: *"White Christmas is an Irving Berlin song"*.

the proposed hybrid kernel consistently outperforms standard reranking models in all settings.

2 Answer Sentence/Comment Selection

We focus on two question answering subtasks: answer sentence selection task (AST) and the comment selection task from cQA.

AST consists in selecting correct answer sentences (i.e., an AP composed of only one sentence) for a question Q from a set of candidate sentences, $S = \{s_1, ..., s_N\}$. In factoid question answering, Q typically asks for an entity or a fact, e.g., time location and date. S is typically a result of so-called *primary search*, a result of fast-recall/low-precision search for potential answer candidates. For example, it could be a set of candidate APs returned when running a search engine over a large corpus using Q as a search query. Many such APs are typically not pertinent to the original question, thus automatic approaches for selecting those useful are very valuable.

cQA proposes a task similar to AST, where Q is a question asked by a user in a web forum and S are the potential answers to the question posted as comments by other users. Again, many comments in a cQA thread do not contain an answer to the original question, thus raising the need for automatic comment selection.

The crucial features for both tasks capture information about the relations between Q and an AP. Manual feature engineering can provide competitive results (Nicosia et al., 2015), however, it requires significant human expertise in the specific domain and is time-consuming. Thus, machine learning methods for automatic feature engineering are extremely valuable.

3 CTK and CNN models

Our baselines are the standalone CTK and CNN models originally proposed in (Severyn et al., 2013; Severyn and Moschitti, 2015) and further advanced in (Tymoshenko et al., 2016a,b). The following subsections provide a brief overview of these models.

3.1 CTK structures

The CTK models are applied to syntactic structural representations of Q and AP. We used shallow chunk-based and constituency tree representations in AST (Tymoshenko et al., 2016a) and cQA (Tymoshenko et al., 2016b), respectively. We follow the tree construction algorithms provided in the work above. Due to the space restrictions, we present only high-level details below.

A shallow chunk-based representations of a text contains lemma nodes at leaf level and their part-of-speech (POS) tag nodes at the preterminal level. The latter are further grouped under the chunk and sentence nodes.

A constituency tree representation is an ordinary constituency parse tree. In all representations, we mark lemmas that occur in both Q and AP by prepending the **REL** tag to the labels of the corresponding preterminal nodes and their parents.

Moreover, in the AST setting, often question and focus classification information is used (Li and Roth, 2002), thus we enrich our representation with the question class and focus information, when is available.

Additionally, we mark AP chunks containing named entities that match the expected answer type of the question by prepending *REL-FOCUS-<QC>* to them. Here, the $< QC >$ placeholder is substituted with the actual question class. Fig. 1 illustrates a shallow chunk-based syntactic structure enriched with relational tags.

3.2 Convolutional Neural Networks

A number of NN-based models have been proposed in the research line of answer selection (Hu et al., 2014; Yu et al., 2014). Here, we employ

task3/

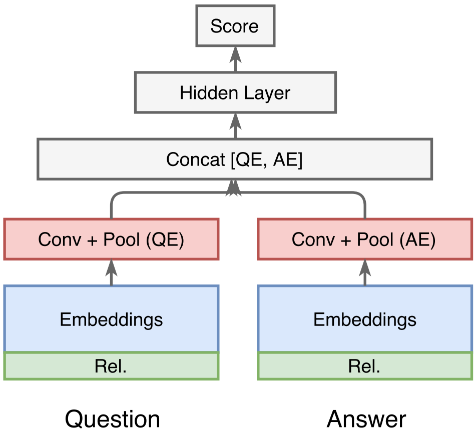


Figure 2: CNN architecture to compute the similarity between question and answer.

the NN model described in (Tymoshenko et al., 2016a) and depicted in Fig. 2. It includes two main components (i) two sentence encoders that map input documents i into fixed size m-dimensional vectors x_{s_i}, and (ii) a feed forward NN that computes the similarity between the two sentences in the input.

We use a sentence model built with a convolution operation followed by a k-max pooling layer with $k = 1$. The sentence vectors, x_{s_i}, are concatenated together and given in input to standard NN layers, which are constituted by a non-linear hidden layer and a sigmoid output layer. The sentence encoder, $x_{s_i} = f(s_i)$ outputs a fixed-size vector representation of the input sentence s_i (we will refer to $f(s_i)$ as question embedding, QE, and answer embedding, AE, respectively).

Additionally, we encode the relational information between Q and AP, by injecting relation features into the network. In particular, we associate each word w of the input sentences with a *word overlap* binary feature indicating if w is shared by both Q and AP.

4 Hybrid learning to rank model

We represent a Q/AP pair as $p = (q, a, \vec{x})$, where q and a are the structural representations of Q and AP (as described in Sec. 3), and $\vec{x}$ is a feature vector that incorporates the features characterizing the Q/AP pair (e.g., similarity features between Q and AP or their embeddings learned by an NN).

Reranking kernel. This kernel captures differences between two Q/AP pairs, p_1 and p_2, and predicts which pair should be ranked higher, i.e., in which pair, AP has higher probability to provide a correct answer to Q. In the reranking setting, a training/classification instance is a pair of Q/AP pairs, $\langle p_1 = (q, a_1, \vec{x}_1), \quad p_2 = (q, a_2, \vec{x}_2)\rangle$. The instance is positive if p_1 is ranked higher than p_2, and negative otherwise. One approach for producing training data is to form pairs both using $\langle p_1, p_2\rangle$ and $\langle p_2, p_1\rangle$, thus generating both positive and negative examples.

However, since these are clearly redundant as formed by the same members, it is more efficient training with a reduced set of examples such that members are not swapped. Algorithm 1 describes how we generate a more compact set of positive (E_+) and negative (E_-) training examples for a specific Q.

Given a pair of examples, $\langle p_1, p_2\rangle$ and $\langle p'_1, p'_2\rangle$, we used the following preference kernel (Shen and Joshi, 2003):

$$\begin{aligned} R_K(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) = K(p_1, p'_1) + \\ +K(p_2, p'_2) - K(p_1, p'_2) - K(p_2, p'_1), \end{aligned} \tag{1}$$

which is equivalent to the dot product between vector subtractions, i.e., $\big(\phi(p_1) - \phi(p_2)\big) \cdot \big(\phi(p'_1) - \phi(p'_2)\big)$, used in preference reranking, where ϕ is a feature map. Additionally, we indicate (i) with R_{TK} the preference kernel using TKs applied to q and a trees, i.e., $TK(p_i, p_j) = TK(q_i, q_j) + TK(a_i, a_j)$; and (ii) with R_V, the preference kernel applied to vectors, i.e., $V(p_i, p_j) = V(\vec{x}_i, \vec{x}_j)$. Our final reranking kernel is:

$$\begin{aligned} RK(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) = R_{TK}(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) + \\ +R_V(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) \end{aligned} \tag{2}$$

Now, if we substitute the explicit form for R_V, we have:

$$\begin{aligned} RK(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) = R_{TK}(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) + \\ +V(p_1, p'_1) + V(p_2, p'_2) - V(p_1, p'_2) - V(p_2, p'_1) \end{aligned}$$

Since our vectors are internal network layers in order to not lose important information with differences (operated by R_V), we only keep $V(p_1, p'_1)$ (or equivalently $V(p_2, p'_2)$), i.e.,

$$\begin{aligned} RK(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) = R_{TK}(\langle p_1, p_2\rangle, \langle p'_1, p'_2\rangle) + \\ V(p_1, p'_1) \end{aligned} \tag{3}$$

Note that our approach also works when using Alg. 1.

Algorithm 1 Generating training data for reranking

```
Require: $S_{q+}, S_{q-}$ - $(q, a, \vec{x})$-triplets for correct and wrong
    answer sentences per question Q
$E_+ \leftarrow \emptyset$, $E_- \leftarrow \emptyset$, $flip \leftarrow$ true
for all $s_+ \in S_{q+}$ do
    for all $s_- \in S_{q-}$ do
        if $flip$ == true then
            $E_+ \leftarrow E_+ \cup (s_+, s_-)$
            $flip \leftarrow$ false
        else
            $E_- \leftarrow E_- \cup (s_-, s_+)$
            $flip \leftarrow$ true
return $E_+, E_-$
```

5 Experiments

In our experiments, we compare various methods of combining CTKs and CNNs, using standard and our hybrid reranking kernels. The software for reproducing our experimental results is available at https://github.com/iKernels/RelTextRank.

5.1 Experimental setup

WikiQA, sentence selection dataset: this was created for open domain QA. Table 1 provides the statistics regarding this dataset. Following Yang et al. (2015), we discard questions that have either only correct or only incorrect answers.

cQA, SemEval-2016 dataset: we used the English data from Task 3, Subtask A[2]. We can exactly compare with the state of the art in SemEval. It contains questions collected from the *Qatar Living forum*[3] and the first ten comments per question manually annotated. The train, dev. and test sets contain 1790, 244 and 327 questions, respectively.

Text Preprocessing: we used the Illinois chunker (Punyakanok and Roth, 2001) and the Stanford CoreNLP (Manning et al., 2014) toolkit, v3.6.0. When experimenting with SemEval-2016, we perform preprocessing as in (Tymoshenko et al., 2016a), e.g., we truncate all the comments to 2000 symbols and sentences to 70 words.

CTKs: we trained our models with SVM-Light-TK[4] using the partial tree kernel (PTK) and the subset tree kernel (STK). We use PTK for WikiQA and STK for SemEval as suggested in our previous work (Tymoshenko et al., 2016a) with default parameters and the polynomial kernel (P) of degree 3 on all feature vectors, which are embeddings learned as described in Section 3.2.

Dataset	Q	AP	Q with AP
WikiQA-train	2,118	20,360	873
WikiQA-test	633	6,165	243
WikiQA-dev	296	2,733	126

Table 1: WikiQA statistics.

Neural Network (CNN) setup: we used the same setup and parameters as (Tymoshenko et al., 2016a): we pre-initialize the word embeddings with skipgram embedding of dimensionality 50 trained on the English Wikipedia dump (Mikolov et al., 2013). We used a single non-linear hidden layer (with hyperbolic tangent activation, Tanh), whose size is equal to the size of the previous layer, i.e., the join layer. The network is trained using SGD with shuffled mini-batches using the Rmsprop update rule (Tieleman and Hinton, 2012). The model is trained until the validation loss stops improving. The size of the sentences embedding (QE and AE) and of the join layer is set as 200.

QA metrics: we report our results in terms of Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and P@1.

5.2 Ranking with trees and embeddings

We evaluate the combination techniques proposed in Sec. 4 on the SemEval-2016 and WikiQA development (DEV) and test (TEST) sets. Additionally, to have more reliable results, it is standard practice to apply n-fold cross-validation. However, we cannot do this on the training (TRAIN) sets, since the embeddings learned in Sec. 3.2 are trained on TRAIN by construction, and therefore cross-validation on TRAIN would exhibit unrealistically high performance. Thus, we employed the following disjoint *Cross Validation* approach: we train 5 models as in traditional 5-fold cross-validation on TRAIN. Then, we merged WikiQA DEV and TEST sets, split the resulting set into 5 subsets, and use *i*-th subset to test the model trained in *i*-th fold (*i*=1,..,5).

Table 2 reports the performance of the models. Here, Rank corresponds to the traditional reranking model described by Eq. 2 in Sec. 4. Hybrid refers to our new reranking/classification kernels described by Eq. 3. V means that the model uses a kernel applied to the embedding feature vectors only. T specifies that the model employs struc-

[2] http://alt.qcri.org/semeval2016/task3/index.php?id=description-of-tasks

[3] http://www.qatarliving.com/forum

[4] http://disi.unitn.it/moschitti/Tree-Kernel.htm

	WikiQA									SemEval-2016, Task 3.A			
	DEV			TEST			Cross Validation			DEV		TEST	
	MRR	MAP	P@1	MRR	MAP	P@1	MRR	MAP	P@1	MAP	MRR	MAP	MRR
Rank:T	72.33	71.96	59.02	71.25	69.71	56.54	71.57±2.09	70.56±1.98	57.56±3.43	65.19	73.16	75.14	82.90
Class:V	70.88	70.55	58.20	66.98	65.18	53.59	68.54±1.78	67.45±1.64	55.95±1.54	65.63	72.69	75.16	82.37
Rank:V	69.39	68.73	54.92	67.56	66.42	54.43	66.64±3.21	65.66±2.19	51.83±2.66	65.29	72.65	74.49	81.77
Rank:T+V	71.54	71.05	59.02	71.56	69.99	57.38	70.23±2.63	69.33±2.15	56.79±3.19	66.22	73.74	74.79	81.69
Hybrid:T+V	**75.05**	**74.02**	**63.93**	**74.08**	**72.19**	**61.60**	**74.36±2.67**	**72.69±1.73**	**62.16±3.31**	**68.08**	**75.09**	**77.10**	**83.45**
CNN	72.04	71.73	59.84	70.34	68.73	56.12	—	—	—	66.48	73.46	76.17	81.32
Rank':T+V	71.29	70.79	57.94	72.51	71.29	59.26	—	—	—	—	—	—	—
ABCNN	—	—	—	71.27	69.14	—	—	—	—	—	—	—	—
KeLP (#1)	—	—	—	—	—	—	—	—	—	—	—	79.19	86.42
ConvKN (#2)	—	—	—	—	—	—	—	—	—	—	—	77.66	84.93

Table 2: Experimental results on WikiQA and SemEval-2016 Task 3.A corpora

tural representations with a tree kernel. Finally, V+T means that both embedding feature vectors and trees are used.

The experiments show that: in general, a standalone model with CTKs applied to the syntactic structures (*Rank:T*) outperforms the standalone feature-based models using embeddings as feature vectors (V).

Then, the straightforward combination of tree and polynomial kernels applied to the syntactic structural representations and embeddings (*Rank: T+V*) does not improve over the *Rank: T* model.

At the same time, the Hybrid model consistently outperforms all the other models in all possible experimental configurations, thus confirming our hypothesis that the classification setting is more appropriate when using embeddings as feature vectors in the kernel-based ranking models.

Additionally, for reference, we report the performance of the CNN we used to obtain the embeddings. It is consistently outperformed by the Hybrid model on all the datasets.

Finally, in the last four lines of Tab. 2, we report the performance of the state-of-the-art models from previous work, measured on exactly the same experimental settings we used.

Here *Rank':T+V* is our model described in (Tymoshenko et al., 2016a), based on the traditional reranking model. Our updated version obtains comparable performance on WikiQA-DEV and slightly lower performance on WikiQA-TEST (probably, just due to differences in preprocessing after we updated our pre-processing pipelines).

ABCNN (Yin et al., 2016) is another state-of-the-art system based on advanced attention-based convolutional networks. All our models involving CTKs outperform it.

KeLP (#1) (Filice et al., 2016) and ConvKN (#2) (Barrón-Cedeño et al., 2016) are the two top-performing SemEval 2016, Task 3.A competition systems (Nakov et al., 2016). ConvKN (#2) is an earlier version of our approach, which also employs CTKs and embeddings. Both KeLP and ConvKN (i) employ cQA-domain-specific hand-crafted features, which also consider the thread-level information, while in this work, we do not use manually engineered features; (ii) they employ PTK, which is capable of learning more powerful features than SST, but it is more computationally complex; (iii) KeLP system parameters were optimized in cross-validation on the training set, while, in this work, we perform no parameter optimization. Nevertheless, the performance of our *Hybrid:T+V* models on SemEval TEST is comparable to that of ConvKN (#2).

6 Conclusion

In this paper, we have studied and compared state-of-the-art feature engineering approaches, namely CTKs and CNNs, on two different QA tasks, AST and cQA. We investigated the ways of combining the two approaches into a single model and proposed a hybrid reranking-classification kernel for combining the structural representations and embeddings learned by CNNs.

We have shown that the combination of CTKs and CNNs with a hybrid kernel in the reranking setting outperforms the state of the art on AST and is comparable to the state of the art in cQA. In particular, in cQA, a combination of CTKs and CNNs performs comparably to the systems using domain specific features that were manually engineered.

Acknowledgments

This work has been partially supported by the EC project CogNet, 671625 (H2020-ICT-2014-2, Research and Innovation action). We would like to thank Raniero Romagnoli for helping us to review an early draft of this paper. Many thanks to the anonymous reviewers for their professional work and valuable suggestions.

References

Alberto Barrón-Cedeño, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad Al-Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. ConvKN at SemEval-2016 Task 3: Answer and Question Selection for Question Answering on Arabic and English Fora. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 896–903, San Diego, California. Association for Computational Linguistics.

S. Filice, D. Croce, A. Moschitti, and R. Basili. 2016. Kelp at semeval-2016 task 3: Learning semantic relations between questions and answers. *Proceedings of SemEval*, 16:1116–1123.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of ACM KDD*, pages 133–142. ACM.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of COLING*, pages 1–7. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL: System Demonstrations*, pages 55–60.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proc. ICML*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in NIPS 26*, pages 3111–3119.

P. Nakov, L. Márquez, A. Moschitti, W. Magdy, H. Mubarak, A. A. Freihat, J. Glass, and B. Randeree. 2016. SemEval-2016 task 3: Community question answering. In *Proceedings of SemEval '16*. ACL.

M. Nicosia, S. Filice, A. Barrón-Cedeño, I. Saleh, H. Mubarak, W. Gao, P. Nakov, G. Da San Martino, A. Moschitti, K. Darwish, L. Màrquez, S. Joty, and W. Magdy. 2015. QCRI: Answer selection for community question answering - experiments for Arabic and English. In *SemEval*.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *NIPS*, pages 995–1001.

A. Severyn, M. Nicosia, and A. Moschitti. 2013. Learning adaptable patterns for passage reranking. *CoNLL-2013*, page 75.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*. ACM.

L. Shen and A. K. Joshi. 2003. An SVM based voting algorithm with application to parse reranking. In *CONLL*, CONLL '03, pages 9–16, Edmonton, Canada.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016a. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *Proceedings of NAACL-HLT*, pages 1268–1278.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016b. Learning to rank non-factoid answers: Comment selection in web forums. In *CIKM*, pages 2049–2052. ACM.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*. ACL.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. In *NIPS Deep Learning Workshop*.

Recovering Question Answering Errors via Query Revision

Semih Yavuz and **Izzeddin Gur** and **Yu Su** and **Xifeng Yan**
Department of Computer Science, University of California, Santa Barbara
{syavuz,izzeddingur,ysu,xyan}@cs.ucsb.edu

Abstract

The existing factoid QA systems often lack a post-inspection component that can help models recover from their own mistakes. In this work, we propose to cross-check the corresponding KB relations behind the predicted answers and identify potential inconsistencies. Instead of developing a new model that accepts evidences collected from these relations, we choose to plug them back to the original questions directly and check if the revised question makes sense or not. A bidirectional LSTM is applied to encode revised questions. We develop a scoring mechanism over the revised question encodings to refine the predictions of a base QA system. This approach can improve the F_1 score of STAGG (Yih et al., 2015), one of the leading QA systems, from 52.5% to 53.9% on WEBQUESTIONS data.

1 Introduction

With the recent advances in building large scale knowledge bases (KB) like Freebase (Bollacker et al., 2008), DBpedia (Auer et al., 2007), and YAGO (Suchanek et al., 2007) that contain the world's factual information, KB-based question answering receives attention of research efforts in this area. Traditional semantic parsing is one of the most promising approaches that tackles this problem by mapping questions onto logical forms using logical languages CCG (Kwiatkowski et al., 2013; Reddy et al., 2014; Choi et al., 2015; Reddy et al., 2016), DCS (Berant et al., 2013; Berant and Liang, 2014, 2015), or directly query graphs (Yih et al., 2015) with predicates closely related to KB schema. Recently, neural network based models have been applied to question answering (Bordes et al., 2015; Yih et al., 2015; Xu et al., 2016a,b).

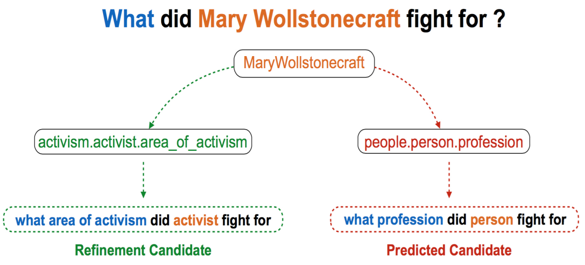

What did Mary Wollstonecraft fight for ?
MaryWollstonecraft
activism.activist.area_of_activism
people.person.profession
what area of activism did activist fight for
what profession did person fight for
Refinement Candidate
Predicted Candidate

Figure 1: Sketch of our approach. Elements in solid round rectangles are KB relation labels. Relation on the left is correct, but the base QA system predicts the one on the right. Dotted rectangles represent **revised questions** with relation labels plugged in. The left revised question looks semantically closer to the original question and itself is more consistent. Hence, it shall be ranked higher than the right one.

While these approaches yielded successful results, they often lack a post-inspection component that can help models recover from their own mistakes. Table 1 shows the potential improvement we can achieve if such a component exists. Can we leverage textual evidences related to the predicted answers to recover from a prediction error? In this work, we show it is possible.

Our strategy is to cross-check the corresponding KB relations behind the predicted answers and identify potential inconsistencies. As an intermediate step, we define **question revision** as a tailored transformation of the original question using textual evidences collected from these relations in a knowledge base, and check if the revised questions make sense or not. Figure 1 illustrates the idea over an example question "*what did Mary Wollstonecraft fight for ?*" Obviously, "*what [area of activism] did [activist] fight for ?*" looks more consistent over "*what [profession] did [person] fight for ?*" We shall build a model that prefers the former one. This model shall be specialized for comparing the revised questions and checking which one makes better sense, not for answering the revised questions. This strategy differentiates

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 903–909
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Refinement	F_1	# Refined Qs
STAGG	52.5	-
w/ Best Alternative	58.9	639

Table 1: What if we know the questions on which the system makes mistakes? Best alternative is computed by replacing the predictions of **incorrectly answered questions** by STAGG with its second top-ranked candidate.

our work from many existing QA studies.

Given a question, we first create its revisions with respect to candidate KB relations. We encode question revisions using a bidirectional LSTM. A scoring mechanism over these encodings is jointly trained with LSTM parameters with the objective that the question revised by a correct KB relation has higher score than that of other candidate KB relations by a certain confidence margin. We evaluate our method using STAGG (Yih et al., 2015) as the base question answering system. Our approach is able to improve the F_1 performance of STAGG (Yih et al., 2015) from 52.5% to 53.9% on a benchmark dataset WEBQUESTIONS (Berant et al., 2013). Certainly, one can develop specialized LSTMs that directly accommodate text evidences without revising questions. We have modified QA-LSTM and ATTENTIVE-LSTM (Tan et al., 2016) accordingly (See Section 4). However, so far the performance is not as good as the question revision approach.

2 Question Revisions

We formalize three kinds of question revisions, namely entity-centric, answer-centric, and relation-centric that revise the question with respect to evidences from topic entity type, answer type, and relation description. As illustrated in Figure 2, we design revisions to capture generalizations at different granularities while preserving the question structure.

Let s_r (e.g., `Activist`) and o_r (e.g., `ActivismIssue`) denote the subject and object types of a KB relation r (e.g., `AreaOfActivism`), respectively. Let α (`type.object.name`) denote a function returning the textual description of a KB element (e.g., relation, entity, or type). Assuming that a candidate answer set is retrieved by executing a KB relation r from a topic entity in question, we can uniquely identify the types of topic entity and answer for the hypothesis by s_r and o_r, respectively. It is also possible that a chain of relations $r = r_1 r_2 \dots r_k$ is used to retrieve an answer set from a topic entity. When $k = 2$, by abuse of notation, we define $s_{r_1 r_2} = s_{r_1}$, $o_{r_1 r_2} = o_{r_2}$, and $\alpha(r_1 r_2) = concat(\alpha(r_1), \alpha(r_2))$.

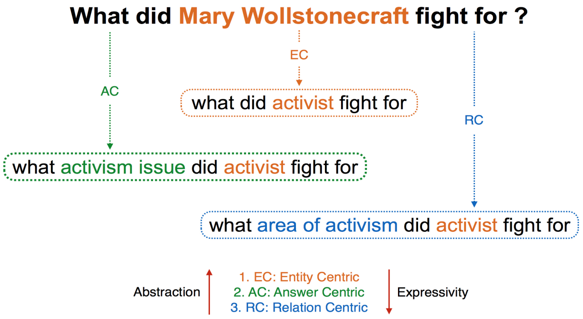

What did Mary Wollstonecraft fight for ?
EC
AC
RC
what did activist fight for
what activism issue did activist fight for
what area of activism did activist fight for
Abstraction
1. EC: Entity Centric
2. AC: Answer Centric
3. RC: Relation Centric
Expressivity

Figure 2: Illustration of different question revision strategies on the running example w.r.t KB relation `activism.activist.area_of_activism`.

Let $m : (q, r) \mapsto q'$ denote a mapping from a given question $q = [w_1, w_2, \dots, w_L]$ and a KB relation r to revised question q'. We denote the index span of wh-words (e.g., "what") and topic entity (e.g., "Mary Wollstonecraft") in question q by $[i_s, i_e]$ and $[j_s, j_e]$, respectively.

Entity-Centric (EC). Entity-centric question revision aims a generalization at the entity level. We construct it by replacing topic entity tokens with its type. For the running example, it becomes "*what did [activist] fight for*". Formally, $m_{EC}(q, r) = [w_{[1:j_s-1]}; \alpha(s_r); w_{[j_e+1:L]}]$.

Answer-Centric (AC). It is constructed by augmenting the wh-words of entity-centric question revision with the answer type. The running example is revised to "*[what activism issue] did [activist] fight for*". We formally define it as $m_{AC}(q, r) = [w'_{[1:i_e]}; \alpha(o_r); w'_{[i_e+1:L']}]$, where w'_i's are the tokens of entity-centric question revision $m_{EC}(q, r)$ of length L' with $[i_s, i_e]$ still denoting the index span of wh-words in w'.

Relation-Centric (RC). Here we augment the wh-words with the relation description instead of answer type. This form of question revision has the most expressive power in distinguishing between the KB relations in question context, but it can suffer more from the training data sparsity. For the running example, it maps to "*[what area of activism] did [activist] fight for*". Formally, it is defined as $m_{RC}(q, r) = [w'_{[1:i_e]}; \alpha(r); w'_{[i_e+1:L']}]$.

3 Model

3.1 Task Formulation

Given a question q, we first run an existing QA system to answer q. Suppose it returns r as the top predicted relation and r' is a candidate relation that is ranked lower. Our objective is to decide if there is a need to replace r with r'. We formulate this task as finding a scoring function $s : (q, r) \to \mathbb{R}$ and a confidence margin threshold $t \in \mathbb{R}_{>0}$ such that the function

$$replace(r, r', q) = \begin{cases} 1, \text{if } s(q, r') - s(q, r) \geq t \\ 0, \text{otherwise} \end{cases} \quad (1)$$

makes the replacement decision.

3.2 Encoding Question Revisions

Let $q' = (w'_1, w'_2, \ldots, w'_l)$ denote a question revision. We first encode all the words into a d-dimensional vector space using an embedding matrix. Let $\mathbf{e}_i$ denote the embedding of word w'_i. To obtain the contextual embeddings for words, we use bi-directional LSTM

$$\overrightarrow{\mathbf{h}}_i = LSTM_{fwd}(\overrightarrow{\mathbf{h}}_{i-1}, \mathbf{e}_i) \quad (2)$$

$$\overleftarrow{\mathbf{h}}_i = LSTM_{bwd}(\overleftarrow{\mathbf{h}}_{i+1}, \mathbf{e}_i) \quad (3)$$

with $\overrightarrow{\mathbf{h}}_0 = \mathbf{0}$ and $\overleftarrow{\mathbf{h}}_{l+1} = \mathbf{0}$. We combine forward and backward contextual embeddings by $\mathbf{h}_i = \mathbf{concat}(\overrightarrow{\mathbf{h}}_i, \overleftarrow{\mathbf{h}}_i)$. We then generate the final encoding of revised question q' by $\mathbf{enc}(q') = \mathbf{concat}(\mathbf{h}_1, \mathbf{h}_l)$.

3.3 Training Objective

Score Function. Given a question revision mapping m, a question q, and a relation r, our scoring function is defined as $s(q, r) = \boldsymbol{w}^T \mathbf{enc}(m(q, r))$ where $\boldsymbol{w}$ is a model parameter that is jointly learnt with the LSTM parameters.

Loss Function. Let $\mathcal{T} = \{(q, a_q)\}$ denote a set of training questions paired with their true answer set. Let $U(q)$ denote the set of all candidate KB relations for question q. Let $f(q, r)$ denote the F_1 value of an answer set obtained by relation r when compared to a_q. For each candidate relation $r \in U(q)$ with a positive F_1 value, we define

$$N(q, r) = \{r' \in U(q) : f(q, r) > f(q, r')\} \quad (4)$$

as the set of its negative relations for question q. Similar to a hinge-loss in (Bordes et al., 2014), we define the objective function $J(\boldsymbol{\theta}, \boldsymbol{w}, \boldsymbol{E})$ as

$$\sum_{(q,r,r')} max(0, \delta_\lambda(q, r, r') - s(q, r) + s(q, r')) \quad (5)$$

where the sum is taken over all valid $\{(q, r, r')\}$ triplets and the penalty margin is defined as $\delta_\lambda(q, r, r') = \lambda(f(q, r) - f(q, r'))$.

We use this loss function because: i) it allows us to exploit partially correct answers via F_1 scores, and ii) training with it updates the model parameters towards putting a large margin between the scores of correct (r) and incorrect (r') relations, which is naturally aligned with our prediction refinement objective defined in Equation 1.

4 Alternative Solutions

Our approach directly integrates additional textual evidences with the question itself, which can be processed by any sequence oriented model, and benefit from its future updates without significant modification. However, we could also design models taking these textual evidences into specific consideration, without even appealing to question revision. We have explored this option and tried two methods that closely follow QA-LSTM and ATTENTIVE-LSTM (Tan et al., 2016). The latter model achieves the state-of-the-art for passage-level question answer matching. Unlike our approach, they encode questions and evidences for candidate answers in parallel, and measure the semantic similarity between them using *cosine* distance. The effectiveness of these architectures has been shown in other studies (Neculoiu et al., 2016; Hermann et al., 2015; Chen et al., 2016; Mueller and Thyagarajan, 2016) as well.

We adopt these models in our setting as follows: (1) Textual evidences $\alpha(s_r)$ (equiv. of **EC** revision), $\alpha(o_r)$ (equiv. of **AC** revision) or $\alpha(r)$ (equiv. of **RC** revision) of a candidate KB relation r is used in place of a candidate answer a in the original model, (2) We replace the entity mention with a universal #entity# token as in (Yih et al., 2015) because individual entities are rare and uninformative for semantic similarity, (3) We train the score function $sim(q, r)$ using the objective defined in Eq. 5. Further details of the alternative solutions can be found in Appendix A.

5 Experiments

Datasets. For evaluation, we use the WEBQUESTIONS (Berant et al., 2013), a benchmark dataset for QA on Freebase. It contains 5,810 questions whose answers are annotated from Freebase using Amazon Mechanical Turk. We also use SIMPLEQUESTIONS (Bordes et al., 2015), a collection of 108,442 question/Freebase-fact pairs, for training data augmentation in some of our experiments, which is denoted by **+SimpleQ.** in results.

Method	F_1
(Dong et al., 2015)	40.8
(Yao, 2015)	44.3
(Berant and Liang, 2015)	49.7
STAGG (Yih et al., 2015)	**52.5**
(Reddy et al., 2016)	50.3
(Xu et al., 2016b)	53.3
(Xu et al., 2016a)	**53.8**
QuesRev on STAGG	**53.9**
Ensemble	
STAGG-Rank (Yavuz et al., 2016)	54.0
QuesRev on STAGG-Rank	**54.3**

Table 2: Comparison of our question revision approach (QuesRev) on STAGG with variety of recent KB-QA works.

Refinement Model	WebQ.	+ SimpleQ.
QA-LSTM-(equiv **EC**)	51.9	52.5
QA-LSTM-(equiv **AC**)	52.4	52.9
QA-LSTM-(equiv **RC**)	52.6	53.0
Attentive-LSTM-(equiv **EC**)	52.2	52.6
Attentive-LSTM-(equiv **AC**)	52.7	53.0
Attentive-LSTM-(equiv **RC**)	52.9	53.1
QuesRev-**EC**	52.9	52.8
QuesRev-**AC**	**53.5**	53.6
QuesRev-**RC**	53.2	53.8
QuesRev-**AC** + **RC**	53.3	**53.9**

Table 3: $\mathbf{F_1}$ performance of variants of our model QuesRev and alternative solutions on base QA system STAGG.

Training Data Preparation. WebQuestions only provides question-answer pairs along with annotated topic entities. We generate candidates $U(q)$ for each question q by retrieving 1-hop and 2-hop KB relations r from annotated topic entity e in Freebase. For each relation r, we query $(e, r, ?)$ against Freebase and retrieve the candidate answers r_a. Then, we compute $f(q, r)$ by comparing the answer set r_a with the annotated answers.

5.1 Implementation Details

Word embeddings are initialized with pretrained GloVe (Pennington et al., 2014) vectors[1], and updated during the training. We take the dimension of word embeddings and the size of LSTM hidden layer equal and experiment with values in $\{50, 100, 200, 300\}$. We apply dropout regularization on both input and output of LSTM encoder with probability 0.5. We hand tuned penalty margin scalar λ as 1. The model parameters are optimized using Adam (Kingma and Ba, 2015) with batch size of 32. We implemented our models in *tensorflow* (Abadi et al., 2016).

To refine predictions r of a base QA system, we take its second top ranked prediction as the refinement candidate r', and employ $replace(r, r', q)$ in Eq. 1. Confidence margin threshold t is tuned by grid search on the training data after the score function is trained. QuesRev-**AC** + **RC** model is obtained by a linear combination of QuesRev-**AC** and QuesRev-**RC**, which is formally defined in Appendix B. To evaluate the alternative solutions for prediction refinement, we apply the same decision mechanism in Eq. 1 with the trained $sim(q, r)$ in Section 4 as the score function.

We use a dictionary[2] to identify wh-words in a question. We find topic entity spans using Stanford NER tagger (Manning et al., 2014). If there are multiple matches, we use the first matching span for both.

5.2 Results

Table 2 presents the main result of our prediction refinement model using STAGG's results. Our approach improves the performance of a strong base QA system by 1.4% and achieves 53.9% in F_1 measure, which is slightly better than the state-of-the-art KB-QA system (Xu et al., 2016a). However, it is important to note here that Xu et al. (2016a) uses DBPedia knowledge base in addition to Freebase and the Wikipedia corpus that we do not utilize. Moreover, applying our approach on the STAGG predictions reranked by (Yavuz et al., 2016), referred as STAGG-Rank in Table 2, leads to a further improvement over a strong ensemble baseline. These suggest that our system captures orthogonal signals to the ones exploited in the base QA models. Improvements of QuesRev over both STAGG and STAGG-Rank are statistically significant.

In Table 3, we present variants of our approach. We observe that **AC** model yields to best refinement results when trained only on WebQuestions data (e.g., **WebQ.** column). This empirical observation is intuitively expected because it has more generalization power than **RC**, which might make **AC** more robust to the training data sparsity. This intuition is further justified by observing that augmenting the training data with SimpleQuestions improves the performance of **RC** model most as it has more expressive power.

Although both QA-LSTM and Attentive-LSTM lead to successful prediction refinements on STAGG, question revision approach consistently outperforms both of the alternative solutions. This suggests that our way of incorporating the new textual evidences by naturally blending them in

[1] http://nlp.stanford.edu/projects/glove/

[2] what, who, where, which, when, how

Example Predictions and Replacements
1. What position did **vince lombardi** play in college ? **STAGG**: person.education / education.institution (2-hop) - what position did person play in college **QUESREV-EC**: football_player.position_s - what position did american football player play in college
2. What did **mary wollstonecraft** fight for ? **STAGG**: person.profession - what profession did person fight for **QUESREV-AC**: activist.area_of_activism - what activism issue did activist fight for
3. Where was **anne boleyn** executed ? **STAGG**: person.place_of_birth - where place of birth was person executed **QUESREV-RC**: deceased_person.place_of_death - where place of death was deceased person executed
4. Where does the **zambezi river** start ? **STAGG**: river.mouth - where mouth does the river start **QUESREV-RC**: river.origin - where origin does the river start

Table 4: Example predictions of STAGG (Yih et al., 2015) and replacements proposed by variants of QUESREV, followed by their corresponding question revisions. The colors *red* and *blue* indicate wrong and correct, respectively. Domain names of KB relations are dropped for brevity.

the question context leads to a better mechanism for checking the consistency of KB relations with the question. It is possible to argue that part of the improvements of refinement models over STAGG in Table 3 may be due to model ensembling. However, the performance gap between QUESREV and the alternative solutions enables us to isolate this effect for query revision approach.

6 Related Work

One of the promising approaches for KB-QA is semantic parsing, which uses logical language CCG (Kwiatkowski et al., 2013; Reddy et al., 2014; Choi et al., 2015) or DCS (Berant et al., 2013) for finding the right grounding of the natural language on knowledge base. Another major line of work (Bordes et al., 2014; Yih et al., 2015; Xu et al., 2016b) exploit vector space embedding approach to directly measure the semantic similarity between questions and candidate answer subgraphs in KB. In this work, we propose a post-inspection step that can help existing KB-QA systems recover from answer prediction errors.

Our work is conceptually related to traditional query expansion, a well-explored technique (Qiu and Frei, 1993; Mitra et al., 1998; Navigli and Velardi, 2003; Riezler et al., 2007; Fang, 2008; Sordoni et al., 2014; Diaz et al., 2016) in information retrieval area. The intuition behind query expansion is to reformulate the original query to improve retrieval performance. Our approach revises questions using candidate answers already retrieved by a base QA system. Revised questions are then used for reasoning about the corresponding predictions themselves, not for retrieving more candidates. Hence, it is specialized rather as a reasoning component than a retrieval one.

Hypothesis generation steps in (Téllez-Valero et al., 2008) and (Trischler et al., 2016) are related to our question revision process. However, hypotheses in these approaches need to be further compared against supporting paragraphs for reasoning. This limits the applicability of them in KB-QA setting due to lack of supporting texts. Our approach modifies the appropriate parts of the question using different KB evidences behind candidate answers that are more informative and generalizable. This enables us to make reasoning about candidate predictions directly via revised questions without relying on any supporting texts.

7 Conclusion

We present a prediction refinement approach for question answering over knowledge bases. We introduce question revision as a tailored augmentation of the question via various textual evidences from KB relations. We exploit revised questions as a way to reexamine the consistency of candidate KB relations with the question itself. We show that our method improves the quality of answers produced by STAGG on the WEBQUESTIONS dataset.

Acknowledgements

The authors would like to thank the anonymous reviewers for their thoughtful comments. This research was sponsored in part by the Army Research Laboratory under cooperative agreements W911NF09-2-0053, NSF IIS 1528175, and NSF CCF 1548848. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

A Implementation details of alternative solutions

Following (Tan et al., 2016), we use the same bidirectional LSTM for both questions and textual evidences. For the attentive model, we apply the attention mechanism on the question side because our objective is to match textual evidences to the question context unlike the original model. We use average pooling for both models and compute the *general* attention via a bilinear term that has been shown effective in (Luong et al., 2015).

For the model and training parameters, we follow the strategy described in Section 5.1 with a difference that λ is tuned to be 0.2 in this setting. This intuitively makes sense because the score $sim(q, r)$ is in $[-1, 1]$.

To clarify the question and answer sides for the alternative models, we provide concrete examples in Table 5 for the running example.

Question Side	Answer Side	Model Name
what did #entity# fight for	activist	ALT.-(equiv **EC**)
what did #entity# fight for	activism issue	ALT.-(equiv **AC**)
what did #entity# fight for	area of activism	ALT.-(equiv **RC**)

Table 5: Question (q) and answer (a) sides used for alternative (e.g., ALT.) solutions QA-LSTM and ATTENTIVE-LSTM.

B Combining multiple question revision strategies

We also performed experiments combining multiple question revisions that may potentially capture complementary signals. To this end, let $s_1, \ldots, s_k$ be the trained scoring functions with question revisions constructed by $m_1, \ldots, m_k$, we define $s(q, r) = \sum_{i=1}^{k} \gamma_i s_i(q, r)$ where $\gamma \in \mathbb{R}^k$ is a weight vector that is trained using the same objective defined in Equation 5. This strategy is used to obtain **AC+RC** model reported in experimental results by combining **AC** and **RC** for $k = 2$.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *International Semantic Web Conference (ISWC)*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. *Annual Meeting of the Association for Computational Linguistics (ACL)* .

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics (TACL)* .

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD International Conference on Management of Data*.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. *ArXiv* .

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *ArXiv* .

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Eunsol Choi, Tom Kwiatkowski, and Luke Zettlemoyer. 2015. Scalable semantic parsing with partial ontologies. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query expansion with locally-trained word embeddings. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multicolumn convolutional neural networks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hui Fang. 2008. A re-examination of query expansion using lexical resources. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Mandar Mitra, Amit Singhal, and Chris Buckley. 1998. Improving automatic query expansion. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Roberto Navigli and Paola Velardi. 2003. An analysis of ontology-based query expansion strategies. In *European Conference on Machine Learning (ECML)*.

Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning text similarity with siamese recurrent networks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Yonggang Qiu and H.P Frei. 1993. Concept based query expansion. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics (TACL)* .

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics (TACL)* .

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alessandro Sordoni, Yoshua Bengio, and Jian-Yun Nie. 2014. Learning concept embeddings for query expansion by quantum entropy minimization. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. In *World Wide Web (WWW)*.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alberto Téllez-Valero, Manuel Montes-y Gómez, Luis Villaseñor-Pineda, and Anselmo Peñas. 2008. Improving question answering by combining multiple systems via answer validation. In *International Conference on Computational Linguistics and Intelligent Text Processing (CICLING)*.

Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. 2016. Natural language comprehension with epireader. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016a. Hybrid question answering over knowledge base and free text. In *International Conference on Computational Linguistics (COLING)*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016b. Question answering on freebase via relation extraction and textual evidence. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *The North American Chapter of the Association for Computational Linguistics (NAACL)*.

Semih Yavuz, Izzeddin Gur, Yu Su, Mudhakar Srivatsa, and Xifeng Yan. 2016. Improving semantic parsing via answer type inference. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Wen-tau Yih, MingWei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

An empirical study on the effectiveness of images in Multimodal Neural Machine Translation

Jean-Benoit Delbrouck and **Stéphane Dupont**
TCTS Lab, University of Mons, Belgium
{jean-benoit.delbrouck, stephane.dupont}@umons.ac.be

Abstract

In state-of-the-art Neural Machine Translation (NMT), an attention mechanism is used during decoding to enhance the translation. At every step, the decoder uses this mechanism to focus on different parts of the source sentence to gather the most useful information before outputting its target word. Recently, the effectiveness of the attention mechanism has also been explored for multimodal tasks, where it becomes possible to focus both on sentence parts and image regions that they describe. In this paper, we compare several attention mechanism on the multimodal translation task (English, image → German) and evaluate the ability of the model to make use of images to improve translation. We surpass state-of-the-art scores on the Multi30k data set, we nevertheless identify and report different misbehavior of the machine while translating.

1 Introduction

In machine translation, neural networks have attracted a lot of research attention. Recently, the attention-based encoder-decoder framework (Sutskever et al., 2014; Bahdanau et al., 2014) has been largely adopted. In this approach, Recurrent Neural Networks (RNNs) map source sequences of words to target sequences. The attention mechanism is learned to focus on different parts of the input sentence while decoding. Attention mechanisms have shown to work with other modalities too, like images, where their are able to learn to attend the salient parts of an image, for instance when generating text captions (Xu et al., 2015). For such applications, Convolutional Neural Networks (CNNs) such as Deep Residual (He et al., 2016) have shown to work best to represent images.

Multimodal models of texts and images empower new applications such as visual question answering or multimodal caption translation. Also, the grounding of multiple modalities against each other may enable the model to have a better understanding of each modality individually, such as in natural language understanding applications.

In the field of Machine Translation (MT), the efficient integration of multimodal information still remains a challenging task. It requires combining diverse modality vector representations with each other. These vector representations, also called context vectors, are computed in order the capture the most relevant information in a modality to output the best translation of a sentence.

To investigate the effectiveness of information obtained from images, a multimodal machine translation shared task (Specia et al., 2016) has been addressed to the MT community[1]. The best results of NMT model were those of Huang et al. (2016) who used LSTM fed with global visual features or multiple regional visual features followed by rescoring. Recently, Calixto et al. (2017) proposed a doubly-attentive decoder that outperformed this baseline with less data and without rescoring.

Our paper is structured as follows. In section 2, we briefly describe our NMT model as well as the conditional GRU activation used in the decoder. We also explain how multi-modalities can be implemented within this framework. In the following sections (3 and 4), we detail three attention mechanisms and explain how we tweak them to work as well as possible with images. Finally, we report and analyze our results in section 5 then conclude in section 6.

[1] http://www.statmt.org/wmt16/multimodal-task.html

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 910–919
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

2 Neural Machine Translation

In this section, we detail the neural machine translation architecture by Bahdanau et al. (2014), implemented as an attention-based encoder-decoder framework with recurrent neural networks (§2.1). We follow by explaining the conditional GRU layer (§2.2) - the gating mechanism we chose for our RNN - and how the model can be ported to a multimodal version (§2.3).

2.1 Text-based NMT

Given a source sentence $X = (x_1, x_2, \ldots, x_M)$, the neural network directly models the conditional probability $p(Y|X)$ of its translation $Y = (y_1, y_2, \ldots, y_N)$. The network consists of one encoder and one decoder with one attention mechanism. The encoder computes a representation C for each source sentence and a decoder generates one target word at a time and by decomposing the following conditional probability :

$$\log p(Y|X) = \sum_{t=1}^{n} \log p(y_t|y < t, C) \qquad (1)$$

Each source word x_i and target word y_i are a column index of the embedding matrix $\boldsymbol{E}_X$ and $\boldsymbol{E}_Y$. The encoder is a bi-directional RNN with Gated Recurrent Unit (GRU) layers (Chung et al., 2014; Cho et al., 2014), where a forward RNN $\overrightarrow{\Psi}_{\text{enc}}$ reads the input sequence as it is ordered (from x_1 to x_M) and calculates a sequence of forward hidden states $(\overrightarrow{\boldsymbol{h}}_1, \overrightarrow{\boldsymbol{h}}_2, \ldots, \overrightarrow{\boldsymbol{h}}_M)$. A backward RNN $\overleftarrow{\Psi}_{\text{enc}}$ reads the sequence in the reverse order (from x_M to x_1), resulting in a sequence of backward hidden states $(\overleftarrow{\boldsymbol{h}}_M, \overleftarrow{\boldsymbol{h}}_{M-1}, \ldots, \overleftarrow{\boldsymbol{h}}_1)$. We obtain an annotation for each word x_i by concatenating the forward and backward hidden state $\boldsymbol{h}_t = [\overrightarrow{\boldsymbol{h}}_t; \overleftarrow{\boldsymbol{h}}_t]$. Each annotation $\boldsymbol{h}_t$ contains the summaries of both the preceding words and the following words. The representation C for each source sentence is the sequence of annotations $C = (\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_M)$.

The decoder is an RNN that uses a conditional GRU (cGRU, more details in §2.2) with an attention mechanism to generate a word y_t at each time-step t. The cGRU uses it's previous hidden state $\boldsymbol{s}_{t-1}$, the whole sequence of source annotations C and the previously decoded symbol y_{t-1} in order to update it's hidden state $\boldsymbol{s}_t$:

$$\mathbf{s}_t = \text{cGRU}\left(\mathbf{s}_{t-1}, y_{t-1}, C\right) \qquad (2)$$

In the process, the cGRU also computes a time-dependent context vector $\boldsymbol{c}_t$. Both $\boldsymbol{s}_t$ and $\boldsymbol{c}_t$ are further used to decode the next symbol. We use a deep output layer (Pascanu et al., 2014) to compute a vocabulary-sized vector :

$$\boldsymbol{o}_t = \boldsymbol{L}_o \tanh(\boldsymbol{L}_s s_t + \boldsymbol{L}_c \boldsymbol{c}_t + \boldsymbol{L}_w \boldsymbol{E}_Y[y_{t-1}]) \qquad (3)$$

where $\boldsymbol{L}_o$, $\boldsymbol{L}_s$, $\boldsymbol{L}_c$, $\boldsymbol{L}_w$ are model parameters. We can parameterize the probability of decoding each word y_t as:

$$p(y_t|y_{t-1}, \boldsymbol{s}_t, \boldsymbol{c}_t) = \text{Softmax}(\boldsymbol{o}_t) \qquad (4)$$

The initial state of the decoder $\boldsymbol{s}_0$ at time-step $t = 0$ is initialized by the following equation :

$$\boldsymbol{s}_0 = f_{\text{init}}(\boldsymbol{h}_M) \qquad (5)$$

where f_{init} is a feedforward network with one hidden layer.

2.2 Conditional GRU

The conditional GRU [2] consists of two stacked GRU activations called REC_1 and REC_2 and an attention mechanism f_{att} in between (called ATT in the footnote paper). At each time-step t, REC1 firstly computes a hidden state proposal $\boldsymbol{s}'_t$ based on the previous hidden state $\boldsymbol{s}_{t-1}$ and the previously emitted word y_{t-1}:

$$\begin{aligned}
\boldsymbol{z}'_t &= \sigma\left(\boldsymbol{W}'_z \boldsymbol{E}_Y[y_{t-1}] + \boldsymbol{U}'_z \boldsymbol{s}_{t-1}\right) \\
\boldsymbol{r}'_t &= \sigma\left(\boldsymbol{W}'_r \boldsymbol{E}_Y[y_{t-1}] + \boldsymbol{U}'_r \boldsymbol{s}_{t-1}\right) \\
\underline{\boldsymbol{s}}'_t &= \tanh\left(\boldsymbol{W}' \boldsymbol{E}_Y[y_{t-1}] + \boldsymbol{r}'_t \odot (\boldsymbol{U}' \boldsymbol{s}_{t-1})\right) \\
\boldsymbol{s}'_t &= (1 - \boldsymbol{z}'_t) \odot \underline{\boldsymbol{s}}'_t + \boldsymbol{z}'_t \odot \boldsymbol{s}_{t-1} \qquad (6)
\end{aligned}$$

Then, the attention mechanism computes c_t over the source sentence using the annotations sequence C and the intermediate hidden state proposal $\boldsymbol{s}'_t$:

$$\boldsymbol{c}_t = f_{\text{att}}\left(\text{C}, \boldsymbol{s}'_t\right) \qquad (7)$$

Finally, the second recurrent cell REC_2, computes the hidden state $\boldsymbol{s}_t$ of the cGRU by looking at the intermediate representation $\boldsymbol{s}'_t$ and context vector $\boldsymbol{c}_t$:

$$\begin{aligned}
\boldsymbol{z}_t &= \sigma\left(\boldsymbol{W}_z \boldsymbol{c}_t + \boldsymbol{U}_z \boldsymbol{s}'_t\right) \\
\boldsymbol{r}_t &= \sigma\left(\boldsymbol{W}_r \boldsymbol{c}_t + \boldsymbol{U}_r \boldsymbol{s}'_t\right) \\
\underline{\boldsymbol{s}}_t &= \tanh\left(\boldsymbol{W} \boldsymbol{c}_t + \boldsymbol{r}_t \odot (\boldsymbol{U} \boldsymbol{s}'_t)\right) \\
\boldsymbol{s}_t &= (1 - \boldsymbol{z}_t) \odot \underline{\boldsymbol{s}}_t + \boldsymbol{z}_t \odot \boldsymbol{s}'_t \qquad (8)
\end{aligned}$$

[2] https://github.com/nyu-dl/dl4mt-tutorial/blob/master/docs/cgru.pdf

2.3 Multimodal NMT

Recently, Calixto et al. (2017) proposed a doubly attentive decoder (referred as the ”MNMT” model in the author’s paper) which can be seen as an expansion of the attention-based NMT model proposed in the previous section. Given a sequence of second a modality annotations $I = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_L)$, we also compute a new context vector based on the same intermediate hidden state proposal $\boldsymbol{s}'_t$:

$$\boldsymbol{i}_t = f'_{\text{att}}\left(\text{I}, \boldsymbol{s}'_t\right) \tag{9}$$

This new time-dependent context vector is an additional input to a modified version of REC2 which now computes the final hidden state $\boldsymbol{s}_t$ using the intermediate hidden state proposal $\boldsymbol{s}'_t$ and both time-dependent context vectors $\boldsymbol{c}_t$ and $\boldsymbol{i}_t$:

$$\begin{aligned}
\boldsymbol{z}_t &= \sigma\left(\boldsymbol{W}_z \boldsymbol{c}_t + \boldsymbol{W}_z \boldsymbol{i}_t + \boldsymbol{U}_z \boldsymbol{s}'_t\right) \\
\boldsymbol{r}_t &= \sigma\left(\boldsymbol{W}_r \boldsymbol{c}_t + \boldsymbol{W}_r \boldsymbol{i}_t + \boldsymbol{U}_r \boldsymbol{s}'_t\right) \\
\underline{\boldsymbol{s}}_t &= \tanh\left(\boldsymbol{W} \boldsymbol{c}_t + \boldsymbol{W} \boldsymbol{i}_t + \boldsymbol{r}_t \odot \left(\boldsymbol{U} \boldsymbol{s}'_t\right)\right) \\
\boldsymbol{s}_t &= (1 - \boldsymbol{z}_t) \odot \underline{\boldsymbol{s}}_t + \boldsymbol{z}_t \odot \boldsymbol{s}'_t
\end{aligned} \tag{10}$$

The probabilities for the next target word (from equation 3) also takes into account the new context vector $\boldsymbol{i}_t$:

$$\boldsymbol{L}_o \tanh(\boldsymbol{L}_s \boldsymbol{s}_t + \boldsymbol{L}_c \boldsymbol{c}_t + \boldsymbol{L}_i \boldsymbol{i}_t + \boldsymbol{L}_w \boldsymbol{E}_Y [y_{t-1}]) \tag{11}$$

where $\boldsymbol{L}_i$ is a new trainable parameter.

In the field of multimodal NMT, the second modality is usually an image computed into feature maps with the help of a CNN. The annotations $a_1, a_2, \ldots, a_L$ are spatial features (i.e. each annotation represents features for a specific region in the image) . We follow the same protocol for our experiments and describe it in section 5.

3 Attention-based Models

We evaluate three models of the image attention mechanism f'_{att} of equation 7. They have in common the fact that at each time step t of the decoding phase, all approaches first take as input the annotation sequence I to derive a time-dependent context vector that contain relevant information in the image to help predict the current target word y_t. Even though these models differ in how the time-dependent context vector is derived, they share the same subsequent steps. For each mechanism, we propose two hand-picked illustrations showing where the attention is placed in an image.

3.1 Soft attention

Soft attention has firstly been used for syntactic constituency parsing by Vinyals et al. (2015) but has been widely used for translation tasks ever since. One should note that it slightly differs from Bahdanau et al. (2014) where their attention takes as input the previous decoder hidden state instead of the current (intermediate) one as shown in equation 7. This mechanism has also been successfully investigated for the task of image description generation (Xu et al., 2015) where a model generates an image’s description in natural language. It has been used in multimodal translation as well (Calixto et al., 2017), for which it constitutes a state-of-the-art.

The idea of the soft attentional model is to consider all the annotations when deriving the context vector $\boldsymbol{i}_t$. It consists of a single feed-forward network used to compute an expected alignment $\boldsymbol{e}_t$ between modality annotation $\boldsymbol{a}_l$ and the target word to be emitted at the current time step t. The inputs are the modality annotations and the intermediate representation of REC1 $\boldsymbol{s}'_t$:

$$\boldsymbol{e}_{t,l} = \boldsymbol{v}^T \tanh(\boldsymbol{U}_a \boldsymbol{s}'_t + \boldsymbol{W}_a \boldsymbol{a}_l) \tag{12}$$

The vector $\boldsymbol{e}_t$ has length L and its l-th item contains a score of how much attention should be put on the l-th annotation in order to output the best word at time t. We compute normalized scores to create an attention mask $\boldsymbol{\alpha}_t$ over annotations:

$$\boldsymbol{\alpha}_{t,i} = \frac{\exp(\boldsymbol{e}_{t,i})}{\sum_{j=1}^{L} \exp(\boldsymbol{e}_{t,j})} \tag{13}$$

$$\boldsymbol{i}_t = \sum_{i=1}^{L} \boldsymbol{\alpha}_{t,i} \boldsymbol{a}_i \tag{14}$$

Finally, the modality time-dependent context vector $\boldsymbol{i}_t$ is computed as a weighted sum over the annotation vectors (equation 14). In the above expressions, $\boldsymbol{v}^T$, $\boldsymbol{U}_a$ and $\boldsymbol{W}_a$ are trained parameters.

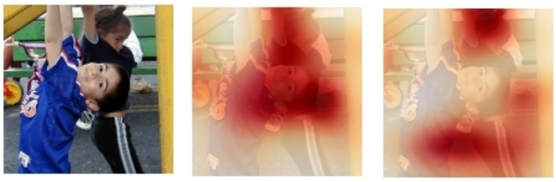


Figure 1: Die beiden Kinder spielen auf dem Spielplatz .

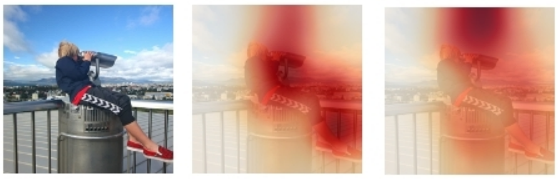


Figure 2: Ein Junge sitzt auf und blickt aus einem Mikroskop .

3.2 Hard Stochastic attention

This model is a stochastic and sampling-based process where, at every timestep t, we are making a hard choice to attend only one annotation. This corresponds to one spatial location in the image. Hard attention has previously been used in the context of object recognition (Mnih et al., 2014; Ba et al., 2015) and later extended to image description generation (Xu et al., 2015). In the context of multimodal NMT, we can follow Xu et al. (2015) because both our models involve the same process on images.

The mechanism f'_{att} is now a function that returns a sampled intermediate latent variables $\gamma_{t,i}$ based upon a multinouilli distribution parameterized by $\boldsymbol{\alpha}$:

$$\gamma_t \sim \text{Multinoulli}(\{\boldsymbol{\alpha}_{1,\ldots,L}\}) \tag{15}$$

where $\gamma_{t,i}$ an indicator one-hot variable which is set to 1 if the i-th annotation (out of L) is the one used to compute the context vector $\boldsymbol{i}_t$:

$$p(\gamma_{t,i} = 1|\boldsymbol{\gamma} < t, I) = \boldsymbol{\alpha}_{t,i} \tag{16}$$

$$\boldsymbol{i}_t = \sum_{i=1}^{L} \gamma_{t,i} \boldsymbol{a}_i \tag{17}$$

Context vector $\boldsymbol{i}_t$ is now seen as the random variable of this distribution. We define the variational lower bound $\mathcal{L}(\gamma)$ on the marginal log evidence $\log p(y|I)$ of observing the target sentence y given modality annotations I.

$$\begin{aligned}\mathcal{L}(\gamma) &= \sum_{\gamma} p(\gamma|I) \log p(y|\gamma, I) \\ &\leq \log \sum_{\gamma} p(\gamma|I) p(y|\gamma, I) \\ &= \log p(y|I)\end{aligned} \tag{18}$$

The learning rules can be derived by taking derivatives of the above variational free energy $\mathcal{L}(\gamma)$ with respect to the model parameter $\boldsymbol{W}$:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} = \sum_{\gamma} p(\gamma|I) \Bigg[\frac{\partial \log p(y|\gamma, I)}{\partial \boldsymbol{W}} + \log p(y|\gamma, I) \frac{\partial \log p(\gamma|I)}{\partial \boldsymbol{W}} \Bigg] \tag{19}$$

In order to propagate a gradient through this process, the summation in equation 19 can then be approximated using Monte Carlo based sampling defined by equation 16:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} \approx \frac{1}{N} \sum_{n=1}^{N} \Bigg[\frac{\partial \log p(y|\tilde{\gamma}^n, I)}{\partial \boldsymbol{W}} + \log p(y|\tilde{\gamma}^n, I) \frac{\partial \log p(\tilde{\gamma}^n|I)}{\partial \boldsymbol{W}} \Bigg] \tag{20}$$

To reduce variance of the estimator in equation 20, we use a moving average baseline estimated as an accumulated sum of the previous log likelihoods with exponential decay upon seeing the k-th mini-batch:

$$b_k = 0.9 \times b_{k-1} + 0.1 \times \log p(y|\tilde{\gamma}_k, I) \tag{21}$$

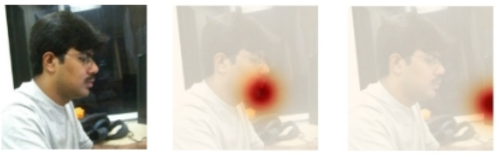


Figure 3: Ein Mann sitzt neben einem Computerbildschirm .

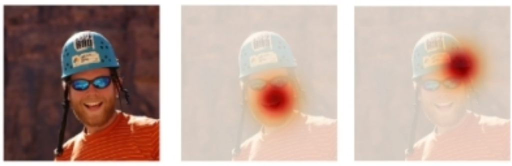


Figure 4: Ein Mann in einem orangefarbenen Hemd und mit Helm .

3.3 Local Attention

In this section, we propose a local attentional mechanism that chooses to focus only on a small subset of the image annotations. Local Attention has been used for text-based translation (Luong et al., 2015) and is inspired by the selective attention model of Gregor et al. (2015) for image generation. Their approach allows the model to select an image patch of varying location and zoom. Local attention uses instead the same ”zoom” for all target positions and still achieved good performance. This model can be seen as a trade-off between the soft and hard attentional models. The model picks one patch in the annotation sequence (one spatial location) and selectively focuses on a small window of context around it. Even though an image can't be seen as a temporal sequence, we still hope that the model finds points of interest and selects the useful information around it. This approach has an advantage of being differentiable whereas the stochastic attention requires more complicated techniques such as variance reduction and reinforcement learning to train as shown in section 3.2. The soft attention has the drawback to attend the whole image which can be difficult to learn, especially because the number of annotations L is usually large (presumably to keep a significant spatial granularity).

More formally, at every decoding step t, the model first generates an aligned position p_t. Context vector $\boldsymbol{i}_t$ is derived as a weighted sum over the annotations within the window $[p_t - N; p_t + N]$ where N is a fixed model parameter chosen empirically[3]. These selected annotations correspond to a squared region in the attention maps around p_t. The attention mask $\boldsymbol{\alpha}_t$ is of size $2N + 1$. The model predicts p_t as an aligned position in the annotation sequence (referred as Predictive alignment (local-m) in the author's paper) according to the following equation:

$$p_t = S \cdot \text{sigmoid}(\boldsymbol{v}^T \tanh(\boldsymbol{U}_a \boldsymbol{s}'_t)) \tag{22}$$

where $\boldsymbol{v}^T$ and $\boldsymbol{U}_a$ are both trainable model parameters and S is the annotation sequence length $|I|$. Because of the sigmoid, $p_t \in [0, S]$. We use equation 12 and 13 respectively to compute the expected alignment vector $\boldsymbol{e}_t$ and the attention mask $\boldsymbol{\alpha}_t$. In addition, a Gaussian distribution centered around p_t is placed on the alphas in order to favor annotations near p_t:

$$\boldsymbol{\alpha}_{t,i} = \boldsymbol{\alpha}_{t,i} \exp\left(-\frac{(i - p_t)^2}{2\sigma^2}\right) \tag{23}$$

where standard deviation $\sigma = \frac{D}{2}$. We obtain context vector $\boldsymbol{i}_t$ by following equation 14.

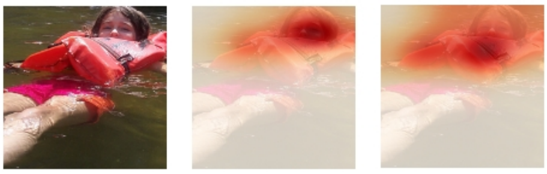


Figure 5: Ein Mädchen mit einer Schwimmweste schwimmt im Wasser .

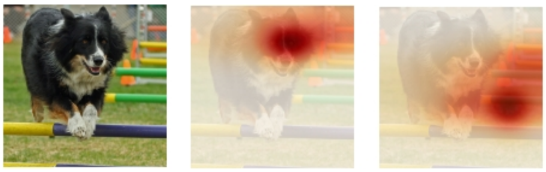


Figure 6: Ein kleiner schwarzer Hund springt über Hindernisse .

4 Image attention optimization

Three optimizations can be added to the attention mechanism regarding the image modality. All lead to a better use of the image by the model and improved the translation scores overall.

At every decoding step t, we compute a gating scalar $\boldsymbol{\beta}_t \in [0, 1]$ according to the previous decoder state $\boldsymbol{s}_{t-1}$:

$$\boldsymbol{\beta}_t = \sigma(\boldsymbol{W}_\beta \boldsymbol{s}_{t-1} + \boldsymbol{b}_\beta) \tag{24}$$

It is then used to compute the time-dependent image context vector :

$$\boldsymbol{i}_t = \boldsymbol{\beta}_t \sum_{l=1}^{L} \boldsymbol{\alpha}_{t,l} \boldsymbol{a}_l \tag{25}$$

Xu et al. (2015) empirically found it to put more emphasis on the objects in the image descriptions generated with their model.

We also double the output size of trainable parameters $\boldsymbol{U}_a$, $\boldsymbol{W}_a$ and $\boldsymbol{v}^T$ in equation 12 when it comes to compute the expected annotations over the image annotation sequence. More

[3] We pick $N = |\boldsymbol{a}_i|/4 = 49$

formally, given the image annotation sequence $I = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_L), \boldsymbol{a}_i \in \mathbb{R}^D$, the three matrices are of size $D \times 2D$, $D \times 2D$ and $2D \times 1$ respectively. We noticed a better coverage of the objects in the image by the alpha weights.

Lastly, we use a grounding attention inspired by Delbrouck and Dupont (2017). The mechanism merge each spatial location $\boldsymbol{a}_i$ in the annotation sequence I with the initial decoder state $\boldsymbol{s}_0$ obtained in equation 5 with non-linearity :

$$I' = (f(\boldsymbol{a}_1 + \boldsymbol{s}_0), f(\boldsymbol{a}_2 + \boldsymbol{s}_0), \ldots, f(\boldsymbol{a}_L + \boldsymbol{s}_0)) \tag{26}$$

where f is $\tanh$ function. The new annotations go through a L2 normalization layer followed by two 1×1 convolutional layers (of size $D \to 512, 512 \to 1$ respectively) to obtain $L \times 1$ weights, one for each spatial location. We normalize the weights with a softmax to obtain a soft attention map $\boldsymbol{\alpha}$. Each annotation $\boldsymbol{a}_i$ is then weighted according to its corresponding $\boldsymbol{\alpha}_i$:

$$I = (\boldsymbol{\alpha}_1 \boldsymbol{a}_1, \boldsymbol{\alpha}_2 \boldsymbol{a}_2, \ldots, \boldsymbol{\alpha}_L \boldsymbol{a}_L) \tag{27}$$

This method can be seen as the removal of unnecessary information in the image annotations according to the source sentence. This attention is used on top of the others - before decoding - and is referred as ”grounded image” in Table 1.

5 Experiments

For this experiments on Multimodal Machine Translation, we used the Multi30K dataset (Elliott et al., 2016) which is an extended version of the Flickr30K Entities. For each image, one of the English descriptions was selected and manually translated into German by a professional translator. As training and development data, 29,000 and 1,014 triples are used respectively. A test set of size 1000 is used for metrics evaluation.

5.1 Training and model details

All our models are build on top of the nematus framework (Sennrich et al., 2017). The encoder is a bidirectional RNN with GRU, one 1024D single-layer forward and one 1024D single-layer backward RNN. Word embeddings for source and target language are of 620D and trained jointly with the model. Word embeddings and other non-recurrent matrices are initialized by sampling from a Gaussian $\mathcal{N}(0, 0.01^2)$, recurrent matrices are random orthogonal and bias vectors are all initialized to zero.

To create the image annotations used by our decoder, we used a ResNet-50 pre-trained on ImageNet and extracted the features of size $14 \times 14 \times 1024$ at its res4f layer (He et al., 2016). In our experiments, our decoder operates on the flattened 196×1024 (i.e $L \times D$). We also apply dropout with a probability of 0.5 on the embeddings, on the hidden states in the bidirectional RNN in the encoder as well as in the decoder. In the decoder, we also apply dropout on the text annotations $\boldsymbol{h}_i$, the image features $\boldsymbol{a}_i$, on both modality context vector and on all components of the deep output layer before the readout operation. We apply dropout using one same mask in all time steps (Gal and Ghahramani, 2016).

We also normalize and tokenize English and German descriptions using the Moses tokenizer scripts (Koehn et al., 2007). We use the byte pair encoding algorithm on the train set to convert space-separated tokens into subwords (Sennrich et al., 2016), reducing our vocabulary size to 9226 and 14957 words for English and German respectively.

All variants of our attention model were trained with ADADELTA (Zeiler, 2012), with mini-batches of size 80 for our monomodal (text-only) NMT model and 40 for our multimodal NMT. We apply early stopping for model selection based on BLEU4 : training is halted if no improvement on the development set is observed for more than 20 epochs. We use the metrics BLEU4 (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014) and TER (Snover et al., 2006) to evaluate the quality of our models' translations.

5.2 Quantitative results

We notice a nice overall progress over Calixto et al. (2017) multimodal baseline, especially when using the stochastic attention. With improvements of +1.51 BLEU and -2.2 TER on both precision-oriented metrics, the model shows a strong similarity of the n-grams of our candidate translations with respect to the references. The more recall-oriented metrics METEOR scores

Model	**Test Scores**					
	BLEU↑		METEOR↑		TER↓	
Monomodal (text only)						
Caglayan et al. (2016)	32.50		49.2			
Calixto et al. (2017)	33.70		52.3		46.7	
NMT	34.11	↑ +0.41	52.4	↑ +0.1	46.2	↓ -0.5
Multimodal						
Caglayan et al. (2016)	27.82		45.0		-	
Huang et al. (2016)	36.50		54.1		-	
Calixto et al. (2017)	36.50		55.0		43.7	
Soft attention	37.10	↑ +0.60	54.8	↓ -0.2	42.8	↓ -0.9
Local attention	37.55	↑ +1.05	54.8	↓ -0.2	42.4	↓ -1.3
Stochastic attention	38.01	↑ +1.51	55.4	↑ +0.4	41.5	↓ -2.2
Soft attention + grounded image	37.62	↑ +1.12	55.3	↑ +0.3	41.8	↓ -1.9
Stochastic attention + grounded image	38.17	↑ +1.67	55.4	↑ +0.4	41.5	↓ -2.2

Table 1: Results on the 1000 test triples of the Multi30K dataset. We pick Calixto et al. (2017) scores as baseline and report our results accordingly (green for improvement and red for deterioration). In each of our experiments, Soft attention is used for text. The comparison is hence with respect to the attention mechanism used for the image modality.

are roughly the same across our models which is expected because all attention mechanisms share the same subsequent step at every time-step t, i.e. taking into account the attention weights of previous time-step $t - 1$ in order to compute the new intermediate hidden state proposal and therefore the new context vector i_t. Again, the largest improvement is given by the hard stochastic attention mechanism (+0.4 METEOR): because it is modeled as a decision process according to the previous choices, this may reinforce the idea of recall. We also remark interesting improvements when using the grounded mechanism, especially for the soft attention. The soft attention may benefit more of the grounded image because of the wide range of spatial locations it looks at, especially compared to the stochastic attention. This motivates us to dig into more complex grounding techniques in order to give the machine a deeper understanding of the modalities.

Note that even though our baseline NMT model is basically the same as Calixto et al. (2017), our experiments results are slightly better. This is probably due to the different use of dropout and subwords. We also compared our results to Caglayan et al. (2016) because our multimodal models are nearly identical with the major exception of the gating scalar (cfr. section 4). This motivated some of our qualitative analysis and hesitation towards the current architecture in the next section.

5.3 Qualitative results

For space-saving and ergonomic reasons, we only discuss about the hard stochastic and soft attention, the latter being a generalization of the local attention.

As we can see in Figure 7, the soft attention model is looking roughly at the same region of the image for every decoding step t. Because the words "*hund*"(dog), "*wald*"(forest) or "*weg*"(way) in left image are objects, they benefit from a high gating scalar. As a matter of fact, the attention mechanism has learned to detect the objects within a scene (at every time-step, whichever word we are decoding as shown in the right image) and the gating scalar has learned to decide whether or not we have to look at the picture (or more accurately whether or not we are translating an object). Without this scalar, the translation scores undergo a massive drop (as seen in Caglayan et al. (2016)) which means that the attention mechanisms don't really understand the more complex relationships between objects, what is really happening in the scene. Surprisingly, the

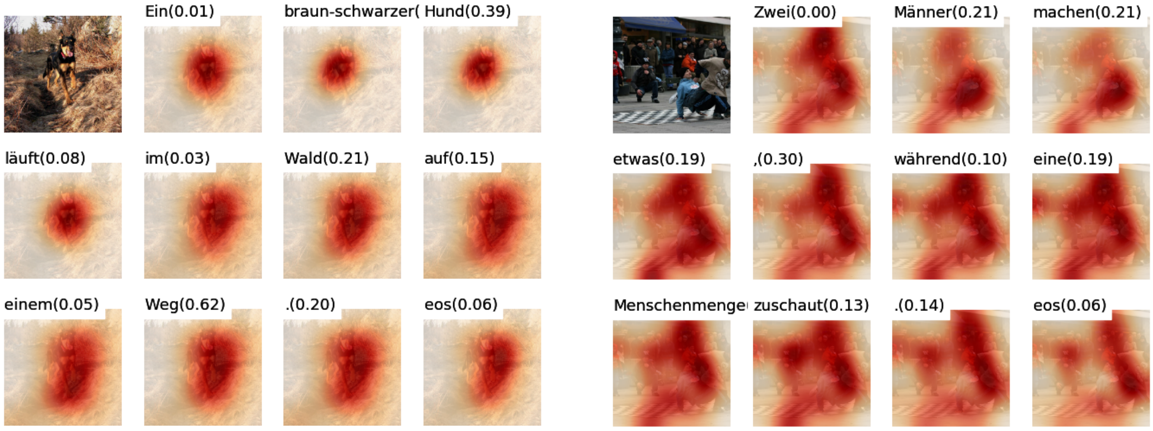


Figure 7: Representative figures of the soft-attention behavior discussed in §5.3

gating scalar happens to be really low in the stochastic attention mechanism: a significant amount of sentences don't have a summed gating scalar ≥ 0.10. The model totally discards the image in the translation process.

It is also worth to mention that we use a ResNet trained on 1.28 million images for a classification tasks. The features used by the attention mechanism are strongly object-oriented and the machine could miss important information for a multimodal translation task. We believe that the robust architecture of both encoders $\{\overleftarrow{\Psi}_{\text{enc}}, \overrightarrow{\Psi}_{\text{enc}}\}$ combined with a GRU layer and word-embeddings took care of the right translation for relationships between objects and time-dependencies. Yet, we noticed a common misbehavior for all our multimodal models: if the attention loose track of the objects in the picture and ”gets lost”, the model still takes it into account and somehow overrides the information brought by the text-based annotations. The translation is then totally mislead. We illustrate with an example:

Source:	A child claps while riding on a woman 's shoulders .
GT:	Ein Kind sitzt auf den Schultern einer Frau und klatscht .
Mono:	Ein Kind sitzt auf den Schultern einer Frau und schläft .
Soft:	Ein Kind , das sich auf der Schultern eines Frau reitet , fährt auf den Schultern .
Hard:	Ein Kind in der Haltung , während er auf den Schultern einer Frau fährt .

The monomodal translation has a sentence-level BLEU of 82.16 whilst the soft attention and hard stochastic attention scores are of 16.82 and 34.45 respectively. Figure 8 shows the attention maps for both mechanism. Nevertheless, one has to concede that the use of images indubitably helps the translation as shown in the score tabular.

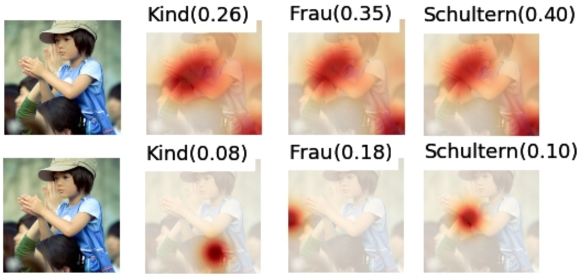


Figure 8: Wrong detection for both Soft attention (top) and Hard stochastic attention (bottom)

6 Conclusion and future work

We have tried different attention mechanism and tweaks for the image modality. We showed improvements and encouraging results overall on the Flickr30K Entities dataset. Even though we identified some flaws of the current attention mechanisms, we can conclude pretty safely that images are an helpful resource for the machine in a translation task. We are looking forward to try out richer and more suitable features for multimodal translation (ie. dense captioning features). Another interesting approach would be to use visually grounded word embeddings to capture visual notions of semantic relatedness.

7 Acknowledgements

This work was partly supported by the Chist-Era project IGLU with contribution from the Belgian Fonds de la Recherche Scientique (FNRS), contract no. R.50.11.15.F, and by the FSO project VCYCLE with contribution from the Belgian Waloon Region, contract no. 1510501.

References

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2015. Multiple object recognition with visual attention. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Ozan Caglayan, Walid Aransa, Yaxing Wang, Marc Masana, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, and Joost van de Weijer. 2016. Does multimodality help human and machine for translation and image captioning? In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, pages 627–633. http://www.aclweb.org/anthology/W/W16/W16-2358.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017. Doubly-attentive decoder for multi-modal neural machine translation. *CoRR* abs/1702.01287. http://arxiv.org/abs/1702.01287.

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. *Empirical evaluation of gated recurrent neural networks on sequence modeling*.

Jean-Benoit Delbrouck and Stephane Dupont. 2017. Multimodal compact bilinear pooling for multimodal neural machine translation. *arXiv preprint arXiv:1703.08084* https://arxiv.org/pdf/1703.08084.pdf.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

D. Elliott, S. Frank, K. Sima'an, and L. Specia. 2016. Multi30k: Multilingual english-german image descriptions pages 70–74.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. 2015. Draw: A recurrent neural network for image generation. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, Lille, France, volume 37 of *Proceedings of Machine Learning Research*, pages 1462–1471. http://proceedings.mlr.press/v37/gregor15.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Po-Yao Huang, Frederick Liu, Sz-Rung Shiang, Jean Oh, and Chris Dyer. 2016. Attention-based multimodal neural machine translation. In *Proceedings of the First Conference on Machine Translation, Berlin, Germany*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '07, pages 177–180. http://dl.acm.org/citation.cfm?id=1557769.1557821.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and koray kavukcuoglu. 2014. Recurrent models of visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 2204–2212. http://papers.nips.cc/paper/5542-recurrent-models-of-visual-attention.pdf.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '02, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. *How to construct deep recurrent neural networks*.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel L"aubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a Toolkit for Neural Machine Translation. In *Proceedings of the Demonstrations at the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. http://www.aclweb.org/anthology/P16-1162.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*. pages 223–231.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, pages 543–553. http://www.aclweb.org/anthology/W/W16/W16-2346.

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

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pages 2773–2781. http://papers.nips.cc/paper/5635-grammar-as-a-foreign-language.pdf.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. JMLR Workshop and Conference Proceedings, pages 2048–2057. http://jmlr.org/proceedings/papers/v37/xuc15.pdf.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.

Sound-Word2Vec: Learning Word Representations Grounded in Sounds

Ashwin Vijayakumar[1], **Ramakrishna Vedantam**[2] and **Devi Parikh**[1]
[1] Georgia Tech, [2] Virgina Tech
{ashwinkv,parikh}@gatech.edu, vrama1@vt.edu

Abstract

To be able to interact better with humans, it is crucial for machines to understand sound – a primary modality of human perception. Previous works have used sound to learn embeddings for improved generic semantic similarity assessment. In this work, we treat sound as a first-class citizen, studying downstream 6textual tasks which require aural grounding. To this end, we propose *sound-word2vec* – a new embedding scheme that learns specialized word embeddings grounded in sounds. For example, we learn that two seemingly (semantically) unrelated concepts, like *leaves* and *paper* are similar due to the similar *rustling* sounds they make. Our embeddings prove useful in textual tasks requiring aural reasoning like text-based sound retrieval and discovering Foley sound effects (used in movies). Moreover, our embedding space captures interesting dependencies between words and onomatopoeia and outperforms prior work on aurally-relevant word relatedness datasets such as AMEN and ASLex.

1 Introduction

Sound and vision are the dominant perceptual signals, while language helps us communicate complex experiences via rich abstractions. For example, a novel can stimulate us to mentally construct the image of the scene despite having never physically perceived it. Indeed, language has evolved to contain numerous constructs that help depict visual concepts. For example, we can easily form the picture of a *white, furry* cat with *blue* eyes via. a description of the cat in terms of its visual attributes (Lampert et al., 2009; Parikh and Grauman, 2011).

Need for Onomatopoeia. However, how would one describe the auditory instantiation of cats? While a first thought might be to use audio descriptors like *loud*, *shrill*, *husky etc*. as mid-level constructs or "attributes", arguably, it is difficult to precisely convey and comprehend sound through such language. Indeed, Wake and Asahi (1998) find that humans first communicate sounds using "onomatopoeia" – words that are suggestive of the phonetics of sounds while having no explicit meaning *e.g. meow, tic-toc*. When asked for further explanation of sounds, humans provide descriptions of potential sound sources or impressions created by the sound (*pleasant, annoying,* etc.)

Need for Grounding in Sound. While onomatopoeic words exist for commonly found concepts, a vast majority of concepts are not as perceptually striking or sufficiently frequent for us to come up with dedicated words describing their sounds. Even worse, some sounds, say, musical instruments, might be difficult to mimic using speech. Thus, for a large number of concepts there seems to be a gap between sound and its counterpart in language (Sundaram and Narayanan, 2006). This becomes problematic in specific situations where we want to talk about the heavy tail of concepts and their sounds, or while describing a particular sound we want to create as an effect (say in movies). To alleviate this, a common literary strategy is to provide metaphors to more relatable exemplars. For example, when we say, "He thundered angrily", we compare the person's angry speech to the sound of thunder to convey the seriousness of the situation. However, without this grounding in sound, thunder and anger both appear to be seemingly unrelated concepts in terms of semantics.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 920–925
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Contributions. In this work, we learn embeddings to bridge the gap between sound and its counterpart in language. We follow a retrofitting strategy, capturing similarity in sounds associated with words, while using distributional semantics (from word2vec) to provide smoothness to the embeddings. Note that we are not interested in capturing phonetic similarity, but the grounding in sound of the concept associated with the word (say "rustling" of leaves and paper.) We demonstrate the effectiveness of our embeddings on three downstream tasks that require reasoning about related aural cues:

1. Text-based sound retrieval – Given a textual query describing the sound and a database containing sounds and associated textual tags, we retrieve sound samples by matching text (Sec. 5.1)
2. Foley Sound Discovery – Given a short phrase that outlines the technique of producing Foley sounds[1], we discover other relevant words (objects or actions) which can produce similar sound effects (Sec. 5.2)
3. Aurally-relevant word relatedness assessment on AMEN and ASLex (Kiela and Clark, 2015) (Sec. 5.3)

We also qualitatively compare with word2vec to highlight the unique notions of word relatedness captured by imposing auditory grounding.

2 Related Work

Audio and Word Embeddings. Multiple works in the recent past (Bruni et al., 2014; Lazaridou et al., 2015; Lopopolo and van Miltenburg, 2015; Kiela and Clark, 2015; Kottur et al., 2016) have explored using perceptual modalities like vision and sound to learn language embeddings. While Lopopolo and van Miltenburg (2015) show preliminary results on using sound to learn distributional representations, Kiela and Clark (2015) build on ideas from Bruni et al. (2014) to learn word embeddings that respect both linguistic and auditory relationships by optimizing a joint objective. Further, they propose various fusion strategies to combine knowledge from both the modalities. Instead, we "specialize" embeddings to exclusively respect relationships defined by sounds, while initializing with word2vec embeddings for smoothness. Similar to previous findings (Melamud et al., 2016), we observe that our specialized embeddings outperform language-only as well as other multi-modal embeddings in the downstream tasks of interest.

In an orthogonal and interesting direction, other recent works (Chung et al., 2016; He et al., 2016; Settle and Livescu, 2016) learn word representations based on similarity in their pronunciation and not the sounds associated with them. In other words, phonetically similar words that have near identical pronunciations are brought closer in the embedding space (*e.g.*, *flower* and *flour*).

Sundaram and Narayanan (2006) study the applicability of onomatopoeia to obtain semantically meaningful representations of audio. Using a novel word-similarity metric and principal component analysis, they find representations for sounds and cluster them in this derived space to reason about similarities. In contrast, we are interested in learning word representations that respect aural-similarity. More importantly, our approach learns word representations for in a data-driven manner without having to first map the sound or its tags to corresponding onomatopoeic words.

Multimodal Learning with Surrogate Supervision. Kottur et al. (2016) and Owens et al. (2016) use a surrogate modality to induce supervision to learn representations for a desired modality. While the former learns word embeddings grounded in cartoon images, the latter learns visual features grounded in sound. In contrast, we use sound as the surrogate modality to supervise representation learning for words.

3 Datasets

Freesound. We use the freesound database (Font et al., 2013), also used in prior work (Kiela and Clark, 2015; Lopopolo and van Miltenburg, 2015) to learn the proposed sound-word2vec embeddings. Freesound is a freely available, collaborative dataset consisting of user uploaded sounds permitting reuse. All uploaded sounds have human descriptions in the form of tags and captions in natural language. The tags contain a broad set of relevant topics for a sound (*e.g.*, *ambience, electronic, birds, city, reverb*) and

[1] Foley sounds are sound effects (typically ambient sounds) that are added to movies in the post-production stage to make actions or situations appear more realistic. These sounds are generally created using easily available proxy objects that mimic the sound of the true situation being depicted. For example, sound of breaking celery sticks is used to create the effect of breaking bones.

captions describing the content of the sound, in addition to details pertaining to audio quality. For the text-based sound retrieval task, we use a subset of 234,120 sounds from this database and divide it into training (80%), validation (10%) and testing splits (10%). Further, for foley sound discovery, we aggregate descriptions of foley sound production provided by sound engineers (epicsound, accessed 23-Jan-2017; Singer, accessed 23-Jan-2017) to create a list of 30 foley sound pairs, which forms our ground truth for the task. For example, the description to produce a foley "driving on gravel" sound is to record the "crunching sound of plastic or polyethene bags".

AMEN and ASLex. AMEN and ASLex (Kiela and Clark, 2015) are subsets of the standard MEN (Bruni et al., 2014) and SimLex (Hill et al., 2015) word similarity datasets consisting of word-pairs that "can be associated with a distinctive associated sound". We evaluate on this dataset for completeness to benchmark our approach against previous work. However, we are primarily interested in the slightly different problem of relating words with similar auditory instantions that may or may not be semantically related as opposed to relating semantically similar words that can be associated with some common auditory signal.

4 Approach

We use the Freesound database to construct a dataset of tuples $\{s, T\}$, where s is a sound and T is the set of associated user-provided tags. We then aim to learn an embedding space for the tags that respects auditory grounding using sound information as cross-modal context – similar to word2vec (Mikolov et al., 2013) that uses neighboring words as context / supervision. We now explain our approach in detail.

Audio Features and Clustering. We represent each sound s by a feature vector consisting of the mean and variance of the following audio descriptors that are readily available as part of Freesound database:

- Mel-Frequency Cepstral Co-efficients: This feature represents the short-term power spectrum of an audio and closely approximates the response of the human auditory system – computed as given in (Ganchev et al., 2005).
- Spectral Contrast: It is the magnitude difference in the peaks and valleys of the spectrum – computed according to (Akkermans et al., 2009).

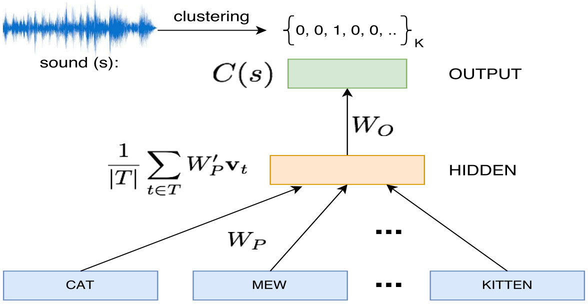

sound (s):
clustering
$\{0, 0, 1, 0, 0, ..\}_K$
$C(s)$
OUTPUT
W_O
$\frac{1}{|T|}\sum_{t \in T} W'_P \mathbf{v}_t$
HIDDEN
W_P
...
CAT
MEW
...
KITTEN

Figure 1: The model used to learn the proposed sound-word2vec embeddings. The projection matrix W_P containing that is used as the sound-word2vec embedding is learned by training the model to accurately predict the cluster assignment of the sound.

- Dissonance: It measures the perceptual roughness of the sound (Plomp and Levelt, 1965).
- Zero-crossing Rate: It is the percentage of sign changes between consecutive signal values and is indicative of noise content.
- Spectral Spread: This feature is the concatenation of the k-order moments of the spectrum, where $k \in \{0, 1, 2, 3, 4\}$.
- Pitch Salience: This feature helps discriminate between musical and non-musical tones. While, pure tones and unpitched sounds have values near 0, musical sounds containing harmonics have higher values (Ricard, 2004).

We then use K-Means algorithm to cluster the sounds in this feature space to assign each sound to a cluster $C(s) \in \{1, \ldots K\}$. We set K to 30 by evaluating the performance of the embeddings on text-based audio-retrieval on the held out validation set. Note that the clustering is only performed once, prior to representation learning described below.

Representation Learning. We represent each tag $t \in T$ using a $|\mathcal{V}|$ dimensional one-hot encoding denoted by $\mathbf{v}_t$, where $\mathcal{V}$ is the set of all unique tags in the training set (the size of our dictionary). This one-hot vector $\mathbf{v}_t$ is projected into a D-dimensional vector space via $W_P \in \mathbb{R}^{|\mathcal{V}| \times D}$, the *projection matrix*. This projection matrix computes the representation for each word in $\mathcal{V}$. The idea of our approach is to use W_P to accurately predict cluster assignments (for sounds associated with words), which enforces grounding in sound. For each data-point, we obtain the summary of the tags T, by averaging the projections of all tags in the set as $\frac{1}{|T|}\sum_{t \in T} W'_P \mathbf{v}_t$. We then transform the so obtained summary representation via a linear layer (with parameters W_O) and pass the output through the softmax function to obtain a distribution, $p(c|T)$ over the K sound clusters. We perform maximum-likelihood training for the correct cluster assignment $C(s)$[2], optimizing for parameters W_P and W_O:

$$\max_{W_P, W_O} \log P(c = C(s)|T) \tag{1}$$

We use SGD with momentum to optimize this objective which essentially is the cross-entropy between cluster assignments and $p(c|T)$. We set D to 300 to be consistent with the publicly available word2vec embeddings.

Initialization. We initialize W_P with word2vec embeddings (Mikolov et al., 2013) trained on the Google news corpus dataset with ∼3M words. We fine-tune on a subset of 9578 tags which are present in both Freesound as well as Google news corpus datasets, which is 55.68% of the original tags in the Freesound dataset. This helps us remove noisy tags unrelated to the content of the sound.

In addition to enlarging the vocabulary, the pre-training helps induce smoothness in the sound-word2vec embeddings – allowing us to transfer semantics learnt from sounds to words that were not present as tags in the Freesound database. Indeed, we find that word2vec pre-training helps improve performance (Sec. 5.3). Our use of language embeddings as an initialization to fine-tune (specialize) from, as opposed to formulating a joint objective with language and audio context (Kiela and Clark, 2015) is driven by the fact that we are interested in embeddings for words grounded in sounds, and not better generic word similarity.

5 Results

Ablations. In addition to the language-only baseline word2vec (Mikolov et al., 2013), we compare against tag-word2vec – that predicts a tag using other tags of the sound as context, inspired by (Font et al., 2014). We also report results with a randomly initialized projection matrix (sound-word2vec(r) to evaluate the effectiveness of pre-training with word2vec.

Prior work. We compare against previous works Lopopolo and van Miltenburg (2015) and Kiela and Clark (2015). While the former uses a standard bag of words and SVD pipeline to arrive at distributional representations for words, the latter trains under a joint objective that respects both linguistic and auditory similarity. We use the openly available implementation for Lopopolo and van Miltenburg (2015) and re-implement Kiela and Clark (2015) and train them on our dataset for a fair comparison of the methods. In addition, we show a comparison to word-vectors released by (Kiela and Clark, 2015) in the supplementary material. All approaches use an embedding size of 300 for consistency.

5.1 Text-based Sound Retrieval

Given a textual description of a sound as query, we compare it with tags associated with sounds in the database to retrieve the sound with the closest matching tags. Note that this is a purely textual task, albeit one that needs awareness of sound. In a sense, this task exactly captures what we want our model to be able to do – bridge the semantic gap between language and sound. We use the training split (Sec. 3) to learn the sound-word2vec vectors, validation to pick the number of clusters (K), and report results on the test split. For retrieval, we represent sounds by averaging the learnt embeddings for the associated tags. We embed the caption provided for the sound (in the Freesound database) in a similar manner, and use it as the query. We then rank sounds based on the cosine similarity between the tag and query representations for retrieval. We evaluate using standard retrieval metrics – `Recall@{1,10,50,100}`. Note that the entire testing set (≈10k sounds) is present in the retrieval pool. So, `recall@100` corresponds to obtaining the correct result in the top 1% of the search results, which is a relatively stringent evaluation criterion.

Results. Table. 1 shows that our sound-word2vec embeddings outperform the baselines. We see that specializing the embeddings for sound using our two-stage training outperforms prior work(Kiela and Clark (2015) and Lopopolo and van Miltenburg (2015)), which did not do specialization. Among our approaches, tag-word2vec performs second best – this is intuitive since the tag distributions implicitly capture auditory relatedness (a sound may have tags *cat* and *meow*), while word2vec and sound-word2vec(r) have the lowest performance.

[2] We also tried to regress directly to sound features instead of clustering, but found that it had poor performance.

Embedding	Recall			
	@1	@10	@50	@100
word2vec	6.47±0.00	14.25±0.05	21.72±0.12	26.03±0.22
tag-word2vec	6.95±0.02	15.10±0.03	22.43±0.09	27.21±0.24
sound-word2vec(r)	6.49±0.00	14.98±0.03	21.96±0.11	26.43±0.20
(Lopopolo and van Miltenburg, 2015)	6.48±0.02	15.09±0.05	21.82±0.13	26.89±0.23
(Kiela and Clark, 2015)	6.52±0.01	15.21±0.03	21.92±0.08	27.74±0.21
sound-word2vec	**7.11**±0.02	**15.88**±0.04	**23.14**±0.09	**28.67**±0.17

Table 1: Text-based sound retrieval (*higher* is better). We find that our sound-word2vec model outperforms all baselines.

Embedding	Spearman Correlation ρ_s	
	AMEN	ASLex
(Lopopolo and van Miltenburg, 2015)	0.410±0.09	0.237±0.04
(Kiela and Clark, 2015)	0.648±0.08	0.366±0.11
sound-word2vec	**0.674**±0.05	**0.391**±0.06

Table 2: Comparison to state of the art AMEN and ASLex datasets (Kiela and Clark, 2015) (*higher* is better). Our approach performs better than Kiela and Clark (2015).

5.2 Foley Sound Discovery

In this task, we evaluate how well embeddings identify matching pairs of target sounds (*flapping bird wings*) and descriptions of Foley sound production techniques (*rubbing a pair of gloves*). Intuitively, one expects sound-aware word embeddings to do better at this task than sound-agnostic ones. We setup a ranking task by constructing a set of original Foley sound pairs and decoy pairs formed by pairing the target description with every word from the vocabulary. We rank using cosine similarity between the average word-vectors in each member of the pair. A good embedding is one in which the original Foley sound pair has the lowest rank. We use the mean rank of the Foley sound in the dataset for evaluation. We transfer the embeddings from Sec. 5.1 to this task, without additional training.

Results. We find that Sound-word2vec performs the best with a mean rank of 34.6 compared to other baselines tag-word2vec (38.9), sound-word2vec(r) (114.3) and word2vec (189.45). As observed previously, the second best performing approach is tag-word2vec. Lopopolo and van Miltenburg (2015) and Kiela and Clark (2015) perform worse than tag-word2vec with a mean rank of 48.4 and 42.1 respectively. Note that random chance gets a rank of $(|\mathcal{V}| + 1)/2 = 4789.5$.

word	word2vec	sound-word2vec
apple	apples, pear, fruit berry, pears, strawberry	bite, snack, chips chew, munch, carton
wood	lumber, timber, softwoods, hardwoods, cedar, birch	wooden, snap, knock, smack, whack, snapping
bones	skull, femur, skeletons, thighbone, pelvis, molar	eggshell, carrot, arm blood, polystyrene, crunch
glass	hand-blown, glassware, tumbler, Plexiglass, wine-glass, bottle	shattered, ceramic, smash clink, beer, spoon
Onomatopoeic query words		
boom	booms, booming, bubble, craze, downturn, upswing	bomb, bang, explosion bombing, exploding, ecstatic
jingle	song, commercial, catchy-tune, ditty, slogan, anthem	magic, tinkle, nails bells, key, doorbell
slam	slams, piledriver, uranage spinkick, hiptoss, hit	shut, lock, opening closing, latch, door
quack	charlatan, quackery, crackpot homeopaths, concoctions, snake-oil	duck, snort, calling chirp, tweet, oink

Table 3: We show nearest neighbors in both word2vec and sound-word2vec spaces for eight words ('regular' words, top half and onomatopoeic words, bottom half).

5.3 Evaluation on AMEN and ASLex

AMEN and ASLex (Kiela and Clark, 2015) are subsets of the MEN and SimLex-999 datasets for word relatedness grounded in sound. From Table 2, we can see that our embeddings outperform (Kiela and Clark, 2015) on both AMEN and ASLex. These datasets were curated by annotating concepts related by sound; however we observe that relatedness is often confounded. For example, *(river, water), (automobile, car)* are marked as aurally related however they do not stand out as aurally-related examples as they are already semantically related. In contrast, we are interested in how onomatopoeic words relate to regular words (Table 3), which we study by explicit grounding in sound. Thus while we show competitive performance on this dataset, it might not be best suited for studying the benefits of our approach.

6 Discussion and Conclusion

We show nearest neighbors in both sound-word2vec and word2vec space (Table 3) to qualitatively demonstrate the unique dependencies captured due to auditory grounding. While word2vec maps a word (say, *apple*) to other semantically similar words (other fruits), similar 'sounding' words (*chips*) or onomatopoeia (*munch*) are closer in our embedding space. Moreover, onomatopoeic words (say, *boom* and *slam*) are mapped to relevant objects (*explosion* and *door*). Interestingly, parts (*e.g.*, *lock, latch*) and actions (*closing*) are also closer to the onomatopoeic query – exhibiting an understanding of the auditory scene.

Conclusion. In this work we introduce a novel word embedding scheme that respects auditory grounding. We show that our embeddings provide strong performance on text-based sound retrieval, Foley sound discovery along with intuitive nearest neighbors for onomatopoeia that are tasks in text requiring auditory reasoning. We hope our work motivates further efforts on understanding and relating onomatopoeia words to "regular" words.

Acknowledgements. We thank the Freesound team and Frederic Font in particular for helping us with the Freesound API. We also thank Khushi Gupta and Stefan Lee for fruitful discussions. This work was funded in part by an NSF CAREER, ONR Grant N00014-16-1-2713, ONR YIP, Sloan Fellowship, Allen Distinguished Investigator, Google Faculty Research Award, Amazon Academic Research Award to DP.

References

Vincent Akkermans, Joan Serrà, and Perfecto Herrera. 2009. Shape-based spectral contrast descriptor. In *Proc. of the Sound and Music Computing Conf.(SMC)*.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research (JAIR)*.

Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung-yi Lee, and Lin-Shan Lee. 2016. Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder. *CoRR*.

epicsound. accessed 23-Jan-2017. The guide to sound effects.

Frederic Font, Sergio Oramas, György Fazekas, and Xavier Serra. 2014. Extending tagging ontologies with domain specific knowledge. In *Proceedings of the International Semantic Web Conference*.

Frederic Font, Gerard Roma, and Xavier Serra. 2013. Freesound technical demo. In *Proceedings of the 21st ACM International Conference on Multimedia*.

Todor Ganchev, Nikos Fakotakis, and George Kokkinakis. 2005. Comparative evaluation of various mfcc implementations on the speaker verification task. In *Proceedings of the SPECOM*.

Wanjia He, Weiran Wang, and Karen Livescu. 2016. Multi-view recurrent neural acoustic word embeddings. *arXiv preprint arXiv:1611.04496*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Douwe Kiela and Stephen Clark. 2015. Multi-and cross-modal semantics beyond vision: Grounding in auditory perception. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Satwik Kottur, Ramakrishna Vedantam, Jose M. F. Moura, and Devi Parikh. 2016. Visual word2vec (vis-w2v): Learning visually grounded word embeddings using abstract scenes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. 2009. Learning to detect unseen object classes by betweenclass attribute transfer. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skip-gram model. *arXiv preprint arXiv:1501.02598*.

Alessandro Lopopolo and Emiel van Miltenburg. 2015. Sound-based distributional models. *IWCS 2015*.

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016. The role of context types and dimensionality in learning word embeddings. *arXiv preprint arXiv:1601.00893*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*.

A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. 2016. Ambient Sound Provides Supervision for Visual Learning. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Devi Parikh and Kristen Grauman. 2011. Relative Attributes. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*.

Reinier Plomp and Willem Johannes Maria Levelt. 1965. Tonal consonance and critical bandwidth. *The journal of the Acoustical Society of America*.

Julien Ricard. 2004. Towards computational morphological description of sound. *DEA pre-thesis research work, Universitat Pompeu Fabra, Barcelona*.

Shane Settle and Karen Livescu. 2016. Discriminative acoustic word embeddings: Recurrent neural network-based approaches. *arXiv preprint arXiv:1611.02550*.

Philip R. Singer. accessed 23-Jan-2017. Art of foley.

Shiva Sundaram and Shrikanth Narayanan. 2006. Vector-based representation and clustering of audio using onomatopoeia words. In *In Proceedings of AAAI 2006 Fall Symposia*.

Sanae Wake and Toshiyuki Asahi. 1998. Sound retrieval with intuitive verbal expressions. In *Proceedings of the 5th International Conference on Auditory Display (ICAD)*.

The Promise of Premise: Harnessing Question Premises in Visual Question Answering

Aroma Mahendru[*,1] **Viraj Prabhu**[*,1] **Akrit Mohapatra**[*,1] **Dhruv Batra**[2] **Stefan Lee**[1]
[1]Virginia Tech [2]Georgia Institute of Technology
{maroma, virajp, akrit}@vt.edu, dbatra@gatech.edu, steflee@vt.edu

Abstract

In this paper, we make a simple observation that questions about images often contain *premises* – objects and relationships implied by the question – and that reasoning about premises can help Visual Question Answering (VQA) models respond more intelligently to irrelevant or previously unseen questions.

When presented with a question that is irrelevant to an image, state-of-the-art VQA models will still answer purely based on learned language biases, resulting in nonsensical or even misleading answers. We note that a visual question is irrelevant to an image if at least one of its premises is false (*i.e.* not depicted in the image). We leverage this observation to construct a dataset for Question Relevance Prediction and Explanation (QRPE) by searching for false premises. We train novel question relevance detection models and show that models that reason about premises consistently outperform models that do not.

We also find that forcing standard VQA models to reason about premises during training can lead to improvements on tasks requiring compositional reasoning.

1 Introduction

The task of providing natural language answers to free-form questions about an image – *i.e.* Visual Question Answering (VQA) – has received substantial attention in the past few years (Malinowski and Fritz, 2014; Antol et al., 2015; Malinowski et al., 2015; Zitnick et al., 2016; Kim et al., 2016; Wu et al., 2016; Lu et al., 2016; Andreas et al., 2016; Lu et al., 2017) and has quickly become a popular problem area. Despite significant progress on VQA benchmarks (Antol et al., 2015), current models still present a number of unintelligent and problematic tendencies.

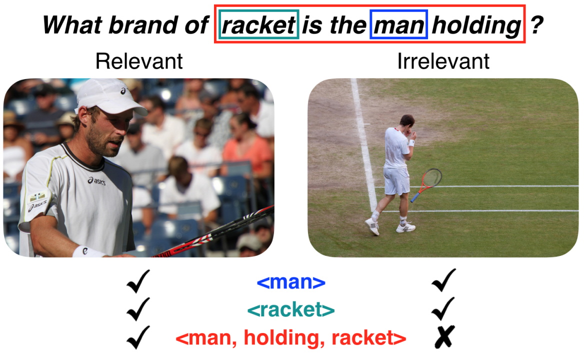


Figure 1: Questions asked about images often contain *'premises'* that imply visual semantics. From the above question, we can infer that a relevant image must contain a man, a racket, and that the man must be holding the racket. We extract these premises from visually grounded questions and use them to construct a new dataset and models for question relevance prediction. We also find that augmenting standard VQA training with simple premise-based questions results in improvements on tasks requiring compositional reasoning.

When faced with questions that are irrelevant or not applicable for an image, current 'forced choice' models will still produce an answer. For example, given an image of a dog and a query "*What color is the bird?*", standard VQA models might answer "*Red*" confidently, based solely on language biases in the training set (*i.e.* an overabundance of the word "red"). In these cases, the predicted answers are senseless at best and misleading at worst, with either case posing serious problems for real-world applications. Like Ray et al. (2016), we argue that practical VQA systems must be able to identify and explain irrelevant questions. For instance, a more intelligent VQA model with this capability might answer "*There is no bird in the image*" for this example.

[*] Denotes equal contribution.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 926–935
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Premises. In this paper, we show that question *premises* - *i.e.* objects and relationships implied by a question - can enable VQA models to respond more intelligently to irrelevant or previously unseen questions. We develop a premise extraction pipeline based on SPICE (Anderson et al., 2016) and demonstrate how these premises can be used to improve modern VQA models in the face of irrelevant or previously unseen questions.

Concretely, we define premises as facts implied by the language of questions, for example the question *"What brand of racket is the man holding?"* shown in Fig. 1 implies the existence of a man, a racket, and that the man is holding the racket. For visually grounded questions (*i.e.* those asked about a particular image) these premises imply visual qualities, including the presence of objects as well as their attributes and relationships.

Broadly speaking, we explore the usefulness of premises in two settings – when visual questions are known to be relevant to the images they are asked on (*e.g.* in the VQA dataset) and in real-life situations where such an assumption cannot be made (*e.g.* when generated by visually impaired users). In the former case, we show that knowing that a question is relevant allows us to perform data augmentation by creating additional simple question-answer pairs using the premises of source questions. In the latter case, we show that explicitly reasoning about premises provides an effective and interpretable way of determining whether a question is relevant to an image.

Irrelevant Question Detection. We consider a question to be relevant to an image if all of the question's premises apply to the corresponding image, that is to say all objects, attributes, and interactions implied by the question are depicted in the image. We refer to premises that apply for a given image as true premises and those that do not apply as false premises. In order to train and evaluate models for this task, we curate a new irrelevant question detection dataset which we call the Question Relevance Prediction and Explanation (QRPE) dataset. QRPE is automatically curated from annotations already present in existing datasets, requiring no additional labeling.

We collect the QRPE dataset by taking each image-question pair in the VQA dataset (Antol et al., 2015) and finding the most visually similar other image for which exactly one of the question premises is false. In this way, we collect tuples consisting of two images, a question, and a premise where the question is relevant for one image and not for the other due to the premise being false.

For context, the only other existing irrelevant question detection dataset (Ray et al., 2016) collected irrelevant question-image pairs by human verification of random pairs. In comparison, QRPE is substantially larger, balanced between irrelevant and relevant examples, and presents a considerably more difficult task due to the closeness of the image pairs both visually and with respect to question premises.

We train novel models for irrelevant question detection on the QRPE dataset and compare to existing methods. In these experiments, we show that models that explicitly reason about question premises consistently outperform baseline models that do not.

VQA Data Augmentation. Finally, we also introduce an approach to generate simple, templated question-answer pairs about elementary concepts from premises of complex training questions. In initial experiments, we show that adding these simple question-answer pairs to VQA training data can improve performance on tasks requiring compositional reasoning. These simple questions improve training by bringing implicit training concepts "to the surface", *i.e.* introducing direct supervision of important implicit concepts by transforming them to simple training pairs.

2 Related Work

Visual Question Answering: Starting from simple bag-of-word and CNN+LSTM models (Antol et al., 2015), VQA architectures have seen considerable innovation. Many top-performing models integrate attention mechanisms (over the image, the question, or both) to focus on important structures (Fukui et al., 2016; Lu et al., 2016, 2017), and some have been designed with compositionality in mind (Andreas et al., 2016; Hendricks et al., 2016). However, improving compositionality or performance through data augmentation remains a largely unstudied area.

Some other recent work has developed models which produce natural language explanations for their outputs (Park et al., 2016; Wang et al., 2016), but there has not been work on generating explanations for irrelevant questions or false premises.

Question Relevance: Most related to our work is that of Ray et al. (2016), which introduced the task of irrelevant question detection for VQA. To evaluate on this task, they created the Visual True and False Question (VTFQ) dataset by pairing VQA questions with random VQA images and having human annotators verify whether or not the question was relevant. As a result, many of the irrelevant image-question pairs exhibit a complete mismatch of image and question content. Our Question Relevance Prediction and Explanation (QRPE) dataset on the other hand is collected such that irrelevant images for each question closely resemble the source image both visually and semantically. We also provide premise-level annotations which can be used to develop models that not only decide whether a question is relevant, but also provide explanations for *why* that is the case.

Semantic Tuple Extraction: Extracting structured facts in the form of semantic tuples from text is a well studied problem (Schuster et al., 2015; Anderson et al., 2016; Elhoseiny et al., 2016); however, recent work has begun extending these techniques to visual domains (Xu et al., 2017; Johnson et al., 2015). Additionally, the Visual Genome (Krishna et al., 2016) dataset contains dense image annotations for objects and their attributes and relationships. However, we are the first to consider these facts to reason about question relevancy and compositionality in VQA.

3 Extracting Premises of a Question

In Section 1, we introduced the concept of premises and how they can be used. We now formalize this concept and explain how premises can be extracted from questions.

We define question premises as facts implied about an image from a question asked about it, which we represent as tuples. Returning to our running example question *"What brand of racket is the man holding?"*, we can express these premises as the tuples '<*man*>', '<*racket*>', and '<*man, holding, racket*>' respectively. We categorize these tuples into three groups based on their complexity. First-order premises representing the presence of objects ('<*man*>', '<*cat*>', '<*sky*>'), second-order premises capturing the attributes of objects ('<*man, tall*>', '<*car, moving*>'), and third-order premises containing interactions between objects (*e.g.* '<*man, kicking, ball*>', '<*cat, above, car*>').

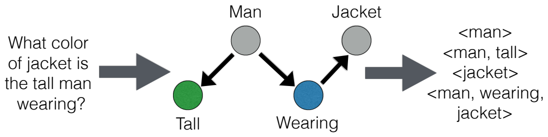


Figure 2: **Premise Extraction Pipeline.** Objects (gray), attributes (green), and relations (blue) scene graph nodes are converted into 1st, 2nd, and 3rd order premises respectively.

Premise Extraction: To extract premises from questions, we use the semantic tuple extraction pipeline used in the SPICE metric (Anderson et al., 2016). Originally defined as a metric for image captioning, SPICE transforms a sentence into a scene graph using the Stanford Scene Graph Parser (Schuster et al., 2015) and then extracts semantic tuples from this representation. Fig. 2 shows this process for a sample question. The question is represented as a graph of objects, attributes, and relationships from which first, second, and third order premises are extracted respectively. As this pipeline was originally designed for descriptive captions rather than questions, we found a number of minor modifications helpful in extracting quality question premises, including disabling pronoun resolution, verb lemmatization and METEOR-based Synset matching. We will release our premise extraction code publicly to encourage reproducibility.

While this extraction process typically produces high quality premise tuples, there are some sources of noise which must be filtered out. The SPICE process occasionally produces duplicate nodes or object nodes not linked to nouns in the question, which we filter out. We also remove premises containing words like photo, image, *etc.* that refer to the image rather than its content.

A more nuanced source of erroneous premises comes from the ambiguity in existential questions, *i.e.* those about the existence of certain image content. For example, while the question *"Is the little girl moving?"* contains the premise '<*girl, little*>', it is unclear without the answer whether '<*girl, moving*>' is also a premise. Similarly, for the question *"How many giraffes are in the image?"*, '<*giraffe, many*>' cannot be considered a premise as there may be 0 giraffes in the image. To avoid introducing false premises, we filter out existential and counting questions.

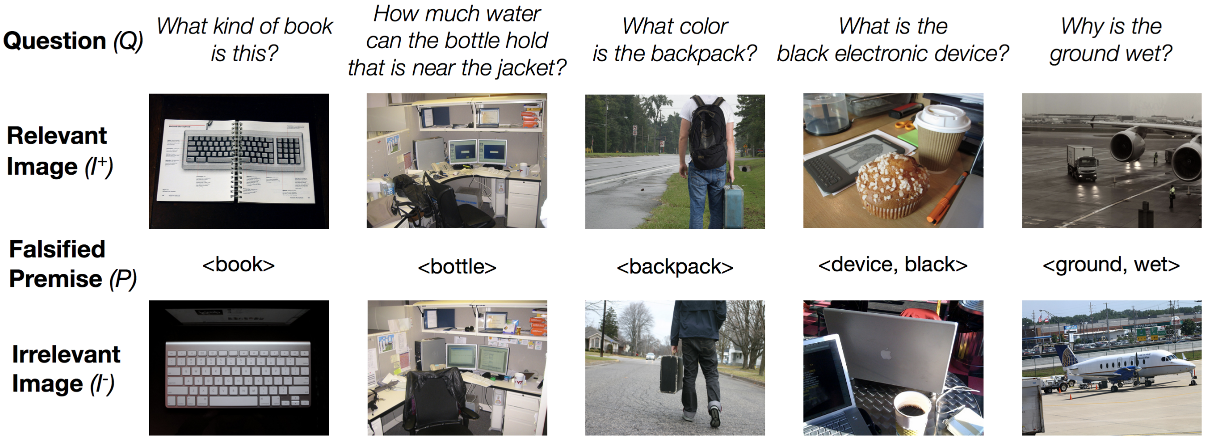


Figure 3: **Some Examples from QRPE Dataset.** For a given question Q and a relevant image I^+, we find an irrelevant image I^- for which exactly one premise P of the question is false. If there are multiple such candidates, we select the candidate most visually most similar to I^+. As can be seen from these examples, the QRPE dataset is very challenging, with only minor visual and semantic differences separating the relevant and irrelevant images.

4 Question Relevance Prediction and Explanation (QRPE) Dataset

As discussed in Section 1, modern VQA models fail to differentiate between relevant and irrelevant questions, answering either with confidence. This behavior is detrimental to the real world application of VQA systems. In this section, we curate a new dataset for question relevance in VQA which we call the Question Relevance Prediction and Explanation (QRPE) dataset. We plan to release QRPE publicly to help future efforts.

In order to train and evaluate models for irrelevant question detection, we would like to create a dataset of tuples (I^+, Q, P, I^-) comprised of a natural language question Q, an image I^+ for which Q is relevant, and an image I^- for which Q is irrelevant because premise P is false. While it is not required to collect both a relevant and irrelevant image for each question, we argue that doing so is a simple way to balance the dataset and it ensures that biases against rarer questions (which would be irrelevant for most images) cannot be exploited to inflate performance.

We base our dataset on the existing VQA corpus (Antol et al., 2015), taking the human-generated (and therefore relevant) image-question pairs from VQA as I^+ and Q. As previously discussed, we can define the relevancy of a question in terms of the validity of its premises for an image, so we extract premises from each question Q and must find a suitable irrelevant image I^-. However, there are certainly many images for which one or more of Q's premises are false and an important design decision is then how to select I^- from this set.

To ensure our dataset is as realistic and challenging as possible, we consider irrelevant images which only have a single false question premise under Q which we denote P. For example, the question *"Is the big red dog old?"* could be matched with an image containing a big, white dog or a small red dog, but not a small white dog. In this way, we ensure that image content is semantically appropriate for the question topic but not quite relevant. Additionally, this provides each irrelevant image with an explanation for why the question does not apply.

Furthermore, we sort this subset of irrelevant image by their visual distance to the source image I^+ based on image encodings from a VGGNet (Simonyan and Zisserman, 2014) pretrained on ImageNet (Russakovsky et al., 2012). This ensures that the relevant and irrelevant images are visually similar and act as difficult examples.

A major difficulty with our proposed data collection process is how to verify whether a premise if true or false for any given image in order to identify irrelevant images. We detail dataset construction and our approach for this problem in the following section.

4.1 Dataset Construction

We curate our QRPE dataset automatically from existing annotations in COCO (Lin et al., 2014) and Visual Genome (Krishna et al., 2016). COCO is a set of over 300,000 images annotated with object segmentations and presence information for 80 classes as well as text descriptions of image content. Visual Genome builds on this dataset, providing more detailed object, attribute, and rela-

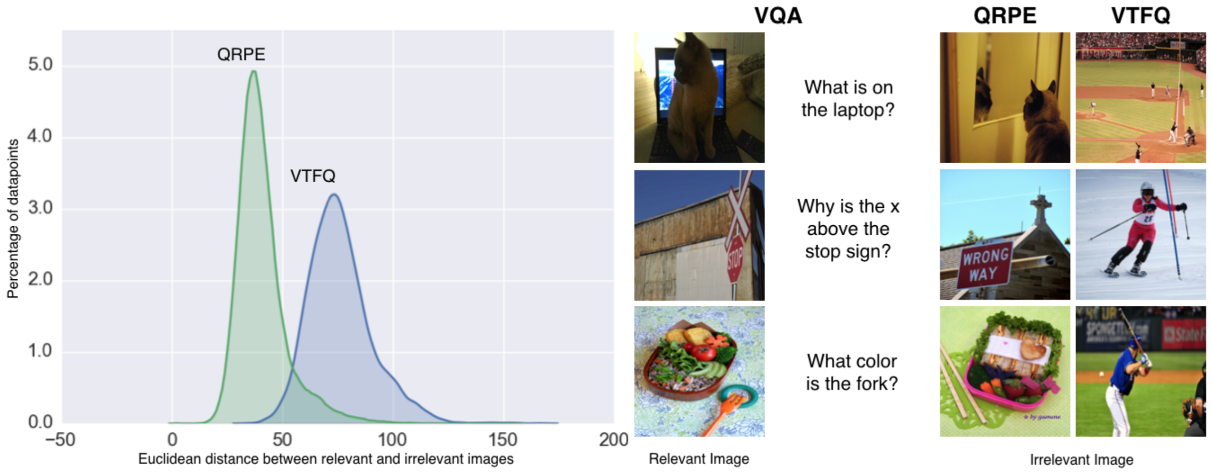


Figure 4: **A comparison of the QRPE and VTFQ Datasets.** On the left, we plot the Euclidean distance between VGGNet-fc7 features extracted from each relevant-irrelevant image pair for each dataset. Note that VTFQ has significantly higher visual distances. On the right, we show some qualitative examples of irrelevant images for questions that occur in both datasets. VTFQ images are significantly less related to the source image and question than in our dataset.

tionship annotations for over 100,000 COCO images. We make use of these data sources to extract first and second order premises from VQA questions which are also based on COCO images.

For first order premises (*i.e.* existential premises), we consider only the 80 classes present in COCO (Lin et al., 2014). As VQA and COCO share the same images, we can easily determine if a first order premise is true or false for a candidate irrelevant image simply by checking for the absence of the appropriate class annotation.

For second order premises (*i.e.* attributed objects), we rely on Visual Genome (Krishna et al., 2016) annotations for object and attribute labels. Unlike in COCO, the lack of a particular object label in an image for Visual Genome does not necessarily indicate that the object is not present, both due to annotation noise and the use of multiple synonyms for objects by human labelers. As a consequence, we restrict the set of candidate irrelevant images to those which contain a matching object to the question premise but a different attribute. Without further restriction, the selected irrelevant attributes do not tend to be mutually exclusive with the source attribute (*i.e.* matching '<*dog, old*>' and '<*dog, red*>'). To correct this and ensure a false premise, we further restrict the set to attributes which are antonyms (*e.g.* '<*young*>' for source attribute '<*old*>') or taxonomic sister terms (*e.g.* '<*green*>' for source attribute '<*red*>') of the original premise attribute. We also experimented with third order premises; however, the lack of a corresponding sense of mutual exclusion for verbs and the sparsity of <object, relationship, object> premises made finding non-trivial irrelevant images difficult.

To recap, our data collection approach is to take each image-question pair in the VQA dataset and extract its first and second order question premises. For each premise, we find all images which lack only this premise and rank them by their visual distance. The closest of these is kept as the irrelevant image for each image-question pair.

4.2 Exploring the Dataset

Fig. 3 shows sample (I^+, Q, P, I^-) tuples from our dataset. These examples illustrate the difficulty of our dataset. For instance, the images in the second column differ only by the presence of the water bottle and images in the fourth column are differentiated by the color of the devices. Both of these are fine details of the image content.

The QRPE dataset contains 53,911 (I^+, Q, P, I^-) tuples generated from as many premises. In total, it contains 1530 unique premises and 28,853 unique questions. Among the 53,911 premises, 3876 are second-order, attributed object premises while the remaining 50,035 are first-order object/scene premises. We divide our dataset into two parts – a training set with 35,486 tuples that are generated from the VQA training set and a validation set with 18,425 tuples generated from the VQA validation set.

Manual Validation. We also manually validated 1000 randomly selected (I^+, Q, P, I^-) tuples from our dataset. We noted that 99.10% of the premises P were valid (*i.e.* implied by the question) in I^+ and 97.3% were false for the negative image I^-. This demonstrates the high reliability of our automated annotation pipeline.

4.3 Comparison to VTFQ

We contrast our approach to the VTFQ dataset of Ray et al. (2016). As discussed prior, VTFQ was collected by selecting a random question and image from the VQA set and asking human annotators to report if the question was relevant, producing a pair. This approach results in irrelevant image-question pairs that are unambiguously unrelated, with the visual content of the image having nothing at all to do with the question or its source image from VQA.

To quantify this effect and compare to QRPE, we pair each irrelevant image-question pair (I^-, Q) from VTFQ with a relevant image from the VQA dataset. Specifically, we find the nearest neighbor question Q^{nn} in the VQA dataset to Q based on an average of the word2vec (Mikolov et al., 2013) embedding of each word, and select the image on which Q^{nn} was asked as I^+ to form (I^+, Q, P, I^-) tuples like in our proposed dataset.

In Fig. 4, we present a quantitative and qualitative comparison of the two datasets based on these tuples. On the left side of the figure, we plot the distributions of Euclidean distance between the fc7 features of each (I^+, I^-) pair in both datasets. We find that the mean distance in the VTFQ dataset is nearly twice that of our QRPE dataset, indicating that irrelevant images in VTFQ are less visually related to source images though we do note the distribution of distances in both datasets is long tailed.

On the right side of Fig. 4, we also provide qualitative examples of questions that occur in both datasets. The example on the last row is perhaps most striking. The source question is asking the color of a fork and the relevant image shows an overhead view of a meal with an orange fork set nearby. The irrelevant image in QRPE is a similar image of food, but with chopsticks! Conversely, the image from VTFQ is a man playing baseball.

5 Question Relevance Detection

In this section, we introduce a simple baseline for irrelevant question detection on the QRPE dataset and demonstrate that explicitly reasoning about premises improves performance for both our new model and existing methods. More formally, we consider the binary classification task of predicting if a question Q_i from an image-question pair (I_i, Q_i) is relevant to image I_i.

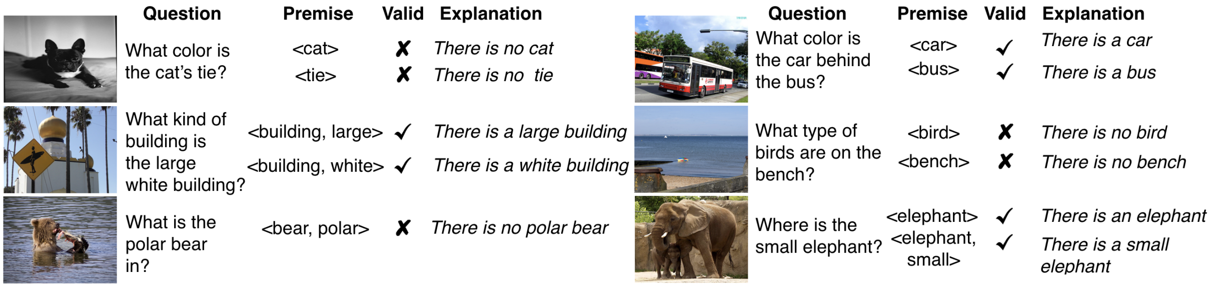
A Simple Premise-Aware Model. Like the standard VQA task, question relevance detection also requires making a prediction based on an encoded image and question. With this in mind, we begin with a straight-forward approach based on the Deeper LSTM VQA model architecture of Antol et al. (2015). This model encodes the image I via a VGGNet and the question Q with an LSTM over one-hot word encodings. The concatenation of these embeddings are input to a multi-layer perceptron. We fine-tune this model for the binary question relevance detection task starting from a model pretrained on the VQA task. We denote this model as `VQA-Bin`.

We extend the `VQA-Bin` model to explicitly reason about premises. We extract first and second order premises from the question Q and encode them as two concatenated one-hot vectors. We add an additional LSTM to encode the premises and concatenate this added feature to the image and question feature. We refer to this premise-aware model as `VQA-Bin-Premise`.

Attention Models. We also extend the attention based Hierarchical Co-Attention VQA model of Lu et al. (2016) for the task of question relevance in a way similar to Deeper LSTM model. We call this model `HieCoAtt-Bin`. The corresponding premise-aware model is referred to as `HieCoAtt-Bin-Prem`.

Existing Methods. We compare our approaches with the best performing model of Ray et al. (2016). This model (which we denote `QC-Sim`) uses a pretrained captioning model to automatically provide natural language image descriptions and reasons about relevance based on a learned similarity between the question and image caption.

Specifically, the approach uses NeuralTalk2 (Karpathy and Li, 2015) trained on the MS COCO dataset (Lin et al., 2014) to generate a caption for each image. Both the caption and question are



Question	Premise	Valid	Explanation
What color is the cat's tie?	<cat> <tie>	✗ ✗	*There is no cat* *There is no tie*
What kind of building is the large white building?	<building, large> <building, white>	✓ ✓	*There is a large building* *There is a white building*
What is the polar bear in?	<bear, polar>	✗	*There is no polar bear*

Question	Premise	Valid	Explanation
What color is the car behind the bus?	<car> <bus>	✓ ✓	*There is a car* *There is a bus*
What type of birds are on the bench?	<bird> <bench>	✗ ✗	*There is no bird* *There is no bench*
Where is the small elephant?	<elephant> <elephant, small>	✓ ✓	*There is an elephant* *There is a small elephant*

Figure 5: **Question relevance explanation:** We provide selected examples of predictions from the False Premise Detection model (`FPD`) on the QRPE test set. Reasoning about premises presents the opportunity to produce natural language statements indicating *why* a question is irrelevant to an image, by pointing to the premise that is invalid.

Models	Overall	First Order	Second Order
VQA-Bin	66.50	67.36	53.00
VQA-Bin-Prem	66.77	67.04	54.38
HieCoAtt-Bin	70.74	71.35	**61.54**
HieCoAtt-Bin-Prem	73.34	73.97	60.35
QC-Sim	74.35	75.82	55.12
PC-Sim	75.05	76.47	56.04
QPC-Sim	**75.31**	**76.67**	55.95

Table 1: Accuracy of Question Relevance models on the QRPE test set. We find that premise-aware models consistently outperform alternative models.

embedded as a fixed length vector through an encoding LSTM (with words being represented as word2vec (Mikolov et al., 2013) vectors). These question and caption embeddings are concatenated and fed to a multilayer perceptron to predict relevance. We consider two additional versions of this approach that consider only premise-caption similarity (`PC-Sim`) and question-premise-caption similarities (`QPC-Sim`).

Results. We train each model on the QRPE train split and report results on the test set in Table 1. As the dataset is balanced in the label space, random accuracy stands at 50%. We find that the simple `VQA-Bin` model achieves 66.5% accuracy while the attention based model `HieCoAtt-Bin` attains 70.74% accuracy. Surprisingly, the caption-similarity based `QC-Sim` model significantly outperforms these baseline, obtaining an accuracy of 74.35% while only reasoning about relevancy from textual descriptions of images. We note that the caption similarity based approaches use a large amount of outside data during pretraining of the captioning model and the word2vec embeddings, which may have contributed to the effectiveness of these methods.

Most interestingly, we find that the addition of extracted premise representations consistently improves performance of base models. `VQA-Bin-Prem`, `HieCoAtt-Bin-Prem`, `PC-Sim`, and `QPC-Sim` outperform their no-premise information counterparts, with `QPC-Sim` being the overall best performing approach at 75.31% accuracy. This is especially interesting given that the models *already* have access to the question from which the premises were extracted. This result seems to imply there is value in explicitly isolating premises from sentence grammar.

We further divide our test set into two splits consisting of (Q, I) pairs created by either falsifying first-order and second-order premises. We find that all our models perform significantly better on the first-order split. We hypothesize that the significant diversity in visual representations of attributed objects and comparatively fewer examples for each type makes it more difficult to learn subtle differences for second-order premises.

5.1 Question Relevance Explanation

In addition to identifying whether a question is irrelevant to an image, being able to indicate *why* carries significant real-world utility. From an interpretability perspective, reporting which premise is false is more informative than simply answering the question in the negative, as it can help to correct the questioner's misconception regarding the scene. We propose to generate such explanations by identifying the particular question premise(s) that do not apply to an image.

By construction, irrelevant images in the QRPE dataset are picked on the basis of negating a single premise – we now use our dataset to train models to detect false premises, and use the premises classified as irrelevant to generate templated natural language explanations.

Fig. 5 illustrates the task setup for false premise detection. Given a question-image pair, say *"What color is the cat's tie?"*, the objective is to identify which (if any) question premises are not grounded in the image, in this case both <*cat*> and <*tie*>. Alternatively, for the question *"What* *kind of building is the large white building?"*, both premises <*building, large*> and <*building, white*> are true premises grounded in the image.

We train a simple false premise detection model for this task. Our model is a multilayer perceptron that takes one-hot encodings of premises and VGGNet (Simonyan and Zisserman, 2014) image features as input to predict whether the premise is grounded in the image or not. We trained our false premise detection model (`FPD`) model on all premises in the QRPE dataset.

Our `FPD` model achieves an accuracy of 61.12% on the QRPE dataset. In Fig. 5, we present qualitative results of our premise classification and explanation pipeline. For the question *"What color is the cat's tie?"*, the model correctly recognizes 'cat' and 'tie' as false premises, and we generate statements in natural language indicating the same. Thus, determining question relevance by reasoning about each premise presents the opportunity to generate simple explanations that can provide valuable feedback to the questioner, and help improve model trust.

6 Premise-Based Visual Question Answering Data Augmentation

In this section, we develop a premise-based data augmentation scheme for VQA that generates simple, templated questions based on premises present in complex visually-grounded questions from the VQA (training) dataset.

Using the pipeline presented in Section 3, we extract premises from questions in the VQA dataset and apply a simple templated question generation strategy to transform premises into question and answer pairs. Note that because the source questions come from sighted humans about an image, we do not need to filter out binary or counting questions in order to avoid false premises as in Section 3. We do however filter based on SPICE similarity between the generated and source questions to avoid generating duplicates.

We design templates for each type of premise – first-order (*e.g.* '<*man*>' – *"Is there a man?" Yes*), second-order (*e.g.* '<*man, walking*>' – *"What is the man doing?" Walking*, and '<*car, red*>' – *"What is the color of the car?" Red*), and third-order ('<*man, holding, racket*>' – *"What is the man holding?" Racket*, *"Who is holding the racket?" Man*). This process transforms implicit premise concepts which previously had to be learned as part of understanding more complex questions into simple, explicit training examples that can be directly supervised.

Training Data	Other	Number	Yes	No	Total
Source	123,817	29,698	57217	35842	246,574
Premise	137,483	1,850	387,941	0	527,274

Table 2: Answer type distribution of source and premise questions on the Compositional VQA train set.

Fig. 6 shows sample premise questions produced from source VQA questions using our pipeline. We note that the distribution of premise questions varies drastically from the source VQA distribution (see Table 2).

We evaluate multiple models with and without premise augmentation on two splits of the VQA dataset - the standard split and the compositional split of Agrawal et al. (2017). The compositional split is specifically designed to test a model's ability to generalize to unseen/rarely seen combinations of concepts at test time.

Augmentation Strategies. We evaluate the Deeper LSTM model of Lu et al. (2015) on the standard and compositional splits with two augmentation strategies - `All` which includes the entire set of premise questions and `Top-1k-A` which includes only questions with answers in the top 1000 most common VQA answers. The results are listed in Table 3. We find minor improvement of 0.34% on the standard split under `Top-1k-A` premise question augmentation. On the compositional split, we observe a 1.16% gain with `Top-1k-A` augmentation over no augmentation. In this setting, explicitly reasoning about objects and attributes seen in the questions seems to help the model disentangle objects from their common characteristics.

Other Models. To check the general effectiveness of our approach, we further evaluate `Top-1k-A` augmentation for three additional VQA models on the compositional split. We find inconsistent improvements for these more advanced models with some improving while others see reductions in accuracy when adding premises.

7 Conclusions and Future Work

In this paper, we made the simple observation that questions about images often contain premises implied by the question and that reasoning about premises can help VQA models respond more in-

What player number is about to swing at the ball? Is there a player number? Yes Is there a ball in the image? Yes Is there a number in the image? Yes	**Why is the man looking at the lady?** Who is looking at the lady? Man Is there a lady in the image? Yes Is there a man in the image? Yes	**How many people are wearing safety jackets?** Can you see people in the image? Yes What are the people wearing? Jacket Who is wearing the jacket? People
What is the child sitting on? What is the child doing? Sitting Is there a child in the image? Yes	**Where is the pink hat?** What is the color of hat? Pink Is there a hat in the image? Yes	**What is the item called that the cat is looking at?** Is there a cat in the image? Yes Is there an item in the image? Yes

Figure 6: Sample generated premise questions from source questions. Source questions are in bold. Ground-truth answers are extracted using the premise tuples.

	Augmentation	Overall	Other	Number	Yes/No
Standard	None	54.23	40.34	33.27	79.82
	All	53.74	39.28	**33.38**	79.89
	Top-1k-A	**54.47**	**40.56**	33.24	**80.19**
Comp.	None	46.69	31.92	29.73	70.49
	All	47.63	31.97	**30.77**	**72.52**
	Top-1k-A	**47.85**	**32.58**	30.59	72.38

Table 3: Accuracy on the standard and compositional VQA validation sets for different augmentation strategies for DeeperLSTM(Antol et al., 2015).

VQA Model	Baseline	+Premises
DeeperLSTM(Lu et al., 2015)	46.69	**47.85**
HieCoAtt(Lu et al., 2016)	**50.17**	49.98
NMN(Andreas et al., 2016)	**49.05**	48.43
MCB(Fukui et al., 2016)	50.13	**50.57**

Table 4: Overall accuracy of different VQA models on the Compositional VQA test split using Top-1k-A augmentation.

telligently to irrelevant or novel questions.

We develop a system for automatically extracting these question premises. Using these premises, we automatically created a novel dataset for Question Relevance Prediction and Explanation (QRPE) which consists of 53,911 question, relevant image, and irrelevant image triplets. We also train novel question relevance prediction models and show that models that take advantage of premise information outperform models that do not. Furthermore, we demonstrated that questions generated from premises may be an effective data augmentation technique for VQA tasks that require compositional reasoning.

Integrating Question Relevance Prediction and Explanation (QRPE) models with existing VQA systems would form a natural extension to our approach. In this setting, the relevance prediction model would determine the applicability of a question to an image, and select an appropriate path of action. If the question is classified as relevant, the VQA model would generate a prediction; otherwise, a question relevance explanation model would provide a natural language sentence indicating which premise(s) are not valid for the image. Such systems would be a step in the direction of making VQA systems move beyond academic settings to real-world environments.

References

A. Agrawal, A. Kembhavi, D. Batra, and D. Parikh. 2017. C-VQA: A Compositional Split of the Visual Question Answering (VQA) v1.0 Dataset. *arXiv preprint arXiv:1704.08243*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, pages 382–398. Springer.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Mohamed Elhoseiny, Scott Cohen, Walter Chang, Brian Price, and Ahmed Elgammal. 2016. Automatic annotation of structured facts in images. *arXiv preprint arXiv:1604.00466*.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. *arXiv preprint arXiv:1606.01847*.

Lisa Anne Hendricks, Subhashini Venugopalan, Marcus Rohrbach, Raymond Mooney, Saenko Kate, and

Trevor Darrell. 2016. Deep compositional captioning: Describing novel object categories without paired training data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. 2015. Image retrieval using scene graphs. In *Conference on Computer Vision and Pattern Recognition*, pages 3668–3678.

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Jin-Hwa Kim, Sang-Woo Lee, Donghyun Kwak, Min-Oh Heo, Jeonghee Kim, Jung-Woo Ha, and Byoung-Tak Zhang. 2016. Multimodal residual learning for visual qa. In *Advances in Neural Information Processing Systems*, pages 361–369.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotations.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*.

Jiasen Lu, Xiao Lin, Dhruv Batra, and Devi Parikh. 2015. Deeper lstm and normalized cnn visual question answering model. `https://github.com/VT-vision-lab/VQA_LSTM_CNN`.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning. *CVPR*.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems*, pages 1682–1690.

Mateusz Malinowski, Marcus Rohrbach, and Mario Fritz. 2015. Ask your neurons: A neural-based approach to answering questions about images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–9.

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

Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. 2016. Attentive explanations: Justifying decisions and pointing to the evidence. *arXiv preprint arXiv:1612.04757*.

Arijit Ray, Gordon Christie, Mohit Bansal, Dhruv Batra, and Devi Parikh. 2016. Question relevance in vqa: Identifying non-visual and false-premise questions. In *EMNLP*.

Olga Russakovsky, Jia Deng, Jonathan Krause, Alex Berg, and Li Fei-Fei. 2012. The ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012). http://www.image-net.org/challenges/LSVRC/2012/.

Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D Manning. 2015. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proceedings of the Fourth Workshop on Vision and Language*, pages 70–80.

K. Simonyan and A. Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Peng Wang, Qi Wu, Chunhua Shen, Anton van den Hengel, and Anthony Dick. 2016. Fvqa: Fact-based visual question answering. *arXiv preprint arXiv:1606.05433*.

Qi Wu, Chunhua Shen, Lingqiao Liu, Anthony Dick, and Anton van den Hengel. 2016. What value do explicit high level concepts have in vision to language problems? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 203–212.

Danfei Xu, Yuke Zhu, Christopher Choy, and Li Fei-Fei. 2017. Scene graph generation by iterative message passing. In *Computer Vision and Pattern Recognition (CVPR)*.

C Lawrence Zitnick, Aishwarya Agrawal, Stanislaw Antol, Margaret Mitchell, Dhruv Batra, and Devi Parikh. 2016. Measuring machine intelligence through visual question answering. *arXiv preprint arXiv:1608.08716*.

Guided Open Vocabulary Image Captioning with Constrained Beam Search

Peter Anderson[1], Basura Fernando[1], Mark Johnson[2], Stephen Gould[1]
[1]The Australian National University, Canberra, Australia
firstname.lastname@anu.edu.au
[2]Macquarie University, Sydney, Australia
mark.johnson@mq.edu.au

Abstract

Existing image captioning models do not generalize well to out-of-domain images containing novel scenes or objects. This limitation severely hinders the use of these models in real world applications dealing with images in the wild. We address this problem using a flexible approach that enables existing deep captioning architectures to take advantage of image taggers at test time, without re-training. Our method uses constrained beam search to force the inclusion of selected tag words in the output, and fixed, pretrained word embeddings to facilitate vocabulary expansion to previously unseen tag words. Using this approach we achieve state of the art results for out-of-domain captioning on MSCOCO (and improved results for in-domain captioning). Perhaps surprisingly, our results significantly outperform approaches that incorporate the same tag predictions into the learning algorithm. We also show that we can significantly improve the quality of generated ImageNet captions by leveraging ground-truth labels.

1 Introduction

Automatic image captioning is a fundamental task that couples visual and linguistic learning. Recently, models incorporating recurrent neural networks (RNNs) have demonstrated promising results on this challenging task (Vinyals et al., 2015; Fang et al., 2015; Devlin et al., 2015), leveraging new benchmark datasets such as the MSCOCO dataset (Lin et al., 2014). However, these datasets are generally only concerned with a relatively small number of objects and interactions. Unsurprisingly, models trained on these datasets do not generalize well to out-of-domain images containing novel scenes or objects (Tran et al., 2016). This limitation severely hinders the use of these models in real world applications dealing with images in the wild.

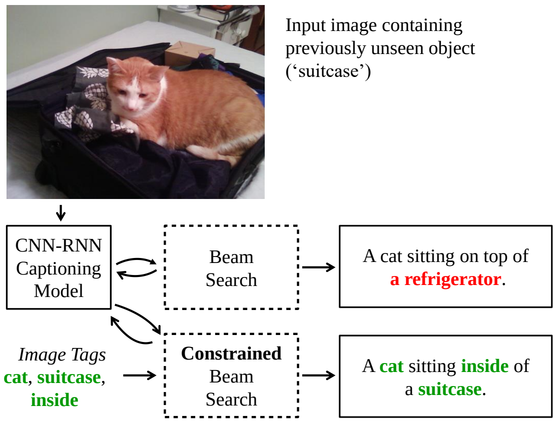

Input image containing previously unseen object ('suitcase')
CNN-RNN Captioning Model
Beam Search
A cat sitting on top of a refrigerator.
Image Tags
cat, suitcase, inside
Constrained Beam Search
A cat sitting inside of a suitcase.

Figure 1: We successfully caption images containing previously unseen objects by incorporating semantic attributes (i.e., image tags) during RNN decoding. Actual example from Section 4.2.

Although available image-caption training data is limited, many image collections are augmented with ground-truth text fragments such as semantic attributes (i.e., image tags) or object annotations. Even if these annotations do not exist, they can be generated using (potentially task specific) image taggers (Chen et al., 2013; Zhang et al., 2016) or object detectors (Ren et al., 2015; Krause et al., 2016), which are easier to scale to new concepts. In this paper our goal is to incorporate text fragments such as these during caption generation, to improve the quality of resulting captions. This goal poses two key challenges. First, RNNs are generally opaque, and difficult to influence at test time. Second, text fragments may include words

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 936–945
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

that are not present in the RNN vocabulary.

As illustrated in Figure 1, we address the first challenge (guidance) by using *constrained beam search* to guarantee the inclusion of selected words or phrases in the output of an RNN, while leaving the model free to determine the syntax and additional details. Constrained beam search is an approximate search algorithm capable of enforcing any constraints over resulting output sequences that can be expressed in a finite-state machine. With regard to the second challenge (vocabulary), empirically we demonstrate that an RNN can successfully generalize from similar words if both the input and output layers are fixed with pretrained word embeddings and then expanded as required.

To evaluate our approach, we use a held-out version of the MSCOCO dataset. Leveraging image tag predictions from an existing model (Hendricks et al., 2016) as constraints, we demonstrate state of the art performance for out-of-domain image captioning, while simultaneously improving the performance of the base model on in-domain data. Perhaps surprisingly, our results significantly outperform approaches that incorporate the same tag predictions into the learning algorithm (Hendricks et al., 2016; Venugopalan et al., 2016). Furthermore, we attempt the extremely challenging task of captioning the ImageNet classification dataset (Russakovsky et al., 2015). Human evaluations indicate that by leveraging ground truth image labels as constraints, the proportion of captions meeting or exceeding human quality increases from 11% to 22%. To facilitate future research we release our code and data from the project page[1].

2 Related Work

While various approaches to image caption generation have been considered, a large body of recent work is dedicated to neural network approaches (Donahue et al., 2015; Mao et al., 2015; Karpathy and Fei-Fei, 2015; Vinyals et al., 2015; Devlin et al., 2015). These approaches typically use a pretrained Convolutional Neural Network (CNN) image encoder, combined with a Recurrent Neural Network (RNN) decoder trained to predict the next output word, conditioned on previous words and the image. In each case the decoding process remains the same—captions are generated by searching over output sequences greedily or with beam search.

Recently, several works have proposed models intended to describe images containing objects for which no caption training data exists (out-of-domain captioning). The Deep Compositional Captioner (DCC) (Hendricks et al., 2016) uses a CNN image tagger to predict words that are relevant to an image, combined with an RNN language model to estimate probabilities over word sequences. The tagger and language models are pretrained separately, then fine-tuned jointly using the available image-caption data.

Building on the DCC approach, the Novel Object Captioner (NOC) (Venugopalan et al., 2016) is contemporary work with ours that also uses pretrained word embeddings in both the input and output layers of the language model. Another recent work (Tran et al., 2016) combines specialized celebrity and landmark detectors into a captioning system. More generally, the effectiveness of incorporating semantic attributes (i.e., image tags) into caption model training for in-domain data has been established by several works (Fang et al., 2015; Wu et al., 2016; Elliot and de Vries, 2015).

Overall, our work differs fundamentally from these approaches as we do not attempt to introduce semantic attributes, image tags or other text fragments into the learning algorithm. Instead, we incorporate text fragments during model decoding. To the best of our knowledge we are the first to consider this more loosely-coupled approach to out-of-domain image captioning, which allows the model to take advantage of information not available at training time, and avoids the need to retrain the captioning model if the source of text fragments is changed.

More broadly, the problem of generating high probability output sequences using finite-state machinery has been previously explored in the context of poetry generation using RNNs (Ghazvininejad et al., 2016) and machine translation using n-gram language models (Allauzen et al., 2014).

3 Approach

In this section we describe the constrained beam search algorithm, the base captioning model used in experiments, and our approach to expanding the model vocabulary with pretrained word embeddings.

[1] www.panderson.me/constrained-beam-search

3.1 Constrained Beam Search

Beam search (Koehn, 2010) is an approximate search algorithm that is widely used to decode output sequences from Recurrent Neural Networks (RNNs). We briefly describe the RNN decoding problem, before introducing constrained beam search, a multiple-beam search algorithm that enforces constraints in the sequence generation process.

Let $\boldsymbol{y}_t = (y_1, ..., y_t)$ denote an output sequence of length t containing words or other tokens from vocabulary V. Given an RNN modeling a probability distribution over such sequences, the RNN decoding problem is to find the output sequence with the maximum log-probability, where the log probability of any partial sequence $\boldsymbol{y}_t$ is typically given by $\sum_{j=1}^{t} \log p(y_j \mid y_1, ..., y_{j-1})$.

As it is computationally infeasible to solve this problem, beam search finds an approximate solution by maintaining a beam B_t containing only the b most likely partial sequences at each decoding time step t, where b is known as the beam size. At each time step t, the beam B_t is updated by retaining the b most likely sequences in the candidate set E_t generated by considering all possible next word extensions:

$$E_t = \big\{(\boldsymbol{y}_{t-1}, w) \mid \boldsymbol{y}_{t-1} \in B_{t-1}, w \in V\big\} \qquad (1)$$

To decode output sequences under constraints, a naive approach might impose the constraints on sequences produced at the end of beam search. However, if the constraints are non-trivial (i.e. only satisfied by relatively low probability output sequences) it is likely that an infeasibly large beam would be required in order to produce sequences that satisfy the constraints. Alternatively, imposing the constraints on partial sequences generated by Equation 1 is also unacceptable, as this would require that constraints be satisfied at every step during decoding—which may be impossible.

To fix ideas, suppose that we wish to generate sequences containing at least one word from each constraint set C1 = {‘chair’, ‘chairs’} and C2 = {‘desk’, ‘table’}. Note that it is possible to *recognize* sequences satisfying these constraints using the finite-state machine (FSM) illustrated in Figure 2, with start state s_0 and accepting state s_3. More generally, any set of constraints that can be represented with a regular expression can also be expressed as an FSM (either deterministic or non-deterministic) that recognizes sequences satisfying those constraints (Sipser, 2012).

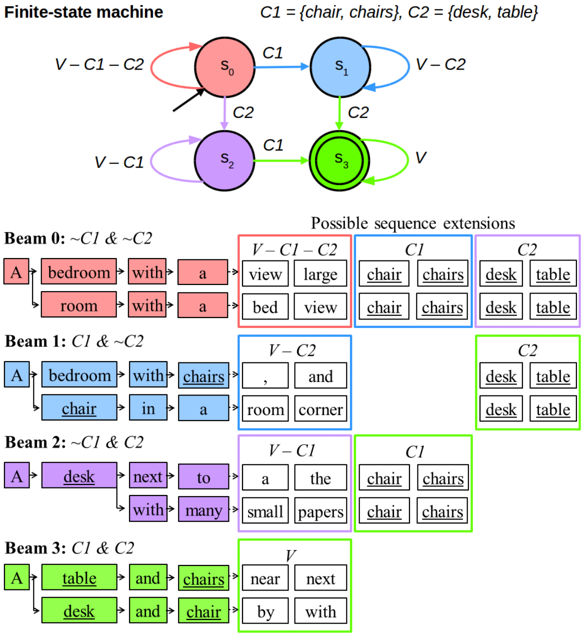


Figure 2: Example of constrained beam search decoding. Each output sequence must include the words ‘chair’ or ‘chairs’, and ‘desk’ or ‘table’ from vocabulary V. A finite-state machine (FSM) that recognizes valid sequences is illustrated at top. Each state in the FSM corresponds to a beam in the search algorithm (bottom). FSM state transitions determine the destination beam for each possible sequence extension. Valid sequences are found in Beam 3, corresponding to FSM accepting state s_3.

Since RNN output sequences are generated from left-to-right, to *generate* constrained sequences, we take an FSM that recognizes sequences satisfying the required constraints, and use the following multiple-beam decoding algorithm. For each state $s \in S$ in the FSM, a corresponding search beam B^s is maintained. As in beam search, each B^s is a set containing at most b output sequences, where b is the beam size. At each time step, each beam B^s_t is updated by retaining the b most likely sequences in its candidate set E^s_t given by:

$$E^s_t = \bigcup_{s' \in S} \big\{(\boldsymbol{y}_{t-1}, w) \mid \boldsymbol{y}_{t-1} \in B^{s'}_{t-1}, w \in V, \\ \delta(s', w) = s\big\} \qquad (2)$$

where $\delta : S \times V \mapsto S$ is the FSM state-transition function that maps states and words to states. As specified by Equation 2, the FSM state-transition function determines the appropriate candidate set for each possible extension of a partial sequence. This ensures that sequences in accepting states must satisfy all constraints as they have been recognized by the FSM during the decoding process.

Initialization is performed by inserting an empty sequence into the beam associated with the start state s_0, so $B_0^0 := \{\epsilon\}$ and $B_0^{i \neq 0} := \emptyset$. The algorithm terminates when an accepting state contains a completed sequence (e.g., containing an end marker) with higher log probability than all incomplete sequences. In the example contained in Figure 2, on termination captions in Beam 0 will not contain any words from C1 or C2, captions in Beam 1 will contain a word from C1 but not C2, captions in Beam 2 will contain a word from C2 but not C1, and captions in Beam 3 will contain a word from both C1 and C2.

3.1.1 Implementation Details

In our experiments we use two types of constraints. The first type of constraint consists of a conjunction of disjunctions $C = D_1, ..., D_m$, where each $D_i = w_{i,1}, ..., w_{i,n_i}$ and $w_{i,j} \in V$. Similarly to the example in Figure 2, a partial caption $\boldsymbol{y}_t$ satisfies constraint C iff for each $D_i \in C$, there exists a $w_{i,j} \in D_i$ such that $w_{i,j} \in \boldsymbol{y}_t$. This type of constraint is used for the experiments in Section 4.2, in order to allow the captioning model freedom to choose word forms. For each image tag, disjunctive sets are formed by using WordNet (Fellbaum, 1998) to map the tag to the set of words in V that share the same lemma.

The use of WordNet lemmas adds minimal complexity to the algorithm, as the number of FSM states, and hence the number of search beams, is not increased by adding disjunctions. Nevertheless, we note that the algorithm maintains one beam for each of the 2^m subsets of disjunctive constraints D_i. In practice $m \leq 4$ is sufficient for the captioning task, and with these values our GPU constrained beam search implementation based on Caffe (Jia et al., 2014) generates 40k captions for MSCOCO in well under an hour.

The second type of constraint consists of a subsequence that must appear in the generated caption. This type of constraint is necessary for the experiments in Section 4.3, because WordNet synsets often contain phrases containing multiple words. In this case, the number of FSM states, and the number of search beams, is linear in the length of the subsequence (the number of states is equal to number of words in a phrase plus one).

3.2 Captioning Model

Our approach to out-of-domain image captioning could be applied to any existing CNN-RNN captioning model that can be decoding using beam search, e.g., (Donahue et al., 2015; Mao et al., 2015; Karpathy and Fei-Fei, 2015; Vinyals et al., 2015; Devlin et al., 2015). However, for empirical evaluation we use the Long-term Recurrent Convolutional Network (Donahue et al., 2015) (LRCN) as our base model. The LRCN consists of a CNN visual feature extractor followed by two LSTM layers (Hochreiter and Schmidhuber, 1997), each with 1,000 hidden units. The model is factored such that the bottom LSTM layer receives only language input, consisting of the embedded previous word. At test time the previous word is the predicted model output, but during training the ground-truth preceding word is used. The top LSTM layer receives the output of the bottom LSTM layer, as well as a per-timestep static copy of the CNN features extracted from the input image.

The feed-forward operation and hidden state update of each LSTM layer in this model can be summarized as follows. Assuming N hidden units within each LSTM layer, the N-dimensional input gate i_t, forget gate f_t, output gate o_t, and input modulation gate g_t at timestep t are updated as:

$$i_t = \text{sigm}\left(W_{xi}x_t + W_{hi}h_{t-1} + b_i\right) \tag{3}$$

$$f_t = \text{sigm}\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right) \tag{4}$$

$$o_t = \text{sigm}\left(W_{xo}x_t + W_{ho}h_{t-1} + b_o\right) \tag{5}$$

$$g_t = \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \tag{6}$$

where $x_t \in \mathbb{R}^K$ is the input vector, $h_t \in \mathbb{R}^N$ is the LSTM output, W's and b's are learned weights and biases, and $\text{sigm}(\cdot)$ and $\tanh(\cdot)$ are the sigmoid and hyperbolic tangent functions, respectively, applied element-wise. The above gates control the memory cell activation vector $c_t \in \mathbb{R}^N$ and output $h_t \in \mathbb{R}^N$ of the LSTM as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{7}$$

$$h_t = o_t \odot \tanh\left(c_t\right) \tag{8}$$

where $\odot$ represents element-wise multiplication.

Using superscripts to represent the LSTM layer index, the input vector for the bottom LSTM is an encoding of the previous word, given by:

$$x_t^1 = W_e \Pi_t \tag{9}$$

where W_e is a word embedding matrix, and Π_t is a one-hot column vector identifying the input word at timestep t. The top LSTM input vector comprises the concatenated output of the bottom LSTM and the CNN feature descriptor of the image I, given by:

$$x_t^2 = (h_t^1, \mathrm{CNN}_\theta(I)) \tag{10}$$

For the CNN component of the model, we evaluate using the 16-layer VGG (Simonyan and Zisserman, 2015) model and the 50-layer Residual Net (He et al., 2016), pretrained on ILSVRC-2012 (Russakovsky et al., 2015) in both cases. Unlike Donahue et. al. (2015), we do not fix the CNN weights during initial training, as we find that performance improves if all training is conducted end-to-end. In training, we use only very basic data augmentation. All images are resized to 256 × 256 pixels and the model is trained on random 224 × 224 crops and horizontal flips using stochastic gradient descent (SGD) with hand-tuned learning rates.

3.3 Vocabulary Expansion

In the out-of-domain scenario, text fragments used as constraints may contain words that are not actually present in the captioning model's vocabulary. To tackle this issue, we leverage pretrained word embeddings, specifically the 300 dimension GloVe (Pennington et al., 2014) embeddings trained on 42B tokens of external text corpora. These embeddings are introduced at both the word input and word output layers of the captioning model and fixed throughout training. Concretely, the ith column of the W_e input embedding matrix is initialized with the GloVe vector associated with vocabulary word i. This entails reducing the dimension of the original LRCN input embedding from 1,000 to 300. The model output is then:

$$v_t = \tanh\left(W_v h_t^2 + b_v\right) \tag{11}$$

$$p(y_t \mid y_{t-1}, ..., y_1, I) = \mathrm{softmax}\left(W_e^T v_t\right) \tag{12}$$

where v_t represents the top LSTM output projected to 300 dimensions, W_e^T contains GloVe embeddings as row vectors, and $p(y_t \mid y_{t-1}, ..., y_1, I)$ represents the normalized probability distribution over the predicted output word y_t at timestep t, given the previous output words and the image. The model is trained with the conventional softmax cross-entropy loss function, and learns to predict v_t vectors that have a high dot-product similarity with the GloVe embedding of the correct output word.

Given these modifications — which could be applied to other similar captioning models — the process of expanding the model's vocabulary at test time is straightforward. To introduce an additional vocabulary word, the GloVe embedding for the new word is simply concatenated with W_e as an additional column, increasing the dimension of both Π_t and p_t by one. In total there are 1.9M words in our selected GloVe embedding, which for practical purposes represents an open vocabulary. Since GloVe embeddings capture semantic and syntactic similarities (Pennington et al., 2014), intuitively the captioning model will generalize from similar words in order to understand how the new word can be used.

4 Experiments

4.1 Microsoft COCO Dataset

The MSCOCO 2014 captions dataset (Lin et al., 2014) contains 123,293 images, split into a 82,783 image training set and a 40,504 image validation set. Each image is labeled with five human-annotated captions.

In our experiments we follow standard practice and perform only minimal text pre-processing, converting all sentences to lower case and tokenizing on white space. It is common practice to filter vocabulary words that occur less than five times in the training set. However, since our model does not learn word embeddings, vocabulary filtering is not necessary. Avoiding filtering increases our vocabulary from around 8,800 words to 21,689, allowing the model to potentially extract a useful training signal even from rare words and spelling mistakes (which are generally close to the correctly spelled word in embedding space). In all experiments we use a beam size of 5, and we also enforce the constraint that a single word cannot be predicted twice in a row.

4.2 Out-of-Domain Image Captioning

To evaluate the ability of our approach to perform out-of-domain image captioning, we replicate an existing experimental design (Hendricks et al., 2016) using MSCOCO. Following this ap-

		Out-of-Domain Test Data				In-Domain Test Data		
Model	CNN	SPICE	METEOR	CIDEr	F1	SPICE	METEOR	CIDEr
DCC (Hendricks et al., 2016)	VGG-16	13.4	21.0	59.1	39.8	15.9	23.0	77.2
NOC (Venugopalan et al., 2016)	VGG-16	-	21.4	-	49.1	-	-	-
Base	VGG-16	12.4	20.4	57.7	0	17.6	24.9	93.0
Base+T1	VGG-16	13.6	21.7	68.9	27.2	17.9	**25.0**	**93.4**
Base+T2	VGG-16	14.8	22.6	75.4	38.7	**18.2**	**25.0**	92.8
Base+T3	VGG-16	15.5	23.0	77.5	48.4	**18.2**	24.8	90.4
Base+T4	VGG-16	**15.9**	**23.3**	**77.9**	**54.0**	18.0	24.5	86.3
Base+T3*	VGG-16	18.7	27.1	119.6	54.5	22.0	29.4	135.5
Base All Data	VGG-16	17.8	25.2	93.8	59.4	17.4	24.5	91.7
Base	ResNet-50	12.6	20.5	56.8	0	18.2	24.9	93.2
Base+T1	ResNet-50	14.2	21.7	68.1	27.3	18.5	25.2	**94.6**
Base+T2	ResNet-50	15.3	22.7	74.7	38.5	**18.7**	**25.3**	94.1
Base+T3	ResNet-50	16.0	23.3	**77.8**	48.2	**18.7**	25.2	92.3
Base+T4	ResNet-50	**16.4**	**23.6**	77.6	**53.3**	18.4	24.9	88.0
Base+T3*	ResNet-50	19.2	27.3	117.9	54.5	22.3	29.4	133.7
Base All Data	ResNet-50	18.6	26.0	96.9	60.0	18.0	25.0	93.8

Table 1: Evaluation of captions generated using constrained beam search with 1 – 4 predicted image tags used as constraints (Base+T1 – 4). Our approach significantly outperforms both the DCC and NOC models, despite reusing the image tag predictions of the DCC model. Importantly, performance on in-domain data is not degraded but can also improve.

Model	bottle	bus	couch	microwave	pizza	racket	suitcase	zebra	Avg
DCC (Hendricks et al., 2016)	4.6	29.8	45.9	28.1	64.6	52.2	13.2	79.9	39.8
NOC (Venugopalan et al., 2016)	**17.8**	**68.8**	25.6	24.7	69.3	**68.1**	39.9	**89.0**	49.1
Base+T4	16.3	67.8	**48.2**	**29.7**	**77.2**	57.1	**49.9**	85.7	**54.0**

Table 2: F1 scores for mentions of objects not seen during caption training. Our approach (Base+T4) reuses the top 4 image tag predictions from the DCC model but generates higher F1 scores by interpreting tag predictions as constraints. All results based on use of the VGG-16 CNN.

proach, all images with captions that mention one of eight selected objects (or their synonyms) are excluded from the image caption training set. This reduces the size of the caption training set from 82,783 images to 70,194 images. However, the complete caption training set is tokenized as a bag of words per image, and made available as image tag training data. As such, the selected objects are unseen in the image caption training data, but not the image tag training data. The excluded objects, selected by Hendricks et. al. (2016) from the 80 main object categories in MSCOCO, are: 'bottle', 'bus', 'couch', 'microwave', 'pizza', 'racket', 'suitcase' and 'zebra'.

For validation and testing on this task, we use the same splits as in prior work (Hendricks et al., 2016; Venugopalan et al., 2016), with half of the original MSCOCO validation set used for validation, and half for testing. We use the validation set to determine hyperparameters and for early-stopping, and report all results on the test set. For evaluation the test set is split into in-domain and out-of-domain subsets, with the out-of-domain designation given to any test image that contains a mention of an excluded object in at least one reference caption.

To evaluate generated caption quality, we use the SPICE (Anderson et al., 2016) metric, which has been shown to correlate well with human judgment on the MSCOCO dataset, as well as the METEOR (Denkowski and Lavie, 2014) and CIDEr (Vedantam et al., 2015) metrics. For consistency with previously reported results, scores on out-of-domain test data are macro-averaged across the eight excluded object classes. To improve the comparability of CIDEr scores, the inverse document frequency statistics used by this metric are determined across the entire test set,

Base: A woman is playing tennis on a tennis court. **Tags:** tennis, player, ball, racket. **Base+T4:** A tennis player swinging a racket at a ball.

Base: A man standing next to a yellow train. **Tags:** bus, yellow, next, street. **Base+T4:** A man standing next to a yellow bus on the street.

Base: A close up of a cow on a dirt ground. **Tags:** zebra, zoo, enclosure, standing. **Base+T4:** A zebra standing in front of a zoo enclosure.

Base: A dog is sitting in front of a tv. **Tags:** dog, head, television, cat. **Base+T4:** A dog with a cat on its head watching television.

Base: A group of people playing a game of tennis. **Tags:** pink, tennis, crowd, ball. **Base+T4:** A crowd of people standing around a pink tennis ball.

Figure 3: Examples of out-of-domain captions generated on MSCOCO using the base model (Base), and the base model constrained to include four predicted image tags (Base+T4). Words never seen in training captions are underlined. The bottom row contains some failure cases.

rather than within subsets. On out-of-domain test data, we also report the F1 metric for mentions of excluded objects. To calculate the F1 metric, the model is considered to have predicted condition positive if the generated caption contains at least one mention of the excluded object, and negative otherwise. The ground truth is considered to be positive for an image if the excluded object in question is mentioned in any of the reference captions, and negative otherwise.

As illustrated in Table 1, on the out-of-domain test data, our base model trained only with image captions (Base) receives an F1 score of 0, as it is incapable of mentioned objects that do not appear in the training captions. In terms of SPICE, METEOR and CIDEr scores, our base model performs slightly worse than the DCC model on out-of-domain data, but significantly better on in-domain data. This may suggest that the DCC model achieves improvements in out-of-domain performance at the expense of in-domain scores (in-domain scores for the NOC model were not available at the time of submission).

Results marked with '+' in Table 1 indicate that our base model has been decoded with constraints in the form of predicted image tags. However, for the fairest comparison, and because re-using existing image taggers at test time is one of the motivations for this work, we did not train an image tagger from scratch. Instead, in results T1–4 we use the top 1–4 tag predictions respectively from the VGG-16 CNN-based image tagger used in the DCC model. This model was trained by the authors to predict 471 MSCOCO visual concepts including adjectives, verbs and nouns. Examples of generated captions, including failure cases, are presented in Figure 3.

As indicated in Table 1, using similar model capacity, the constrained beam search approach with predicted tags significantly outperforms prior work in terms SPICE, METEOR and CIDEr scores, across both out-of-domain and in-domain test data, utilizing varying numbers of tag predictions. Overall these results suggest that, perhaps surprisingly, it may be better to incorporate image tags into captioning models during decoding rather than during training. It also appears that, while introducing image tags improves performance on both out-of-domain and in-domain evaluations, it is beneficial to introduce more tag constraints when the test data is likely to contain previously unseen objects. This reflects the trading-off of influence between the image tags and the captioning model. For example, we noted that when using two tag constraints, 36% of generated captions were identical to the base model, but when using four tags this proportion dropped to only 3%.

To establish performance upper bounds, we train the base model on the complete MSCOCO training set (Base All Data). We also evaluate captions generated using our approach combined with an 'oracle' image tagger consisting of the top 3 ground-truth image tags (T3*). These were determined by selecting the 3 most frequently mentioned words in the reference captions for each test image (after eliminating stop words). The very high scores recorded for this approach may motivate the use of more powerful image taggers in future work. Finally, replacing VGG-16 with the more powerful ResNet-50 (He et al., 2016) CNN leads to modest improvements as indicated in the lower half of Table 1.

Evaluating F1 scores for object mentions (see Table 2), we note that while our approach outperforms prior work when four image tags are used, a significant increase in this score should not be expected as the underlying image tagger is the same.

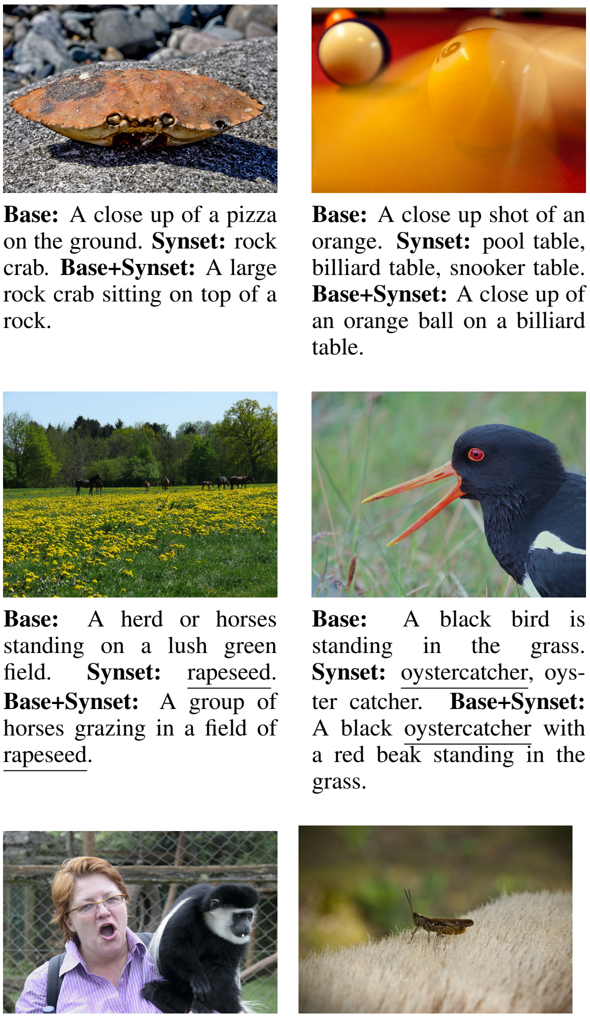
4.3 Captioning ImageNet

Consistent with our observation that many image collections contain useful annotations, and that we should seek to use this information, in this section we caption a 5,000 image subset of the ImageNet (Russakovsky et al., 2015) ILSVRC 2012 classification dataset for assessment. The dataset contains 1.2M images classified into 1,000 object categories, from which we randomly select five images from each category.

For this task we use the ResNet-50 (He et al., 2016) CNN, and train the base model on a combined training set containing 155k images comprised of the MSCOCO (Chen et al., 2015) training and validation datasets, and the full Flickr 30k (Young et al., 2014) captions dataset. We use constrained beam search and vocabulary expansion to ensure that each generated caption includes a phrase from the WordNet (Fellbaum, 1998) synset representing the ground-truth image category. For synsets that contain multiple entries, we run constrained beam search separately for each phrase and select the predicted caption with the highest log probability overall.

Note that even with the use of ground-truth object labels, the ImageNet captioning task remains extremely challenging as ImageNet contains a wide variety of classes, many of which are not evenly remotely represented in the available image-caption training datasets. Nevertheless, the injection of the ground-truth label frequently improves the overall structure of the caption over the base model in multiple ways. Examples of generated captions, including failure cases, are presented in Figure 4.

As the ImageNet dataset contains no existing caption annotations, following the human-evaluation protocol established for the MSCOCO 2015 Captioning Challenge (Chen et al., 2015), we used Amazon Mechanical Turk (AMT) to collect a human-generated caption for each sample image.



Base: A close up of a pizza on the ground. **Synset:** rock crab. **Base+Synset:** A large rock crab sitting on top of a rock.

Base: A close up shot of an orange. **Synset:** pool table, billiard table, snooker table. **Base+Synset:** A close up of an orange ball on a billiard table.

Base: A herd or horses standing on a lush green field. **Synset:** rapeseed. **Base+Synset:** A group of horses grazing in a field of rapeseed.

Base: A black bird is standing in the grass. **Synset:** oystercatcher, oyster catcher. **Base+Synset:** A black oystercatcher with a red beak standing in the grass.

Base: A man and a woman standing next to each other. **Synset:** colobus, colobus monkey. **Base+Synset:** Two colobus standing next to each other near a fence.

Base: A bird standing on top of a grass covered field. **Synset:** cricket. **Base+Synset:** A bird standing on top of a cricket field.

Figure 4: Examples of ImageNet captions generated by the base model (Base), and by the base model constrained to include the ground-truth synset (Base+Synset). Words never seen in the MSCOCO / Flickr 30k caption training set are underlined. The bottom row contains some failure cases.

For each of the 5,000 samples images, three human evaluators were then asked to compare the caption generated using our approach with the human-generated caption (Base+Syn v. Human). Using a smaller sample of 1,000 images, we also collected evaluations comparing our approach to the base model (Base+Syn v. Base), and comparing the base model with human-generated captions (Base v. Human). We used only US-based AMT

	Better	Equally Good	Equally Poor	Worse
Base v. Human	0.05	0.06	0.04	0.86
Base+Syn v. Human	0.12	0.10	0.05	0.73
Base+Syn v. Base	0.39	0.06	0.42	0.13

Table 3: In human evaluations our approach leveraging ground-truth synset labels (Base+Syn) improves significantly over the base model (Base) in both direct comparison and in comparison to human-generated captions.

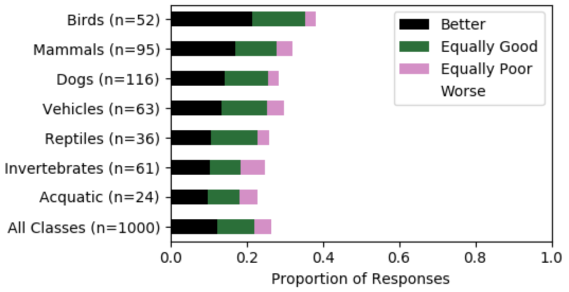


Figure 5: AMT evaluations of generated (Base+Syn) ImageNet captions versus human captions, by super-category.

workers, screened according to their performance on previous tasks. For both tasks, the user interface and question phrasing was identical to the MSCOCO collection process. The results of these evaluations are summarized in Table 3.

Overall, Base+Syn captions were judged to be equally good or better than human-generated captions in 22% of pairwise evaluations (12% 'better', 10% 'equally good'), and equally poor or worse than human-generated captions in the remaining 78% of evaluations. Although still a long way from human performance, this is a significant improvement over the base model with only 11% of captions judged to be equally good or better than human. For context, using the identical evaluation protocol, the top scoring model in the MSCOCO Captioning Challenge (evaluating on in-domain data) received 11% 'better', and 17% 'equally good' evaluations.

To better understand performance across synsets, in Figure 5 we cluster some class labels into super-categories using the WordNet hierarchy, noting particularly strong performances in super-categories that have some representation in the caption training data — such as birds, mammals and dogs. These promising results suggest that fine-grained object labels can be successfully integrated with a general purpose captioning model using our approach.

5 Conclusion and Future Research

We investigate *constrained beam search*, an approximate search algorithm capable of enforcing any constraints over resulting output sequences that can be expressed in a finite-state machine. Applying this approach to out-of-domain image captioning on a held-out MSCOCO dataset, we leverage image tag predictions to achieve state of the art results. We also show that we can significantly improve the quality of generated ImageNet captions by using the ground-truth labels.

In future work we hope to use more powerful image taggers, and to consider the use of constrained beam search within an expectation-maximization (EM) algorithm for learning better captioning models from weakly supervised data.

Acknowledgements

We thank the anonymous reviewers for providing insightful comments and for helping to identify relevant prior literature. This research is supported by an Australian Government Research Training Program (RTP) Scholarship and by the Australian Research Council Centre of Excellence for Robotic Vision (project number CE140100016).

References

Cyril Allauzen, Bill Byrne, Adrià de Gispert, Gonzalo Iglesias, and Michael Riley. 2014. Pushdown automata in statistical machine translation. *Computational Linguistics* 40(3):687–723.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: Semantic propositional image caption evaluation. In *ECCV*.

Minmin Chen, Alice X Zheng, and Kilian Q Weinberger. 2013. Fast image tagging. In *ICML*.

Xinlei Chen, Tsung-Yi Lin Hao Fang, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325* .

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. 2015. Language models for image captioning: The quirks and what works. In *ACL*.

Jeffrey Donahue, Lisa A. Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*.

Desmond Elliot and Arjen P. de Vries. 2015. Describing images using inferred visual dependency representations. In *ACL*.

Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh Srivastava, Li Deng, Piotr Dollar, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2015. From captions to visual concepts and back. In *CVPR*.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating topical poetry. In *EMNLP*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Lisa Anne Hendricks, Subhashini Venugopalan, Marcus Rohrbach, Raymond Mooney, Kate Saenko, and Trevor Darrell. 2016. Deep compositional captioning: Describing novel object categories without paired training data. In *CVPR*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8).

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093* .

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Jonathan Krause, Benjamin Sapp, Andrew Howard, Howard Zhou, Alexander Toshev, Tom Duerig, James Philbin, and Li Fei-Fei. 2016. The unreasonable effectiveness of noisy data for fine-grained recognition. In *ECCV*.

T.Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. 2014. Microsoft COCO: Common objects in context. In *ECCV*.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L. Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-RNN). In *ICLR*.

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

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)* 115(3):211–252.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Michael Sipser. 2012. *Introduction to the Theory of Computation*. Cengage Learning, 3rd edition.

Kenneth Tran, Xiaodong He, Lei Zhang, Jian Sun, Cornelia Carapcea, Chris Thrasher, Chris Buehler, and Chris Sienkiewicz. 2016. Rich image captioning in the wild. In *CVPR Workshop*.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *CVPR*.

Subhashini Venugopalan, Lisa Anne Hendricks, Marcus Rohrbach, Raymond J. Mooney, Trevor Darrell, and Kate Saenko. 2016. Captioning images with diverse objects. *arXiv preprint arXiv:1606.07770* .

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *CVPR*.

Q. Wu, C. Shen, L. Liu, A. Dick, and A. van den Hengel. 2016. What Value Do Explicit High Level Concepts Have in Vision to Language Problems? In *CVPR*.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL* .

Yang Zhang, Boqing Gong, and Mubarak Shah. 2016. Fast zero-shot image tagging. In *CVPR*.

Zero-Shot Activity Recognition with Verb Attribute Induction

Rowan Zellers and **Yejin Choi**
Paul G. Allen School of Computer Science & Engineering
University of Washington
Seattle, WA 98195, USA
{rowanz,yejin}@cs.washington.edu

Abstract

In this paper, we investigate large-scale zero-shot activity recognition by modeling the visual and linguistic attributes of action verbs. For example, the verb "salute" has several properties, such as being a light movement, a social act, and short in duration. We use these attributes as the internal mapping between visual and textual representations to reason about a previously unseen action. In contrast to much prior work that assumes access to gold standard attributes for zero-shot classes and focuses primarily on object attributes, our model uniquely learns to infer action attributes from dictionary definitions and distributed word representations. Experimental results confirm that action attributes inferred from language can provide a predictive signal for zero-shot prediction of previously unseen activities.

1 Introduction

We study the problem of inferring action verb attributes based on how the word is defined and used in context. For example, given a verb such as "swig" shown in Figure 1, we want to infer various properties of actions such as *motion dynamics* (moderate movement), *social dynamics* (solitary act), *body parts* involved (face, arms, hands), and *duration* (less than 1 minute) that are generally true for the range of actions that can be denoted by the verb "swig".

Our ultimate goal is to improve *zero-shot learning* of activities in computer vision: predicting a previously unseen activity by integrating background knowledge about the conceptual properties of actions. For example, a computer vision system may have seen images of "drink" activities during training, but not "swig". Ideally, the system should infer the likely visual characteristics of "swig" using world knowledge implicitly available in dictionary definitions and word embeddings.

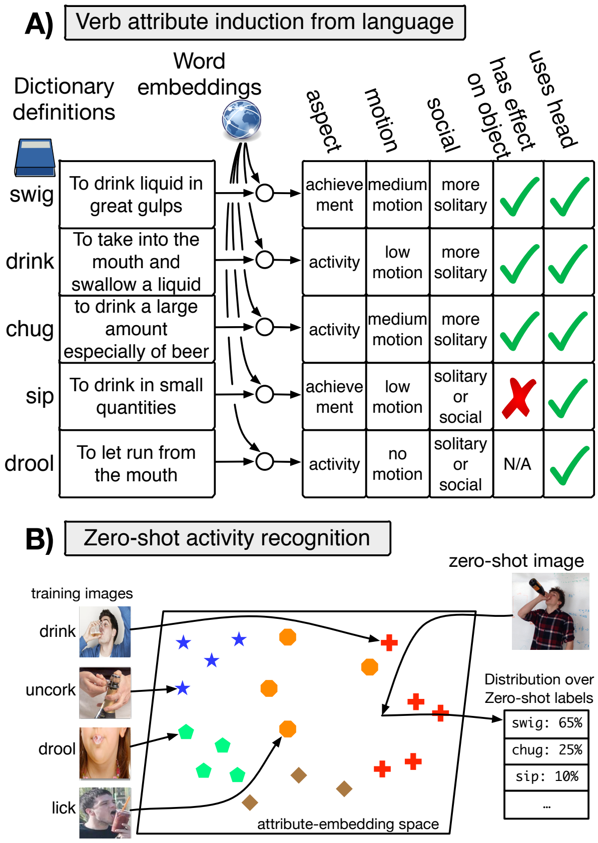


Figure 1: An overview of our task. Our goal is twofold. **A:** we seek to use use distributed word embeddings in tandem with dictionary definitions to obtain a high level understanding of verbs. **B:** we seek to use these predicted attributes to allow a classifier to recognize a broader set of activities than what was seen in training time.

However, most existing literature on zero-shot learning has focused on object recognition with only a few notable exceptions (see Related Work

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 946–958
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

in Section 8). There are two critical reasons: *object attributes*, such as color, shape, and texture, are conceptually straightforward to enumerate. In addition, they have distinct visual patterns which are robust for current vision systems to recognize. In contrast, *activity attributes* are more difficult to conceptualize as they involve varying levels of abstractness, which are also more challenging for computer vision as they have less distinct visual patterns. Noting this difficulty, Antol et al. (2014) instead employ cartoon illustrations as intermediate mappings for zero-shot dyadic activity recognition. We present a complementary approach: that of tackling the abstractness of verb attributes directly. We develop and use a corpus of verb attributes, using linguistic theories on verb semantics (e.g., aspectual verb classes of Vendler (1957)) and also drawing inspiration from studies on linguistic categorization of verbs and their properties (Friedrich and Palmer, 2014; Siegel and McKeown, 2000).

In sum, we present the first study aiming to recover general action attributes for a diverse collection of verbs, and probe their predictive power for zero-shot activity recognition on the recently introduced imSitu dataset (Yatskar et al., 2016). Empirical results show that action attributes inferred from language can help classifying previously unseen activities and suggest several avenues for future research on this challenging task. We publicly share our dataset and code for future research.[1]

2 Action Verb Attributes

We consider seven different groups of action verb attributes. They are motivated in part by potential relevance for visual zero-shot inference, and in part by classical literature on linguistic theories on verb semantics. The attribute groups are summarized below.[2] Each attribute group consists of a set of attributes, which sums to $K = 24$ distinct attributes annotated over the verbs.[3]

[1] Aspectual Classes We include the aspectual verb classes of Vendler (1957):

- *state*: a verb that does not describe a changing situation (e.g. “have”, “be”)
- *achievement*: a verb that can be *completed* in a short period of time (e.g. “open”, “jump”)
- *accomplishment*: a verb with a sense of completion over a longer period of time (e.g. “climb”)
- *activity*: a verb without a clear sense of completion (e.g. “swim”, “walk”, “talk”)

[2] Temporal Duration This attribute group relates to the aspectual classes above, but provides additional estimation of typical time duration with four categories. We categorize verbs by best-matching temporal units: seconds, minutes, hours, or days, with an additional option for verbs with unclear duration (e.g., “provide”).

[3] Motion Dynamics This attribute group focuses on the energy level of motion dynamics in four categories: no motion (“sleep”), low motion (“smile”), medium (“walk”), or high (“run”). We add an additional option for verbs whose motion level depends highly on context, such as ‘finish.’

[4] Social Dynamics This attribute group focuses on the likely social dynamics, in particular, whether the action is usually performed as a solitary act, a social act, or either. This is graded on a 5-part scale from least social (-2) to either $(+0)$ to most social $(+2)$

[5] Transitivity This attribute group focuses on whether the verb can take an object, or be used without. This gives the model a sense of the implied action dynamics of the verb between the agent and the world. We record three variables: whether or not the verb is naturally transitive on a person (“I hug her” is natural), on a thing (“I eat it”), and whether the verb is intransitive (“I run”).

[6] Effects on Arguments This attribute group focuses on the effects of actions on agents and other arguments. For each of the possible transitivities of the verb, we annotate whether or not it involves *location change* (“travel”), *world change* (“spill”), *agent or object change* (“cry”) , or *no visible change* (“ponder”).

[7] Body Involvements This attribute group specifies prominent body parts involved in carrying out the action. For example, “open” typically involves “hands” and “arms” when used in a physical sense. We use five categories: head, arms, torso, legs, and other body parts.

[1] Available at http://github.com/rowanz/verb-attributes

[2] The full list is available in the supplemental section.

[3] Several of our attributes are categorical; if converted to binary attributes, we would have 40 attributes in total.

Action Attributes and Contextual Variations In general, contextual variations of action attributes are common, especially for frequently used verbs that describe everyday physical activities. For example, while "open" typically involves "hands", there are exceptions, e.g. "open one's eyes". In this work, we focus on stereotypical or prominent characteristics across a range of actions that can be denoted using the same verb. Thus, three investigation points of our work include: (1) crowd-sourcing experiments to estimate the distribution of human judgments on the prominent characteristics of everyday physical action verbs, (2) the feasibility of learning models for inferring the prominent characteristics of the everyday action verbs despite the potential noise in the human annotation, and (3) their predictive power in zero-shot action recognition despite the potential noise from contextual variations of action attributes. As we will see in Section 7, our study confirms the usefulness of studying action attributes and motivates the future study in this direction.

Relevance to Linguistic Theories The key idea in our work that action verbs project certain expectations about their influence on their arguments, their pre- and post-conditions, and their implications on social dynamics, etc., relates to the original Frame theories of Baker et al. (1998a). The study of action verb attributes are also closely related to formal studies on verb categorization based on the characteristics of the actions or states that a verb typically associates to (Levin, 1993), and cognitive linguistics literature that focus on causal structure and force dynamics of verb meanings (Croft, 2012).

3 Learning Verb Attributes from Language

In this section we present our models for learning verb attributes from language. We consider two complementary types of language-based input: dictionary definitions and word embeddings. The approach based on dictionary definitions resembles how people acquire the meaning of a new word from a dictionary lookup, while the approach based on word embeddings resembles how people acquire the meaning of words in context.

Overview This task follows the standard supervised learning approach where the goal is to predict K attributes per word in the vocabulary $\mathcal{V}$. Let $x_v \in \mathcal{X}$ represent the input representation of a word $v \in \mathcal{V}$. For instance, x_v could denote a word embedding, or a definition looked up from a dictionary (modeled as a list of tokens). Our goal is to produce a model $f : \mathcal{X} \to \mathbb{R}^d$ that maps the input to a representation of dimension d. Modeling options include using pretrained word embeddings, as in Section 3.1, or using a sequential model to encode a dictionary, as in Section 3.2.

Then, the estimated probability distribution over attribute k is given by:

$$\hat{y}_{v,k} = \sigma(\mathbf{W}^{(k)} f(x_v)). \qquad (1)$$

If the attribute is binary, then $\mathbf{W}^{(k)}$ is a vector of dimension d and σ is the sigmoid function. Otherwise, $\mathbf{W}^{(k)}$ is of shape $d_k \times d$, where d_k is the dimension of attribute k, and σ is the softmax function. Let the vocabulary $\mathcal{V}$ be partitioned into sets $\mathcal{V}_{train}$ and $\mathcal{V}_{test}$; then, we train by minimizing the cross-entropy loss over $\mathcal{V}_{train}$ and report attribute-level accuracy over words in $\mathcal{V}_{test}$.

Connection to Learning Object Attributes This problem has been studied before for zero-shot object recognition, but there are several key differences. Al-Halah et al. (2016) build the 'Class-Attribute Association Prediction' model (CAAP) that classifies the attributes of an object class from its name. They apply it on the Animals with Attributes dataset, a dataset containing 50 animal classes, each described by 85 attributes (Lampert et al., 2014). Importantly, these attributes are concrete details with semantically meaningful names such as "has horns" and "is furry". The CAAP model takes advantage of this, consisting of a tensor factorization model initialized by the word embeddings of the object class names as well as the attribute names. On the other hand, *verb attributes* such as the ones we outline in Section 2, are technical linguistic terms. Since word embeddings principally capture common word senses, they are unsuited for verb attributes. Thus, we evaluate two versions of CAAP as a baseline: one where the model is preinitialized with GloVe embeddings (Pennington et al., 2014) for the attribute names (CAAP-pretrained), and one where the model is learned from random initialization (CAAP-learned).

3.1 Learning from Distributed Embeddings

One way of producing attributes is from distributed word embeddings such as word2vec

(Mikolov et al., 2013). Intuitively, we expect similar verbs to have similar distributions of nearby nouns and adverbs, which can greatly help us in zero-shot prediction. In our experiments, we use 300-dimensional GloVe vectors trained on 840B tokens of web data (Pennington et al., 2014). We use logistic regression to predict each attribute, as we found that extra hidden layers did not improve performance. Thus, we let $f^{emb}(x_v) = \mathbf{w}_v$, the GloVe embedding of v, and use Equation 1 to get the distribution over labels.

We additionally experiment with retrofitted embeddings, in which embeddings are mapped in accordance with a lexical resource. Following the approach of Faruqui et al. (2015), we retrofit embeddings using WordNet (Miller, 1995), Paraphrase-DB (Ganitkevitch et al., 2013), and FrameNet (Baker et al., 1998b).

3.2 Learning from Dictionary Definitions

We additionally propose a model that learns the attribute-grounded meaning of verbs through dictionary definitions. This is similar in spirit to the task of using a dictionary to predict word embeddings (Hill et al., 2016).

BGRU encoder Our first model involves a Bidirectional Gated Recurrent Unit (BGRU) encoder (Cho et al., 2014). Let $x_{v,1:T}$ be a definition for verb v, with T tokens. To encode the input, we pass it through the GRU equation:

$$\vec{\mathbf{h}}_t = \text{GRU}(x_{v,t}, \vec{\mathbf{h}}_{t-1}). \qquad (2)$$

Let $\overleftarrow{\mathbf{h}}_1$ denote the output of a GRU applied on the reversed input $x_{v,T:1}$. Then, the BGRU encoder is the concatenation $f^{bgru} = \vec{\mathbf{h}}_T \| \overleftarrow{\mathbf{h}}_1$.

Bag-of-words encoder Additionally, we try two common flavors of a Bag-of-Words model. In the standard case, we first construct a vocabulary of 5000 words by frequency on the dictionary definitions. Then, $f^{bow}(x_v)$ represents the one-hot encoding $f^{bow}(x_v)_i = [i \in x_v]$, in other words, whether word i appears in definiton x_v for verb v.

Additionally, we try out a Neural Bag-of-Words model where the word embeddings in a definition are averaged (Iyyer et al., 2015). This is $f^{nbow}(x_{v,1:T}) = \frac{1}{|T|}\sum_{t=1}^{T} f^{emb}(x_{v,t})$.

Dealing with multiple definitions per verb One potential pitfall with using dictionary definitions is that there are often many defnitions associated with each verb. This creates a dataset bias since polysemic verbs are seen more often. Additionally, dictionary definitions tend to be sorted by relevance, thus lowering the quality of the data if all definitions are weighted equally during training. To counteract this, we randomly oversample the definitions at training time so that each verb has the same number of definitions.[4] At test time, we use the first-occurring (and thus generally most relevant) definition per verb.

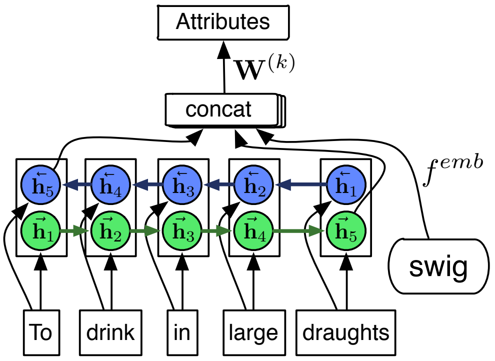


Figure 2: Overview of our combined dictionary + embedding to attribute model. Our encoding is the concatenation of a Bidirectional GRU of a definition and the word embedding for that word. The encoding is then mapped to the space of attributes using parameters $\mathbf{W}^{(k)}$.

3.3 Combining Dictionary and Embedding representations

We hypothesize that the two modalities of the dictionary and distributional embeddings are complementary. Therefore, we propose an early fusion (concatenation) of both categories. Figure 2 describes the GRU + embedding model – in other words, $f^{BGRU+emb} = f^{BGRU} \| f^{emb}$. This can likewise be done with any choice of definition encoder and word embedding.

4 Zero-Shot Activity Recognition

4.1 Verb Attributes as Latent Mappings

Given learned attributes for a collection of activities, we would like to evaluate their performance at describing these activities from real world images in a zero-shot setting. Thus, we consider several models that classify an image's label by pivoting through an attribute representation.

[4] For the (neural) bag of words models, we also tried concatenating the definitions together per verb and then doing the encoding. However, we found that this gave worse results.

Overview A formal description of the task is at follows. Let the space of labels be $\mathcal{V}$, partitioned into $\mathcal{V}_{train}$ and $\mathcal{V}_{test}$. Let $z_v \in \mathcal{Z}$ represent an image with label $v \in \mathcal{V}$; our goal is to correctly predict this label amongst verbs $v \in \mathcal{V}_{test}$ at test time, despite never seeing any images with labels in $\mathcal{V}_{test}$ during training.

Generalization will be done through a lookup table $\mathbf{A}$, with known attributes for each $v \in \mathcal{V}$. Formally, for each attribute k we define it as:

$$\mathbf{A}^{(k)}_{v',i} = \begin{cases} 1 & \text{if attribute } k \text{ for verb } v' \text{ is } i \\ -1 & \text{otherwise} \end{cases} \tag{3}$$

For binary attributes, we need only one entry per verb, making $\mathbf{A}^{(k)}$ a single column vector. Let our image encoder be represented by the map $g : \mathcal{Z} \to \mathbb{R}^d$. We then use the linear map in Equation 1 to produce the log-probability distribution over each attribute k. The distribution over labels is then:

$$P(\cdot|z_v) = \underset{v'}{\text{softmax}} \left(\sum_k \mathbf{A}^{(k)} \mathbf{W}^{(k)} g(z_v) \right) \tag{4}$$

where $\mathbf{W}^{(k)}$ is a learned parameter that maps the image encoder to the attribute representation. We then train our model by minimizing the cross-entropy loss over the training verbs $\mathcal{V}_{train}$.

Convolutional Neural Network (CNN) Encoder Our image encoder is a CNN with the Resnet-152 architecture (He et al., 2016). We use weights pretrained on ImageNet (Deng et al., 2009) and perform additional pretraining on ImSitu using the classes $\mathcal{V}_{train}$. After this, we remove the top layer and set $g(x_v)$ to be the 2048-dimensional image representation from the network.

4.1.1 Connection to other attribute models

Our model is similar to those of Akata et al. (2013) and Romera-Paredes and Torr (2015) in that we predict the attributes indirectly and train the model through the class labels.[5] It differs from several other zeroshot models, such as Lampert et al. (2014)'s Direct Attribute Prediction (DAP) model, in that DAP is trained by maximizing the probability of predicting each attribute and then multiplies the probabilities at test time. Our use of joint training allows the recognition model to directly optimize class-discrimination rather than attribute-level accuracy.

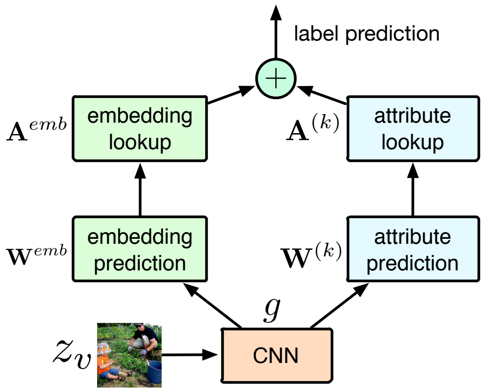


Figure 3: Our proposed model for combining attribute-based zero-shot learning and word-embedding based transfer learning. The embedding and attribute lookup layers are used to predict a distribution of labels over $\mathcal{V}_{train}$ during training and $\mathcal{V}_{test}$ during testing.

4.2 Verb Embeddings as Latent Mappings

An additional method of doing zero-shot image classification is by using word embeddings directly. Frome et al. (2013) build DeVISE, a model for zero-shot learning on ImageNet object recognition where the objective is for the image model to predict a class's word embedding directly. DeVISE is trained by minimizing

$$\sum_{v' \in \mathcal{V}_{train} \setminus \{v\}} \max\{0, .1 + (\mathbf{w}_{v'} - \mathbf{w}_v)\mathbf{W}^{emb} g(z_v)\}$$

for each image z_v. We compare against a version of this model with fixed GloVe embeddings $\mathbf{w}$.

Additionally, we employ a variant of our model using only word embeddings. The equation is the same as Equation 4, except using the matrix $\mathbf{A}^{emb}$ as a matrix of word embeddings: i.e., for each label v we consider, we have $\mathbf{A}^{emb}_v = \mathbf{w}_v$.

4.3 Joint prediction from Attributes and Embeddings

To combine the representation power of the attribute and embedding models, we build an ensemble combining both models. This is done by adding the logits before the softmax is applied:

$$\underset{v'}{\text{softmax}} \left(\sum_k \mathbf{A}^{(k)} \mathbf{W}^{(k)} g(z_v) + \mathbf{A}^{emb} \mathbf{W}^{emb} g(z_v) \right)$$

A diagram is shown in Figure 3. We find that during optimization, this model can easily overfit, presumably by excessive coadaption of the embedding and attribute components. To solve this,

[5] Unlike these models, however, we utilize (some) categorical attributes and optimize using cross-entropy.

we train the model to minimize the cross entropy of three sources independently: the attributes only, the embeddings only, and the sum, weighting each equally.

Incorporating predicted and gold attributes We additionally experiment with an ensemble of our model, combining predicted and gold attributes of $\mathcal{V}_{test}$. This allows the model to hedge against cases where a verb attribute might have several possible correct answers. A single model is trained; at test time, we multiply the class level probabilities $P(\cdot|z_v)$ of each together to get the final predictions.

5 Actions and Attributes Dataset

To evaluate our hypotheses on action attributes and zero-shot learning, we constructed a dataset using crowd-sourcing experiments. The *Actions and Attributes* dataset consists of annotations for 1710 verb templates, each consisting of a verb and an optional particle (e.g. “put” or “put *up*”).

We selected all verbs from the ImSitu corpus, consisting of images representing verbs from many categories (Yatskar et al., 2016), then extended the set using the MPII movie visual description dataset and ScriptBase datasets, (Rohrbach et al., 2015; Gorinski and Lapata, 2015). We used the spaCy dependency parser (Honnibal and Johnson, 2015) to extract the verb template for each sentence, and collected annotations on Mechanical Turk to filter out nonliteral and abstract verbs. Turkers annotated this filtered set of templates using the attributes described in Section 2. In total, 1203 distinct verbs are included. The templates are split randomly by verb; out of 1710 total templates, we save 1313 for training, 81 for validation, and 316 for testing.

To provide signal for classifying these verbs, we collected dictionary definitions for each verb using the Wordnik API,[6] including only senses that are explicitly labeled “verb.” This leaves us with 23,636 definitions, an average of 13.8 per verb.

6 Experimental Setup

BGRU pretaining We pretrain the BGRU model on the Dictionary Challenge, a collection of 800,000 word-definition pairs obtained from Wordnik and Wikipedia articles (Hill et al., 2016); the objective is to obtain a word’s embedding given one of its definitions. For the BGRU model, we use an internal dimension of 300, and embed the words to a size 300 representation. The vocabulary size is set to 30,000 (including all verbs for which we have definitions). During pretraining, we keep the architecture the same, except a different 300-dimensional final layer is used to predict the GloVe embeddings.

Following Hill et al. (2016), we use a ranking loss. Let $\hat{\mathbf{w}} = \mathbf{W}^{emb} f(x)$ be the predicted word embeddings for each definition x of a word in the dictionary (not necessarily a verb). Let $\mathbf{w}$ be the word’s embedding, and $\widetilde{\mathbf{w}}$ be the embedding of a random dictionary word. The loss is then given by:

$$L = \max\{0, .1 - \cos(\mathbf{w}, \hat{\mathbf{w}}) + \cos(\mathbf{w}, \widetilde{\mathbf{w}})\}$$

After pretraining the model on the Dictionary Challenge, we fine-tune the attribute weights $\mathbf{W}^{(k)}$ using the cross-entropy over Equation 1.

Zero-shot with the imSitu dataset We build our image-to-verb model on the newly introduced imSitu dataset, which contains a diverse collection of images depicting one of 504 verbs. The images represent a variety of different semantic role labels (Yatskar et al., 2016). Figure 4 shows examples from the dataset. We apply our attribute split to the dataset and are left with 379 training classes, 29 validation classes, and 96 test classes.

Zero-shot activity recognition baselines We compare against several additional baseline models for learning from attributes and embeddings. Romera-Paredes and Torr (2015) propose “Embarassingly Simple Zero-shot Learning” (ESZL), a linear model that directly predicts class labels through attributes and incorporates several types of regularization. We compare against a variant of Lampert et al. (2014)’s DAP model discussed in Section 4.1.1. We additionally compare against DeVISE (Frome et al., 2013), as mentioned in Section 4.2. We use a Resnet-152 CNN finetuned on the imSitu $\mathcal{V}_{train}$ classes as the visual features for these baselines (the same as discussed in Section 4.1).

Additional implementation details are provided in the Appendix.

[6] Available at http://developer.wordnik.com/ with access to American Heriatge Dictionary, the Century Dictionary, the GNU Collaborative International Dictionary, Wordnet, and Wiktionary.

		acc-macro	acc-micro	Body	Duration	Aspect	Motion	Social	Effect	Transi.
	most frequent class	61.33	75.45	76.84	76.58	43.67	35.13	42.41	84.97	69.73
Emb	CAAP-pretrained	64.96	78.06	84.81	72.15	50.95	46.84	43.67	85.21	71.10
	CAAP-learned	68.15	81.00	86.33	76.27	52.85	52.53	**45.57**	88.29	75.21
	GloVe	66.60	79.69	85.76	75.00	50.32	48.73	43.99	86.52	75.84
	GloVe + framenet	67.42	80.79	86.27	76.58	49.68	50.32	44.94	88.19	75.95
	GloVe + ppdb	67.52	80.75	85.89	76.58	51.27	50.95	43.99	88.21	75.74
	GloVe + wordnet	68.04	81.13	86.58	76.90	54.11	50.95	43.04	**88.34**	**76.37**
Dict	BGRU	66.05	79.44	85.70	76.90	51.27	48.42	40.51	86.92	72.68
	BoW	62.53	77.61	83.54	76.58	48.42	35.76	36.39	86.31	70.68
	NBoW	65.41	78.96	85.00	76.58	52.85	42.41	43.35	86.87	70.78
D+E	NBoW + GloVe	67.52	80.76	**86.84**	75.63	53.48	51.90	41.77	88.03	75.00
	BoW + GloVe	63.15	77.89	84.11	**77.22**	49.68	34.81	38.61	86.18	71.41
	BGRU + GloVe	**68.43**	**81.18**	86.52	76.58	**56.65**	**53.48**	41.14	88.24	**76.37**

Table 1: Results on the text-to-attributes task. All values reported are accuracies (in %). For attributes where multiple labels can be selected, the accuracy is averaged over all instances (e.g., the accuracy of "Body" is given by the average of accuracies from correctly predicting Head, Torso, etc.). As such, we report two ways of averaging the results: microaveraging (where the accuracy is the average of accuracies on the underlying labels) and macroaveraging (where the accuracy is averaged together from the groups).

7 Experimental Results

7.1 Predicting Action Attributes from Text

Our results for action attribute prediction from text are given in Table 1. Several examples are given in the supplemental section in Table 3. Our results on the text-to-attributes challenge confirm that it is a challenging task for two reasons. First, there is noise associated with the attributes: many verb attributes are hard to annotate given that verb meanings can change in context.[7] Second, there is a lack of training data inherent to the problem: there are not many common verbs in English, and it can be difficult to crowdsource annotations for rare ones. Third, any system must compete with strong frequency-based baselines, as attributes are generally sparse. Moreover, we suspect that were more attributes collected (so as to cover more obscure patterns), the sparsity would only increase.

Despite this, we report strong baseline results on this problem, particularly with our embedding based models. The performance gap between embedding- and definition-based models can possibly be explained by the fact that the word embeddings are trained on a very large corpus of real-world *examples* of the verb, while the definition is only a single high-level representation meant to be understood by someone who already speaks that language. For instance, it is likely difficult for the definition-based model to infer whether a verb is transitive or not (Transi.), since definitions might assume commonsense knowledge about the underlying concepts the verb represents. The strong performance of embedding models is further enhanced by using retrofitted word embeddings, suggesting an avenue for improvement on language grounding through better representation of linguistic corpora.

We additionally see that both joint dictionary-embedding models outperform the dictionary-only models overall. In particular, the BGRU+GloVe model performs especially well at determining the aspect and motion attributes of verbs, particularly relative to the baseline. The strong performance of the BGRU+GloVe model indicates that there is some signal that is missing from the distributional embeddings that can be recovered from the dictionary definition. We thus use the predictions of this model for zero-shot image recognition.

Based on error analysis, we found that one common mode of failure is where contextual knowledge is required. To give an example, the embedding based model labels "shop" as a likely solitary action. This is possibly because there are a lack of similar verbs in $\mathcal{V}_{train}$; by random chance, "buy" is also in the test set. We see that this can be partially mitigated by the dictionary, as evidenced by the fact that the dictionary-based models label "shop" as in between social and solitary. Still, it is a difficult task to infer that people like to "visit stores in search of merchandise" together.

7.2 Zero-shot Action Recognition

Our results for verb prediction from images are given in Table 2. Despite the difficulty of predicting the correct label over 96 unseen choices,

[7] As such, our attributes have a median Krippendorff Alpha of $\alpha = .359$.

	Attributes used			$v \in \mathcal{V}_{test}$	
Model	atts(P)	atts(G)	GloVe	top-1	top-5
Random				1.04	5.20
DeVISE			✓	16.50	37.56
ESZL		✓		3.60	14.81
	✓			3.27	13.21
DAP		✓		3.35	16.69
	✓			4.33	17.56
Ours		✓		4.79	19.98
	✓			7.04	22.19
	✓	✓		7.71	24.90
			✓	17.60	39.29
		✓	✓	18.10	41.46
	✓		✓	16.75	40.44
	✓	✓	✓	**18.15**	**42.17**

Table 2: Results on the image-to-verb task. atts(P) refers to attributes predicted from the BGRU+GloVe model described in Section 3, atts(G) to gold attributes, and GloVe to GloVe vectors. The accuracies reported are amongst the 96 unseen labels of $\mathcal{V}_{test}$.

our models show predictive power. Although our attribute models do not outperform our embedding models and DeVISE alone, we note that our joint attribute and embedding model scores the best overall, reaching 18.10% in top-1 and 41.46% in top-5 accuracy when using gold attribute annotations for the zero-shot verbs. This result is possibly surprising given the small number of attributes ($K = 24$) in total, of which most tend to be sparse (as can be seen from the baseline performance in Table 1). We thus hypothesize that collecting more activity attributes would further improve performance.

We also note the success in performing zero-shot learning with predicted attributes. Perhaps paradoxically, our attribute-only models (along with DAP) perform better in both accuracy metrics when given predicted attributes at test time, as opposed to gold attributes. Further, we get an extra boost by ensembling predictions of our model when given two sets of attributes at test time, giving us the best results overall at 18.15% top-1 accuracy and 42.17% top-5. Interestingly, better performance with predicted attributes is also reported by Al-Halah et al. (2016): predicting the attributes with their CAAP model and then running the DAP model on these predicted attributes outperforms the use of gold attributes at test time. It is somewhat unclear why this is the case - possibly, there is some bias in the attribute labeling, which the attribute predictor can correct for.

In addition to quantitative results, we show some zero-shot examples in Figure 4. The examples show inherent difficulty of zero-shot action recognition. Incorrect predictions are often reasonably related to the situation ("rub" vs "dye") but picking the correct target verb based on attribute-based inference is still a challenging task.

Although our results appear promising, we argue that our model still fails to represent much of the semantic information about each image class. In particular, our model is prone to *hubness*: the overprediction of a limited set of labels at test time: those that closely match signatures of examples in the training set. This problem has previously been observed with the use of word embeddings for zero-shot learning (Marco and Georgiana, 2015) and can be seen in our examples (for instance, the over-prediction of "buy"). Unfortunately, we were unable to mitigate this problem in a way that also led to better quantitative results (for instance, by using a ranking loss as in DeVISE (Frome et al., 2013)). We thus leave resolving the hubness problem in zero-shot activity recognition as a question for future work.

8 Related Work

Learning attributes from embeddings Rubinstein et al. (2015) seek to predict McRae et al. (2005)'s feature norms from word embeddings of concrete nouns. Likewise, the CAAP model of Al-Halah et al. (2016) predicts the object attributes of concrete nouns for use in zero-shot learning. In contrast, we predict verb attributes. A related task is that of improving word embeddings using multimodal data and linguistic resources (Faruqui et al., 2015; Silberer et al., 2013; Vendrov et al., 2016). Our work runs orthogonal to this, as we focus on word attributes as a tool for a zero-shot activity recognition pipeline.

Zero-shot learning with objects Though distinct, our work is related to zero-shot learning of objects in computer vision. There are several datasets (Nilsback and Zisserman, 2008; Welinder et al., 2010) and models developed on this task (Romera-Paredes and Torr (2015); Lampert et al. (2014); Mukherjee and Hospedales (2016); Farhadi et al. (2010)). In addition,

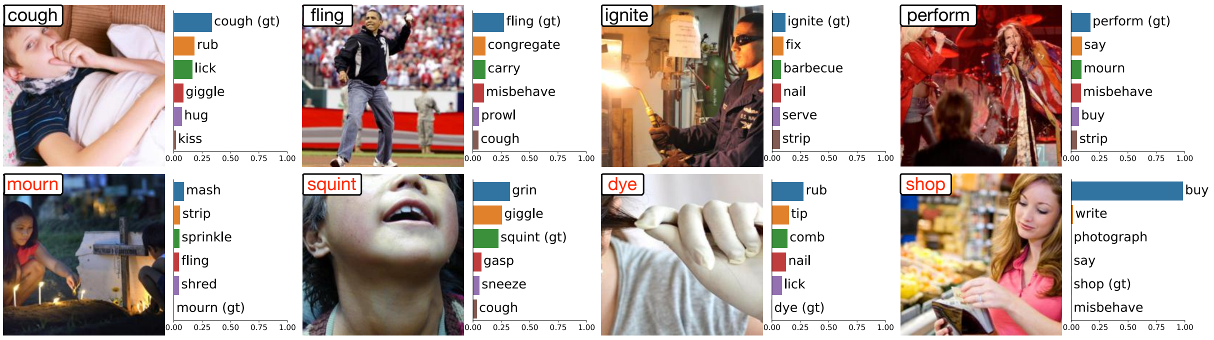


Figure 4: Predictions on unseen classes from our attribute+embedding model with gold attributes. The top and bottom rows show successful and failure cases respectively. The bars to the right of each image represent a probability distribution, showing the ground truth class and the top 5 scoring incorrect classes.

Ba et al. (2015) augment existing datasets with descriptive Wikipedia articles so as to learn novel objects from descriptive text. As illustrated in Section 1, action attributes pose unique challenges compared to object attributes, thus models developed for zero-shot object recognition are not as effective for zero-shot action recognition, as has been empirically shown in Section 7.

Zero-shot activity recognition In prior work, zero-shot activity recognition has been studied on video datasets, each containing a selection of concrete physical actions. The MIXED action dataset, itself a combination of three action recognition datasets, has 2910 labeled videos with 21 actions, each described by 34 action attributes (Liu et al., 2011). These action attributes are concrete binary attributes corresponding to low-level physical movements, for instance, “arm only motion,” “leg: up-forward motion.” By using word embeddings instead of attributes, Xu et al. (2017) study video activity recognition on a variety of action datasets, albeit in the transductive setting wherein access to the test labels is provided during training. In comparison with our work on imSitu (Yatskar et al., 2016), these video datasets lack broad coverage of verb-level classes (and for some, sufficient data points per class).

The abstractness of broad-coverage activity labels makes the problem much more difficult to study with attributes. To get around this, Antol et al. (2014) present a synthetic dataset of cartoon characters performing dyadic actions, and use these cartoon illustrations as internal mappings for zero-shot recognition of dyadic actions in real-world images. We investigate an alternative approach by using linguistically informed verb attributes for activity recognition.

9 Future work / Conclusion

Several possibilities remain open for future work. First, more attributes could be collected and evaluated, possibly integrating other linguistic theories about verbs, with more accurate modeling of context. Second, while our experiments use attributes as a pivot between language and vision domains, the effects of this could be explored more in future work. In particular, since our experiments show that unsupervised word embeddings significantly help performance, it might be desirable to learn data-driven attributes in an end-to-end fashion directly from a large corpus or from dictionary definitions. Third, future research on action attributes should ideally include videos to better capture attributes that require temporal signals.

Overall, however, our work presents a strong early step towards zero-shot activity recognition, a relatively less studied task that poses several unique challenges over zero-shot object recognition. We introduce new action attributes motivated by linguistic theories and demonstrate their empirical use for reasoning about previously unseen activities.

Acknowledgements We thank the anonymous reviewers, Mark Yatskar, Luke Zettlemoyer, and Yonatan Bisk, for their helpful feedback. We also thank the Mechanical Turk workers and members of the XLab, who helped with the annotation process. This work is supported by the National Science Foundataion Graduate Research Fellowship (DGE-1256082), the NSF grant (IIS-1524371), DARPA CwC program through ARO (W911NF-15-1-0543), and gifts by Google and Facebook.

References

Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. 2013. Label-embedding for attribute-based classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 819–826.

Ziad Al-Halah, Makarand Tapaswi, and Rainer Stiefelhagen. 2016. Recovering the missing link: Predicting class-attribute associations for unsupervised zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5975–5984.

Stanislaw Antol, C. Lawrence Zitnick, and Devi Parikh. 2014. Zero-Shot Learning via Visual Abstraction. In *ECCV*.

Jimmy Ba, Kevin Swersky, Sanja Fidler, and Ruslan Salakhutdinov. 2015. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *ICCV*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998a. The Berkeley FrameNet project. In *COLING-ACL '98: Proceedings of the Conference*, pages 86–90, Montreal, Canada.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998b. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

William Croft. 2012. *Verbs: Aspect and causal structure*. OUP Oxford. Pg. 16.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

Ali Farhadi, Ian Endres, and Derek Hoiem. 2010. Attribute-centric recognition for cross-category generalization. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2352–2359. IEEE.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting Word Vectors to Semantic Lexicons. Association for Computational Linguistics.

Annemarie Friedrich and Alexis Palmer. 2014. Automatic prediction of aspectual class of verbs in context. In *ACL (2)*, pages 517–523.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marco Aurelio Ranzato, and Tomas Mikolov. 2013. DeViSE: A Deep Visual-Semantic Embedding Model. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2121–2129. Curran Associates, Inc.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *HLT-NAACL*, pages 758–764.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1066–1076, Denver, Colorado. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to Understand Phrases by Embedding the Dictionary. *Transactions of the Association for Computational Linguistics*, 4(0):17–30.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Mohit Iyyer, Varun Manjunatha, Jordan L Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL (1)*, pages 1681–1691.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. 2014. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465.

Beth Levin. 1993. *English verb classes and alternations : a preliminary investigation*.

Jingen Liu, Benjamin Kuipers, and Silvio Savarese. 2011. Recognizing human actions by attributes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3337–3344. IEEE.

Angeliki Lazaridou Marco, Baroni and Dinu Georgiana. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. ACL.

Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behav Res Methods*, 37(4):547–559.

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

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Tanmoy Mukherjee and Timothy Hospedales. 2016. Gaussian visual-linguistic embedding for zero-shot recognition. EMNLP.

Maria-Elena Nilsback and Andrew Zisserman. 2008. Automated flower classification over a large number of classes. In *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*, pages 722–729. IEEE.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Anna Rohrbach, Marcus Rohrbach, Niket Tandon, and Bernt Schiele. 2015. A Dataset for Movie Description. *arXiv:1501.02530 [cs]*. ArXiv: 1501.02530.

Bernardino Romera-Paredes and Philip HS Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *ICML*, pages 2152–2161.

Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. 2015. How well do distributional models capture different types of semantic knowledge? In *Proceedings of the 53nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Eric V Siegel and Kathleen R McKeown. 2000. Learning methods to combine linguistic indicators: Improving aspectual classification and revealing linguistic insights. *Computational Linguistics*, 26(4):595–628.

Carina Silberer, Vittorio Ferrari, and Mirella Lapata. 2013. Models of semantic representation with visual attributes. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 572–582, Sofia, Bulgaria. Association for Computational Linguistics.

Zeno Vendler. 1957. Verbs and Times. *The Philosophical Review*, 66(2):143–160.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *ICLR*.

Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. 2010. Caltech-ucsd birds 200.

Xun Xu, Timothy Hospedales, and Shaogang Gong. 2017. Transductive zero-shot action recognition by word-vector embedding. *International Journal of Computer Vision*, pages 1–25.

Mark Yatskar, Luke Zettlemoyer, and Ali Farhadi. 2016. Situation recognition: Visual semantic role labeling for image understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5534–5542.

A Supplemental

Implementation details

Our CNN and BGRU models are built in PyTorch[8]. All of our one-layer neural network models are built in Scikit-learn (Pedregosa et al., 2011) using the provided LogisticRegression class (using one-versus-rest if appropriate). Our neural models use the Adam optimizer (Kingma and Ba, 2014), though we weak the default hyperparameters somewhat.

Recall that our dictionary definition model is a bidirectional GRU with a hidden size of 300, with a vocabulary size of 30,000. After pretraining on the Dictionary Challenge, we freeze the word embeddings and apply a dropout rate of 50% before the final hidden layer. We found that such an aggressive dropout rate was necessary due to the small size of the training set. During pretraining, we used a learning rate of 10^{-4}, a batch size of 64, and set the Adam parameter ϵ to the default $1e^{-8}$. During finetuning, we set $\epsilon = 1.0$ and the batch size to 32. In general, we found that setting too low of an ϵ during finetuning caused our zero-shot models to update parameters too aggressively during the first couple of updates, leading to poor results.

For our CNN models, we pretrained the Resnet 152 (initialized with imagenet weights) on the

[8] pytorch.org

training classes of the imSitu dataset, using a learning rate of 10^{-4} and $\epsilon = 10^{-8}$. During finetuning, we dropped the learning rate to 10^{-5} and set $\epsilon = 10^{-1}$. We also froze all parameters except for the final resnet block, and the linear attribute and embedding weights. We also found L2 regularization quite important in reducing overfitting, and we applied regularization at a weight of 10^{-4} to all trainable parameters.

Full list of attributes

The following is a full list of the attributes. In addition to the attributes presented here, we also crowdsourced attributes for the emotion content of each verb (e.g., happiness, sadness, anger, and surprise). However, we found these annotations to be skewed towards "no emotion", since most verbs do not strongly associate with a specific emotion. Thus, we omit them in our experiments.

(1) Aspectual Classes: one attribute with 5 values:

- (a) State
- (b) Achievement
- (c) Accomplishment
- (d) Activity
- (e) Unclear without context

(2) Temporal Duration: one attribute with 5 values:

- (a) Atemporal
- (b) On the order of seconds
- (c) On the order of hours
- (d) On the order of days

(3) Motion Dynamics: One attribute with 5 values:

- (a) unclear without context
- (b) No motion
- (c) Low motion
- (d) Medium motion
- (e) High motion

(4) Social Dynamics: One attribute with 5 values:

- (a) solitary
- (b) likely solitary
- (c) solitary or social
- (d) likely social
- (e) social

(5) Transitivity: Three binary attributes:

- (a) Intransitive: 1 if the verb can be used intransitively, 0 otherwise
- (b) Transitive (person): 1 if the verb can be used in the form "<someone>", 0 otherwise
- (c) Transitive (object): 1 if the verb can be used in the form "<verb> something", 0 otherwise

(6) Effects on Arguments: 12 binary attributes

- (a) Intransitive 1: 1 if the verb is intransitive and the subject moves somewhere
- (b) Intransitive 2: 1 if the verb is intransitive and the external world changes
- (c) Intransitive 3: 1 if the verb is intransitive, and the subject's state changes
- (d) Intransitive 4: 1 if the verb is intransitive, and nothing changes
- (e) Transitive (obj) 1: 1 if the verb is transitive for objects and the object moves somewhere
- (f) Transitive (obj) 2: 1 if the verb is transitive for objects and the external world changes
- (g) Transitive (obj) 3: 1 if the verb is transitive for objects and the object's state changes
- (h) Transitive (obj) 4: 1 if the verb is transitive for objects and nothing changes
- (i) Transitive (person) 1: 1 if the verb is transitive for people and the object is a person that moves somewhere
- (j) 'Transitive (person) 2: 1 if the verb is transitive for people and the external world changes
- (k) Transitive (person) 3: 1 if the verb is transitive for people and if the object is a person whose state changes
- (l) Transitive (person) 4: 1 if the verb is transitive for people and nothing changes

(7) Body Involements: 5 binary attributes

- (a) Arms: 1 if arms are used
- (b) Head: 1 if head is used
- (c) Legs: 1 if legs are used
- (d) Torso: 1 if torso is used
- (e) Other: 1 if another body part is used

	Model	Definition	Social	Aspect	Energy	Time	Body part
shop	GT	To visit stores in search of merchandise or bargains	likely social	accomplish.	high	hours	arms,head
	embed		likely solitary	activity	medium	minutes	
	BGRU		solitary or social	activity	medium	minutes	
	BGRU+		solitary or social	activity	medium	minutes	
mash	GT	To convert malt or grain into mash	likely solitary	activity	high	seconds	arms
	embed		likely solitary	activity	medium	seconds	arms
	BGRU		solitary or social	achievement	medium	seconds	arms
	BGRU+		likely solitary	activity	high	seconds	arms
photograph	GT	To take a photograph of	solitary or social	achievement	low	seconds	arms,head
	embed		solitary or social	accomplish.	medium	minutes	arms
	BGRU		solitary or social	achievement	medium	seconds	arms
	BGRU+		solitary or social	unclear	low	seconds	arms
spew out	GT	eject or send out in large quantities also metaphorical	solitary or social	achievement	high	seconds	head
	embed		likely solitary	achievement	medium	seconds	
	BGRU		solitary or social	achievement	high	seconds	arms
	BGRU+		likely solitary	achievement	medium	seconds	
tear	GT	To pull apart or into pieces by force rend	likely solitary	achievement	low	seconds	arms
	embed		solitary or social	achievement	medium	seconds	arms
	BGRU		solitary or social	achievement	high	seconds	arms
	BGRU+		solitary or social	achievement	high	seconds	arms
squint	GT	To look with the eyes partly closed as in bright sunlight	likely solitary	achievement	low	seconds	head
	embed		likely solitary	achievement	low	seconds	head
	BGRU		likely solitary	achievement	low	seconds	head
	BGRU+		likely solitary	achievement	low	seconds	head
shake	GT	To cause to move to and fro with jerky movements	solitary or social	activity	medium	seconds	
	embed		likely solitary	achievement	medium	seconds	arms
	BGRU		likely solitary	activity	medium	seconds	
	BGRU+		likely solitary	activity	medium	seconds	
doze	GT	To sleep lightly and intermittently	likely solitary	state	none	minutes	head
	embed		likely solitary	achievement	medium	seconds	
	BGRU		likely solitary	achievement	low	seconds	
	BGRU+		likely solitary	activity	low	seconds	
writhe	GT	To twist as in pain struggle or embarrassment	solitary or social	activity	high	seconds	arms,torso
	embed		likely solitary	activity	medium	seconds	
	BGRU		likely solitary	activity	medium	seconds	arms
	BGRU+		likely solitary	activity	medium	seconds	

Table 3: Example sentences and predicted attributes. Due to space constraints, we only list a few representative attributes and verbs.

Deriving continous grounded meaning representations from referentially structured multimodal contexts

Sina Zarrieß and **David Schlangen**
Dialogue Systems Group // CITEC // Faculty of Linguistics and Literary Studies
Bielefeld University, Germany
{sina.zarriess,david.schlangen}@uni-bielefeld.de

Abstract

Corpora of referring expressions paired with their visual referents are a good source for learning word meanings directly grounded in visual representations. Here, we explore additional ways of extracting from them word representations linked to multi-modal context: through expressions that refer to the *same* object, and through expressions that refer to *different* objects in the *same scene*. We show that continuous meaning representations derived from these contexts capture complementary aspects of similarity, even if not outperforming textual embeddings trained on very large amounts of raw text when tested on standard similarity benchmarks. We propose a new task for evaluating grounded meaning representations—detection of potentially co-referential phrases—and show that it requires precise denotational representations of attribute meanings, which our method provides.

1 Introduction

Various routes for linking language to extra-linguistic context have been explored in recent years. A lot of research has looked at integrating visual representations, either directly (Matuszek et al., 2012; Krishnamurthy and Kollar, 2013; Yu et al., 2016; Schlangen et al., 2016) or through mapping into a multi-modal distributional space (Feng and Lapata, 2010; Bruni et al., 2012; Kiela and Bottou, 2014; Lazaridou et al., 2015). Young et al. (2014) have explored a less direct link, by representing the extension of phrasal expressions as sets of images, and deriving from this a precise notion of denotational similarity. In very recent work, Cocos and Callison-Burch (2017) use spatial context from geo-located tweets to induce word embeddings that capture situational similarity between lexical items.

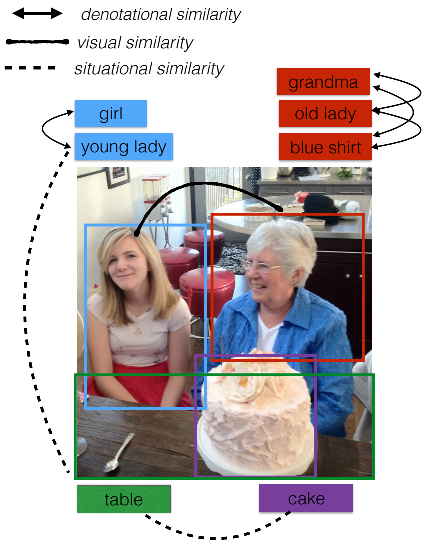


Figure 1: Dimensions of context in referential, visually grounded language, and similarity relations that can be derived from it, image from MsCoco (Lin et al., 2014))

In this paper, we explore an approach that combines aspects of several of these paths. Starting point is the observation that corpora of exophoric referring expressions provide richly structured contexts that go beyond just linking individual expressions with their denotations. As an example consider the scene in Figure 1 depicting several referents and corresponding referring expressions produced by different speakers. This scene provides a learner not only with an example of a referent for the word *lady*, it also provides the information that *lady* can co-refer with *girl*, and that its denotations can spatially / situationally co-occur with those of *table* and *cake*. From these types of information we infer word embeddings, following the method from Levy and Goldberg (2014) for training embeddings on arbi-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 959–965
Copenhagen, Denmark, September 7–11, 2017.

trary non-linear context, and we show that these capture complementary aspects of word similarity that purely textual induction methods conflate. We also show that these representations handle a more directly referential similarity task better.

2 Word Embeddings from Multi-Modal Referential Contexts

We base our study on the REFERIT and REFCOCO corpus (Kazemzadeh et al., 2014; Yu et al., 2016) building upon image collections by (Grubinger et al., 2006) and (Lin et al., 2014); for the latter, we also use referring expressions collected by Mao et al. (2015). This corpus gives us visual scenes containing sets of objects, $s = o_1, \ldots, o_n$. Each object is associated with a set of referring expressions $r_1, \ldots, r_m$; and we use a standard method (a ConvNet) for providing a visual representation vis_i for it. Each referring expression, in turn, is defined as a linear sequence of words $r_i = w_1 \ldots w_k$. In the following, we structure this context into four dimensions—visual, textual, situational and denotational—which we use to derive different word embeddings.

2.1 Textual Context (TXT)

We learn standard distributional word embeddings from our corpus, ignoring extra-linguistic context. We train a skip-gram model (Mikolov et al., 2013) with negative sampling with window width 5, 300 dimensions. For comparison, we also use the textual word embeddings provided by Baroni et al. (2014), trained on a much larger web corpus (5-word context window, 10 negative samples, 400 dimensions). We distinguish the two textual embeddings using the subscripts TXT$_{ref}$, TXT$_{web}$.

2.2 Visual Grounding (VIS)

Given a set of referring expressions containing the word w and their corresponding referent $(o_j, r_j), w \in r_j$, we can derive a visual context for the word w by averaging over the visual representations of its referents vis_j, as proposed for instance by Kiela and Bottou (2014). The visual context of a word can be seen as a 'visual prototype'. We derive representations of our visual inputs with a convolutional neural network, "GoogLeNet" (Szegedy et al., 2015), that was trained on data from the ImageNet corpus (Deng et al., 2009), and extract the final fully-connected layer before the classification layer, to give us a 1024 dimensional representation of the region. Following (Schlangen et al., 2016), we also add 7 features that encode information about the region relative to the image, the full representation hence is a vector of 1031 features. Each word is then represented as the average over its visual vectors.

2.3 Situational Grounding (SIT)

We also train word embeddings (dim. 300) that predict words paired with their situational context, following the method by Levy and Goldberg (2014). This captures similarities between words occurring for different objects in the same scene, e.g. *cake* in the context of *table* in Figure 1. Given a pair of referring expressions $(r_i, o_i), (r_j, o_j), o_i \neq o_j$, r_i and r_j are co-situational expressions. Thus, for a word $w_i \in r_i$, we consider all words $w_j \in r_j$ as its situational context. In practice, we compute situational contexts only for the head nouns of each referring expression, as we expect situational similarities to be useful for capturing similarities between nouns.

2.4 Denotational Grounding (DEN)

As our data typically records multiple co-referential expressions for an object (3 expressions on average in the REFCOCO data), we define the denotational context based on sets of expressions referring to the same object $(r_1, o_i) \ldots (r_n, o_i)$. For a word $w_i \in r_i$, we consider all words w_{j_l} (with $w_{j_l} \in r_j$) as denotational context, where r_j and r_i refer to the same object. When two words occur in a denotational context, we have strong evidence that they are semantically compatible, i.e. can refer to the same objects as *girl* and *lady* in Figure 1 do. Similar to our training procedure for situational embeddings, we now learn 300-dimensional word embeddings that predict occurrences of a word based on co-referential contexts, pairing each word with all words from referring expressions describing the same object.

3 Word Similarity and Relatedness

We now have four different continuous representations for words; in the following, we evaluate them for how well they predict semantic relations.

Similarity We evaluate on some similarity data sets, reporting Spearman ρ correlations between human ratings and cosine similarities for word vectors. We use the MEN (Bruni et al., 2012) and

Silberer and Lapata (2014)'s data with semantic (SemSim) and visual similarity (VisSim) ratings.

Compatibility As generic semantic similarity judgements are known to be "fuzzy" (Faruqui et al., 2016), we also evaluate on Kruszewski and Baroni (2015)'s benchmark on semantic compatibility. They define two words as being *semantically compatible* "if they can potentially refer to the same thing". We expect our denotational and visual embeddings to be highly useful for this task. We report unsupervised results obtained from cosine similarities between word embeddings.

Hypernym Directionality We adopt an evaluation procedure by Kiela et al. (2015b) on hypernym pairs in the BLESS data set (Baroni and Lenci, 2011). Given a general (e.g. 'animal') and a concrete noun (e.g. 'dog') that stand in the hypernym relation, the task is to identify the noun that is more general. Lazaridou et al. (2015) found that the generality or concreteness of a noun's meaning is reflected in the entropy of its embedding, and we adopt that measure for our purposes. Thus, we compute entropies of our word embeddings and report accuracies corresponding to the proportion of noun pairs where the entropy of the more general noun is higher than the more concrete noun.

Vocabulary We intersect the vocabularies covered by the different embeddings, which amounts to 1960 words in total. We restrict evaluation to the corresponding word pairs in the above data sets, coverage is reported in Table 1.

Results As shown in Table 1, the performance of embeddings learned on referring expression corpora are generally below state-of-the-art distributional vectors trained on large web corpora. However, some interesting tendencies can be observed by comparing embeddings learned from different context dimensions. Denotational embeddings in isolation provide a precise representation of meaning that outperforms the other types of embeddings on semantic similarity judgements in MEN and SemSim, and detects hypernym directionality most accurately. An interesting exception is the compatibility data set where visual embeddings clearly outperform textual and denotational embeddings. Situational embeddings perform less well than textual and denotational embeddings but, interestingly, are similar in performance to visual embeddings on semantic similarity, suggesting that visual and situational similarity seem to be equally important aspects of general semantic similarity. Concatenation of denotational and textual embeddings yields the best results for correlations with human similarity judgements. This is expected as denotational similarity is probably too restricted for generic semantic similarity. We experimented with further embedding combinations, but only the fusion of the textual and denotational dimension outperformed the embeddings obtained from a particular grounding dimension.

Model	MEN	SemSim	VisSim	Compat.	Hyp.Dir.
# pairs	*989*	*2041*	*2041*	*4843*	*334*
VIS	0.404	0.469	0.427	**0.241**	78.14
TXT$_{ref}$	0.550	0.584	0.484	0.230	55.69
DEN	0.646	0.583	0.491	0.163	**81.14**
SIT	0.470	0.468	0.371	0.134	59.58
DEN‖TXT$_{ref}$	**0.654**	**0.632**	**0.531**	0.207	79.94
TXT$_{web}$	0.799	0.708	0.578	0.262	90.42

Table 1: Word similarity and relatedness evaluation

Model	TXT	DEN	SIT	VIS
TXT	1	0.60	0.45	0.30
DEN	0.60	1	0.45	0.35
SIT	0.45	0.35	1	0.26
VIS	0.30	0.35	0.26	1

Table 2: Model correlations

Table 2 shows correlations on cosine similarities on all word pairs from MEN, SemSim, VisSim and Compatibility between our word embeddings. This further corroborates the finding that different dimensions of grounding lead to complementary notions of similarity. In particular, correlation between visual and situational embeddings is relatively low, as compared to more fuzzy textual embeddings which correlate well with denotational embeddings. For a qualitative analysis, more examples are shown in Appendix A.

Qualitative Discussion Table 3 illustrates similarities learned from different grounding dimensions by means of some qualitative examples. Whereas denotational and visual embeddings rank semantically compatible words on top (e.g. *grass-grassy*), situational embeddings clearly focus more on topical similarity (*grass-clouds*). Given these examples, the finding that visual embeddings outperform denotational embeddings on the semantic compatibility task (see Table 1) seems rather contradictory. A preliminary error analysis suggests that the compatibility ratings that humans provide 'out of context' in a rating task differ

woman	txt$_{ref}$	lady, girl, man, chick
	den	lady, girl, women, blouse
	sit	girl, guy, man, lady
	vis	lady, girl, women, chick
sidewalk	txt$_{ref}$	pavement, ground, walkway, steps
	den	street, sidewlak, walkway, pavement
	sit	buildin, bldg, lamppost, street
	vis	pavement, street, walkway, concrete
grass	txt$_{ref}$	shrubs, dirt, bushes, sand
	den	grassy, patch, bounded, plains
	sit	clouds, church, trees, building
	vis	grassy, path, shrubs, bushes
couch	txt$_{ref}$	sofa, chair, bench, bed
	den	sofa, pillows, cushions, loveseat
	sit	sofa, leather, armchair, seater
	vis	sofa, pillow, pillows, love

Table 3: Top nearest neighbours for some example noun embeddings

to some extent from referential choices in our corpus. As an example, in the compatibility data set, the words *pigeon* and *mother* are rated as being equally similar to *animal*. However, in our corpus of referring expressions, *mother* is never used to refer to animal entities and our denotational embeddings predict them to be highly dissimilar, whereas visual embeddings are slightly more robust in this case.

More generally, textual embeddings learned from referring expressions captures a much more fuzzy and generic notion of similarity than denotational, visual or situational embeddings, e.g. *grass* is similar to *shrubs* and to *sand* in the textual space. This fuzziness has been found for word embeddings trained on large amounts of raw text as well (Faruqui et al., 2016).

4 Approximate Co-Reference Detection

Another important testbed for models of lexical meaning is their ability to capture semantic inference, with textual entailment as a well-known paradigm: here the task is to predict whether a textual hypothesis h can be inferred from a given premise p (Dagan et al., 2006). Young et al. (2014) have proposed a less strict variant of this called "approximate textual entailment". The main idea is that premise and hypothesis candidates can be automatically extracted from a corpus of captioned images. Given a set of captions known to describe the same image and an hypothesis, the task is to determine whether the hypothesis can describe the same image as the premise.

Inspired by this approach, we use the multimodal corpus of referring expressions to set up a new task for evaluating word embeddings, which consists of capturing approximate inferential relations between referring expressions. Thus, in our case, the hypothesis and the premise are expressions referring to objects, and the task is to determine whether they could (potentially) refer to the same object. Note that this is also similar to the notion of semantic compatibility proposed by Kruszewski and Baroni (2015), but extended to phrasal expressions. We can automatically extract positive and negative pairs from the data (see Section 2) by looking at pairs of expressions referring to objects in the same image and distinguishing **coreferential expressions** referring to the same entity (e.g. *grandma* - *old lady*), and **non-coreferential expressions** referring to different entities, e.g. *old lady* - *young lady*. In contrast to the majority of existing similarity and relatedness benchmarks which are centered around nouns, this task requires precise meaning representations for attribute-like words (e.g. *left-right*, *old-young*) which occur frequently in our data and which are frequently used to distinguish between objects occurring in the same situation. In particular, as the scenes in our data sets contain many objects of the same category (e.g. in the REFCOCO data), the distinction can often not be made by looking at the noun only, e.g. for classifying '*old* lady' - '*young* lady' as non-coreferential.

We call this task *approximate* coreference detection as the premise and hypothesis might describe complementary aspects of the same object such that the distinction cannot be made perfectly without the original perceptual context. For instance, in some cases, *lady in blue* and *young lady* might denote the same referent, in others not (see Figure 1). Thus, we note that the upper bound for automatic (or human) performance in this task is clearly not 100%. In future work, we plan to combine this with a reference resolution system that grounds the expressions in a given image.

Data and Set-up Given an image with several objects and a set of expressions referring to these, we compute the set of expression pairs P for that image. This set now divides into positive instances, i.e. expressions that both refer to the same object in the image, and negative instances, i.e. expressions that describe distinct entities in the scene. As this gives us a lot of data, we adopt a supervised learning approach for modeling the task of approximate co-reference detection. Thus, we use our embeddings to extract a range of similarity measures between the expression pairs and feed these metrics as features into a classifier, trained to predict whether two phrases co-refer. This set-up is largely similar to Young et al. (2014)'s evaluation setting for approximate textual entailment.

Similarity Measures Given a pair P of expressions $r_i = w_{i_1} \dots w_{i_n}, r_j = w_{j_1} \dots w_{j_m}$, we extract pairwise cosine similarities between the embeddings $cos(w_{i_x}, w_{j_y})$, using average $(\sum_{(w_i,w_j)\in P} cos(w_i, w_j) \times \frac{1}{|P|})$, maximum $(\max_{(w_i,w_j)\in P} cos(w_i, w_j))$ and minimum distance $(\min_{(w_i,w_j)\in P} cos(w_i, w_j))$ as features for classification. Furthermore, we restrict the words in each expression such that they are disjunct sets excluding words that occur in both expressions, $w_i \neq w_j, \forall(w_i, w_j) \in P$. We extract the same average, maximum and minimum distance measures on these lexically disjunct expressions. Finally, we compose word embeddings for each expressions via addition ($r_i = w_{i_1} + \dots + w_{i_n}$) and add the cosine between the composed embeddings ($cos(r_i, r_j)$) to our list of features. Here, we compare textual, visual and denotational embeddings, as our situational embeddings only cover nouns.

Training From REFERIT, we extract 161K training and 18K test pairs, dividing into 66% non-coreferential and 34% coreferential expressions. We re-train our embeddings on the training portions of this data. We only consider non-coreferential expressions that refer to objects of the same type, according to their label annotated in the data set. From REFCOCO, we extract 300k pairs from the training set and 95k pairs from the test set, dividing into roughly 70% non-coreferential and 30% coreferential expressions. We randomly sample these pairs, the overall number of possible pairs in REFCOCO exceeds 2 million. We train a binary logistic regression classifier on each corpus, given the similarity measures extracted for each word embedding.

Results We report accuracies on co-referential expression detection in Table 4, on REFERIT and REFCOCO. Similarities derived from denotational embeddings clearly outperform the other classifiers on both data sets, including state-of-the-art textual embeddings learned on a much larger web corpus. On REFCOCO, only denotational embeddings lead to a clear improvement over the majority baseline. While the low performance of standard distributional embeddings is rather expected on this task (see previous findings on e.g. predicting antonyms (Nguyen et al., 2016)), the clear advange of denotational over visual embeddings is noteworthy. Whereas visual grounding is relatively effective for modeling compatibility between nouns (see Table 1), it does not seem to capture attribute meaning accurately as illustrated in Table 5. Here, the average of all visual objects referred to as e.g. *small* seems to be rather noisy and lead to high similarity with rather random words (*directly*) whereas denotational embeddings model accurate compatibility relations between e.g. *small-smaller*.

	ReferIt	RefCoco
Majority	66.05	71.64
VIS	70.14	71.63
TXT$_{ref}$	68.49	71.57
DEN	**73.67**	**74.32**
TXT$_{web}$	69.16	71.89

Table 4: Accuracies for co-referential expression detection

top	txt$_{ref}$	upper, bototm, bottom, bottem
	den	upper, topmost, tippy, above
	vis	upper, above, of, corner
red	text	yellow, purple, maroon, blue
	den	maroon, redman, reddish, allmiddle
	vis	and, purple, yellow, pink
small	txt$_{ref}$	large, smaller, big, tiny
	den	smaller, smallest, little, littiest
	vis	directly, of, between, slightly

Table 5: Top nearest neighbours for some example adjectives embeddings

5 Conclusion

Whereas it is notoriously difficult to tailor or specialise distributional meaning representations inferred from text to particular aspects of semantic relatedness (Kiela et al., 2015a; Nguyen et al., 2016; Rimell et al., 2017), this work has shown that a multi-modal corpus of referring expressions can be used to derive a range of continuous meaning representations grounded in different aspects of context, capturing different notions of similarity. As compared to visual embeddings used in previous works, we found that denotational embeddings are particularly useful for detecting semantic relations. Other, recently proposed tasks related to modeling word association (Vulić et al., 2017), commonsense knowledge (Vedantam et al., 2015) or child-directed input (Lazaridou et al., 2016) provide interesting testbeds for future work.

Acknowledgments

We acknowledge support by the Cluster of Excellence "Cognitive Interaction Technology" (CITEC; EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*, pages 238–247.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10. Association for Computational Linguistics.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Anne Cocos and Chris Callison-Burch. 2017. The language of place: Semantic value from geospatial context. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 99–104. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Jia Deng, W. Dong, Richard Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. In *Proc. of the 1st Workshop on Evaluating Vector Space Representations for NLP*.

Yansong Feng and Mirella Lapata. 2010. Visual information in semantic representation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 91–99. Association for Computational Linguistics.

Michael Grubinger, Paul Clough, Henning Müller, and Thomas Deselaers. 2006. The IAPR TC-12 benchmark: a new evaluation resource for visual information systems. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2006)*, pages 13–23, Genoa, Italy.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L Berg. 2014. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 787–798, Doha, Qatar.

Douwe Kiela and Léon Bottou. 2014. Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015a. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2044–2048, Lisbon, Portugal. Association for Computational Linguistics.

Douwe Kiela, Laura Rimell, Ivan Vulić, and Stephen Clark. 2015b. Exploiting image generality for lexical entailment detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 119–124, Beijing, China. Association for Computational Linguistics.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1:193–206.

Germán Kruszewski and Marco Baroni. 2015. So similar and yet incompatible: Toward the automated identification of semantically compatible words. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 964–969, Denver, Colorado. Association for Computational Linguistics.

Angeliki Lazaridou, Grzegorz Chrupała, Raquel Fernández, and Marco Baroni. 2016. Multimodal semantic learning from child-directed input. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 387–392, San Diego, California. Association for Computational Linguistics.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skip-gram model. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 153–163, Denver, Colorado. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland. Association for Computational Linguistics.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C.Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision ECCV 2014*, volume 8693, pages 740–755. Springer International Publishing.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2015. Generation and comprehension of unambiguous object descriptions. *ArXiv / CoRR*, abs/1511.02283.

Cynthia Matuszek, Nicholas Fitzgerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A Joint Model of Language and Perception for Grounded Attribute Learning. In *Proceedings of the International Conference on Machine Learning (ICML 2012)*.

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

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 454–459, Berlin, Germany. Association for Computational Linguistics.

Laura Rimell, Amandla Mabona, Luana Bulat, and Douwe Kiela. 2017. Learning to negate adjectives with bilinear models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 71–78. Association for Computational Linguistics.

David Schlangen, Sina Zarriess, and Casey Kennington. 2016. Resolving references to objects in photographs using the words-as-classifiers model. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, page To appear.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 721–732, Baltimore, Maryland. Association for Computational Linguistics.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR 2015*, Boston, MA, USA.

Ramakrishna Vedantam, Xiao Lin, Tanmay Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Learning common sense through visual abstraction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2542–2550.

Ivan Vulić, Douwe Kiela, and Anna Korhonen. 2017. Evaluation by association: A systematic study of quantitative word association evaluation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 163–175. Association for Computational Linguistics.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. 2016. *Modeling Context in Referring Expressions*. Springer International Publishing.

Hierarchically-Attentive RNN for Album Summarization and Storytelling

Licheng Yu and **Mohit Bansal** and **Tamara L. Berg**
UNC Chapel Hill
{licheng, mbansal, tlberg}@cs.unc.edu

Abstract

We address the problem of end-to-end visual storytelling. Given a photo album, our model first selects the most representative (summary) photos, and then composes a natural language story for the album. For this task, we make use of the Visual Storytelling dataset and a model composed of three hierarchically-attentive Recurrent Neural Nets (RNNs) to: encode the album photos, select representative (summary) photos, and compose the story. Automatic and human evaluations show our model achieves better performance on selection, generation, and retrieval than baselines.

1 Introduction

Since we first developed language, humans have always told stories. Fashioning a good story is an act of creativity and developing algorithms to replicate this has been a long running challenge. Adding pictures as input can provide information for guiding story construction by offering visual illustrations of the storyline. In the related task of image captioning, most methods try to generate descriptions only for individual images or for short videos depicting a single activity. Very recently, datasets have been introduced that extend this task to longer temporal sequences such as movies or photo albums (Rohrbach et al., 2016; Pan et al., 2016; Lu and Grauman, 2013; Huang et al., 2016).

The type of data we consider in this paper provides input illustrations for story generation in the form of photo albums, sampled over a few minutes to a few days of time. For this type of data, generating textual descriptions involves telling a temporally consistent story about the depicted visual information, where stories must be coherent and take into account the temporal context of the images. Applications of this include constructing visual and textual summaries of albums, or even enabling search through personal photo collections to find photos of life events.

Previous visual storytelling works can be classified into two types, vision-based and language-based, where image or language stories are constructed respectively. Among the vision-based approaches, unsupervised learning is commonly applied: e.g., (Sigurdsson et al., 2016) learns the latent temporal dynamics given a large amount of albums, and (Kim and Xing, 2014) formulate the photo selection as a sparse time-varying directed graph. However, these visual summaries tend to be difficult to evaluate and selected photos may not agree with human selections. For language-based approaches, a sequence of natural language sentences are generated to describe a set of photos. To drive this work (Park and Kim, 2015) collected a dataset mined from Blog Posts. However, this kind of data often contains contextual information or loosely related language. A more direct dataset was recently released (Huang et al., 2016), where multi-sentence stories are collected describing photo albums via Amazon Mechanical Turk.

In this paper, we make use of the Visual Storytelling Dataset (Huang et al., 2016). While the authors provide a seq2seq baseline, they only deal with the task of generating stories given 5-representative (summary) photos hand-selected by people from an album. Instead, we focus on the more challenging and realistic problem of end-to-end generation of stories from entire albums. This requires us to either generate a story from all of the album's photos or to learn selection mechanisms to identify representative photos and then generate stories from those summary photos. We evaluate each type of approach.

Ultimately, we propose a model of hierarchically-attentive recurrent neural nets,

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 966–971
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

consisting of three RNN stages. The first RNN encodes the whole album context and each photo's content, the second RNN provides weights for photo selection, and the third RNN takes the weighted representation and decodes to the resulting sentences. Note that during training, we are only given the full input albums and the output stories, and our model needs to learn the summary photo selections latently.

We show that our model achieves better performance over baselines under both automatic metrics and human evaluations. As a side product, we show that the latent photo selection also reasonably mimics human selections. Additionally, we propose an album retrieval task that can reliably pick the correct photo album given a sequence of sentences, and find that our model also outperforms the baselines on this task.

2 Related work

Recent years have witnessed an explosion of interest in vision and language tasks, reviewed below.

Visual Captioning: Most recent approaches to image captioning (Vinyals et al., 2015b; Xu et al., 2015) have used CNN-LSTM structures to generate descriptions. For captioning video or movie content (Venugopalan et al., 2015; Pan et al., 2016), sequence-to-sequence models are widely applied, where the first sequence encodes video frames and the second sequence decodes the description. Attention techniques (Xu et al., 2015; Yu et al., 2016; Yao et al., 2015) are commonly incorporated for both tasks to localize salient temporal or spatial information.

Video Summarization: Similar to documentation summarization (Rush et al., 2015; Cheng and Lapata, 2016; Mei et al., 2016; Woodsend and Lapata, 2010) which extracts key sentences and words, video summarization selects key frames or shots. While some approaches use unsupervised learning (Lu and Grauman, 2013; Khosla et al., 2013) or intuitive criteria to pick salient frames, recent models learn from human-created summaries (Gygli et al., 2015; Zhang et al., 2016b,a; Gong et al., 2014). Recently, to better exploit semantics, (Choi et al., 2017) proposed textually customized summaries.

Visual Storytelling: Visual storytelling tries to tell a coherent visual or textual story about an image set. Previous works include storyline graph modeling (Kim and Xing, 2014), unsupervised mining (Sigurdsson et al., 2016), blog-photo alignment (Kim et al., 2015), and language retelling (Huang et al., 2016; Park and Kim, 2015). While (Park and Kim, 2015) collects data by mining Blog Posts, (Huang et al., 2016) collects stories using Mechanical Turk, providing more directly relevant stories.

3 Model

Our model (Fig. 1) is composed of three modules: Album Encoder, Photo Selector, and Story Generator, jointly learned during training.

3.1 Album Encoder

Given an album $A = \{a_1, a_2, ..., a_n\}$, composed of a set of photos, we use a bi-directional RNN to encode the local album context for each photo. We first extract the 2048-dimensional visual representation $f_i \in R^k$ for each photo using ResNet101 (He et al., 2016), then a bi-directional RNN is applied to encode the full album. Following (Huang et al., 2016), we choose a Gated Recurrent Unit (GRU) as the RNN unit to encode the photo sequence. The sequence output at each time step encodes the local album context for each photo (from both directions). Fused with the visual representation followed by ReLU, our final photo representation is (top module in Fig. 1):

$$\begin{aligned} f_i &= \text{ResNet}(a_i) \\ \vec{h}_i &= \overrightarrow{\text{GRU}}_{album}(f_i, \vec{h}_{i-1}) \\ \overleftarrow{h}_i &= \overleftarrow{\text{GRU}}_{album}(f_i, \overleftarrow{h}_{i+1}) \\ v_i &= \text{ReLU}([\vec{h}_i, \overleftarrow{h}_i] + f_i). \end{aligned}$$

3.2 Photo Selector

The Photo Selector (illustrated in the middle yellow part of Fig. 1) identifies representative photos to summarize an album's content. As discussed, we do not assume that we are given the ground-truth album summaries during training, instead regarding selection as a latent variable in the end-to-end learning. Inspired by Pointer Networks (Vinyals et al., 2015a), we use another GRU-RNN to perform this task [1].

Given the album representation $V^{n \times k}$, the photo selector outputs probabilities $p_t \in R^n$ (likelihood of selection as t-th summary image) for all photos using soft attention.

$$\begin{aligned} \bar{h}_t &= \text{GRU}_{select}(p_{t-1}, \bar{h}_{t-1}), \\ p(y_{a_i}(t) = 1) &= \sigma(\text{MLP}([\bar{h}_t, v_i])), \end{aligned}$$

[1] While the pointer network requires grounding labels, we regard the labels as latent variables

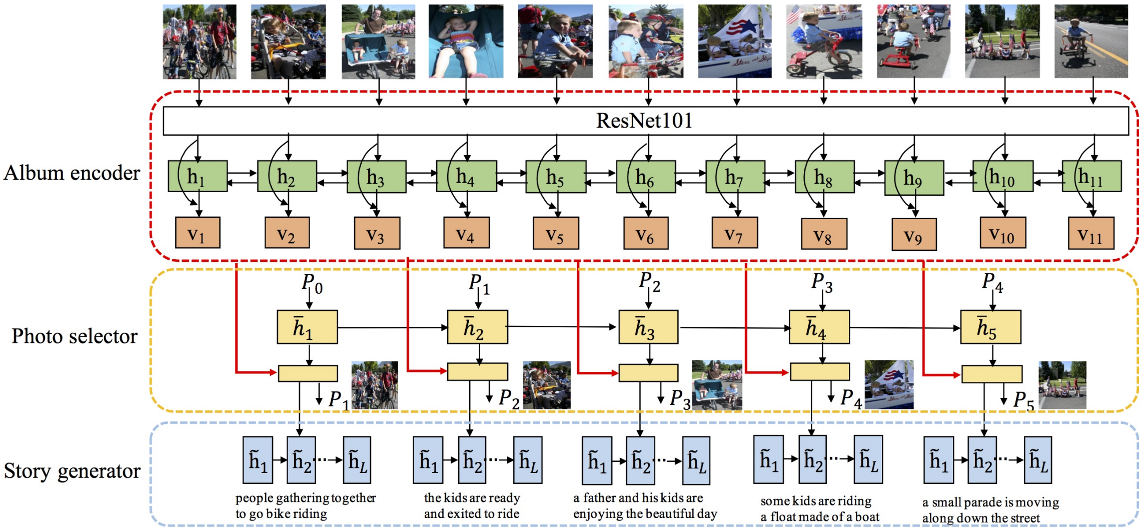

ResNet101
Album encoder
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8 h_9 h_{10} h_{11}
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11}
Photo selector
P_0 P_1 P_2 P_3 P_4
$\bar{h}_1$ $\bar{h}_2$ $\bar{h}_3$ $\bar{h}_4$ $\bar{h}_5$
P_1 P_2 P_3 P_4 P_5
Story generator
$\tilde{h}_1$ $\tilde{h}_2$ $\tilde{h}_L$
people gathering together to go bike riding
the kids are ready and exited to ride
a father and his kids are enjoying the beautiful day
some kids are riding a float made of a boat
a small parade is moving along down the street

Figure 1: Model: the *album encoder* is a bi-directional GRU-RNN that encodes all album photos; the *photo selector* computes the probability of each photo being the tth album-summary photo; and finally, the *story generator* outputs a sequence of sentences that combine to tell a story for the album.

At each summarization step, t, the GRU takes the previous p_{t-1} and previous hidden state as input, and outputs the next hidden state $\bar{h}_t$. $\bar{h}_t$ is fused with each photo representation v_i to compute the i^{th} photo's attention $p_t^i = p(y_{a_i}(t) = 1)$. At test time, we simply pick the photo with the highest probability to be the summary photo at step t.

3.3 Story Generator

To generate an album's story, given the album representation matrix V and photo summary probabilities p_t from the first two modules, we compute the visual summary representation $g_t \in R^k$ (for the t-th summary step). This is a weighted sum of the album representations, i.e., $g_t = p_t^T V$. Each of these 5 g_t embeddings (for $t = 1$ to 5) is then used to decode 1 of the 5 story sentences respectively, as shown in the blue part of Fig. 1.

Given a story $S = \{s_t\}$, where s_t is t-th summary sentence. Following Donahue et al. (2015), the l-th word probability of the t-th sentence is:

$$\begin{aligned} w_{t,l-1} &= W_e s_{t,l-1}, \\ \tilde{h}_{t,l} &= \text{GRU}_{story}(w_{t,l-1}, g_t, \tilde{h}_{t,l-1}), \\ p(s_{t,l}) &= \text{softmax}(\text{MLP}(\tilde{h}_{t,l})), \end{aligned} \tag{1}$$

where W_e is the word embedding. The GRU takes the joint input of visual summarization g_t, the previous word embedding $w_{t,l}$, and the previous hidden state, then outputs the next hidden state. The generation loss is then the sum of the negative log likelihoods of the correct words: $L_{gen}(S) = -\sum_{t=1}^{T} \sum_{l=1}^{L_t} \log p_{t,l}(s_{t,l})$.

To further exploit the notion of temporal coherence in a story, we add an order-preserving constraint to order the sequence of sentences within a story (related to the story-sorting idea in Agrawal et al. (2016)). For each story S we randomly shuffle its 5 sentences to generate negative story instances S'. We then apply a max-margin ranking loss to encourage correctly-ordered stories: $L_{rank}(S, S') = \max(0, m - \log p(S') + \log p(S))$. The final loss is then a combination of the generation and ranking losses:

$$L = L_{gen}(S) + \lambda L_{rank}(S, S'). \tag{2}$$

4 Experiments

We use the Visual Storytelling Dataset (Huang et al., 2016), consisting of 10,000 albums with 200,000 photos. Each album contains 10-50 photos taken within a 48-hour span with two annotations: 1) 2 album summarizations, each with 5 selected representative photos, and 2) 5 stories describing the selected photos.

4.1 Story Generation

This task is to generate a 5-sentence story describing an album. We compare our model with two sequence-to-sequence baselines: 1) an encoder-decoder model (enc-dec), where the sequence of album photos is encoded and the last hidden state is fed into the decoder for story generation, 2) an encoder-attention-decoder model (Xu et al., 2015) (enc-attn-dec) with weights computed using a soft-attention mechanism. At each decoding time step, a weighted sum of hidden states from the encoder is decoded. For fair comparison, we

	beam size=3			
	Bleu3	Rouge	Meteor	CIDEr
enc-dec	19.58	29.23	33.02	4.65
enc-attn-dec	19.73	28.94	32.98	4.96
h-attn	20.53	29.82	33.81	6.84
h-attn-rank	**20.78**	**29.82**	**33.94**	**7.38**
h-(gd)attn-rank	21.02	29.53	34.12	7.51

Table 1: Story generation evaluation.

enc-dec (29.50%)	h-attn-rank (70.50%)
enc-attn-dec (30.75%)	h-attn-rank (69.25%)
h-attn-rank (30.50%)	gd-truth (69.50%)

Table 2: Human evaluation showing how often people prefer one model over the other.

use the same album representation (Sec. 3.1) for the baselines.

We test two variants of our model trained with and without ranking regularization by controlling λ in our loss function, denoted as h-attn (without ranking), and h-attn-rank (with ranking). Evaluations of each model are shown in Table 1. The h-attn outperforms both baselines, and h-attn-rank achieves the best performance for all metrics. Note, we use beam-search with beam size=3 during generation for a reasonable performance-speed trade-off (we observe similar improvement trends with beam size = 1).[2] To test performance under optimal image selection, we use one of the two ground-truth human-selected 5-photo-sets as an oracle to hard-code the photo selection, denoted as h-(gd)attn-rank. This achieves only a slightly higher Meteor compared to our end-to-end model.

Additionally, we also run human evaluations in a forced-choice task where people choose between stories generated by different methods. For this evaluation, we select 400 albums, each evaluated by 3 Turkers. Results are shown in Table 2. Experiments find significant preference for our model over both baselines. As a simple Turing test, we also compare our results with human written stories (last row of Table 2), indicating room for improvement of methods.

4.2 Album Summarization

We evaluate the precision and recall of our generated summaries (output by the photo selector) compared to human selections (the combined set of both human-selected 5-photo stories). For comparison, we evaluate enc-attn-dec on the same task by aggregating predicted attention and selecting the 5 photos with highest accumulated attention. Additionally, we also run DPP-based video summarization (Kulesza et al., 2012) using the same album features. Our models have higher performance compared to baselines as shown in Table 3 (though DPP also achieves strong results, indicating that there is still room to improve the pointer network).

	precision	recall
DPP	43.75%	27.41%
enc-attn-dec	38.53%	24.25%
h-attn	42.85%	27.10%
h-attn-rank	**45.51%**	**28.77%**

Table 3: Album summarization evaluation.

	R@1	R@5	R@10	MedR
enc-dec	10.70%	29.30%	41.40%	14.5
enc-attn-dec	11.60%	33.00%	45.50%	11.0
h-attn	18.30%	**44.50%**	**57.60%**	**6.0**
h-attn-rank	**18.40%**	43.30%	55.50%	7.0

Table 4: 1000 album retrieval evaluation.

4.3 Output Example Analysis

Fig. 2 shows several output examples for both summarization and story generation, comparing our model to the baseline and ground-truth. More examples are provided in the supplementary.

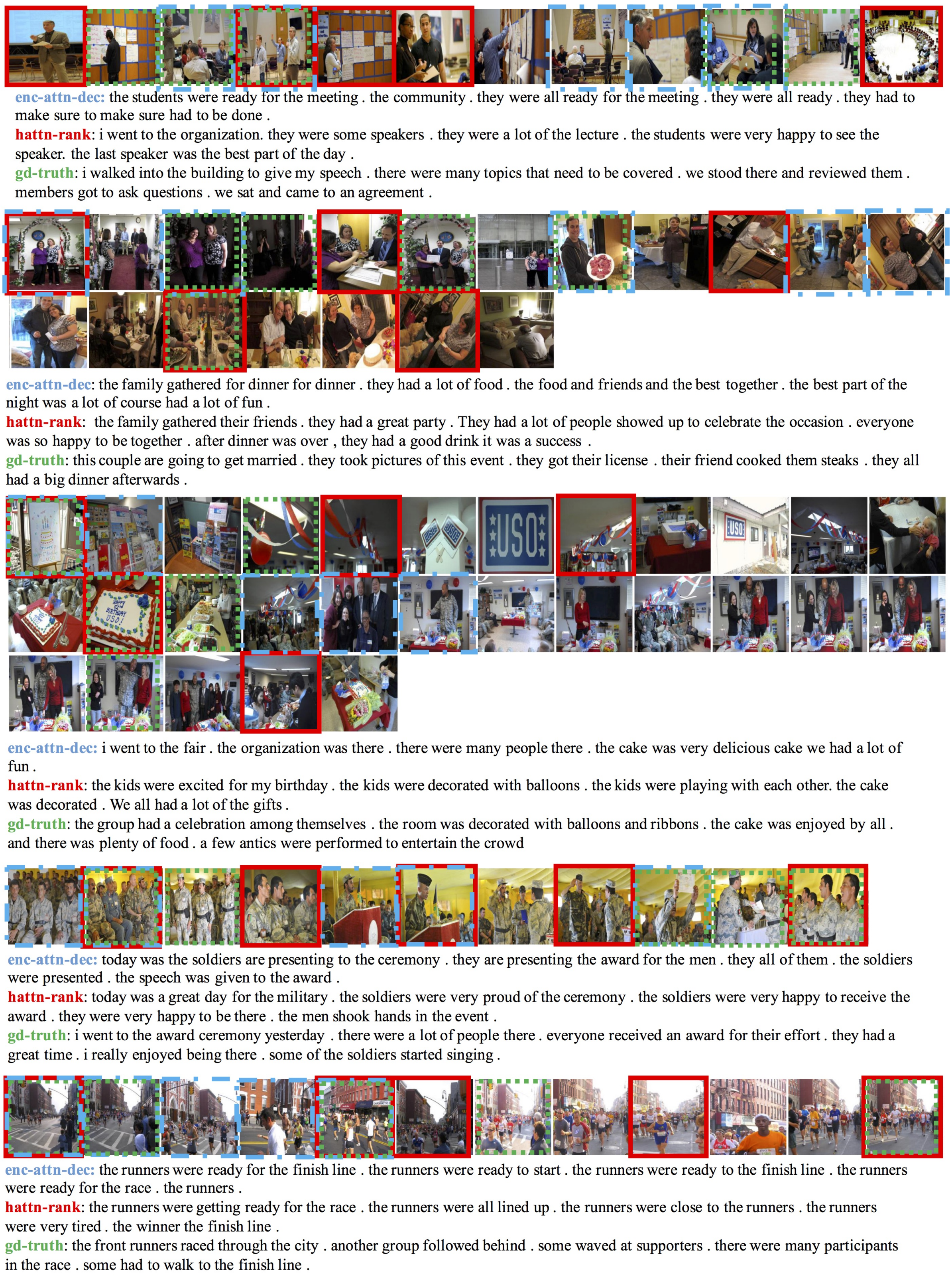
4.4 Album Retrieval

Given a human-written story, we introduce a task to retrieve the album described by that story. We randomly select 1000 albums and one ground-truth story from each for evaluation. Using the generation loss, we compute the likelihood of each album A_m given the query story S and retrieve the album with the highest generation likelihood, $A = \text{argmax}_{A_m} p(S|A_m)$. We use Recall@k and Median Rank for evaluation. As shown in Table 4), we find that our models outperform the baselines, but the ranking term in Eqn.2 does not improve performance significantly.

5 Conclusion

Our proposed hierarchically-attentive RNN based models for end-to-end visual storytelling can jointly summarize and generate relevant stories from full input photo albums effectively. Automatic and human evaluations show that our method outperforms strong sequence-to-sequence

[2] We also compute the p-value of Meteor on 100K samples via the bootstrap test (Efron and Tibshirani, 1994), as Meteor has better agreement with human judgments than Bleu/Rouge (Huang et al., 2016). Our h-attn-rank model has strong statistical significance ($p = 0.01$) over the enc-dec and enc-attn-dec models (and is similar to the h-attn model).



enc-attn-dec: the students were ready for the meeting . the community . they were all ready for the meeting . they were all ready . they had to make sure to make sure had to be done .

hattn-rank: i went to the organization. they were some speakers . they were a lot of the lecture . the students were very happy to see the speaker. the last speaker was the best part of the day .

gd-truth: i walked into the building to give my speech . there were many topics that need to be covered . we stood there and reviewed them . members got to ask questions . we sat and came to an agreement .

enc-attn-dec: the family gathered for dinner for dinner . they had a lot of food . the food and friends and the best together . the best part of the night was a lot of course had a lot of fun .

hattn-rank: the family gathered their friends . they had a great party . They had a lot of people showed up to celebrate the occasion . everyone was so happy to be together . after dinner was over , they had a good drink it was a success .

gd-truth: this couple are going to get married . they took pictures of this event . they got their license . their friend cooked them steaks . they all had a big dinner afterwards .

enc-attn-dec: i went to the fair . the organization was there . there were many people there . the cake was very delicious cake we had a lot of fun .

hattn-rank: the kids were excited for my birthday . the kids were decorated with balloons . the kids were playing with each other. the cake was decorated . We all had a lot of the gifts .

gd-truth: the group had a celebration among themselves . the room was decorated with balloons and ribbons . the cake was enjoyed by all . and there was plenty of food . a few antics were performed to entertain the crowd

enc-attn-dec: today was the soldiers are presenting to the ceremony . they are presenting the award for the men . they all of them . the soldiers were presented . the speech was given to the award .

hattn-rank: today was a great day for the military . the soldiers were very proud of the ceremony . the soldiers were very happy to receive the award . they were very happy to be there . the men shook hands in the event .

gd-truth: i went to the award ceremony yesterday . there were a lot of people there . everyone received an award for their effort . they had a great time . i really enjoyed being there . some of the soldiers started singing .

enc-attn-dec: the runners were ready for the finish line . the runners were ready to start . the runners were ready to the finish line . the runners were ready for the race . the runners .

hattn-rank: the runners were getting ready for the race . the runners were all lined up . the runners were close to the runners . the runners were very tired . the winner the finish line .

gd-truth: the front runners raced through the city . another group followed behind . some waved at supporters . there were many participants in the race . some had to walk to the finish line .

Figure 2: Examples of album summarization and storytelling by enc-attn-dec (blue), h-attn-rank (red), and ground-truth (green). We randomly select 1 out of 2 human album summaries as ground-truth here.

baselines on selection, generation, and retrieval tasks.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This research is supported by NSF Awards #1633295, 1444234, 1445409, 1562098.

References

Harsh Agrawal, Arjun Chandrasekaran, Dhruv Batra, Devi Parikh, and Mohit Bansal. 2016. Sort story: Sorting jumbled images and captions into stories. In *EMNLP*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *ACL*.

Jinsoo Choi, Tae-Hyun Oh, and In So Kweon. 2017. Textually customized video summaries. *arXiv preprint arXiv:1702.01528*.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. 2014. Diverse sequential subset selection for supervised video summarization. In *NIPS*.

Michael Gygli, Helmut Grabner, and Luc Van Gool. 2015. Video summarization by learning submodular mixtures of objectives. In *CVPR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Ting-Hao Kenneth Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. 2016. Visual storytelling. In *NACCL*.

Aditya Khosla, Raffay Hamid, Chih-Jen Lin, and Neel Sundaresan. 2013. Large-scale video summarization using web-image priors. In *CVPR*.

Gunhee Kim, Seungwhan Moon, and Leonid Sigal. 2015. Joint photo stream and blog post summarization and exploration. In *CVPR*.

Gunhee Kim and Eric P Xing. 2014. Reconstructing storyline graphs for image recommendation from web community photos. In *CVPR*.

Alex Kulesza, Ben Taskar, et al. 2012. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*.

Zheng Lu and Kristen Grauman. 2013. Story-driven summarization for egocentric video. In *CVPR*.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL*.

Yingwei Pan, Tao Mei, Ting Yao, Houqiang Li, and Yong Rui. 2016. Jointly modeling embedding and translation to bridge video and language. In *CVPR*.

Cesc C Park and Gunhee Kim. 2015. Expressing an image stream with a sequence of natural sentences. In *NIPS*.

Anna Rohrbach, Atousa Torabi, Marcus Rohrbach, Niket Tandon, Christopher Pal, Hugo Larochelle, Aaron Courville, and Bernt Schiele. 2016. Movie description. *IJCV*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

Gunnar A Sigurdsson, Xinlei Chen, and Abhinav Gupta. 2016. Learning visual storylines with skipping recurrent neural networks. In *ECCV*.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence-video to text. In *ICCV*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In *NIPS*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015b. Show and tell: A neural image caption generator. In *CVPR*.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *ACL*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Describing videos by exploiting temporal structure. In *ICCV*.

Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. 2016. Video paragraph captioning using hierarchical recurrent neural networks. In *CVPR*.

Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. 2016a. Summary transfer: Exemplar-based subset selection for video summarization. In *CVPR*.

Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. 2016b. Video summarization with long short-term memory. In *ECCV*.

Video Highlight Prediction Using Audience Chat Reactions

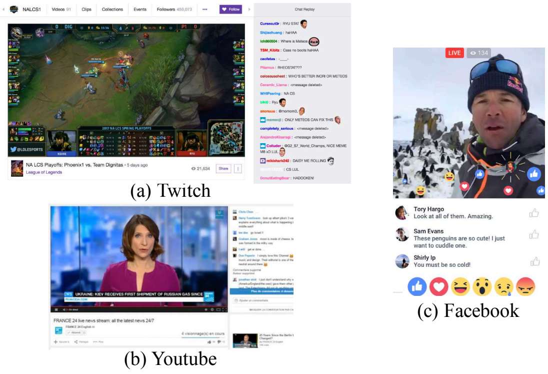
Cheng-Yang Fu, Joon Lee, Mohit Bansal, Alex C. Berg
UNC Chapel Hill
{cyfu, joonlee, mbansal, aberg}@cs.unc.edu

Abstract

Sports channel video portals offer an exciting domain for research on multimodal, multilingual analysis. We present methods addressing the problem of automatic video highlight prediction based on joint visual features and textual analysis of the real-world audience discourse with complex slang, in both English and traditional Chinese. We present a novel dataset based on League of Legends championships recorded from North American and Taiwanese Twitch.tv channels (will be released for further research), and demonstrate strong results on these using multimodal, character-level CNN-RNN model architectures.

1 Introduction

On-line eSports events provide a new setting for observing large-scale social interaction focused on a visual story that evolves over time—a video game. While watching sporting competitions has been a major source of entertainment for millennia, and is a significant part of today's culture, eSports brings this to a new level on several fronts. One is the global reach, the same games are played around the world and across cultures by speakers of several languages. Another is the scale of on-line text-based discourse during matches that is public and amendable to analysis. One of the most popular games, League of Legends, drew 43 million views for the 2016 world series final matches (broadcast in 18 languages) and a peak concurrent viewership of 14.7 million[1]. Finally, players interact through what they see on screen while fans (and researchers) can see *exactly* the same views.


(a) Twitch

(b) Youtube

Tory Hargo
Look at all of them. Amazing.

Sam Evans
These penguins are so cute! I just want to cuddle one.

Shirly Ip
You must be so cold!

(c) Facebook

Figure 1: Pictures of Broadcasting platforms:(a) Twitch: League of Legends Tournament Broadcasting, (b) Youtube: News Channel, (c)Facebook: Personal live sharing

This paper builds on the wealth of interaction around eSports to develop predictive models for match video highlights based on the audience's online chat discourse as well as the visual recordings of matches themselves. ESports journalists and fans create highlight videos of important moments in matches. Using these as ground truth, we explore automatic prediction of highlights via multimodal CNN+RNN models for multiple languages. Appealingly this task is *natural*, as the community already produces the ground truth and is global, allowing multilingual multimodal grounding.

Highlight prediction is about capturing the exciting moments in a specific video (a game match in this case), and depends on the context, the state of play, and the players. This task of predicting the exciting moments is hence different from summarizing the entire match into a story summary. Hence, highlight prediction can benefit from the available real-time text commentary from fans, which is valuable in exposing more abstract background context, that may not be accessible with

[1] http://www.lolesports.com/en_US/articles/2016-league-legends-world-championship-numbers

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 972–978
Copenhagen, Denmark, September 7–11, 2017.

computer vision techniques that can easily identify some aspects of the state of play. As an example, computer vision may not understand why Michael Jordan's dunk is a highlight over that of another player, but concurrent fan commentary might reveal this.

We collect our dataset from Twitch.tv, one of the live-streaming platforms that integrates comments (see Fig. 1), and the largest live-streaming platform for video games. We record matches of the game League of Legends (LOL), one of the largest eSports game in two subsets, 1) the spring season of the North American League of Legends Championship Series (NALCS), and 2) the League of Legends Master Series (LMS) hosted in Taiwan/Macau/HongKong, with chat comments in English and traditional Chinese respectively. We use the community created highlights to label each frame of a match as highlight or not.

In addition to our new dataset, we present several experiments with multilingual character-based models, deep-learning based vision models either per-frame or tied together with a video-sequence LSTM-RNN, and combinations of language and vision models. Our results indicate that while *surprisingly* the visual models generally outperform language-based models, we can still build reasonably useful language models that help disambiguate difficult cases for vision models, and that combining the two sources is the most effective model (across multiple languages).

2 Related Work

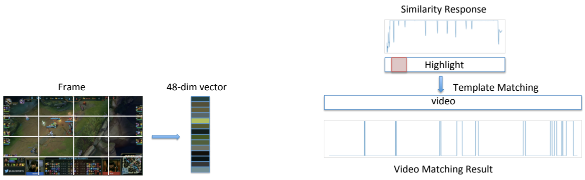
We briefly discuss a small sample of the related work on language and vision datasets, summarization, and highlight prediction. There has been a surge of vision and language datasets focusing on captions over the last few years, (Rashtchian et al., 2010; Ordonez et al., 2011; Lin et al., 2014), followed by efforts to focus on more specific parts of images (Krishna et al., 2016), or referring expressions (Kazemzadeh et al., 2014), or on the broader context (Huang et al., 2016). For video, similar efforts have collected descriptions (Chen and Dolan, 2011), while others use existing descriptive video service (DVS) sources (Rohrbach et al., 2015; Torabi et al., 2015). Beyond descriptions, other datasets use questions to relate images and language (Antol et al., 2015; Yu et al., 2015). This approach is extended to movies in Tapaswi et al. (2016).

The related problem of visually summarizing videos (as opposed to finding the highlights) has produced datasets of holiday and sports events with multiple users making summary videos (Gygli et al., 2014) and multiple users selecting summary key-frames (de Avila et al., 2011) from short videos. For language-based summarization, *Extractive models* (Filippova and Altun, 2013; Filippova et al., 2015) generate summaries by selecting important sentences and then assembling these, while *Abstractive models* (Chopra et al., 2016; Mei et al., 2016; Nallapati et al., 2016; See et al., 2017) generate/rewrite the summaries from scratch.

Closer to our setting, there has been work on highlight prediction in football (soccer) and basketball based on audio of broadcasts (Cheng and Hsu, 2006) (Wang et al., 2004) where commentators may have an outsized impact or visual features (Bertini et al., 2005). In the spirit of our study, there has been work looking at tweets during sporting events (Hsieh et al., 2012), but the tweets are not as immediate or as well aligned with the games as the eSports comments. More closely related to our work, Song (2016) collects videos for Heroes of the Storm, League of Legends, and Dota2 on online broadcasting websites of around 327 hours total. They also provide highlight labeling annotated by four annotators. Our method, on the other hand, has a similar scale of data, but we use existing highlights, and we also employ textual *audience* chat commentary, thus providing a new resource and task for Language and Vision research. In summary, we present the first language-vision dataset for video highlighting that contains audience reactions in chat format, in multiple languages. The community produced ground truth provides labels for each frame and can be used for supervised learning. The language side of this new dataset presents interesting challenges related to real-world Internet-style slang.

3 Data Collection

Our dataset covers 218 videos from NALCS and 103 from LMS for a total of 321 videos from week 1 to week 9 in 2017 spring series from each tournament. Each week there are 10 matches for NALCS and 6 matches for LMS. Matches are best of 3, so consist of two games or three games. The first and third games are used for training. The second games in the first 4 weeks are used as valida-



(a) Feature vector of frame

(b) Template Matching

Figure 2: Highlight Labeling: (a) The feature representation of each frame is calculated by averaging each color channel in each subregion. (b) After template matching, the top bar shows the maximum of similarity matching of each frame in the highlight and the bottom bar is the labeling result of the video.

tion and the remainder of second games are used as test. Table 1 lists the numbers of videos in train, validation, and test subsets.

Dataset	Train	Val	Testing	Total
NALCS	128	40	50	218
LMS	57	18	28	103

Table 1: Dataset statistics (number of videos).

Each game's video ranges from 30 to 50 minutes in length which contains image and chat data linked to the specific timestamp of the game. The average number of chats per video is 7490 with a standard deviation of 4922. The high value of standard deviation is mostly due to the fact that NALCS simultaneously broadcasts matches in two different channels (nalcs1[2] and nalcs2[3]) which often leads to the majority of users watching the channel with a relatively more popular team causing an imbalance in the number of chats. If we only consider LMS which broadcasts with a single channel, the average number of chats are 7210 with standard deviation of 2719. The number of viewers for each game averages about 21526, and the number of unique users who type in chat is on average 2185, i.e., roughly 10% of the viewers.

Highlight Labeling For each game, we collected community generated highlights ranging from 5 minutes to 7 minutes in length. For the purpose of consistency within our data, we collected the highlights from a single Youtube channel, Onivia,[4] which provided highlights for both championship tournaments in a consistent arrangement. We expect such consistency will aid our model to better pick up characteristics for determining highlights. We next need to align the position of the frames from the highlight video to frames in the full game video. For this, we adopted a template matching approach. For each frame in the video and the highlight, we divide it into 16 regions of 4 by 4 and use the average value of each color channel in each region as the feature. The feature representation of each frame ends up as a 48-dim vector as shown in Figure 2a. For each frame in the highlight, we can find the most similar frame in the video by calculating distance between these two vectors. However, matching a single frame to another suffers from noise. Therefore, we alternatively concatenate the following frames to form a window and use template matching to find the best matching location in the video. We found out that when the window size is 60 frames, it gives consistent and high quality results. For each frame, the result contains not only the best matching score but also the location of that match in the video.[5] Figure 2b illustrates this matching process.

4 Model

In this section, we explain the proposed models and components. We first describe the notation and definition of the problem, plus the evaluation metric used. Next, we explain our vision model V-CNN-LSTM and language model L-Char-LSTM. Finally, we describe the joint multimodal model lv-LSTM.

Problem Definition Our basic task is to determine if a frame of the full input video should be labeled as being part of the output highlight or not. To simplify our notation, we use $X = \{x_1, x_2, ..., x_t\}$ to denote a sequence of features for frames. Chats are expressed as $C = \{(c_1, ts_1), ..., (c_n, ts_n)\}$. where each chat c comes with a timestamp ts. Methods take the image features and/or chats and predict labels for the frames, $Y = \{y_1, y_2, ..., y_t\}$.

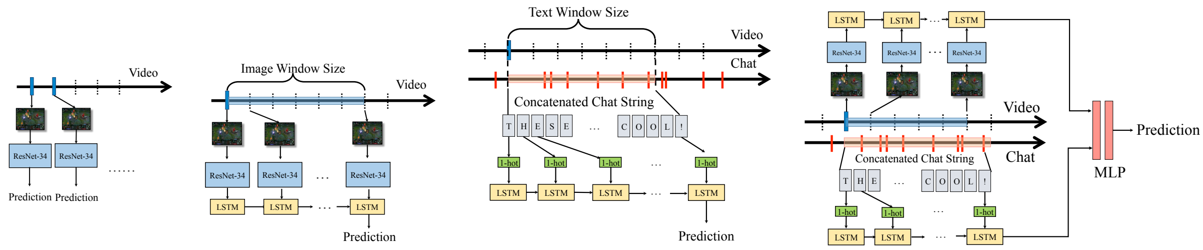
Evaluation Metric: We refer to the set of frames with positive ground truth label as S_{gt} and the set

[2] https://www.twitch.tv/nalcs1

[3] https://www.twitch.tv/nalcs2

[4] https://www.youtube.com/channel/UCPhab209KEicqPJFAk9IZEA

[5] When the window contains a moment of clip transition in highlights, the best matching score appears low. This is used to separate all clips in the highlight. Then we can use the starting and end locations of each clip to label the video.



(a) V-CNN (b) V-CNN-LSTM (c) L-Char-LSTM (d) Full model : lv-LSTM

Figure 3: Network architecture of proposed models.

of predicted frames with a positive label as S_{pred}. Following (Gygli et al., 2014; Song et al., 2015), we use the harmonic mean F-score in Eq.2 widely used in video summarization task for evaluation:

$$P = \frac{S_{gt} \cap S_{pred}}{|S_{pred}|}, \quad R = \frac{S_{gt} \cap S_{pred}}{|S_{gt}|} \qquad (1)$$

$$F = \frac{2PR}{P + R} \times 100\% \qquad (2)$$

V-CNN We use the ResNet-34 model (He et al., 2016) to represent frames, motivated by its strong results on the ImageNet Challenge (Russakovsky et al., 2015). Our naive V-CNN model (Figure 3a) uses features from the pre-trained version of this network [6] directly to make prediction at each frame (which are resized to 224x224).

V-CNN-LSTM In order to exploit visual video information sequentially over time, we use a memory-based LSTM-RNN on top of the image features, so as to model long-term dependencies. All of our videos are 30FPS. As the difference between consecutive frames is usually minor, we run prediction every 10th frame during evaluation and interpolate predictions between these frames. During training, due to the GPU memory constraints, we unfold the LSTM cell 16 times. Therefore the image window size is around 5-seconds (16 samples every 10th frame from 30fps video). The hidden state from the last cell is used as the V-CNN-LSTM feature. This process is shown in Figure 3b.

L-Word-LSTM and L-Char-LSTM Next, we discuss our language-based models using the audience chat text. Word-level LSTM-RNN models (Sutskever et al., 2014) are a common approach to embedding sentences. Unfortunately, this does not fit our Internet-slang style language with irregularities, “mispelled” words (hapy, happppppy), emojis (^_^), abbreviations (LOL), marks (?!?!?!?!), or onomatopoeic cases (e.g., 4 which sounds like yes in traditional Chinese). People may type variant length of 4, e.g.,, 4444444 to express their remarks.

Therefore, alternatively, we model the audience chat with a character-level LSTM-RNN model (Graves, 2013). Characters of the language, Chinese, English, or Emojis, are expanded to multiple ASCII characters according to the two-character Unicode or other representations used on the chat servers. We encode a 1-hot vector for each ASCII input character. For each frame we use all chats that occur in the next W_t seconds which are called text window size to form the input for L-Char-LSTM. We concatenate all the chats in a window, separating them by a special stop character, and then fed to a 3-layer L-Char-LSTM model.[7] This model is shown in Figure 3c. Following the setting in Sec. 5, we evaluate the text window size from 5 seconds to 9 seconds, and got the following accuracies:32.1%, 29.6%, 41.5%, 28.2%, 34.4%. We achieved best results with text window size as 7 seconds, and used this in rest of the experiments.

Joint lv-LSTM Model Our final lv-LSTM model combines the best vision and language models: V-CNN-LSTM and L-Char-LSTM. For the vision and language models, we can extract features F_v and F_l from V-CNN-LSTN and L-Char-LSTM, respectively. Then we concatenate F_v and F_l, and feed it into a 2-layer MLP. The completed model is shown in Figure 3d. We expect there is room to improve this approach, by using more involved representations, e.g., Bilinear Pooling (Fukui et al., 2016), Memory Networks (Xiong et al., 2016), and Attention Models (Lu et al., 2016); this is future work.

[6] https://github.com/pytorch/pytorch

[7] The number of these stop characters is then an encoding of the number of chats in the window. Therefore, the L-Char-LSTM could learn to use this #chats information, if it is a useful feature. Also, some content has been deleted by Twitch.tv or the channel itself due to the usage of improper words. We use symbol ”\n” to replace such cases.

Method	Data	UF	P	R	F
L-Char-LSTM	C	100%	0.11	0.99	19.6
L-Char-LSTM	C	last 25%	0.35	0.51	41.5
L-Word-LSTM	C	last 25%	0.10	0.99	19.2
V-CNN	V	100%	0.40	0.93	56.2
V-CNN	V	last 25%	0.57	0.74	64.0
V-CNN-LSTM	V	last 25%	0.58	0.82	68.3
lv-LSTM	C+V	last 25%	0.77	0.72	**74.8**

Table 2: Ablation Study: Effects of various models. **C**:Chat, **V**:Video, **UF**: % of frames Used in highlight clips as positive training examples; **P**: Precision, **R**: Recall, **F**: F-score.

5 Experiments and Results

Training Details In development and ablation studies, we use train and val splits of the data from NALCS to evaluate models in Section 3. For the final results, models are retrained on the combination of train and val data (following major vision benchmarks e.g. PASCAL-VOC and COCO), and performance is measured on the test set. We separate the highlight prediction to three different tasks based on using different input data: videos, chats, and videos+chats. The details of dataset split are in Section 3. Our code is implemented in PyTorch.

To deal with the large number of frames total, we sample only 5k positive and 5k negative examples in each epoch. We use batch size of 32 and run 60 epochs in all experiments. Weight decay is 10^{-4} and learning rate is set as 10^{-2} in the first 20 epochs and 10^{-3} after that. Cross entropy loss is used. Highlights are generated by fans and consist of clips. We match each clip to when it happened in the full match and call this the highlight clip (non-overlapping). The action of interest (kill, objective control, etc.) often happens in the later part of a highlight clip, while the clip contains some additional context before that action that may help set the stage. For some of our experimental settings (Table 2), we used a heuristic of only including the last 25% frames in every highlight clip as positive training examples. During evaluation, we used all frames in the highlight clip.

Ablation Study Table 2 shows the performance of each module separately on the dev set. For the basic L-Char-LSTM and V-CNN models, using only the last 25% of frames in highlight clips in training works best. In order to evaluate the performance of L-Char-LSTM model, we also train a Word-LSTM model by tokenizing all the chats and only considering the words that appeared more than 10 times, which results in 10019 words. We use this vocabulary to encode the words to 1-hot vectors. The L-Char-LSTM outperforms L-Word-LSTM by 22.3%.

Method	Data	NALCS	LMS
L-Char-LSTM	chat	43.2	39.7
V-CNN-LSTM	video	72.2	69.2
lv-LSTM	chat+video	**74.7**	**70.0**

Table 3: Test Results on the NALCS (English) and LMS (Traditional Chinese) datasets.

Test Results Test results are shown in Table 3. Somewhat surprisingly, the vision only model is more accurate than the language only model, despite the real-time nature of the comment stream. This is perhaps due to the visual form of the game, where highlight events may have similar animations. However, including language with vision in the lv-LSTM model significantly improves over vision alone, as the comments may exhibit additional contextual information. Comparing results between ablation and the final test, it seems more data contributes to higher accuracy. This effect is more apparent in the vision models, perhaps due to complexity. Moreover, L-Char-LSTM performs better in English compared to traditional Chinese. From the numbers given in Section 3, variation in the number of chats in NALCS was much higher than LMS, which one may expect to have a critical effect in the language model. However, our results seem to suggest that the L-Char-LSTM model can pickup other factors of the chat data (e.g. content) instead of just counting the number of chats. We expect a different language model more suitable for the traditional Chinese language should be able to improve the results for the LMS data.

6 Conclusion

We presented a new dataset and multimodal methods for highlight prediction, based on visual cues and textual audience chat reactions in multiple languages. We hope our new dataset can encourage further multilingual, multimodal research.

Acknowledgments

We thank Tamara Berg, Phil Ammirato, and the reviewers for their helpful suggestions, and we acknowledge support from NSF 1533771.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *ICCV*.

Sandra E. F. de Avila, Ana P. B. Lopes, Antonio da Luz Jr., and Arnaldo de A. Arajo. 2011. Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method. In *Pattern Recognition Letters*.

M. Bertini, A. Del Bimbo, and W. Nunziati. 2005. Soccer videos highlight prediction and annotation in real time. *ICIAP*.

David L. Chen and William B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *ACL*.

Chih-Chieh Cheng and Chiou-Ting Hsu. 2006. Fusion of audio and motion information on hmm-based highlight extraction for baseball games. *IEEE Trans. Multimedia*.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL*.

Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. *EMNLP*.

Katja Filippova and Yasemin Altun. 2013. The lack of parallel data in sentence compression. *EMNLP*.

Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *EMNLP*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *Neural computation*.

Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool. 2014. Creating summaries from user videos. In *ECCV*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Liang-Chi Hsieh, Ching-Wei Lee, Tzu-Hsuan Chiu, and Winston Hsu. 2012. Live semantic sport highlight detection based on analyzing tweets of twitter. *ICME*.

Ting-Hao K. Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Aishwarya Agrawal, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. 2016. Visual storytelling. In *NAACL*.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. Referitgame: Referring to objects in photographs of natural scenes. In *EMNLP*.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotations. In *arXiv:1602.07332*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C. Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *NIPS*.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *NAACL*.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL*.

Vicente Ordonez, Girish Kulkarni, and Tamara L Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *NIPS*.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting image annotations using amazon's mechanical turk. *NAACL HLT workshop*.

Anna Rohrbach, Marcus Rohrbach, Niket Tandon, and Bernt Schiele. 2015. A dataset for movie description. In *CVPR*.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. Imagenet large scale visual recognition challenge. In *IJCV*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Yale Song. 2016. Real-time video highlights for yahoo esports. In *arXiv:1611.08780*.

Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes. 2015. Tvsum: Summarizing web videos using titles. In *CVPR*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. In *CVPR*.

Atousa Torabi, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Using descriptive video services to create a large data source for video annotation research. In *arXiv:1503.01070v1*.

Jinjun Wang, Changsheng Xu, Engsiong Chng, and Qi Tian. 2004. Sports highlight detection from keyword sequences using hmm. *ICME*.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *ICML*.

Licheng Yu, Eunbyung Park, Alexander C. Berg, and Tamara L. Berg. 2015. Visual madlibs: Fill-in-the-blank image description and question answering. In *ICCV*.

Reinforced Video Captioning with Entailment Rewards

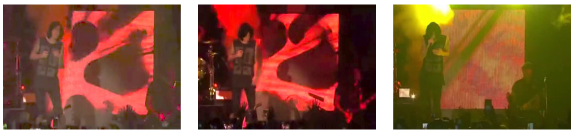
Ramakanth Pasunuru and **Mohit Bansal**
UNC Chapel Hill
{ram, mbansal}@cs.unc.edu

Abstract

Sequence-to-sequence models have shown promising improvements on the temporal task of video captioning, but they optimize word-level cross-entropy loss during training. First, using policy gradient and mixed-loss methods for reinforcement learning, we directly optimize sentence-level task-based metrics (as rewards), achieving significant improvements over the baseline, based on both automatic metrics and human evaluation on multiple datasets. Next, we propose a novel entailment-enhanced reward (CIDEnt) that corrects phrase-matching based metrics (such as CIDEr) to only allow for logically-implied partial matches and avoid contradictions, achieving further significant improvements over the CIDEr-reward model. Overall, our CIDEnt-reward model achieves the new state-of-the-art on the MSR-VTT dataset.

1 Introduction

The task of video captioning (Fig. 1) is an important next step to image captioning, with additional modeling of temporal knowledge and action sequences, and has several applications in online content search, assisting the visually-impaired, etc. Advancements in neural sequence-to-sequence learning has shown promising improvements on this task, based on encoder-decoder, attention, and hierarchical models (Venugopalan et al., 2015a; Pan et al., 2016a). However, these models are still trained using a word-level cross-entropy loss, which does not correlate well with the sentence-level metrics that the task is finally evaluated on (e.g., CIDEr, BLEU). Moreover, these models suffer from exposure bias (Ranzato et al., 2016), which occurs when a model is only exposed to the training data distribution, instead of its own predictions. First, using a sequence-level training, policy gradient approach (Ranzato et al., 2016), we allow video captioning models to directly optimize these non-differentiable metrics, as rewards in a reinforcement learning paradigm. We also address the exposure bias issue by using a mixed-loss (Paulus et al., 2017; Wu et al., 2016), i.e., combining the cross-entropy and reward-based losses, which also helps maintain output fluency.



Ground truth: A band is performing a song.
A rock concert performance by a band.
Baseline-XE: A person is playing a video game.
CIDEr-RL: A band is playing a video game.
CIDEnt-RL: A band is performing a song.

Figure 1: A correctly-predicted video caption generated by our CIDEnt-reward model.

Next, we introduce a novel entailment-corrected reward that checks for logically-directed partial matches. Current reinforcement-based text generation works use traditional phrase-matching metrics (e.g., CIDEr, BLEU) as their reward function. However, these metrics use *undirected* n-gram matching of the machine-generated caption with the ground-truth caption, and hence fail to capture its *directed logical correctness*. Therefore, they still give high scores to even those generated captions that contain a single but critical wrong word (e.g., negation, unrelated action or object), because all the other words still match with the ground truth. We introduce CIDEnt, which penalizes the phrase-matching metric (CIDEr) based reward, when the entailment score is low. This ensures that a generated caption gets a high re-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 979–985
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

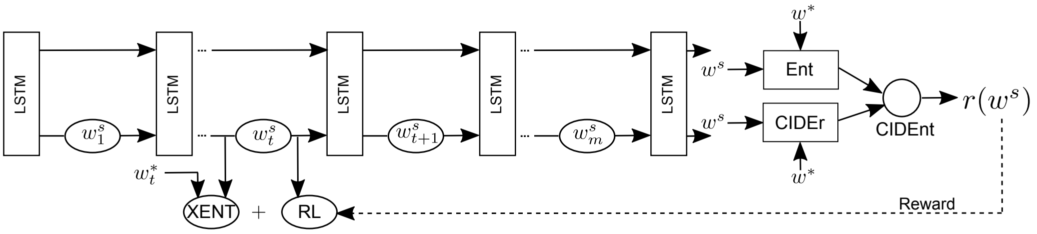


Figure 2: Reinforced (mixed-loss) video captioning using entailment-corrected CIDEr score as reward.

ward only when it is a directed match with (i.e., it is logically implied by) the ground truth caption, hence avoiding contradictory or unrelated information (e.g., see Fig. 1). Empirically, we show that first the CIDEr-reward model achieves significant improvements over the cross-entropy baseline (on multiple datasets, and automatic and human evaluation); next, the CIDEnt-reward model further achieves significant improvements over the CIDEr-based reward. Overall, we achieve the new state-of-the-art on the MSR-VTT dataset.

2 Related Work

Past work has presented several sequence-to-sequence models for video captioning, using attention, hierarchical RNNs, 3D-CNN video features, joint embedding spaces, language fusion, etc., but using word-level cross entropy loss training (Venugopalan et al., 2015a; Yao et al., 2015; Pan et al., 2016a,b; Venugopalan et al., 2016).

Policy gradient for image captioning was recently presented by Ranzato et al. (2016), using a mixed sequence level training paradigm to use non-differentiable evaluation metrics as rewards.[1] Liu et al. (2016b) and Rennie et al. (2016) improve upon this using Monte Carlo roll-outs and a test inference baseline, respectively. Paulus et al. (2017) presented summarization results with ROUGE rewards, in a mixed-loss setup.

Recognizing Textual Entailment (RTE) is a traditional NLP task (Dagan et al., 2006; Lai and Hockenmaier, 2014; Jimenez et al., 2014), boosted by a large dataset (SNLI) recently introduced by Bowman et al. (2015). There have been several leaderboard models on SNLI (Cheng et al., 2016; Rocktäschel et al., 2016); we focus on the decomposable, intra-sentence attention model of Parikh et al. (2016). Recently, Pasunuru and Bansal (2017) used multi-task learning to combine video captioning with entailment and video generation.

3 Models

Attention Baseline (Cross-Entropy) Our attention-based seq-to-seq baseline model is similar to the Bahdanau et al. (2015) architecture, where we encode input frame level video features $\{f_{1:n}\}$ via a bi-directional LSTM-RNN and then generate the caption $w_{1:m}$ using an LSTM-RNN with an attention mechanism. Let θ be the model parameters and $w^*_{1:m}$ be the ground-truth caption, then the cross entropy loss function is:

$$L(\theta) = -\sum_{t=1}^{m} \log p(w^*_t | w^*_{1:t-1}, f_{1:n}) \qquad (1)$$

where $p(w_t|w_{1:t-1}, f_{1:n}) = softmax(W^T h^d_t)$, W^T is the projection matrix, and w_t and h^d_t are the generated word and the RNN decoder hidden state at time step t, computed using the standard RNN recursion and attention-based context vector c_t. Details of the attention model are in the supplementary (due to space constraints).

Reinforcement Learning (Policy Gradient) In order to directly optimize the sentence-level test metrics (as opposed to the cross-entropy loss above), we use a policy gradient p_θ, where θ represent the model parameters. Here, our baseline model acts as an agent and interacts with its environment (video and caption). At each time step, the agent generates a word (action), and the generation of the end-of-sequence token results in a reward r to the agent. Our training objective is to minimize the negative expected reward function:

$$L(\theta) = -\mathbb{E}_{w^s \sim p_\theta}[r(w^s)] \qquad (2)$$

where w^s is the word sequence sampled from the model. Based on the REINFORCE algorithm (Williams, 1992), the gradients of this non-differentiable, reward-based loss function are:

$$\nabla_\theta L(\theta) = -\mathbb{E}_{w^s \sim p_\theta}[r(w^s) \cdot \nabla_\theta \log p_\theta(w^s)] \quad (3)$$

We follow Ranzato et al. (2016) approximating the above gradients via a single sampled word

[1] Several papers have presented the relative comparison of image captioning metrics, and their pros and cons (Vedantam et al., 2015; Anderson et al., 2016; Liu et al., 2016b; Hodosh et al., 2013; Elliott and Keller, 2014).

Ground-truth caption	Generated (sampled) caption	CIDEr	Ent
a man is spreading some butter in a pan	puppies is melting butter on the pan	140.5	0.07
a panda is eating some bamboo	a panda is eating some fried	256.8	0.14
a monkey pulls a dogs tail	a monkey pulls a woman	116.4	0.04
a man is cutting the meat	a man is cutting meat into potato	114.3	0.08
the dog is jumping in the snow	a dog is jumping in cucumbers	126.2	0.03
a man and a woman is swimming in the pool	a man and a whale are swimming in a pool	192.5	0.02

Table 1: Examples of captions sampled during policy gradient and their CIDEr vs Entailment scores.

sequence. We also use a variance-reducing bias (baseline) estimator in the reward function. Their details and the partial derivatives using the chain rule are described in the supplementary.

Mixed Loss During reinforcement learning, optimizing for only the reinforcement loss (with automatic metrics as rewards) doesn't ensure the readability and fluency of the generated caption, and there is also a chance of gaming the metrics without actually improving the quality of the output (Liu et al., 2016a). Hence, for training our reinforcement based policy gradients, we use a mixed loss function, which is a weighted combination of the cross-entropy loss (XE) and the reinforcement learning loss (RL), similar to the previous work (Paulus et al., 2017; Wu et al., 2016). This mixed loss improves results on the metric used as reward through the reinforcement loss (and improves relevance based on our entailment-enhanced rewards) but also ensures better readability and fluency due to the cross-entropy loss (in which the training objective is a conditioned language model, learning to produce fluent captions). Our mixed loss is defined as:

$$L_{\text{MIXED}} = (1-\gamma)L_{\text{XE}} + \gamma L_{\text{RL}} \tag{4}$$

where γ is a tuning parameter used to balance the two losses. For annealing and faster convergence, we start with the optimized cross-entropy loss baseline model, and then move to optimizing the above mixed loss function.[2]

4 Reward Functions

Caption Metric Reward Previous image captioning papers have used traditional captioning metrics such as CIDEr, BLEU, or METEOR as reward functions, based on the match between the generated caption sample and the ground-truth reference(s). First, it has been shown by Vedantam et al. (2015) that CIDEr, based on a consensus measure across several human reference captions, has a higher correlation with human evaluation than other metrics such as METEOR, ROUGE, and BLEU. They further showed that CIDEr gets better with more number of human references (and this is a good fit for our video captioning datasets, which have 20-40 human references per video).

More recently, Rennie et al. (2016) further showed that CIDEr as a reward in image captioning outperforms all other metrics as a reward, not just in terms of improvements on CIDEr metric, but also on all other metrics. In line with these above previous works, we also found that CIDEr as a reward ('CIDEr-RL' model) achieves the best metric improvements in our video captioning task, and also has the best human evaluation improvements (see Sec. 6.3 for result details, incl. those about other rewards based on BLEU, SPICE).

Entailment Corrected Reward Although CIDEr performs better than other metrics as a reward, all these metrics (including CIDEr) are still based on an undirected n-gram matching score between the generated and ground truth captions. For example, the wrong caption "a man is playing football" w.r.t. the correct caption "a man is playing basketball" still gets a high score, even though these two captions belong to two completely different events. Similar issues hold in case of a negation or a wrong action/object in the generated caption (see examples in Table 1).

We address the above issue by using an entailment score to correct the phrase-matching metric (CIDEr or others) when used as a reward, ensuring that the generated caption is logically implied by (i.e., is a paraphrase or directed partial match with) the ground-truth caption. To achieve an accurate entailment score, we adapt the state-of-the-art decomposable-attention model of Parikh et al. (2016) trained on the SNLI corpus (image caption domain). This model gives us a probability for whether the sampled video caption (generated by our model) is entailed by the ground truth caption as premise (as opposed to a contradiction or neu-

[2] We also experimented with the curriculum learning 'MIXER' strategy of Ranzato et al. (2016), where the XE+RL annealing is based on the decoder time-steps; however, the mixed loss function strategy (described above) performed better in terms of maintaining output caption fluency.

tral case).[3] Similar to the traditional metrics, the overall 'Ent' score is the maximum over the entailment scores for a generated caption w.r.t. each reference human caption (around 20/40 per MSR-VTT/YouTube2Text video). CIDEnt is defined as:

$$\text{CIDEnt} = \begin{cases} \text{CIDEr} - \lambda, & \text{if Ent} < \beta \\ \text{CIDEr}, & \text{otherwise} \end{cases} \quad (5)$$

which means that if the entailment score is very low, we penalize the metric reward score by decreasing it by a penalty λ. This agreement-based formulation ensures that we only trust the CIDEr-based reward in cases when the entailment score is also high. Using CIDEr$-\lambda$ also ensures the smoothness of the reward w.r.t. the original CIDEr function (as opposed to clipping the reward to a constant). Here, λ and β are hyperparameters that can be tuned on the dev-set; on light tuning, we found the best values to be intuitive: $\lambda =$ roughly the baseline (cross-entropy) model's score on that metric (e.g., 0.45 for CIDEr on MSR-VTT dataset); and $\beta = 0.33$ (i.e., the 3-class entailment classifier chose contradiction or neutral label for this pair). Table 1 shows some examples of sampled generated captions during our model training, where CIDEr was misleadingly high for incorrect captions, but the low entailment score (probability) helps us successfully identify these cases and penalize the reward.

5 Experimental Setup

Datasets We use 2 datasets: MSR-VTT (Xu et al., 2016) has $10,000$ videos, 20 references/video; and YouTube2Text/MSVD (Chen and Dolan, 2011) has 1970 videos, 40 references/video. Standard splits and other details in supp.

Automatic Evaluation We use several standard automated evaluation metrics: METEOR, BLEU-4, CIDEr-D, and ROUGE-L (from MS-COCO evaluation server (Chen et al., 2015)).

Human Evaluation We also present human evaluation for comparison of baseline-XE, CIDEr-RL, and CIDEnt-RL models, esp. because the automatic metrics cannot be trusted solely. Relevance measures how related is the generated caption w.r.t, to the video content, whereas coherence measures readability of the generated caption.

Training Details All the hyperparameters are tuned on the validation set. All our results (including baseline) are based on a 5-avg-ensemble. See supplementary for extra training details, e.g., about the optimizer, learning rate, RNN size, Mixed-loss, and CIDEnt hyperparameters.

6 Results

6.1 Primary Results

Table 2 shows our primary results on the popular MSR-VTT dataset. First, our baseline attention model trained on cross entropy loss ('Baseline-XE') achieves strong results w.r.t. the previous state-of-the-art methods.[4] Next, our policy gradient based mixed-loss RL model with reward as CIDEr ('CIDEr-RL') improves significantly[5] over the baseline on all metrics, and not just the CIDEr metric. It also achieves statistically significant improvements in terms of human relevance evaluation (see below). Finally, the last row in Table 2 shows results for our novel CIDEnt-reward RL model ('CIDEnt-RL'). This model achieves statistically significant[6] improvements on top of the strong CIDEr-RL model, on all automatic metrics (as well as human evaluation). Note that in Table 2, we also report the CIDEnt reward scores, and the CIDEnt-RL model strongly outperforms CIDEr and baseline models on this entailment-corrected measure. Overall, we are also the new Rank1 on the MSR-VTT leaderboard, based on their ranking criteria.

Human Evaluation We also perform small human evaluation studies (250 samples from the MSR-VTT test set output) to compare our 3 models pairwise.[7] As shown in Table 3 and Table 4, in terms of relevance, first our CIDEr-RL model stat. significantly outperforms the baseline XE model ($p < 0.02$); next, our CIDEnt-RL model significantly outperforms the CIDEr-RL model ($p <$

[3] Our entailment classifier based on Parikh et al. (2016) is 92% accurate on entailment in the caption domain, hence serving as a highly accurate reward score. For other domains in future tasks such as new summarization, we plan to use the new multi-domain dataset by Williams et al. (2017).

[4] We list previous works' results as reported by the MSR-VTT dataset paper itself, as well as their 3 leaderboard winners (http://ms-multimedia-challenge.com/leaderboard), plus the 10-ensemble video+entailment generation multi-task model of Pasunuru and Bansal (2017).

[5] Statistical significance of $p < 0.01$ for CIDEr, METEOR, and ROUGE, and $p < 0.05$ for BLEU, based on the bootstrap test (Noreen, 1989; Efron and Tibshirani, 1994).

[6] Statistical significance of $p < 0.01$ for CIDEr, BLEU, ROUGE, and CIDEnt, and $p < 0.05$ for METEOR.

[7] We randomly shuffle pairs to anonymize model identity and the human evaluator then chooses the better caption based on relevance and coherence (see Sec. 5). 'Not Distinguishable' are cases where the annotator found both captions to be equally good or equally bad).

Models	BLEU-4	METEOR	ROUGE-L	CIDEr-D	CIDEnt	Human*
PREVIOUS WORK						
Venugopalan (2015b)*	32.3	23.4	-	-	-	-
Yao et al. (2015)*	35.2	25.2	-	-	-	-
Xu et al. (2016)	36.6	25.9	-	-	-	-
Pasunuru and Bansal (2017)	**40.8**	**28.8**	60.2	47.1	-	-
Rank1: v2t_navigator	**40.8**	28.2	60.9	44.8	-	-
Rank2: Aalto	39.8	26.9	59.8	45.7	-	-
Rank3: VideoLAB	39.1	27.7	60.6	44.1	-	-
OUR MODELS						
Cross-Entropy (Baseline-XE)	38.6	27.7	59.5	44.6	34.4	-
CIDEr-RL	39.1	28.2	60.9	51.0	37.4	11.6
CIDEnt-RL (**New Rank1**)	**40.5**	**28.4**	**61.4**	**51.7**	**44.0**	**18.4**

Table 2: Our primary video captioning results on MSR-VTT. All CIDEr-RL results are statistically significant over the baseline XE results, and all CIDEnt-RL results are stat. signif. over the CIDEr-RL results. Human* refers to the 'pairwise' comparison of human relevance evaluation between CIDEr-RL and CIDEnt-RL models (see full human evaluations of the 3 models in Table 3 and Table 4).

	Relevance	Coherence
Not Distinguishable	64.8%	92.8%
Baseline-XE Wins	13.6%	4.0%
CIDEr-RL Wins	**21.6%**	3.2%

Table 3: Human eval: Baseline-XE vs CIDEr-RL.

	Relevance	Coherence
Not Distinguishable	70.0%	94.6%
CIDEr-RL Wins	11.6%	2.8%
CIDEnt-RL Wins	**18.4%**	2.8%

Table 4: Human eval: CIDEr-RL vs CIDEnt-RL.

0.03). The models are statistically equal on coherence in both comparisons.

6.2 Other Datasets

We also tried our CIDEr and CIDEnt reward models on the YouTube2Text dataset. In Table 5, we first see strong improvements from our CIDEr-RL model on top of the cross-entropy baseline. Next, the CIDEnt-RL model also shows some improvements over the CIDEr-RL model, e.g., on BLEU and the new entailment-corrected CIDEnt score. It also achieves significant improvements on human relevance evaluation (250 samples).[8]

6.3 Other Metrics as Reward

As discussed in Sec. 4, CIDEr is the most promising metric to use as a reward for captioning, based on both previous work's findings as well as ours. We did investigate the use of other metrics as the reward. When using BLEU as a reward (on MSR-VTT), we found that this BLEU-RL model achieves BLEU-metric improvements, but was worse than the cross-entropy baseline on human evaluation. Similarly, a BLEUEnt-RL model achieves BLEU and BLEUEnt metric improvements, but is again worse on human evaluation.

Models	B	M	R	C	CE	H*
Baseline-XE	52.4	35.0	71.6	83.9	68.1	-
CIDEr-RL	53.3	35.1	72.2	89.4	69.4	8.4
CIDEnt-RL	54.4	34.9	72.2	88.6	71.6	13.6

Table 5: Results on YouTube2Text (MSVD) dataset. CE = CIDEnt score. H* refer to the pairwise human comparison of relevance.

We also experimented with the new SPICE metric (Anderson et al., 2016) as a reward, but this produced long repetitive phrases (as also discussed in Liu et al. (2016b)).

6.4 Analysis

Fig. 1 shows an example where our CIDEnt-reward model correctly generates a ground-truth style caption, whereas the CIDEr-reward model produces a non-entailed caption because this caption will still get a high phrase-matching score. Several more such examples are in the supp.

7 Conclusion

We first presented a mixed-loss policy gradient approach for video captioning, allowing for metric-based optimization. We next presented an entailment-corrected CIDEnt reward that further improves results, achieving the new state-of-the-art on MSR-VTT. In future work, we are applying our entailment-corrected rewards to other directed generation tasks such as image captioning and document summarization (using the new multi-domain NLI corpus (Williams et al., 2017)).

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was supported by a Google Faculty Research Award, an IBM Faculty Award, a Bloomberg Data Science Research Grant, and NVidia GPU awards.

[8] This dataset has a very small dev-set, causing tuning issues – we plan to use a better train/dev re-split in future work.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. SPICE: Semantic propositional image caption evaluation. In *ECCV*, pages 382–398.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

David L Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 190–200. Association for Computational Linguistics.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *EMNLP*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Desmond Elliott and Frank Keller. 2014. Comparing automatic evaluation measures for image description. In *ACL*, pages 452–457.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

Sergio Jimenez, George Duenas, Julia Baquero, Alexander Gelbukh, Av Juan Dios Bátiz, and Av Mendizábal. 2014. UNAL-NLP: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *In SemEval*, pages 732–742.

Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: A denotational and distributional approach to semantics. *Proc. SemEval*, 2:5.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016a. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*.

Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. 2016b. Improved image captioning via policy gradient optimization of SPIDEr. *arXiv preprint arXiv:1612.00370*.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Pingbo Pan, Zhongwen Xu, Yi Yang, Fei Wu, and Yueting Zhuang. 2016a. Hierarchical recurrent neural encoder for video representation with application to captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1029–1038.

Yingwei Pan, Tao Mei, Ting Yao, Houqiang Li, and Yong Rui. 2016b. Jointly modeling embedding and translation to bridge video and language. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4594–4602.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*.

Ramakanth Pasunuru and Mohit Bansal. 2017. Multitask video captioning with video and entailment generation. In *Proceedings of ACL*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. *arXiv preprint arXiv:1612.00563*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *ICLR*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575.

Subhashini Venugopalan, Lisa Anne Hendricks, Raymond Mooney, and Kate Saenko. 2016. Improving lstm-based video description with linguistic knowledge mined from text. In *EMNLP*.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015a. Sequence to sequence-video to text. In *CVPR*, pages 4534–4542.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2015b. Translating videos to natural language using deep recurrent neural networks. In *NAACL HLT*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

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

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. MSR-VTT: A large video description dataset for bridging video and language. In *CVPR*, pages 5288–5296.

Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Describing videos by exploiting temporal structure. In *CVPR*, pages 4507–4515.

Evaluating Hierarchies of Verb Argument Structure with Hierarchical Clustering

Jesse Mu and **Joshua K. Hartshorne**
Department of Psychology
Boston College
Chestnut Hill, MA
{muj,hartshoj}@bc.edu

Timothy J. O'Donnell
Department of Linguistics
McGill University
Montreal, Canada
timothy.odonnell@mcgill.ca

Abstract

Verbs can only be used with a few specific arrangements of their arguments (*syntactic frames*). Most theorists note that verbs can be organized into a hierarchy of *verb classes* based on the frames they admit. Here we show that such a hierarchy is objectively well-supported by the patterns of verbs and frames in English, since a systematic hierarchical clustering algorithm converges on the same structure as the handcrafted taxonomy of VerbNet, a broad-coverage verb lexicon. We also show that the hierarchies capture meaningful psychological dimensions of generalization by predicting novel verb coercions by human participants. We discuss limitations of a simple hierarchical representation and suggest similar approaches for identifying the representations underpinning verb argument structure.

1 Introduction

Why can Sally *like to read* but not **appreciate to read*? Key to the grammar of sentences are verbs and the arguments with which they appear. How children learn the constraints that govern the ways verbs and arguments combine is a central question in language acquisition.

Theorists have long noted that verbs can be organized into *classes* based on their syntactic constructions and the events they express (see Levin and Rappaport Hovav, 2005 for review). Verb classes are included in most theories of argument structure acquisition, whether as first class objects (Perfors et al., 2010) or mere epiphenomena of other claims about the structure of form-meaning mappings (Pinker, 1989; Goldberg, 1995).

Most theories also propose further structure between classes. One common assumption is that verb argument structure can be at least partially described by a hierarchy: Each verb belongs to a class, which itself may belong to a number of broader *superclasses*.

While many theories predict more complex structure (e.g. cross-cutting categories; Levin and Rappaport Hovav, 2005), providing (psycho)linguistic evidence for a simple hierarchy of verbs is an important starting point for investigating more complex theories. VerbNet (Kipper et al., 2008), the largest English verb argument structure resource,[1] organizes verbs and classes into a shallow hierarchy, but its structure has been handcrafted incrementally over time (starting with seminal work by Levin, 1993). On the other hand, recently-developed, state-of-the-art machine learning methods offer a unique alternative approach to constructing such a hierarchy.

In this paper, we first conduct a broad-coverage analysis of how verbs might be hierarchically arranged by comparing VerbNet's handcrafted hierarchy to structure systematically inferred by a Bayesian hierarchical clustering algorithm. We find that the two arrive at similar structure, thus substantiating both methods (i.e. intuition vs. clustering) and the common hierarchy they find.

Second, we investigate the psychological validity of this representation: if classes capture meaningful dimensions of generalization, one would intuit that a verb in a class should behave more similarly to verbs in nearby classes than distant classes according to some measure of "distance". Indeed, this kind of assumption plays an important role

[1] VerbNet combines many important expert-crafted verb resources into a single database, and thus is used in diverse NLP applications including semantic parsing (Giuglea and Moschitti, 2006), natural language inference (Palmer et al., 2009), and information extraction (Maynard et al., 2009).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 986–991
Copenhagen, Denmark, September 7–11, 2017.

in theoretical (Suttle and Goldberg, 2011; Pinker, 1989) and empirical (Ambridge et al., 2011) work. We thus ask human participants to rate the compatibility of a wide range of existing verbs in attested and unattested syntactic frames. We find that such coercions are indeed predicted by a hierarchical taxonomy of verbs.

2 Related work

There is a substantial literature from both the NLP and psycholinguistics communities on unsupervised learning of verb classes from corpora and other resources (e.g. Reichart and Korhonen, 2013; Vlachos et al., 2009; Sun et al., 2008; Joanis and Stevenson, 2003) and computational cognitive models of argument structure acquisition (e.g. Barak et al., 2016; Ambridge and Blything, 2015; Barak et al., 2014; Parisien and Stevenson, 2010; Perfors et al., 2010), respectively.

Our work differs in several ways. First, we do not consider the basic problem of learning verb classes from semantic or syntactic primitives (cf. Sun et al., 2008) or verb usages extracted from corpora; instead, we examine what higher-level structure is implied by the gold-standard catalog of already-clustered verbs and syntactic frames in VerbNet. Second, we do not attempt to model incremental learning (cf. Parisien and Stevenson, 2010) or instantiate a specific theory (cf. Ambridge and Blything, 2015). Rather, we conduct an at-scale investigation of verb argument structure through cluster analysis.

3 Discovering structure via clustering

VerbNet suggests a shallow and disconnected hierarchy of verbs, with lower-level *subclasses* of verbs that take the exact same frames, broader *standard classes*, and at the top, 101 unrelated *superclasses* (Figure 1a). There is a broad assumption of weaker relations between members of higher-level classes than lower-level classes.

We compared this to the hierarchy obtained from Bayesian Hierarchical Clustering (BHC; Heller and Ghahramani, 2005) implemented in R by Savage et al. (2009), a state-of-the-art agglomerative clustering method that can be seen as a bottom-up approximation to a Dirichlet Process Mixture Model. Unlike traditional hierarchical clustering algorithms, BHC uses Bayesian hypothesis testing to merge subtrees: at each proposed merge, BHC evaluates the probability p that the data are generated from a single probabilistic model, rather than two or more different models consistent with the subtrees.[2] Crucially, nodes with probability $p < 0.5$ are merges that BHC prefers *not* to make; the tree can be cut at these nodes to obtain a flat clustering (Figure 1b), which can then be compared to VerbNet.

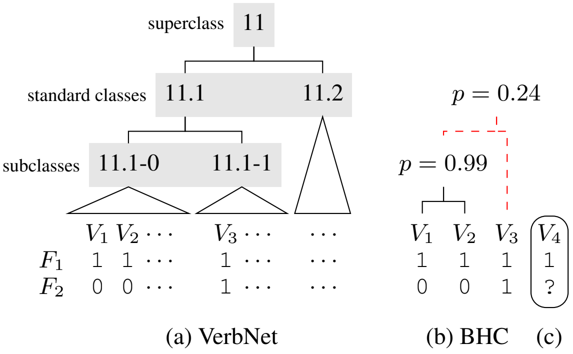


Figure 1: (a) Simplified VerbNet hierarchy, depicting a superclass, standard classes, subclasses, and toy verbs V_i and frames F_i. (b) We train BHC on the frame data $\mathcal{D}$. Dotted lines are merges BHC prefers not to make ($p < 0.5$). To obtain a flat clustering, the tree is cut at nodes where $p < 0.5$ and each subtree is a cluster. (c) Using BHC to evaluate $P(V_4 \text{ admits } F_2 \mid V_4 \text{ admits } F_1, \mathcal{D})$.

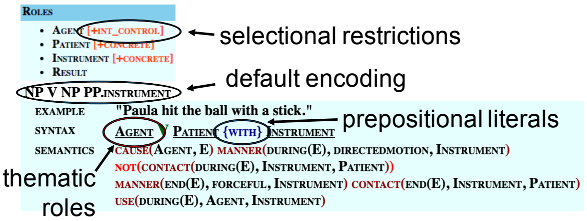
3.1 Data

As input to BHC, we used VerbNet's comprehensive set of verb-frame combinations. VerbNet v3.2[3] can be represented as a 6334 verb × n frame binary matrix, with 1s in cells with attested verb-frame pairs (Figure 1a). Thus, each verb is represented as a binary vector of frames.

The number of frames n depends on what semantic and syntactic annotations are considered to be part of the frame. VerbNet includes 3 kinds of annotations: selectional restrictions on arguments, thematic roles, and prepositional literals (Figure 2). For this paper, we included selectional restrictions and thematic roles, resulting in 1613 frames. These annotations made it easiest to produce experimental stimuli in Section 4, although our analysis produced similar results across the other possible frame encodings (see Appendix A).

[2] In our case, the assumed generative model is a set of independent Beta-Bernoulli models predicting the probability of occurrence of each frame, with the priors found by optimizing the marginal likelihood of the overall model. For full details of the algorithm, see Heller and Ghahramani (2005) and Savage et al. (2009).

[3] verbs.colorado.edu/verb-index



Roles

- Agent [+int_control]
- Patient [+concrete]
- Instrument [+concrete]
- Result

NP V NP PP.instrument

example "Paula hit the ball with a stick."

syntax Agent V Patient {with} Instrument

semantics cause(Agent, E) manner(during(E), directedmotion, Instrument)
not(contact(during(E), Instrument, Patient))
manner(end(E), forceful, Instrument) contact(end(E), Instrument, Patient)
use(during(E), Agent, Instrument)

Figure 2: Information associated with a VerbNet frame entry.

3.2 Evaluation

Here we evaluated the extent to which BHC converged on VerbNet's structure at low (sub and standard classes) and high levels (superclasses).

Comparing flat clusterings with H and C First, we obtained the flat clustering from BHC (Figure 1b) and asked how it compared to VerbNet. Here, we used homogeneity (H) and completeness (C), entropy-based measures of clustering similarity analogous to precision and recall in binary classification (Rosenberg and Hirschberg, 2007). Treating VerbNet classes as ground truth, $H = 1$ indicates that every BHC cluster contains only members of a single VerbNet class. $C = 1$ indicates that members of a VerbNet class are always assigned to the same BHC cluster. The worst case for both is 0.

H and C have different meanings depending on what we consider to be VerbNet's flat ground truth classes. We consider ground truth classes across the levels of VerbNet granularity: low-level subclasses ($H_{\text{sub}}, C_{\text{sub}}$), standard classes ($H_{\text{standard}}, C_{\text{standard}}$), and superclasses ($H_{\text{super}}, C_{\text{super}}$) (Table 1).

Table 1: Homogeneity and completeness. Random baselines are mean statistics across 1000 clusterings made by uniformly sampling a BHC cluster for each verb. C_{sub} is trivially 1, since members of VerbNet subclasses have identical features and were always grouped into the same class by BHC.

Statistic	Granularity	BHC	Random
H	H_{super}	.88	.31
	H_{standard}	.88	.19
	H_{sub}	.83	.34
C	C_{super}	.72	.31
	C_{standard}	.99	.14
	C_{sub}	1	.37

The important comparison is with superclasses, for which both H and C were high. This indicates that BHC clusters rarely included verbs from multiple VerbNet superclasses ($H_{\text{super}} = .88$) and rarely split verbs from the same VerbNet superclass into different BHC clusters ($C_{\text{super}} = .72$).

Tanglegram While H and C focus on the size and membership of two clustering solutions, *tanglegrams* (Huson and Scornavacca, 2012) allow a more general visualization and comparison of two hierarchies. Using the heuristic of Scornavacca et al. (2011),[4] we drew the *optimal tanglegram* of VerbNet and BHC, where the two trees are drawn such that lines connect common leaves and the number of intersections made by these lines is minimized. We computed the *entanglement* of the tanglegram by normalizing the number of intersections to the 0–1 interval by dividing by the worst case; this is a holistic measure of the similarity of the hierarchies (Galili, 2015).

The tanglegram (Figure 3) shows that qualitatively, much of VerbNet's structure aligns well between the trees. We observed an entanglement of 0.20, compared to a random baseline of 0.66.

3.3 Discussion

The high H and C (Table 1) and low entanglement (Figure 3) suggest that both VerbNet's handcrafted hierarchical taxonomy and the one systematically created by BHC converge on similar results. Interestingly, both methods result in a fairly shallow hierarchy with many unrelated subtrees. This suggests that while small clusters of verbs are highly related, the principles governing verb argument structure are relatively narrow and do not generalize across more than a small subset of verbs. Alternatively, it could suggest that a hierarchical taxonomy is too simple to fully capture argument structure patterns.

4 Human coercion judgments

We next evaluated the hierarchies for their ability to account for human generalization. Researchers often test generalization along a specific dimension through extension to novel verbs ("wug tests"; Ambridge et al., 2013; Pinker, 1989). While this works well for studies of specific phenomena, it is difficult to deploy in a large study like ours, where we do not have hypotheses

[4] dendroscope.org

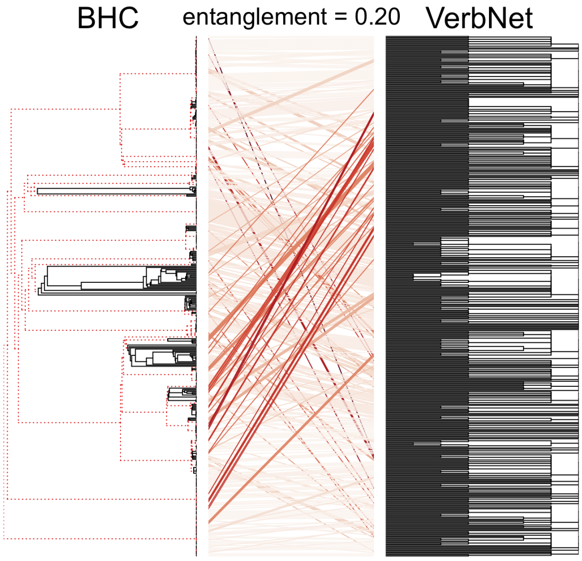


Figure 3: Tanglegram. We prune leaves (verbs) from BHC and VerbNet so that each leaf here represents a VerbNet subclass. Dotted lines are merges BHC prefers not to make. Connectors are colored by shades of red indicating worse alignment, as measured by vertical distance traveled. The vast majority of lines are light-colored, indicating strong alignment.

about what drives generalization language-wide. Thus, we assessed generalization through a coercion task, asking whether speakers are more likely to extend a known verb to a unattested frame if the frame is attested for verbs in a closely-related class. This matches a common theoretical claim that verbs are attracted to the frames of similar verbs, with the notion of similarity varying by theory (Ambridge et al., 2011; Suttle and Goldberg, 2011).

4.1 Predicting verb-frame coercion

VerbNet makes straightforward coarse predictions. For any syntactic frame, we grouped verbs into 3 categories: *Exact*, if the verb can take the frame; *Sibling*, if one of the verb's super or subclasses can take the frame; and *None* otherwise. Conversely, as a Bayesian probabilistic model, BHC defines a predictive distribution on new data. We were interested in whether this precision resulted in better fit, so we also tested BHC: for any verb and frame, we can evaluate the posterior probability that the verb admits the frame of interest while conditioning on the verb's other frames (Figure 1c; see Appendix B for details).

Table 2: 2 sampled frames and their corresponding sentence templates, each with 3 example verbs and predicted compatibilities. To form the stimuli, each verb is placed into the sentence template, e.g. *Beyond the place arose the thing*.

Frame	Sentence	Verb	VN[1]	BHC[2]
BEYOND NP.LOCATION V NP.THEME	Beyond the place V-ed the thing	arise stretch assume	E S N	6.1 −7.6 −9.1
NP.THEME V THROUGH NP.LOCATION	He V-ed through the place	hum motor regard	E S N	6.1 −6.9 −7.3

[1] VerbNet. E = Exact; S = Sibling; N = None
[2] log odds $P(\text{verb takes frame} \mid \text{verb's other frames}, \mathcal{D})$

4.2 Materials and methods

We sampled 10 frames and 10 verbs for each frame, resulting in 100 verb-frame pairs. To control for possible verb frequency effects (Braine and Brooks, 1995), we ensured there was no significant correlation between the predicted compatibility of a verb-frame pair and the Brown corpus (Kučera and Francis, 1967) frequency of the verb ($r = 0.13, p = 0.17$). We then converted verb-frame pairs into sentence stimuli, which required that we choose nouns to represent NPs in frames. We chose the most generic noun compatible with the thematic role restriction, if present. For example, for NP.AGENT, we used a generic name, and for NP.LOCATION, we used *place*. Example stimuli are located in Table 2.

We recruited 50 native English speakers from Mechanical Turk. For each sentence, participants judged the grammaticality of the sentence on a Likert scale, from 1 ("not at all") to 5 ("perfect").

4.3 Results and discussion

First, we noticed that all verbs in some frames received consistently low coercion judgments (< 3). For example, while the verb *fly* and the frame THERE V NP.THEME FOR NP.LOCATION is attested (Exact), *There flew a thing for the place* received a mean judgment of 2.4. We translated judgments so that the mean judgments across verbs for each frame was average (3), to examine the *relative* effects of coercing verbs into frames.

Figure 4a shows that VerbNet's 3 categories predict differences in the mean coercion ratings of verb-frame pairs ($F = 43.46,\ p < 0.001$). Notably, there was a significant difference between the means of the unattested categories (Sibling vs.

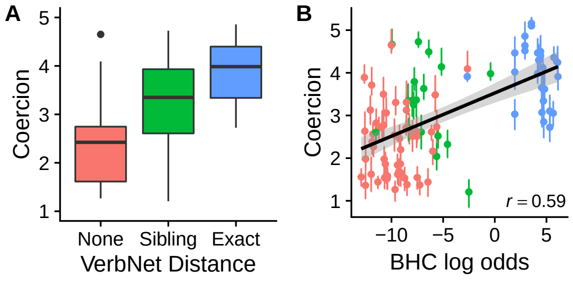


Figure 4: (a) distribution of mean coercion judgments for verb-frame pairs in the 3 VerbNet categories. (b) correlation between the same mean coercion judgments and BHC posterior predictive predictions, colored by VerbNet category. Error bars are bootstrapped 95% confidence intervals.

None; $t = 3.55,\ p < 0.01$). While there was a high correlation between the judgments and BHC predictions (Figure 4b; $r = 0.59$), BHC's hierarchy did not significantly improve fit to the data.

These results provide additional psychological evidence for the effects associated with VerbNet's coarse distinctions: for unattested verb-frame pairs, participants tend to assign a higher compatibility rating when the verb has sibling VerbNet classes that can take the frame. However, the range of compatibility judgments is highly variable across all three categories, and BHC's finer-grained predictions fail to account for much of this variability. Given the similarity of BHC to VerbNet's hierarchy, this result is unsurprising.

5 General discussion

We presented converging evidence that a shallow hierarchy of verbs (1) is well supported by the distribution of verbs and syntactic frames in language, since VerbNet's hand-crafted hierarchy and a systematic unsupervised learner (BHC) reach similar results; and (2) captures important features of verb argument structure by predicting human generalization intuitions in a coercion task.

Of course, it is clear from the variability of our coercion data that a simple hierarchy is not a sufficiently sophisticated representation of argument structure to fully explain language-wide coercion. However, our novel computational framework (unsupervised learning on VerbNet data) opens up many potentially fruitful avenues for providing language-wide evidence for argument structure hypotheses. The lack of broad-coverage predictions is often a limitation of work in this area (see Section 2).

Sophisticated machine learning models that make the assumptions proposed by richer theories of argument structure and can operate at VerbNet scale are only recently coming into fruition. For example, since some theories argue for a cross-categorization of verbs and argument structures (Levin and Rappaport Hovav, 2005), using models that find such a (possibly hierarchical) cross-categorization (e.g. Mansinghka et al., 2016; Li and Shafto, 2011) is a particularly interesting avenue for further exploration.

Acknowledgments

We thank anonymous reviewers for helpful comments and suggestions. JM is supported by an undergraduate research fellowship from the Boston College Psychology Department and a Barry M. Goldwater scholarship.

References

Ben Ambridge and Ryan P Blything. 2015. A connectionist model of the retreat from verb argument structure overgeneralization. *Journal of Child Language*, 43(6):1245–1276.

Ben Ambridge, Julian M Pine, and Caroline F Rowland. 2011. Children use verb semantics to retreat from overgeneralization errors: A novel verb grammaticality judgment study. *Cognitive Linguistics*, 22(2).

Ben Ambridge, Julian M Pine, Caroline F Rowland, Franklin Chang, and Amy Bidgood. 2013. The retreat from overgeneralization in child language acquisition: Word learning, morphology, and verb argument structure. *Wiley Interdisciplinary Reviews: Cognitive Science*, 4(1):47–62.

Libby Barak, Afsaneh Fazly, and Suzanne Stevenson. 2014. Learning verb classes in an incremental model. In *Proceedings of the Fifth Workshop on Cognitive Modeling and Computational Linguistics*, pages 37–45.

Libby Barak, Adele E. Goldberg, and Suzanne Stevenson. 2016. Comparing computational cognitive models of generalization in a language acquisition task. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 96–106.

Martin D S Braine and Patricia J Brooks. 1995. Verb argument structure and the problem of avoiding an overgeneral grammar. In Michael Tomasello and William E C Merriman, editors, *Beyond names for things: Young children's acquisition of verbs*, pages 352–376. Erlbaum, Hillsdale, NJ.

Tal Galili. 2015. dendextend: an R package for visualizing, adjusting, and comparing trees of hierarchical clustering. *Bioinformatics*, 31(22):3718–3720.

Ana-Maria Giuglea and Alessandro Moschitti. 2006. Semantic role labeling via FrameNet, VerbNet and PropBank. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 929–936.

Adele E Goldberg. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. University of Chicago Press, Cambridge, MA.

Katherine A Heller and Zoubin Ghahramani. 2005. Bayesian hierarchical clustering. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 297–304.

Daniel H Huson and Celine Scornavacca. 2012. Dendroscope 3: an interactive tool for rooted phylogenetic trees and networks. *Systematic Biology*, 61(6):1061–1067.

Eric Joanis and Suzanne Stevenson. 2003. A general feature space for automatic verb classification. In *Proceedings of the Tenth Conference of the European Chapter of the Association for Computational Linguistics*, pages 163–170.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of English verbs. *Language Resources and Evaluation*, 42(1):21–40.

Henry Kučera and Winthrop Nelson Francis. 1967. *Computational analysis of present-day American English*. Brown University Press, Providence, RI.

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University Of Chicago Press, Chicago, IL.

Beth Levin and Malka Rappaport Hovav. 2005. *Argument realization: Research surveys in linguistics*. Cambridge University Press.

Dazhuo Li and Patrick Shafto. 2011. Bayesian hierarchical cross-clustering. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 443–451.

Vikash Mansinghka, Patrick Shafto, Eric Jonas, Cap Petschulat, Max Gasner, and Joshua B. Tenenbaum. 2016. CrossCat: A fully Bayesian nonparametric method for analyzing heterogeneous, high dimensional data. *Journal of Machine Learning Research*, 17(138):1–49.

Diana Maynard, Adam Funk, and Wim Peters. 2009. Using lexico-syntactic ontology design patterns for ontology creation and population. In *Proceedings of the 2009 International Conference on Ontology Patterns*, pages 39–52.

Martha Palmer, Jena D Hwang, Susan Windisch Brown, Karin Kipper Schuler, and Arrick Lanfranchi. 2009. Leveraging lexical resources for the detection of event relations. In *AAAI Spring Symposium: Learning by Reading and Learning to Read*, pages 81–87.

Christopher Parisien and Suzanne Stevenson. 2010. Learning verb alternations in a usage-based Bayesian model. *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*, pages 2674–2679.

Amy Perfors, Joshua B Tenenbaum, and Elizabeth Wonnacott. 2010. Variability, negative evidence, and the acquisition of verb argument constructions. *Journal of Child Language*, 37(3):607–642.

Steven Pinker. 1989. *Learnability and Cognition*. The MIT Press, Cambridge, MA.

Roi Reichart and Anna Korhonen. 2013. Improved lexical acquisition through DPP-based verb clustering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 862–872.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420.

Richard S Savage, Katherine Heller, Yang Xu, Zoubin Ghahramani, William M Truman, Murray Grant, Katherine J Denby, and David L Wild. 2009. R/BHC: fast Bayesian hierarchical clustering for microarray data. *BMC Bioinformatics*, 10(1):1.

Celine Scornavacca, Franziska Zickmann, and Daniel H Huson. 2011. Tanglegrams for rooted phylogenetic trees and networks. *Bioinformatics*, 27(13):i248–i256.

Lin Sun, Anna Korhonen, and Yuval Krymolowski. 2008. Automatic classification of english verbs using rich syntactic features. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Laura Suttle and Adele E Goldberg. 2011. The partial productivity of constructions as induction. *Linguistics*, 49(6):1237–1269.

Andreas Vlachos, Anna Korhonen, and Zoubin Ghahramani. 2009. Unsupervised and constrained Dirichlet Process Mixture Models for verb clustering. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 74–82.

Incorporating Global Visual Features into Attention-Based Neural Machine Translation

Iacer Calixto
ADAPT Centre
Dublin City University
School of Computing
Glasnevin, Dublin 9
iacer.calixto@adaptcentre.ie

Qun Liu
ADAPT Centre
Dublin City University
School of Computing
Glasnevin, Dublin 9
qun.liu@adaptcentre.ie

Abstract

We introduce multi-modal, attention-based Neural Machine Translation (NMT) models which incorporate visual features into different parts of both the encoder and the decoder. Global image features are extracted using a pre-trained convolutional neural network and are incorporated *(i)* as words in the source sentence, *(ii)* to initialise the encoder hidden state, and *(iii)* as additional data to initialise the decoder hidden state. In our experiments, we evaluate translations into English and German, how different strategies to incorporate global image features compare and which ones perform best. We also study the impact that adding synthetic multi-modal, multilingual data brings and find that the additional data have a positive impact on multi-modal models. We report new state-of-the-art results and our best models also significantly improve on a comparable Phrase-Based Statistical MT (PBSMT) model trained on the Multi30k data set according to all metrics evaluated. To the best of our knowledge, it is the first time a purely neural model significantly improves over a PBSMT model on all metrics evaluated on this data set.

1 Introduction

Neural Machine Translation (NMT) has recently been proposed as an instantiation of the *sequence to sequence* (*seq2seq*) learning problem (Kalchbrenner and Blunsom, 2013; Cho et al., 2014b; Sutskever et al., 2014). In this problem, each training example consists of one source and one target variable-length sequence, with no prior information regarding the alignments between the two. A model is trained to *translate* sequences in the source language into corresponding sequences in the target. This framework has been successfully used in many different tasks, such as handwritten text generation (Graves, 2013), image description generation (Hodosh et al., 2013; Kiros et al., 2014; Mao et al., 2014; Elliott et al., 2015; Karpathy and Fei-Fei, 2015; Vinyals et al., 2015), machine translation (Cho et al., 2014b; Sutskever et al., 2014) and video description generation (Donahue et al., 2015; Venugopalan et al., 2015).

Recently, there has been an increase in the number of natural language generation models that explicitly use *attention-based decoders*, i.e. decoders that model an *intra-sequential* mapping between source and target representations. For instance, Xu et al. (2015) proposed an attention-based model for the task of Image Description Generation (IDG) where the model learns to *attend to* specific parts of an image (the source) as it generates its description (the target). In MT, one can intuitively interpret this attention mechanism as inducing an *alignment* between source and target sentences, as first proposed by Bahdanau et al. (2015). The common idea is to explicitly frame a learning task in which the decoder learns to attend to the relevant parts of the source sequence when generating each part of the target sequence.

We are inspired by recent successes in using attention-based models in both IDG and NMT. Our main goal in this work is to propose end-to-end multi-modal NMT models which effectively incorporate visual features in different parts of the attention-based NMT framework. The main contributions of our work are:

- We propose novel attention-based multi-modal NMT models which incorporate visual features into the encoder and the decoder.
- We discuss the impact that adding synthetic

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 992–1003
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

multi-modal and multilingual data brings to multi-modal NMT.

- We show that images bring useful information to an NMT model and report state-of-the-art results.

One additional contribution of our work is that we corroborate previous findings by Vinyals et al. (2015) that suggested that using image features directly as additional context to update the hidden state of the decoder (at each time step) prevents learning.

The remainder of this paper is structured as follows. In §1.1 we briefly discuss relevant previous related work. We then revise the attention-based NMT framework and further expand it into different multi-modal NMT models (§2). In §3 we introduce the data sets we use in our experiments. In §4 we detail the hyperparameters, parameter initialisation and other relevant details of our models. Finally, in §6 we draw conclusions and provide some avenues for future work.

1.1 Related work

Attention-based encoder-decoder models for MT have been actively investigated in recent years. Some researchers have studied how to improve attention mechanisms (Luong et al., 2015; Tu et al., 2016) and how to train attention-based models to translate between many languages (Dong et al., 2015; Firat et al., 2016).

However, multi-modal MT has only recently been addressed by the MT community in the form of a shared task (Specia et al., 2016). We note that in the official results of this first shared task no submissions based on a purely neural architecture could improve on the Phrase-Based SMT (PBSMT) baseline. Nevertheless, researchers have proposed to include global visual features in reranking n-best lists generated by a PBSMT system or directly in a purely NMT framework with some success (Caglayan et al., 2016; Calixto et al., 2016; Libovický et al., 2016; Shah et al., 2016). The best results achieved by a purely NMT model in this shared task are those of Huang et al. (2016), who proposed to use global and regional image features extracted with the VGG19 (Simonyan and Zisserman, 2014) and the RCNN (Girshick et al., 2014) convolutional neural networks (CNNs).

Similarly to one of the three models we propose,[1] Huang et al. (2016) extract global features for an image, project these features into the vector space of the source words and then add it as a word in the input sequence. Their best model improves over a strong NMT baseline and is comparable to results obtained with a PBSMT model trained on the same data, although not significantly better. For that reason, their models are used as baselines in our experiments. Next, we point out some key differences between the work of Huang et al. (2016) and ours.

Architecture Their implementation is based on the attention-based model of Luong et al. (2015), which has some differences to that of Bahdanau et al. (2015), used in our work (§2.1). Their encoder is a single-layer unidirectional LSTM and they use the last hidden state of the encoder to initialise the decoder's hidden state, therefore indirectly using the image features to do so. We use a bi-directional recurrent neural network (RNN) with GRU (Cho et al., 2014a) as our encoder, better encoding the semantics of the source sentence.

Image features We include image features separately either as a word in the source sentence (§2.2.1) or *directly* for encoder (§2.2.2) or decoder initialisation (§2.2.3), whereas Huang et al. (2016) only use it as a word. We also show it is better to include an image exclusively for the encoder *or* the decoder initialisation (Tables 1 and 2).

Data Huang et al. (2016) use object detections obtained with the RCNN of Girshick et al. (2014) as additional data, whereas we study the impact that additional back-translated data brings.

Performance All our models outperform Huang et al. (2016)'s according to all metrics evaluated, even when they use additional object detections. If we use additional back-translated data, the difference becomes even larger.

2 Attention-based NMT

In this section, we briefly revise the attention-based NMT framework (§2.1) and expand it into a multi-modal NMT framework (§2.2).

2.1 Text-only attention-based NMT

We follow the notation of Bahdanau et al. (2015) and Firat et al. (2016) throughout this section.

[1] This idea has been developed independently by both research groups.

Given a source sequence $X = (x_1, x_2, \cdots, x_N)$ and its translation $Y = (y_1, y_2, \cdots, y_M)$, an NMT model aims at building a single neural network that translates X into Y by directly learning to model $p(Y \mid X)$. Each x_i is a row index in a source lookup matrix $\boldsymbol{W}_x \in \mathbb{R}^{|V_x| \times d_x}$ (the *source word embeddings matrix*) and each y_j is an index in a target lookup matrix $\boldsymbol{W}_y \in \mathbb{R}^{|V_y| \times d_y}$ (the *target word embeddings matrix*). V_x and V_y are source and target vocabularies and d_x and d_y are source and target word embeddings dimensionalities, respectively.

A bidirectional RNN with GRU is used as the encoder. A forward RNN $\overrightarrow{\Phi}_{\text{enc}}$ reads X word by word, from left to right, and generates a sequence of *forward annotation vectors* $(\overrightarrow{\boldsymbol{h}}_1, \overrightarrow{\boldsymbol{h}}_2, \cdots, \overrightarrow{\boldsymbol{h}}_N)$ at each encoder time step $i \in [1, N]$. Similarly, a backward RNN $\overleftarrow{\Phi}_{\text{enc}}$ reads X from right to left, word by word, and generates a sequence of *backward annotation vectors* $(\overleftarrow{\boldsymbol{h}}_1, \overleftarrow{\boldsymbol{h}}_2, \cdots, \overleftarrow{\boldsymbol{h}}_N)$, as in (1):

$$\begin{aligned} \overrightarrow{\boldsymbol{h}}_i &= \overrightarrow{\Phi}_{\text{enc}}\big(\boldsymbol{W}_x[x_i], \overrightarrow{\boldsymbol{h}}_{i-1}\big), \\ \overleftarrow{\boldsymbol{h}}_i &= \overleftarrow{\Phi}_{\text{enc}}\big(\boldsymbol{W}_x[x_i], \overleftarrow{\boldsymbol{h}}_{i+1}\big). \end{aligned} \tag{1}$$

The final annotation vector for a given time step i is the concatenation of forward and backward vectors $\boldsymbol{h}_i = \big[\overrightarrow{\boldsymbol{h}}_i; \overleftarrow{\boldsymbol{h}}_i\big]$.

In other words, each source sequence X is encoded into a sequence of annotation vectors $h = (\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_N)$, which are in turn used by the decoder: essentially a neural language model (LM) (Bengio et al., 2003) conditioned on the previously emitted words and the source sentence via an attention mechanism.

At each time step t of the decoder, we compute a *time-dependent* context vector $\boldsymbol{c}_t$ based on the annotation vectors h, the decoder's previous hidden state $\boldsymbol{s}_{t-1}$ and the target word $\tilde{y}_{t-1}$ emitted by the decoder in the previous time step.[2]

We follow Bahdanau et al. (2015) and use a single-layer feed-forward network to compute an *expected alignment* $e_{t,i}$ between each source annotation vector $\boldsymbol{h}_i$ and the target word to be emitted at the current time step t, as in (2):

$$e_{t,i} = \boldsymbol{v}_a{}^T \tanh(\boldsymbol{U}_a \boldsymbol{s}_{t-1} + \boldsymbol{W}_a \boldsymbol{h}_i). \tag{2}$$

In Equation (3), these expected alignments are normalised and converted into probabilities:

$$\alpha_{t,i} = \frac{\exp\left(e_{t,i}\right)}{\sum_{j=1}^{N} \exp\left(e_{t,j}\right)}, \tag{3}$$

where $\alpha_{t,i}$ are called the model's *attention weights*, which are in turn used in computing the time-dependent context vector $\boldsymbol{c}_t = \sum_{i=1}^{N} \alpha_{t,i} \boldsymbol{h}_i$. Finally, the context vector $\boldsymbol{c}_t$ is used in computing the decoder's hidden state $\boldsymbol{s}_t$ for the current time step t, as shown in Equation (4):

$$\boldsymbol{s}_t = \Phi_{\text{dec}}(\boldsymbol{s}_{t-1}, \boldsymbol{W}_{\boldsymbol{y}}[\tilde{y}_{t-1}], \boldsymbol{c}_t), \tag{4}$$

where $\boldsymbol{s}_{t-1}$ is the decoder's previous hidden state, $\boldsymbol{W}_{\boldsymbol{y}}[\tilde{y}_{t-1}]$ is the embedding of the word emitted in the previous time step, and $\boldsymbol{c}_t$ is the updated time-dependent context vector.

We use a single-layer feed-forward neural network to initialise the decoder's hidden state $\boldsymbol{s}_0$ at time step $t = 0$ and feed it the concatenation of the last hidden states of the encoder's forward RNN ($\overrightarrow{\Phi}_{\text{enc}}$) and backward RNN ($\overleftarrow{\Phi}_{\text{enc}}$), as in (5):

$$\boldsymbol{s}_0 = \tanh\big(\boldsymbol{W}_{di}[\overleftarrow{\boldsymbol{h}}_1; \overrightarrow{\boldsymbol{h}}_N] + \boldsymbol{b}_{di}\big), \tag{5}$$

where $\boldsymbol{W}_{di}$ and $\boldsymbol{b}_{di}$ are model parameters. Since RNNs normally better store information about recent inputs in comparison to more distant ones (Hochreiter and Schmidhuber, 1997; Bahdanau et al., 2015), we expect to initialise the decoder's hidden state with a strong source sentence representation, i.e. a representation with a strong focus on both the first and the last tokens in the source sentence.

2.2 Multi-modal NMT (MNMT)

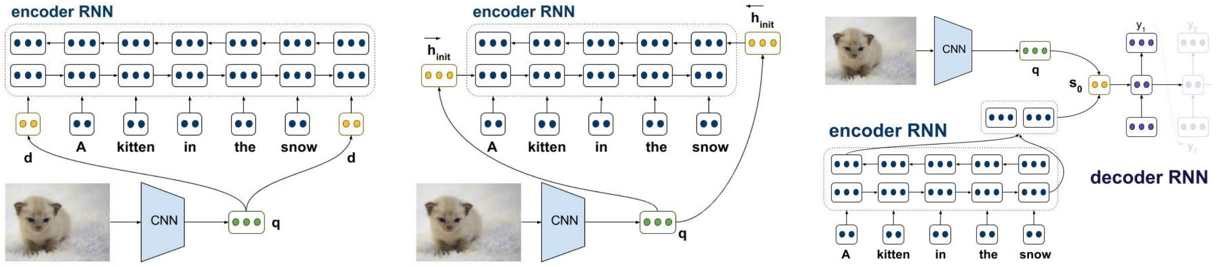
Our models can be seen as expansions of the attention-based NMT framework described in §2 with the addition of a *visual component* to incorporate image features.

Simonyan and Zisserman (2014) trained and evaluated an extensive set of deep Convolutional Neural Networks (CNNs) for classifying images into one out of the 1000 classes in ImageNet (Russakovsky et al., 2015). We use their 19-layer VGG network (VGG19) to extract image feature vectors for all images in our dataset. We feed an image to the pre-trained VGG19 network and use the 4096D activations of the penultimate fully-connected layer FC7[3] as our *image feature vector*, henceforth referred to as $\boldsymbol{q}$.

We propose three different methods to incorporate images into the attentive NMT framework:

[2] At training time, the correct previous target word y_{t-1} is known and therefore used instead of $\tilde{y}_{t-1}$. At test or inference time, y_{t-1} is not known and $\tilde{y}_{t-1}$ is used instead. Bengio et al. (2015) discussed problems that may arise from this difference between training and inference distributions.

[3] We use the activations of the FC7 layer, which encode information about the entire image, of the VGG19 network (configuration E) in Simonyan and Zisserman (2014)'s paper.



(a) An encoder bidirectional RNN that uses image features as words in the source sequence.

(b) Using an image to initialise the encoder hidden states.

(c) Image as additional data to initialise the decoder hidden state s_0.

Figure 1: Multi-modal neural machine translation models IMG_{W}, IMG_{E}, and IMG_{D}.

using an image as words in the source sentence (§2.2.1), using an image to initialise the source language encoder (§2.2.2) and the target language decoder (§2.2.3).

We also evaluated a fourth mechanism to incorporate images into NMT, namely to use an image as one of the different contexts available to the decoder at each time step of the decoding process. We added the image features directly as an additional context, in addition to $\boldsymbol{W}_y[\tilde{y}_{t-1}]$, $\boldsymbol{s}_{t-1}$ and $\boldsymbol{c}_t$, to compute the hidden state $\boldsymbol{s}_t$ of the decoder at a given time step t. We corroborate previous findings by Vinyals et al. (2015) in that adding the image features as such prevents the model from learning.[4]

2.2.1 Images as source words: IMG_{W}

One way we propose to incorporate images into the encoder is to project an image feature vector into the space of the words of the source sentence. We use the projected image as the first and/or last word of the source sentence and let the attention model learn when to attend to the image representation. Specifically, given the global image feature vector $\boldsymbol{q} \in \mathbb{R}^{4096}$, we compute (6):

$$\boldsymbol{d} = \boldsymbol{W}_I^2 \cdot (\boldsymbol{W}_I^1 \cdot \boldsymbol{q} + \boldsymbol{b}_I^1) + \boldsymbol{b}_I^2, \qquad (6)$$

where $\boldsymbol{W}_I^1 \in \mathbb{R}^{4096 \times 4096}$ and $\boldsymbol{W}_I^2 \in \mathbb{R}^{4096 \times d_x}$ are image transformation matrices, $\boldsymbol{b}_I^1 \in \mathbb{R}^{4096}$ and $\boldsymbol{b}_I^2 \in \mathbb{R}^{d_x}$ are bias vectors, and d_x is the source words vector space dimensionality, all trained with the model. We then directly use $\boldsymbol{d}$ as words in the source words vector space: as the first word only (model IMG_{1W}), and as the first and last words of the source sentence (model IMG_{2W}).

An illustration of this idea is given in Figure 1a, where a source sentence that originally contained N tokens, after including the image as source words will contain $N + 1$ tokens (model IMG_{1W}) or $N + 2$ tokens (model IMG_{2W}). In model IMG_{1W}, the image is projected as the first source word only (solid line in Figure 1a); in model IMG_{2W}, it is projected into the source words space as both first and last words (both solid and dashed lines in Figure 1a).

Given a sequence $X = (x_1, x_2, \cdots, x_N)$ in the source language, we concatenate the transformed image vector $\boldsymbol{d}$ to $\boldsymbol{W}_x[X]$ and apply the forward and backward encoder RNN passes, generating hidden vectors as in Figure 1a. When computing the context vector $\boldsymbol{c}_t$ (Equations (2) and (3)), we effectively make use of the transformed image vector, i.e. the $\alpha_{t,i}$ attention weight parameters will use this information to attend or not to the image features.

By including images into the encoder in models IMG_{1W} and IMG_{2W}, our intuition is that *(i)* by including the image as the *first word*, we propagate image features into the source sentence vector representations when applying the forward RNN $\overrightarrow{\Phi}_{\text{enc}}$ (vectors $\overrightarrow{\boldsymbol{h}_i}$), and *(ii)* by including the image as the *last word*, we propagate image features into the source sentence vector representations when applying the backward RNN $\overleftarrow{\Phi}_{\text{enc}}$ (vectors $\overleftarrow{\boldsymbol{h}_i}$).

2.2.2 Images for encoder initialisation: IMG_{E}

In the original attention-based NMT model described in §2, the hidden state of the encoder is initialised with the zero vector $\overrightarrow{0}$. Instead, we propose to use two new single-layer feed-forward neural networks to compute the initial states of the forward RNN $\overrightarrow{\Phi}_{\text{enc}}$ and the backward RNN $\overleftarrow{\Phi}_{\text{enc}}$,

[4] Outputs would typically consist of sets of 2-5 words repeated many times, usually without any syntax. For comparison, translations for the translated Multi30k test set (described in §3) achieve just 3.8 BLEU (Papineni et al., 2002), 15.5 METEOR (Denkowski and Lavie, 2014) and 93.0 TER (Snover et al., 2006).

respectively, as illustrated in Figure 1b.

Similarly to §2.2.1, given a global image feature vector $\boldsymbol{q} \in \mathbb{R}^{4096}$, we compute a vector $\boldsymbol{d}$ using Equation (6), only this time the parameters $\boldsymbol{W}_I^2$ and $\boldsymbol{b}_I^2$ project the image features into the same dimensionality as the textual encoder hidden states.

The feed-forward networks used to initialise the encoder hidden state are computed as in (7):

$$\overleftarrow{\boldsymbol{h}}_{\text{init}} = \tanh\left(\boldsymbol{W}_f \boldsymbol{d} + \boldsymbol{b}_f\right),$$
$$\overrightarrow{\boldsymbol{h}}_{\text{init}} = \tanh\left(\boldsymbol{W}_b \boldsymbol{d} + \boldsymbol{b}_b\right), \qquad (7)$$

where $\boldsymbol{W}_f$ and $\boldsymbol{W}_b$ are multi-modal projection matrices that project the image features $\boldsymbol{d}$ into the encoder forward and backward hidden states dimensionality, respectively, and $\boldsymbol{b}_f$ and $\boldsymbol{b}_b$ are bias vectors.

2.2.3 Images for decoder initialisation: IMG$_\text{D}$

To incorporate an image into the decoder, we introduce a new single-layer feed-forward neural network to be used instead of the one described in Equation 5. Originally, the decoder's initial hidden state was computed using the concatenation of the last hidden states of the encoder forward RNN ($\overrightarrow{\Phi}_{\text{enc}}$) and backward RNN ($\overleftarrow{\Phi}_{\text{enc}}$), respectively $\overrightarrow{\boldsymbol{h}}_N$ and $\overleftarrow{\boldsymbol{h}}_1$.

Our proposal is that we include the image features as additional input to initialise the decoder hidden state at time step $t = 0$, as in (8):

$$\boldsymbol{s}_0 = \tanh\left(\boldsymbol{W}_{di}[\overleftarrow{\boldsymbol{h}}_1; \overrightarrow{\boldsymbol{h}}_N] + \boldsymbol{W}_m \boldsymbol{d} + \boldsymbol{b}_{di}\right), \quad (8)$$

where $\boldsymbol{W}_m$ is a multi-modal projection matrix that projects the image features $\boldsymbol{d}$ into the decoder hidden state dimensionality and $\boldsymbol{W}_{di}$ and $\boldsymbol{b}_{di}$ are the same as in Equation (5).

Once again we compute $\boldsymbol{d}$ by applying Equation (6) onto a global image feature vector $\boldsymbol{q} \in \mathbb{R}^{4096}$, only this time the parameters $\boldsymbol{W}_I^2$ and $\boldsymbol{b}_I^2$ project the image features into the same dimensionality as the decoder hidden states. We illustrate this idea in Figure 1c.

3 Data set

Our multi-modal NMT models need bilingual sentences accompanied by one or more images as training data. The original Flickr30k data set contains 30K images and 5 English sentence descriptions for each image (Young et al., 2014). We use the translated and the comparable Multi30k datasets (Elliott et al., 2016), henceforth referred to as M30k$_\text{T}$ and M30k$_\text{C}$, respectively, which are multilingual expansions of the original Flickr30k.

For each of the 30K images in the Flickr30k, the M30k$_\text{T}$ has one of its English descriptions manually translated into German by a professional translator. Training, validation and test sets contain 29K, 1014 and 1K images, respectively, each accompanied by one sentence pair (the original English sentence and its German translation). For each of the 30K images in the Flickr30k, the M30k$_\text{C}$ has five descriptions in German collected independently of the English descriptions. Training, validation and test sets contain 29K, 1014 and 1K images, respectively, each accompanied by 5 English and 5 German sentences.

We use the scripts in Moses (Koehn et al., 2007) to normalise, truecase and tokenize English and German descriptions and we also convert space-separated tokens into subwords (Sennrich et al., 2016b). All models use a common vocabulary of ∼83K English and ∼91K German subword tokens. If sentences in English or German are longer than 80 tokens, they are discarded.

We use the entire M30k$_\text{T}$ training set for training, its validation set for model selection with BLEU, and its test set to evaluate our models. In order to study the impact that additional training data brings to the models, we use the baseline model described in §2 trained on the textual part of the M30k$_\text{T}$ data set (German→English and English→German) without the images to build back-translation models (Sennrich et al., 2016a). We back-translate the 145K German (English) descriptions in the M30k$_\text{C}$ into English (German) and include the triples (synthetic English description, German description, image) as additional training data when translating into German, and (synthetic German description, English description, image) when translating into English.

We train models to translate from English into German and from German into English, and report evaluation of cased, tokenized sentences with punctuation.

4 Experimental setup

Our encoder is a bidirectional RNN with GRU (one 1024D single-layer forward RNN and one 1024D single-layer backward RNN). Source and target word embeddings are 620D each and both are trained jointly with our model. All non-recurrent matrices are initialised by sampling from a Gaussian ($\mu = 0, \sigma = 0.01$), recurrent matrices are orthogonal and bias vectors are all initialised to zero. Our decoder RNN also uses GRU and is a neural LM (Bengio et al., 2003) conditioned on its previous emissions and the source sentence by means of the source attention mechanism.

Image features are obtained by feeding images to the pre-trained VGG19 network of Simonyan and Zisserman (2014) and using the activations of the penultimate fully-connected layer FC7. We apply dropout with a probability of 0.2 in both source and target word embeddings and with a probability of 0.5 in the image features (in all MNMT models), in the encoder and decoder RNNs inputs and recurrent connections, and before the readout operation in the decoder RNN. We follow Gal and Ghahramani (2016) and apply dropout to the encoder bidirectional RNN and decoder RNN using the same mask in all time steps.

Our models are trained using stochastic gradient descent with Adadelta (Zeiler, 2012) and minibatches of size 40 for improved generalisation (Keskar et al., 2017), where each training instance consists of one English sentence, one German sentence and one image. We apply early stopping for model selection based on BLEU scores, so that if a model does not improve on the validation set for more than 20 epochs, training is halted.

We evaluate our models' translation quality quantitatively in terms of BLEU4 (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), TER (Snover et al., 2006), and chrF3 scores[5] (Popović, 2015) and we report statistical significance for the three first metrics using approximate randomisation computed with MultEval (Clark et al., 2011).

	BLEU4↑	METEOR↑	TER↓	chrF3↑
English→German				
PBSMT	32.9	$\underline{54.1}$	$\underline{45.1}$	$\underline{67.4}$
NMT	$\underline{33.7}$	52.3	46.7	64.5
Huang	35.1	52.2	—	—
+ RCNN	36.5	54.1	—	—
IMG_{1W}	$37.1^{\dagger\ddagger}$ (↑ 3.4)	$54.5^{\ddagger}$ (↑ 0.4)	$42.7^{\dagger\ddagger}$ (↓ 2.4)	66.9 (↓ 0.5)
IMG_{2W}	$36.9^{\dagger\ddagger}$ (↑ 3.2)	$54.3^{\ddagger}$ (↑ 0.2)	**41.9**†‡ (↓ 3.2)	66.8 (↓ 0.6)
IMG_{E}	$37.1^{\dagger\ddagger}$ (↑ 3.4)	$55.0^{\dagger\ddagger}$ (↑ 0.9)	$43.1^{\dagger\ddagger}$ (↓ 2.0)	67.6 (↑ 0.2)
IMG_{D}	**37.3**†‡ (↑ 3.6)	**55.1**†‡ (↑ 1.0)	$42.8^{\dagger\ddagger}$ (↓ 2.3)	**67.7** (↑ 0.3)
IMG_{2W+D}	$35.7^{\dagger\ddagger}$ (↑ 2.0)	$53.6^{\ddagger}$ (↓ 0.5)	$43.3^{\dagger\ddagger}$ (↓ 1.8)	66.2 (↓ 1.2)
IMG_{E+D}	$37.0^{\dagger\ddagger}$ (↑ 3.3)	$54.7^{\ddagger}$ (↑ 0.6)	$42.6^{\dagger\ddagger}$ (↓ 2.5)	67.2 (↓ 0.2)
German→English				
PBSMT	32.8	34.8	43.9	61.8
NMT	$\underline{38.2}$	$\underline{35.8}$	$\underline{40.2}$	$\underline{62.8}$
IMG_{2W}	$39.5^{\ddagger}$ (↑ 1.3)	$37.1^{\dagger\ddagger}$ (↑ 1.3)	$37.1^{\dagger\ddagger}$ (↓ 3.1)	63.8 (↑ 1.0)
IMG_{E}	$41.1^{\dagger\ddagger}$ (↑ 2.9)	$37.7^{\dagger\ddagger}$ (↑ 1.9)	$37.9^{\dagger\ddagger}$ (↓ 2.3)	65.7 (↑ 2.9)
IMG_{D}	$41.3^{\dagger\ddagger}$ (↑ 3.1)	$37.8^{\dagger\ddagger}$ (↑ 2.0)	$37.9^{\dagger\ddagger}$ (↓ 2.3)	65.7 (↑ 2.9)
IMG_{2W+D}	$39.9^{\dagger\ddagger}$ (↑ 1.7)	$37.2^{\dagger\ddagger}$ (↑ 1.4)	**37.0**†‡ (↓ 3.2)	64.4 (↑ 1.6)
IMG_{E+D}	**41.9**†‡ (↑ 3.7)	**37.9**†‡ (↑ 2.1)	$37.1^{\dagger\ddagger}$ (↓ 3.1)	**66.0** (↑ 3.2)

Table 1: BLEU4, METEOR, chrF3 (higher is better) and TER scores (lower is better) on the M30k_T test set for the two text-only baselines PBSMT and NMT, the two multi-modal NMT models by Huang et al. (2016) (English→German only) and our MNMT models that: *(i)* use images as words in the source sentence (IMG_{1W}, IMG_{2W}), *(ii)* use images to initialise the encoder (IMG_{E}), and *(iii)* use images as additional data to initialise the decoder (IMG_{D}). Best text-only baselines are underscored and best overall results appear in bold. We highlight in parentheses the improvements brought by our models compared to the best corresponding text-only baseline score. Results differ significantly from PBSMT baseline (†) or NMT baseline (‡) with $p = 0.05$.

As our main baseline we train an attention-based NMT model (§2) in which only the textual part of M30k_T is used for training. We also train a PBSMT model built with Moses on the same English→German (German→English) data, where the LM is a 5–gram LM with modified Kneser-Ney smoothing (Kneser and Ney, 1995) trained on the German (English) of the M30k_T dataset. We use minimum error rate training (Och, 2003) for tuning the model parameters for BLEU scores. Our third baseline (English→German), is the best comparable multi-modal model by Huang et al. (2016) and also their best model with additional object detections: respectively models `m1` (image at head) and `m3` in the authors' paper. Finally, our fourth baseline (German→English) is the best-performing model in the WMT'16 multimodal MT shared task (Shah et al., 2016), henceforth PBSMT^{+}. It uses image features as well as additional data from WordNet (Miller, 1995) to rerank n-best lists.

4.1 Results

The Multi30K dataset contains images and bilingual descriptions. Overall, it is a small dataset with a small vocabulary whose sentences have simple syntactic structures and not much ambiguity (Elliott et al., 2016). This is reflected in the fact that even the simplest baselines perform fairly well on it, i.e. the smallest BLEU scores of 32.9 for translating into German, which are still reasonably good results.

[5] We specifically compute character 6-gram F3 scores.

	BLEU4↑	METEOR↑	TER↓	chrF3↑
English→German				
	original training data			
IMG$_{2W}$	36.9	54.3	41.9	66.8
IMG$_{E}$	37.1	55.0	43.1	67.6
IMG$_{D}$	37.3	55.1	42.8	67.7
	+ back-translated training data			
PBSMT	34.0	$\underline{55.0}$	44.7	$\underline{68.0}$
NMT	$\underline{35.5}$	53.4	$\underline{43.3}$	65.3
IMG$_{2W}$	36.7†‡ **(↑ 1.2)**	54.6 ‡ **(↓ 0.4)**	42.0†‡ **(↓ 1.3)**	66.8 **(↓ 1.2)**
IMG$_{E}$	**38.5**†‡ **(↑ 3.0)**	55.7†‡ **(↑ 0.9)**	**41.4**†‡ **(↓ 1.9)**	68.3 **(↑ 0.3)**
IMG$_{D}$	**38.5**†‡ **(↑ 3.0)**	**55.9**†‡ **(↑ 1.1)**	41.6†‡ **(↓ 1.7)**	**68.4** **(↑ 0.4)**
German→English				
PBSMT^{+}	42.5	$\underline{39.5}$	$\underline{35.6}$	$\underline{\mathbf{68.7}}$
	original training data			
IMG$_{2W}$	39.5	37.1	37.1	63.8
IMG$_{E}$	41.1	37.7	37.9	65.7
IMG$_{D}$	41.3	37.8	37.9	65.7
	+ back-translated training data			
NMT	$\underline{42.6}$	38.9	36.1	67.6
IMG$_{2W}$	42.4 ‡ (↓ 0.2)	39.0 ‡ (↑ 0.1)	**34.7**†‡ **(↓ 1.4)**	67.6 (↑ 0.0)
IMG$_{E}$	**43.9**†‡ **(↑ 1.3)**	**39.7** ‡ **(↑ 0.8)**	34.8†‡ **(↓ 1.3)**	**68.7** **(↑ 1.1)**
IMG$_{D}$	43.4 ‡ (↑ 0.8)	39.3 ‡ (↑ 0.4)	35.2 ‡ (↓ 0.9)	67.8 (↑ 0.2)
Improvements (original vs. + back-translated) English→German / German→English				
IMG$_{2W}$	↓ 0.2 / ↑ **2.9**	↑ 0.1 / ↑ **1.9**	↑ 0.1 / ↓ **2.4**	↑ 0.0 / ↑ **3.8**
IMG$_{E}$	↑ **1.4** / ↑ **2.8**	↑ 0.7 / ↑ **2.0**	↓ **1.8** / ↓ **3.1**	↑ **0.7** / ↑ **2.9**
IMG$_{D}$	↑ 1.2 / ↑ **2.1**	↑ **0.8** / ↑ **1.5**	↓ 1.2 / ↓ **2.7**	↑ **0.7** / ↑ **2.1**

Table 2: BLEU4, METEOR, TER and chrF3 scores on the M30k$_{T}$ test set for models trained on original and additional back-translated data. Best text-only baselines are underscored and best overall results in bold. We highlight in parentheses the improvements brought by our models compared to the best baseline score. Results differ significantly from PBSMT baseline (†) or NMT baseline (‡) with $p = 0.05$. We also show the improvements each model yields in each metric when only trained on the original M30k$_{T}$ training set vs. also including additional back-translated data. PBSMT^{+} is the best model in the multi-modal MT shared task (Specia et al., 2016).

Multi30k In Table 1, we show results for translating from English→German and German→English. When translating into German, our multi-modal models perform well, with models IMG$_{E}$ and IMG$_{D}$ improving on both baselines according to all metrics analysed. We also note that all models but IMG$_{2W+D}$ perform consistently better than the strong multi-modal NMT baseline of Huang et al. (2016), even when this model has access to more data (+RCNN features).[6] Combining image features in the encoder and the decoder at the same time does not seem to improve results compared to using the image features in only the encoder or the decoder when translating into German. To the best of our knowledge, it is the first time a purely neural model significantly improves over a PBSMT model in all metrics on this data set.

When translating into English, all multi-modal models significantly improve over the NMT baseline, with the only exception being model IMG$_{2W}$'s BLEU scores. In this scenario, model IMG$_{E+D}$ is the best performing one according to all but one metric. However, differences between multi-modal models are not statistically significant, i.e. all multi-modal models but IMG$_{2W}$ and IMG$_{2W+D}$ perform comparably.

Additional back-translated data Arguably, the main downside of applying multi-modal NMT in a real-world scenario is the small amount of publicly available training data (∼30K entries). For that reason, we back-translated the German and English sentences in the M30k$_{C}$ and created two sets of 145K synthetic triples, one for each translation direction, as described in §3.

In Table 2, we present results for some of the models evaluated in Table 1 but when also trained on the additional data. In order to add more data to our PBSMT baseline, we simply added the German sentences in the M30k$_{C}$ to train the LM.[7] We also include results for PBSMT^{+}, which uses image features as well as additional features extracted using WordNet (Shah et al., 2016). When translating into German, both our models IMG$_{E}$ and IMG$_{D}$ that use global image features to initialise the encoder and the decoder, respectively, significantly improve according to BLEU, METEOR and TER with the additional back-translated data, and also achieved better chrF3 scores. When translating into English, IMG$_{E}$ is the only model to significantly improve over both baselines according to all metrics with the additional back-translated data, also improving chrF3 scores. We highlight that our best-performing model IMG$_{E}$ significantly outperforms PBSMT^{+} according to BLEU and TER, and all our other multi-modal models perform comparably to it. This is a noteworthy finding, since

[6] In fact, model IMG$_{2W+D}$ still improves on the multi-modal baseline of Huang et al. (2016) when trained on the same data.

[7] Adding the synthetic sentence pairs to train the baseline PBSMT model, as we did with all neural MT models, deteriorated the results.

Multi30k test set (English→German)

	Ensemble?	BLEU4 ↑	METEOR↑	TER↓	chrF3↑
IMG$_D$	×	37.3	55.1	42.8	67.7
IMG$_D$ + IMG$_E$	✓	40.1 (↑ 2.8)	58.5 (↑ 3.4)	40.7 (↓ 2.1)	68.1 (↑ 0.4)
IMG$_D$ + IMG$_E$ + IMG$_{2W}$	✓	41.0 (↑ 3.7)	58.9 (↑ 3.8)	39.7 (↓ 3.1)	68.3 (↑ 0.6)
IMG$_D$ + IMG$_E$ + IMG$_{2W}$ + IMG$_D$	✓	**41.3** (**↑ 4.0**)	**59.2** (**↑ 4.1**)	**39.5** (**↓ 3.3**)	**68.5** (**↑ 0.8**)

Table 3: Results for different combinations of multi-modal models, all trained on the original M30k$_T$ training data only, evaluated on the M30k$_T$ test set.

PBSMT$^+$ uses image features as well as additional data from WordNet and, to the best of our knowledge, is the best published model in this language pair and data set to date.

Ensemble decoding We now report on how can ensemble decoding be used to improve translations obtained with our multi-modal NMT models. In order to do that, we use different combinations of models trained on the original M30k$_T$ training set to translate from English into German. We built ensembles of different models by starting with our best performing multi-modal model on this language pair and data set, IMG$_D$, and by adding new models to the ensemble one by one, until we reach a maximum of four independent models, all of which are trained separately and on the original M30k$_T$ training data only. In Table 3, we show results when translating the M30k$_T$'s test set. These models were also evaluated in our recent participation in the WMT 2017 multi-modal MT shared task (Calixto et al., 2017a).

We first note that to add more models to the ensemble seems to always improve translations, and by a considerable margin ($\sim$ 3 BLEU/METEOR points). Adding model IMG$_{2W}$ to the ensemble already consisting of models IMG$_E$ and IMG$_D$ improves translations according to all metrics evaluated. This is an interesting result, since compared to these other two multi-modal models, model IMG$_{2W}$ performs poorly according to BLEU, METEOR and chrF3. Regardless of that fact, our best results are obtained with an ensemble of four different multi-modal models, including model IMG$_{2W}$.

By using an ensemble of four different multi-modal NMT models trained on the translated Multi30k training data, we were able to obtain translations comparable to or even better than those obtained with the strong multi-modal NMT model of Calixto et al. (2017b), which is pre-trained on large amounts of English–German data and uses local image features. Finally, we have recently participated in the WMT 2017 multi-modal MT shared task, and our system submissions ranked among the best performing systems under the constrained data regime (Calixto et al., 2017a). We note that our models performed particularly well on the ambiguous MSCOCO test set (Elliott et al., 2017), which indicate its ability to use the image information in disambiguating difficult source sentences into their correct translations.

5 Error Analysis

In Table 4 we show translations into German generated by different models for one entry of the M30k test set. In this example, the last three multi-modal models extrapolate the reference+image and describe "ceremony" as a "wedding ceremony" (IMG$_{2W}$) and as an "Olympics ceremony" (IMG$_E$ and IMG$_D$). This could be due to the fact that the training set is small, depicts a small variation of different scenes and contains different forms of biasses (van Miltenburg, 2015).

In Table 5, we draw attention to an example where some models generate what seems to be *novel visual terms*. Neither the source German sentence nor the English reference translation contained the translated units "having fun" or "Mexican restaurant", although both could have been inferred at least partially from the image. In this example, the visual term "having fun" is also generated by the baseline NMT model, making it clear that at times what seems like a translation extracted exclusively from the image may have been learnt from the training text data. However, none of the two baselines translated "Mexikanischen Setting" as "Mexican restaurant", but four out of the five multi-modal models did. The multi-modal models also had problems translating the German "trinkt Shots" (drinking shots). We observe translations such as "having drinks" (IMG$_{2W}$), which

src.	a woman with long hair is at a graduation ceremony .
ref.	eine Frau mit langen Haaren bei einer Abschluss Feier.
NMT	eine Frau mit langen Haaren **ist an einer StaZeremonie**.
PBSMT	eine Frau mit langen Haaren steht **an einem Abschluss**
IMG$_{1W}$	eine Frau mit langen Haaren **ist an einer warmen Zeremonie teil**
IMG$_{2W}$	eine Frau mit langen Haaren steht bei einer **Hochzeit Feier**.
IMG$_{E}$	eine **lang haarige** Frau bei einer **olympischen** Zeremonie.
IMG$_{D}$	eine **lang haarige** Frau bei einer **olympischen** Zeremonie.

Table 4: Translations for the 668th example in the M30k test set.

src.	eine Gruppe junger Menschen trinkt Shots in einem Mexikanischen Setting .
ref.	a group of young people take shots in a Mexican setting .
NMT	a group of young people are **having fun** in an auditorium .
PBSMT	a group of young people drinking at a Shots Mexikanischen Setting .
IMG$_{2W}$	a group of young people having drinks in a **Mexican restaurant** .
IMG$_{E}$	a group of young people drinking apples in a **Mexican restaurant** .
IMG$_{D}$	a group of young people drinking food in a **Mexican restaurant** .
IMG$_{2W+D}$	a group of young people **having fun** in a Mexican room .
IMG$_{E+D}$	a group of young people drinking dishes in a **Mexican restaurant** .

Table 5: Translations for 300th example in the M30k test set.

although not a novel translation is still correct, but also "drinking apples" (IMG$_{E}$), "drinking food" (IMG$_{D}$), and "drinking dishes" (IMG$_{E+D}$), which are clearly incorrect.

6 Conclusions and future work

In this work, we introduced models that incorporate images into state-of-the-art attention-based NMT, by using images as words in the source sentence, to initialise the encoder's hidden state and as additional data in the initialisation of the decoder's hidden state. The intuition behind our effort is to use images to visually *ground* translations, and consequently increase translation quality. We demonstrate with extensive experiments that adding global image features into NMT significantly improves the translations of image descriptions compared to text-only NMT and PBSMT. It also improves significantly on the previous state-of-the-art model of Huang et al. (2016) (English→German), and performs comparably to the best published results of Shah et al. (2016) (German→English). Overall, we note that using images as words in the source sequence (IMG$_{1W}$, IMG$_{2W}$), an idea similarly entertained by Huang et al. (2016), does not fare as well as to directly incorporate the image either in the encoder or the decoder (IMG$_{E}$ and IMG$_{D}$), independently of the target language. The fact that multi-modal NMT models can benefit from back-translated data is also an interesting finding.

In future work, we will conduct a more systematic study on the impact that synthetic back-translated data brings to multi-modal NMT, and run an error analysis to identify what particular types of errors our models make (and prevent).

Acknowledgments

This project has received funding from Science Foundation Ireland in the ADAPT Centre for Digital Content Technology (`www.adaptcentre.ie`) at Dublin City University funded under the SFI Research Centres Programme (Grant 13/RC/2106) co-funded under the European Regional Development Fund and the European Union Horizon 2020 research and innovation programme under grant agreement 645452 (QT21).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations, ICLR 2015*. San Diego, California.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam M. Shazeer. 2015. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Advances in Neural Information Processing Systems, NIPS*. http://arxiv.org/abs/1506.03099.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* 3:1137–1155. http://dl.acm.org/citation.cfm?id=944919.944966.

Ozan Caglayan, Walid Aransa, Yaxing Wang, Marc Masana, Mercedes García-Martínez, Fethi Bougares, Loïc Barrault, and Joost van de Weijer. 2016. Does Multimodality Help Human and Machine for Translation and Image Captioning? In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany, pages 627–633. http://www.aclweb.org/anthology/W/W16/W16-2358.

Iacer Calixto, Koel Dutta Chowdhury, and Qun Liu. 2017a. DCU System Report on the WMT 2017 Multi-modal Machine Translation Task. In *Proceedings of the Second Conference on Machine Translation*. Copenhagen, Denmark.

Iacer Calixto, Desmond Elliott, and Stella Frank. 2016. DCU-UvA Multimodal MT System Report. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany, pages 634–638. http://www.aclweb.org/anthology/W/W16/W16-2359.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017b. Doubly-Attentive Decoder for Multi-modal Neural Machine Translation. In *Proceedings of the 55th Conference of the Association for Computational Linguistics: Volume 1, Long Papers*. Vancouver, Canada (Paper Accepted).

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation* page 103.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar, pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*. Portland, Oregon, HLT '11, pages 176–181. http://dl.acm.org/citation.cfm?id=2002736.2002774.

Michael Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, and Kate Saenko. 2015. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. Boston, US, pages 2625–2634.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-Task Learning for Multiple Language Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 1723–1732. http://www.aclweb.org/anthology/P15-1166.

Desmond Elliott, Stella Frank, Loïc Barrault, Fethi Bougares, and Lucia Specia. 2017. Findings of the Second Shared Task on Multimodal Machine Translation and Multilingual Image Description. In *Proceedings of the Second Conference on Machine Translation*. Copenhagen, Denmark.

Desmond Elliott, Stella Frank, and Eva Hasler. 2015. Multi-language image description with neural sequence models. *CoRR* abs/1510.04709. http://arxiv.org/abs/1510.04709.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30K: Multilingual English-German Image Descriptions. In *Proceedings of the 5th Workshop on Vision and Language, VL@ACL 2016*. Berlin, Germany. http://aclweb.org/anthology/W/W16/W16-3210.pdf.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-Way, Multilingual Neural Machine Translation with a Shared Attention Mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California, pages 866–875. http://www.aclweb.org/anthology/N16-1101.

Yarin Gal and Zoubin Ghahramani. 2016. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Advances in Neural Information Processing Systems, NIPS*, Barcelona, Spain, pages 1019–1027. http://papers.nips.cc/paper/6241-a-theoretically-grounded-application-of-dropout-in-recurrent-neural-networks.pdf.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA, CVPR '14, pages 580–587. https://doi.org/10.1109/CVPR.2014.81.

Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *CoRR* abs/1308.0850. http://arxiv.org/abs/1308.0850.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing Image Description As a Ranking Task: Data, Models and Evaluation Metrics. *J. Artif. Int. Res.* 47(1):853–899. http://dl.acm.org/citation.cfm?id=2566972.2566993.

Po-Yao Huang, Frederick Liu, Sz-Rung Shiang, Jean Oh, and Chris Dyer. 2016. Attention-based Multimodal Neural Machine Translation. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany, pages 639–645. http://www.aclweb.org/anthology/W/W16/W16-2360.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*. Seattle, USA, pages 1700–1709.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*. Boston, Massachusetts, pages 3128–3137.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2017. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *International Conference on Learning Representations, ICLR 2017*. Toulon, France.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. 2014. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR* abs/1411.2539. http://arxiv.org/abs/1411.2539.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. Detroit, Michigan, volume I, pages 181–184.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, Prague, Czech Republic, ACL '07, pages 177–180. http://dl.acm.org/citation.cfm?id=1557769.1557821.

Jindřich Libovický, Jindřich Helcl, Marek Tlustý, Ondřej Bojar, and Pavel Pecina. 2016. CUNI System for WMT16 Automatic Post-Editing and Multimodal Translation Tasks. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany, pages 646–654. http://www.aclweb.org/anthology/W/W16/W16-2361.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Lisbon, Portugal, pages 1412–1421.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L. Yuille. 2014. Explain Images with Multimodal Recurrent Neural Networks. http://arxiv.org/abs/1410.1090.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM* 38(11):39–41. https://doi.org/10.1145/219717.219748.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*. Sapporo, Japan, ACL '03, pages 160–167. https://doi.org/10.3115/1075096.1075117.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Philadelphia, Pennsylvania, ACL '02, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Maja Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal, pages 392–395. http://aclweb.org/anthology/W15-3049.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115(3):211–252. https://doi.org/10.1007/s11263-015-0816-y.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 86–96. http://www.aclweb.org/anthology/P16-1009.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 1715–1725. http://www.aclweb.org/anthology/P16-1162.

Kashif Shah, Josiah Wang, and Lucia Specia. 2016. SHEF-Multimodal: Grounding Machine Translation on Images. In *Proceedings of the First Conference on Machine Translation*. Berlin, Germany, pages 660–665. http://www.aclweb.org/anthology/W/W16/W16-2363.

K. Simonyan and A. Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR* abs/1409.1556.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*. Cambridge, MA, pages 223–231.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A Shared Task on Multimodal Machine Translation and Crosslingual Image Description. In *Proceedings of the First Conference on Machine Translation, WMT 2016*. Berlin, Germany, pages 543–553. http://aclweb.org/anthology/W/W16/W16-2346.pdf.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*. Montréal, Canada, pages 3104–3112.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling Coverage for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 76–85. http://www.aclweb.org/anthology/P16-1008.

Emiel van Miltenburg. 2015. Stereotyping and bias in the flickr30k dataset. In *Proceedings of the Workshop on Multimodal Corpora, MMC-2016*. Portorož, Slovenia, pages 1–4.

Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to Sequence - Video to Text. In *Proceedings of the IEEE International Conference on Computer Vision*. Santiago, Chile, pages 4534–4542.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*. Boston, Massachusetts, pages 3156–3164.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. JMLR Workshop and Conference Proceedings, Lille, France, pages 2048–2057. http://jmlr.org/proceedings/papers/v37/xuc15.pdf.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics* 2:67–78.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.

Mapping Instructions and Visual Observations to Actions with Reinforcement Learning

Dipendra Misra[†], **John Langford**[‡], **and Yoav Artzi**[†]

[†] Dept. of Computer Science and Cornell Tech, Cornell University, New York, NY 10044
{dkm, yoav}@cs.cornell.edu

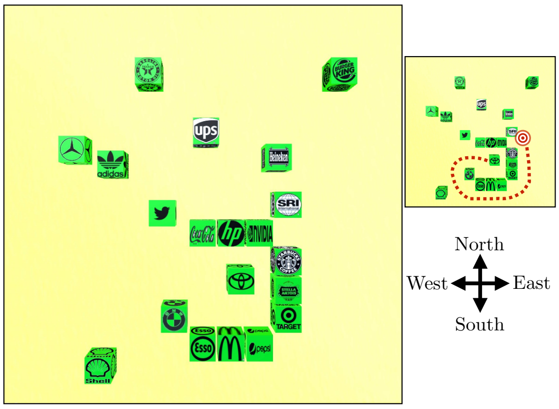
[‡] Microsoft Research, New York, NY 10011
jcl@microsoft.com

Abstract

We propose to directly map raw visual observations and text input to actions for instruction execution. While existing approaches assume access to structured environment representations or use a pipeline of separately trained models, we learn a single model to jointly reason about linguistic and visual input. We use reinforcement learning in a contextual bandit setting to train a neural network agent. To guide the agent's exploration, we use reward shaping with different forms of supervision. Our approach does not require intermediate representations, planning procedures, or training different models. We evaluate in a simulated environment, and show significant improvements over supervised learning and common reinforcement learning variants.

1 Introduction

An agent executing natural language instructions requires robust understanding of language and its environment. Existing approaches addressing this problem assume structured environment representations (e.g.,. Chen and Mooney, 2011; Mei et al., 2016), or combine separately trained models (e.g., Matuszek et al., 2010; Tellex et al., 2011), including for language understanding and visual reasoning. We propose to directly map text and raw image input to actions with a single learned model. This approach offers multiple benefits, such as not requiring intermediate representations, planning procedures, or training multiple models.



Put the Toyota block in the same row as the SRI block, in the first open space to the right of the SRI block
Move Toyota to the immediate right of SRI, evenly aligned and slightly separated
Move the Toyota block around the pile and place it just to the right of the SRI block
Place Toyota block just to the right of The SRI Block
Toyota, right side of SRI

Figure 1: Instructions in the Blocks environment. The instructions all describe the same task. Given the observed RGB image of the start state (large image), our goal is to execute such instructions. In this task, the direct-line path to the target position is blocked, and the agent must plan and move the Toyota block around. The small image marks the target and an example path, which includes 34 steps.

Figure 1 illustrates the problem in the Blocks environment (Bisk et al., 2016). The agent observes the environment as an RGB image using a camera sensor. Given the RGB input, the agent must recognize the blocks and their layout. To understand the instruction, the agent must identify the block to move (Toyota block) and the destination (just right of the SRI block). This requires solving semantic and grounding problems. For example, consider the topmost instruction in the figure. The agent needs to identify the phrase referring to the block to move, *Toyota block*, and ground it. It must resolve and ground the phrase *SRI block* as a reference position, which is then modified by the spatial meaning recovered from *the same row as* or *first open space to the right of*, to identify the goal position. Finally, the agent needs to generate actions, for example moving the Toyota block around obstructing blocks.

To address these challenges with a single model,

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1004–1015
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

we design a neural network agent. The agent executes instructions by generating a sequence of actions. At each step, the agent takes as input the instruction text, observes the world as an RGB image, and selects the next action. Action execution changes the state of the world. Given an observation of the new world state, the agent selects the next action. This process continues until the agent indicates execution completion. When selecting actions, the agent jointly reasons about its observations and the instruction text. This enables decisions based on close interaction between observations and linguistic input.

We train the agent with different levels of supervision, including complete demonstrations of the desired behavior and annotations of the goal state only. While the learning problem can be easily cast as a supervised learning problem, learning only from the states observed in the training data results in poor generalization and failure to recover from test errors. We use reinforcement learning (Sutton and Barto, 1998) to observe a broader set of states through exploration. Following recent work in robotics (e.g., Levine et al., 2016; Rusu et al., 2016), we assume the training environment, in contrast to the test environment, is instrumented and provides access to the state. This enables a simple problem reward function that uses the state and provides positive reward on task completion only. This type of reward offers two important advantages: (a) it is a simple way to express the ideal agent behavior we wish to achieve, and (b) it creates a platform to add training data information.

We use reward shaping (Ng et al., 1999) to exploit the training data and add to the reward additional information. The modularity of shaping allows varying the amount of supervision, for example by using complete demonstrations for only a fraction of the training examples. Shaping also naturally associates actions with immediate reward. This enables learning in a contextual bandit setting (Auer et al., 2002; Langford and Zhang, 2007), where optimizing the immediate reward is sufficient and has better sample complexity than unconstrained reinforcement learning (Agarwal et al., 2014).

We evaluate with the block world environment and data of Bisk et al. (2016), where each instruction moves one block (Figure 1). While the original task focused on source and target prediction only, we build an interactive simulator and formulate the task of predicting the complete sequence of actions. At each step, the agent must select between 81 actions with 15.4 steps required to complete a task on average, significantly more than existing environments (e.g., Chen and Mooney, 2011). Our experiments demonstrate that our reinforcement learning approach effectively reduces execution error by 24% over standard supervised learning and 34-39% over common reinforcement learning techniques. Our simulator, code, models, and execution videos are available at: `https://github.com/clic-lab/blocks`.

2 Technical Overview

Task Let $\mathcal{X}$ be the set of all *instructions*, $\mathcal{S}$ the set of all *world states*, and $\mathcal{A}$ the set of all *actions*. An instruction $\bar{x} \in \mathcal{X}$ is a sequence $\langle x_1, \ldots, x_n \rangle$, where each x_i is a token. The agent executes instructions by generating a sequence of actions, and indicates execution completion with the special action STOP. Action execution modifies the world state following a transition function $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. The execution $\bar{e}$ of an instruction $\bar{x}$ starting from s_1 is an m-length sequence $\langle (s_1, a_1), \ldots, (s_m, a_m) \rangle$, where $s_j \in \mathcal{S}$, $a_j \in \mathcal{A}$, $T(s_j, a_j) = s_{j+1}$ and $a_m = \texttt{STOP}$. In Blocks (Figure 1), a state specifies the positions of all blocks. For each action, the agent moves a single block on the plane in one of four directions (north, south, east, or west). There are 20 blocks, and 81 possible actions at each step, including STOP. For example, to correctly execute the instructions in the figure, the agent's likely first action is TOYOTA-WEST, which moves the Toyota block one step west. Blocks can not move over or through other blocks.

Model The agent observes the world state via a visual sensor (i.e., a camera). Given a world state s, the agent observes an RGB image I generated by the function $\textsc{Img}(s)$. We distinguish between the world state s and the *agent context*[1] $\tilde{s}$, which includes the instruction, the observed image $\textsc{Img}(s)$, images of previous states, and the previous action. To map instructions to actions, the agent reasons about the agent context $\tilde{s}$ to generate a sequence of actions. At each step, the agent generates a single action. We model the agent with a

[1] We use the term *context* similar to how it is used in the contextual bandit literature to refer to the information available for decision making. While agent contexts capture information about the world state, they do not include physical information, except as captured by observed images.

neural network policy. At each step j, the network takes as input the current agent context $\tilde{s}_j$, and predicts the next action to execute a_j. We formally define the agent context and model in Section 4.

Learning We assume access to training data with N examples $\{(\bar{x}^{(i)}, s_1^{(i)}, \bar{e}^{(i)})\}_{i=1}^N$, where $\bar{x}^{(i)}$ is an instruction, $s_1^{(i)}$ is a start state, and $\bar{e}^{(i)}$ is an execution demonstration of $\bar{x}^{(i)}$ starting at $s_1^{(i)}$. We use policy gradient (Section 5) with reward shaping derived from the training data to increase learning speed and exploration effectiveness (Section 6). Following work in robotics (e.g., Levine et al., 2016), we assume an instrumented environment with access to the world state to compute the reward during training only. We define our approach in general terms with demonstrations, but also experiment with training using goal states.

Evaluation We evaluate task completion error on a test set $\{(\bar{x}^{(i)}, s_1^{(i)}, s_g^{(i)})\}_{i=1}^M$, where $\bar{x}^{(i)}$ is an instruction, $s_1^{(i)}$ is a start state, and $s_g^{(i)}$ is the goal state. We measure execution error as the distance between the final execution state and $s_g^{(i)}$.

3 Related Work

Learning to follow instructions was studied extensively with structured environment representations, including with semantic parsing (Chen and Mooney, 2011; Kim and Mooney, 2012, 2013; Artzi and Zettlemoyer, 2013; Artzi et al., 2014a,b; Misra et al., 2015, 2016), alignment models (Andreas and Klein, 2015), reinforcement learning (Branavan et al., 2009, 2010; Vogel and Jurafsky, 2010), and neural network models (Mei et al., 2016). In contrast, we study the problem of an agent that takes as input instructions and raw visual input. Instruction following with visual input was studied with pipeline approaches that use separately learned models for visual reasoning (Matuszek et al., 2010, 2012; Tellex et al., 2011; Paul et al., 2016). Rather than decomposing the problem, we adopt a single-model approach and learn from instructions paired with demonstrations or goal states. Our work is related to Sung et al. (2015). While they use sensory input to select and adjust a trajectory observed during training, we are not restricted to training sequences. Executing instructions in non-learning settings has also received significant attention (e.g., Winograd, 1972; Webber et al., 1995; MacMahon et al., 2006).

Our work is related to a growing interest in problems that combine language and vision, including visual question answering (e.g., Antol et al., 2015; Andreas et al., 2016b,a), caption generation (e.g., Chen et al., 2015, 2016; Xu et al., 2015), and visual reasoning (Johnson et al., 2016; Suhr et al., 2017). We address the prediction of the next action given a world image and an instruction.

Reinforcement learning with neural networks has been used for various NLP tasks, including text-based games (Narasimhan et al., 2015; He et al., 2016), information extraction (Narasimhan et al., 2016), co-reference resolution (Clark and Manning, 2016), and dialog (Li et al., 2016).

Neural network reinforcement learning techniques have been recently studied for behavior learning tasks, including playing games (Mnih et al., 2013, 2015, 2016; Silver et al., 2016) and solving memory puzzles (Oh et al., 2016). In contrast to this line of work, our data is limited. Observing new states in a computer game simply requires playing it. However, our agent also considers natural language instructions. As the set of instructions is limited to the training data, the set of agent contexts seen during learning is constrained. We address the data efficiency problem by learning in a contextual bandit setting, which is known to be more tractable (Agarwal et al., 2014), and using reward shaping to increase exploration effectiveness. Zhu et al. (2017) address generalization of reinforcement learning to new target goals in visual search by providing the agent an image of the goal state. We address a related problem. However, we provide natural language and the agent must learn to recognize the goal state.

Reinforcement learning is extensively used in robotics (Kober et al., 2013). Similar to recent work on learning neural network policies for robot control (Levine et al., 2016; Schulman et al., 2015; Rusu et al., 2016), we assume an instrumented training environment and use the state to compute rewards during learning. Our approach adds the ability to specify tasks using natural language.

4 Model

We model the agent policy π with a neural network. The agent observes the instruction and an RGB image of the world. Given a world state s, the image I is generated using the function $\textsc{Img}(s)$. The instruction execution is generated one step at a time. At each step j, the agent observes an image I_j of the current world state s_j and the instruction $\bar{x}$, predicts the action a_j, and executes it to transition to the next state s_{j+1}.

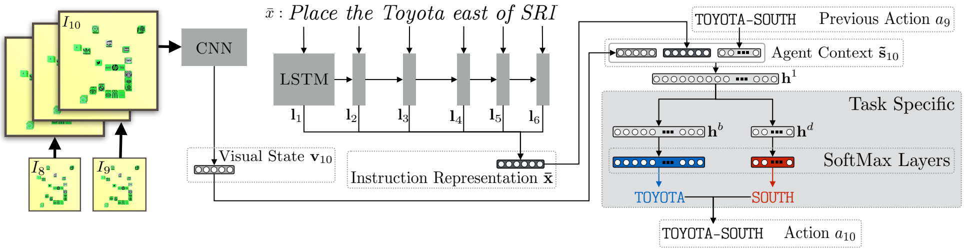

$\bar{x}$: Place the Toyota east of SRI
I_{10}
I_8
I_9
CNN
LSTM
$\mathbf{l}_1$
$\mathbf{l}_2$
$\mathbf{l}_3$
$\mathbf{l}_4$
$\mathbf{l}_5$
$\mathbf{l}_6$
Visual State $\mathbf{v}_{10}$
Instruction Representation $\bar{\mathbf{x}}$
TOYOTA-SOUTH Previous Action a_9
Agent Context $\tilde{\mathbf{s}}_{10}$
$\mathbf{h}^1$
Task Specific
$\mathbf{h}^b$
$\mathbf{h}^d$
SoftMax Layers
TOYOTA
SOUTH
TOYOTA-SOUTH Action a_{10}

Figure 2: Illustration of the policy architecture showing the 10th step in the execution of the instruction *Place the Toyota east of SRI* in the state from Figure 1. The network takes as input the instruction $\bar{x}$, image of the current state I_{10}, images of previous states I_8 and I_9 (with $K = 2$), and the previous action a_9. The text and images are embedded with LSTM and CNN. The actions are selected with the task specific multi-layer perceptron.

This process continues until STOP is predicted and the agent stops, indicating instruction completion. The agent also has access to K images of previous states and the previous action to distinguish between different stages of the execution (Mnih et al., 2015). Figure 2 illustrates our architecture.

Formally,[2] at step j, the agent considers an agent context $\tilde{s}_j$, which is a tuple $(\bar{x}, I_j, I_{j-1}, \ldots, I_{j-K}, a_{j-1})$, where $\bar{x}$ is the natural language instruction, I_j is an image of the current world state, the images $I_{j-1}, \ldots, I_{j-K}$ represent K previous states, and a_{j-1} is the previous action. The agent context includes information about the current state and the execution. Considering the previous action a_{j-1} allows the agent to avoid repeating failed actions, for example when trying to move in the direction of an obstacle. In Figure 2, the agent is given the instruction *Place the Toyota east of SRI*, is at the 10-th execution step, and considers $K = 2$ previous images.

We generate continuous vector representations for all inputs, and jointly reason about both text and image modalities to select the next action. We use a recurrent neural network (RNN; Elman, 1990) with a long short-term memory (LSTM; Hochreiter and Schmidhuber, 1997) recurrence to map the instruction $\bar{x} = \langle x_1, \ldots, x_n \rangle$ to a vector representation $\bar{\mathbf{x}}$. Each token x_i is mapped to a fixed dimensional vector with the learned embedding function $\psi(x_i)$. The instruction representation $\bar{\mathbf{x}}$ is computed by applying the LSTM recurrence to generate a sequence of hidden states $\mathbf{l}_i = \text{LSTM}(\psi(x_i), \mathbf{l}_{i-1})$, and computing the mean $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{l}_i$ (Narasimhan et al., 2015). The current image I_j and previous images I_{j-1},…,I_{j-K} are concatenated along the channel dimension and embedded with a convolutional neural network (CNN) to generate the visual state $\mathbf{v}$ (Mnih et al., 2013). The last action a_{j-1} is embedded with the function $\psi_a(a_{j-1})$. The vectors $\mathbf{v}_j$, $\bar{\mathbf{x}}$, and $\psi_a(a_{j-1})$ are concatenated to create the agent context vector representation $\tilde{\mathbf{s}}_j = [\mathbf{v}_j, \bar{\mathbf{x}}, \psi_a(a_{j-1})]$.

To compute the action to execute, we use a feedforward perceptron that decomposes according to the domain actions. This computation selects the next action conditioned on the instruction text and observations from both the current world state and recent history. In the block world domain, where actions decompose to selecting the block to move and the direction, the network computes block and direction probabilities. Formally, we decompose an action a to direction a^D and block a^B. We compute the feedforward network:

$$\begin{aligned}
\mathbf{h}^1 &= \max(\mathbf{W}^{(1)}\tilde{\mathbf{s}}_j + \mathbf{b}^{(1)}, 0) \\
\mathbf{h}^D &= \mathbf{W}^{(D)}\mathbf{h}^1 + \mathbf{b}^{(D)} \\
\mathbf{h}^B &= \mathbf{W}^{(B)}\mathbf{h}^1 + \mathbf{b}^{(B)} \quad ,
\end{aligned}$$

and the action probability is a product of the component probabilities:

$$\begin{aligned}
P(a_j^D = d \mid \bar{x}, s_j, a_{j-1}) &\propto \exp(\mathbf{h}_d^D) \\
P(a_j^B = b \mid \bar{x}, s_j, a_{j-1}) &\propto \exp(\mathbf{h}_b^B) \quad .
\end{aligned}$$

At the beginning of execution, the first action a_0 is set to the special value NONE, and previous images are zero matrices. The embedding function ψ is a learned matrix. The function ψ_a concatenates the embeddings of a_{j-1}^D and a_{j-1}^B, which are obtained from learned matrices, to compute the embedding of a_{j-1}. The model parameters θ include $\mathbf{W}^{(1)}$, $\mathbf{b}^{(1)}$, $\mathbf{W}^{(D)}$, $\mathbf{b}^{(D)}$, $\mathbf{W}^{(B)}$, $\mathbf{b}^{(B)}$, the parameters of the LSTM recurrence, the parameters of the convolutional network CNN, and the embedding matrices. In our experiments (Section 7), all parameters are learned without external resources.

5 Learning

We use policy gradient for reinforcement learning (Williams, 1992) to estimate the parameters θ of the agent policy. We assume access to a

[2] We use bold-face capital letters for matrices and bold-face lowercase letters for vectors. Computed input and state representations use bold versions of the symbols. For example, $\bar{\mathbf{x}}$ is the computed representation of an instruction $\bar{x}$.

training set of N examples $\{(\bar{x}^{(i)}, s_1^{(i)}, \bar{e}^{(i)})\}_{i=1}^{N}$, where $\bar{x}^{(i)}$ is an instruction, $s_1^{(i)}$ is a start state, and $\bar{e}^{(i)}$ is an execution demonstration starting from $s_1^{(i)}$ of instruction $\bar{x}^{(i)}$. The main learning challenge is learning how to execute instructions given raw visual input from relatively limited data. We learn in a contextual bandit setting, which provides theoretical advantages over general reinforcement learning. In Section 8, we verify this empirically.

Reward Function The instruction execution problem defines a simple problem reward to measure task completion. The agent receives a positive reward when the task is completed, a negative reward for incorrect completion (i.e., STOP in the wrong state) and actions that fail to execute (e.g., when the direction is blocked), and a small penalty otherwise, which induces a preference for shorter trajectories. To compute the reward, we assume access to the world state. This learning setup is inspired by work in robotics, where it is achieved by instrumenting the training environment (Section 3). The agent, on the other hand, only uses the agent context (Section 4). When deployed, the system relies on visual observations and natural language instructions only. The reward function $R^{(i)} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is defined for each training example $(\bar{x}^{(i)}, s_1^{(i)}, \bar{e}^{(i)})$, $i = 1 \ldots N$:

$$R^{(i)}(s,a) = \begin{cases} 1.0 & \text{if } s = s_{m^{(i)}} \text{ and } a = \texttt{STOP} \\ -1.0 & s \neq s_{m^{(i)}} \text{ and } a = \texttt{STOP} \\ -1.0 & a \text{ fails to execute} \\ -\delta & \text{else} \end{cases} ,$$

where $m^{(i)}$ is the length of $\bar{e}^{(i)}$.

The reward function does not provide intermediate positive feedback to the agent for actions that bring it closer to its goal. When the agent explores randomly early during learning, it is unlikely to encounter the goal state due to the large number of steps required to execute tasks. As a result, the agent does not observe positive reward and fails to learn. In Section 6, we describe how reward shaping, a method to augment the reward with additional information, is used to take advantage of the training data and address this challenge.

Policy Gradient Objective We adapt the policy gradient objective defined by Sutton et al. (1999) to multiple starting states and reward functions:

$$\mathcal{J} = \frac{1}{N} \sum_{i=1}^{N} V_{\pi}^{(i)}(s_1^{(i)}) \ ,$$

where $V_{\pi}^{(i)}(s_1^{(i)})$ is the value given by $R^{(i)}$ starting from $s_1^{(i)}$ under the policy π. The summation expresses the goal of learning a behavior parameterized by natural language instructions.

Contextual Bandit Setting In contrast to most policy gradient approaches, we apply the objective to a contextual bandit setting where immediate reward is optimized rather than total expected reward. The primary theoretical advantage of contextual bandits is much tighter sample complexity bounds when comparing upper bounds for contextual bandits (Langford and Zhang, 2007) even with an adversarial sequence of contexts (Auer et al., 2002) to lower bounds (Krishnamurthy et al., 2016) or upper bounds (Kearns et al., 1999) for total reward maximization. This property is particularly suitable for the few-sample regime common in natural language problems. While reinforcement learning with neural network policies is known to require large amounts of training data (Mnih et al., 2015), the limited number of training sentences constrains the diversity and volume of agent contexts we can observe during training. Empirically, this translates to poor results when optimizing the total reward (REINFORCE baseline in Section 8). To derive the approximate gradient, we use the likelihood ratio method:

$$\nabla_{\theta}\mathcal{J} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}[\nabla_{\theta} \log \pi(\tilde{s}, a) R^{(i)}(s,a)] \ ,$$

where reward is computed from the world state but policy is learned on the agent context. We approximate the gradient using sampling.

This training regime, where immediate reward optimization is sufficient to optimize policy parameters θ, is enabled by the shaped reward we introduce in Section 6. While the objective is designed to work best with the shaped reward, the algorithm remains the same for any choice of reward definition including the original problem reward or several possibilities formed by reward shaping.

Entropy Penalty We observe that early in training, the agent is overwhelmed with negative reward and rarely completes the task. This results in the policy π rapidly converging towards a suboptimal deterministic policy with an entropy of 0. To delay premature convergence we add an entropy term to the objective (Williams and Peng, 1991; Mnih et al., 2016). The entropy term encourages a uniform distribution policy, and in practice stimulates exploration early during training. The regularized gradient is:

$$\nabla_{\theta}\mathcal{J} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}[\nabla_{\theta} \log \pi(\tilde{s}, a) R^{(i)}(s,a) + \lambda \nabla_{\theta} H(\pi(\tilde{s}, \cdot))] \ ,$$

Algorithm 1 Policy gradient learning

```
Input: Training set $\{(\bar{x}^{(i)}, s_1^{(i)}, \bar{e}^{(i)})\}_{i=1}^N$, learning rate $\mu$,
    epochs $T$, horizon $J$, and entropy regularization term $\lambda$.
Definitions: $\textsc{Img}(s)$ is a camera sensor that reports an RGB
    image of state $s$. $\pi$ is a probabilistic neural network
    policy parameterized by $\theta$, as described in Section 4.
    $\textsc{Execute}(s, a)$ executes the action $a$ at the state $s$, and
    returns the new state. $R^{(i)}$ is the reward function for
    example $i$. $\textsc{Adam}(\Delta)$ applies a per-feature learning rate
    to the gradient $\Delta$ (Kingma and Ba, 2014).
Output: Policy parameters $\theta$.
 1: » Iterate over the training data.
 2: for $t = 1$ to $T$, $i = 1$ to $N$ do
 3:     $I_{1-K}, \ldots, I_0 = \vec{0}$
 4:     $a_0 = \texttt{NONE}$, $s_1 = s_1^{(i)}$
 5:     $j = 1$
 6:     » Rollout up to episode limit.
 7:     while $j \leq J$ and $a_j \neq \texttt{STOP}$ do
 8:         » Observe world and construct agent context.
 9:         $I_j = \textsc{Img}(s_j)$
10:         $\tilde{s}_j = (\bar{x}^{(i)}, I_j, I_{j-1}, \ldots, I_{j-K}, a_{j-1}^d)$
11:         » Sample an action from the policy.
12:         $a_j \sim \pi(\tilde{s}_j, a)$
13:         $s_{j+1} = \textsc{Execute}(s_j, a_j)$
14:         » Compute the approximate gradient.
15:         $\Delta_j \leftarrow \nabla_\theta \log \pi(\tilde{s}_j, a_j) R^{(i)}(s_j, a_j)$
                $+ \lambda \nabla_\theta H(\pi(\tilde{s}_j, \cdot))$
16:         $j += 1$
17:     $\theta \leftarrow \theta + \mu \textsc{Adam}(\frac{1}{j} \sum_{j'=1}^{j} \Delta_{j'})$
18: return $\theta$
```

where $H(\pi(\tilde{s}, \cdot))$ is the entropy of π given the agent context $\tilde{s}$, λ is a hyperparameter that controls the strength of the regularization. While the entropy term delays premature convergence, it does not eliminate it. Similar issues are observed for vanilla policy gradient (Mnih et al., 2016).

Algorithm Algorithm 1 shows our learning algorithm. We iterate over the data T times. In each epoch, for each training example $(\bar{x}^{(i)}, s_1^{(i)}, \bar{e}^{(i)})$, $i = 1 \ldots N$, we perform a rollout using our policy to generate an execution (lines 7 - 16). The length of the rollout is bound by J, but may be shorter if the agent selected the STOP action. At each step j, the agent updates the agent context $\tilde{s}_j$ (lines 9 - 10), samples an action from the policy π (line 12), and executes it to generate the new world state s_{j+1} (line 13). The gradient is approximated using the sampled action with the computed reward $R^{(i)}(s_j, a_j)$ (line 15). Following each rollout, we update the parameters θ with the mean of the gradients using $\textsc{Adam}$ (Kingma and Ba, 2014).

6 Reward Shaping

Reward shaping is a method for transforming a reward function by adding a *shaping term* to the problem reward. The goal is to generate more informative updates by adding information to the reward. We use this method to leverage the training demonstrations, a common form of supervision for training systems that map language to actions. Reward shaping allows us to fully use this type of supervision in a reinforcement learning framework, and effectively combine learning from demonstrations and exploration.

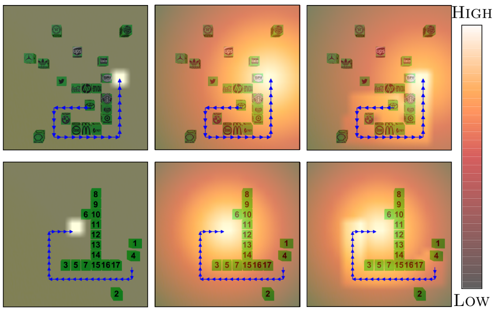


Figure 3: Visualization of the shaping potentials for two tasks. We show demonstrations (blue arrows), but omit instructions. To visualize the potentials intensity, we assume only the target block can be moved, while rewards and potentials are computed for any block movement. We illustrate the sparse problem reward (left column) as a potential function and consider only its positive component, which is focused on the goal. The middle column adds the distance-based potential. The right adds both potentials.

Adding an arbitrary shaping term can change the optimality of policies and modify the original problem, for example by making bad policies according to the problem reward optimal according to the shaped function.[3] Ng et al. (1999) and Wiewiora et al. (2003) outline potential-based terms that realize sufficient conditions for *safe* shaping.[4] Adding a shaping term is safe if the order of policies according to the shaped reward is identical to the order according to the original problem reward. While safe shaping only applies to optimizing the total reward, we show empirically the effectiveness of the safe shaping terms we design in a contextual bandit setting.

We introduce two shaping terms. The final shaped reward is a sum of them and the problem reward. Similar to the problem reward, we define example-specific shaping terms. We modify the reward function signature as required.

Distance-based Shaping (F_1) The first shaping term measures if the agent moved closer to the goal state. We design it to be a safe potential-based

[3] For example, adding a shaping term $F = -R$ will result in a shaped reward that is always 0, and any policy will be trivially optimal with respect to it.

[4] For convenience, we briefly overview the theorems of Ng et al. (1999) and Wiewiora et al. (2003) in Appendix A.

term (Ng et al., 1999):

$$F_1^{(i)}(s_j, a_j, s_{j+1}) = \phi_1^{(i)}(s_{j+1}) - \phi_1^{(i)}(s_j) \ .$$

The potential $\phi_1^{(i)}(s)$ is proportional to the negative distance from the goal state $s_g^{(i)}$. Formally, $\phi_1^{(i)}(s) = -\eta \| s - s_g^{(i)} \|$, where η is a constant scaling factor, and $\|.\|$ is a distance metric. In the block world, the distance between two states is the sum of the Euclidean distances between the positions of each block in the two states, and η is the inverse of block width. The middle column in Figure 3 visualizes the potential $\phi_1^{(i)}$.

Trajectory-based Shaping (F_2) Distance-based shaping may lead the agent to sub-optimal states, for example when an obstacle blocks the direct path to the goal state, and the agent must temporarily increase its distance from the goal to bypass it. We incorporate complete trajectories by using a simplification of the shaping term introduced by Brys et al. (2015). Unlike F_1, it requires access to the previous state and action. It is based on the look-back advice shaping term of Wiewiora et al. (2003), who introduced safe potential-based shaping that considers the previous state and action. The second term is:

$$F_2^{(i)}(s_{j-1}, a_{j-1}, s_j, a_j) = \phi_2^{(i)}(s_j, a_j) - \phi_2^{(i)}(s_{j-1}, a_{j-1}) \ .$$

Given $\bar{e}^{(i)} = \langle (s_1, a_1), \dots, (s_m, a_m) \rangle$, to compute the potential $\phi_2^{(i)}(s, a)$, we identify the closest state s_j in $\bar{e}^{(i)}$ to s. If $\eta \| s_j - s \| < 1$ and $a_j = a$, $\phi_2^{(i)}(s, a) = 1.0$, else $\phi_2^{(i)}(s, a) = -\delta_f$, where δ_f is a penalty parameter. We use the same distance computation and parameter η as in F_1. When the agent is in a state close to a demonstration state, this term encourages taking the action taken in the related demonstration state. The right column in Figure 3 visualizes the effect of the potential $\phi_2^{(i)}$.

7 Experimental Setup

Environment We use the environment of Bisk et al. (2016). The original task required predicting the source and target positions for a single block given an instruction. In contrast, we address the task of moving blocks on the plane to execute instructions given visual input. This requires generating the complete sequence of actions needed to complete the instruction. The environment contains up to 20 blocks marked with logos or digits. Each block can be moved in four directions. Including the STOP action, in each step, the agent selects between 81 actions. The set of actions is constant and is not limited to the blocks present. The transition function is deterministic. The size of each block step is 0.04 of the board size. The agent observes the board from above. We adopt a relatively challenging setup with a large action space. While a simpler setup, for example decomposing the problem to source and target prediction and using a planner, is likely to perform better, we aim to minimize task-specific assumptions and engineering of separate modules. However, to better understand the problem, we also report results for the decomposed task with a planner.

Data Bisk et al. (2016) collected a corpus of instructions paired with start and goal states. Figure 1 shows example instructions. The original data includes instructions for moving one block or multiple blocks. Single-block instructions are relatively similar to navigation instructions and referring expressions. While they present much of the complexity of natural language understanding and grounding, they rarely display the planning complexity of multi-block instructions, which are beyond the scope of this paper. Furthermore, the original data does not include demonstrations. While generating demonstrations for moving a single block is straightforward, disambiguating action ordering when multiple blocks are moved is challenging. Therefore, we focus on instructions where a single block changes its position between the start and goal states, and restrict demonstration generation to move the changed block. The remaining data, and the complexity it introduces, provide an important direction for future work.

To create demonstrations, we compute the shortest paths. While this process may introduce noise for instructions that specify specific trajectories (e.g., *move SRI two steps north and . . .*) rather than only describing the goal state, analysis of the data shows this issue is limited. Out of 100 sampled instructions, 92 describe the goal state rather than the trajectory. A secondary source of noise is due to discretization of the state space. As a result, the agent often can not reach the exact target position. The demonstrations error illustrates this problem (Table 3). To provide task completion reward during learning, we relax the state comparison, and consider states to be equal if the sum of block distances is under the size of one block.

The corpus includes 11,871/1,719/3,177 instructions for training/development/testing. Table 1 shows corpus statistic compared to the commonly used SAIL navigation corpus (MacMahon

	SAIL	Blocks
Number of instructions	3,237	16,767
Mean instruction length	7.96	15.27
Vocabulary	563	1,426
Mean trajectory length	3.12	15.4

Table 1: Corpus statistics for the block environment we use and the SAIL navigation domain.

et al., 2006; Chen and Mooney, 2011). While the SAIL agent only observes its immediate surroundings, overall the blocks domain provides more complex instructions. Furthermore, the SAIL environment includes only 400 states, which is insufficient for generalization with vision input. We compare to other data sets in Appendix D.

Evaluation We evaluate task completion error as the sum of Euclidean distances for each block between its position at the end of the execution and in the gold goal state. We divide distances by block size to normalize for the image size. In contrast, Bisk et al. (2016) evaluate the selection of the source and target positions independently.

Systems We report performance of ablations, the upper bound of following the demonstrations (Demonstrations), and five baselines: (a) STOP: the agent immediately stops, (b) RANDOM: the agent takes random actions, (c) SUPERVISED: supervised learning with maximum-likelihood estimate using demonstration state-action pairs, (d) DQN: deep Q-learning with both shaping terms (Mnih et al., 2015), and (e) REINFORCE: policy gradient with cumulative episodic reward with both shaping terms (Sutton et al., 1999). Full system details are given in Appendix B.

Parameters and Initialization Full details are in Appendix C. We consider $K = 4$ previous images, and horizon length $J = 40$. We initialize our model with the SUPERVISED model.

8 Results

Table 2 shows development results. We run each experiment three times and report the best result. The RANDOM and STOP baselines illustrate the task complexity of the task. Our approach, including both shaping terms in a contextual bandit setting, significantly outperforms the other methods. SUPERVISED learning demonstrates lower performance. A likely explanation is test-time execution errors leading to unfamiliar states with poor later performance (Kakade and Langford, 2002), a form of the covariate shift problem. The low performance of REINFORCE and DQN illustrates the challenge of general reinforcement learning with limited data due to relatively high sample complexity (Kearns et al., 1999; Krishnamurthy et al., 2016). We also report results using ensembles of the three models.

Algorithm	**Distance Error**		**Min. Distance**	
	Mean	**Med.**	**Mean**	**Med.**
Demonstrations	0.35	0.30	0.35	0.30
Baselines				
STOP	5.95	5.71	5.95	5.71
RANDOM	15.3	15.70	5.92	5.70
SUPERVISED	4.65	4.45	3.72	3.26
REINFORCE	5.57	5.29	4.50	4.25
DQN	6.04	5.78	5.63	5.49
Our Approach	3.60	3.09	2.72	2.21
w/o Sup. Init	3.78	3.13	2.79	2.21
w/o Prev. Action	3.95	3.44	3.20	2.56
w/o F_1	4.33	3.74	3.29	2.64
w/o F_2	3.74	3.11	3.13	2.49
w/ Distance Reward	8.36	7.82	5.91	5.70
Ensembles				
SUPERVISED	4.64	4.27	3.69	3.22
REINFORCE	5.28	5.23	4.75	4.67
DQN	5.85	5.59	5.60	5.46
Our Approach	**3.59**	**3.03**	**2.63**	**2.15**

Table 2: Mean and median (Med.) development results.

Algorithm	**Distance Error**		**Min. Distance**	
	Mean	**Med.**	**Mean**	**Med.**
Demonstrations	0.37	0.31	0.37	0.31
STOP	6.23	6.12	6.23	6.12
RANDOM	15.11	15.35	6.21	6.09
Ensembles				
SUPERVISED	4.95	4.53	3.82	3.33
REINFORCE	5.69	5.57	5.11	4.99
DQN	6.15	5.97	5.86	5.77
Our Approach	**3.78**	**3.14**	**2.83**	**2.07**

Table 3: Mean and median (Med.) test results.

We ablate different parts of our approach. Ablations of supervised initialization (our approach w/o sup. init) or the previous action (our approach w/o prev. action) result in increase in error. While the contribution of initialization is modest, it provides faster learning. On average, after two epochs, we observe an error of 3.94 with initialization and 6.01 without. We hypothesize that the F_2 shaping term, which uses full demonstrations, helps to narrow the gap at the end of learning. Without supervised initialization and F_2, the error increases to 5.45 (the 0% point in Figure 4). We observe the contribution of each shaping term and their combination. To study the benefit of potential-based shaping, we experiment with a negative distance-to-goal reward. This reward replaces the problem reward and encourages getting closer to the goal (our approach w/distance reward). With this reward, learning fails to converge, leading to a relatively high error.

Figure 4 shows our approach with varying

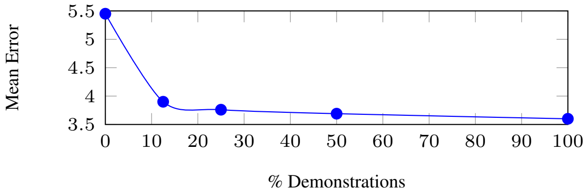


Figure 4: Mean distance error as a function of the ratio of training examples that include complete trajectories. The rest of the data includes the goal state only.

amount of supervision. We remove demonstrations from both supervised initialization and the F_2 shaping term. For example, when only 25% are available, only 25% of the data is available for initialization and the F_2 term is only present for this part of the data. While some demonstrations are necessary for effective learning, we get most of the benefit with only 12.5%.

Table 3 provides test results, using the ensembles to decrease the risk of overfitting the development. We observe similar trends to development result with our approach outperforming all baselines. The remaining gap to the demonstrations upper bound illustrates the need for future work.

To understand performance better, we measure *minimal distance* (min. distance in Tables 2 and 3), the closest the agent got to the goal. We observe a strong trend: the agent often gets close to the goal and fails to stop. This behavior is also reflected in the number of steps the agent takes. While the mean number of steps in development demonstrations is 15.2, the agent generates on average 28.7 steps, and 55.2% of the time it takes the maximum number of allowed steps (40). Testing on the training data shows an average 21.75 steps and exhausts the number of steps 29.3% of the time. The mean number of steps in training demonstrations is 15.5. This illustrates the challenge of learning how to be behave at an absorbing state, which is observed relatively rarely during training. This behavior also shows in our video.[5]

We also evaluate a supervised learning variant that assumes a perfect planner.[6] This setup is similar to Bisk et al. (2016), except using raw image input. It allows us to roughly understand how well the agent generates actions. We observe a mean error of 2.78 on the development set, an improvement of almost two points over supervised learning with our approach. This illustrates the complexity of the complete problem.

We conduct a shallow linguistic analysis to understand the agent behavior with regard to differences in the language input. As expected, the agent is sensitive to unknown words. For instructions without unknown words, the mean development error is 3.49. It increases to 3.97 for instructions with a single unknown word, and to 4.19 for two.[7] We also study the agent behavior when observing new phrases composed of known words by looking at instructions with new n-grams and no unknown words. We observe no significant correlation between performance and new bi-grams and tri-grams. We also see no meaningful correlation between instruction length and performance. Although counterintuitive given the linguistic complexities of longer instructions, it aligns with results in machine translation (Luong et al., 2015).

9 Conclusions

We study the problem of learning to execute instructions in a situated environment given only raw visual observations. Supervised approaches do not explore adequately to handle test time errors, and reinforcement learning approaches require a large number of samples for good convergence. Our solution provides an effective combination of both approaches: reward shaping to create relatively stable optimization in a contextual bandit setting, which takes advantage of a signal similar to supervised learning, with a reinforcement basis that admits substantial exploration and easy avenues for smart initialization. This combination is designed for a few-samples regime, as we address. When the number of samples is unbounded, the drawbacks observed in this scenario for optimizing longer term reward do not hold.

Acknowledgments

This research was supported by a Google Faculty Award, an Amazon Web Services Research Grant, and a Schmidt Sciences Research Award. We thank Alane Suhr, Luke Zettlemoyer, and the anonymous reviewers for their helpful feedback, and Claudia Yan for technical help. We also thank the Cornell NLP group and the Microsoft Research Machine Learning NYC group for their support and insightful comments.

[5] https://github.com/clic-lab/blocks

[6] As there is no sequence of decisions, our reinforcement approach is not appropriate for the planner experiment. The architecture details are described in Appendix B.

[7] This trend continues, although the number of instructions is too low (< 20) to be reliable.

References

Alekh Agarwal, Daniel J. Hsu, Satyen Kale, John Langford, Lihong Li, and Robert E. Schapire. 2014. Taming the monster: A fast and simple algorithm for contextual bandits. In *Proceedings of the International Conference on Machine Learning*.

Jacob Andreas and Dan Klein. 2015. Alignment-based compositional semantics for instruction following. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/D15-1138.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016a. Learning to compose neural networks for question answering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. https://doi.org/10.18653/v1/N16-1181.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016b. Neural module networks. In *Conference on Computer Vision and Pattern Recognition*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *International Journal of Computer Vision*.

Yoav Artzi, Dipanjan Das, and Slav Petrov. 2014a. Learning compact lexicons for CCG semantic parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.3115/v1/D14-1134.

Yoav Artzi, Maxwell Forbes, Kenton Lee, and Maya Cakmak. 2014b. Programming by demonstration with situated semantic parsing. In *AAAI Fall Symposium Series*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics* 1:49–62. http://aclweb.org/anthology/Q13-1005.

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. 2002. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.* 32(1):48–77.

Yonatan Bisk, Deniz Yuret, and Daniel Marcu. 2016. Natural language communication with robots. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. https://doi.org/10.18653/v1/N16-1089.

S.R.K. Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. http://aclweb.org/anthology/P09-1010.

S.R.K. Branavan, Luke Zettlemoyer, and Regina Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. http://aclweb.org/anthology/P10-1129.

Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E. Taylor, and Ann Nowé. 2015. Reinforcement learning from demonstration through shaping. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

Wenhu Chen, Aurélien Lucchi, and Thomas Hofmann. 2016. Bootstrap, review, decode: Using out-of-domain textual data to improve image captioning. *CoRR* abs/1611.05321.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *CoRR* abs/1504.00325.

Kevin Clark and D. Christopher Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. http://aclweb.org/anthology/D16-1245.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science* 14:179–211.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. https://doi.org/10.18653/v1/P16-1153.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. 2016. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR* abs/1612.06890.

Sham Kakade and John Langford. 2002. Approximately optimal approximate reinforcement learning. In *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*.

Michael Kearns, Yishay Mansour, and Andrew Y. Ng. 1999. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *Proeceediings of the International Joint Conference on Artificial Intelligence*.

Joohyun Kim and Raymond Mooney. 2012. Unsupervised PCFG induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. http://aclweb.org/anthology/D12-1040.

Joohyun Kim and Raymond Mooney. 2013. Adapting discriminative reranking to grounded language learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. http://aclweb.org/anthology/P13-1022.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Jens Kober, J. Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research* 32:1238–1274.

Akshay Krishnamurthy, Alekh Agarwal, and John Langford. 2016. PAC reinforcement learning with rich observations. In *Advances in Neural Information Processing Systems*.

John Langford and Tong Zhang. 2007. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. http://aclweb.org/anthology/D16-1127.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. http://aclweb.org/anthology/D15-1166.

Matthew MacMahon, Brian Stankiewics, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, action in route instructions. In *Proceedings of the National Conference on Artificial Intelligence*.

Cynthia Matuszek, Dieter Fox, and Karl Koscher. 2010. Following directions using statistical machine translation. In *Proceedings of the international conference on Human-robot interaction*.

Cynthia Matuszek, Evan Herbst, Luke S. Zettlemoyer, and Dieter Fox. 2012. Learning to parse natural language commands to a robot control system. In *Proceedings of the International Symposium on Experimental Robotics*.

Hongyuan Mei, Mohit Bansal, and R. Matthew Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. https://doi.org/10.18653/v1/N16-1086.

Dipendra K. Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. 2016. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *The International Journal of Robotics Research* 35(1-3):281–300.

Kumar Dipendra Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. 2015. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. https://doi.org/10.3115/v1/P15-1096.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing atari with deep reinforcement learning. In *Advances in Neural Information Processing Systems*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, and Georg Ostrovski. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540).

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/D15-1001.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. http://aclweb.org/anthology/D16-1261.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the International Conference on Machine Learning*.

Junhyuk Oh, Valliappa Chockalingam, Satinder P. Singh, and Honglak Lee. 2016. Control of memory, active perception, and action in minecraft. In *Proceedings of the International Conference on Machine Learning*.

Rohan Paul, Jacob Arkin, Nicholas Roy, and Thomas M. Howard. 2016. Efficient grounding of abstract spatial concepts for natural language interaction with robot manipulators. In *Robotics: Science and Systems*.

Andrei A. Rusu, Matej Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. 2016. Sim-to-real robot learning from pixels with progressive nets. *CoRR* .

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2015. Trust region policy optimization .

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529 7587:484–9.

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. A corpus of compositional language for visual reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Jaeyong Sung, Seok Hyun Jin, and Ashutosh Saxena. 2015. Robobarista: Object part based transfer of manipulation trajectories from crowd-sourcing in 3d pointclouds. In *International Symposium on Robotics Research*.

Richard S. Sutton and Andrew G. Barto. 1998. Reinforcement learning: An introduction. *IEEE Trans. Neural Networks* 9:1054–1054.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis G. Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the National Conference on Artificial Intelligence*.

Adam Vogel and Daniel Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. http://aclweb.org/anthology/P10-1083.

Bonnie Webber, Norman Badler, Barbara Di Eugenio, Christopher Geib, Libby Levison, and Michael Moore. 1995. Instructions, intentions and expectations. *Artificial Intelligence* 73(1):253–269.

Eric Wiewiora, Garrison W. Cottrell, and Charles Elkan. 2003. Principled methods for advising reinforcement learning agents. In *Proceedings of the International Conference on Machine Learning*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8.

Ronald J Williams and Jing Peng. 1991. Function optimization using connectionist reinforcement learning algorithms. *Connection Science* 3(3):241–268.

Terry Winograd. 1972. Understanding natural language. *Cognitive Psychology* 3(1):1–191.

Kelvin Xu, Jimmy Ba, Jamie Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. 2017. Target-driven visual navigation in indoor scenes using deep reinforcement learning.

An analysis of eye-movements during reading for the detection of mild cognitive impairment

Kathleen C. Fraser[1], **Kristina Lundholm Fors**[1], **Dimitrios Kokkinakis**[1], **Arto Nordlund**[2]
[1]The Swedish Language Bank, Department of Swedish
[2]Department of Psychiatry and Neurochemistry
University of Gothenburg, Sweden
{kathleen.fraser, kristina.lundholm, dimitrios.kokkinakis, arto.nordlund}@gu.se

Abstract

We present a machine learning analysis of eye-tracking data for the detection of mild cognitive impairment, a decline in cognitive abilities that is associated with an increased risk of developing dementia. We compare two experimental configurations (reading aloud versus reading silently), as well as two methods of combining information from the two trials (concatenation and merging). Additionally, we annotate the words being read with information about their frequency and syntactic category, and use these annotations to generate new features. Ultimately, we are able to distinguish between participants with and without cognitive impairment with up to 86% accuracy.

1 Introduction

As the global population ages, the prevalence of dementia is increasing (Prince et al., 2013). The term “dementia” refers to an atypical and pathological decline in cognitive abilites, encompassing a range of possible underlying causes. Detecting the onset of dementia as early as possible is important for a number of reasons, including timely access to medication and treatment, increasing support for activities of daily living (such as maintaining proper nutrition and hygiene), reducing the individual’s engagement in potentially risky activities (e.g. driving an automobile), and giving individuals, families, and caregivers time to prepare (Solomon and Murphy, 2005; Ashford et al., 2006; Calzà et al., 2015).

In this study, we investigate the possibility of using eye-tracking data and machine learning to detect early, subtle signs of cognitive impairment. Previous work has suggested that changes in eye movements while reading do occur in Alzheimer’s disease (Lueck et al., 2000; Fernández et al., 2013; Pereira et al., 2014; Biondi et al., 2017). However, our participants do not have a dementia diagnosis; rather, they have been diagnosed with “mild cognitive impairment”, meaning they are starting to show very early signs of cognitive decline, and are at an increased risk of developing dementia. We test the relative merits of collecting eye-tracking data while reading silently and aloud, and explore the idea of augmenting eye-tracking features with linguistic information.

We begin by presenting some background information on cognitive and linguistic changes in dementia, and discuss previous work on eye-tracking and natural language processing approaches to detecting cognitive decline. We then explain our experimental set-up, feature extraction, and machine learning pipeline. We present results for reading silently and reading aloud, and discuss the overall implications and interpretation of our results. Finally we acknowledge the limitations of the current work and suggest areas of future research.

2 Background

There are several different types of dementia, Alzheimer’s disease (AD) being the most common one. AD typically debuts with symptoms related to executive cognitive functioning and memory, but also included are specific linguistic impairments, primarily related to semantic processing. Mild cognitive impairment (MCI) can be seen as a stage of pre-clinical dementia, and may manifest years before an actual dementia diagnosis. Persons with MCI show symptoms across several cognitive domains, where global cognitive ability, episodic memory, perceptual speed, and executive functioning are most clearly affected. However, the performance of persons with MCI over-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1016–1026
Copenhagen, Denmark, September 7–11, 2017.

lap greatly with the performance of healthy controls, which highlights the complexity and heterogeneity of the diagnosis (Bäckman et al., 2005).

Taler and Phillips (2008) reviewed the literature on language impairments in MCI and Alzheimer's disease, and found that the linguistic deficits seen in AD are also present in MCI, albeit to a lesser degree. The main deficits are located on the semantic level (for example, difficulty naming pictures or coming up with words from a particular semantic category), whereas there are no clear evidence of problems regarding syntactic processing. Sentence comprehension is typically impaired in persons with MCI, but there is a great degree of individual variation. Previous research suggests that using tasks that include a possibility to analyse temporal measures (such as reaction time) will improve the ability to distinguish between MCI and healthy controls, and may also be useful as a prognostic factor when investigating which subjects with MCI will convert to AD (Taler and Phillips, 2008).

There has been growing interest in applying machine learning techniques to detect mild cognitive impairment from various linguistic data. Roark et al. (2011) measured the complexity and information content of narrative story re-tellings from 37 participants with MCI and 37 healthy controls, and was able to classify the groups with an AUC of 0.73 using these features alone, or 0.86 by combining this information with clinical test scores. Tóth et al. (2015) leveraged acoustic features (including articulation rate, speech rate, utterance length, pause duration, number of pauses, and hesitation rate) to distinguish between 32 participants with MCI and 19 elderly controls with a best accuracy of 80.4%.

Other research has considered the closely related problem of distinguishing dementia patients from controls through automated analysis of speech and language production (Thomas et al., 2005; Pakhomov et al., 2010; Guinn and Habash, 2012; Meilán et al., 2014; Jarrold et al., 2014; Fraser et al., 2016; Rentoumi et al., 2014; Garrard et al., 2014; Prud'hommeaux and Roark, 2015; Yancheva et al., 2015).

In contrast, computational analyses of language processing and comprehension for the goal of detecting cognitive decline are much rarer, possibly because it is more difficult to quantify automatically. Classical studies of language processing in dementia have considered both listening (for example, Rochon et al. 1994; Kempler et al. 1998; Welland et al. 2002) and reading (for example, Patterson et al. 1994; Storandt et al. 1995); here we focus on reading as the input modality. One well-established method for estimating the processing demands during reading comprehension is through eye-tracking. There is a vast literature on eye-tracking in reading which we will not attempt to fully summarize here, but merely introduce some key vocabulary and basic concepts.

When reading, the eye moves through the text in a series of *fixations* and *saccades*. A fixation occurs when the eye temporarily rests on a word. This time is used to process the incoming information, and to plan the next eye movement. Fixations typically last for around 200-300 ms, and are on average slightly longer in oral reading than in silent reading (Rayner, 1998). In between fixations, the eye makes a rapid movement called a *saccade*. Saccades can move the eye forward through the text (a *forward saccade*) or backward (a *saccadic regression* or simply a *regression*). Saccades tend to be around 6-8 characters in size in English (although this is language-dependent; for example, Liversedge et al. (2016) found longer saccades in Finnish and shorter saccades in Chinese), and around 10-15% of saccades in reading are regressions. Both strong and poor readers make regressions, but stronger readers seem to have the ability to accurately direct their eyes back to a difficult or ambiguous passage, whereas weaker readers perform more general back-tracking (Murray and Kennedy, 1988).

Whether a word is fixated on, and for how long, is influenced by a number of word-level and contextual factors. Content words are fixated on approximately 85% of the time, while function words are fixated on only 35% of the time (Rayner, 1998). There is some evidence that word type effects may be even more fine-grained, as work by Barrett et al. (2016) demonstrates the possibility for part-of-speech tagging based on eye-tracking information. The number and duration of fixations is also affected by word frequency (Raney and Rayner, 1995), word predictability in context (Kliegl et al., 2004), the position of the word in the sentence (Rayner et al., 2000), the emotional valence of the word (Scott et al., 2012), and word length (Rayner, 1998).

While sharing several features, silent reading and reading aloud are believed to potentially differ in some ways. The main division between the two types of reading is related to the access of phonological and semantic representations in the brain. In silent reading, there has been a great deal of discussion on whether the decoding of orthographic information is directly mapped to semantic meaning, or whether letters are mapped to phonemes, which are then connected to semantic meaning. By using a computational approach based on previous research about reading, Harm and Seidenberg (2004) investigate the two proposed routes and suggest a combined model, where the phonological path and direct path are simultaneously activated and share the workload depending on factors such as word frequency and spelling-sound consistency. The activation of semantic information during reading aloud is also a matter that has been discussed and researched for some time. It was previously thought that during reading aloud, the semantic level of information did not need to be activated, but rather letters could be matched directly to phonemes and then articulated. However, computational models (Coltheart et al., 2001) and for example fMRI data (Graves et al., 2010) have shown that semantic processing is involved in reading aloud, but to varying degrees.

Previous work has identified differences between the eye-movements of individuals with cognitive impairment relative to healthy controls. Lueck et al. (2000) reported that participants with AD had more irregular eye movements when reading, longer fixation times, and more saccadic regressions. Fernández et al. (2013) found that participants with AD had an increased number of fixations and regressions, and also skipped more words than healthy controls. Pereira et al. (2014) presented a review of the literature on eye-tracking in MCI and AD, and suggested that such techniques may be able to predict the conversion from MCI to AD, partly due to the sensitivity of eye-movements to early changes in memory, visual, and executive processes.

Earlier this year, in a paper posted on arXiv, Biondi et al. (2017) reported a classification accuracy of 88.3% in distinguishing between participants with AD and healthy controls through eye-tracking measures. They recorded eye movements from 40 healthy elderly adults and 20 AD patients while they read 120 sentences. The sentences varied in terms of predictability and familiarity (for example, some of the sentences were well-known proverbs). Each sentence was recorded as a separate trial. After removing 10% of the trials as outliers, 90% of the remaining trials were used to train a deep sparse-autoencoder, and 10% were reserved as test data. It is assumed that some of the training data and test data originated from the same participants.

In this paper, we first aim to reproduce aspects of the Biondi et al. (2017) study, although with some notable differences. Our study was conducted in Swedish, rather than Spanish, and in each trial the participant was presented with an entire paragraph, rather than individual sentences, which affects our feature calculations and choice of classifiers. Additionally, we present a comparison of two different trial configurations (reading silently versus reading aloud), and introduce new word-level features to associate linguistic information with the eye-tracking features. Furthermore, perhaps the most critical difference from a clinical standpoint is that our participants are in a milder stage of cognitive decline, and have not received AD diagnoses. Thus we aim explore whether this promising approach can be used to detect the earliest stages of cognitive impairment.

3 Methods

3.1 Participants

The participants were recruited from the Gothenburg MCI study, which is a large longitudinal study on mild cognitive impairment (Wallin et al., 2016). The overall Gothenburg MCI study is approved by the local ethical committee review board (reference number: L09199, 1999; T479-11, 2011); while the currently described study was approved by the local ethical committee decision 206-16, 2016.

To be included in this study, the participants had to fulfill certain inclusion and exclusion criteria: participants had to be native Swedish speakers and had to be able to read and understand information about the project, and be able to give consent. Participants could not have dyslexia or other reading difficulties not relating to their current cognitive impairment. We also excluded patients with deep depression, ongoing substance abuse, poor vision that cannot be corrected with glasses or contact lenses, and participants that were diagnosed with other serious psychiatric, neurological or brain-related diseases, such as Parkinson's dis-

	MCI ($n = 27$)	HC ($n = 30$)
Age (years)	70.3 (5.8)	68.0 (7.5)
Education (years)	14.2 (3.6)	13.3 (3.7)
Sex (M/F)	13/14	9/21
MMSE	28.2 (1.3)	29.6 (0.6)

Table 1: Demographic information for participants with mild cognitive impairment (MCI) and healthy controls (HC). Age, education, and Mini-Mental State Exam (MMSE) scores are given in the format: mean (standard deviation). The MMSE is a general test of cognitive status and has a maximum score of 30.

ease, amyotrophic lateral sclerosis, brain tumour or stroke. Three groups of participants took part in the study: persons with mild cognitive impairment (MCI), persons with subjective cognitive impairment (SCI), and healthy controls (HC). Participants have all been assessed with a battery of tests, from neuropsychological examinations to structural MRI, blood tests, and lumbar punctures. The groups analysed and compared in this paper are the MCI group and the control group. Six control participants and five MCI participants were excluded from the current analysis as a result of calibration problems with the eye-tracker (e.g. due to cataracts or eye inflammation).

Participant information can be seen in Table 1. There is no significant difference between the groups on age or education. The controls do have significantly higher Mini-Mental State Exam (MMSE) scores, on average ($p < 0.0001$). However, we note that the average MMSE score for our MCI participants is 28.2 (out of 30), which is considered to be “normal” (Grut et al., 1993). We contrast this with the AD participants in the study by Biondi et al. (2017), who had an average MMSE score of 24.2. In fact, the healthy control participants in that study had an average MMSE of 27.8, very similar to our MCI group. This indicates the subtle nature of the impairment seen in the MCI category.

3.2 Eye-tracking experiments

The eye-tracking experiments were carried out in a quiet lab environment. We used an EyeLink 1000 Desktop Mount with monocular eye-tracking, and used a headrest for head stabilization. Head stabilization provides an increased eye-tracking performance. The sampling rate was set to 1000 Hz.

The participants read two short texts, and after each text they answered five questions about the texts. The first text was read silently, while the second text was read aloud. Both texts were taken from the International Reading Speed Texts (IReST), which is a collection of texts that is available in 17 different languages. They are 146 words long in Swedish, and were developed to be used as an evaluation tool for impairments in vision or reading ability (Trauzettel-Klosinski et al., 2012). We chose to present complete paragraphs (rather than individual sentences) to simulate a more natural reading task, requiring the integration and recollection of information from the beginning through to the end of the paragraph.

Areas of interest (AOIs) were defined in the text, and each word was labeled as a separate AOI. Eye movements, such as saccades and fixations, are then calculated with respect to the predefined AOIs. Fixations occurring outside the AOIs are not considered in this analysis.

The eye-tracker was calibrated for each participant using a 9-point calibration procedure, and drift-corrected between Trial 1 and Trial 2. However, visual inspection of the data revealed a tendency for downward drift, particularly in the second trial. This was corrected manually, where necessary, to the degree agreed upon by two of the authors (K.C.F. and K.L.F.).

3.3 Features

As our baseline, we consider the 13 features presented in Biondi et al. (2017), and summarized in Table 2. Duration and amplitude features were log-transformed before computing the mean and standard deviation (Wotschack, 2009). The first fixation of each trial is discarded, and analysis starts from the second fixation (Holmqvist et al., 2011). As in Biondi et al. (2017), we partition the fixations into 4 categories: first-pass first fixations, later-pass first fixations[1], multi-fixations, and refixations. These definitions are given in Table 2, but for the sake of clarity we also present a simple truth table summarizing the four types of fixations in Table 3.

We then augment these baseline features with information about the words in the text, namely their frequency and word type. We first perform basic syntactic and morphological analysis of the

[1] Biondi et al. (2017) refer to these as “unique” fixations, but this terminology could be ambiguous and thus we have avoided it here.

Gaze duration (mean and s.d.) The mean and standard deviation of the length of time spent fixating on a word, averaged over all words in a trial.

Saccade amplitude (mean and s.d.) The mean and standard deviation of the amplitude of the saccades, averaged over all saccades in a trial.

Total fixations The total number of fixations in a trial.

Total first-pass first fixations The total number of first fixations occurring in the first pass of a trial. That is, a first-pass first fixation occurs when it is the first fixation on the given word, and there have been no fixations on any words occurring later in the text.

Total later-pass first fixations The total number of first fixations occurring outside the first-pass of a trial. That is, a later-pass first fixation occurs when it is the first fixation on the given word, but there have already been fixations on words occurring later in the text.

Total multi-fixations The total number of fixations on a word in the first-pass, excluding the first fixation. That is, a multi-fixation occurs when a word is fixated on multiple times in the run which starts with a first-pass first fixation.

Total re-fixations The total number of fixations on a word outside the first pass, excluding the first fixation.

First-pass first fixation duration (mean and s.d.) The mean and standard deviation of the duration of the first-pass first fixations.

Later-pass first fixation duration (mean and s.d.) The mean and standard deviation of the duration of the later-pass first fixations.

Table 2: Eye-movement features.

		Have any later words been visited?	
		No	Yes
Has this word been visited yet?	No	First-pass first fixation	Later-pass first fixation
	Yes	Multi-fixation	Re-fixation

Table 3: Four types of fixations

two texts using the Sparv annotation tool[2] (Borin et al., 2016). Specifically, each word was lemmatized and labeled with its part-of-speech (POS).

We assign a frequency value for each word lemma according to the number of times it occurs (per one million words) in the "Modern" language section of the Korp Swedish language corpus[3], which contained 10.7 billion word tokens at the time of writing (Borin et al., 2012). These frequency values are POS-disambiguated. We then partition the frequency values into *high* and *low* frequencies, with a threshold of 20 occurrences per million words. This threshold was chosen manually by observing the frequency distribution of the words in the two texts. We also partition the POS labels into two categories: *content* words and *function* words. Content words are defined as nouns, verbs, adjectives, and adverbs; everything else is considered to be a function word.

We then define an augmented feature set, hereafter *Biondi+word*, which takes into account these word-level annotations. Specifically, we create new features corresponding to each of the fixation-based baseline features. (The original feature set also includes saccade amplitude, the computation of which is not attached to any one particular word.) When the original feature involves a mean and standard deviation, we compute the ratio of those values computed on the low:high frequency words and the content:function words. To give an example, for "mean gaze duration", we compute the ratio of the mean gaze duration on low frequency words to mean gaze duration on high frequency words, and the ratio of gaze duration on content words to gaze duration on function words. When the original feature is a raw count, we compute a proportion instead. So for "total fixations", we compute the proportion of total fixations which occur on low-frequency words, and the proportion of total fixations which occur on content words. In this way we define 22 new features to augment the

[2] https://spraakbanken.gu.se/eng/research/infrastructure/sparv

[3] https://spraakbanken.gu.se/eng/korp-info

original Biondi set.

Clearly, we expect these new features to be somewhat correlated with each other, since function words tend also be high-frequency words. However, many content words are also labeled as high-frequency in our methodology, such as *bil* (English: *car*) and *potatis* (English: *potato*).

3.4 Classification framework

We consider three classification algorithms: naïve Bayes (NB), support vector machine (SVM), and logistic regression (LR), implemented in WEKA Version 3.9.1 (Hall et al., 2009). Given the small size of our data set, we forego parameter optimization and use the default parameters; i.e., for LR we use a ridge regression parameter of 10^{-8}, and for SVM we use a first degree polynomial kernel and a complexity parameter of 1.0. For feature selection, we use a wrapper method with a NB classifier. We evaluate the classifier using leave-one-out cross validation, in which at every iteration one data point is held out as a test point, and all remaining points are used for feature selection and classifier training. We report the average classification accuracy across folds. For our dataset, the majority class baseline is 52.6%.

4 Results

4.1 Individual trials

We first consider each trial individually, as we expect there may be differences in eye-movements when reading silently (Trial 1) versus reading aloud (Trial 2). The results for each classifier and each feature set for the first trial are given in Table 4a. Using the augmented feature set hurts classification accuracy in all cases, and the best accuracy of 75.4% is achieved using the naïve Bayes classifier and the Biondi feature set.

When using the data from Trial 2 (Table 4b), the augmented feature set again leads to lower accuracies in all cases, and the best result of 66.7% is achieved by the SVM and naïve Bayes classifiers with the Biondi feature set. In every case, we observe that the classification accuracies are the same or worse on Trial 2 compared to Trial 1. That is, we are able to extract less diagnostically-useful information when the participant is reading aloud than when they are reading silently. This makes sense, since reading aloud is a more constrained task: the reader must keep moving forward at a reasonable pace to avoid disruptions in the spoken narrative. This limits the opportunity for the eyes to move freely around the text. Furthermore, in the reading aloud paradigm, the examiner presented the comprehension questions as soon as the participant had reached the end of the text, in contrast to the silent reading paradigm, in which the participants themselves indicated when they were ready for the questions to be displayed.

	SVM	NB	LR
Biondi	66.7	75.4	73.6
Biondi+word	64.9	71.9	68.4

(a) Trial 1: Reading silently

	SVM	NB	LR
Biondi	66.7	66.7	64.9
Biondi+word	63.1	64.9	63.1

(b) Trial 2: Reading aloud

Table 4: Classifier accuracies for individual trials.

4.2 Combining the trials

We now examine whether we can combine information from the two trials to improve classification accuracy. We consider two different methods for combining the data: (1) concatenating the feature vectors from each trial, and (2) computing the features across both trials, as if they are simply two halves of a single trial. The first method has the advantage of preserving any salient differences between the two experimental paradigms (e.g. if a feature is relevant only when reading silently, that signal will remain in the data). The second method, which we will refer to as *merging*, has the benefit of essentially doubling the amount of data used to compute each feature, possibly leading to more accurate estimates.

The results for each combination are given in Table 5. In most cases, the best accuracy is achieved using the Biondi feature set alone. However, the highest accuracy is 86.0%, which occurs in the merged configuration using the naïve Bayes classifier with the Biondi+word feature set. In every case, a higher accuracy is achieved by merging, rather than concatenating, the data.

4.3 Classification summary

Figure 1 shows the results for each trial and feature set, averaged over the three classifiers. In general, the classifiers trained on Trial 2 did worse than those trained on Trial 1. Concatenating the feature vectors from the two trials resulted in better accu-

	SVM	NB	LR
Biondi	64.9	73.7	68.4
Biondi+word	63.2	73.7	66.7

(a) Concatenated trials

	SVM	NB	LR
Biondi	84.2	82.5	78.9
Biondi+word	84.2	86.0	77.2

(b) Merged trials

Table 5: Classifier accuracies for combined trials.

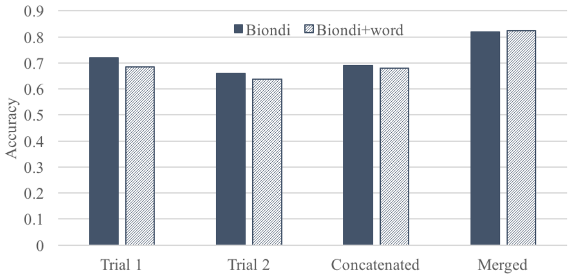


Figure 1: Average accuracies for each trial and feature set, averaged across classifiers.

racies than using Trial 2 data alone, but marginally worse accuracies than using Trial 1 data alone. The best results were achieved by merging the data from the two trials. Using the Biondi feature set alone did better than using the augmented feature set in the first three cases, but the Biondi+word feature set led to slightly higher accuracies in the merged configuration.

However, not all of the observed trends are statistically significant. A 2-way ANOVA revealed a significant effect of trial ($p = 5.0 \times 10^{-7}$) but not feature set on classification accuracy. A Tukey post-hoc test determined that the accuracies in the merged trials are significantly better than in Trial 1 ($p = 6.8 \times 10^{-4}$), Trial 2 ($p = 4.0 \times 10^{-7}$), and the concatenated trials ($p = 1.2 \times 10^{-5}$). However, there is no significant difference between Trial 1 and Trial 2, nor between either of those trials and the concatenated trials.

4.4 Feature analysis

To determine which features help distinguish between the groups, we perform a two-tailed heteroscedastic t-test on all of the features, with Bonferroni correction for repeated comparisons. For this analysis, we consider data from the merged trials, since they led to the best accuracies. Only two features were found to be significantly different between the groups after correction; these are given in Table 6. Consistent with the classification results, none of the frequency or word type features are significant. The total number of first-pass first fixations is significantly higher in the control group but, in contrast, the number of later-pass first fixations is higher in the MCI group. This suggests that the controls have a greater tendency to read through the text from start-to-finish, while the MCI participants tend to skip over words and then return to them later. An example of these different reading patterns can be seen in Figure 2. While this figure only shows data for two participants, it is interesting to note that there is a qualitatively greater difference on the silent trial (Figure 2a and Figure 2c) than in the reading aloud trial (Figure 2b and Figure 2d).

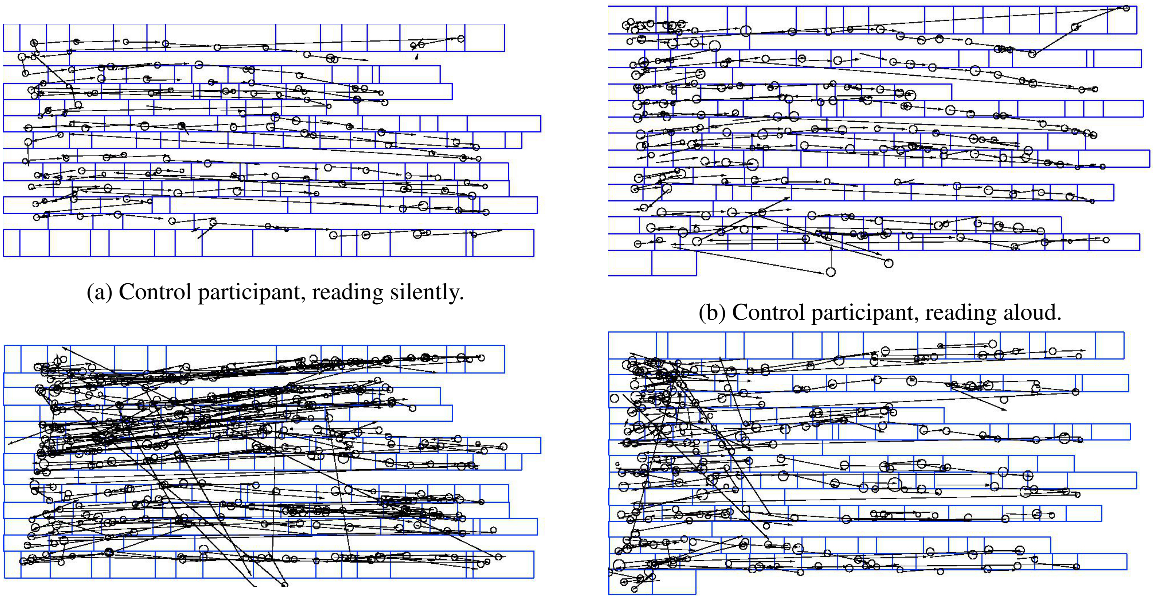
Feature	HC mean	MCI mean	p
First-pass first fixations	98.9	69.1	5.2×10^{-4}
Later-pass first fixations	100.9	133.7	5.8×10^{-6}

Table 6: Features which differ significantly between the groups.

Fernández et al. (2013) found that participants with AD had an increased number of total fixations, first-pass fixations, and second-pass fixations. However, they noted that the second-pass fixations showed an even more striking increase than first-pass fixations. Our results are consistent with this notable increase in second-pass fixations, but not with the reported increase in first pass fixations. One potential reason for this discrepancy could lie in the definition of "first pass fixations", which in Fernández et al. (2013) is given as "the initial reading consisting of all forward fixations on a word", while second-pass fixations are defined as "re-reading"; it is possible that our later-pass first fixations could be classified as first pass fixations under this framework. Nonetheless, both the Fernandez study and our current results suggest a pattern of skipping and back-tracking that is not seen in the control data.

5 Limitations

In this study, as in many studies involving clinical data, our sample is rather small. Furthermore, the two texts were not particularly difficult to read,



(a) Control participant, reading silently.

(b) Control participant, reading aloud.

(c) MCI participant, reading silently.

(d) MCI participant, reading aloud.

Figure 2: Examples of eye movements from a cognitively healthy participant (top) and an MCI participant (bottom), as they read the text from Trial 1 silently (left) and the text from Trial 2 aloud (right). Each blue box in the figures represents an AOI (i.e. a word in the text); the circles indicate fixations and the lines show the movements of the eye. Figure (a) illustrates an example of a relatively straightforward path through the Text 1, while Figure (c) shows one containing more backtracking and re-reading.

nor did they specifically contain words that might be difficult to people with cognitive impairment (for example, low-frequency words with irregular pronunciation, as in Patterson et al. 1994). Additionally, some data had to be either adjusted or in some cases excluded altogether due to calibration quality.

6 Conclusions and future work

In this analysis, we found that we can use eye-tracking information to distinguish between MCI participants and controls with over 80% accuracy, and up to 86% accuracy in the best case. As expected, this is somewhat lower than the accuracy for distinguishing between controls and AD participants reported in Biondi et al. (2017), but demonstrates that eye-tracking may hold promise as a method for detecting the earliest stages of cognitive decline.

We also found that tracking eye movements while the participant reads silently provides more diagnostic information than when reading aloud. Merging data from the two trial conditions led to a significant increase in classification accuracy, compared to using either trial alone. In the merged data set, significant differences between the participant groups were observed for the number of first-pass first fixations (higher in the control group) and later-pass first fixations (higher in the MCI group), suggesting a somewhat disorganized and non-linear path through the text.

Although annotating fixations with the frequency and syntactic category of the word on which the fixation occurs did ultimately lead to the highest classification accuracy, this improvement was not statistically significant, and none of the augmented features showed a significant difference between the HC and MCI groups. It may be that the participants were too early in their decline (and the texts too linguistically simple) for any effect to be seen, or it could be that these variables are not capturing the most relevant linguistic information. In particular, the features were very coarse, making only a binary distinction between high/low frequency words and function/content words. One avenue for future research will be to design more sophisticated ways of incorporating linguistic information into the eye-tracking model, especially features that take into account context, rather than operating at the single word level.

Another untapped source of information is the acoustic signal in the reading aloud trial. Correlating eye movements with acoustic information, such as pauses, fillers, hesitations, and word errors may provide a more complete representation of cognitive processing while reading. Furthermore, other eye-tracking features in addition to those included in the Biondi study may prove to be more sensitive to early cognitive impairment.

In future work we also plan to explore the connection between eye movements and reading comprehension. Each participant in this study also answered comprehension questions related to the passages they read. Analysing the relationship between different eye movement features and the accuracy of the responses may help us better understand the reading strategies used by healthy and cognitively impaired readers.

Finally, future work will include the subjective-cognitive impairment (SCI) group in the analysis. These participants score normally on neuropsychological tests, and so a reliable method for distinguishing them from healthy controls could help provide an early warning system, even before clinical symptoms develop.

Acknowledgments

This work has received support from *Riksbankens Jubileumsfond* - The Swedish Foundation for Humanities and Social Sciences, through the grant agreement no: NHS 14-1761:1.

References

J. Wesson Ashford, Soo Borson, Ruth O'Hara, Paul Dash, Lori Frank, Philippe Robert, William R. Shankle, Mary C. Tierney, Henry Brodaty, Frederick A. Schmitt, Helena C. Kraemer, and Herman Buschke. 2006. Should older adults be screened for dementia? *Alzheimer's & Dementia*, 2(2):76–85.

Lars Bäckman, Sari Jones, Anna-Karin Berger, Erika Jonsson Laukka, and Brent J Small. 2005. Cognitive impairment in preclinical Alzheimer's disease: A meta-analysis. *Neuropsychology*, 19(4):520–531.

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

Juan Biondi, Gerardo Fernandez, Silvia Castro, and Osvaldo Agamenonni. 2017. Eye-movement behavior identification for AD diagnosis. *arXiv preprint arXiv:1702.00837*.

Lars Borin, Markus Forsberg, Martin Hammarstedt, Dan Rosen, Roland Schäfer, and Anne Schumacher. 2016. Sparv: Språkbanken's corpus annotation pipeline infrastructure. In *The Sixth Swedish Language Technology Conference (SLTC), Umeå University, 17-18 November*.

Lars Borin, Markus Forsberg, and Johan Roxendal. 2012. Korp - the corpus infrastructure of Språkbanken. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*, pages 474–478.

Laura Calzà, Daniela Beltrami, Gloria Gagliardi, Enrico Ghidoni, Norina Marcello, Rema Rossini-Favretti, and Fabio Tamburini. 2015. Should we screen for cognitive decline and dementia? *Maturitas*, 82(1):28–35.

Max Coltheart, Kathleen Rastle, Conrad Perry, Robyn Langdon, Johannes Ziegler, Sally Andrews, Rita Berndt, Derek Besner, Anne Castles, Veronika Coltheart, Martin Davies, Colin Davis, Ken Forster, Carol Fowler, Ram Frost, Jonathan Harrington, Arthur Jacobs, Sachiko Ki-noshita, Ken Paap, Karalyn Patterson, David Plaut, Karen Smith-Lock, Marcus Taft, Anna Woollams, Marco Zorzi We also thank Alice Coltheart, Robert Mannell, Steve Saunders, and Carl Windhorst. 2001. DRC: A Dual Route Cascaded Model of Visual Word Recognition and Reading Aloud. *Psychological Review*, 108:204–256.

Gerardo Fernández, Pablo Mandolesi, Nora P Rotstein, Oscar Colombo, Osvaldo Agamennoni, and Luis E Politi. 2013. Eye movement alterations during reading in patients with early Alzheimer disease. *Investigative ophthalmology & Visual Science*, 54(13):8345–8352.

Kathleen C Fraser, Jed A Meltzer, and Frank Rudzicz. 2016. Linguistic features identify Alzheimer's disease in narrative speech. *Journal of Alzheimer's Disease*, 49(2):407–422.

Peter Garrard, Vassiliki Rentoumi, Benno Gesierich, Bruce Miller, and Maria Luisa Gorno-Tempini. 2014. Machine learning approaches to diagnosis and laterality effects in semantic dementia discourse. *Cortex*, 55:122–129.

William W Graves, Rutvik Desai, Colin Humphries, Mark S Seidenberg, and Jeffrey R Binder. 2010. Neural Systems for Reading Aloud: A Multiparametric Approach. *Cerebral Cortex August*, 20:1799–1815.

Michaela Grut, L Fratiglioni, M Viitanen, and B Winblad. 1993. Accuracy of the Mini-Mental Status Examination as a screening test for dementia in a Swedish elderly population. *Acta Neurologica Scandinavica*, 87(4):312–317.

Curry I. Guinn and Anthony Habash. 2012. Language analysis of speakers with dementia of the

Alzheimer's type. In *AAAI Fall Symposium: Artificial Intelligence for Gerontechnology*, pages 8–13.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutmann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *ACM Special Interest Group on Knowledge Discovery and Data Mining Explorations*, 2(1).

Michael W Harm and Mark S Seidenberg. 2004. Computing the meanings of words in reading: Cooperative division of labor between visual and phonological processes. *Psychological Review*, 111(3):662–720.

Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, and Joost Van de Weijer. 2011. *Eye tracking: A comprehensive guide to methods and measures*. Oxford University Press.

William Jarrold, Bart Peintner, David Wilkins, Dimitra Vergryi, Colleen Richey, Maria Luisa Gorno-Tempini, and Jennifer Ogar. 2014. Aided diagnosis of dementia type through computer-based analysis of spontaneous speech. In *Proceedings of the ACL Workshop on Computational Linguistics and Clinical Psychology*, pages 27–36.

Daniel Kempler, Amit Almor, and Maryellen C MacDonald. 1998. Teasing apart the contribution of memory and language impairments in Alzheimer's disease: An online study of sentence comprehension. *American Journal of Speech-Language Pathology*, 7(1):61–67.

Reinhold Kliegl, Ellen Grabner, Martin Rolfs, and Ralf Engbert. 2004. Length, frequency, and predictability effects of words on eye movements in reading. *European Journal of Cognitive Psychology*, 16(1-2):262–284.

Simon P Liversedge, Denis Drieghe, Xin Li, Guoli Yan, Xuejun Bai, and Jukka Hyönä. 2016. Universality in eye movements and reading: A trilingual investigation. *Cognition*, 147:1–20.

Kristin L Lueck, Mario F Mendez, and Kent M Perryman. 2000. Eye movement abnormalities during reading in patients with Alzheimer disease. *Cognitive and Behavioral Neurology*, 13(2):77–82.

Juan José G Meilán, Francisco Martínez-Sánchez, Juan Carro, Dolores E López, Lymarie Millian-Morell, and José M Arana. 2014. Speech in Alzheimer's Disease: Can temporal and acoustic parameters discriminate dementia? *Dementia and Geriatric Cognitive Disorders*, 37(5-6):327–334.

Wayne S Murray and Alan Kennedy. 1988. Spatial coding in the processing of anaphor by good and poor readers: Evidence from eye movement analyses. *The Quarterly Journal of Experimental Psychology*, 40(4):693–718.

Serguei V.S. Pakhomov, Glen E. Smith, Susan Marino, Angela Birnbaum, Neill Graff-Radford, Richard Caselli, Bradley Boeve, and David D. Knopman. 2010. A computerized technique to asses language use patterns in patients with frontotemporal dementia. *Journal of Neurolinguistics*, 23:127–144.

Karalyn E Patterson, Naida Graham, and John R Hodges. 1994. Reading in dementia of the Alzheimer type: A preserved ability? *Neuropsychology*, 8(3):395.

Marta LG Pereira, Marina von Zuben A Camargo, Ivan Aprahamian, and Orestes V Forlenza. 2014. Eye movement analysis and cognitive processing: detecting indicators of conversion to Alzheimer's disease. *Neuropsychiatric Disease and Treatment*, 10:1273–1285.

Martin Prince, Renata Bryce, Emiliano Albanese, Anders Wimo, Wagner Ribeiro, and Cleusa P. Ferri. 2013. The global prevalence of dementia: A systematic review and metaanalysis. *Alzheimer's & Dementia*, 9(1):63–75.

Emily Prud'hommeaux and Brian Roark. 2015. Graph-based word alignment for clinical language evaluation. *Computational Linguistics*, 41(4):549–578.

Gary E Raney and Keith Rayner. 1995. Word frequency effects and eye movements during two readings of a text. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 49(2):151.

Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124(3):372–422.

Keith Rayner, Gretchen Kambe, and Susan A Duffy. 2000. The effect of clause wrap-up on eye movements during reading. *The Quarterly Journal of Experimental Psychology: Section A*, 53(4):1061–1080.

Vassiliki Rentoumi, Ladan Raoufian, Samrah Ahmed, Celeste A. de Jager, and Peter Garrard. 2014. Features and machine learning classification of connected speech samples from patients with autopsy proven Alzheimer's disease with and without additional vascular pathology. *Journal of Alzheimer's Disease*, 42(S3):S3–S17.

Brian Roark, Margaret Mitchell, John-Paul Hosom, Kristy Hollingshead, and Jeffery Kaye. 2011. Spoken language derived measures for detecting mild cognitive impairment. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2081–2090.

Elizabeth Rochon, Gloria S Waters, and David Caplan. 1994. Sentence comprehension in patients with Alzheimer's disease. *Brain and Language*, 46(2):329–349.

Graham G Scott, Patrick J O'Donnell, and Sara C Sereno. 2012. Emotion words affect eye fixations during reading. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 38(3):783–792.

Paul R. Solomon and Cynthia A. Murphy. 2005. Should we screen for Alzheimer's disease? *Geriatrics*, 60(11):26–31.

Martha Storandt, Katherine Stone, and Emily LaBarge. 1995. Deficits in reading performance in very mild dementia of the Alzheimer type. *Neuropsychology*, 9(2):174.

Vanessa Taler and Natalie A Phillips. 2008. Language performance in Alzheimer's disease and mild cognitive impairment: a comparative review. *Journal of clinical and experimental neuropsychology*, 30(5):501–556.

Calvin Thomas, Vlado Keselj, Nick Cercone, Kenneth Rockwood, and Elissa Asp. 2005. Automatic detection and rating of dementia of Alzheimer type through lexical analysis of spontaneous speech. In *Proceedings of the IEEE International Conference on Mechatronics and Automation*, pages 1569–1574.

László Tóth, Gábor Gosztolya, Veronika Vincze, Ildikó Hoffmann, and Gréta Szatlóczki. 2015. Automatic detection of mild cognitive impairment from spontaneous speech using ASR. In *Proceedings of INTERSPEECH*, pages 2694–2698. ISCA.

Susanne Trauzettel-Klosinski, Klaus Dietz, Dürrwächter U, Sokolov AN, Reinhard J, and Klosinski G. 2012. Standardized Assessment of Reading Performance: The New International Reading Speed Texts IReST. *Investigative Opthalmology & Visual Science*, 53(9):5452.

Anders Wallin, Arto Nordlund, Michael Jonsson, Karin Lind, Åke Edman, Mattias Göthlin, Jacob Stålhammar, Marie Eckerström, Silke Kern, Anne Börjesson-Hanson, Mårten Carlsson, Erik Olsson, Henrik Zetterberg, Kaj Blennow, Johan Svensson, Annika Öhrfelt, Maria Bjerke, Sindre Rolstad, and Carl Eckerström. 2016. The Gothenburg MCI study: Design and distribution of Alzheimer's disease and subcortical vascular disease diagnoses from baseline to 6-year follow-up. *Journal of Cerebral Blood Flow and Metabolism : Official journal of the International Society of Cerebral Blood Flow and Metabolism*, 36(1):114–31.

Richard J Welland, Rosemary Lubinski, and D Jeffery Higginbotham. 2002. Discourse comprehension test performance of elders with dementia of the Alzheimer type. *Journal of Speech, Language, and Hearing Research*, 45(6):1175–1187.

Christiane Wotschack. 2009. *Eye Movements in Reading Strategies: How Reading Strategies Modulate Effects of Distributed Processing and Oculomotor Control*. Ph.D. thesis, University of Postdam.

Maria Yancheva, Kathleen Fraser, and Frank Rudzicz. 2015. Using linguistic features longitudinally to predict clinical scores for Alzheimer's disease and related dementias. In *Proceedings of the 6th Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)*, pages 134–140.

A Structured Learning Approach to Temporal Relation Extraction

Qiang Ning[1] and **Zhili Feng** [2] and **Dan Roth** [1,2]
[1]Department of Electrical and Computer Engineering
[2]Department of Computer Science
University of Illinois, Urbana, IL 61801
{qning2,zfeng6,danr}@illinois.edu

Abstract

Identifying temporal relations between events is an essential step towards natural language understanding. However, the temporal relation between two events in a story depends on, and is often dictated by, relations among other events. Consequently, effectively identifying temporal relations between events is a challenging problem even for human annotators. This paper suggests that it is important to take these dependencies into account while learning to identify these relations and proposes a structured learning approach to address this challenge. As a byproduct, this provides a new perspective on handling missing relations, a known issue that hurts existing methods. As we show, the proposed approach results in significant improvements on the two commonly used data sets for this problem.

1 Introduction

Understanding temporal information described in natural language text is a key component of natural language understanding (Mani et al., 2006; Verhagen et al., 2007; Chambers et al., 2007; Bethard and Martin, 2007) and, following a series of TempEval (TE) workshops (Verhagen et al., 2007, 2010; UzZaman et al., 2013), it has drawn increased attention. Time-slot filling (Surdeanu, 2013; Ji et al., 2014), storyline construction (Do et al., 2012; Minard et al., 2015), clinical narratives processing (Jindal and Roth, 2013; Bethard et al., 2016), and temporal question answering (Llorens et al., 2015) are all explicit examples of temporal processing.

The fundamental tasks in temporal processing, as identified in the TE workshops, are 1) time expression (the so-called "timex") extraction and normalization and 2) temporal relation (also known as TLINKs (Pustejovsky et al., 2003a)) extraction. While the first task has now been well handled by the state-of-the-art systems (HeidelTime (Strötgen and Gertz, 2010), SUTime (Chang and Manning, 2012), IllinoisTime (Zhao et al., 2012), NavyTime (Chambers, 2013), UWTime (Lee et al., 2014), etc.) with end-to-end F_1 scores being around 80%, the second task has long been a challenging one; even the top systems only achieved F_1 scores of around 35% in the TE workshops.

The goal of the temporal relation task is to generate a *directed temporal graph* whose nodes represent temporal entities (i.e., events or timexes) and edges represent the TLINKs between them. The task is challenging because it often requires global considerations – considering the entire graph, the TLINK annotation is quadratic in the number of nodes and thus very expensive, and an overwhelming fraction of the temporal relations are missing in human annotation. In this paper, we propose a structured learning approach to temporal relation extraction, where local models are updated based on feedback from global inferences. The structured approach also gives rise to a semi-supervised method, making it possible to take advantage of the readily available unlabeled data. As a byproduct, this approach further provides a new, effective perspective on handling those missing relations.

In the common formulations, temporal relations are categorized into three types: the E-E TLINKs (those between a pair of events), the T-T TLINKs (those between a pair of timexes), and the E-T TLINKs (those between an event and a timex). While the proposed approach can be generally applied to all three types, this paper focuses on the majority type, i.e., the E-E TLINKs. For example, consider the following snippet taken from the

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1027–1037
Copenhagen, Denmark, September 7–11, 2017.

training set provided in the TE3 workshop. We want to construct a temporal graph as in Fig. 1 for the events in boldface in Ex1.

Ex1 *. . . tons of earth* ***cascaded*** *down a hillside,* ***ripping*** *two houses from their foundations. No one was* ***hurt****, but firefighters* ***ordered*** *the evacuation of nearby homes and said they'll* ***monitor*** *the shifting ground. . . .*

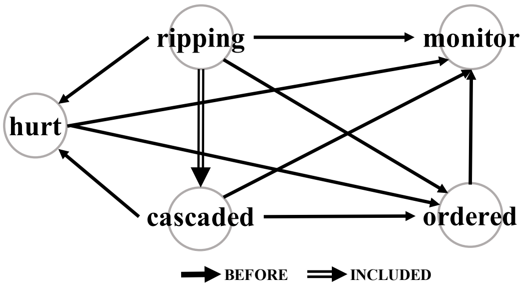


Figure 1: The desired event temporal graph for Ex1. Reverse TLINKs such as ***hurt*** is *after* ***ripping*** are omitted for simplicity.

As discussed in existing work (Verhagen, 2004; Bramsen et al., 2006; Mani et al., 2006; Chambers and Jurafsky, 2008), the structure of a temporal graph is constrained by some rather simple rules:

1. *Symmetry*. For example, if *A* is *before B*, then *B* must be *after A*.
2. *Transitivity*. For example, if *A* is *before B* and *B* is *before C*, then *A* must be *before C*.

This particular structure of a temporal graph (especially the transitivity structure) makes its nodes highly interrelated, as can be seen from Fig. 1. It is thus very challenging to identify the TLINKs between them, even for human annotators: The inter-annotator agreement on TLINKs is usually about 50%-60% (Mani et al., 2006). Fig. 2 shows the actual human annotations provided by TE3. Among all the ten possible pairs of nodes, only three TLINKs were annotated. Even if we only look at main events in consecutive sentences and at events in the same sentence, there are still quite a few missing TLINKs, e.g., the one between ***hurt*** and ***cascaded*** and the one between ***monitor*** and ***ordered***.

Early attempts by Mani et al. (2006); Chambers et al. (2007); Bethard et al. (2007); Verhagen and Pustejovsky (2008) studied *local* methods – learning models that make pairwise decisions between each pair of events. State-of-the-art local methods, including ClearTK (Bethard, 2013), UTTime (Laokulrat et al., 2013), and NavyTime (Chambers, 2013), use better designed rules or more features such as syntactic tree paths and achieve better results. However, the decisions made by these (local) models are often globally inconsistent (i.e., the symmetry and/or transitivity constraints are not satisfied for the entire temporal graph). Integer linear programming (ILP) methods (Roth and Yih, 2004) were used in this domain to enforce global consistency by several authors including Bramsen et al. (2006); Chambers and Jurafsky (2008); Do et al. (2012), which formulated TLINK extraction as an ILP and showed that it improves over local methods for densely connected graphs. Since these methods perform inference ("I") on top of pre-trained local classifiers ("L"), they are often referred to as *L+I* (Punyakanok et al., 2005). In a state-of-the-art method, CAEVO (Chambers et al., 2014), many hand-crafted rules and machine learned classifiers (called sieves therein) form a pipeline. The global consistency is enforced by inferring all possible relations before passing the graph to the next sieve. This best-first architecture is conceptually similar to L+I but the inference is greedy, similar to Mani et al. (2007); Verhagen and Pustejovsky (2008).

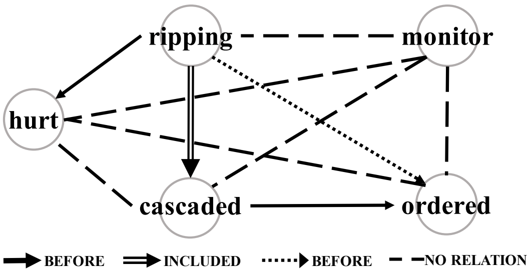


Figure 2: The human-annotation for Ex1 provided in TE3, where many TLINKs are missing due to the annotation difficulty. Solid lines: original human annotations. Dotted lines: TLINKs inferred from solid lines. Dashed lines: missing relations.

Although L+I methods impose global constraints in the inference phase, this paper argues that global considerations are necessary in the learning phase as well (i.e., structured learning). In parallel to the work presented here, Leeuwenberg and Moens (2017) also proposed a structured learning approach to extracting the temporal relations. Their work focuses on a domain-specific dataset from Clinical TempEval (Bethard et al., 2016), so their work does not need to address some of the difficulties of the general problem that our work addresses. More importantly, they compared structured learning to local baselines, while we find that the comparison between structured learning and L+I is more interesting and important for understanding the effect of global considerations in the learning phase. In difference from existing methods, we also discuss how to effectively use unlabeled data and how to handle the overwhelming fraction of missing relations in a principled way. Our solution targets on these issues and, as we show, achieves significant improvements on two commonly used evaluation sets.

The rest of this paper is organized as follows. Section 2 clarifies the temporal relation types and the evaluation metric of a temporal graph used in this paper, Section 3 explains the structured learning approach in detail, and Section 4 discusses the practical issue of missing relations. We provide experiments and discussion in Section 5 and conclusion in Section 6.

2 Background

2.1 Temporal Relation Types

Existing corpora for temporal processing often follows the interval representation of events proposed in Allen (1984), and makes use of 13 relation types in total. In many systems, *vague* or *none* is also included as another relation type when a TLINK is not clear or missing. However, current systems usually use a reduced set of relation types, mainly due to the following reasons.

1. The non-uniform distribution of all the relation types makes it difficult to separate low-frequency ones from the others (see Table 1 in Mani et al. (2006)). For example, relations such as *immediately_before* or *immediately_after* barely exist in a corpus compared to *before* and *after*.

2. Due to the ambiguity in natural language, determining relations like *before* and *immediately_before* can be a difficult task itself (Chambers et al., 2014).

In this work, we follow the reduced set of temporal relation types used in CAEVO (Chambers et al., 2014): *before*, *after*, *includes*, *is_included*, *equal*, and *vague*.

2.2 Quality of A Temporal Graph

The most recent evaluation metric in TE3, i.e., the temporal awareness (UzZaman and Allen, 2011), is adopted in this work. Specifically, let G_{sys} and G_{true} be two temporal graphs from the system prediction and the ground truth, respectively. The precision and recall of temporal awareness are defined as follows.

$$P = \frac{|G_{sys}^- \cap G_{true}^+|}{|G_{sys}^-|},\ R = \frac{|G_{true}^- \cap G_{sys}^+|}{|G_{true}^-|}$$

where G^+ is the closure of graph G, G^- is the reduction of G, "$\cap$" is the intersection between TLINKs in two graphs, and $|G|$ is the number of TLINKs in G. The temporal awareness metric better captures how "useful" a temporal graph is. For example, if system 1 produces ***ripping*** is *before* ***hurt*** and ***hurt*** is *before* ***monitor***, and system 2 adds ***ripping*** is *before* ***monitor*** on top of system 1. Since system 2 is simply a transitive closure of system 1, they would have the same evaluation scores. Note that *vague* relations are usually considered as non-existing TLINKs and are not counted during evaluation.

3 A Structured Training Approach

As shown in Fig. 1, the learning problem in temporal relation extraction is global in nature. Even the top local method in TE3, UTTime (Laokulrat et al., 2013), only achieved F_1=56.5 when presented with a pair of temporal entities (Task C–relation only (UzZaman et al., 2013)). Since the success of an L+I method strongly relies on the quality of the local classifiers, a poor local classifier is obviously a roadblock for L+I methods. Following the insights from Punyakanok et al. (2005), we propose to use a structured learning approach (also called "Inference Based Training" (IBT)).

Unlike the current L+I approach, where local classifiers are trained independently beforehand without knowledge of the predictions on neighboring pairs, we train local classifiers with feedback that accounts for other relations, by performing global inference in each round of the learning process. In order to introduce the structured learning algorithm, we first explain its most important component, the global inference step.

3.1 Inference

In a document with n pairs of events, let $\phi_i \in \mathcal{X} \subseteq \mathbb{R}^d$ be the extracted d-dimensional feature and $y_i \in \mathcal{Y}$ be the temporal relation for the i-th pair of events, $i = 1, 2, \ldots, n$, where $\mathcal{Y} = \{r_j\}_{j=1}^6$ is the label set for the six temporal relations we use. Moreover, let $\mathbf{x} = \{\phi_1, \ldots, \phi_n\} \in \mathcal{X}^n$ and $\mathbf{y} = \{y_1, \ldots, y_n\} \in \mathcal{Y}^n$ be more compact representations of all the features and labels in this document. Given the weight vector $\mathbf{w}_r$ of a linear classifier trained for relation $r \in \mathcal{Y}$ (i.e., using the one-vs-all scheme), the global inference step is to solve the following constrained optimization problem:

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}\in\mathcal{C}(\mathcal{Y}^n)} f(\mathbf{x}, \mathbf{y}), \qquad (1)$$

where $\mathcal{C}(\mathcal{Y}^n) \subseteq \mathcal{Y}^n$ constrains the temporal graph to be symmetrically and transitively consistent, and $f(\mathbf{x}, \mathbf{y})$ is the scoring function:

$$f(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} f_{y_i}(\phi_i) = \sum_{i=1}^{n} \frac{e^{\mathbf{w}_{y_i}^T \phi_i}}{\sum_{r\in\mathcal{Y}} e^{\mathbf{w}_r^T \phi_i}}.$$

Specifically, $f_{y_i}(\phi_i)$ is the probability of the i-th event pair having relation y_i. $f(x, y)$ is simply the sum of these probabilities over all the event pairs in a document, which we think of as the confidence of assigning $\mathbf{y} = \{y_1, ..., y_n\}$ to this document and therefore, it needs to be maximized in Eq. (1).

Note that when $\mathcal{C}(\mathcal{Y}^n) = \mathcal{Y}^n$, Eq. (1) can be solved for each $\hat{y}_i$ independently, which is what the so-called local methods do, but the resulting $\hat{\mathbf{y}}$ may not satisfy global consistency in this way. When $\mathcal{C}(\mathcal{Y}^n) \neq \mathcal{Y}^n$, Eq. (1) cannot be decoupled for each $\hat{y}_i$ and is usually formulated as an ILP problem (Roth and Yih, 2004; Chambers and Jurafsky, 2008; Do et al., 2012). Specifically, let $\mathcal{I}_r(ij) \in \{0, 1\}$ be the indicator function of relation r for event i and event j and $f_r(ij) \in [0, 1]$ be the corresponding soft-max score. Then the ILP objective for global inference is formulated as follows.

$$\hat{\mathcal{I}} = \underset{\mathcal{I}}{\text{argmax}} \textstyle\sum_{ij\in\mathcal{E}} \sum_{r\in\mathcal{Y}} f_r(ij)\mathcal{I}_r(ij) \qquad (2)$$

$$\text{s.t.} \quad \underset{\text{(uniqueness)}}{\Sigma_r \mathcal{I}_r(ij) = 1}, \; \underset{\text{(symmetry)}}{\mathcal{I}_r(ij) = \mathcal{I}_{\bar{r}}(ji)},$$

$$\underset{\text{(transitivity)}}{\mathcal{I}_{r_1}(ij) + \mathcal{I}_{r_2}(jk) - \Sigma_{m=1}^{N} \mathcal{I}_{r_3^m}(ik) \leq 1},$$

for all distinct events i, j, and k, where $\mathcal{E} = \{ij \mid \text{sentence dist}(i, j) \leq 1\}$, $\bar{r}$ is the reverse of r, and N is the number of possible relations for r_3 when r_1 and r_2 are true.

Our formulation in Eq. (2) is different from previous work (Chambers and Jurafsky, 2008; Do et al., 2012) in two aspects: 1) We restrict our event pairs ij to a smaller set $\mathcal{E} = \{ij \mid \text{sentence dist}(i, j) \leq 1\}$ where pairs that are more than one sentence away are deleted for computational efficiency and (usually) for better performance. In fact, to make better use of global constraints, we should have allowed more event pairs in Eq. (2). However, $f_r(ij)$ is usually more reliable when i and j are closer in text. Many participating systems in TE3 (UzZaman et al., 2013) have used this *pre-filtering* strategy to balance the trade-off between confidence in $f_r(ij)$ and global constraints. We observe that the strategy fits very well to the existing datasets: As shown in Fig. 3, annotated TLINKs barely exist if two events are two sentences away. 2) Previously, transitivity constraints were formulated as $\mathcal{I}_{r_1}(ij) + \mathcal{I}_{r_2}(jk) - \mathcal{I}_{r_3}(ik) \leq 1$, which is a special case when $N = 1$ and can be understood as "r_1 and r_2 determine a single r_3". However, it was overlooked that, although some r_1 and r_2 cannot uniquely determine r_3, they can still constrain the set of labels r_3 can take. For example, as shown in Fig. 4, when r_1=*before* and r_2=*is_included*, r_3 is not determined but we know that $r_3 \in \{$*before*, *is_included*$\}$[1]. This information can be easily exploited by allowing $N > 1$.

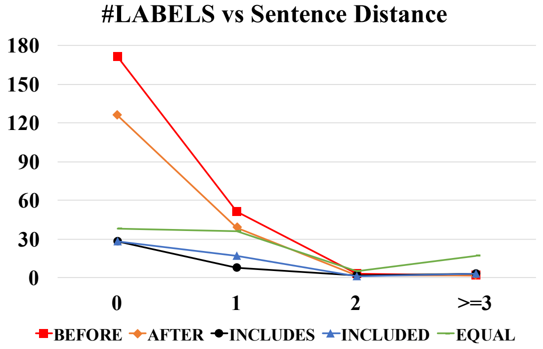


Figure 3: #TLINKs vs sentence distance on the TE3 Platinum dataset. The tail of *equal* is due to event coreference and beyond our focus.

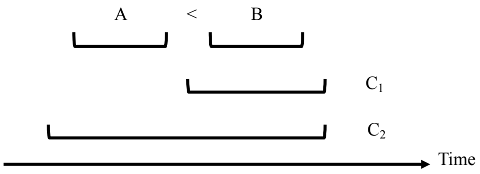


Figure 4: When A *is before* B and B *is_included in* C, A can either be *before* C_1 or *is_included in* C_2. We propose to incorporate this via the transitivity constraints for Eq. (2).

With these two differences, the optimization problem (2) can still be efficiently solved using off-the-shelf ILP packages such as GUROBI

[1] The transitivity table in Allen (1983) shows two more possible relations, *overlap* and *immediately_before*, which are not in our label set.

(Gurobi Optimization, Inc., 2012).

3.2 Learning

With the inference solver defined above, we propose to use the structured perceptron (Collins, 2002) as a representative for the inference based training (IBT) algorithm to learn those weight vectors $\mathbf{w}_r$. Specifically, let $\mathcal{L} = \{\mathbf{x}_k, \mathbf{y}_k\}_{k=1}^K$ be the labeled training set of K instances (usually documents). The structured perceptron training algorithm for this problem is shown in Algorithm 1. The Illinois-SL package (Chang et al., 2010) was used in our experiments for its structured perceptron component. In terms of the features used in this work, we adopt the same set of features designed for E-E TLINKs in Sec. 3.1 of Do et al. (2012).

In Algorithm 1, Line 6 is the inference step as in Eq. (1) or (2), which is augmented with a closure operation on $\hat{\mathbf{y}}$ in the following line. In the case in which there is only one pair of events in each instance (thus no structure to take advantage of), Algorithm 1 reduces to the conventional perceptron algorithm and Line 6 simply chooses the top scoring label. With a structured instance instead, Line 6 becomes slower to solve, but it can provide valuable information so that the perceptron learner is able to look further at other labels rather than an isolated pair. For example in Ex1 and Fig. 1, the fact that (ripping,ordered)=*before* is established through two other relations: 1) ***ripping*** is an adverbial participle and thus *included* in ***cascaded*** and 2) ***cascaded*** is *before* ***ordered***. If (ripping,ordered)=*before* is presented to a local learning algorithm without knowing its predictions on (ripping,cascaded) and (cascaded,ordered), then the model either cannot support it or overfits it. In IBT, however, if the classifier was correct in deciding (ripping,cascaded) and (cascaded,ordered), then (ripping,ordered) would be correct automatically and would not contribute to updating the classifier.

Algorithm 1: Structured perceptron algorithm for temporal relations

```
   Input: Training set $\mathcal{L} = \{\mathbf{x}_k, \mathbf{y}_k\}_{k=1}^K$,
          learning rate $\lambda$
1  Perform graph closure on each $\mathbf{y}_k$
2  Initialize $\mathbf{w}_r = \mathbf{0}$, $\forall r \in \mathcal{Y}$
3  while convergence criteria not satisfied do
4      Shuffle the examples in $\mathcal{L}$
5      foreach $(\mathbf{x}, \mathbf{y}) \in \mathcal{L}$ do
6          $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{C}} f(\mathbf{x}, \mathbf{y})$
7          Perform graph closure on $\hat{\mathbf{y}}$
8          if $\hat{\mathbf{y}} \neq \mathbf{y}$ then
9              $\mathbf{w}_r = \mathbf{w}_r + \lambda(\sum_{i:\mathbf{y}_i=r} \phi_i -$
               $\sum_{i:\hat{\mathbf{y}}_i=r} \phi_i), \forall r \in \mathcal{Y}$
10 return $\{\mathbf{w}_r\}_{r\in\mathcal{Y}}$
```

3.3 Semi-supervised Structured Learning

The scarcity of training data and the difficulty in annotation have long been a bottleneck for temporal processing systems. Given the inherent global constraints in temporal graphs, we propose to perform semi-supervised structured learning using the constraint-driven learning (CoDL) algorithm (Chang et al., 2007, 2012), as shown in Algorithm 2, where the function "Learn" in Lines 2 and 9 represents any standard learning algorithm (e.g., perceptron, SVM, or even structured perceptron; here we used the averaged perceptron (Freund and Schapire, 1998)) and subscript "r" means selecting the learned weight vector for relation $r \in \mathcal{Y}$. CoDL improves the model learned from a small amount of labeled data by repeatedly generating feedback through labeling unlabeled examples, which is in fact a semi-supervised version of IBT. Experiments show that this scheme is indeed helpful in this problem.

Algorithm 2: Constraint-driven learning algorithm

```
   Input: Labeled set $\mathcal{L}$, unlabeled set $\mathcal{U}$,
          weighting coefficient $\gamma$
1  Perform closure on each graph in $\mathcal{L}$
2  Initialize $\mathbf{w}_r = \text{Learn}(\mathcal{L})_r, \forall\, r \in \mathcal{Y}$
3  while convergence criteria not satisfied do
4      $\mathcal{T} = \emptyset$
5      foreach $\mathbf{x} \in \mathcal{U}$ do
6          $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{C}} f(\mathbf{x}, \mathbf{y})$
7          Perform graph closure on $\hat{\mathbf{y}}$
8          $\mathcal{T} = \mathcal{T} \cup \{(\mathbf{x}, \hat{\mathbf{y}})\}$
9      $\mathbf{w}_r = \gamma\mathbf{w}_r + (1-\gamma)\text{Learn}(\mathcal{T})_r, \forall\, r \in \mathcal{Y}$
10 return $\{\mathbf{w}_r\}_{r\in\mathcal{Y}}$
```

4 Missing Annotations

Since even human annotators find it difficult to annotate temporal graphs, many of the TLINKs are left unspecified by annotators (compare Fig. 2 to Fig. 1). While some of these missing TLINKs can be inferred from existing ones, the vast majority still remain unknown as shown in Table 1. Despite the existence of denser annotation schemes (e.g., Cassidy et al. (2014)), the TLINK annotation task is quadratic in the number of nodes, and it is practically infeasible to annotate complete graphs. Therefore, the problem of identifying these unknown relations in training and test is a major issue that dramatically hurts existing methods.

Table 1: Categories of E-E TLINKs in the TE3 Platinum dataset. Among all pairs of events, 98.2% of them are left unspecified by the annotators. Graph closure can automatically add 8.7%, but most of the event pairs are still unknown.

Type		#TLINK	%
Annotated		582	1.8
Missing	Inferred	2840	8.7
	Unknown	29240	89.5
Total		32662	100

We could simply use these unknown pairs (or some filtered version of them) to design rules or train classifiers to identify whether a TLINK is *vague* or not. However, we propose to exclude both the unknown pairs and the *vague* classifier from the training process – by changing the structured loss function to ignore the inference feedback on *vague* TLINKs (see Line 9 in Algorithm 1 and Line 9 in Algorithm 2). The reasons are discussed below.

First, it is believed that a lot of the unknown pairs are not really *vague* but rather pairs that the annotators failed to look at (Bethard et al., 2007; Cassidy et al., 2014; Chambers et al., 2014). For example, (cascaded, monitor) should be annotated as *before* but is missing in Fig. 2. It is hard to exclude this noise in the data during training. Second, compared to the overwhelmingly large number of unknown TLINKs (89.5% as shown in Table 1), the scarcity of non-vague TLINKs makes it hard to learn a good *vague* classifier. Third, *vague* is fundamentally different from the other relation types. For example, if a *before* TLINK can be established given a sentence, then it always holds as *before* regardless of other events around it, but if a TLINK is *vague* given a sentence, it may still change to other types afterwards if a connection can later be established through other nodes from the context. This distinction emphasizes that *vague* is a consequence of lack of background/contextual information, rather than a concrete relation type to be trained on. Fourth, without the *vague* classifier, the predicted temporal graph tends to become more densely connected, thus the global transitivity constraints can be more effective in correcting local mistakes (Chambers and Jurafsky, 2008).

However, excluding the local classifier for *vague* TLINKs would undesirably assign non-vague TLINKs to every pair of events. To handle this, we take a closer look at the *vague* TLINKs. We note that a *vague* TLINK could arise in two situations if the annotators did not fail to look at it. One is that an annotator looks at this pair of events and decides that multiple relations can exist, and the other one is that two annotators disagree on the relation (similar arguments were also made in Cassidy et al. (2014)). In both situations, the annotators first try to assign all possible relations to a TLINK, and then change the relation to *vague* if more than one can be assigned. This human annotation process for *vague* is different from many existing methods, which either identify the existence of a TLINK first (using rules or machine-learned classifiers) and then classify, or directly include *vague* as a classification label along with other non-vague relations.

In this work, however, we propose to mimic this mental process by a *post-filtering* method[2]. Specifically, we take each TLINK produced by ILP and determine whether it is *vague* using its relative entropy (the Kullback-Leibler divergence) to the uniform distribution. Let $\{r_m\}_{m=1}^M$ be the set of relations that the i-th pair of events can take, we filter the i-th TLINK given by ILP by:

$$\delta_i = \sum_{m=1}^{M} f_{r_m}(\phi_i) \log\left(M f_{r_m}(\phi_i)\right),$$

where $f_{r_m}(\phi_i)$ is the soft-max score of r_m, obtained by the local classifier for r_m. We then compare δ_i to a fixed threshold τ to determine the vagueness of this TLINK; we accept its originally predicted label if $\delta_i > \tau$, or change it to *vague* otherwise. Using relative entropy here is intuitively appealing and empirically useful as shown in the experiments section; better metrics are of course yet to be designed.

5 Experiments

5.1 Datasets

The TempEval3 (TE3) workshop (UzZaman et al., 2013) provided the TimeBank (TB) (Pustejovsky et al., 2003b), AQUAINT (AQ) (Graff, 2002), Silver (TE3-SV), and Platinum (TE3-PT) datasets,

[2] Some systems (e.g., TARSQI (Verhagen and Pustejovsky, 2008)) employed a similar idea from a different standpoint, by thresholding TLINKs based on confidence scores.

where TB and AQ are usually for training, and TE3-PT is usually for testing. The TE3-SV dataset is a much larger, machine-annotated and automatically-merged dataset based on multiple systems, with the intention to see if these “silver” standard data can help when included in training (although almost all participating systems saw performance drop with TE3-SV included in training).

Two popular augmentations on TB are the Verb-Clause temporal relation dataset (VC) and TimebankDense dataset (TD). The VC dataset has specially annotated event pairs that follow the so-called Verb-Clause structure (Bethard et al., 2007), which is usually beneficial to be included in training (UzZaman et al., 2013). The TD dataset contains 36 documents from TB which were re-annotated using the dense event ordering framework proposed in Cassidy et al. (2014). The experiments included in this paper will involve the TE3 datasets as well as these augmentations. Therefore, some statistics on them are shown in Table 2 for the readers’ information.

Table 2: Facts about the datasets used in this paper. The TD dataset is split into train, dev, and test in the same way as in Chambers et al. (2014). Note that the column of TLINKs only counts the non-vague TLINKs, from which we can see that the TD dataset has a much higher ratio of #TLINKs to #Events. The TLINK annotations in TE3-SV is not used in this paper and its number is thus not shown.

Dataset	Doc	Event	TLINK	Note
TB+AQ	256	12K	12K	Training
VC	132	1.6K	0.9K	Training
TD	36	1.6K	5.7K	Training
TD-Train	22	1K	3.8K	Training
TD-Dev	5	0.2K	0.6K	Dev
TD-Test	9	0.4K	1.3K	Eval
TE3-PT	20	0.7K	0.9K	Eval
TE3-SV	2.5K	81K	-	Unlabeled

5.2 Baseline Methods

In addition to the state-of-the-art systems, another two baseline methods were also implemented for a better understanding of the proposed ones. The first is the regularized averaged perceptron (AP) (Freund and Schapire, 1998) implemented in the LBJava package (Rizzolo and Roth, 2010) and is a local method. On top of the first baseline, we performed global inference in Eq.(2), referred to as the L+I baseline (AP+ILP). Both of them used the same feature set (i.e., as designed in Do et al. (2012)) as in the proposed structured perceptron (SP) and CoDL for fair comparisons. To clarify, SP and CoDL are training algorithms and their immediate outputs are the weight vectors $\{\mathbf{w}_r\}_{r\in\mathcal{Y}}$ for local classifiers. An ILP inference was performed on top of them to yield the final output, and we refer to it as “S+I” (i.e., structured learning+inference) methods.

Table 3: Temporal awareness scores on TE3-PT given gold event pairs. Systems that are significantly better (per McNemar’s test with $p < 0.0005$) than the previous rows are underlined. The last column shows the relative improvement in F1 score over AP-1, which identifies the source of improvement: 5.2% from additional training data, 9.3% (14.5%-5.2%) from constraints, and 10.4% from structured learning.

Method	P	R	F1	%
UTTime	55.6	57.4	56.5	+5.0
AP-1	56.3	51.5	53.8	0
AP-2	58.0	55.3	56.6	+5.2
AP+ILP	62.2	61.1	61.6	+14.5
SP+ILP	**69.1**	**65.5**	**67.2**	**+24.9**

5.3 Results and Discussion

5.3.1 TE3 Task C - Relation Only

To show the benefit of using structured learning, we first tested one scenario where the gold pairs of events that have a non-vague TLINK were known priori. This setup was a standard task presented in TE3, so that the difficulty of detecting *vague* TLINKs was ruled out. This setup also helps circumvent the issue that TE3 penalizes systems which assign extra labels that do not exist in the annotated graph, while these extra labels may be actually correct because the annotation itself might be incomplete. UTTime (Laokulrat et al., 2013) was the top system in this task in TE3. Since UTTime is not available to us, and its performance was reported in TE3 in terms of both E-E and E-T TLINKs together, we locally trained an E-T classifier based on Do et al. (2012) and included its prediction only for fair comparison.

UTTime is a local method and was trained on TB+AQ and tested on TE3-PT. We used the same datasets for our local baseline and its performance is shown in Table 3 under the name “AP-1”. Note that the reported numbers below are the temporal awareness scores obtained from the official evaluation script provided in TE3. We can see that UTTime is about 3% better than AP-1 in the absolute value of F_1, which is expected since UTTime included more advanced features derived from syntactic parse trees. By adding the VC and TD datasets into the training set, we retrained our local baseline and achieved comparable performance to

Table 4: Temporal awareness scores given gold events but with no gold pairs, which show that the proposed S+I methods outperformed state-of-the-art systems in various settings. The fourth column indicates the annotation sources used, with additional unlabeled dataset in the parentheses. The "Filters" column shows if the pre-filtering method (Sec. 3.1) or the proposed post-filtering method (Sec. 4) were used. The last column is the relative improvement in F_1 score compared to baseline systems on line 1, 7, and 11, respectively. Systems that are significantly better than the "*"-ed systems are underlined (per McNemar's test with $p < 0.0005$).

No.	System	Method	Anno. (Unlabeled)	Testset	Filters	P	R	F1	%
1	ClearTK	Local	TB, AQ, VC, TD	TE3-PT	pre	**37.2**	33.1	35.1	0
2	AP*	Local	TB, AQ, VC, TD	TE3-PT	pre	35.3	37.1	36.1	+2.8
3	AP+ILP	L+I	TB, AQ, VC, TD	TE3-PT	pre	35.7	35.0	35.3	+0.6
4	SP+ILP	S+I	TB, AQ, VC, TD	TE3-PT	pre	32.4	45.2	37.7	+7.4
5	SP+ILP	S+I	TB, AQ, VC, TD	TE3-PT	pre+post	33.1	**49.2**	39.6	+12.8
6	CoDL+ILP	S+I	TB, AQ, VC, TD (TE3-SV)	TE3-PT	pre+post	35.5	46.5	**40.3**	**+14.8**
7	ClearTK*	Local	TB, VC	TE3-PT	pre	**35.9**	38.2	37.0	0
8	SP+ILP	S+I	TB, VC	TE3-PT	pre+post	30.7	**47.1**	37.2	+0.5
9	CoDL+ILP	S+I	TB, VC (TE3-SV)	TE3-PT	pre+post	33.9	45.9	**39.0**	**+5.4**
10	ClearTK	Local	TD-Train	TD-Test	pre	46.04	20.90	28.74	-
11	CAEVO*	L+I	TD-Train	TD-Test	pre	**54.17**	39.49	45.68	0
12	SP+ILP	S+I	TD-Train	TD-Test	pre+post	45.34	48.68	46.95	+3.0
13	CoDL+ILP	S+I	TD-Train (TE3-SV)	TD-Test	pre+post	45.57	**51.89**	**48.53**	**+6.3**

UTTime ("AP-2" in Table 3). On top of AP-2, a global inference step enforcing symmetry and transitivity constraints ("AP+ILP") can further improve the F_1 score by 9.3%, which is consistent with previous observations (Chambers and Jurafsky, 2008; Do et al., 2012). SP+ILP further improved the performance in precision, recall, and F_1 significantly (per the McNemar's test (Everitt, 1992; Dietterich, 1998) with p <0.0005), reaching an F_1 score of 67.2%. This meets our expectation that structured learning can be better when the local problem is difficult (Punyakanok et al., 2005).

5.3.2 TE3 Task C

In the first scenario, we knew in advance which TLINKs existed or not, so the "pre-filtering" (i.e., ignoring distant pairs as mentioned in Sec. 3.1 and "post-filtering" methods were not used when generating the results in Table 3. We then tested a more practical scenario, where we only knew the events, but did not know which ones are related. This setup was Task C in TE3 and the top system was ClearTK (Bethard, 2013). Again, for fair comparison, we simply added the E-T TLINKs predicted by ClearTK. Moreover, 10% of the training data was held out for development. Corresponding results on the TE3-PT testset are shown in Table 4.

From lines 2-4, all systems see significant drops in performance if compared with the same entries in Table 3. It confirms our assertion that how to handle *vague* TLINKs is a major issue for this temporal relation extraction problem. The improvement of SP+ILP (line 4) over AP (line 2) was small and AP+ILP (line 3) was even worse than AP, which necessitates the use of a better approach towards *vague* TLINKs. By applying the post-filtering method proposed in Sec. 4, we were able to achieve better performances using SP+ILP (line 5), which shows the effectiveness of this strategy. Finally, by setting $\mathcal{U}$ in Algorithm 2 to be the TE3-SV dataset, CoDL+ILP (line 6) achieved the best F_1 score with a relative improvement over ClearTK being 14.8%. Note that when using TE3-SV in this paper, we did not use its annotations on TLINKs because of its well-known large noise (UzZaman et al., 2013).

In UzZaman et al. (2013), we notice that the best performance of ClearTK was achieved when trained on TB+VC (line 7 is higher than its reported values in TE3 because of later changes in ClearTK), so we retrained the proposed systems on the same training set and results are shown on lines 8-9. In this case, the improvement of S+I over Local was small, which may be due to the lack of training data. Note that line 8 was still significantly different to line 7 per the McNemar's test, although there was only 0.2% absolute difference in F_1, which can be explained from their large differences in precision and recall.

5.3.3 Comparison with CAEVO

The proposed structured learning approach was further compared to a recent system, a CAscading EVent Ordering architecture (CAEVO) proposed in Chambers et al. (2014) (lines 10-13). We used the same training set and test set as CAEVO in the S+I systems. Again, we added the E-T TLINKs predicted by CAEVO to both S+I systems. In Chambers et al. (2014), CAEVO was reported on the straightforward evaluation metric including the *vague* TLINKs, but the temporal awareness scores were used here, which explains the difference between line 11 in Table 4 and what was reported in Chambers et al. (2014).

ClearTK was reported to be outperformed by CAEVO on TD-Test (Chambers et al., 2014), but we observe that ClearTK on line 10 was much worse even than itself on line 7 (trained on TB+VC) and on line 1 (trained on TB+AQ+VC+TD) due to the annotation scheme difference between TD and TB/AQ/VC. ClearTK was designed mainly for TE3, aiming for high precision, which is reflected by its high precision on line 10, but it does not have enough flexibility to cope with two very different annotation schemes. Therefore, we have chosen CAEVO as the baseline system to evaluate the significance of the proposed ones. On the TD-Test dataset, all systems other than ClearTK had better F_1 scores compared to their performances on TE3-PT. This notable difference (i.e., 48.53 vs 40.3) indicates the better quality of the dense annotation scheme that was used to create TD (Cassidy et al., 2014). SP+ILP outperformed CAEVO and if additional unlabeled dataset TE3-SV was used, CoDL+ILP achieved the best score with a relative improvement in F_1 score being 6.3%.

We notice that the proposed systems often have higher recall than precision, and that this is less an issue on a densely annotated testset (TD-Test), so their low precision on TE3-PT possibly came from the missing annotations on TE3-PT. It is still under investigation how to control precision and recall in real applications.

6 Conclusion

We develop a structured learning approach to identifying temporal relations in natural language text and show that it captures the global nature of this problem better than state-of-the-art systems do. A new perspective towards *vague* relations is also proved to gain from fully taking advantage of the structured approach. In addition, the global nature of this problem gives rise to a better way of making use of the readily available unlabeled data, which further improves the proposed method. The improved performance on both TE3-PT and TD-Test, two differently annotated datasets, clearly shows the advantage of the proposed method over existing methods. We plan to build on the notable improvements shown here and expand this study to deal with additional temporal reasoning problems in natural language text.

Acknowledgements

We thank all the reviewers for providing useful comments. This research is supported in part by a grant from the Allen Institute for Artificial Intelligence (allenai.org); the IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM Cognitive Horizon Network; by the US Defense Advanced Research Projects Agency (DARPA) under contract FA8750-13-2-0008; and by the Army Research Laboratory (ARL) under agreement W911NF-09-2-0053. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies of the U.S. Government.

References

James F Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11):832–843.

James F Allen. 1984. Towards a general theory of action and time. *Artificial intelligence* 23(2):123–154.

Steven Bethard. 2013. ClearTK-TimeML: A minimalist approach to TempEval 2013. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*. volume 2, pages 10–14.

Steven Bethard and James H Martin. 2007. CU-TMP: Temporal relation classification using syntactic and semantic features. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. Association for Computational Linguistics, pages 129–132.

Steven Bethard, James H Martin, and Sara Klingenstein. 2007. Timelines from text: Identification of syntactic temporal relations. In *Semantic Computing, 2007. ICSC 2007. International Conference on*. IEEE, pages 11–18.

Steven Bethard, Guergana Savova, Wei-Te Chen, Leon Derczynski, James Pustejovsky, and Marc Verhagen. 2016. SemEval-2016 Task 12: Clinical TempEval. *Proceedings of SemEval* pages 1052–1062.

P. Bramsen, P. Deshpande, Y. K. Lee, and R. Barzilay. 2006. Inducing temporal graphs. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Sydney, Australia, pages 189–198.

Taylor Cassidy, Bill McDowell, Nathanel Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. Technical report, DTIC Document.

N. Chambers and D. Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Nathanael Chambers. 2013. NavyTime: Event and time ordering from raw text. Technical report, DTIC Document.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics* 2:273–284.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, pages 173–176.

Angel X Chang and Christopher D Manning. 2012. SUTIME: A library for recognizing and normalizing time expressions. In *LREC*. volume 2012, pages 3735–3740.

M. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics, Prague, Czech Republic, pages 280–287.

M. Chang, L. Ratinov, and D. Roth. 2012. Structured learning with constrained conditional models. *Machine Learning* 88(3):399–431.

M. Chang, V. Srikumar, D. Goldwasser, and D. Roth. 2010. Structured output learning with indirect supervision. In *Proc. of the International Conference on Machine Learning (ICML)*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. of ACL*.

Thomas G Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation* 10(7):1895–1923.

Q. Do, W. Lu, and D. Roth. 2012. Joint inference for event timeline construction. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Brian S Everitt. 1992. *The analysis of contingency tables*. CRC Press.

Y. Freund and R. Schapire. 1998. Large margin classification using the Perceptron algorithm. In *Proceedings of the Annual ACM Workshop on Computational Learning Theory (COLT)*. pages 209–217.

David Graff. 2002. The AQUAINT corpus of english news text. *Linguistic Data Consortium, Philadelphia* .

Gurobi Optimization, Inc. 2012. Gurobi optimizer reference manual.

Heng Ji, Taylor Cassidy, Qi Li, and Suzanne Tamang. 2014. Tackling representation, annotation and classification challenges for temporal knowledge base population. *Knowledge and Information Systems* 41(3):611–646.

P. Jindal and D. Roth. 2013. Extraction of events and temporal expressions from clinical narratives. *Journal of Biomedical Informatics (JBI)* .

Natsuda Laokulrat, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. UTTime: Temporal relation classification using deep syntactic features. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*. volume 2, pages 88–92.

Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent semantic parsing for time expressions. In *ACL (1)*. pages 1437–1447.

Tuur Leeuwenberg and Marie-Francine Moens. 2017. Structured learning for temporal relation extraction from clinical records. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Hector Llorens, Nathanael Chambers, Naushad UzZaman, Nasrin Mostafazadeh, James Allen, and James Pustejovsky. 2015. SemEval-2015 Task 5: QA TEMPEVAL - evaluating temporal information understanding with question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. pages 792–800.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 753–760.

Inderjeet Mani, Ben Wellner, Marc Verhagen, and James Pustejovsky. 2007. Three approaches to learning TLINKs in TimeML. *Technical Report CS-07–268, Computer Science Department* .

Anne-Lyse Minard, Manuela Speranza, Eneko Agirre, Itziar Aldabe, Marieke van Erp, Bernardo Magnini, German Rigau, Ruben Urizar, and Fondazione Bruno Kessler. 2015. SemEval-2015 Task 4: TimeLine: Cross-document event ordering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. pages 778–786.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2005. Learning and inference over constrained output. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*. pages 1124–1129.

James Pustejovsky, José M Castano, Robert Ingria, Roser Sauri, Robert J Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R Radev. 2003a. TimeML: Robust specification of event and temporal expressions in text. *New directions in question answering* 3:28–34.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003b. The TIMEBANK corpus. In *Corpus linguistics*. volume 2003, page 40.

N. Rizzolo and D. Roth. 2010. Learning based java for rapid development of nlp systems. In *Proc. of the International Conference on Language Resources and Evaluation (LREC)*. Valletta, Malta.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*. Association for Computational Linguistics, pages 1–8.

Jannik Strötgen and Michael Gertz. 2010. HeidelTime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 321–324.

Mihai Surdeanu. 2013. Overview of the TAC2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *TAC*.

Naushad UzZaman and James F Allen. 2011. Temporal evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, pages 351–356.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TEMPEVAL-3: Evaluating time expressions, events, and temporal relations. *Second Joint Conference on Lexical and Computational Semantics* 2:1–9.

Marc Verhagen. 2004. *Times Between The Lines*. Ph.D. thesis, Brandeis University.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. SemEval-2007 Task 15: TempEval temporal relation identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*. Association for Computational Linguistics, pages 75–80.

Marc Verhagen and James Pustejovsky. 2008. Temporal processing with the TARSQI toolkit. In *22nd International Conference on on Computational Linguistics: Demonstration Papers*. Association for Computational Linguistics, pages 189–192.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th international workshop on semantic evaluation*. Association for Computational Linguistics, pages 57–62.

R. Zhao, Q. Do, and D. Roth. 2012. A robust shallow temporal reasoning system. In *Proc. of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL) Demo*.

Importance sampling for unbiased on-demand evaluation of knowledge base population

Arun Tejasvi Chaganty[*] and **Ashwin Pradeep Paranjape**[*] and **Percy Liang**
Computer Science Department
Stanford University
{chaganty,ashiwnp,pliang}@cs.stanford.edu

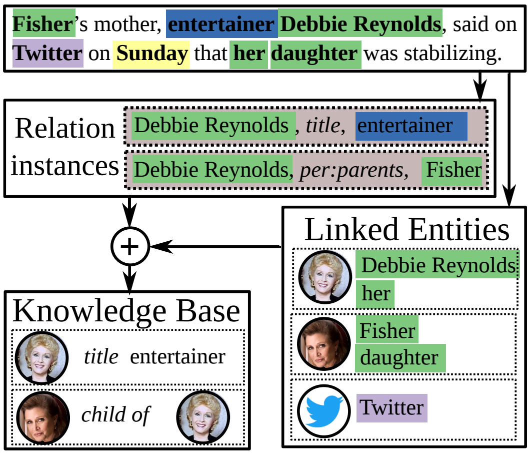
Christopher D. Manning
Computer Science Department
Stanford University
{manning}@cs.stanford.edu

Abstract

Knowledge base population (KBP) systems take in a large document corpus and extract entities and their relations. Thus far, KBP evaluation has relied on judgements on the pooled predictions of existing systems. We show that this evaluation is problematic: when a new system predicts a previously unseen relation, it is penalized even if it is correct. This leads to significant bias against new systems, which counterproductively discourages innovation in the field. Our first contribution is a new importance-sampling based evaluation which corrects for this bias by annotating a new system's predictions on-demand via crowdsourcing. We show this eliminates bias and reduces variance using data from the 2015 TAC KBP task. Our second contribution is an implementation of our method made publicly available as an online KBP evaluation service. We pilot the service by testing diverse state-of-the-art systems on the TAC KBP 2016 corpus and obtain accurate scores in a cost effective manner.

1 Introduction

Harnessing the wealth of information present in unstructured text online has been a long standing goal for the natural language processing community. In particular, knowledge base population seeks to automatically construct a knowledge base consisting of relations between entities from a document corpus. Knowledge bases have found many applications including question answering (Berant et al., 2013; Fader et al., 2014; Reddy et al., 2014), automated reasoning (Kalyanpur et al., 2012) and dialogue (Han et al., 2015).



Fisher's mother, **entertainer** **Debbie Reynolds**, said on **Twitter** on **Sunday** that **her** **daughter** was stabilizing.

Figure 1: An example describing entities and relations in knowledge base population.

Evaluating these systems remains a challenge as it is not economically feasible to exhaustively annotate every possible candidate relation from a sufficiently large corpus. As a result, a pooling-based methodology is used in practice to construct datasets, similar to them methodology used in information retrieval (Jones and Rijsbergen, 1975; Harman, 1993). For instance, at the annual NIST TAC KBP evaluation, all relations predicted by participating systems are pooled together, annotated and released as a dataset for researchers to develop and evaluate their systems on. However, during development, if a new system predicts a previously unseen relation it is considered to be wrong even if it is correct. The discrepancy between a system's true score and the score on the pooled dataset is called *pooling bias* and is typically assumed to be insignificant in practice (Zobel, 1998).

The key finding of this paper contradicts this assumption and shows that the pooling bias is actu-

[*] Authors contributed equally.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1038–1048
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

ally significant, and it penalizes newly developed systems by 2% F_1 on average (Section 3). Novel improvements, which typically increase scores by less than 1% F_1 on existing datasets, are therefore likely to be clouded by pooling bias during development. Worse, the bias is larger for a system which predicts qualitatively different relations systematically missing from the pool. Of course, systems participating in the TAC KBP evaluation do not suffer from pooling bias, but this requires researchers to wait a year to get credible feedback on new ideas.

This bias is particularly counterproductive for machine learning methods as they are trained assuming the pool is the complete set of positives. Predicting unseen relations and learning novel patterns is penalized. The net effect is that researchers are discouraged from developing innovative approaches, in particular from applying machine learning, thereby slowing progress on the task.

Our second contribution, described in Section 4, addresses this bias through a new evaluation methodology, *on-demand evaluation*, which avoids pooling bias by querying crowdworkers, while minimizing cost by leveraging previous systems' predictions when possible. We then compute the new system's score based on the predictions of past systems using importance weighting. As more systems are evaluated, the marginal cost of evaluating a new system decreases. We show how the on-demand evaluation methodology can be applied to knowledge base population in Section 5. Through a simulated experiment on evaluation data released through the TAC KBP 2015 Slot Validation track, we show that we are able to obtain unbiased estimates of a new systems score's while significantly reducing variance.

Finally, our third contribution is an implementation of our framework as a publicly available evaluation service at https://kbpo.stanford.edu, where researchers can have their own KBP systems evaluated. The data collected through the evaluation process could even be valuable for relation extraction, entity linking and coreference, and will also be made publicly available through the website. We evaluate three systems on the 2016 TAC KBP corpus for about $150 each (a fraction of the cost of official evaluation). We believe the public availability of this service will speed the pace of progress in developing KBP systems.

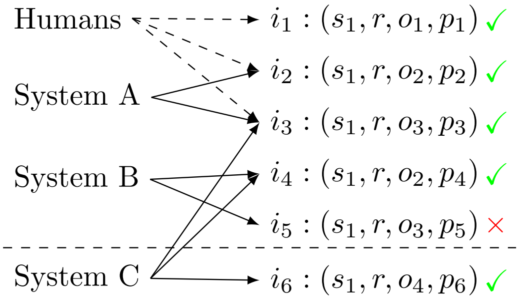

Humans
System A
System B
System C
$i_1 : (s_1, r, o_1, p_1)$ ✓
$i_2 : (s_1, r, o_2, p_2)$ ✓
$i_3 : (s_1, r, o_3, p_3)$ ✓
$i_4 : (s_1, r, o_2, p_4)$ ✓
$i_5 : (s_1, r, o_3, p_5)$ ✗
$i_6 : (s_1, r, o_4, p_6)$ ✓

Figure 2: In pooled evaluation, an evaluation dataset is constructed by labeling relation instances collected from the pooled systems (A and B) and from a team of human annotators (Humans). However, when a new system (C) is evaluated on this dataset, some of its predictions (i_6) are missing and can not be fairly evaluated. Here, the precision and recall for C should be $\frac{3}{3}$ and $\frac{3}{4}$ respectively, but its evaluation scores are estimated to be $\frac{2}{3}$ and $\frac{2}{3}$. The discrepancy between these two scores is called *pooling bias*.

2 Background

In knowledge base population, each relation is a triple (SUBJECT, PREDICATE, OBJECT) where SUBJECT and OBJECT are some globally unique entity identifiers (e.g. Wikipedia page titles) and PREDICATE belongm to a specified schema.[1] A KBP system returns an output in the form of *relation instances* (SUBJECT, PREDICATE, OBJECT, PROVENANCE), where PROVENANCE is a description of where exactly in the document corpus the relation was found. In the example shown in Figure 1, CARRIE FISHER and DEBBIE REYNOLDS are identified as the subject and object, respectively, of the predicate CHILD OF, and the whole sentence is provided as provenance. The provenance also identifies that CARRIE FISHER is referenced by **Fisher** within the sentence. Note that the same relation can be expressed in multiple sentences across the document corpus; each of these is a different relation instance.

Pooled evaluation. The primary source of evaluation data for KBP comes from the annual TAC KBP competition organized by NIST (Ji et al.,

[1] The TAC KBP guidelines specify a total of 65 predicates (including inverses) such as per:title or org:founded_on, etc. Subject entities can be people, organizations, geopolitical entities, while object entities also include dates, numbers and arbitrary string-values like job titles.

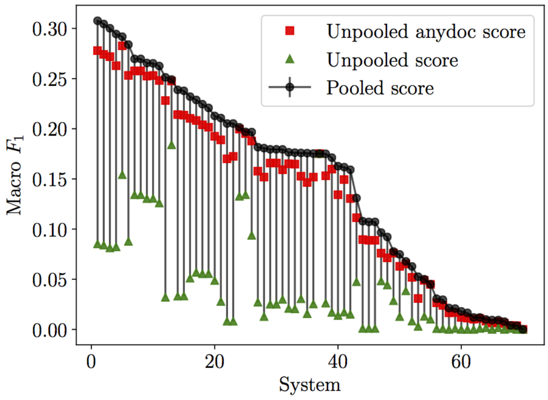
2011). Let E be a held-out set of *evaluation entities*. There are two steps performed in parallel: First, each participating system is run on the document corpus to produce a set of relation instances; those whose subjects are in E are labeled as either positive or negative by annotators. Second, a team of annotators identify and label correct relation instances for the evaluation entities E by manually searching the document corpus within a time budget (Ellis et al., 2012). These labeled relation instances from the two steps are combined and released as the evaluation dataset. In the example in Figure 2, systems A and B were used in constructing the pooling dataset, and there are 3 distinct relations in the dataset, between s_1 and o_1, o_2, o_3.

A system is evaluated on the precision of its predicted relation instances for the evaluation entities E and on the recall of the corresponding predicted *relations* (not instances) for the same entities (see Figure 2 for a worked example). When using the evaluation data during system development, it is common practice to use the more lenient `anydoc` score that ignores the provenance when checking if a relation instance is true. Under this metric, predicting the relation (CARRIE FISHER, CHILD OF, DEBBIE REYNOLDS) from an ambiguous provenance like "**Carrie Fisher** and **Debbie Reynolds** arrived together at the awards show" would be considered correct even though it would be marked wrong under the official metric.

3 Measuring pooling bias

The example in Figure 2 makes it apparent that pooling-based evaluation can introduce a systematic bias against unpooled systems. However, it has been assumed that the bias is insignificant in practice given the large number of systems pooled in the TAC KBP evaluation. We will now show that the assumption is not valid using data from the TAC KBP 2015 evaluation.[2]

Measuring bias. In total, there are 70 system submissions from 18 teams for 317 evaluation entities (E) and the evaluation set consists of 11,008 labeled relation instances.[3] The original evaluation dataset gives us a good measure of the true scores for the participating systems. Similar to Zobel (1998), which studied pooling bias in information retrieval, we simulate the condition of a team not being part of the pooling process by removing any predictions that are unique to its systems from the evaluation dataset. The pooling bias is then the difference between the true and unpooled scores.



	Median bias		
	Precision	Recall	Macro F_1
Official	17.93%	17.00%	15.51%
`anydoc`	2.34%	1.93%	2.05%

Figure 3: Median pooling bias (difference between pooled and unpooled scores) on the top 40 systems of TAC KBP 2015 evaluation using the official and `anydoc` scores. The bias is much smaller for the lenient `anydoc` metric, but even so, it is larger than the largest difference between adjacent systems (1.5% F_1) and typical system improvements (around 1% F_1).

Results. Figure 3 shows the results of measuring pooling bias on the TAC KBP 2015 evaluation on the F_1 metric using the official and `anydoc` scores.[4][5] We observe that even with lenient `anydoc` heuristic, the median bias (2.05% F_1) is much larger than largest difference between adjacently ranked systems (1.5% F_1). This experiment shows that pooling evaluation is significantly and systematically biased against systems that make novel predictions!

[2] Our results are not qualitatively different on data from previous years of the shared task.

[3] The evaluation set is actually constructed from compositional queries like, "what does Carrie Fisher's parents do?": these queries select relation instances that answer the question "who are Carrie Fisher's parents?", and then use those answers (e.g. "Debbie Reynolds") to select relation instances that answer "what does Debbie Reynolds do?". We only consider instances selected in the first part of this process.

[4] We note that `anydoc` scores are on average 0.88%F_1 larger than the official scores.

[5] The outlier at rank 36 corresponds to a University of Texas, Austin system that only filtered predictions from other systems and hence has no unique predictions itself.

4 On-demand evaluation with importance sampling

Pooling bias is fundamentally a sampling bias problem where relation instances from new systems are underrepresented in the evaluation dataset. We could of course sidestep the problem by exhaustively annotating the entire document corpus, by annotating all mentions of entities and checking relations between all pairs of mentions. However, that would be a laborious and prohibitively expensive task: using the interfaces we've developed (Section 6), it costs about $15 to annotate a single document by non-expert crowdworkers, resulting in an estimated cost of at least $1,350,000 for a reasonably large corpus of 90,000 documents (Dang, 2016). The annotation effort would cost significantly more with expert annotators. In contrast, *labeling* relation instances from system predictions can be an order of magnitude cheaper than finding them in documents: using our interfaces, it costs only about $0.18 to verify each relation instance compared to $1.60 per instance extracted through exhaustive annotations.

We propose a new paradigm called on-demand evaluation which takes a lazy approach to dataset construction by annotating predictions from systems *only when they are underrepresented*, thus correcting for pooling bias as it arises. In this section, we'll formalize the problem solved by on-demand evaluation independent of KBP and describe a cost-effective solution that allows us to accurately estimate evaluation scores without bias using importance sampling. We'll then instantiate the framework for KBP in Section 5.

4.1 Problem statement

Let $\mathcal{X}$ be the universe of (relation) instances, $\mathcal{Y} \subseteq \mathcal{X}$ be the unknown subset of correct instances, $X_1, \ldots X_m \subseteq \mathcal{X}$ be the predictions for m systems, and let $Y_i = X_i \cap \mathcal{Y}$. Let $X = \bigcup_{i=1}^{m} X_i$ and $Y = \bigcup_{i=1}^{m} Y_i$. Let $f(x) \stackrel{\text{def}}{=} \mathbb{I}[x \in \mathcal{Y}]$ and $g_i(x) = \mathbb{I}[x \in X_i]$, then the precision, π_i, and recall, r_i, of the set of predictions X_i is

$$\pi_i \stackrel{\text{def}}{=} \mathbb{E}_{x \sim p_i}[f(x)] \qquad r_i \stackrel{\text{def}}{=} \mathbb{E}_{x \sim p_0}[g_i(x)],$$

where p_i is a distribution over X_i and p_0 is a distribution over $\mathcal{Y}$. We assume that p_i is known, e.g. the uniform distribution over X_i and that we know p_0 up to normalization constant and can sample from it.

In on-demand evaluation, we can query $f(x)$ (e.g. labeling an instance) or draw a sample from p_0; typically, querying $f(x)$ is significantly cheaper than sampling from p_0. We obtain prediction sets $X_1, \ldots, X_m$ sequentially as the systems are submitted for evaluation. Our goal is to estimate π_i and r_i for each system $i = 1, \ldots, m$.

4.2 Simple estimators

We can estimate each π_i and r_i independently with simple Monte Carlo integration. Let $\hat{X}_1, \ldots, \hat{X}_m$ be multi-sets of $n_1, \ldots, n_j$ i.i.d. samples from $X_1, \ldots, X_m$ respectively, and let $\hat{Y}_0$ be a multiset of n_0 samples drawn from $\mathcal{Y}$. Then, the simple estimators for precision and recall are:

$$\hat{\pi}_i^{(\text{simple})} = \frac{1}{n_i} \sum_{x \in \hat{X}_i} f(x) \quad \hat{r}_i^{(\text{simple})} = \frac{1}{n_0} \sum_{x \in \hat{Y}_0} g_i(x).$$

4.3 Joint estimators[6]

The simple estimators are unbiased but have wastefully large variance because evaluating a new system does not leverage labels acquired for previous systems.

On-demand evaluation with the joint estimator works as follows: First $\hat{Y}_0$ is randomly sampled from $\mathcal{Y}$ once when the evaluation framework is launched. For every new set of predictions X_m submitted for evaluation, the minimum number of samples n_m required to accurately evaluate X_m is calculated based on the current evaluation data, $\hat{Y}_0$ and $\hat{X}_1, \ldots, \hat{X}_{m-1}$. Then, the set $\hat{X}_m$ is added to the evaluation data by evaluating $f(x)$ on n_m samples drawn from X_m. Finally, estimates π_i and r_i are updated for each system $i = 1, \ldots, m$ using the joint estimators that will be defined next. In the rest of this section, we will answer the following three questions:

1. How can we use all the samples $\hat{X}_1, \ldots \hat{X}_m$ when estimating the precision π_i of system i?
2. How can we use all the samples $\hat{X}_1, \ldots, \hat{X}_m$ with $\hat{Y}_0$ when estimating recall r_i?
3. Finally, to form $\hat{X}_m$, how many samples should we draw from X_m given existing samples and $\hat{X}_1, \ldots, \hat{X}_{m-1}$ and $\hat{Y}_0$?

Estimating precision jointly. Intuitively, if two systems have very similar predictions X_i and X_j,

[6] Proofs for claims made in this section can be found in Appendix B of the supplementary material.

we should be able to use samples from one to estimate precision on the other. However, it might also be the case that X_i and X_j only overlap on a small region, in which case the samples from X_j do not accurately represent instances in X_i and could lead to a biased estimate. We address this problem by using importance sampling (Owen, 2013), a standard statistical technique for estimating properties of one distribution using samples from another distribution.

In importance sampling, if $\hat{X}_i$ is sampled from q_i, then $\frac{1}{n_i}\sum_{x\in\hat{X}_i}\frac{p_i(x)}{q_i(x)}f(x)$ is an unbiased estimate of π_i. We would like the proposal distribution q_i to both leverage samples from all m systems and be tailored towards system i. To this end, we first define a distribution over systems j, represented by probabilities w_{ij}. Then, define q_i as sampling a j and drawing $x \sim p_j$; formally $q_i(x) = \sum_{j=1}^{m} w_{ij}p_j(x)$.

We note that $q_i(x)$ not only significantly differs between systems, but also changes as new systems are added to the evaluation pool. Unfortunately, the standard importance sampling procedure requires us to draw and use samples from each distribution $q_i(x)$ independently and thus can not effectively reuse samples drawn from different distributions. To this end, we introduce a practical refinement to the importance sampling procedure: we independently draw n_j samples according to $p_j(x)$ from each of the m systems independently and then numerically integrate over these samples using the weights w_{ij} to "mix" them appropriately to produce and unbiased estimate of π_i while reducing variance. Formally, we define the *joint precision estimator*:

$$\hat{\pi}_i^{\text{(joint)}} \overset{\text{def}}{=} \sum_{j=1}^{m} \frac{w_{ij}}{n_j} \sum_{x\in\hat{X}_j} \frac{p_i(x)f(x)}{q_i(x)},$$

where each $\hat{X}_j$ consists of n_j i.i.d. samples drawn from p_j.

It is a hard problem to determine what the optimal mixing weights w_{ij} should be. However, we can formally verify that if X_i and X_j are disjoint, then $w_{ij} = 0$ minimizes the variance of π_i, and if $X_i = X_j$, then $w_{ij} \propto n_j$ is optimal. This motivates the following heuristic choice which interpolates between these two extremes: $w_{ij} \propto n_j \sum_{x\in\mathcal{X}} p_j(x)p_i(x)$.

Estimating recall jointly. The recall of system i can be expressed can be expressed as a product $r_i = \theta\nu_i$, where θ is the *recall of the pool*, which measures the fraction of all positive instances predicted by the pool (any system), and ν_i is the *pooled recall of system* i, which measures the fraction of the pool's positive instances predicted by system i. Letting $g(x) \overset{\text{def}}{=} \mathbb{I}[x \in X]$, we can define these as:

$$\nu_i \overset{\text{def}}{=} \mathbb{E}_{x\sim p_0}[g_i(x) \mid x \in X] \quad \theta \overset{\text{def}}{=} \mathbb{E}_{x\sim p_0}[g(x)].$$

We can estimate θ analogous to the simple recall estimator $\hat{r}_i$, except we use the pool g instead a system g_i. For ν_i, the key is to leverage the work from estimating precision. We already evaluated $f(x)$ on $\hat{X}_i$, so we can compute $\hat{Y}_i \overset{\text{def}}{=} \hat{X}_i \cap \mathcal{Y}$ and form the subset $\hat{Y} = \bigcup_{i=1}^{m} \hat{Y}_i$. $\hat{Y}$ is an approximation of $\mathcal{Y}$ whose bias we can correct through importance reweighting. We then define estimators as follows:

$$\hat{\nu}_i \overset{\text{def}}{=} \frac{\sum_{j=1}^{m}\frac{w_{ij}}{n_j}\sum_{x\in\hat{Y}_j}\frac{p_0(x)g_i(x)}{q_i(x)}}{\sum_{j=1}^{m}\frac{w_{ij}}{n_j}\sum_{x\in\hat{Y}_j}\frac{p_0(x)}{q_i(x)}}$$

$$\hat{r}_i^{\text{(joint)}} \overset{\text{def}}{=} \hat{\theta}\hat{\nu}_i \quad \hat{\theta} \overset{\text{def}}{=} \frac{1}{n_0}\sum_{x\in\hat{Y}_0} g(x).$$

where q_i and w_{ij} are the same as before.

Adaptively choosing the number of samples. Finally, a desired property for on-demand evaluation is to label new instances only when the current evaluation data is insufficient, e.g. when a new set of predictions X_m contains many instances not covered by other systems. We can measure how well the current evaluation set covers the predictions X_m by using a conservative estimate of the variance of $\hat{\pi}_m^{\text{(joint)}}$.[7] In particular, the variance of $\hat{\pi}_m^{\text{(joint)}}$ is a monotonically decreasing function in n_m, the number of samples drawn from X_m. We can easily solve for the minimum number of samples required to estimate $\hat{\pi}_m^{\text{(joint)}}$ within a confidence interval ϵ by using the bisection method (Burden and Faires, 1985).

5 On-demand evaluation for KBP

Applying the on-demand evaluation framework to a task requires us to answer three questions:

1. What is the desired distribution over system predictions p_i?

[7] Further details can be found in Appendix B of the supplementary material.

2. How do we label an instance x, i.e. check if $x \in \mathcal{Y}$?
3. How do we sample from the unknown set of true instances $x \sim p_0$?

In this section, we present practical implementations for knowledge base population.

5.1 Sampling from system predictions

Both the official TAC-KBP evaluation and the on-demand evaluation we propose use micro-averaged precision and recall as metrics. However, in the official evaluation, these metrics are computed over a fixed set of evaluation entities chosen by LDC annotators, resulting in two problems: (a) defining evaluation entities requires human intervention and (b) typically a large source of variability in evaluation scores comes from not having enough evaluation entities (see e.g. (Webber, 2010)). In our methodology, we replace manually chosen evaluation entities by sampling entities from each system's output according p_i. In effect, p_i makes explicit the decision process of the annotator who chooses evaluation entities.

Identifying a reasonable distribution p_i is an important implementation decision that depends on what one wishes to evaluate. Our goal for the on-demand evaluation service we have implemented is to ensure that KBP systems are fairly evaluated on diverse subjects and predicates, while at the same time, ensuring that entities with multiple relations are represented to measure completeness of knowledge base entries. As a result, we propose a distribution that is inversely proportional to the frequency of the subject and predicate and is proportional to the number of unique relations identified for an entity (to measure knowledge base completeness). See Appendix A in the supplementary material for an analysis of this distribution and a study of other potential choices.

5.2 Labeling predicted instances

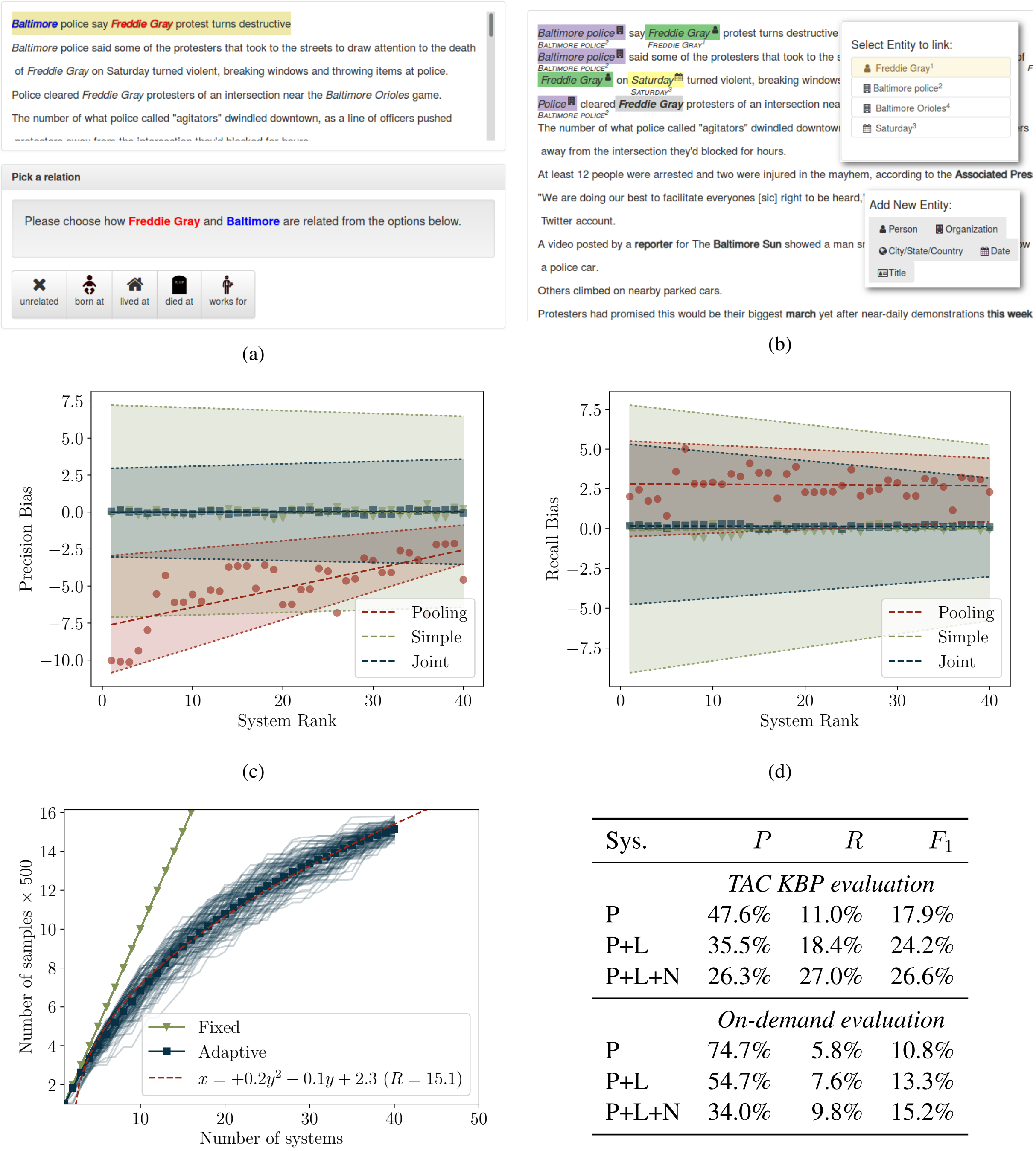
We label predicted relation instances by presenting the instance's provenance to crowdworkers and asking them to identify if a relation holds between the identified subject and object mentions (Figure 4a). Crowdworkers are also asked to link the subject and object mentions to their canonical mentions within the document and to pages on Wikipedia, if possible, for entity linking. On average, we find that crowdworkers are able to perform this task in about 20 seconds, corresponding to about \$0.05 per instance. We requested 5 crowdworkers to annotate a small set of 200 relation instances from the 2015 TAC-KBP corpus and measured a substantial inter-annotator agreement with a Fleiss' kappa of 0.61 with 3 crowdworkers and 0.62 with 5. Consequently, we take a majority vote over 3 workers in subsequent experiments.

5.3 Sampling true instances

Sampling from the set of true instances $\mathcal{Y}$ is difficult because we can't even enumerate the elements of $\mathcal{Y}$. As a proxy, we assume that relations are identically distributed across documents and have crowdworkers annotate a random subset of documents for relations using an interface we developed (Figure 4b). Crowdworkers begin by identifying every mention span in a document. For each mention, they are asked to identify its type, canonical mention within the document and associated Wikipedia page if possible. They are then presented with a separate interface to label predicates between pairs of mentions within a sentence that were identified earlier.

We compare crowdsourced annotations against those of expert annotators using data from the TAC KBP 2015 EDL task on 10 randomly chosen documents. We find that 3 crowdworkers together identify 92% of the entity spans identified by expert annotators, while 7 crowdworkers together identify 96%. When using a token-level majority vote to identify entities, 3 crowdworkers identify about 78% of the entity spans; this number does not change significantly with additional crowdworkers. We also measure substantial token-level inter-annotator agreement using Fleiss' kappa for identifying typed mention spans ($\kappa = 0.83$), canonical mentions ($\kappa = 0.75$) and entity links ($\kappa = 0.75$) with just three workers. Based on this analysis, we use token-level majority over 3 workers in subsequent experiments.

The entity annotation interface is far more involved and takes on average about 13 minutes per document, corresponding to about \$2.60 per document, while the relation annotation interface takes on average about \$2.25 per document. Because documents vary significantly in length and complexity, we set rewards for each document based on the number of tokens (.75c per token) and mention pairs (5c per pair) respectively. With 3 workers per document, we paid about \$15 per document on average. Each document contained an average



Baltimore police say *Freddie Gray* protest turns destructive

Baltimore police said some of the protesters that took to the streets to draw attention to the death of *Freddie Gray* on Saturday turned violent, breaking windows and throwing items at police.

Police cleared *Freddie Gray* protesters of an intersection near the *Baltimore Orioles* game.

The number of what police called "agitators" dwindled downtown, as a line of officers pushed

protesters away from the intersection they'd blocked for hours

Pick a relation

Please choose how **Freddie Gray** and **Baltimore** are related from the options below.

(a)

Baltimore police say *Freddie Gray* protest turns destructive

BALTIMORE POLICE[2] FREDDIE GRAY[1]

Baltimore police said some of the protesters that took to the s

BALTIMORE POLICE[2]

Freddie Gray on *Saturday* turned violent, breaking windows

SATURDAY[3]

Police cleared ***Freddie Gray*** protesters of an intersection nea

BALTIMORE POLICE[2]

The number of what police called "agitators" dwindled downtowr

away from the intersection they'd blocked for hours.

At least 12 people were arrested and two were injured in the mayhem, according to the **Associated Pres**

"We are doing our best to facilitate everyones [sic] right to be heard,'

Twitter account.

A video posted by a **reporter** for The **Baltimore Sun** showed a man sı

a police car.

Others climbed on nearby parked cars.

Protesters had promised this would be their biggest **march** yet after near-daily demonstrations **this week**

Select Entity to link:

- Freddie Gray[1]
- Baltimore police[2]
- Baltimore Orioles[4]
- Saturday[3]

Add New Entity:

(b)

(c)

(d)

(e)

Sys.	P	R	F_1
	TAC KBP evaluation		
P	47.6%	11.0%	17.9%
P+L	35.5%	18.4%	24.2%
P+L+N	26.3%	27.0%	26.6%
	On-demand evaluation		
P	74.7%	5.8%	10.8%
P+L	54.7%	7.6%	13.3%
P+L+N	34.0%	9.8%	15.2%

(f)

Figure 4: **(a, b):** Interfaces for annotating relations and entities respectively. **(c, d):** A comparison of bias for the pooling, simple and joint estimators on the TAC KBP 2015 challenge. Each point in the figure is a mean of 500 repeated trials; dotted lines show the 90% quartile. Both the simple and joint estimators are unbiased, and the joint estimator is able to significantly reduce variance. **(e):** A comparison of the number of samples used to estimate scores under the fixed and adaptive sample selection scheme. Each faint line shows the number of samples used during a single trial, while solid lines show the mean over 100 trials. The dashed line shows a square-root relationship between the number of systems evaluated and the number of samples required. Thus joint estimation combined with adaptive sample selection can reduce the number of labeled annotations required by an order of magnitude. **(f):** Precision (P), recall (R) and F_1 scores from a pilot run of our evaluation service for ensembles of a rule-based system (R), a logistic classifier (L) and a neural network classifier (N) run on the TAC KBP 2016 document corpus.

9.2 relations, resulting in a cost of about $1.61 per relation instance. We note that this is about ten times as much as labeling a relation instance.

We defer details regarding how documents themselves should be weighted to capture diverse entities that span documents to Appendix A.

6 Evaluation

Let us now see how well on-demand evaluation works in practice. We begin by empirically studying the bias and variance of the joint estimator proposed in Section 4 and find it is able to correct for pooling bias while significantly reducing variance in comparison with the simple estimator. We then demonstrate that on-demand evaluation can serve as a practical replacement for the TAC KBP evaluations by piloting a new evaluation service we have developed to evaluate three distinct systems on TAC KBP 2016 document corpus.

6.1 Bias and variance of the on-demand evaluation.

Once again, we use the labeled system predictions from the TAC KBP 2015 evaluation and treat them as an exhaustively annotated dataset. To evaluate the pooling methodology we construct an evaluation dataset using instances found by human annotators and labeled instances pooled from 9 randomly chosen teams (i.e. half the total number of participating teams), and use this dataset to evaluate the remaining 9 teams. On average, the pooled evaluation dataset contains between 5,000 and 6,000 labeled instances and evaluates 34 different systems (since each team may have submitted multiple systems). Next, we evaluated sets of 9 randomly chosen teams with our proposed simple and joint estimators using a total of 5,000 samples: about 150 of these samples are drawn from $\mathcal{Y}$, i.e. the full TAC KBP 2015 evaluation data, and 150 samples from each of the systems being evaluated.

We repeat the above simulated experiment 500 times and compare the estimated precision and recall with their true values (Figure 4). The simulations once again highlights that the pooled methodology is biased, while the simple and joint estimators are not. Furthermore, the joint estimators significantly reduce variance relative to the simple estimators: the median 90% confidence intervals reduce from 0.14 to 0.06 precision and from 0.14 to 0.08 for recall.

6.2 Number of samples required by on-demand evaluation

Separately, we evaluate the efficacy of the adaptive sample selection method described in Section 4.3 through another simulated experiment. In each trial of this experiment, we evaluate the top 40 systems in random order. As each subsequent system is evaluated, the number of samples to pick from the system is chosen to meet a target variance and added to the current pool of labeled instances. To make the experiment more interpretable, we choose the target variance to correspond with the estimated variance of having 500 samples. Figure 4 plots the results of the experiment. The number of samples required to estimate systems quickly drops off from the benchmark of 500 samples as the pool of labeled instances covers more systems. This experiment shows that on-demand evaluation using joint estimation can scale up to an order of magnitude more submissions than a simple estimator for the same cost.

6.3 A mock evaluation for TAC KBP 2016

We have implemented the on-demand evaluation framework described here as an evaluation service to which researchers can submit their own system predictions. As a pilot of the service, we evaluated three relation extraction systems that also participated in the official 2016 TAC KBP competition. Each system uses Stanford CoreNLP (Manning et al., 2014) to identify entities, the Illinois Wikifier (Ratinov et al., 2011) to perform entity linking and a combination of a rule-based system (P), a logistic classifier (L), and a neural network classifier (N) for relation extraction. We used 15,000 Newswire documents from the 2016 TAC KBP evaluation as our document corpus. In total, 100 documents were exhaustively annotated for about $2,000 and 500 instances from each system were labeled for about $150 each. Evaluating all three system only took about 2 hours.

Figure 4f reports scores obtained through on-demand evaluation of these systems as well as their corresponding official TAC evaluation scores. While the relative ordering of systems between the two evaluations is the same, we note that precision and recall as measured through on-demand evaluation are respectively higher and lower than the official scores. This is to be expected because on-demand evaluation measures precision using each systems output as opposed to an externally defined set of evaluation entities. Likewise, recall is measured using exhaustive annotations of relations within the corpus instead of annotations from pooled output in the official evaluation.

7 Related work

The subject of pooling bias has been extensively studied in the information retrieval (IR) community starting with Zobel (1998), which examined the effects of pooling bias on the TREC AdHoc task, but concluded that pooling bias was not a significant problem. However, when the topic was later revisited, Buckley et al. (2007) identified that the reason for the small bias was because the submissions to the task were too similar; upon repeating the experiment using a novel system as part of the TREC Robust track, they identified a 23% point drop in average precision scores![8]

Many solutions to the pooling bias problem have been proposed in the context of information retrieval, e.g. adaptively constructing the pool to collect relevant data more cost-effectively (Zobel, 1998; Cormack et al., 1998; Aslam et al., 2006), or modifying the scoring metrics to be less sensitive to unassessed data (Buckley and Voorhees, 2004; Sakai and Kando, 2008; Aslam et al., 2006). Many of these ideas exploit the ranking of documents in IR which does not apply to KBP. While both Aslam et al. (2006) and Yilmaz et al. (2008) estimate evaluation metrics by using importance sampling estimators, the techniques they propose require knowing the set of all submissions beforehand. In contrast, our on-demand methodology can produce unbiased evaluation scores for new development systems as well.

There have been several approaches taken to crowdsource data pertinent to knowledge base population (Vannella et al., 2014; Angeli et al., 2014; He et al., 2015; Liu et al., 2016). The most extensive annotation effort is probably Pavlick et al. (2016), which crowdsources a knowledge base for gun-violence related events. In contrast to previous work, our focus is on *evaluating systems*, not collecting a dataset. Furthermore, our main contribution is not a large dataset, but an evaluation service that allows anyone to use crowdsourcing predictions made by their system.

8 Discussion

Over the last ten years of the TAC KBP task, the gap between human and system performance has barely narrowed despite the community's best efforts: top automated systems score less than 36% F_1 while human annotators score more than 60%. In this paper, we've shown that the current evaluation methodology may be a contributing factor because of its bias against novel system improvements. The new on-demand framework proposed in this work addresses this problem by obtaining human assessments of new system output through crowdsourcing. The framework is made economically feasible by carefully sampling output to be assessed and correcting for sample bias through importance sampling.

Of course, simply providing better evaluation scores is only part of the solution and it is clear that better datasets are also necessary. However, the very same difficulties in scale that make evaluating KBP difficult also make it hard to collect a high quality dataset for the task. As a result, existing datasets (Angeli et al., 2014; Adel et al., 2016) have relied on the output of existing systems, making it likely that they exhibit the same biases against novel systems that we've discussed in this paper. We believe that providing a fair and standardized evaluation platform as a service allows researchers to exploit such datasets and while still being able to accurately measure their performance on the knowledge base population task.

There are many other tasks in NLP that are even harder to evaluate than KBP. Existing evaluation metrics for tasks with a generation component—such as summarization or dialogue—leave much to be desired. We believe that adapting the ideas of this paper to those tasks is a fruitful direction, as progress of a research community is strongly tied to the fidelity of evaluation.

Acknowledgments

We would like to thank Yuhao Zhang, Hoa Deng, Eduard Hovy, and Jacob Steinhardt for discussions, William E. Webber for his excellent thesis that helped shape this project and the anonymous reviewers for their detailed and pertinent feedback. The first and second authors are supported under DARPA DEFT program under ARFL prime contract no. FA8750-13-2-0040.

[8] For the interested reader, Webber (2010) presents an excellent survey of the literature on pooling bias.

References

H. Adel, B. Roth, and H. Schütze. 2016. Comparing convolutional neural networks to traditional models for slot filling. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*.

G. Angeli, J. Tibshirani, J. Y. Wu, and C. D. Manning. 2014. Combining distant and partial supervision for relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. A. Aslam, V. Pavlu, and E. Yilmaz. 2006. A statistical method for system evaluation using incomplete judgments. In *ACM Special Interest Group on Information Retreival (SIGIR)*, pages 541–548.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

C. Buckley, D. Dimmick, I. Soboroff, and E. Voorhees. 2007. Bias and the limits of pooling for large collections. In *ACM Special Interest Group on Information Retreival (SIGIR)*.

C. Buckley and E. M. Voorhees. 2004. Retrieval evaluation with incomplete information. In *ACM Special Interest Group on Information Retreival (SIGIR)*, pages 25–32.

R. L. Burden and J. D. Faires. 1985. *Numerical Analysis (3rd ed.)*. PWS Publishers.

G. V. Cormack, C. R. Palmer, and C. L. A. Clarke. 1998. Efficient construction of large test collections. In *ACM Special Interest Group on Information Retreival (SIGIR)*.

H. T. Dang. 2016. Cold start knowledge base population at TAC KBP 2016. *Text Analytics Conference*.

J. Ellis, X. Li, K. Griffitt, and S. M. Strassel. 2012. Linguistic resources for 2012 knowledge base population evaluations. *Text Analytics Conference*.

A. Fader, L. Zettlemoyer, and O. Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1156–1165.

S. Han, J. Bang, S. Ryu, and G. G. Lee. 2015. Exploiting knowledge base to generate responses for natural language dialog listening agents. *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 129–133.

D. K. Harman. 1993. The first text retrieval conference (trec-1) rockville, md, u.s.a., 4-6 november, 1992. *Information Processing and Management*, 29:411–414.

L. He, M. Lewis, and L. Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Empirical Methods in Natural Language Processing (EMNLP)*.

H. Ji, R. Grishman, and H. Trang Dang. 2011. Overview of the TAC 2011 knowledge base population track. In *Text Analytics Conference*.

K. S. Jones and C. V. Rijsbergen. 1975. Report on the need for and provision of an "ideal test collection. *Information Retrieval Test Collection*.

A. Kalyanpur, B. K. Boguraev, S. Patwardhan, J. W. Murdock, A. Lally, C. A. Welty, J. M. Prager, B. Coppola, A. Fokoue-Nkoutche, L. Zhang, Y. Pan, and Z. M. Qui. 2012. Structured data and inference in deepqa. *IBM Journal of Research and Development*, 56:351–364.

A. Liu, S. Soderland, J. Bragg, C. H. Lin, X. Ling, and D. S. Weld. 2016. Effective crowd annotation for relation extraction. In *North American Association for Computational Linguistics (NAACL)*, pages 897–906.

C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. 2014. The stanford coreNLP natural language processing toolkit. In *ACL system demonstrations*.

A. B. Owen. 2013. *Monte Carlo theory, methods and examples*.

E. Pavlick, H. Ji, X. Pan, and C. Callison-Burch. 2016. The gun violence database: A new task and data set for NLP. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1018–1024.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to Wikipedia. In *Association for Computational Linguistics (ACL)*.

S. Reddy, M. Lapata, and M. Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics (TACL)*, 2(10):377–392.

T. Sakai and N. Kando. 2008. On information retrieval metrics designed for evaluation with incomplete relevance assessments. In *ACM Special Interest Group on Information Retreival (SIGIR)*, pages 447–470.

D. Vannella, D. Jurgens, D. Scarfini, D. Toscani, and R. Navigli. 2014. Validating and extending semantic knowledge bases using video games with a purpose. In *Association for Computational Linguistics (ACL)*, pages 1294–1304.

W. E. Webber. 2010. *Measurement in Information Retrieval Evaluation*. Ph.D. thesis, University of Melbourne.

E. Yilmaz, E. Kanoulas, and J. A. Aslam. 2008. A simple and efficient sampling method for estimating AP and NDCG. In *ACM Special Interest Group on Information Retreival (SIGIR)*, pages 603–610.

J. Zobel. 1998. How reliable are the results of large-scale information retrieval experiments? In *ACM Special Interest Group on Information Retreival (SIGIR)*.

PACRR: A Position-Aware Neural IR Model for Relevance Matching

Kai Hui
MPI for Informatics
Saarbrücken
Graduate School of
Computer Science

Andrew Yates
MPI for Informatics

Klaus Berberich
htw saar
MPI for Informatics

Gerard de Melo
Rutgers University
New Brunswick

Abstract

In order to adopt deep learning for information retrieval, models are needed that can capture all relevant information required to assess the relevance of a document to a given user query. While previous works have successfully captured unigram term matches, how to fully employ position-dependent information such as proximity and term dependencies has been insufficiently explored. In this work, we propose a novel neural IR model named *PACRR* aiming at better modeling position-dependent interactions between a query and a document. Extensive experiments on six years' TREC Web Track data confirm that the proposed model yields better results under multiple benchmarks.

1 Introduction

Despite the widespread use of deep neural models across a range of linguistic tasks, to what extent such models can improve information retrieval (IR) and which components a deep neural model for IR should include remain open questions. In ad-hoc IR, the goal is to produce a ranking of relevant documents given an open-domain ("ad hoc") query and a document collection. A ranking model thus aims at evaluating the interactions between different documents and a query, assigning higher scores to documents that better match the query. Learning to rank models, like the recent IRGAN model (Wang et al., 2017), rely on handcrafted features to encode query document interactions, e.g., the relevance scores from unsupervised ranking models. Neural IR models differ in that they extract interactions directly based on the queries and documents. Many early neural IR models can be categorized as *semantic matching* models, as they embed both queries and documents into a low-dimensional space, and then assess their similarity based on such dense representations. Examples in this regard include *DSSM* (Huang et al., 2013) and *DESM* (Mitra et al., 2016). The notion of relevance is inherently asymmetric, however, making it different from well-studied semantic matching tasks such as semantic relatedness and paraphrase detection. Instead, *relevance matching* models such as MatchPyramid (Pang et al., 2016), DRMM (Guo et al., 2016) and the recent K-NRM (Xiong et al., 2017) resemble traditional IR retrieval measures in that they directly consider the relevance of documents' contents with respect to the query. The DUET model (Mitra et al., 2017) is a hybrid approach that combines signals from a local model for relevance matching and a distributed model for semantic matching. The two classes of models are fairly distinct. In this work, we focus on relevance matching models.

Given that relevance matching approaches mirror ideas from traditional retrieval models, the decades of research on ad-hoc IR can guide us with regard to the specific kinds of relevance signals a model ought to capture. Unigram matches are the most obvious signals to be modeled, as a counterpart to the term frequencies that appear in almost all traditional retrieval models. Beyond this, positional information, including where query terms occur and how they depend on each other, can also be exploited, as demonstrated in retrieval models that are aware of term proximity (Tao and Zhai, 2007) and term dependencies (Huston and Croft, 2014; Metzler and Croft, 2005). Query coverage is another factor that can be used to ensure that, for queries with multiple terms, top-ranked documents contain multiple query terms rather than emphasizing only one query term. For example, given the query

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1049–1058
Copenhagen, Denmark, September 7–11, 2017.

"dog adoption requirements", unigram matching signals correspond to the occurrences of the individual terms "dog", "adoption", or "requirements". When considering positional information, text passages with "dog adoption" or "requirements for dog adoption" are highlighted, distinguishing them from text that only includes individual terms. Query coverage, meanwhile, further emphasizes that matching signals for "dog", "adoption", and "requirements" should all be included in a document.

Similarity signals from unigram matches are taken as input by DRMM (Guo et al., 2016) after being summarized as histograms, whereas K-NRM (Xiong et al., 2017) directly digests a query-document similarity matrix and summarizes it with multiple kernel functions. As for positional information, both the MatchPyramid (Pang et al., 2016) and local DUET (Mitra et al., 2017) models account for it by incorporating convolutional layers based on similarity matrices between queries and documents. Although this leads to more complex models, both have difficulty in significantly outperforming the DRMM model (Guo et al., 2016; Mitra et al., 2017). This indicates that it is non-trivial to go beyond unigrams by utilizing positional information in deep neural IR models. Intuitively, unlike in standard sequence-based models, the interactions between a query and a document are sequential along the query axis as well as along the document axis, making the problem multi-dimensional in nature. In addition, this makes it non-trivial to combine matching signals from different parts of the documents and over different query terms. In fact, we argue that both MatchPyramid and local DUET models fail to fully account for one or more of the aforementioned factors. For example, as a pioneering work, MatchPyramid is mainly motivated by models developed in computer vision, resulting in its disregard of certain IR-specific considerations in the design of components, such as pooling sizes that ignore the query and document dimensions. Meanwhile, local DUET's CNN filters match entire documents against individual query terms, neglecting proximity and possible dependencies among different query terms.

We conjecture that a suitable combination of convolutional kernels and recurrent layers can lead to a model that better accounts for these factors. In particular, we present a novel re-ranking model called *PACRR* (*Position-Aware Convolutional-Recurrent Relevance* Matching). Our approach first produces similarity matrices that record the semantic similarity between each query term and each individual term occurring in a document. These matrices are then fed through a series of convolutional, max-k-pooling, and recurrent layers so as to capture interactions corresponding to, for instance, bigram and trigram matches, and finally to aggregate the signals in order to produce global relevance assessments. In our model, the convolutional layers are designed to capture both unigram matching and positional information over text windows with different lengths; k-max pooling layers are along the query dimension, preserving matching signals over different query terms; the recurrent layer combines signals from different query terms to produce a query-document relevance score.

Organization. The rest of this paper unfolds as follows. Section 2 describes our approach for computing similarity matrices and the architecture of our deep learning model. The setup and results of our extensive experimental evaluation can be found in Section 3, before concluding in Section 4.

2 The PACRR Model

We now describe our proposed *PACRR* approach, which consists of two main parts: a relevance matching component that converts each query-document pair into a similarity matrix $sim_{|q|\times|d|}$ and a deep architecture that takes a given query-document similarity matrix as input and produces a query-document relevance score $rel(q,d)$. Note that in principle the proposed model can be trained end-to-end by backpropagating through the word embeddings, as in (Xiong et al., 2017). In this work, however, we focus on highlighting the building blocks aiming at capturing positional information, and freeze the word embedding layer to achieve better efficiency. The pipeline is summarized in Figure 1.

2.1 Relevance Matching

We first encode the query-document relevance matching via query-document similarity matrices $sim_{|q|\times|d|}$ that encodes the similarity between terms from a query q and a document d, where sim_{ij} corresponds to the similarity between the i-th term from q and the j-th term from d. When using cosine similarity, we have

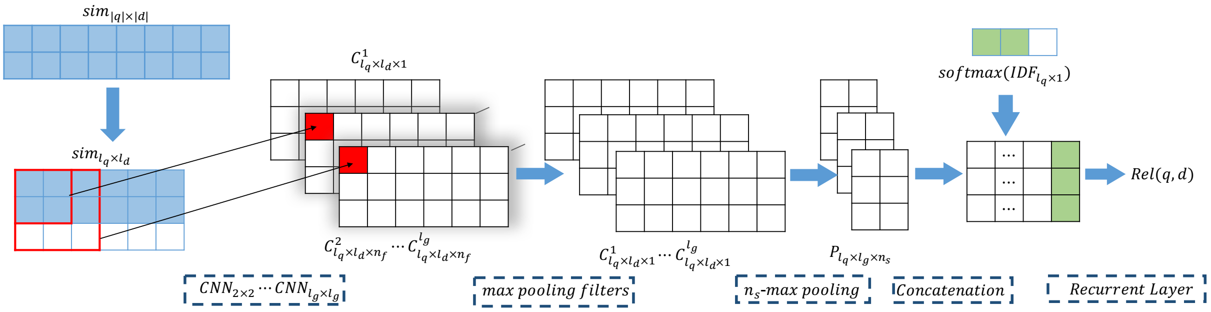

$sim_{|q|\times|d|}$
$sim_{l_q\times l_d}$
$C^1_{l_q\times l_d\times 1}$
$C^2_{l_q\times l_d\times n_f}\cdots C^{l_g}_{l_q\times l_d\times n_f}$
$C^1_{l_q\times l_d\times 1}\cdots C^{l_g}_{l_q\times l_d\times 1}$
$P_{l_q\times l_g\times n_s}$
$softmax(IDF_{l_q\times 1})$
...
$Rel(q,d)$
$CNN_{2\times 2}\cdots CNN_{l_g\times l_g}$
max pooling filters
n_s-max pooling
Concatenation
Recurrent Layer

Figure 1: The pipeline of *PACRR*. Each query q and document d is first converted into a query-document similarity matrix $sim_{|q|\times|d|}$. Thereafter, a distillation method (*firstk* is displayed) transforms the raw similarity matrix into unified dimensions, namely, $sim_{l_q\times l_d}$. Here, $l_g - 1$ convolutional layers (CNN) are applied to the distilled similarity matrices. As $l_g = 3$ is shown, layers with kernel size 2 and 3 are applied. Next, max pooling is applied, leading to l_g matrices $C^1 \cdots C^{l_g}$. Following this, n_s-max pooling captures the strongest n_s signals over each query term and n-gram size, and the case for $n_s = 2$ is shown here. Finally, the similarity signals from different n-gram sizes are concatenated, the query terms' normalized IDFs are added, and a recurrent layer combines these signals for each query term into a query-document relevance score $rel(q, d)$.

$sim \in [-1, 1]^{|q|\times|d|}$. As suggested in (Hui et al., 2017), query-document similarity matrices preserve a rich signal that can be used to perform relevance matching beyond unigram matches. In particular, n-gram matching corresponds to consecutive document terms that are highly similar to at least one of the query terms. Query coverage is reflected in the number of rows in *sim* that include at least one cell with high similarity. The similarity between a query term q and document term d is calculated by taking the cosine similarity using the pre-trained[1] *word2vec* (Mikolov et al., 2013).

The subsequent processing in PACRR's convolutional layers requires that each query-document similarity matrix have the same dimensionality. Given that the lengths of queries and documents vary, we first transform the raw similarity matrices $sim_{|q|\times|d|}$ into $sim_{l_q\times l_d}$ matrices with uniform l_q and l_d as the number of rows and columns. We unify the query dimension l_q by zero padding it to the maximum query length. With regard to the document dimension l_d, we describe two strategies: *firstk* and *kwindow*.

PACRR-firstk. Akin to (Mitra et al., 2017), the *firstk* distillation method simply keeps the first k columns in the matrix, which correspond to the first k terms in the document. If $k > |d|$, the remaining columns are zero padded.

PACRR-kwindow. As suggested in (Guo et al., 2016), relevance matching is local. Document terms that have a low query similarity relative to a document's other terms cannot contribute substantially to the document's relevance score. Relevance matching can be extracted in terms of pieces of text that include relevant information. That is, one can segment documents according to relevance relative to the given query and retain only the text that is highly relevant to the given query. Given this observation, we prune query-document similarity cells with a low similarity score. In the case of unigrams, we simply choose the top l_d terms with the highest similarity to query terms. In the case for *text snippets* beyond length n, we produce a similarity matrix $sim^n_{l_q\times l_d}$ for each query-document pair and each n, because n consecutive terms must be co-considered later on. For each text snippet with length n in the document, *kwindow* calculates the maximum similarity between each term and the query terms, and then calculates the average similarity over each n-term window. It then selects the top $k = \lfloor l_d/n \rfloor$ windows by averaging similarity and discards all other terms in the document. The document dimension is zero padded if $\lfloor l_d/n \rfloor$ is not a multiple of k. When the convolutional layer later operates on a similarity matrix produced by *kwindow*, the model's stride is set to n (i.e., the sliding window moves ahead n terms at a time rather than one term at a time) since it can consider at most n consecutive terms that are

[1] https://code.google.com/archive/p/word2vec/

present in the original document. This variant's output is a similarity matrix $sim^n_{l_q \times l_d}$ for each n.

2.2 Deep Retrieval Model

Given a query-document similarity matrix $sim_{l_q \times l_d}$ as input, our deep architecture relies on convolutional layers to match every text snippet with length n in a query and in a document to produce similarity signals for different n. Subsequently, two consecutive max pooling layers extract the document's strongest similarity cues for each n. Finally, a recurrent layer aggregates these salient signals to predict a global query-document relevance score $rel(q, d)$.

Convolutional relevance matching over local text snippets. The purpose of this step is to match text snippets with different length from a query and a document given their query-document similarity matrix as input. This is accomplished by applying multiple two-dimensional convolutional layers with different kernel sizes to the input similarity matrix. Each convolutional layer is responsible for a specific n; by applying its kernel on $n \times n$ windows, it produces a similarity signal for each window. When the *firstk* method is used, each convolutional layer receives the same similarity matrix $sim_{l_q \times l_d}$ as input because *firstk* produces the same similarity matrix regardless of the n. When the *kwindow* method is used, each convolutional layer receives a similarity matrix $sim^n_{l_q \times l_d}$ corresponding to the convolutional layer with a $n \times n$ kernel. We use $l_g - 1$ different convolutional layers with kernel sizes $2 \times 2, 3 \times 3, \ldots, l_g \times l_g$, corresponding to bi-gram, tri-gram, ..., l_g-gram matching, respectively, where the length of the longest text snippet to consider is governed by a hyperparameter l_g. The original similarity matrix corresponds to unigram matching, while a convolutional layer with kernel size $n \times n$ is responsible for capturing matching signals on n-term text snippets. Each convolutional layer applies n_f different filters to its input, where n_f is another hyperparameter. We use a stride of size (1, 1) for the *firstk* distillation method, meaning that the convolutional kernel advances one step at a time in both the query and document dimensions. For the *kwindow* distillation method, we use a stride of (1, n) to move the convolutional kernel one step at a time in the query dimension, but n steps at a time in the document dimension. This ensures that the convolutional kernel only operates over consecutive terms that existed in the original document. Thus, we end up with $l_g - 1$ matrices $\mathcal{C}^n_{l_q \times l_d \times n_f}$, and the original similarity matrix is directly employed to handle the signals over unigrams.

Two max pooling layers. The purpose of this step is to capture the n_s strongest similarity signals for each query term. Measuring the similarity signals separately for each query term allows the model to consider query term coverage, while capturing the n_s strongest similarity signals for each query term allows the model to consider signals from different kinds of relevance matching patterns, e.g., n-gram matching and non-contiguous matching. In practice, we use a small n_s to prevent the model from being biased by document length; while each similarity matrix contains the same number of document term scores, longer documents have more opportunity to contain terms that are similar to query terms. To capture the strongest n_s similarity signals for each query term, we first perform max pooling over the filter dimension n_f to keep only the strongest signal from the n_f different filters, assuming that there only exists one particular true matching pattern in a given $n \times n$ window, which serves different purposes compared with other tasks, such as the sub-sampling in computer vision. We then perform k-max pooling (Kalchbrenner et al., 2014) over the query dimension l_q to keep the strongest n_s similarity signals for each query term. Both pooling steps are performed on each of the $l_g - 1$ matrices $\mathcal{C}^i$ from the convolutional layer and on the original similarity matrix, which captures unigram matching, to produce the 3-dimensional tensor $\mathcal{P}_{l_q \times l_g \times n_s}$. This tensor contains the n_s strongest signals for each query term and for each n-gram size across all n_f filters.

Recurrent layer for global relevance. Finally, our model transforms the query term similarity signals in $\mathcal{P}_{l_q \times l_g \times n_s}$ into a single document relevance score $rel(q, d)$. It achieves this by applying a recurrent layer to $\mathcal{P}$, taking a sequence of vectors as input and learning weights to transform them into the final relevance score. More precisely, akin to (Guo et al., 2016), the IDF of each query term q_i is passed through a softmax layer for normalization. Thereafter, we split up the query term dimension to produce a matrix $\mathcal{P}_{l_g \times n_s}$ for each query term q_i, subsequently forming the recurrent layer's input by flattening each matrix $\mathcal{P}_{l_g \times n_s}$ into a vector by concatenating the matrix's rows together and appending query term q_i's normalized

IDF onto the end of the vector. This sequence of vectors for each query term q_i is passed into a Long Short-Term Memory (LSTM) recurrent layer (Hochreiter and Schmidhuber, 1997) with an output dimensionality of one. That is, the LSTM's input is a sequence of query term vectors where each vector is composed of the query term's normalized IDF and the aforementioned salient signals for the query term along different kernel sizes. The LSTM's output is then used as our document relevance score $rel(q, d)$.

Training objective. Our model is trained on triples consisting of a query q, relevant document d^+, and non-relevant document d^-, minimizing a standard pairwise max margin loss as in Eq. 1.

$$\mathcal{L}(q,d^+,d^-;\Theta)=max(0,1-rel(q,d^+)+rel(q,d^-)) \qquad (1)$$

3 Evaluation

In this section, we empirically evaluate *PACRR* models using manual relevance judgments from the standard TREC Web Track. We compare them against several state-of-the-art neural IR models[2], including *DRMM* (Guo et al., 2016), DUET (Mitra et al., 2017), *MatchPyramid* (Pang et al., 2016), and *K-NRM* (Xiong et al., 2017). The comparisons are over three task settings: re-ranking search results from a simple initial ranker (RERANKSIMPLE); re-ranking all runs from the TREC Web Track (RERANKALL); and examining neural IR models' classification accuracy between document pairs (PAIRACCURACY).

3.1 Experimental Setup

We rely on the widely-used 2009–2014 TREC Web Track ad-hoc task benchmarks[3]. The benchmarks are based on the CLUEWEB09 and CLUEWEB12 datasets as document collections. In total, there are 300 queries and more than 100k judgments (qrels). Three years (2012–14) of query-likelihood baselines[4] provided by TREC[5] serve as baseline runs in the RERANKSIMPLE benchmark. In the RERANKALL setting, the search results from runs submitted by participants from each year are also considered: there are 71 (2009), 55 (2010), 62 (2011), 48 (2012), 50 (2013), and 27 (2014) runs. ERR@20 (Chapelle et al., 2009) and nDCG@20 (Järvelin and Kekäläinen, 2002) are employed as evaluation measures, and both are computed with the script from TREC[6].

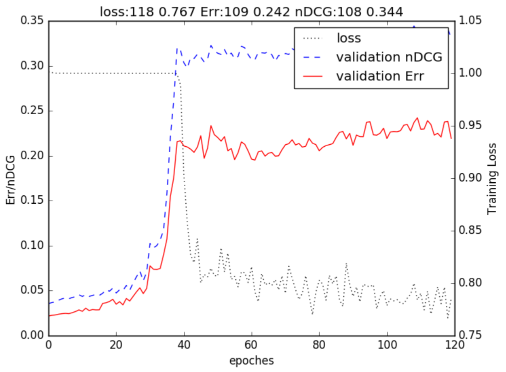


Figure 2: The training loss, ERR@20 and nDCG@20 per iteration on validation data when training on Web Track 2010–14. The x-axis denotes the iterations. The y-axis indicates the ERR@20/nDCG@20 (left) and the loss (right). The best performance appears on 109th iteration with ERR@20=0.242. The lowest training loss (0.767) occurs after 118 iterations.

Training. At each step, we perform Stochastic Gradient Descent (SGD) with a mini-batch of 32 triples. For the purpose of choosing the triples, we consider all documents that are judged with a label more relevant than *Rel*[7] as *highly relevant*, and put the remaining relevant documents into a *relevant* group. To pick each triple, we sample a relevance group with probability proportional to the number of documents in the group within the training set, and then we randomly sample a document with the chosen label to serve as the positive document d^+. If the chosen group is the highly relevant group, we randomly sample a document from the relevant group to serve as the negative document d^-. If the chosen group is the relevant group, we randomly sample a non-relevant document as d^-. This sampling procedure ensures that we differentiate between highly relevant documents (i.e., those with a relevance label of *HRel*, *Key* or *Nav*) and relevant documents (i.e., those are labeled as *Rel*). The training continues until a

[2] We also attempted to include IRGAN (Wang et al., 2017) model as a baseline, but failed to obtain reasonable results when training on TREC data.

[3] `http://trec.nist.gov/tracks.html`

[4] Terrier (Ounis et al., 2006) version without filtering spam documents

[5] `https://github.com/trec-web/trec-web-2014`

[6] http://trec.nist.gov/data/web/12/gdeval.pl

[7] Judgments from TREC include junk pages (*Junk*), non-relevance (*NRel*), relevance (*Rel*), high relevance (*HRel*), key pages (*Key*) and navigational pages (*Nav*).

given number of iterations is reached. The model is saved at every iteration. We use the model with the best ERR@20 on the validation set to make predictions. Proceeding in a round-robin manner, we report test results on one year by exploiting the respective remaining five years (250 queries) for training. From these 250 queries, we reserve 50 random queries as a held-out set for validation and hyper-parameter tuning, while the remaining 200 queries serve as the actual training set.

As mentioned, model parameters and training iterations are chosen by maximizing the ERR@20 on the validation set. The selected model is then used to make predictions on the test data. An example of this training procedure is shown in Figure 2. There are four hyper-parameters that govern the behavior of the proposed *PACRR-kwindow* and *PACRR-firstk*: the unified length of the document dimension l_d, the k-max pooling size n_s, the maximum n-gram size l_g, and the number of filters used in convolutional layers n_f. Due to limited computational resources, we determine the range of hyper-parameters to consider based on pilot experiments and domain insights. In particular, we evaluate $l_d \in [256, 384, 512, 640, 768]$, $n_s \in [1, 2, 3, 4]$, and $l_g \in [2, 3, 4]$. Due to the limited possible matching patterns given a small kernel size (e.g., $l_g = 3$), n_f is fixed to 32. For *PACRR-firstk*, we intuitively desire to retain as much information as possible from the input, and thus l_d is always set to 768.

DRMM (*DRMM*$_{LCH \times IDF}$), *DUET*, *MatchPyramid* and *K-NRM* are trained under the same settings using the hyperparameters described in their respective papers. In particular, as our focus is on the deep relevance matching model as mentioned in Section 1, we only compare against DUET's local model, denoted as *DUETL*. In addition, *K-NRM* is trained slightly different from the one described in (Xiong et al., 2017), namely, with a frozen word embedding layer. This is to guarantee its fair comparison with other models, given that most of the compared models can be enhanced by co-training the embedding layers, whereas the focus here is the strength coming from the model architecture. A fully connected middle layer with 30 neurons is added to compensate for the reduction of trainable parameters in *K-NRM*, mirroring the size of DRMM's first fully connected layer.

All models are implemented with Keras (Chollet et al., 2015) using Tensorflow as backend, and are trained on servers with multiple CPU cores. In particular, the training of *PACRR* takes 35 seconds per iteration on average, and in total at most 150 iterations are trained for each model variant.

3.2 Results

RERANKSIMPLE. We first examine the proposed model by re-ranking the search results from the *QL* baseline on Web Track 2012–14. The results are summarized in Table 1. It can be seen that *DRMM* can significantly improve *QL* on WT12 and WT14, whereas *MatchPyramid* fails on WT12 under ERR@20. While *DUETL* and *K-NRM* can consistently outperform *QL*, the two variants of *PACRR* are the only models that can achieve significant improvements at a 95% significance level on all years under both ERR@20 and nDCG@20. More remarkably, by solely re-ranking the search results from *QL*, *PACRR-firstk* can already rank within the top-3 participating systems on all three years as measured by both ERR and nDCG. The re-ranked search results from *PACRR-kwindow* also ranks within the top-5 based on nDCG@20. On average, both *PACRR-kwindow* and *PACRR-firstk* achieve 60% improvements over *QL*.

RERANKALL. In this part, we would like to further examine the performance of the proposed models in re-ranking different sets of search results. Thus, we extend our analysis to re-rank search results from all submitted runs from six years of the TREC Web Track ad-hoc task. In particular, we only consider the judged documents from TREC, which loosely correspond to top-20 documents in each run. The tested models make predictions for individual documents, which are used to re-rank the documents within each submitted run. Given that there are about 50 runs for each year, it is no longer feasible to list the scores for each re-ranked run. Instead, we summarize the results by comparing the performance of each run before and after re-ranking, and provide statistics over each year to compare the methods under consideration in Table 2. In the top portion of Table 2, we report the relative changes in metrics before and after re-ranking in terms of percentages ("average Δ measure score"). In the bottom portion, we report the percentage of systems whose results have increased after re-ranking. Note that these results assess two different aspects: the average Δ measure score in Table 2 captures the degree to which a model can improve an initial run, while

Measure	Years	PACRR-firstk	Rank	PACRR-kwindow	Rank	DUETL	Rank	DRMM	Rank	MatchPyramid	Rank	K-NRM	Rank	QL	Rank
ERR@20	wt12	0.318 (mQ)	2	0.313 (MQ)	4	0.281 (Q)	10	0.289 (Q)	10	0.227	16	0.258 (Q)	12	0.177	26
	wt13	0.166 (DKQ)	3	0.139 (Q)	14	0.147 (Q)	12	0.124	25	0.141 (q)	13	0.134 (q)	14	0.101	38
	wt14	0.221 (LMQ)	2	0.208 (Q)	3	0.179 (Q)	12	0.193 (Q)	10	0.176 (Q)	12	0.201 (Q)	8	0.131	25
nDCG@20	wt12	0.243 ($DLMQ$)	2	0.250 ($DLMQ$)	2	0.186 (Q)	11	0.197 (Q)	8	0.164 (Q)	16	0.222 (Q)	4	0.106	39
	wt13	0.295 ($DLkQ$)	3	0.279 (DQ)	4	0.248 (q)	11	0.228	20	0.258 (Q)	7	0.251 (Q)	11	0.190	36
	wt14	0.339 (LMQ)	1	0.331 (LMQ)	1	0.267 (q)	11	0.300 (Q)	6	0.278 (Q)	10	0.324 (Q)	2	0.231	23

Table 1: ERR@20 and nDCG@20 on TREC Web Track 2012–14 when re-ranking search results from *QL*. The comparisons are conducted between two variants of PACRR and DRMM (D/d), DUETL (L/l), MatchPyramid (M/m) and K-NRM (K/k). All methods are compared against the *QL* (Q/q) baseline. The upper/lower-case characters in the brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach. In addition, the relative ranks among all runs within the respective years according to ERR@20 and nDCG@20 are also reported directly after the absolute scores.

	Measures	Tested Methods	wt09	wt10	wt11	wt12	wt13	wt14
average Δ measure score over each year (%): $\frac{\text{re-rank score}-\text{original score}}{\text{original score}}$	ERR@20	*PACRR-firstk*	66% (DLK)	362% (dm)	43% ($DLMK$)	76% ($DLMK$)	37% ($DLMK$)	41% ($DLMK$)
		PACRR-kwindow	70% ($DLmK$)	393% (DlM)	10% (LMK)	83% ($DLMK$)	21% (DLM)	36% ($DLMK$)
		DUETL	80% (DMK)	316%	15% (DMK)	64% (M)	26% (DM)	19% (MK)
		DRMM	54% (LMK)	315%	11% (LMK)	61% (M)	5% (LMK)	19% (MK)
		MatchPyramid	65% (DL)	313%	2% (DLK)	48% (DLK)	29% (DLK)	14% (DLK)
		K-NRM	59% (DL)	333%	31% (DLM)	63% (M)	25% (DM)	32% (DLM)
	nDCG@20	*PACRR-firstk*	69% ($DLMK$)	304% (LM)	56% ($DLMK$)	100% ($DLMK$)	31% ($DLMK$)	31% (DLM)
		PACRR-kwindow	63% (DmK)	345% ($DLMK$)	27% ($DLMK$)	113% ($DLMK$)	23% (DLK)	30% (DLM)
		DUETL	62% (DMK)	237% (DK)	17% (DMK)	55% (DMK)	17% (DMK)	10% (DMK)
		DRMM	49% (LMK)	274% (LMk)	8% (LMK)	70% (LMK)	9% (LMK)	15% (LK)
		MatchPyramid	59% (DLk)	232% (DK)	1% (DLK)	37% (DLK)	21% (DLk)	14% (LK)
		K-NRM	52% (DLm)	288% (dLM)	36% (DLM)	85% (DLM)	19% (DLm)	30% (DLM)
% of runs that get better performance after re-ranking	ERR@20	*PACRR-firstk*	94%	95%	**97%**	92%	**87%**	**100%**
		PACRR-kwindow	**97%**	**100%**	47%	**96%**	65%	76%
		DUETL	94%	95%	61%	86%	69%	59%
		DRMM	82%	95%	47%	86%	40%	66%
		MatchPyramid	85%	93%	40%	78%	81%	59%
		K-NRM	87%	95%	89%	82%	67%	86%
	nDCG@20	*PACRR-firstk*	**94%**	**100%**	**100%**	**100%**	**92%**	**93%**
		PACRR-kwindow	93%	**100%**	84%	**100%**	81%	86%
		DUETL	86%	93%	69%	92%	79%	59%
		DRMM	86%	**100%**	50%	88%	62%	55%
		MatchPyramid	76%	93%	39%	80%	81%	69%
		K-NRM	**94%**	**100%**	97%	96%	81%	**93%**

Table 2: The average statistics when re-ranking all runs from the TREC Web Track 2009–14 based on ERR@20 and nDCG@20. The average differences of the scores for individual runs are reported in the top portion. The comparisons are conducted between two variants of PACRR and DRMM (D/d), DUETL (L/l), MatchPyramid (M/m) and K-NRM (K/k). The upper/lower-case characters in parentheses indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels, respectively, relative to the corresponding approach. The percentage of runs that show improvements in terms of a measure is summarized in the bottom portion.

the percentages of runs indicate to what extent an improvement can be achieved over runs from different systems. In other words, the former measures the strength of the models, while the latter measures the adaptability of the models. Both *PACRR* variants improve upon existing rankings by at least 10% across different years. Remarkably, in terms of nDCG@20, at least 80% of the submitted runs are improved after re-ranking by the proposed models on individual years, and on 2010–12, all submitted runs are consistently improved by *PACRR-firstk*. Moreover, both variants of *PACRR* can significantly outperform all baseline models on at least three years out of the six years in terms of average improvement. However, it is clear that none of the tested models can make consistent improvements over all submitted runs across all six years. In other words, there still exist document pairs that are predicted contradicting to the judgments from TREC. Thus, in the next part, we further investigate the performance in terms of prediction over document pairs.

Label Pairs	Volume (%)	# Queries	Tested Methods					
			PACRR-firstk	*PACRR-kwindow*	*DUETL*	*DRMM*	*MatchPyramid*	*K-NRM*
Nav-HRel	0.3%	49	45.8%	45.5%	45.2%	48.2%	47.3%	51.6%
Nav-Rel	1.1%	65	56.0% (m)	56.3% (M)	54%	57% (M)	53.2% (D)	57.4%
Nav-NRel	3.6%	67	76.1% ($DLMK$)	76.6% ($DLMK$)	67.1% (M)	71.5% (M)	64.7% (DLK)	70.8% (M)
HRel-Rel	8.4%	257	57.3%	57.0%	55.5%	55.8%	52.8%	56.1%
HRel-NRel	23.1%	262	76.7% ($DLMK$)	76.4% ($DLMK$)	68.4% (K)	70.1% (MK)	65.6% (DK)	72.5% (DLM)
Rel-NRel	63.5%	290	73.0% ($DLMK$)	72.5% ($DLMK$)	63.9% (DMK)	65.9% (LMK)	61.4% (DLK)	68.7% (DLM)
weighted average			72.4%	72.0%	64.2%	66.1%	61.6%	68.4%

Table 3: Comparison among tested methods in terms of accuracy when comparing document pairs with different labels. The "volume" column indicates the percentage of occurrences of each label combination out of the total pairs. The "# Queries" column records the number of queries that include a particular label combination. The comparisons are conducted between two variants of PACRR and DRMM (D/d), DUETL (L/l), MatchPyramid (M/m) and K-NRM (K/k). The upper/lower-case characters in parentheses indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels, respectively, relative to the corresponding approach. In the last row, the average accuracy among different kinds of label combinations is computed, weighted by their corresponding volume.

PAIRACCURACY. The ranking of documents can be decomposed into rankings of document pairs as suggested in (Radinsky and Ailon, 2011). Specifically, a model's retrieval quality can be examined by checking across a range of individual document pairs, namely, how likely a model can assign a higher score for a more relevant document. Thus, it is possible for us to compare different models over the same set of complete judgments, removing the issue of different initial runs. Moreover, although ranking is our ultimate target, a direct inspection of pairwise prediction results can indicate which kinds of document pairs a model succeeds at or fails on. We first convert the graded judgments from TREC into ranked document pairs by comparing their labels. Document pairs are created among documents that have different labels. A prediction is counted as correct if it assigns a higher score to the document from the pair that is labeled with a higher degree of relevance. The judgments from TREC contain at most six relevance levels, and we merge and unify the original levels from the six years into four grades, namely, *Nav*, *HRel*, *Rel* and *NRel*. We compute the accuracy for each pair of labels. The statistics are summarized in Table 3. The volume column lists the percentage of a given label combination out of all document pairs, and the # query column provides the number of queries for which the label combination exists. In Table 3, we observe that both *PACRR* models always perform better than all baselines on label combinations *HRel* vs. *NRel*, *Rel* vs. *NRel* and *Nav* vs. *NRel*, which in total cover 90% of all document pairs. Meanwhile, apart from *Nav-Rel*, there is no significant difference when distinguishing *Nav* from other types. *K-NRM* and *DRMM* perform better than the other two baseline models.

3.3 Discussion

Hyper-parameters. As mentioned, models are selected based on the ERR@20 over validation data. Hence, it is sufficient to use a reasonable and representative validation dataset, rather than handpicking a specific set of parameter settings. However, to gain a better understanding of the influence of different hyper-parameters, we explore *PACRR-kwindow*'s effectiveness when several hyper-parameters are varied. The results when re-ranking *QL* search results are given in Figure 3. The results are reported based on the models with the highest validation scores after fixing certain hyper-parameters. For example, the ERR@20 in the leftmost figure is obtained when fixing l_d to the values shown. The crosses in Figure 3 correspond to the models that were selected for use on the test data, based on their validation set scores. It can be seen that the selected models are not necessarily the best model on the test data, as evidenced by the differences between validation and test data results, but we consistently obtain scores within a reasonable margin. Owing to space constraints, we omit the plots for *PACRR-firstk*.

Choice between *kwindow* and *firstk* approaches. As mentioned, both *PACRR-kwindow* and *PACRR-firstk* serve to address the variable-length chal-

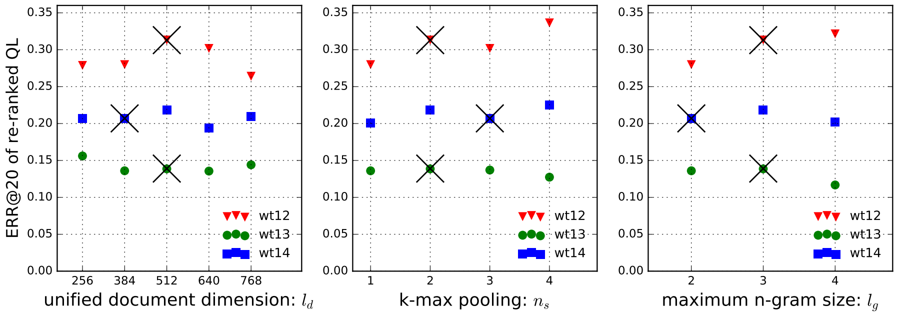


Figure 3: The ERR@20 of re-ranked *QL* with *PACRR-kwindow* when applying different hyper-parameters: l_d, n_s and l_g. The x-axis reflects the settings for hyper-parameters, and the y-axis is the ERR@20. Crosses correspond to the selected models.

lenge for documents and queries, and to make the training feasible and more efficient. In general, if both training and test documents are known to be short enough to fit in memory, then *PACRR-firstk* can be used directly. Otherwise, *PACRR-kwindow* is a reasonable choice to provide comparable results. Alternatively, one can regard this choice as another hyper-parameter, and make a selection based on held-out validation data.

Accuracy in PAIRACCURACY. Beyond the observations in Section 3.2, we further examine the methods' accuracy over binary judgments by merging the *Nav*, *HRel* and *Rel* labels. The accuracies become 73.5%, 74.1% and 67.4% for *PACRR-kwindow*, *PACRR-firstk*, and *DRMM*, respectively. Note that the manual judgments that indicate a document as relevant or non-relevant relative to a given query contain disagreements (Carterette et al., 2008; Voorhees, 2000) and errors (Alonso and Mizzaro, 2012). In particular, a 64% agreement (cf. Table 2 (b) therein) is observed over the inferred relative order among document pairs based on graded judgments from six trained judges (Carterette et al., 2008). When reproducing TREC judgments, Al-Maskari et al. (Al-Maskari et al., 2008) reported a 74% agreement (cf. Table 1 therein) with the original judgments from TREC when a group of users re-judged 56 queries on the TREC-8 document collections. Meanwhile, Alonso and Mizzaro (Alonso and Mizzaro, 2012) observed a 77% agreement relative to judgments from TREC when collecting judgments via crowdsourcing. Therefore, the more than 73% agreement achieved by both *PACRR* methods is close to the aforementioned agreement levels among different human assessors. However, when distinguishing *Nav*, *HRel*, and *Rel*, the tested models still fall significantly short of the human judges' agreement levels. These distinctions are important for a successful ranker, especially when measuring with graded metrics such as ERR@20 and nDCG@20. Hence, further research is needed for better discrimination among relevant documents with different degrees of relevance. In addition, as for the distinction between *Nav* documents and *Rel* or *HRel* documents, we argue that since *Nav* actually indicates that a document mainly satisfies a navigational intent, this makes such documents qualitatively different from *Rel* and *HRel* documents. Specifically, a *Nav* is more relevant for a user with navigational intent, whereas for other users it may in some cases be less useful than a document that directly includes highly pertinent information content. Therefore, we hypothesize that further improvements can be obtained by introducing a classifier for user intents, e.g., navigational pages, before employing neural IR models.

4 Conclusion

In this work, we have demonstrated the importance of preserving positional information for neural IR models by incorporating domain insights into the proposed *PACRR* model. In particular, *PACRR* captures term dependencies and proximity through multiple convolutional layers with different sizes. Thereafter, following two max-pooling layers, it combines salient signals over different query terms with a recurrent layer. Extensive experiments show that *PACRR* substantially outperforms four state-of-the-art neural IR models on TREC Web Track ad-hoc datasets and can dramatically improve search results when used as a re-ranking model.

References

Azzah Al-Maskari, Mark Sanderson, and Paul Clough. 2008. Relevance judgments between trec and non-trec assessors. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 683–684. ACM.

Omar Alonso and Stefano Mizzaro. 2012. Using crowdsourcing for trec relevance assessment. *Information Processing & Management*, 48(6):1053–1066.

Ben Carterette, Paul N Bennett, David Maxwell Chickering, and Susan T Dumais. 2008. Here or there: Preference Judgments for Relevance. In *Advances in Information Retrieval*, pages 16–27. Springer.

Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. 2009. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 621–630, New York, NY, USA. ACM.

François Chollet et al. 2015. Keras. `https://github.com/fchollet/keras`.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64. ACM.

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

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 2333–2338, New York, NY, USA. ACM.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. Position-aware representations for relevance matching in neural information retrieval. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 799–800. International World Wide Web Conferences Steering Committee.

Samuel Huston and W. Bruce Croft. 2014. A comparison of retrieval models using term dependencies. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 111–120. ACM.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Donald Metzler and W Bruce Croft. 2005. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479. ACM.

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

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of WWW 2017*. ACM.

Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. 2016. A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137*.

Iadh Ounis, Gianni Amati, Vassilis Plachouras, Ben He, Craig Macdonald, and Christina Lioma. 2006. Terrier: A high performance and scalable information retrieval platform. In *Proceedings of the OSIR Workshop*, pages 18–25.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2016. A study of matchpyramid models on ad-hoc retrieval. *CoRR*, abs/1606.04648.

Kira Radinsky and Nir Ailon. 2011. Ranking from pairs and triplets: Information quality, evaluation methods and query complexity. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 105–114, New York, NY, USA. ACM.

Tao Tao and ChengXiang Zhai. 2007. An exploration of proximity measures in information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 295–302. ACM.

Ellen M Voorhees. 2000. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information processing & management*, 36(5):697–716.

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. *arXiv preprint arXiv:1705.10513*.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. *arXiv preprint arXiv:1706.06613*.

Globally Normalized Reader

Jonathan Raiman and **John Miller**
Baidu Silicon Valley Artificial Intelligence Lab
{jonathanraiman,millerjohn}@baidu.com

Abstract

Rapid progress has been made towards question answering (QA) systems that can extract answers from text. Existing neural approaches make use of expensive bidirectional attention mechanisms or score all possible answer spans, limiting scalability. We propose instead to cast extractive QA as an iterative search problem: select the answer's sentence, start word, and end word. This representation reduces the space of each search step and allows computation to be conditionally allocated to promising search paths. We show that globally normalizing the decision process and back-propagating through beam search makes this representation viable and learning efficient. We empirically demonstrate the benefits of this approach using our model, Globally Normalized Reader (GNR), which achieves the second highest single model performance on the Stanford Question Answering Dataset (68.4 EM, 76.21 F1 dev) and is 24.7x faster than bi-attention-flow. We also introduce a data-augmentation method to produce semantically valid examples by aligning named entities to a knowledge base and swapping them with new entities of the same type. This method improves the performance of all models considered in this work and is of independent interest for a variety of NLP tasks.

1 Introduction

Question answering (QA) and information extraction systems have proven to be invaluable in wide variety of applications such as medical information collection on drugs and genes (Quirk and Poon, 2016), large scale health impact studies (Althoff et al., 2016), or educational material development (Koedinger et al., 2015). Recent progress in neural-network based extractive question answering models are quickly closing the gap with human performance on several benchmark QA tasks such as SQuAD (Rajpurkar et al., 2016), MS MARCO (Nguyen et al., 2016), or NewsQA (Trischler et al., 2016a). However, current approaches to extractive question answering face several limitations:

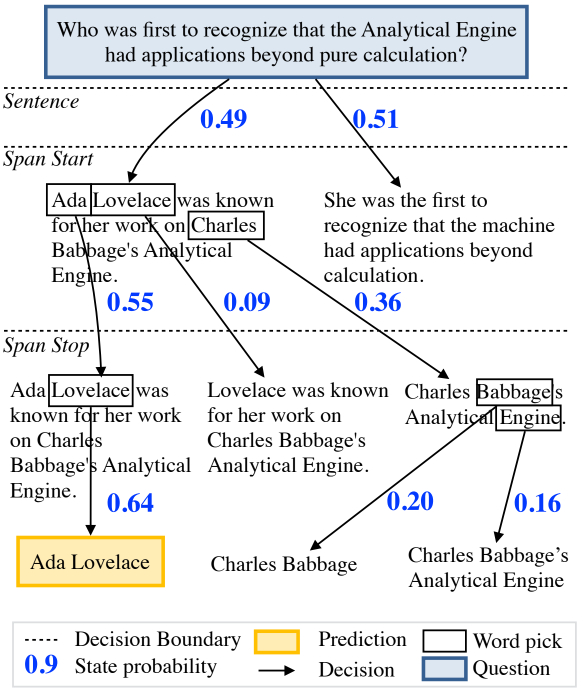

Who was first to recognize that the Analytical Engine had applications beyond pure calculation?
Sentence
0.49
0.51
Span Start
Ada Lovelace was known for her work on Charles Babbage's Analytical Engine.
She was the first to recognize that the machine had applications beyond calculation.
0.55
0.09
0.36
Span Stop
Ada Lovelace was known for her work on Charles Babbage's Analytical Engine.
Lovelace was known for her work on Charles Babbage's Analytical Engine.
Charles Babbage's Analytical Engine.
0.64
0.20
0.16
Ada Lovelace
Charles Babbage
Charles Babbage's Analytical Engine
Decision Boundary
0.9 State probability
Prediction
Decision
Word pick
Question

Figure 1: GNR answering a question. It first picks a sentence, then start word, then end word. Probabilities are global and normalized over the beam. Model initially picks the wrong sentence, but global normalization lets it recover. Final prediction's probability (0.64) exceeds sentence pick (0.49), whereas with local normalization each probability is upper bounded by the previous step.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1059–1069
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

1. Computation is allocated equally to the entire document, regardless of answer location, with no ability to ignore or focus computation on specific parts. This limits applicability to longer documents.

2. They rely extensively on expensive bi-directional attention mechanisms (Seo et al., 2016) or must rank all possible answer spans (Lee et al., 2016).

3. While data-augmentation for question answering have been proposed (Zhou et al., 2017), current approaches still do not provide training data that can *improve* the performance of existing systems.

In this paper we demonstrate a methodology for addressing these three limitations, and make the following claims:

1. Extractive Question Answering can be cast as a nested search process, where sentences provide a powerful document decomposition and an easy to learn search step. This factorization enables conditional computation to be allocated to sentences and spans likely to contain the right answer.

2. When cast as a search process, models without bi-directional attention mechanisms and without ranking all possible answer spans can achieve near state of the art results on extractive question answering.

3. Preserving narrative structure and explicitly incorporating type and question information into synthetic data generation is key to generating examples that actually improve the performance of question answering systems.

Our claims are supported by experiments on the SQuAD dataset where we show that the Globally Normalized Reader (GNR), a model that performs an iterative search process through a document (shown visually in Figure 1), and has computation conditionally allocated based on the search process, achieves near state of the art Exact Match (EM) and F1 scores without resorting to more expensive attention or ranking of all possible spans. Furthermore, we demonstrate that Type Swaps, a type-aware data augmentation strategy that aligns named entities with a knowledge base and swaps them out for new entities that share the same type, improves the performance of all models on extractive question answering.

We structure the paper as follows: in Section 2 we introduce the task and our model. Section 3 describes our data-augmentation strategy. Section 4 introduces our experiments and results. In Section 5 we discuss our findings. In Section 6 we relate our work to existing approaches. Conclusions and directions for future work are given in Section 7.

2 Model

Given a document d and a question q, we pose extractive question answering as a search problem. First, we select the sentence, the first word of the span, and finally the last word of the span. A example of the output of the model is shown in Figure 1, and the network architecture is depicted in Figure 2.

More formally, let $d_1, \ldots, d_n$ denote each sentence in the document, and for each sentence d_i, let $d_{i,1}, \ldots, d_{i,m_i}$ denote the word vectors corresponding to the words in the sentence. Similarly, let $q_1, \ldots, q_\ell$ denote the word vectors corresponding to words in the question. An answer is a tuple $a = (i^*, j^*, k^*)$ indicating the correct sentence i^*, start word in the sentence j^* and end word in the sentence k^*. Let $\mathcal{A}(d)$ denote the set of valid answer tuples for document d. We now describe each stage of the model in turn.

2.1 Question Encoding

Each question is encoded by running a stack of bidirectional LSTM (Bi-LSTM) over each word in the question, producing hidden states $(h_1^{\text{fwd}}, h_1^{\text{bwd}}), \ldots, (h_\ell^{\text{fwd}}, h_\ell^{\text{bwd}})$ (Graves and Schmidhuber, 2005). Following Lee et al. (2016), these hidden states are used to compute a *passage-independent question embedding*, q^{indep}. Formally,

$$s_j = w_q^\top \text{MLP}([h_j^{\text{bwd}}; h_j^{\text{fwd}}]) \tag{1}$$

$$\alpha_j = \frac{\exp(s_j)}{\sum_{j'=1}^{\ell} \exp(s_{j'})} \tag{2}$$

$$q^{\text{indep}} = \sum_{j=1}^{\ell} \alpha_j [h_j^{\text{bwd}}; h_j^{\text{fwd}}], \tag{3}$$

where w_q is a trainable embedding vector, and MLP is a two-layer neural network with a Relu non-linearity. The question is represented by concatenating the final hidden states of the forward and backward LSTMs and the passage-independent embedding, $q = [h_1^{\text{bwd}}; h_\ell^{\text{fwd}}; q^{\text{indep}}]$.

2.2 Question-Aware Document Encoding

Conditioned on the question vector, we compute a representation of each document word that is sensitive to both the surrounding context and the question. Specifically, each word in the document is represented as the concatenation of its word vector $d_{i,j}$, the question vector q, boolean features indicating if a word appears in the question or is repeated, and a *question-aligned embedding* from Lee et al. (2016). The question-aligned embedding $q_{i,j}^{\text{align}}$ is given by

$$s_{i,j,k} = \text{MLP}(d_{i,j})^\top \text{MLP}(q_k) \tag{4}$$

$$\alpha_{i,j,k} = \frac{\exp(s_{i,j,k})}{\sum_{k'=1}^{\ell} \exp(s_{i,j,k'})} \tag{5}$$

$$q_{i,j}^{\text{align}} = \sum_{k=1}^{\ell} \alpha_{i,j,k} q_k. \tag{6}$$

The document is encoded by a separate stack of Bi-LSTMs, producing a sequence of hidden states $(h_{1,1}^{\text{fwd}}, h_{1,1}^{\text{bwd}}), \ldots, (h_{n,m_n}^{\text{fwd}}, h_{n,m_n}^{\text{bwd}})$. The search procedure then operates on these hidden states.

2.3 Answer Selection

Sentence selection. The first phase of our search process picks the sentence that contains the answer span. Each sentence d_i is represented by the hidden state of the first and last word in the sentence for the backward and forward LSTM respectively, $[h_{i,1}^{\text{bwd}}; h_{i,m_i}^{\text{fwd}}]$, and is scored by passing this representation through a fully connected layer that outputs the unnormalized sentence score for sentence d_i, denoted $\phi_{\text{sent}}(d_i)$.

Span start selection. After selecting a sentence d_i, we pick the start of the answer span within the sentence. Each potential start word $d_{i,j}$ is represented as its corresponding document encoding $[h_{i,j}^{\text{fwd}}; h_{i,j}^{\text{bwd}}]$, and is scored by passing this encoding through a fully connected layer that outputs the unnormalized start word score for word j in sentence i, denoted $\phi_{\text{sw}}(d_{i,j})$.

Span end selection. Conditioned on sentence d_i and starting word $d_{i,j}$, we select the end word from the remaining words in the sentence $d_{i,j}, \ldots, d_{i,m_i}$. To do this, we run a Bi-LSTM over the remaining document hidden states $(h_{i,j}^{\text{fwd}}, h_{i,j}^{\text{bwd}}), \ldots, (h_{i,m_i}^{\text{fwd}}, h_{i,m_i}^{\text{bwd}})$ to produce representations $(\tilde{h}_{i,j}^{\text{fwd}}, \tilde{h}_{i,j}^{\text{bwd}}), \ldots, (\tilde{h}_{i,m_i}^{\text{fwd}}, \tilde{h}_{i,m_i}^{\text{bwd}})$. Each end word $d_{i,k}$ is then scored by passing $[\tilde{h}_{i,k}^{\text{fwd}}; \tilde{h}_{i,k}^{\text{bwd}}]$ through a fully connected layer that outputs the unnormalized end word score for word k in sentence i, with start word j, denoted $\phi_{\text{ew}}(d_{i,j:k})$.

2.4 Global Normalization

The scores for each stage of our model can be normalized at the local or global level. Previous work demonstrated that locally-normalized models have a weak ability to correct mistakes made in previous decisions, while globally normalized models are strictly more expressive than locally normalized models (Andor et al., 2016; Zhou et al., 2015; Collins and Roark, 2004).

In a locally normalized model each decision is made conditional on the previous decision. The probability of some answer $a = (i, j, k)$ is decomposed as

$$\begin{aligned}\mathbb{P}(a|d,q) =& \mathbb{P}_{\text{sent}}(i|d,q) \cdot \mathbb{P}_{\text{sw}}(j|i,d,q) \cdot \\ & \mathbb{P}_{\text{ew}}(k|j,i,d,q).\end{aligned} \tag{7}$$

Each sub-decision is locally normalized by applying a softmax to the relevant selection scores:

$$\mathbb{P}_{\text{sent}}(i|d,q) = \frac{\exp(\phi_{\text{sent}}(d_i))}{\sum_{x=1}^{n} \exp(\phi_{\text{sent}}(d_x))}, \tag{8}$$

$$\mathbb{P}_{\text{sw}}(j|i,d,q) = \frac{\exp(\phi_{\text{sw}}(d_{i,j}))}{\sum_{x=1}^{m_i} \exp(\phi_{\text{sw}}(d_{i,x}))}, \tag{9}$$

$$\mathbb{P}_{\text{ew}}(k|j,i,d,q) = \frac{\exp(\phi_{\text{ew}}(d_{i,j:k}))}{\sum_{x=j}^{m_i} \exp(\phi_{\text{ew}}(d_{i,j:x}))}. \tag{10}$$

To allow our model to recover from incorrect sentence or start word selections, we instead globally normalize the scores from each stage of our procedure. In a globally normalized model, we define

$$\text{score}(a,d,q) = \phi_{\text{sent}}(d_i) + \phi_{\text{sw}}(d_{i,j}) + \phi_{\text{ew}}(d_{i,j:k}). \tag{11}$$

Then, we model

$$\mathbb{P}(a \mid d,q) = \frac{\exp(\text{score}(a,d,q))}{Z}, \tag{12}$$

where Z is the partition function

$$Z = \sum_{a' \in \mathcal{A}(d)} \exp(\text{score}(a',d,q)). \tag{13}$$

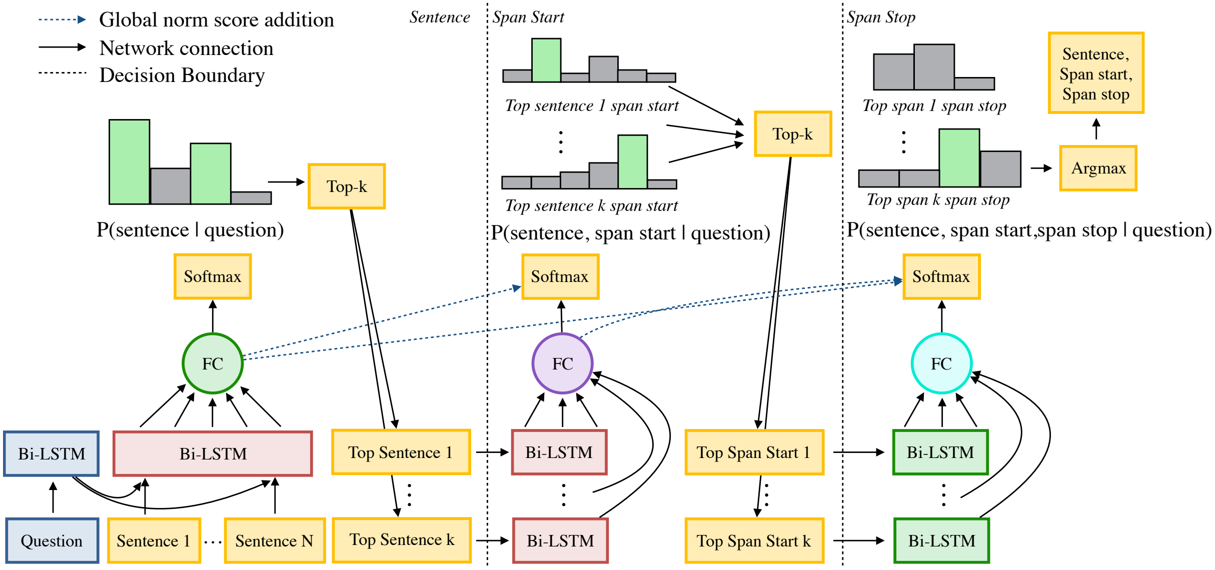


Figure 2: Globally Normalized Reader's search process. Same color Bi-LSTMs share weights.

In contrast to locally-normalized models, the model is normalized over all possible search paths instead of normalizing each step of search procedure. At inference time, the problem is to find

$$\arg\max_{a \in \mathcal{A}(d)} \mathbb{P}(a \mid d, q), \tag{14}$$

which can be approximately computed using beam search.

2.5 Objective and Training

We minimize the negative log-likelihood on the training set using stochastic gradient descent. For a single example (a, d, q), the negative log-likelihood

$$-\text{score}(a, d, q) + \log Z \tag{15}$$

requires an expensive summation to compute $\log Z$. Instead, to ensure learning is efficient, we use beam search during training and early updates (Andor et al., 2016; Zhou et al., 2015; Collins and Roark, 2004). Concretely, we approximate Z by summing only over candidates on the final beam $\mathcal{B}$:

$$Z \approx \sum_{a' \in \mathcal{B}} \exp(\text{score}(a', d, q)). \tag{16}$$

At training time, if the gold sequence falls off the beam at step t during decoding, a stochastic gradient step is performed on the partial objective computed through step t and normalized over the beam at step t.

2.6 Implementation

Our best performing model uses a stack of 3 Bi-LSTMs for the question and document encodings, and a single Bi-LSTM for the end of span prediction. The hidden dimension of all recurrent layers is 200.

We use the 300 dimensional 8.4B token Common Crawl GloVe vectors (Pennington et al., 2014). Words missing from the Common Crawl vocabulary are set to zero. In our experiments, all architectures considered have sufficient capacity to overfit the training set. We regularize the models by fixing the word embeddings throughout training, dropping out the inputs of the Bi-LSTMs with probability 0.3 and the inputs to the fully-connected layers with probability 0.4 (Srivastava et al., 2014), and adding gaussian noise to the recurrent weights with $\sigma = 10^{-6}$. Our models are trained using Adam with a learning rate of 0.0005, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and a batch size of 32 (Kingma and Ba, 2014).

All our experiments are implemented in Tensorflow (Abadi et al., 2016), and we tokenize using Ciseau (Raiman, 2017). Despite performing beam-search during training, our model trains to convergence in under 4 hours through the use of efficient LSTM primitives in CuDNN (Chetlur et al., 2014) and batching our computation over examples and search beams. We release our code and augmented dataset.[1]

[1] https://github.com/baidu-research/GloballyNormalizedReader

Our implementation of the GNR is 24.7 times faster at inference time than the official Bi-Directional Attention Flow implementation[2]. Specifically, on a machine running Ubuntu 14 with 40 Intel Xeon 2.6Ghz processors, 386GB of RAM, and a 12GB TitanX-Maxwell GPU, the GNR with beam size 32 and batch size 32 requires 51.58 ± 0.266 seconds (mean ± std)[3] to process the SQUAD validation set. By contrast, the Bi-Directional Attention Flow model with batch size 32 requires 1260.23 ± 17.26 seconds. We attribute this speedup to avoiding expensive bi-directional attention mechanisms and making computation conditional on the search beams.

3 Type Swaps

Question: Who said in $\underset{\text{December 2012}}{\underline{\text{April 25, 2011}}}$ that the fight would change from military to law enforcement?
Answer: $\underset{\text{Jeh Johnson}}{\underline{\text{Sheryl Sandberg}}}$
Document (snippet): …Basic objectives of the $\underset{\text{Bush administration}}{\underline{\text{Cabinet of Japan}}}$ "war on terror", such as targeting al Qaeda and building international counterterrorism alliances, remain in place. In $\underset{\text{December 2012}}{\underline{\text{April 25, 2011}}}$, $\underset{\text{Jeh Johnson}}{\underline{\text{Sheryl Sandberg}}}$, the General Counsel of the $\underset{\text{Department of Defense}}{\underline{\text{ministry of education}}}$, stated that the military fight will be replaced by a law enforcement operation when speaking at $\underset{\text{Oxford University}}{\underline{\text{Ain Shams University}}}$···

Figure 3: Type Swaps example. Replacements underlined with originals underneath.

In extractive question answering, the set of possible answer spans can be pruned by only keeping answers whose nature (person, object, place, date, etc.) agrees with the question type (Who, What, Where, When, etc.). While this heuristic helps human readers filter out irrelevant parts of a document when searching for information, no explicit supervision of this kind is present in the dataset. Despite this absence, the distribution question representations learned by our models appear to utilize this heuristic. The final hidden state of the question-encoding LSTMs naturally cluster based on question type (Table 1).

In other words, the task induces a question encoding that superficially respects type information. This property is a double-edged sword: it allows the model to easily weed out answers that are inapplicable, but also leads it astray by selecting a text span that shares the answer's type but has the wrong underlying entity. A similar observation was made in the error analysis of (Weissenborn et al., 2017). We propose Type Swaps, an augmentation strategy that leverages this emergent behavior in order to improve the model's ability to prune wrong answers, and make it more robust to surface form variation. This strategy has three steps:

1. Locate named entities in document and question.
2. Collect surface variation for each entity type:

   human → {Ada Lovelace, Daniel Kahnemann,...},

   country → {USA, France, ...}, ...
3. Generate new document-question-answer examples by swapping each named entity in an original triplet with a surface variant that shares the same type from the collection.

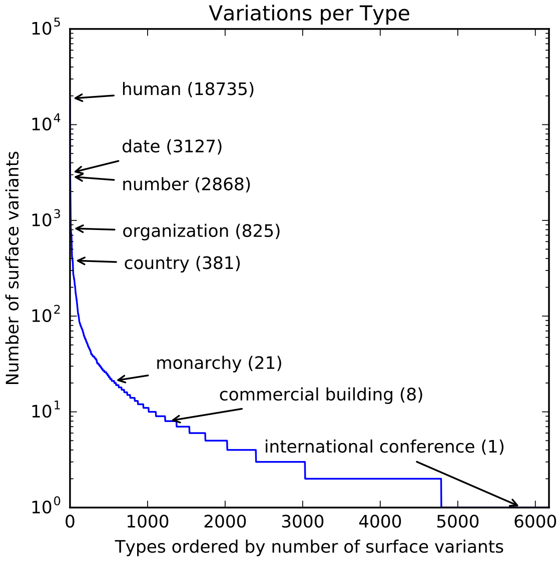


Figure 4: The majority of the surface variations occur for people, numbers, dates, and organizations.

Assigning types to named entities in natural language is an open problem, nonetheless when faced

[2] https://github.com/allenai/bi-att-flow
[3] All numbers are averaged over 5 runs.

Table 1: Top bigrams in K-means ($K = 7$) clusters of question after Bi-LSTM. We observe emergent clustering according to question type: e.g. *Where*→ Cluster 7, *Who*→ Cluster 3. "What" granularity only observable with more clusters.

Cluster	1	2	3	4	5	6	7
Size	84789	42187	53061	130022	27549	16894	28377
Bigram	Bigram Occurences						
what is	3339	520	87	3736	20	8	138
what did	2463	3	3	112	1	0	1
how many	2	5095	1	1	0	0	0
how much	7	1102	0	12	0	0	0
who was	2	0	1934	0	0	0	1
who did	2	0	683	2	0	0	0
what was	2177	508	105	2034	71	31	92
when did	0	0	0	1	2772	0	0
when was	0	0	1	1	1876	0	0
what year	0	0	0	1	13	2690	0
in what	52	3	9	727	110	1827	518
where did	0	0	0	13	1	0	955
where is	0	1	0	11	0	0	665

with documents where we can safely assume that the majority of the entities will be contained in a large knowledge base (KB) such as Wikidata Vrandečić and Krötzsch (2014) we find that simple string matching techniques are sufficiently accurate. Specifically, we use a part of speech tagger (Honnibal, 2017) to extract nominal groups in the training data and string-match them with entities in Wikidata. Using this technique, we are able to extract 47,598 entities in SQuAD that fall under 6,380 Wikidata `instance of`[4] types. Additionally we assign "number types" (e.g. *year*, *day of the week*, *distance*, etc.) to nominal groups that contain dates, numbers, or quantities[5]. These extraction steps produce 84,632 unique surface variants (on average 16.93 per type) with the majority of the variation found in humans, numbers or organizations as visible in Figure 4.

With this method, we can generate $2.92 \cdot 10^{369}$ unique documents (average of $3.36 \cdot 10^{364}$ new documents for each original document). To ensure there is sufficient variation in the generated documents, we sample from this set and only keep variations where the question or answer is mutated. At each training epoch, we train on T Type Swap examples and the full original training data. An example output of the method is shown in Figure 3.

4 Results

We evaluate our model on the 100,000 example SQuAD dataset (Rajpurkar et al., 2016) and perform several ablations to evaluate the relative importance of the proposed methods.

4.1 Learning to Search

In our first experiment, we aim to quantify the importance of global normalization on the learning and search process. We use $T = 10^4$ Type Swap samples and vary beam width B between 1 and 32 for a locally and globally normalized models and summarize the Exact-Match and F1 score of the model's predicted answer and ground truth computed using the evaluation scripts from (Rajpurkar et al., 2016) (Table 3). We additionally report another metric, the *Sentence* score, which is a measure for how often the predicted answer came from the correct sentence. This metric provides a measure for where mistakes are made during prediction.

4.2 Type Swaps

In our second experiment, we evaluate the impact of the amount of augmented data on the performance of our model. In this experiment, we use

[4] `https://www.wikidata.org/wiki/Property:P31`

[5] In our experiments we found that not including numerical variation in the generated examples led to an imbalanced dataset and lower final performance.

Table 2: Model comparison

Model	EM	F1
Human (Rajpurkar et al., 2016)	80.3	90.5
Single model		
Sliding Window (Rajpurkar et al., 2016)	13.3	20.2
Match-LSTM (Wang and Jiang, 2016)	64.1	73.9
DCN (Xiong et al., 2016)	65.4	75.6
Rasor (Lee et al., 2016)	66.4	74.9
Bi-Attention Flow (Seo et al., 2016)	67.7	77.3
R-Net(Wang et al., 2017)	**72.3**	**80.6**
Globally Normalized Reader w/o Type Swaps (Ours)	66.6	75.0
Globally Normalized Reader (Ours)	68.4	76.21

Table 3: Impact of Beam Width B

Model	B	EM	F1	Sentence
Local, $T = 10^4$	1	65.7	74.8	**89.0**
	2	66.6	75.0	88.3
	10	66.7	75.0	88.6
	32	66.3	74.6	88.0
	64	66.6	75.0	88.8
Global, $T = 10^4$	1	58.8	68.4	84.5
	2	64.3	73.0	86.8
	10	66.6	75.2	88.1
	32	**68.4**	**76.21**	88.4
	64	67.0	75.6	88.4

Table 4: Impact of Augmentation Sample Size T.

Model	T	EM	F1	Sentence
Local	0	65.8	74.0	88.0
Local	10^3	66.3	74.6	88.9
Local	10^4	66.7	74.9	**89.0**
Local	$5 \cdot 10^4$	66.7	75.0	**89.0**
Local	10^5	66.2	74.5	88.6
Global	0	66.6	75.0	88.2
Global	10^3	66.9	75.0	88.1
Global	10^4	**68.4**	**76.21**	88.4
Global	$5 \cdot 10^4$	66.8	75.3	88.3
Global	10^5	66.1	74.3	86.9

Table 5: Impact of Type Swaps on the DCN+

T	Train F1	Dev F1
0	81.3	78.1
$5 \cdot 10^4$	72.5	**78.2**

the best beam sizes for each model ($B = 10$ for local and $B = 32$ for global) and vary the augmentation from $T = 0$ (no augmentation) to $T = 5 \cdot 10^4$. The results of this experiment are summarized in (Table 4).

We observe that both models improve in performance with $T > 0$ and performance degrades past $T = 10^4$. Moreover, data augmentation and global normalization are complementary. Combined, we obtain 1.6 EM and 2.0 F1 improvement over the locally normalized baseline.

We also verify that the effects of Type Swaps are not limited to our specific model by observing the impact of augmented data on the DCN+ (Xiong et al., 2016)[6]. We find that it strongly reduces generalization error, and helps improve F1, with potential further improvements coming by reducing other forms of regularization (Table 5).

5 Discussion

In this section we will discuss the results presented in Section 4, and explain how they relate to our main claims.

5.1 Extractive Question Answering as a Search Problem

Sentences provide a natural and powerful document decomposition for search that can be easily learnt as a search step: for all the models and configurations considered, the *Sentence* score was

[6] The DCN+ is the DCN with additional hyperparameter tuning by the same authors as submitted on the SQuAD leaderboard `https://rajpurkar.github.io/SQuAD-explorer/`.

above 88% correct (Table 3)[7]. Thus, sentence selection is the easy part of the problem, and the model can allocate more computation (such as the end-word selection Bi-LSTM) to spans likely to contain the answer. This approach avoids wasteful work on unpromising spans and is important for further scaling these methods to long documents.

5.2 Global Normalization

The Globally Normalized Reader outperforms previous approaches and achieves the second highest EM behind (Wang et al., 2017), without using bi-directional attention and only scoring spans in its final beam. Increasing the beam width improves the results for both locally and globally normalized models (Table 3), suggesting search errors account for a significant portion of the performance difference between models. Models such as Lee et al. (2016) and Wang and Jiang (2016) overcome this difficulty by ranking all possible spans and thus never skipping a possible answer. Even with large beam sizes, the locally normalized model underperforms these approaches. However, by increasing model flexibility and performing search during training, the globally normalized model is able to recover from search errors and achieve much of the benefits of scoring all possible spans.

5.3 Type-Aware Data Augmentation

Type Swaps, our data augmentation strategy, offers a way to incorporate the nature of the question and the types of named entities in the answers into the learning process of our model and reduce sensitivity to surface variation. Existing neural-network approaches to extractive QA have so far ignored this information. Augmenting the dataset with additional type-sensitive synthetic examples improves performance by providing better coverage of different answer types. Growing the number of augmented samples used improves the performance of all models under study (Table 4-5). With $T \in [10^4, 5 \cdot 10^4]$, (EM, F1) improve from $(65.8 \rightarrow 66.7, 74.0 \rightarrow 75.0)$ for locally normalized models, and $(66.6 \rightarrow 68.4, 75.0 \rightarrow 76.21)$ for globally normalized models.

Past a certain amount of augmentation, we observe performance degradation. This suggests that despite efforts to closely mimic the original training set, there is a train-test mismatch or excess duplication in the generated examples.

Our experiments are conducted on two vastly different architectures and thus these benefits are expected to carry over to different models (Weissenborn et al., 2017; Seo et al., 2016; Wang et al., 2017), and perhaps more broadly in other natural language tasks that contain named entities and have limited supervised data.

6 Related Work

Our work is closely related to existing approaches in learning to search, extractive question answering, and data augmentation for NLP tasks.

Learning to Search. Several approaches to learning to search have been proposed for various NLP tasks and conditional computation. Most recently, Andor et al. (2016) and Zhou et al. (2015) demonstrated the effectiveness of globally normalized networks and training with beam search for part of speech tagging and transition-based dependency parsing, while Wiseman and Rush (2016) showed that these techniques could also be applied to sequence-to-sequence models in several application areas including machine translation. These works focus on parsing and sequence prediction tasks and have a fixed computation regardless of the search path, while we show that the same techniques can also be straightforwardly applied to question answering and extended to allow for conditional computation based on the search path.

Learning to search has also been used in context of modular neural networks with conditional computation in the work of Andreas et al. (2016) for image captioning. In their work reinforcement learning was used to learn how to turn on and off computation, while we find that conditional computation can be easily learnt with maximum likelihood and the help of early updates (Andor et al., 2016; Zhou et al., 2015; Collins and Roark, 2004) to guide the training process.

Our framework for conditional computation whereby the search space is pruned by a sequence of increasingly complex models is broadly reminiscent of the structured prediction cascades of (Weiss and Taskar, 2010). Trischler et al. (2016b)

[7] The objective function difference explains the lower performance of globally versus locally normalized models on the *Sentence* score: local models must always assign the highest probability to the correct sentence, while global models only ensure the correct span has the highest probability. Thus global models do not need to enforce a high margin between the correct answer's sentence score and others and are more likely to keep alternate sentences around.

also explored this approach in the context of question answering.

Extractive Question Answering. Since the introduction of the SQuAD dataset, numerous systems have achieved strong results. Seo et al. (2016); Wang et al. (2017) and Xiong et al. (2016) make use of a bi-directional attention mechanisms, whereas the GNR is more lightweight and achieves similar results without this type of attention mechanism. The document representation used by the GNR is very similar to Lee et al. (2016). However, both Lee et al. (2016) and Wang and Jiang (2016) must score all $O(N^2)$ possible answer spans, making training and inference expensive. The GNR avoids this complexity by learning to search during training and outperforms both systems while scoring only $O(B)$ spans. Weissenborn et al. (2017) is a locally normalized model that first predicts start and then end words of each span. Our experiments lead us to believe that further factorizing the problem and using global normalization along with our data augmentation would yield corresponding improvements.

Data augmentation. Several works use data augmentation to control the generalization error of deep learning models. Zhang and LeCun (2015) use a thesaurus to generate new training examples based on synonyms. Vijayaraghavan et al. (2016) employs a similar method, but uses Word2vec and cosine similarity to find similar words. Jia and Liang (2016) use a high-precision synchronous context-free grammar to generate new semantic parsing examples. Our data augmentation technique, Type Swaps, is unique in that it leverages an external knowledge-base to provide new examples that have more variation and finer-grained changes than methods that use only a thesaurus or Word2Vec, while also keeping the narrative and grammatical structure intact.

More recently Zhou et al. (2017) proposed a sequence-to-sequence model to generate diverse and realistic training question-answer pairs on SQuAD. Similar to their approach, our technique makes use of existing examples to produce new examples that are fluent, however we also are able to explicitly incorporate entity type information into the generation process and use the generated data to improve the performance of question answering models.

7 Conclusions and Future Work

In this work, we provide a methodology that overcomes several limitations of existing approaches to extractive question answering. In particular, our proposed model, the Globally Normalized Reader, reduces the computational complexity of previous models by casting the question answering as search and allocating more computation to promising answer spans. Empirically, we find that this approach, combined with global normalization and beam search during training, leads to near state of the art results. Furthermore, we find that a type-aware data augmentation strategy improves the performance of all models under study on the SQuAD dataset. The method is general, only requiring that the training data contains named entities from a large KB. We expect it to be applicable to other NLP tasks that would benefit from more training data.

As future work we plan to apply the GNR to other question answering datasets such as MS MARCO (Nguyen et al., 2016) or NewsQA (Trischler et al., 2016a), as well as investigate the applicability and benefits of Type Swaps to other tasks like named entity recognition, entity linking, machine translation, and summarization. Finally, we believe there a broad range of structured prediction problems (code generation, generative models for images, audio, or videos) where the size of original search space makes current techniques intractable, but if cast as learning-to-search problems with conditional computation, might be within reach.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback. In addition, we thank Adam Coates, Carl Case, Andrew Gibiansky, and Szymon Sidor for thoughtful comments and fruitful discussion. We also thank James Bradbury and Bryan McCann for running Type Swap experiments on the DCN+.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Tim Althoff, Ryen W White, and Eric Horvitz. 2016. Influence of pokémon go on physical activity: Study and implications. *Journal of Medical Internet Research*, 18(12).

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. *arXiv preprint arXiv:1601.01705*.

Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. 2014. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111. Association for Computational Linguistics.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Matthew Honnibal. 2017. Spacy. https://github.com/explosion/spaCy.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kenneth R Koedinger, Sidney D'Mello, Elizabeth A McLaughlin, Zachary A Pardos, and Carolyn P Rosé. 2015. Data mining and education. *Wiley Interdisciplinary Reviews: Cognitive Science*, 6(4):333–353.

Kenton Lee, Tom Kwiatkowski, Ankur Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Chris Quirk and Hoifung Poon. 2016. Distant supervision for relation extraction beyond the sentence boundary. *arXiv preprint arXiv:1609.04873*.

Jonathan Raiman. 2017. Ciseau. https://github.com/jonathanraiman/ciseau.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016a. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, Phillip Bachman, and Kaheer Suleman. 2016b. A parallel-hierarchical model for machine comprehension on sparse data. *arXiv preprint arXiv:1603.08884*.

Prashanth Vijayaraghavan, Ivan Sysoev, Soroush Vosoughi, and Deb Roy. 2016. Deepstance at semeval-2016 task 6: Detecting stance in tweets using character and word-level cnns. *arXiv preprint arXiv:1606.05694*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

David Weiss and Benjamin Taskar. 2010. Structured prediction cascades. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 916–923.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Fastqa: A simple and efficient neural architecture for question answering. *arXiv preprint arXiv:1703.04816*.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv preprint arXiv:1611.01604*.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *ACL (1)*, pages 1213–1222.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. *arXiv preprint arXiv:1704.01792*.

Speech segmentation with a neural encoder model of working memory

Micha Elsner and **Cory Shain**
melsner0@gmail.com and shain.3@osu.edu
Department of Linguistics
The Ohio State University, Columbus, OH 43210

Abstract

We present the first unsupervised LSTM speech segmenter as a cognitive model of the acquisition of words from unsegmented input. Cognitive biases toward phonological and syntactic predictability in speech are rooted in the limitations of human memory (Baddeley et al., 1998); compressed representations are easier to acquire and retain in memory. To model the biases introduced by these memory limitations, our system uses an LSTM-based encoder-decoder with a small number of hidden units, then searches for a segmentation that minimizes autoencoding loss. Linguistically meaningful segments (e.g. words) should share regular patterns of features that facilitate decoder performance in comparison to random segmentations, and we show that our learner discovers these patterns when trained on either phoneme sequences or raw acoustics. To our knowledge, ours is the first fully unsupervised system to be able to segment both symbolic and acoustic representations of speech.

1 Introduction

This paper describes a new cognitive model of the acquisition of word-like units from unsegmented input. The model is intended to describe the process by which pre-linguistic infants learn their earliest words, a stage they pass through during the first year of life (Jusczyk and Aslin, 1995; Bergelson and Swingley, 2012). Our model is based on the standard memory model of Baddeley and Hitch (1974) in which the listener encodes lexical items into phonological working memory, but represents the entire sentence as a higher-level syntactic structure without phonological detail. Our model implements this architecture using encoder-decoder LSTMs with limited memory capacity, then searches for word segmentations which make it easy to remember the sentence.[1]

Word learning has been extensively studied in previous research, both with transcribed symbolic input and acoustics. Why attempt yet another approach? Our model has three main advantages. First, as a cognitive model, it relates the kinds of learning biases used in previous work to the wider literature on working memory. Second, its sequence-to-sequence neural architecture allows it to handle either one-hot symbolic input or dense vectors of acoustic features. In contrast, existing models are typically designed for "clean" symbolic input, then retrofitted with additional mechanisms to cope with acoustics. Finally, neural networks have been impressively successful in supervised language processing domains, yet are still underused in unsupervised learning. Even systems which do use neural nets to model lexical acquisition generally require an auxiliary model for clustering the embeddings, which can make their learning objectives difficult to understand. Our system uses the well-understood autoencoder objective to perform the segmentation task without requiring auxiliary clustering, and thus suggests a new direction for neural unsupervised learning.

In an experiment conducted on the widely used Brent corpus (Brent, 1999), our system achieves performance close to that of Fleck (2008), although subsequent systems outperform ours by a wider margin. We show that memory limitations do indeed drive the performance of the system, with smaller LSTM hidden states outperforming larger ones in the development set.

In a follow-up experiment designed to ex-

[1] The system is available from https://github.com/melsner/neural-segmentation.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1070–1080
Copenhagen, Denmark, September 7–11, 2017.

plore the flexibility of our model, we deploy the segmenter on acoustic input: the English portion of the Zerospeech 2015 challenge (Versteegh et al., 2015). Our model outperforms the winning model from that challenge (Räsänen et al., 2015), although we underperform more recent unsupervised acoustic segmentation systems (Kamper et al., 2016; Räsänen et al., under review).

To our knowledge, our system is the first unsupervised LSTM speech segmenter, as well as the first unsupervised speech segmenter to succeed on both symbolic and acoustic representations of speech. Our results are of note for several reasons. First, they provide modeling support for the claim that memory limitations encourage lexical acquisition. Second, they show that a general strategy of searching for maximally compressible representations can realistically guide lexical acquisition without explicit reference to perceptual biases (c.f. e.g. Räsänen et al., 2015), regardless of input representation. And third, they demonstrate the benefits of our adaptation of neural sequence modeling to unsupervised learning.

2 Motivations

We begin with a short overview of previous approaches to the word learning problem, then explain each of our main contributions in detail. Many cognitive models of the word learning problem draw on Brent (1999), which used a simple unigram model of the lexicon to discover repeated patterns in phonemically transcribed input. Brent's model laid the groundwork for later generative models with more sophisticated prior distributions over word frequencies, co-occurrence statistics and phonological shapes (Johnson and Goldwater, 2009, among others). Other modeling architectures for segmentation have focused on detecting phonological boundaries between words using transitional probabilities (Christiansen et al., 1998, among others) or inducing words procedurally by "subtracting" known word forms from utterances (Lignos, 2011).

All these modeling architectures are designed to work with phonemically transcribed input, and require some degree of retrofitting to work with more realistic inputs. In the Bayesian framework, this typically takes the form of a transducer which probabilistically transforms "underlying" lexical items to "surface" acoustics (Lee et al., 2015) or discrete symbols (Elsner et al., 2013); the same framework is used for morphological segmentation in Cotterell et al. (2015). For transition-based models, the input must be transformed into discrete symbols from which segment-to-segment probabilities can be extracted; this transformation requires an externally trained preprocessor (a phone recognizer). Transition-based models are fairly robust to variation in the symbols (Rytting, 2007; Rytting et al., 2010; Daland and Pierrehumbert, 2011; Fleck, 2008) and can be relatively successful in this framework. Extensions using neural nets (Christiansen et al., 1998; Rytting et al., 2010) are discussed in more detail below (subsec. 2.3). Lignos (2011) requires the most complex preprocessing of the input (segmentation into syllables, with marked lexical stresses); adapting it to noisy input is an open problem.

2.1 Working memory and learning biases

Cognitive models of word segmentation rely on two kinds of learning biases to structure their inferred lexicons: predictability within words (often expressed as a prior over phonological forms), and Zipfian unigram and bigram frequencies of words (a prior over word distributions). These biases control the entropy of utterances, making it easy for adult listeners to remember what they hear and reconstruct any missing parts from context (Piantadosi et al., 2012). The biases correspond to different components in a standard model of working memory (Baddeley, 2007; Baddeley and Hitch, 1974). In this model, listeners can store the last few items they heard in a *phonological loop*, from which words are transferred into *episodic memory* which represents them at a syntactic/semantic level.

Baddeley et al. (1998) claim that the phonological loop functions in word *learning* as well as processing by proficient listeners, aiding in the acquisition of unfamiliar words. They summarize a number of studies showing that the vocabulary size of typically developing infants correlates with their ability to remember a sequence of phonologically plausible non-words, a test of phonological loop capacity. Children with Specific Language Impairment, meanwhile, remember nonwords poorly, a deficit which may contribute to their atypically small vocabularies. Baddeley et al. (1998) argue that the ability to remember an unfamiliar phonological form in the short term is essential if it is to be transferred to long-term memory as a datapoint for lexical learning. This account of word learning is one of a growing number which attempt to unify acquisition and speech processing in terms of the same real-time, resource-constrained mechanisms (Apfelbaum and McMurray, 2016).

In our model, memorization itself can be viewed as the objective for early word learning. The model attempts to reconstruct its input from memory; chunks that are easy to reconstruct (and that make the context reconstructible) are good candidate words. The working memory model accounts for the two types of bias normally found in Bayesian segmenters. Phonological predictability due to consistent word shapes (Börschinger and Johnson, 2014; Johnson and Goldwater, 2009) reduces the load on the phonological loop. Predictability between words reduces the load on syntactic memory. The two memory systems draw on different cognitive resources, which correspond to different parameters of the model.

2.2 Input representations

As stated above, traditional segmentation models operate on phonemic transcriptions and must be adapted to cope with phonetic or acoustic input. For models which infer an explicit lexicon (i.e., those which do not simply count segment transitions), this takes the form of a mapping between the data and the space of "underlying" latent word forms.

Learning such a mapping can be problematic. Traditional generative learning models use parametric distributions over the data— for acoustics, Gaussians (Vallabha et al., 2007; Feldman et al., 2009) or Gaussian-HMMs (Lee and Glass, 2012; Lee et al., 2015). But these are a notoriously poor fit to real speech sounds (Glass, 2003).

An example of an alternative approach to representation learning from acoustics is Räsänen et al. (2015). They exploit known acoustic indicators of syllable boundaries to infer syllable segments, cluster those segments using expectation-maximization (EM), and then identify multisyllabic words by searching for recurring cluster n-grams. As a result, their system is constrained to propose word boundaries only at proposed syllable boundaries regardless of the representations acquired downstream. Furthermore, EM is known to find non-optimal solutions for many problems in natural language (Johnson, 2007). To the extent that this inhibits their system's ability to exploit information in the acoustic feature space, it might lead to misidentification of recurrent syllable n-grams and consequently to segmentation error.

Latent underlying representations can also cause search problems, since the model must explore all the possible underlying forms which might map to some utterance on the surface. In a probabilistic system capable of mapping every word to every possible realization, this quickly becomes intractable. Many systems use dynamic programming (Mochihashi et al., 2009; Neubig et al., 2010), sometimes with pruning (Van Gael et al., 2008). But these algorithms require Markov models with small context windows, and in any case can still be slow and prone to search errors.

Neural nets, on the other hand, learn a nonlinear mapping between input and output. This allows them to model speech more flexibly, outcompeting Gaussian/HMMs for supervised speech recognition (Graves et al., 2013; Hinton et al., 2012). Recurrent neural nets also produce hidden representations differently than HMMs. Rather than use dynamic programming to search a latent space, they produce a single vector deterministically at each timestep. Models such as LSTMs (Hochreiter and Schmidhuber, 1997) can learn long-distance sequential dependencies in their input without making inference more expensive.

2.3 Neural unsupervised learning

A few previous papers have used neural networks for word segmentation. Christiansen et al. (1998), drawing on older work with Simple Recurrent Networks (Elman, 1990), trains a recurrent network as a language model. Word boundaries are extracted at points where the network predicts an upcoming utterance boundary; that is, utterance boundaries are used as distant supervision for the locations of word boundaries. While effective, this system uses symbolic rather than acoustic input. Moreover, it may have trouble with word endings which do not end utterances, such as the endings of function words; experiments show that infants learn detailed representations of function words by 13 months (Shi et al., 2006) and use known words as "anchors" for segmentation within utterances (Bortfeld et al., 2005).

Rytting (2007) adapts the Christiansen model to variable input by using the posterior probability distribution from a phone recognizer as its feature representation. This system was run on natural data; results for word boundary detection were significantly above chance, though still much less accurate than results for symbolic input. The use of utterance boundaries as distant supervision may create problems for this system similar to those pointed out for Christiansen above. Moreover, the use of an SRN rather than an LSTM means that the system is essentially phonotatic; it makes its decisions based on the previous one or two phones, without the capacity to remember whole lexical items.

Recent work (Kamper et al., 2016) has attempted to harness the flexibility of neural feature extractors within the generative model framework. This model has a hybrid architecture consisting of a neural feature extractor, the Correspondence Autoencoder, pretrained using distant supervision (Kamper et al., 2015), and a Bayesian clustering/segmentation model. The system represents each word by neurally encoding its frames, then downsampling to obtain a fixed-dimensional word vector; the clustering model assumes that these vectors can be modeled with Gaussian clusters. The advantage of this approach is its ability to exploit the known strengths of both Bayesian and neural learning systems. The disadvantage is its indirectness: there is no end-to-end objective to be optimized, and the system's lexical learning does not inform its phonetic representations.

Even outside the domain of segmentation, neural networks have been most successful for supervised problems, and are not widely used for unsupervised learning of discrete structures (trees, clusters, segment boundaries). While some researchers have proposed information-theoretic objectives for learning clusters (Klapper-Rybicka et al., 2001), the most widely used unsupervised objective is the one used here: autoencoding. Yet autoencoders are rarely used to learn discrete hidden structures. One exception, Socher et al. (2011), uses autencoders to find a latent tree structure for sentiment analysis by greedily merging adjacent nodes so as to minimize the reconstruction error.

Chung et al. (2017) describe a model similar to our own which performs a segmentation task using autoencoders. Both models use multiscale autoencoding to learn a sequence model with unknown segment boundaries. The main difference is the different technique used to deal with the discontinuities caused by switching discrete segment boundary variables. However, they evaluate their model on downstream tasks (notably, character language modeling) without evaluating the segmentations directly.

3 The Model

The model uses a basic encoder-decoder architecture now typical in machine translation (Cho et al., 2014) and image captioning (Vinyals et al., 2015). In a typical encoder-decoder, the input is fed into an LSTM sequence model (Hochreiter and Schmidhuber, 1997) which represents it as a latent numeric embedding. This embedding is then fed into another sequence model, which uses it to generate an output sequence. Our two-level model performs this process in stages, first encoding every word, character-by-character, and then encoding the word sequence, vector-by-vector. In an autoencoder, the objective is to make input and output match; thus, the decoder performs the encoding stages in reverse. We provide the final encoder hidden state as input to each decoder unit. To force the system's learned embeddings to be robust to noise caused by mishearing or misremembering, we use dropout (Srivastava et al., 2014) at the input (deleting individual timesteps) and at the word encoding layer (deleting entire words). This architecture is illustrated in Figure 1.

The encoder-decoder does not predict segment boundaries directly, but gives an objective function (reconstruction loss) which can be used to guide segmentation. Because the segment boundary decisions are hard (there are no "partial" boundaries), the loss function is not differentiable as a function of the boundary indicators. We use sampling to estimate the gradient, as in previous work (Mnih et al., 2014; Xu et al., 2015). Our sampling system works as follows: we begin with a proposal distribution P_{seg} over sequences of segment boundaries for the current utterance x. We sample m sequences of boundaries, $B_{1:m}$ from P_{seg}. Each boundary sequence splits the utterance into words. We use the autoencoder network to encode and decode the words, and obtain the loss (the cross-entropy of the reconstructed input) for each sequence, $L_{1:m}$.

We can use the cross-entropy to estimate the posterior probability of the data given a breakpoint sequence (Eq. 1), assuming a uniform prior over

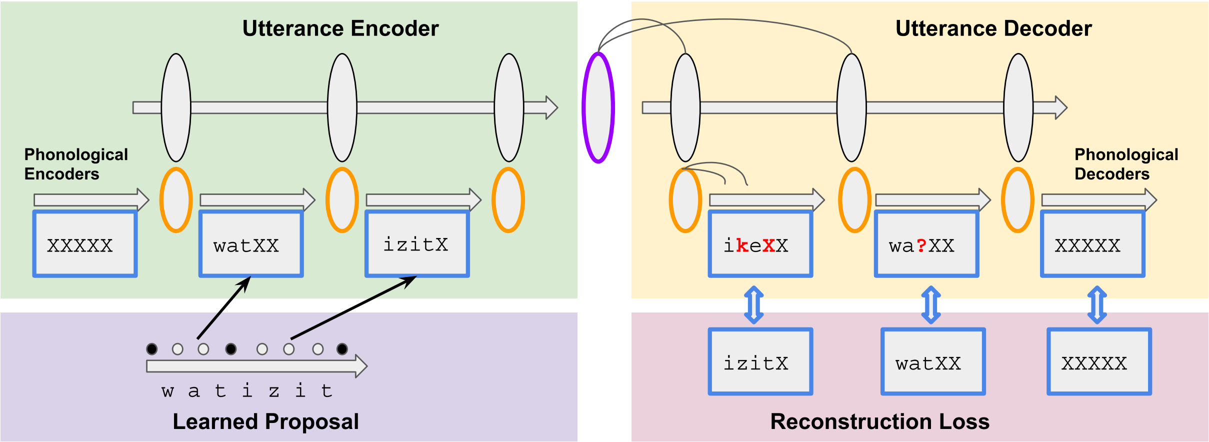


Figure 1: Architecture of the model: top two panels show the encoder/decoder, bottom panels show computation of breakpoints and resulting loss. Horizontal arrows represent LSTMs.

break positions. We then treat each breakpoint t in the utterance independently: for each one, we use the losses and the proposal probabilities to compute an importance weight w_i^t for sample i and position t (Eq. 2), then compute the expected probability of a boundary at that position by summing over the weighted samples (Eq. 3). Essentially, a breakpoint will be more likely if it appeared in samples with low reconstruction loss, especially if it is not encouraged by the current proposal.

$$P(x|B_i) = \frac{P(B_i|x)P(B_i)}{P(x)} \approx \frac{\exp(L_i)}{\sum_j \exp(L_j)} \quad (1)$$

$$w_i^t = \frac{P(x|B_i)}{P_{seg}^t(B_i^t)} \quad (2)$$

$$\mathbb{E}[B(t)] \approx \frac{1}{\sum_i w_i^t} \sum_i w_i^t B_i^t \quad (3)$$

We initialize by making random breakpoint proposals (with probability .1 at each position). The random proposal does not search the space of segmentation boundaries particularly efficiently, so we train a better proposal using another LSTM. This LSTM simply reads the input from left to right and predicts a binary output (segment or not) at each timestep. We update the proposal LSTM by using the sampling-derived P_{seg} as a training target after each batch. Thus, the proposal learns to predict segment boundaries that are likely to result in low reconstruction loss for the main network. To force the system to explore the space, we smooth the learned proposal by interpolating it with a uniform distribution: $P_{seg} = .9 \times P_{LSTM} + .1 \times \frac{1}{2}$.

We control the memory capacity of the system using four tunable parameters: the number of hidden states at the phonological level (H_p) and at the utterance level (H_u) and the dropout probability of mishearing a phonological segment (D_p) or a word (D_u). We discuss parameter tuning results below.

The system also has several other parameters which were not tuned against the evaluation metric. For convenience in GPU training, we treat all sequences as fixed length, either clipping them or padding with a dummy symbol. This requires us to set a maximum length for each word (in characters), and each utterance (in words and characters); we set these parameters to ensure 99% coverage of the input (for the Brent corpus, 7, 10, and 30 respectively).

Clipping creates the possibility of pathological outcomes where the system deliberately creates extremely long words, exploiting the fact that the excess characters will be discarded and will not have to be predicted in the output. We penalize this by subtracting 50 for each deleted character. Finally, we find that, despite pre-training, the system may settle into an initial state where the phonological network simply embeds the characters and the utterance network learns a character LM. To avoid this, we subtract 10 from the objective for each one-symbol word. These parameters were tuned only lightly; we increased the values until the problematic behavior (segmentation of the entire utterance as one word, or each character as a word) ceased.

We implemented the network in Keras (Chollet, 2015), using Adam (Kingma and Ba, 2014) with default settings for optimization. We use minibatches of 128 and take 100 samples of potential segment boundaries per sequence. We perform 10 iterations of pretraining with random boundaries, 10 iterations of boundary induction with random proposals, and 70 iterations of full training with the learned LSTM proposal.

4 Results

4.1 Brent Corpus

The Brent corpus (Brent, 1999) is a standard benchmark dataset for segmentation, consisting of 9790 utterances from Bernstein-Ratner (1987), translated into phonemic transcription using the CMU dictionary. The standard metrics for segmentation are F-score for word boundary detection (treating each boundary in isolation) and F-score for word token segmentation (a word is correct only if both its boundaries are correct and no spurious boundaries intervene). Although early work on Brent used all 9790 utterances for both development and test, we use the first 8000 utterances for parameter tuning. Thus, we present results for the whole corpus (for comparison with previous work) and clean test results for the last 1790.

We tune the four parameters of our system, H_p, H_u, D_p and D_u, using a grid search (see Figure 2). Each subplot shows a particular dropout setting, D_p/D_u; the cells within represent settings of H_p (rows) and H_u (columns), where darker cells have higher boundary F-score. Excessive noise decreases scores, especially high word dropout (right side of the plot). For low levels of dropout, the best systems tend to have small numbers of hidden units (dark regions in the lower left); for larger dropout, more hidden units can be useful. For instance, compare the top left subplot, with 0 dropout and good performance with $H_p = 20, H_u = 100$, to subplot 3,3, with optimal performance at $H_p = 80, H_u = 200$. In other words, limiting the system's memory resources is indeed the key to its performance. The best score occurs at $H_p = 80, H_u = 400, D_p = 0.5, D_u = 0.25$ with a dev boundary F-score of 83%. We used these parameters for our final evaluation, along with 100 hidden units in the proposal network.

To further demonstrate that limited memory can bias the network to learn a low-entropy lexicon, we perform a separate experiment using the phonological encoder/decoder alone. We create networks with varying H_p (setting D_p to 0); for each network size, we train one net on real words from the gold segmentation of Brent, and another on length-matched pseudowords sampled by randomly segmenting the Brent corpus. Figure 3 shows the reconstruction error rates as a function of H_p. The gap between the green and orange lines shows the difference in reconstruction error obtained by using real words rather than pseudowords. For the smallest H_p, neither network does a good job; for the largest, both networks learn the sequences perfectly. For values in between, however, the lines are relatively far apart, showing that the real words are easier for the network to remember.

Our results for Brent, along with selected comparisons, are shown in Table 1.[2] Our system per-

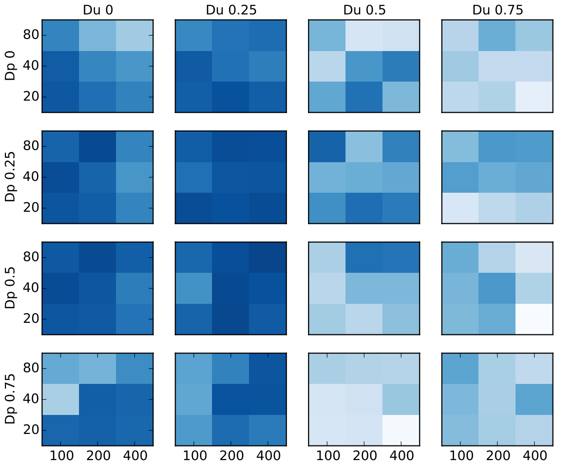


Figure 2: Tuning results on Brent development. Cell axes represent H_u and H_p, darker cells have higher scores (best 83%, worst 60%).

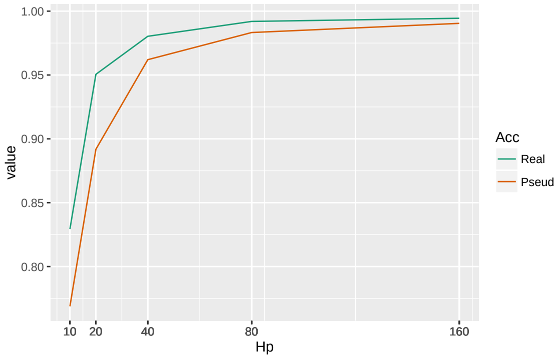


Figure 3: Reconstruction accuracy of the phonological encoder/decoder on real words vs. length-matched pseudowords from Brent.

[2] Comparison system scores are those reported in their

System	Bd P	Bd R	Bd F	Wd F
Goldwater 09	90	74	87	74
Johnson 09	-	-	-	88
Berg-Kirkpatrick 10	-	-	-	88
Fleck 08	95	74	83	71
Ours (all)	81	85	83	72
Ours (test)	81	86	83	72

Table 1: Selected segmentation results on Brent.

forms at the lower end of the reported range for Brent segmenters, scoring 83% for boundary detection and 72% for word detection (comparable to (Fleck, 2008)). (Lignos (2011) scores 93% boundary F on a different corpus with marked syllable boundaries.) From a cognitive modeling point of view, it is not clear what performance we should expect on Brent to model the performance of a young human infant. Models of early word segmentation are motivated by studies showing that, by their first birthday, infants can distinguish many common words from nonwords (Vihman et al., 2004; Swingley, 2005). But this does not imply that they learn every word they hear, or that they can use their word knowledge to segment every utterance correctly. Thus, while our result is not state-of-the-art, it is good enough to conform with the reported infant results and suggest that our neural architecture is a promising direction.

Learning curves for segmentation on the Brent corpus are shown in Figure 4. The first 10 iterations show a gradual increase in segmentation performance using the random proposal. Performance increases sharply with the activation of the learned proposal, then climbs slowly over time. Precision initially exceeds recall (that is, the system proposes too few boundaries) but recall climbs over time as the system exploits known words as "anchors" to discover new ones, a pattern consistent with the infant data (Bortfeld et al., 2005).

4.2 Zerospeech 2015 acoustic segmentation

We began by claiming that an advantage of our model was its flexible architecture that permits dense acoustic features as input (rather than symbolic phone labels) with little modification. In this section, we present preliminary results from a follow-up experiment in which we tested this claim by deploying our system as an acoustic segmenter on the English portion of the Zerospeech 2015 challenge dataset (Versteegh et al., 2015). We preprocess the raw acoustic data by extracting 25ms 13-dimensional mel frequency cepstral coefficients with first and second order deltas at a step size of 10ms. We then train the network on the resulting sequences of 39-dimensional frames.

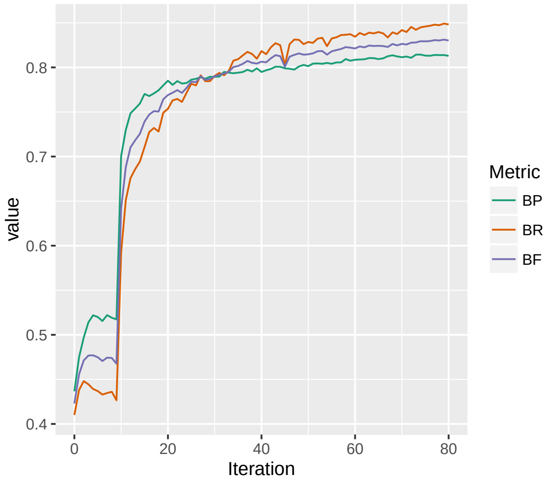


Figure 4: Boundary precision, recall, and F1 score by iteration on the entire Brent dataset.

Given that the goal of the experiment was to test the existing architecture on a novel task, we intentionally conducted this experiment with minimal parameter tuning or architectural modification. However, we made several key changes in response to the unique challenges presented by acoustic input.

First, since we are now reconstructing dense vectors of acoustic features, we use mean squared error (MSE) instead of categorical cross-entropy as the autoencoder loss function. We consequently rescale our clipping penalty from 50 to 1, a coefficient which seemed more in balance with the variation in decoder loss produced by MSE. We also increase our one-letter penalty from 10 to 50, modeling our strong prior assumption that a 1-frame segment will never correspond to a word.

Second, in contrast to the phoneme sequences in the Brent corpus discussed above, utterance boundaries are not observed in acoustic input. The input to the two-level autoencoder must be divided into sequences of utterances, so we imposed utterance boundaries by iteratively consuming the next discovered word in the time series up to the maxi-

respective publications, except for Goldwater et al. (2009), which are corrected numbers published with their software release. Not all systems report all metrics.

System	Bd P	Bd R	Bd F	Wd F
Lyzinski 15	18.8	64.0	29.0	2.4
Räsänen 15	75.7	33.7	46.7	9.6
Räsänen new	61.1	50.1	55.2	12.4
Kamper 16	66.5	58.8	62.4	20.6
Ours	62.4	43.2	51.1	9.3

Table 2: Selected word segmentation results on the the Zerospeech 2015 English corpus.

mum utterance length (in frames). The aforementioned clipping penalties punish the system for utterances that contain too many words, preventing it from optimizing its autoencoder loss by segmenting everywhere.

Third, for our initial proposal distribution we use the speech region segmentation provided by the Zerospeech challenge, consisting of speech intervals identified through automatic voice activity detection (VAD), rather than using the uniform initialization described above for symbolic mode. We interpolate the initial distribution with a uniform prior as described above.

Fourth, we discovered in practice that the assumption of independence between samples made by the importance scoring scheme as implemented for symbolic mode was distortionary in acoustic mode, such that the "best" segmentation discovered through sampling often contained many times more segments than any of its component samples.[3] To prevent this from happening, we simply used 1-best rather than importance sampling for acoustic segmentation.

We trained the system for 80 iterations using parameters $H_p = 20, H_u = 400, D_p = 0, D_u = 0.25$ and 1500 hidden units in the proposal LSTM. In the auto-encoder network, we limited frames per utterance, words per utterance, and frames per word to 400, 16, and 100, respectively. Results are presented in Table 2, along with a comparison to results from other systems. Lyzinski et al. (2015) and Räsänen et al. (2015) were entrants in the Zerospeech 2015 challenge, in which Räsänen et al. (2015) performed best in the word boundary detection measure. As shown in the table, our system beats both of these competitors' boundary detection scores, with a word detection score comparable to that of Räsänen et al. (2015). However, since the challenge concluded, Räsänen et al. (under review) have modified their system and improved their segmentation score,[4] and Kamper et al. (2016) have established a new state of the art for this task. While our system currently remains far from these newer benchmarks, we expect that with systematic parameter tuning and investigation into appropriate sampling procedures for acoustic input, we might be able to improve substantially on the results presented here. We believe that the results of this preliminary investigation into the acoustic domain are promising, and that they bear out our claims about the flexibility of our general architecture.

5 Conclusions and future directions

This work presented a new unsupervised LSTM architecture for discovering meaningful segments in representations of continuous speech. Memory limitations in the autoencoder part of the network apply pressure to discover compressed representations much as human memory limitations have been argued to guide lexical acquisition. By varying the size of the LSTM's hidden state, we showed that word segmentation performance on the Brent corpus is driven by memory limitations, with performance improving (up to a point) as we constrain the system's memory capacity. And by successfully deploying our system on both symbolic (character) and acoustic representations of speech, we demonstrated that our approach is flexible enough to adapt to either representation of the speech stimulus.

In the future we hope to pursue a number of lines of inquiry. We plan to conduct more detailed parameter tuning in the acoustic domain and to segment the Xitsonga dataset supplied with the Zerospeech 2015 challenge. We also intend to introduce additional layers into the autoencoder network so as to allow for joint acquisition of phone-like, morph-like, and/or word-like units in the acoustic signal; this may benefit from the alternate model structure of Chung et al. (2017). And we plan to explore clustering techniques that

[3] We believe this is driven by training batches in which multiple samples receive similar scores but have fairly non-overlapping segmentations. In this case, the output segmentation can contain something close to the union of the best samples' segmentation points, leading to oversegmentation. This effect is likely exaggerated in acoustic mode as compared to symbolic mode because acoustic word segments are generally much longer (in frames) than their corresponding symbolic word segments (in characters).

[4] The new results are not yet published. Those reported above are copied from the results summary in Kamper et al. (2016).

would allow our system to discover categories in addition to probable segmentation points.

Acknowledgments

We thank members of the Clippers discussion group (especially Joo-Kyung Kim, Eric Fosler-Lussier and Wei Xu) and three anonymous reviewers. Computations for this project were run on a Titan-X GPU donated by the NVIDIA Hardware Grant program and on the Ohio Supercomputer (1987). Funding was provided by NSF #1422987.

References

Keith S Apfelbaum and Bob McMurray. 2016. Learning during processing: Word learning doesn't wait for word recognition to finish. *Cognitive Science* .

Alan Baddeley. 2007. *Working memory, thought and action*. Oxford University Press, Oxford, UK.

Alan Baddeley, Susan Gathercole, and Costanza Papagno. 1998. The phonological loop as a language learning device. *Psychological review* 105(1):158.

Alan Baddeley and Graham Hitch. 1974. Working memory. *Psychology of learning and motivation* 8:47–89.

Elika Bergelson and Daniel Swingley. 2012. At 6–9 months, human infants know the meanings of many common nouns. *Proceedings of the National Academy of Sciences* 109(9):3253–3258.

Nan Bernstein-Ratner. 1987. The phonology of parent-child speech. In K. Nelson and A. van Kleeck, editors, *Children's Language*, Erlbaum, Hillsdale, NJ, volume 6.

Benjamin Börschinger and Mark Johnson. 2014. Exploring the role of stress in Bayesian word segmentation using adaptor grammars. *Transactions of the Association for Computational Linguistics* 2:93–104.

Heather Bortfeld, James L. Morgan, Roberta Michnick Golinkoff, and Karen Rathbun. 2005. Mommy and me. *Psychological Science* 16(4):298–304. https://doi.org/10.1111/j.0956-7976.2005.01531.x.

Michael R. Brent. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning* 34:71–105.

Ohio Supercomputer Center. 1987. Ohio supercomputer center. `http://osc.edu/ark:/19495/f5s1ph73`.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Morten H. Christiansen, Joseph Allen, and Mark S. Seidenberg. 1998. Learning to segment speech using multiple cues: A connectionist model. *Language and Cognitive Processes* 13(2/3):221–269.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical multiscale recurrent neural networks. In *Proceedings of ICLR*.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2015. Modeling word forms using latent underlying morphs and phonology. *Transactions of the Association for Computational Linguistics* 3:433–447.

Robert Daland and Janet B. Pierrehumbert. 2011. Learning diphone-based segmentation. *Cognitive Science* 35(1):119–155. https://doi.org/10.1111/j.1551-6709.2010.01160.x.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14(2):179–211.

Micha Elsner, Sharon Goldwater, Naomi Feldman, and Frank Wood. 2013. A joint learning model of word segmentation, lexical acquisition, and phonetic variability. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 42–54. http://www.aclweb.org/anthology/D13-1005.

Naomi Feldman, Thomas Griffiths, and James Morgan. 2009. Learning phonetic categories by learning a lexicon. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*.

Margaret M. Fleck. 2008. Lexicalized phonotactic word segmentation. In *Proceedings of ACL-08: HLT*. Association for Computational Linguistics, Columbus, Ohio, pages 130–138. http://www.aclweb.org/anthology/P/P08/P08-1016.

James Glass. 2003. A probabilistic framework for segment-based speech recognition. *Computer Speech and Language* 17:137–152.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition* 112(1):21–54.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP)*. IEEE, pages 6645–6649.

Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine* 28(6):82–97.

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

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *EMNLP*.

Mark Johnson and Sharon Goldwater. 2009. Improving nonparametric Bayesian inference: Experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Boulder, Colorado.

Peter W Jusczyk and Richard N Aslin. 1995. Infants' detection of the sound patterns of words in fluent speech. *Cognitive psychology* 29(1):1–23.

Herman Kamper, Micha Elsner, Aren Jansen, and Sharon Goldwater. 2015. Unsupervised neural network based feature extraction using weak top-down constraints. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, pages 5818–5822.

Herman Kamper, Aren Jansen, and Sharon Goldwater. 2016. A segmental framework for fully-unsupervised large-vocabulary speech recognition. *arXiv preprint arXiv:1606.06950* .

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Magdalena Klapper-Rybicka, Nicol N Schraudolph, and Jürgen Schmidhuber. 2001. Unsupervised learning in LSTM recurrent neural networks. In *International Conference on Artificial Neural Networks*. Springer, pages 684–691.

Chia-ying Lee and James Glass. 2012. A nonparametric Bayesian approach to acoustic model discovery. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Jeju Island, Korea, pages 40–49. http://www.aclweb.org/anthology/P12-1005.

Chia-ying Lee, Timothy J O'Donnell, and James Glass. 2015. Unsupervised lexicon discovery from acoustic input. *Transactions of the Association for Computational Linguistics* 3:389–403.

Constantine Lignos. 2011. Modeling infant word segmentation. In *Proceedings of the fifteenth conference on computational natural language learning*. Association for Computational Linguistics, pages 29–38.

Vince Lyzinski, Gregory Sell, and Aren Jansen. 2015. An evaluation of graph clustering methods for unsupervised term discovery. In *Proceedings of Interspeech 2015*.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*. pages 2204–2212.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, Suntec, Singapore, pages 100–108. http://www.aclweb.org/anthology/P/P09/P09-1012.

Graham Neubig, Masato Mimura, Shinsuke Mori, and Tatsuya Kawahara. 2010. Learning a language model from continuous speech. In *11th Annual Conference of the International Speech Communication Association (InterSpeech 2010)*. Makuhari, Japan, pages 1053–1056.

Steven T Piantadosi, Harry Tily, and Edward Gibson. 2012. The communicative function of ambiguity in language. *Cognition* 122(3):280–291.

Okko Räsänen, Gabriel Doyle, and Michael C. Frank. 2015. Unsupervised word discovery from speech using automatic segmentation into syllable-like units. In *Proceedings of Interspeech 2015*. pages 3204–3208.

Okko Räsänen, Gabriel Doyle, and Michael C. Frank. under review. Pre-linguistic rhythmic segmentation of speech into syllabic units.

Anton Rytting. 2007. *Preserving Subsegmental Variation in Modeling Word Segmentation (Or, the Raising of Baby Mondegreen)*. Ph.D. thesis, The Ohio State University.

Anton Rytting, Chris Brew, and Eric Fosler-Lussier. 2010. Segmenting words from natural speech: subsegmental variation in segmental cues. *Journal of Child Language* 37(3):513–543.

Rushen Shi, Janet F Werker, and Anne Cutler. 2006. Recognition and representation of function words in english-learning infants. *Infancy* 10(2):187–198.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 151–161.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Daniel Swingley. 2005. Statistical clustering and the contents of the infant vocabulary. *Cognitive Psychology* 50:86–132.

Gautam K. Vallabha, James L. McClelland, Ferran Pons, Janet F. Werker, and Shigeaki Amano. 2007. Unsupervised learning of vowel categories from infant-directed speech. *Proceedings of the National Academy of Sciences* 104(33):13273–13278.

Jurgen Van Gael, Yunus Saatci, Yee Whye Teh, and Zoubin Ghahramani. 2008. Beam sampling for the infinite Hidden Markov model. In *Proceedings of the 25th International Conference on Machine learning*. ACM, New York, NY, USA, ICML '08, pages 1088–1095. https://doi.org/10.1145/1390156.1390293.

Maarten Versteegh, Roland Thiollière, Thomas Schatz, Xuan Nga Cao, Xavier Anguerra, Aren Jansen, and Emmanuel Dupoux. 2015. The zero resource speech challenge 2015. In *Proceedings of Interspeech 2015*.

Marilyn M Vihman, Satsuki Nakai, Rory A DePaolis, and Pierre Hallé. 2004. The role of accentual pattern in early lexical representation. *Journal of Memory and Language* 50(3):336–353.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3156–3164.

Kelvin Xu, Jimmy L. Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of ICML*. JMLR, Lille, France.

Speaking, Seeing, Understanding: Correlating semantic models with conceptual representation in the brain

Luana Bulat, Stephen Clark and Ekaterina Shutova
Computer Laboratory
University of Cambridge
{luana.bulat,stephen.clark,ekaterina.shutova}@cl.cam.ac.uk

Abstract

Research in computational semantics is increasingly guided by our understanding of human semantic processing. However, semantic models are typically studied in the context of natural language processing system performance. In this paper, we present a systematic evaluation and comparison of a range of widely-used, state-of-the-art semantic models in their ability to predict patterns of conceptual representation in the human brain. Our results provide new insights both for the design of computational semantic models and for further research in cognitive neuroscience.

1 Introduction

Recent years have witnessed many breakthroughs in data-driven semantic modelling: from the log-linear skip-gram model of Mikolov et al. (2013a) to multi-modal meaning representations (Bruni et al., 2012; Kiela and Bottou, 2014; Kiela and Clark, 2015; Kiela et al., 2015a). These models boast of a higher performance accuracy in numerous semantic tasks, including modeling semantic similarity and relatedness (Silberer and Lapata, 2012), lexical entailment (Kiela et al., 2015b), analogy (Mikolov et al., 2013b) and metaphor (Shutova et al., 2016). However, less is known about the extent to which such models correlate with and reflect human conceptual representation.

Much research in the cognitive neuroscience community has been concerned with uncovering how the brain represents conceptual knowledge, by leveraging brain activation data associated with the meanings of concepts obtained during functional magnetic resonance imaging (fMRI) experiments. In the computational linguistics community, the availability of such fMRI data provides researchers with a benchmark for evaluating semantic model performance in terms of their ability to represent human semantic memory. Mitchell et al. (2008) were the first to demonstrate that distributional semantic models encode some of the patterns found in the fMRI data. Other researchers followed in their steps, evaluating traditional count-based distributional models (Devereux et al., 2010; Murphy et al., 2012), topic model-based semantic features (Pereira et al., 2013), psycholinguistic and behavioural features (Palatucci et al., 2009; Chang et al., 2010; Fernandino et al., 2015) and visual representations (Anderson et al., 2013, 2017). While all of these studies report correlation between the investigated semantic models and patterns found in the brain imaging data, their focus on individual models and the use of different datasets and prediction methods make their results difficult to compare and to integrate into a coherent evaluation landscape. The work of Murphy et al. (2012) is an exception, in that the authors systematically compare several distributional models with a range of parameters on the same brain imaging dataset. However, they focus on the traditional count-based distributional models only.

We take inspiration from the works of Mitchell et al. (2008) and Murphy et al. (2012); however, we conduct a more extensive study of the ability of different types of semantic models to predict the patterns of brain activity associated with conceptual representation. We evaluate and compare several kinds of semantic models, using different modalities and data sources: (1) traditional count-based distributional models (with word window-based and dependency-based contexts) learnt from text; (2) log-linear skip-gram models (with word window-based and dependency-based contexts); (3) behavioural models based on the free association task; (4) word representations learnt from

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1081–1091
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

visual data; and (5) multi-modal word representations combining linguistic and visual information. Unlike previous studies, where evaluations were typically conducted using a single technique, we evaluate our models using several methods: ridge regression (Hoerl and Kennard, 1970), similarity-based encoding and similarity-based decoding (Anderson et al., 2016). Such an experimental setup allows for a comprehensive evaluation and comparison of the models.

To the best of our knowledge the dependency-based skip-gram model and the free association-based model, as well as their multimodal counterparts, have not been previously evaluated on the brain activity prediction task. Other models have been evaluated individually and have not yet been systematically compared within a single evaluation framework. Providing such a comparison, our experiments and results demonstrate that (1) visual information is a stronger predictor of brain activity than the linguistic information for concrete nouns; (2) sparse text-based models, whether dependency-based or built using linear bag-of-words context, tend to predict neural activity more accurately than dense models; (3) cognitively-motivated association-based models perform on par with or better than other linguistic models, which suggests that they provide an interesting avenue in computational semantics research.

2 Related work

The seminal work of Mitchell et al. (2008) introduced a new semantic model able to predict brain activation data associated with the meanings of concrete nouns from their corpus-harvested semantic representations. They chose a set of 25 verbs to act as semantic features in their distributional model, inspired by the importance of sensory-motor features in neural representations of concepts (Cree and McRae, 2003).

Since then, various studies have shown that distributional semantic models encode and are able to predict neural activation patterns associated with concepts (Devereux et al., 2010; Murphy et al., 2012; Pereira et al., 2013). Devereux et al. (2010) build on the work of Mitchell et al. (2008) and show that automatically acquired feature-norm like semantic representations can make equally powerful predictions about brain activity associated with the presentation of words. Pereira et al. (2013) use semantic features learnt from topic models on Wikipedia to predict neural activation patterns for unseen concepts.

Several other studies have demonstrated the fit of semantic models built from human behavioural data with regard to predicting neural activation patterns (Palatucci et al., 2009; Chang et al., 2010; Fernandino et al., 2015). Chang et al. (2010) use brain region encodings as well as detailed taxonomic encodings of McRae et al. (2005) feature norms to predict brain activation patterns using a linear regression model. They demonstrate that learned brain activity patterns can be used to decode mental states. Fernandino et al. (2015) use human elicited attribute salience scores based on five sensory-motor attributes (sound, color, visual motion, shape and manipulation) to derive fMRI brain activation patterns for concrete words, but are unsuccessful at modeling neural activation patterns for abstract words.

Recent advances in multi-modal semantics have shown that grounding semantic models in sensory modalities improves performance on a variety of tasks (Silberer and Lapata, 2012; Bruni et al., 2012; Kiela and Bottou, 2014; Bulat et al., 2016). Anderson et al. (2013) show that semantic models built from visual data correlate highly with fMRI-based brain activation patterns. Anderson et al. (2015) find that similarity in activity in the brain areas related to linguistic processing can be better predicted from text-based semantic representations, whilst image-based representations perform better at predicting similarity in the visual processing areas of the brain. In line with the dual coding theory, Anderson et al. (2017) demonstrate an advantage in decoding brain activity patterns of abstract words for text-based semantic models over the image-based ones. Contrary to previous findings, Anderson et al. (2017) find no advantage in decoding neural activity patterns associated with concrete words for image-based models.

Murphy et al. (2012) present the first study systematically comparing several text-based semantic models on the brain activity prediction task. They focus on the traditional count-based distributional models and achieve the best performance using dependency-based features. Our study is more extensive than that of Murphy et al. (2012), as we evaluate both the count-based models and the more recent skip-gram word embeddings, as well as comparing them to free association-based, visual and multi-modal semantic representations. While Murphy and colleagues evaluate the models using one method only — linear regression, we compare predicted neural activation patterns obtained using both regression and the similarity-based encoding and decoding methods proposed by Anderson et al. (2016).

3 Brain imaging data

We use the dataset of fMRI neural activation patterns associated with the meanings of nouns, created by Mitchell et al. (2008) as described below.

3.1 fMRI experiment

Nine right-handed adults between the age of 18 and 32 (five female) participated in the study. They were presented with line drawings and noun labels for 60 concrete nouns from 12 semantic classes – animals, body parts, buildings, building parts, clothing, furniture, insects, kitchen items, tools, vegetables, vehicles and man-made objects – with five exemplars per class. The task for the participants during the scanning was to think about the properties of the noun stimuli they were presented with. The entire set of 60 stimulus words was presented six times to every participant, in a different order for each presentation.

The fMRI images were acquired on a Siemens Allegra 3.0T scanner. The initial data was corrected for slice timing, motion and linear trend; spatially normalised and resampled to 3x3x6mm^3 voxels. Only those voxels overlapping with the cortex were selected (approximately 20000 for every participant).

3.2 Voxel selection

We employ the same voxel selection procedure as Mitchell et al. (2008) for evaluating the similarity between actual fMRI images and model-predicted fMRI images. Similarity is computed by only taking into account 500 voxels with the most stable activation profile across words – with profiles compared across the six presentations. The evaluation is performed using leave-two-out cross validation. Voxel selection was performed independently for each of the cross validation folds, at training time. A voxel's stability score across the six presentations was approximated as the mean pairwise Pearson correlation between its activation profiles over the 58 training words in the cross-validation fold. The 500 voxels with the highest stability score were chosen.

3.3 Brain activity vectors

We evaluate our models on the data of each participant independently. Following Mitchell et al. (2008), we obtain a single fMRI image per concept (a representative image) by first computing the mean fMRI response over its six presentations, and then subtracting the mean of all 60 of these representative images from each. In the rest of this paper we will refer to these representations as *brain activity vectors*.

4 Semantic models

MITCHELL As a benchmark for all other semantic models, we use the publicly available[1] co-occurrence based semantic vectors developed in the Mitchell et al. (2008) study. The features of this semantic space are 25 sensory-motor verbs. Co-occurrence statistics were collected using a window size of 5 words either side of the target word, on a trillion-word corpus provided by Google.

4.1 Text-based semantic models

We train a variety[2] of context-counting and context-predicting text-based semantic models on the January 2016 dump of Wikipedia, which was tokenised using the Stanford NLP tools[3], lemmatised with the Morpha lemmatiser (Minnen et al., 2001), and parsed with the C&C parser (Clark and Curran, 2007).

DISTRIB We obtain count-based distributional semantic models, using the top 10K most frequent lemmatised words in the corpus (excluding stopwords) as contexts. The context window is defined as sentence boundaries. Counts are reweighted using positive pointwise mutual information (PPMI) and vectors are L2-normalised.

SVD300 We also construct 300-dimensional dense semantic representations by applying singular value decomposition (SVD) (Deerwester et al., 1990) to DISTRIB.

[1] https://www.cs.cmu.edu/afs/cs/project/theo-73/www/science2008/data.html

[2] We have experimented with different parameter settings for each type of language-based semantic space (e.g. size of the vectors, number of iterations when learning the embeddings etc.) and found that the reported vectors with "standard' settings perform the best (or do not get significantly outperformed).

[3] https://nlp.stanford.edu/software/index.shtml

DEPS Following Murphy et al. (2012), who find that dependency-based semantic vectors perform best on a neurosemantic decoding task, we also include such a semantic space in our comparison. Vector representations are created by leveraging the dependency relations output by the C&C parser (Clark and Curran, 2007) as features. We use both the incoming and outgoing dependency relations as features; for example, given the dependency relation (RUN, DOBJ, MARATHON) we extract the tuple (DOBJ, MARATHON) as a feature for RUN and (!DOBJ, RUN) as a feature for MARATHON. The top 10K most frequent dependency features are used as contexts and counts are re-weighted using PPMI.

DEPS-SVD300 We also obtain 300-dimensional dense dependency-based semantic representations by applying SVD to DEPS.

EMBED-BOW We train 300-dimensional embeddings using the standard log-linear skipgram model with negative sampling of Mikolov et al. (2013a). The embeddings were trained using linear bag-of-words contexts, with the window defined as $k = 2$ (**EMBED-BOW2**) or $k = 5$ (**EMBED-BOW5**) words either side of the target word. We use 10 negative samples per word-context pair and 15 iterations over the corpus.

EMBED-DEPS In addition to the embeddings trained with linear bag-of-words contexts, we also obtain 300-dimensional dependency-based word embeddings using the Levy and Goldberg (2014) implementation of the generalised skip-gram with arbitrary contexts model. Using both incoming and outgoing dependency relations output by the C&C parser, we create word-context pairs using all words and contexts occurring more than 400 times in the corpus. This resulted in a vocabulary of about 92,000 words, with over 250,000 distinct syntactic contexts. We use 10 negative samples per word-context pair and 15 iterations over the corpus.

4.2 Association-based semantic model

Free word association datasets (Nelson et al., 2004; De Deyne et al., 2016) represent a rich source of semantic information and have been successfully used in NLP, including research on semantic memory (Steyvers et al., 2004) and multi-modal semantics (Hill and Korhonen, 2014). Recent studies have shown the superiority of semantic models built using data collected from *multiple-response* free association tasks — where subjects are asked to list multiple associative cues for every target word rather than a single association — over the models built from single-response ones (De Deyne et al., 2013). Moreover, such association-based semantic models have been shown to outperform current state-of-the-art text-based language models on concept relatedness and similarity judgments (De Deyne et al., 2016).

We make use of the word association dataset collected as part of the Small World of Words[4] project, where more than 100K fluent English speakers were asked to list three associations for each target word. The dataset contains multiple-response association data for over 10K words. We use a subset of this dataset, where all target words have at least 50 primary, 50 secondary and 50 tertiary responses and all responses also appear as normed target words[5].

ASSOC We construct a count-based semantic model of word associations (henceforth ASSOC) similarly to a count-based distributional model: the responses are treated as semantic features, and counts are replaced by the sum of primary, secondary and tertiary association frequencies between the target word and the responses. Counts are re-weighted using PPMI and vectors are L2-normalised. The association-based representations obtained for the 60 target words in the Mitchell et al. (2008) dataset under this model are 9854-dimensional.

4.3 Image-based semantic model

We also build state-of-the-art deep visual semantic representations (henceforth VISUAL) for the 60 concepts in the Mitchell et al. (2008) dataset. Following previous work in multi-modal semantics (Bergsma and Goebel, 2011; Kiela and Bottou, 2014) and the findings of a recent study of system architectures and data sources for constructing visual representations (Kiela et al., 2016), we retrieve 10 images per concept from Google Images. We use the MMFeat toolkit[6] (Kiela, 2016) to build our image representations. We extract the 4096-dimensional pre-softmax layer from a for-

[4] `https://smallworldofwords.org/`

[5] Total of 9854 words (appearing as both target and responses) and 1092251 association pairs

[6] `https://github.com/douwekiela/mmfeat`

ward pass through a convolutional neural network (Krizhevsky et al., 2012), which has been pre-trained on the ImageNet classification task using Caffe (Jia et al., 2014). We obtain the visual representation for a given concept by taking the mean of the 10 resulting image representations.

4.4 Multi-modal semantic models

We also included multi-modal semantic spaces in our analysis, as these are currently widely used in NLP and have been previously shown to achieve the best performance at predicting conceptual encodings in the brain (Anderson et al., 2015). Multi-modal semantic spaces are constructed by combining the visual (VISUAL) and respective linguistic (e.g. MITCHELL, DISTRIB, DEPS) or association-based (ASSOC) representations into a multi-modal representation by concatenating their respective L2-normalized vectors.

5 Methods

In this study, we use two different ways of analysing the correlation between the semantic models described in Section 4 and the fMRI brain activation patterns used as a proxy for human conceptual representation. First, we compare these semantic models in their predictive power, by looking at how well they can synthesise, i.e. predict, brain activation patterns for unseen concepts (Section 5.1). Secondly, we look at how well they are able to decode neural activation patterns by measuring their success at predicting the stimulus that produced an unlabeled (unseen) fMRI pattern (Section 5.2).

5.1 Predicting brain activity patterns

The brain activity prediction task has been used in previous NLP research as a method of evaluating different semantic models in their ability to model conceptual representation. Most of these studies learn a mapping function between the semantic model of choice and the fMRI neural activity patterns using regression techniques (Mitchell et al., 2008; Devereux et al., 2010; Murphy et al., 2012). Recent work by Anderson et al. (2016) introduce a new method for synthesising fMRI activity patterns through similarity-based encoding that does not require model fitting. We compare the prediction performance of the semantic models detailed in Section 4 by implementing both a regression-based model and the similarity-based encoding algorithm of Anderson et al. (2016).

Regression-based learning Following previous work (Mitchell et al., 2008; Devereux et al., 2010; Murphy et al., 2012), for every participant, we learn a mapping function between semantic model features and brain activation vectors using linear regression. The learned weights are used to make predictions about brain activation vectors associated with concepts that were not seen during training. We implement Ridge regression (Hoerl and Kennard, 1970), a multiple linear regression model that uses a least squares loss function and L2 regularisation.

Similarity-based encoding We implement the similarity-based encoding method introduced by Anderson et al. (2016). This method predicts the brain activity vector for an unseen concept by exploiting its similarity (with respect to a particular semantic model) to words for which we have observed brain activity vectors.

The first step in predicting a brain activity vector for an unseen concept is to compute its *semantic model similarity code*. This is a N-dimensional[7] vector of similarity scores — computed using Pearson's correlation — between the unseen concept and the N words for which we have brain activation vectors[8]. The predicted brain activity vector for the unseen concept is then "synthesised" by using its semantic model similarity code to weight a superposition of brain activity vectors:

$$\vec{b\prime} = \frac{1}{C}\sum_{i=1}^{N} \vec{b}_i \cdot corr(\vec{v}_i, \vec{v}_{N+1}) \qquad (1)$$

Assuming the unseen word is indexed $N+1$ and $\vec{v}_j$ is the semantic model representation of word j, C is a normalisation constant defined as the sum of absolute values of elements in the semantic model similarity code:

$$C = |\sum_{i=1}^{N} corr(\vec{v}_i, \vec{v}_{N+1})| \qquad (2)$$

5.2 Decoding neural activity patterns

We then evaluate our semantic models in terms of their ability to decode unseen fMRI activation patterns. The analysis in this case does not involve

[7] Assuming that we have N words for which we have both semantic model representations (e.g. DISTRIB vectors) and observed brain activation vectors.

[8] The similarities are measured w.r.t. the semantic model we use as "predictor", e.g. DISTRIB, SVD300 or VISUAL

synthesising brain activation vectors for new concepts, but predicting the correct label (stimulus) associated with a given fMRI pattern.

We implement the similarity-based decoding procedure as detailed in Anderson et al. (2016). The first step is to obtain the *semantic model similarity matrix* — by computing the semantic model similarity codes for each of the 60 concepts in the Mitchell et al. (2008) dataset (as described above) — and the *brain activity similarity matrix* — by computing brain activity similarity codes.

At test time, two of the N words are chosen for decoding, together with their respective semantic model similarity codes $(\vec{s}_i, \vec{s}_j)$ and brain activity similarity codes $(\vec{a}_i, \vec{a}_j)$. Next, $\vec{s'_i}$, $\vec{s'_j}$, $\vec{a'_i}$ and $\vec{a'_j}$ are obtained by removing the i-th and j-th elements in $\vec{s}_i$, $\vec{s}_j$, $\vec{a}_i$ and $\vec{a}_j$ respectively, because entries in the similarity vectors corresponding to the test words would reveal the correct answer in the matching task. We will refer to $\vec{s'_i}$ and $\vec{s'_j}$ as *reduced semantic model similarity codes*, and by analogy to $\vec{a'_i}$ and $\vec{a'_j}$ as *reduced neural similarity codes*.

Decoding is considered a success if the sum of Pearson's correlations for the correct pairings ($corr(\vec{s'_i}, \vec{a'_i}) + corr(\vec{s'_j}, \vec{a'_j})$) is higher than the sum of Pearson's correlations for the incorrect pairings ($corr(\vec{s'_i}, \vec{a'_j}) + corr(\vec{s'_j}, \vec{a'_i})$).

6 Experiments

All semantic spaces presented in Section 4 have full coverage on the Mitchell et al. (2008) dataset. All experiments detailed in this section were performed separately for every participant and evaluated using leave-two-out cross validation.

6.1 Regression experiments

We repeatedly train a regression model to fit brain activation vectors for each of the semantic spaces described in Section 4, using only 58 of the 60 available concept representations (leave-two-out cross validation). This resulted in 1770 cross-validation folds.[9] The only hyperparameter in the regression is λ, which controls the degree of regularisation. The λ hyperparameter was optimised when training each cross-validation fold, by choosing from the range 0.0001 to 100 through generalised cross validation (i.e. λ was optimised by only looking at the training items during each cross-validation fold).

During each testing round, we used the learned mapping function to construct predicted brain activation vectors for the two held out words.We evaluated each of the semantic models by computing its accuracy of matching the two predicted brain activation vectors with the two observed ones. A matching score was computed by analysing the cosine similarity between the predicted and the observed brain activation vectors. If the sum of similarities for the correct pairing was higher than the one for the incorrect pairing the matching accuracy was set to 1 for this cross-validation fold, and otherwise it was set to 0. If the model was choosing the match at random, the expected accuracy is 0.50. The similarity between two brain activation vectors was computed by only taking into account the 500 most stable voxels (during each cross-validation fold) as detailed in Section 3.2. The cross-validated accuracies for each of our semantic models are presented in Table 1, with selected results also shown in Figure 1. We only report results on two multi-modal models (VISUAL+MITCHELL and VISUAL+ASSOC), as there was no significant difference in performance between any pair of multi-modal models.

All semantic models learn to predict neural activation patterns for unseen words significantly above chance level. Association-based semantic models (ASSOC) significantly[10] outperform all dense semantic representations (whether embedding-based or SVD-reduced), with $p < 0.05$. Sparse text-based representations with linear context (DISTRIB and DEPS) significantly outperform some dense semantic representations. However, no dense semantic models significantly outperform DISTRIB and DEPS. There is no significant difference between the performance of ASSOC, DISTRIB and DEPS. Contrary to the findings of Murphy et al. (2012), we do not find any advantage in predicting brain activation patterns from dependency-based text models.

Both VISUAL and multi-modal models significantly outperform text-based models overall ($p < 0.05$), excepting MITCHELL with $p < 0.11$ when comparing to VISUAL and $p < 0.09$ when comparing against multi-modal semantic models. These results support previous findings regarding the importance of grounding semantic models in perceptual input. These grounded semantic models per-

[9] There are (60 choose 2) ways to choose two test items from the 60 Mitchell et al. (2008) concepts.

[10] We used (pairwise) paired t-tests to judge the statistical significance of the difference in performance between any two models within the same experiment.

MODEL	P1	P2	P3	P4	P5	P6	P7	P8	P9	*mean*
MITCHELL	0.78	0.72	0.71	0.75	**0.76**	0.56	0.71	**0.63**	0.63	0.70
DISTRIB	0.85	0.67	0.73	0.84	0.72	0.55	0.70	0.54	0.69	0.70
SVD300	0.85	0.65	0.68	0.77	0.67	0.53	0.66	0.52	0.62	0.66
DEPS	0.85	0.70	0.77	0.86	0.74	0.40	0.70	0.59	**0.72**	0.70
DEPS-SVD300	0.80	0.68	0.74	0.81	0.70	0.32	0.68	0.61	0.66	0.67
EMBED-BOW2	0.85	0.65	0.70	0.78	0.64	0.55	0.60	0.57	0.65	0.66
EMBED-BOW5	0.83	0.62	0.72	0.74	0.66	0.56	0.70	0.56	0.58	0.66
EMBED-DEPS	0.82	0.60	0.67	0.81	0.67	0.49	0.63	0.62	**0.72**	0.67
ASSOC	**0.90**	0.65	0.78	0.87	0.74	0.51	0.75	0.60	0.67	0.72
VISUAL	**0.90**	**0.78**	**0.85**	**0.88**	0.69	0.56	0.75	0.57	0.69	0.74
VISUAL+ASSOC	**0.90**	**0.78**	0.84	0.86	0.70	**0.58**	**0.76**	0.56	0.70	0.74
VISUAL+MITCHELL	**0.90**	**0.78**	0.84	0.86	0.70	**0.58**	**0.76**	0.56	0.70	0.74

Table 1: *Regression results.* Cross-validated accuracies for models trained on participants P1 through P9, together with mean over participants.

form as well as models that encode mental representations through associations (ASSOC). There is no significant advantage for multi-modal models over VISUAL.

6.2 Similarity-based encoding experiments

We also compare performance of the semantic models when the predicted brain activation vector is computed using the Anderson et al. (2016) similarity based encoding method. We use a leave-two-out cross validation strategy, to match previous work and our experiments detailed in Section 6.1. The similarity-based encoding approach does not require any mapping function to be learned, hence is a robust and fast way to obtain synthesised brain activation vectors for unseen words.

During each cross-validation fold, semantic model similarity codes of the two test words were computed using the procedure outlined in Section 5.1. Predicted brain activation vectors were then synthesised for the two test words by weighting a superposition of brain activity vectors using their semantic model similarity codes. The matching score for each of the cross-validation folds was computed in the same way as in the case of the regression model (Section 6.1). The only difference was that we measured the similarity between the two brain activation vectors using Pearson's correlation coefficient, following Anderson et al. (2016). As in the previous experiment, the expected chance performance of this method is 0.5. The cross-validated accuracies for each of our semantic models are shown in Table 2, with selected results also shown in Figure 1.

All semantic models perform significantly above chance level. As in the case of the regression experiments, there is a clear advantage in synthesising brain activation vectors for visually grounded models (VISUAL and multi-modal models) over the language-based ones (this time including MITCHELL), as well as ASSOC. When looking at the performance of the text-based models in general, there is no difference in performance when comparing context-predicting models to count-based ones, or sparse semantic models to dense ones.

6.3 Brain activation pattern decoding

In the similarity-based decoding experiments, we assess the ability of semantic models to identify the correct stimulus for a given brain activation pattern, using the same leave-two-out cross-validation strategy. At test time, we obtain the reduced semantic model similarity codes and the reduced neural similarity codes for the two test items as described in Section 5.2. It is important to note that these similarity code vectors do not contain any information about the true labeling, since entries corresponding to the test items were removed. Decoding is considered successful if the matching score (computed as the sum of Pearson's correlations) is higher for the congruent pair than for the incorrect one. Again, the expected performance for a model decoding at random is 0.50. Table 3 shows the performance of our semantic models, with selected results also shown in Figure 1.

The performance of all semantic models in the decoding task is significantly above chance level. Grounded semantic models (visual and multi-modal) prove once again to have a significant advantage in decoding brain activation patterns over the text-based models and association-based model ($p < 0.05$). There is no signifi-

MODEL	P1	P2	P3	P4	P5	P6	P7	P8	P9	*mean*
MITCHELL	0.79	0.76	0.74	0.8	0.78	0.66	0.69	0.62	0.74	0.73
DISTRIB	0.87	0.69	0.79	0.89	0.79	0.75	0.75	0.52	0.79	0.76
SVD-300	0.89	0.72	0.79	0.90	0.79	0.74	0.78	0.56	0.83	0.78
DEPS	0.88	0.74	0.83	0.91	**0.81**	0.68	0.76	0.58	0.83	0.78
DEPS-SVD300	0.89	0.75	0.84	0.91	**0.81**	0.67	0.77	0.57	0.83	0.78
EMBED-BOW2	0.92	0.74	0.81	0.91	0.75	0.75	0.77	0.59	0.81	0.78
EMBED-BOW5	0.91	0.73	0.83	0.91	0.76	0.73	0.79	0.55	0.80	0.78
EMBED-DEPS	0.91	0.71	0.80	**0.92**	0.75	0.71	0.79	0.62	**0.85**	0.78
ASSOC	0.91	0.72	0.81	0.91	0.73	0.69	0.75	0.62	0.79	0.77
VISUAL	**0.94**	**0.82**	**0.88**	0.90	0.78	**0.76**	**0.83**	**0.65**	0.82	0.82
VISUAL+ASSOC	**0.94**	**0.82**	**0.88**	0.90	0.79	**0.76**	**0.83**	**0.65**	0.83	0.82
VISUAL+MITCHELL	**0.94**	**0.82**	**0.88**	0.90	0.78	**0.76**	**0.83**	**0.65**	0.82	0.82

Table 2: *Similarity based encoding results*: Cross-validated accuracies for models trained on participants P1 through P9, together with mean over participants.

MODEL	P1	P2	P3	P4	P5	P6	P7	P8	P9	*mean*
MITCHELL	0.80	0.76	0.75	0.82	0.77	0.7	0.71	**0.65**	0.75	0.75
DISTRIB	0.87	0.70	0.79	0.90	**0.80**	0.76	0.77	0.58	0.80	0.77
SVD300	0.88	0.73	0.79	0.89	**0.80**	0.76	0.79	0.61	**0.85**	0.79
DEPS	0.88	0.75	0.84	0.91	**0.80**	0.70	0.78	0.61	0.84	0.79
DEPS-SVD300	0.89	0.76	0.84	0.90	0.81	0.70	0.79	0.61	**0.85**	0.79
EMBED-BOW2	0.91	0.75	0.81	0.90	0.76	0.76	0.78	0.60	0.82	0.79
EMBED-BOW5	0.91	0.74	0.83	0.91	0.77	0.75	0.80	0.58	0.82	0.79
EMBED-DEPS	0.91	0.71	0.80	**0.92**	0.75	0.71	0.79	0.62	**0.85**	0.78
ASSOC	0.90	0.73	0.79	0.90	0.73	0.69	0.76	0.63	0.81	0.77
VISUAL	**0.94**	**0.83**	**0.89**	0.90	0.79	**0.78**	**0.84**	**0.65**	0.84	0.83
VISUAL+ASSOC	**0.94**	**0.83**	**0.89**	0.90	0.79	**0.78**	**0.84**	**0.65**	0.84	0.83
VISUAL+MITCHELL	**0.94**	**0.83**	**0.89**	0.90	0.79	**0.78**	**0.84**	**0.65**	0.84	0.83

Table 3: *Similarity based decoding results*: Cross-validated accuracies for models trained on participants P1 through P9, together with mean over participants.

cant difference in performance between any of the multi-modal models and VISUAL.

6.4 Discriminating between words in the same semantic class

Following Mitchell et al. (2008), we also compare the models in their ability to make accurate predictions when the two test words are exemplars of the same semantic category[11]. This formulation of the task is more difficult, since items in the same semantic class (e.g. *dog* and *cat*) are more similar than items from different semantic classes (e.g. *eye* and *desk*).

In order to measure the performance of our models in this task, we recompute the cross-validated accuracies for all three experiments (regression-based learning, encoding and decoding) by only taking into account the performance on the 120 cross-validation folds where the test items share the same semantic class. The results across models and experiments show very similar trends as the ones computed using all 1770 cross-validation folds. The majority of the models still perform above chance level, but as expected they perform worse than when evaluated using the entire dataset. Visually-grounded models still perform the best in all three experiments (mean performance across participants for multimodal models in all three tasks is in the [0.61-0.63] range).

7 Conclusion and future work

We presented the first systematic comparison of a range of widely-used, state-of-the-art semantic models in their ability to predict patterns of conceptual representation in the human brain. Firstly, we demonstrated that visual information is a stronger predictor of brain activity than linguistic information for concrete nouns. These findings provide further support to the existing hypotheses about the interplay of linguistic, conceptual and perceptual systems in the human brain (Barsalou, 2008). These results also resonate with the success of the rapidly growing field of multimodal semantics (Kiela et al., 2016).

[11] The 60 concepts are exemplars of 12 semantic classes.

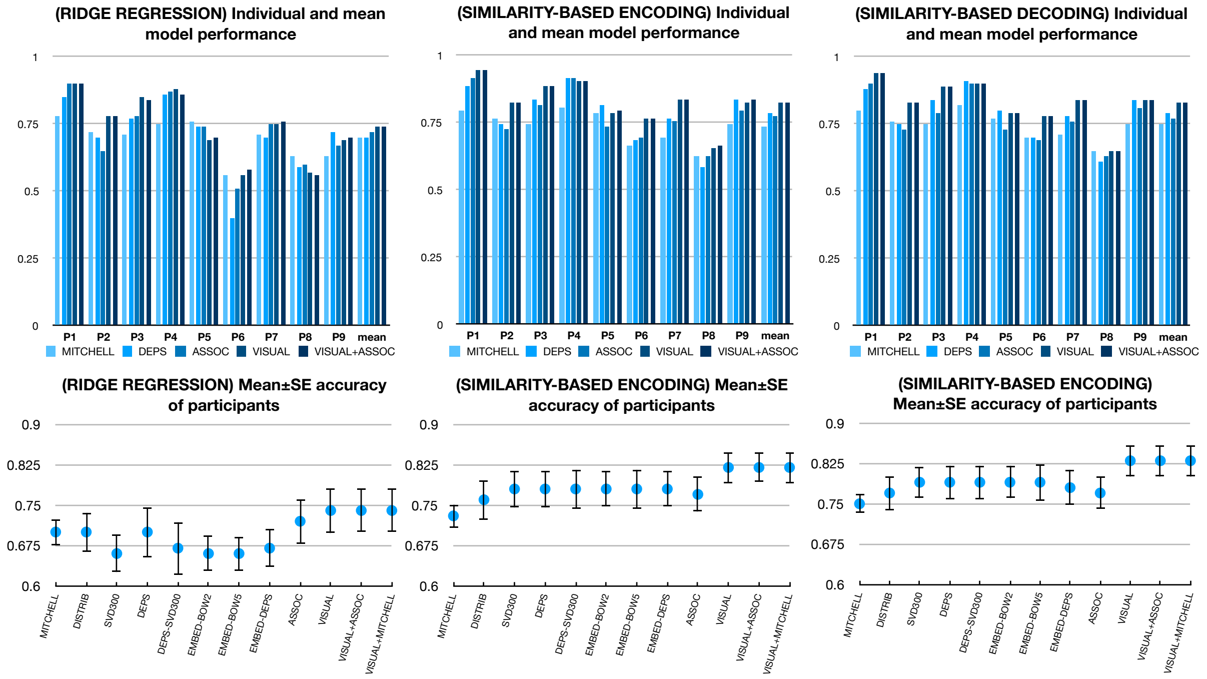


Figure 1: (*TOP*) Comparison of individual and mean model performance for five selected models (MITCHELL, DEPS, ASSOC, VISUAL, VISUAL+ASSOC), using results in Table 1 (*Ridge regression*), Table 2 (*Similarity-based encoding*) and Table 3 (*Similarity-based decoding*). (*BOTTOM*) Mean±SE accuracy of participants for all models.

Secondly, our results suggest that sparse text-based models, whether dependency-based or built using linear bag-of-words context, predict neural activity more accurately than dense models. We also show that the structure of the text-based semantic model (sparse vs. dense) has more influence on the performance than the type of information used to construct the context (linear bag-of-words vs. dependency-based).

Finally, we found that cognitively-motivated association-based models perform on par with or better than other linguistic models. These results are in line with the previous findings of behavioural research suggesting that humans represent the meanings of concepts through association with other concepts (Barsalou et al., 2008) which in turn endorses the association-based semantic models as a promising direction in computational semantics research.

An interesting avenue for future work would be to investigate the variance of results amongst individual participants (Figure 1). Previous studies that use fMRI data always report variation across participants (Devereux et al., 2010; Anderson et al., 2017) and most often attribute it to head motion. However, understanding how individual variations in participants can impact modeling decisions would be of great value to the computational semantics community.

Acknowledgments

Luana Bulat is supported by an EPSRC Doctoral Training Grant and ERC Proof of Concept Grant GroundForce (693579). Stephen Clark is supported by ERC Starting Grant DisCoTex (306920) and ERC Proof of Concept Grant GroundForce (693579). Ekaterina Shutova is supported by the Leverhulme Trust Early Career Fellowship. We thank the anonymous reviewers for their helpful comments.

References

Andrew J Anderson, Elia Bruni, Ulisse Bordignon, Massimo Poesio, and Marco Baroni. 2013. Of words, eyes and brains: Correlating image-based distributional semantic models with neural representations of concepts. In *EMNLP*, pages 1960–1970.

Andrew J Anderson, Elia Bruni, Alessandro Lopopolo, Massimo Poesio, and Marco Baroni. 2015. Reading visually embodied meaning from the brain: Visually grounded computational models decode visual-object mental imagery induced by written text. *NeuroImage*, 120:309–322.

Andrew J Anderson, Douwe Kiela, Stephen Clark, and Massimo Poesio. 2017. Visually grounded and textual semantic models differentially decode brain activity associated with concrete and abstract nouns. *Transactions of the Association for Computational Linguistics*, 5:17–30.

Andrew J Anderson, Benjamin D Zinszer, and Rajeev DS Raizada. 2016. Representational similarity encoding for fMRI: Pattern-based synthesis to predict brain activity using stimulus-model-similarities. *NeuroImage*, 128:44–53.

Lawrence W Barsalou. 2008. Grounded cognition. *Annu. Rev. Psychol.*, 59:617–645.

Lawrence W Barsalou, Ava Santos, W Kyle Simmons, and Christine D Wilson. 2008. Language and simulation in conceptual processing. *Symbols, embodiment, and meaning*, pages 245–283.

Shane Bergsma and Randy Goebel. 2011. Using visual information to predict lexical preference. In *RANLP*, pages 399–405.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Luana Bulat, Douwe Kiela, and Stephen Clark. 2016. Vision and feature norms: Improving automatic feature norm learning through cross-modal maps. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 579–588, San Diego, California. Association for Computational Linguistics.

Kai-min Kevin Chang, Tom Mitchell, and Marcel Adam Just. 2010. Quantitative modeling of the neural representation of objects: How semantic feature norms can account for fMRI activation. *Neuroimage: Special Issue on Multi-variate Deciding and Brain Reading*, 56:716–727.

Stephen Clark and James R Curran. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics*, 33(4):493–552.

George S Cree and Ken McRae. 2003. Analyzing the factors underlying the structure and computation of the meaning of chipmunk, cherry, chisel, cheese, and cello (and many other such concrete nouns). *Journal of Experimental Psychology: General*, 132(2):163.

Simon De Deyne, Daniel J Navarro, and Gert Storms. 2013. Better explanations of lexical and semantic cognition using networks derived from continued rather than single-word associations. *Behavior Research Methods*, 45(2):480–498.

Simon De Deyne, Amy Perfors, and Daniel J Navarro. 2016. Predicting human similarity judgments with distributional models: The value of word associations. *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391.

Barry Devereux, Colin Kelly, and Anna Korhonen. 2010. Using fMRI activation to conceptual stimuli to evaluate methods for extracting conceptual representations from corpora. In *Proceedings of the NAACL HLT 2010 First Workshop on Computational Neurolinguistics*, pages 70–78. Association for Computational Linguistics.

Leonardo Fernandino, Colin J Humphries, Mark S Seidenberg, William L Gross, Lisa L Conant, and Jeffrey R Binder. 2015. Predicting brain activation patterns associated with individual lexical concepts based on five sensory-motor attributes. *Neuropsychologia*, 76:17–26.

Felix Hill and Anna Korhonen. 2014. Learning abstract concept embeddings from multi-modal data: Since you probably can't see what I mean. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Arthur E Hoerl and Robert W Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM.

Douwe Kiela. 2016. MMFEAT: A toolkit for extracting multi-modal features. In *Proceedings of ACL*.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of EMNLP*, volume 2014.

Douwe Kiela, Luana Bulat, and Stephen Clark. 2015a. Grounding semantics in olfactory perception. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 231–236, Beijing, China. Association for Computational Linguistics.

Douwe Kiela and Stephen Clark. 2015. Multi- and cross-modal semantics beyond vision: Grounding in auditory perception. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2461–2470, Lisbon, Portugal. Association for Computational Linguistics.

Douwe Kiela, Laura Rimell, Ivan Vulić, and Stephen Clark. 2015b. Exploiting image generality for lexical entailment detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Beijing, China.

Douwe Kiela, Anita L. Verő, and Stephen Clark. 2016. Comparing Data Sources and Architectures for Deep Visual Representation Learning in Semantics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-16)*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *ACL (2)*, pages 302–308. Citeseer.

Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior research methods*, 37(4):547–559.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of english. *Natural Language Engineering*, 7(03):207–223.

Tom M Mitchell, Svetlana V Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L Malave, Robert A Mason, and Marcel Adam Just. 2008. Predicting human brain activity associated with the meanings of nouns. *Science*, 320(5880):1191–1195.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Selecting corpus-semantic models for neurolinguistic decoding. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 114–123. Association for Computational Linguistics.

Douglas L Nelson, Cathy L McEvoy, and Thomas A Schreiber. 2004. The University of South Florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers*, 36(3):402–407.

Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. 2009. Zero-shot learning with semantic output codes. In *Advances in Neural Information Processing Systems*, pages 1410–1418.

Francisco Pereira, Matthew Botvinick, and Greg Detre. 2013. Using wikipedia to learn semantic feature representations of concrete concepts in neuroimaging experiments. *Artificial intelligence*, 194:240–252.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black holes and white rabbits: Metaphor identification with visual features. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 160–170.

Carina Silberer and Mirella Lapata. 2012. Grounded models of semantic representation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1423–1433. Association for Computational Linguistics.

Mark Steyvers, Richard M Shiffrin, and Douglas L Nelson. 2004. Word association spaces for predicting semantic similarity effects in episodic memory. *Experimental cognitive psychology and its applications: Festschrift in honor of Lyle Bourne, Walter Kintsch, and Thomas Landauer*, pages 237–249.

Multi-modal Summarization for Asynchronous Collection of Text, Image, Audio and Video

Haoran Li[1,2], **Junnan Zhu**[1,2], **Cong Ma**[1,2], **Jiajun Zhang**[1,2] and **Chengqing Zong**[1,2,3]
[1] National Laboratory of Pattern Recognition, CASIA, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] CAS Center for Excellence in Brain Science and Intelligence Technology, Shanghai, China
{haoran.li, junnan.zhu, cong.ma, jjzhang, cqzong}@nlpr.ia.ac.cn

Abstract

The rapid increase in multimedia data transmission over the Internet necessitates the multi-modal summarization (MMS) from collections of text, image, audio and video. In this work, we propose an extractive multi-modal summarization method that can automatically generate a textual summary given a set of documents, images, audios and videos related to a specific topic. The key idea is to bridge the semantic gaps between multi-modal content. For audio information, we design an approach to selectively use its transcription. For visual information, we learn the joint representations of text and images using a neural network. Finally, all of the multi-modal aspects are considered to generate the textual summary by maximizing the salience, non-redundancy, readability and coverage through the budgeted optimization of submodular functions. We further introduce an MMS corpus in English and Chinese, which is released to the public[1]. The experimental results obtained on this dataset demonstrate that our method outperforms other competitive baseline methods.

1 Introduction

Multimedia data (including text, image, audio and video) have increased dramatically recently, which makes it difficult for users to obtain important information efficiently. Multi-modal summarization (MMS) can provide users with textual summaries that can help acquire the gist of multimedia data in a short time, without reading documents or watching videos from beginning to end.

The existing applications related to MMS include meeting record summarization (Erol et al., 2003; Gross et al., 2000), sport video summarization (Tjondronegoro et al., 2011; Hasan et al., 2013), movie summarization (Evangelopoulos et al., 2013; Mademlis et al., 2016), pictorial storyline summarization (Wang et al., 2012), timeline summarization (Wang et al., 2016b) and social multimedia summarization (Del Fabro et al., 2012; Bian et al., 2013; Schinas et al., 2015; Bian et al., 2015; Shah et al., 2015, 2016). When summarizing meeting recordings, sport videos and movies, such videos consist of synchronized voice, visual and captions. For the summarization of pictorial storylines, the input is a set of images with text descriptions. None of these applications focus on summarizing multimedia data that contain asynchronous information about general topics.

In this paper, as shown in Figure 1, we propose an approach to a generate textual summary from a set of asynchronous documents, images, audios and videos on the same topic.

Since multimedia data are heterogeneous and contain more complex information than pure text does, MMS faces a great challenge in addressing the semantic gap between different modalities. The framework of our method is shown in Figure 1. For the audio information contained in videos, we obtain speech transcriptions through Automatic Speech Recognition (ASR) and design a method to use these transcriptions selectively. For visual information, including the key-frames extracted from videos and the images that appear in documents, we learn the joint representations of texts and images by using a neural network; we then can identify the text that is relevant to the image. In this way, audio and visual information can be integrated into a textual summary.

Traditional document summarization involves two essential aspects: (1) Salience: the summa-

[1] http://www.nlpr.ia.ac.cn/cip/jjzhang.htm

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1092–1102
Copenhagen, Denmark, September 7–11, 2017.

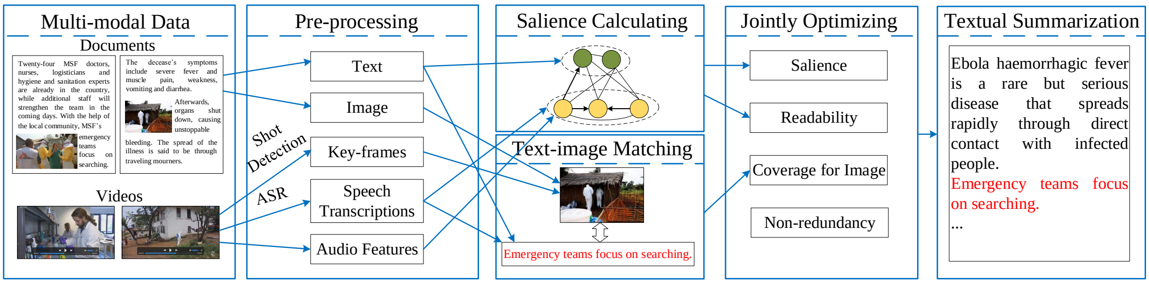

Multi-modal Data
Documents
Twenty-four MSF doctors, nurses, logisticians and hygiene and sanitation experts are already in the country, while additional staff will strengthen the team in the coming days. With the help of the local community, MSF's emergency teams focus on searching.
The decease's symptoms include severe fever and muscle pain, weakness, vomiting and diarrhea. Afterwards, organs shut down, causing unstoppable bleeding. The spread of the illness is said to be through traveling mourners.
Videos
Pre-processing
Shot Detection
ASR
Text
Image
Key-frames
Speech Transcriptions
Audio Features
Salience Calculating
Text-image Matching
Emergency teams focus on searching.
Jointly Optimizing
Salience
Readability
Coverage for Image
Non-redundancy
Textual Summarization
Ebola haemorrhagic fever is a rare but serious disease that spreads rapidly through direct contact with infected people.
Emergency teams focus on searching.
...

Figure 1: The framework of our MMS model.

ry should retain significant content of the input documents. (2) Non-redundancy: the summary should contain as little redundant content as possible. For MMS, we consider two additional aspects: (3) Readability: because speech transcriptions are occasionally ill-formed, we should try to get rid of the errors introduced by ASR. For example, when a transcription provides similar information to a sentence in documents, we should prefer the sentence to the transcription presented in the summary. (4) Coverage for the visual information: images that appear in documents and videos often capture event highlights that are usually very important. Thus, the summary should cover as much of the important visual information as possible. All of the aspects can be jointly optimized by the budgeted maximization of submodular functions (Khuller et al., 1999).

Our main contributions are as follows:

- We design an MMS method that can automatically generate a textual summary from a set of asynchronous documents, images, audios and videos related to a specific topic.

- To select the representative sentences, we consider four criteria that are jointly optimized by the budgeted maximization of submodular functions.

- We introduce an MMS corpus in English and Chinese. The experimental results on this dataset demonstrate that our system can take advantage of multi-modal information and outperforms other baseline methods.

2 Related Work

2.1 Multi-document Summarization

Multi-document summarization (MDS) attempts to extract important information for a set of documents related to a topic to generate a short summary. Graph based methods (Mihalcea and Tarau, 2004; Wan and Yang, 2006; Zhang et al., 2016) are commonly used. LexRank (Erkan and Radev, 2011) first builds a graph of the documents, in which each node represents a sentence and the edges represent the relationship between sentences. Then, the importance of each sentence is computed through an iterative random walk.

2.2 Multi-modal Summarization

In recent years, much work has been done to summarize meeting recordings, sport videos, movies, pictorial storylines and social multimedia.

Erol et al. (2003) aim to create important segments of a meeting recording based on audio, text and visual activity analysis. Tjondronegoro et al. (2011) propose a way to summarize a sporting event by analyzing the textual information extracted from multiple resources and identifying the important content in a sport video. Evangelopoulos et al. (2013) use an attention mechanism to detect salient events in a movie. Wang et al. (2012) and Wang et al. (2016b) use image-text pairs to generate a pictorial storyline and timeline summarization. Li et al. (2016) develop an approach for multimedia news summarization for searching results on the Internet, in which the hLDA model is introduced to discover the topic structure of the news documents. Then, a news article and an image are chosen to represent each topic. For social media summarization, Fabro et al. (2012) and Schinas et al. (2015) propose to summarize the real-life events based on multimedia content such as photos from Flickr and videos from YouTube. Bian et al. (2013; 2015) propose a multimodal LDA to detect topics by capturing the correlations between textual and visual features of microblogs with embedded images. The output of their method is a set of representative images that describe the events. Shah et al. (2015; 2016) introduce EventBuilder which produces text summaries for a social event leveraging Wikipedia and visualizes the event with social media activities.

Most of the above studies focus on synchronous multi-modal content, i.e., in which images are paired with text descriptions and videos are paired with subtitles. In contrast, we perform summarization from asynchronous (i.e., there is no given description for images and no subtitle for videos) multi-modal information about news topics, including multiple documents, images and videos, to generate a fixed length textual summary. This task is both more general and more challenging.

3 Our Model

3.1 Problem Formulation

The input is a collection of multi-modal data $\mathcal{M} = \{D_1, ..., D_{|D|}, V_1, ..., V_{|V|}\}$ related to a news topic $\mathcal{T}$, where each document $D_i = \{T_i, I_i\}$ consists of text T_i and image I_i (there may be no image for some documents). V_i denotes video. $|\cdot|$ denotes the cardinality of a set. The objective of our work is to automatically generate textual summary to represent the principle content of $\mathcal{M}$.

3.2 Model Overview

There are many essential aspects in generating a good textual summary for multi-modal data. The salient content in documents should be retained, and the key facts in videos and images should be covered. Further, the summary should be readable and non-redundant and should follow the fixed length constraint. We propose an extraction-based method in which all these aspects can be jointly optimized by the budgeted maximization of submodular functions defined as follows:

$$\max_{S \subseteq T} \{\mathcal{F}(S) : \sum_{s \in S} l_s \leq \mathcal{L}\} \tag{1}$$

where T is the set of sentences, S is the summary, l_s is length (number of words) of sentence s, $\mathcal{L}$ is budget, i.e., length constraint for the summary, and submodular function $\mathcal{F}(S)$ is the summary score related to the above-mentioned aspects.

Text is the main modality of documents, and in some cases, images are embedded in documents. Videos consist of at least two types of modalities: audio and visual. Next, we give overall processing methods for different modalities.

Audio, i.e., speech, can be automatically transcribed into text by using an ASR system[2]. Then, we can leverage a graph-based method to calculate the salience score for all of the speech transcriptions and for the original sentences in documents. Note that speech transcriptions are often ill-formed; thus, to improve the readability, we should try to avoid the errors introduced by ASR. In addition, audio features including acoustic confidence (Valenza et al., 1999), audio power (Christel et al., 1998) and audio magnitude (Dagtas and Abdel-Mottaleb, 2001) have proved to be helpful for speech and video summarization which will benefit our method.

For visual, which is actually a sequence of images (frames), because most of the neighboring frames contain redundant information, we first extract the most meaningful frames, i.e., the keyframes, which can provide the key facts for the whole video. Then, it is necessary to perform semantic analysis between text and visual. To this end, we learn the joint representations for textual and visual modalities and can then identify the sentence that is relevant to the image. In this way, we can guarantee the coverage of generated summary for the visual information.

3.3 Salience for Text

We apply a graph-based LexRank algorithm (Erkan and Radev, 2011) to calculate salience score of the text unit, including the sentences in documents and the speech transcriptions from videos. LexRank first constructs a graph based on the text units and their relationship and then conducts an iteratively random walk to calculate the salience score of the text unit, $sa(t_i)$, until convergence using the following equation:

$$Sa(t_i) = \mu \sum_j Sa(t_j) \cdot M_{ji} + \frac{1-\mu}{N} \tag{2}$$

where μ is the damping factor that is set to 0.85. N is the total number of the text units. M_{ji} is the relationship between text unit t_i and t_j, which is computed as follows:

$$M_{ji} = sim(t_j, t_i) \tag{3}$$

The text unit t_i is represented by averaging the embeddings of the words (except stop-words) in t_i. $sim(\cdot)$ denotes cosine similarity between two texts (negative similarities are replaced with 0).

[2] We use IBM Watson Speech to Text service: www.ibm.com/watson/developercloud/speech-to-text.html

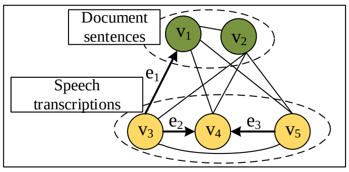


Figure 2: LexRank with guidance strategies. e_1 is guided because speech transcription v_3 is related to document sentence v_1; e_2 and e_3 are guided because of audio features. Other edges without arrow are bidirectional.

For MMS task, we propose two guidance strategies to amend the affinity matrix M and calculate salience score of the text as shown in Figure 2.

3.3.1 Readability Guidance Strategies

The random walk process can be understood as a recommendation: M_{ji} in Equation 2 denotes that t_j will recommend t_i to the degree of M_{ji}. The affinity matrix M in the LexRank model is symmetric, which means $M_{ij} = M_{ji}$. In contrast, for MMS, considering the unsatisfactory quality of speech recognition, symmetric affinity matrices are inappropriate. Specifically, to improve the readability, for a speech transcription, if there is a sentence in document that is related to this transcription, we would prefer to assign the text sentence a higher salience score than that assigned to the transcribed one. To this end, the process of a random walk should be guided to control the recommendation direction: when a document sentence is related to a speech transcription, the symmetric weighted edge between them should be transformed into a unidirectional edge, in which we invalidate the direction from document sentence to the transcribed one. In this way, speech transcriptions will not be recommended by the corresponding document sentences. Important speech transcriptions that cannot be covered by documents still have the chance to obtain high salience scores. For the pair of a sentence t_i and a speech transcription t_j, M_{ij} is computed as follows:

$$M_{ij} = \begin{cases} 0, & \text{if } sim(t_i, t_j) > T_{text} \\ sim(t_i, t_j), & \text{otherwise} \end{cases} \tag{4}$$

where threshold T_{text} is used to determine whether a sentence is related to others. We obtain the proper semantic similarity threshold by testing on Microsoft Research Paraphrase (MSRParaphrase) dataset (Quirk et al., 2004). It is a publicly available paraphrase corpus that consists of 5801 pairs of sentences, of which 3900 pairs are semantically equivalent.

3.3.2 Audio Guidance Strategies

Some audio features can guide the summarization system to select more important and readable speech transcriptions. Valenza et al. (1999) use acoustic confidence to obtain accurate and readable summaries of broadcast news programs. Christel et al. (1998) and Dagtas and Abdel-Mottaleb (2001) apply audio power and audio magnitude to find significant audio events. In our work, we first balance these three feature scores for each speech transcription by dividing their respective maximum values among the whole amount of audio, and we then average these scores to obtain the final audio score for speech transcription. For each adjacent speech transcription pair $(t_k, t_{k'})$, if the audio score $a(t_k)$ for t_k is smaller than a certain threshold while $a(t_{k'})$ is greater, which means that $t_{k'}$ is more important and readable than t_k, then t_k should recommend $t_{k'}$, but $t_{k'}$ should not recommend t_k. We formulate it as follows:

$$\begin{cases} M_{kk'} = sim(t_k, t_{k'}) \\ M_{k'k} = 0 \end{cases} \\ \text{if } a(t_k) < T_{audio} \text{ and } a(t_{k'}) > T_{audio} \tag{5}$$

where the threshold T_{audio} is the average audio score for all the transcriptions in the audio.

Finally, affinity matrices are normalized so that each row adds up to 1.

3.4 Text-Image Matching

The key-frames contained in videos and the images embedded in documents often captures news highlights in which the important ones should be covered by the textual summary. Before measuring the coverage for images, we should train the model to bridge the gap between text and image, i.e., to match the text and image.

We start by extracting key-frames of videos based on shot boundary detection. A shot is defined as an unbroken sequence of frames. The abrupt transition of RGB histogram features often indicates shot boundaries (Zhuang et al., 1998). Specifically, when the transition of the RGB histogram feature for adjacent frames is greater than a certain ratio[3] of the average transition for the whole video, we segment the shot. Then, the frames

[3] The ratio is determined by testing on the

in the middle of each shot are extracted as key-frames. These key-frames and images in documents make up the image set that the summary should cover.

Next, it is necessary to perform a semantic analysis between the text and the image. To this end, we learn the joint representations for textual and visual modalities by using a model trained on the Flickr30K dataset (Young et al., 2014), which contains 31,783 photographs of everyday activities, events and scenes harvested from Flickr. Each photograph is manually labeled with 5 textual descriptions. We apply the framework of Wang et al. (2016a), which achieves state-of-the-art performance for text-image matching task on the Flickr30K dataset. The image is encoded by the VGG model (Simonyan and Zisserman, 2014) that has been trained on the ImageNet classification task following the standard procedure (Wang et al., 2016a). The 4096-dimensional feature from the pre-softmax layer is used to represent the image. The text is first encoded by the Hybrid Gaussian-Laplacian mixture model (HGLMM) using the method of Klein et al. (2014). Then, the HGLMM vectors are reduced to 6000 dimensions through PCA. Next, the sentence vector v_s and image vector v_i are mapped to a joint space by a two-branch neural network as follows:

$$\begin{cases} x = W_2 \cdot f(W_1 \cdot v_s + b_s) \\ y = V_2 \cdot f(V_1 \cdot v_i + b_i) \end{cases} \tag{6}$$

where $W_1 \in \mathbb{R}^{2048\times 6000}$, $b_s \in \mathbb{R}^{2048}$, $W_2 \in \mathbb{R}^{512\times 2048}$, $V_1 \in \mathbb{R}^{2048\times 4096}$, $b_i \in \mathbb{R}^{2048}$, $V_2 \in \mathbb{R}^{512\times 2048}$, f is Rectified Linear Unit (ReLU).

The max-margin learning framework is applied to optimize the neural network as follows:

$$\begin{aligned} L = &\sum_{i,k} \max[0, m + s(x_i, y_i) - s(x_i, y_k)] \\ &+ \lambda_1 \sum_{i,k} \max[0, m + s(x_i, y_i) - s(x_k, y_i)] \end{aligned} \tag{7}$$

where for positive text-image pair (x_i, y_i), the top K most violated negative pairs (x_i, y_k) and (x_k, y_i) in each mini-batch are sampled. The objective function L favors higher matching score $s(x_i, y_i)$ (cosine similarity) for positive text-image pairs than for negative pairs[4].

Note that the images in Flickr30K are similar to our task. However, the image descriptions are much simpler than the text in news, so the model trained on Flickr30K cannot be directly used for our task. For example, some of the information contained in the news, such as the time and location of events, cannot be directly reflected by images. To solve this problem, we simplify each sentence and speech transcription based on semantic role labelling (Gildea and Jurafsky, 2002), in which each predicate indicates an event and the arguments express the relevant information of this event. ARG0 denotes the agent of the event, and ARG1 denotes the action. The assumption is that the concepts including agent, predicate and action compose the body of the event, so we extract "ARG0+predicate+ARG1" as the simplified sentence that is used to match the images. It is worth noting that there may be multiple predicate-argument structures for one sentence and we extract all of them.

After the text-image matching model is trained and the sentences are simplified, for each text-image pair (T_i, I_j) in our task, we can identify the matched pairs if the score $s(T_i, I_j)$ is greater than a threshold T_{match}. We set the threshold as the average matching score for the positive text-image pair in Flickr30K, although the matching performance for our task could in principle be improved by adjusting this parameter.

3.5 Multi-modal Summarization

We model the salience of a summary S as the sum of salience scores $Sa(t_i)$[5] of the sentence t_i in the summary, combining a λ-weighted redundancy penalty term:

$$\mathcal{F}_s(S) = \sum_{t_i \in S} Sa(t_i) - \frac{\lambda_s}{|S|} \sum_{t_i, t_j \in S} sim(t_i, t_j) \tag{8}$$

We model the summary S coverage for the image set I as the weighted sum of image covered by the summary:

$$\mathcal{F}_c(S) = \sum_{p_i \in I} Im(p_i) b_i \tag{9}$$

where the weight $Im(p_i)$ for the image p_i is the length ratio between the shot p_i and the whole videos. b_i is a binary variable to indicate

shot detection dataset of TRECVID. http://www-nlpir.nist.gov/projects/trecvid/

[4] In the experiments, $K = 50$, $m = 0.1$ and $\lambda_1 = 2$. Wang et al. (2016a) also proved that structure-preserving constraints can make 1% Recall@1 improvement.

[5] Normalized by the maximum value among all the sentences.

whether an image p_i is covered by the summary, i.e., whether there is at least one sentence in the summary matching the image.

Finally, considering all the modalities, the objective function is defined as follows:

$$\mathcal{F}_m(S) = \frac{1}{M_s}\sum_{t_i \in S} Sa(t_i) + \frac{1}{M_c}\sum_{p_i \in I} Im(p_i)b_i - \frac{\lambda_m}{|S|}\sum_{i,j \in S} sim(t_i, t_j) \tag{10}$$

where M_s is the summary score obtained by Equation 8 and M_c is the summary score obtained by Equation 9. The aim of M_s and M_c is to balance the aspects of salience and coverage for images. λ_s, and λ_m are determined by testing on development set. Note that to guaranteed monotone of $\mathcal{F}$, λ_s, and λ_m should be lower than the minimum salience score of sentences. To further improve non-redundancy, we make sure that similarity between any pair of sentences in the summary is lower than T_{text}.

Equations 8,9 and 10 are all monotone submodular functions under the budget constraint. Thus, we apply the greedy algorithm (Lin and Bilmes, 2010) guaranteeing near-optimization to solve the problem.

4 Experiment

4.1 Dataset

There is no benchmark dataset for MMS. We construct a dataset as follows. We select 50 news topics in the most recent five years, 25 in English and 25 in Chinese. We set 5 topics for each language as a development set. For each topic, we collect 20 documents within the same period using Google News search[6] and 5-10 videos in CCTV.com[7] and Youtube[8]. More details of the corpus are illustrated in Table 1. Some examples of news topics are provided Table 2.

We employ 10 graduate students to write reference summaries after reading documents and watching videos on the same topic. We keep 3 reference summaries for each topic. The criteria for summarizing documents lie in: (1) retaining important content of the input documents and videos; (2) avoiding redundant information; (3) having a good readability; (4) following the length limit. We set the length constraint for each English and Chinese summary to 300 words and 500 characters, respectively.

	#Sentence	#Word	#Shot	Video Length
English	492.1	12,104.7	47.2	197s
Chinese	402.1	9,689.3	49.3	207s

Table 1: Corpus statistics.

English	(1) Nepal earthquake (2) Terror attack in Paris (3) Train derailment in India (4) Germanwings crash (5) Refugee crisis in Europe
Chinese	(6) “东方之星”客船翻沉 (“Oriental Star”passenger ship sinking) (7) 银川公交大火 (The bus fire in Yinchuan) (8) 香港占中 (Occupy Central in HONG KONG) (9) 李娜澳网夺冠 (Li Na wins Australian Open) (10) 抗议“萨德”反导系统 (Protest against “THAAD”anti-missile system)

Table 2: Examples of news topics.

4.2 Comparative Methods

Several models are compared in our experiments, including generating summaries with different modalities and different approaches to leverage images.

Text only. This model generates summaries only using the text in documents.

Text + audio. This model generates summaries using the text in documents and the speech transcriptions but without guidance strategies.

Text + audio + guide. This model generates summaries using the text in documents and the speech transcriptions with guidance strategies.

The following models generate summaries using both documents and videos but take advantage of images in different ways. The salience scores for text are obtained with guidance strategies.

Image caption. The image is first captioned using the model of Vinyals et al. (2016) which achieved first place in the 2015 MSCOCO Image Captioning Challenge. This model generates summaries using text in documents, speech transcription and image captions.

Note that the above-mentioned methods generate summaries by using Equation 8 and the follow-

[6] http://news.google.com/

[7] http://www.cctv.com/

[8] https://www.youtube.com/

ing methods using Equation 8 ,9 and 10.

Image caption match. This model uses generated image captions to match the text; i.e., if the similarity between a generated image caption and a sentence exceeds the threshold T_{text}, the image and the sentence match.

Image alignment. The images are aligned to the text in the following ways: The images in a document are aligned to all the sentences in this document and the key-frames in a shot are aligned to all the speech transcriptions in this shot.

Image match. The texts are matched with images using the approach introduced in Section 3.4.

4.3 Implementation Details

We perform sentence[9] and word tokenization, and all the Chinese sentences are segmented by Stanford Chinese Word Segmenter (Tseng et al., 2005). We apply Stanford CoreNLP toolkit (Levy and D. Manning, 2003; Klein and D. Manning, 2003) to perform lexical parsing and use semantic role labelling approach proposed by Yang and Zong (2014). We use 300-dimension skip-gram English word embeddings which are publicly available[10]. Given that text-image matching model and image caption generation model are trained in English, to create summaries in Chinese, we first translate the Chinese text into English via Google Translation[11] and then conduct text and image matching.

4.4 Multi-modal Summarization Evaluation

We use the ROUGE-1.5.5 toolkit (Lin and Hovy, 2003) to evaluate the output summaries. This evaluation metric measures the summary quality by matching n-grams between generated summary and reference summary. Table 3 and Table 4 show the averaged ROUGE-1 (R-1), ROUGE-2 (R-2) and ROUGE-SU4 (R-SU4) F-scores regarding to the three reference summaries for each topic in English and Chinese.

For the results of the English MMS, from the first three lines in Table 3 we can see that when summarizing without visual information, the method with guidance strategies performs slightly better than do the first two methods. Because Rouge mainly measures word overlaps, manual evaluation is needed to confirm the impact of guidance strategies on improving readability. It is introduced in Section 4.5. The rating ranges from 1 (the poorest) to 5 (the best). When summarizing with textual and visual modalities, performances are not always improved, which indicates that the models of **image caption**, **image caption match** and **image alignment** are not suitable to MMS. The **image match** model has a significant advantage over other comparative methods, which illustrates that it can make use of multi-modal information.

Method	R-1	R-2	R-SU4
Text only	0.422	0.114	0.166
Text + audio	0.422	0.109	0.164
Text + audio + guide	0.440	0.117	0.171
Image caption	0.435	0.111	0.167
Image caption match	0.429	0.115	0.166
Image alignment	0.409	0.082	0.082
Image match	**0.442**	**0.133**	**0.187**

Table 3: Experimental results (F-score) for English MMS.

Method	R-1	R-2	R-SU4
Text only	0.409	0.113	0.167
Text + audio	0.407	0.111	0.166
Text + audio + guide	0.411	0.115	**0.173**
Image caption match	0.381	0.092	0.149
Image alignment	0.368	0.096	0.143
Image match	**0.414**	**0.125**	**0.173**

Table 4: Experimental results (F-score) for Chinese MMS.

Table 4 shows the Chinese MMS results, which are similar to the English results that the **image match** model achieves the best performance. We find that the performance enhancement for the **image match** model is smaller in Chinese than it is in English, which may be due to the errors introduced by machine translation.

We provides a generated summary in English using the **image match** model, which is shown in Figure 3.

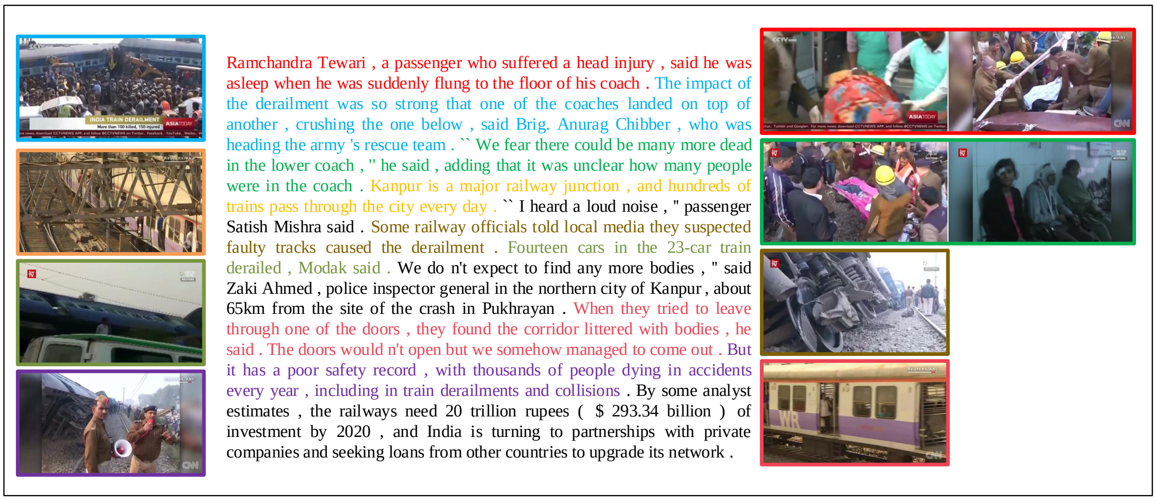
4.5 Manual Summary Quality Evaluation

The readability and informativeness for summaries are difficult to evaluate formally. We ask five graduate students to measure the quality of summaries generated by different methods. We calculate the average score for all of the topics, and the results are displayed in Table 5. Overall, our method with guidance strategies achieves higher scores than do the other methods, but it is still obviously poorer than the reference sum-

[9] We exclude sentences containing less than 5 words.

[10] https://code.google.com/archive/p/word2vec/

[11] https://translate.google.com



Ramchandra Tewari , a passenger who suffered a head injury , said he was asleep when he was suddenly flung to the floor of his coach . The impact of the derailment was so strong that one of the coaches landed on top of another , crushing the one below , said Brig. Anurag Chibber , who was heading the army 's rescue team . `` We fear there could be many more dead in the lower coach , '' he said , adding that it was unclear how many people were in the coach . Kanpur is a major railway junction , and hundreds of trains pass through the city every day . `` I heard a loud noise , '' passenger Satish Mishra said . Some railway officials told local media they suspected faulty tracks caused the derailment . Fourteen cars in the 23-car train derailed , Modak said . We do n't expect to find any more bodies , '' said Zaki Ahmed , police inspector general in the northern city of Kanpur , about 65km from the site of the crash in Pukhrayan . When they tried to leave through one of the doors , they found the corridor littered with bodies , he said . The doors would n't open but we somehow managed to come out . But it has a poor safety record , with thousands of people dying in accidents every year , including in train derailments and collisions . By some analyst estimates , the railways need 20 trillion rupees ($ 293.34 billion) of investment by 2020 , and India is turning to partnerships with private companies and seeking loans from other countries to upgrade its network .

Figure 3: An example of generated summary for the news topic "India train derailment". The sentences covering the images are labeled by the corresponding colors. The text can be partly related to the image because we use simplified sentence based on SRL to match the images. We can find some mismatched sentences, such as the sentence "Fourteen cars in the 23-car train derailed , Modak said ." where our text-image matching model may misunderstand the "car " as a "motor vehicle" but not a "coach".

maries. Specifically, when speech transcriptions are not considered, the informativeness of the summary is the worst. However, adding speech transcriptions without guidance strategies decreases readability to a large extent, which indicates that guidance strategies are necessary for MMS. The **image match** model achieves higher informativeness scores than do the other methods without using images.

We give two instances of readability guidance that arise between document text (DT) and speech transcriptions (ST) in Table 6. The errors introduced by ASR include segmentation (instance A) and recognition (instance B) mistakes.

	Method	Read	Inform
English	Text only	3.72	3.28
	Text + audio	3.08	3.44
	Text + audio + guide	3.68	3.64
	Image match	3.67	3.83
	Reference	4.52	4.36
Chinese	Text only	3.64	3.40
	Text + audio	3.16	3.48
	Text + audio + guide	3.60	3.72
	Image match	3.62	3.92
	Reference	4.88	4.84

Table 5: Manual summary quality evaluation. "Read" denotes "Readability" and "Inform" denotes "informativeness".

A	DT	There were 12 bodies at least pulled from the rubble in the square.
	ST	Still being pulled from the rubble.
	CST	Many people are still being pulled from the rubble.
B	DT	Conflict between police and protesters lit up on Tuesday.
	ST	Late night tensions between police and protesters briefly lit up this Baltimore neighborhood Tuesday.
	CST	Late-night tensions between police and protesters briefly lit up in a Baltimore neighborhood Tuesday.

Table 6: Guidance examples. "CST" denotes manually modified correct ST. ASR errors are marked red and revisions are marked blue.

4.6 How Much is the Image Worth

Text-image matching is the toughest module for our framework. Although we use a state-of-the-art approach to match the text and images, the performance is far from satisfactory. To find a somewhat strong upper-bound of the task, we choose five topics for each language to manually label the text-image matching pairs. The MMS results on these topics are shown in Table 7 and Table 8. The experiments show that with the ground truth text-image matching result, the summary quality can be promoted to a considerable extent, which indicates visual information is crucial for MMS.

An image and the corresponding texts obtained using different methods are given in Figure 4 an d Figure 5. We can conclude that the **image caption**

Image caption:
A group of people standing on top of a lush green field.
Image caption match:
We could barely stay standing.
Image hard alignment:
The need for doctors would grow as more survivors were pulled from the rubble.
Image match:
The search, involving US, Indian and Nepali military choppers and a battalion of 400 Nepali soldiers, has been joined by two MV-22B Osprey.
Image manually match:
The military helicopter was on an aid mission in Dolakha district near Tibet.

Figure 4: An example image with corresponding English texts that different methods obtain.

and the **image caption match** contain little of the image's intrinsically intended information. The **image alignment** introduces more noise because it is possible that the whole text in documents or the speech transcriptions in shot are aligned to the document images or the key-frames, respectively. The **image match** can obtain similar results to the **image manually match**, which illustrates that the **image match** can make use of visual information to generate summaries.

Method	R-1	R-2	R-SU4
Text + audio + guide	0.426	0.105	0.167
Image caption	0.423	0.106	0.167
Image caption match	0.400	0.086	0.149
Image alignment	0.399	0.069	0.136
Image match	0.436	0.126	0.177
Image manually match	**0.446**	**0.150**	**0.207**

Table 7: Experimental results (F-score) for English MMS on five topics with manually labeled text-image pairs.

Method	R-1	R-2	R-SU4
Text + audio + guide	0.417	0.115	0.171
Image caption match	0.396	0.095	0.152
Image alignment	0.306	0.072	0.111
Image match	0.401	0.127	0.179
Image manually match	**0.419**	**0.162**	**0.208**

Table 8: Experimental results (F-score) for Chinese MMS on five topics with manually labeled text-image pairs.

Image caption match:
就 星 州 民众 举行 抗议 集会， 文尚 均 表示， 国防部 愿意 与 当地 居民 沟通 。
(On behalf of the protest rally of people in Seongju, Moon Sang-gyun said that the Ministry of National Defense is willing to communicate with local residents.)
Image hard alignment:
朴槿惠 在 国家 安全 保障 会议 上 呼吁 民众 支持 “ 萨德 ” 部署 。
(Park Geun-hye called on people to support the "THAAD" deployment in the National Security Council.)
Image match:
从 7月 12日 开始， 当地 民众 连续 数日 在 星 州郡 厅 门口 请愿 。
(The local people petitioned in front of the Seongju County Office for days from July 12.)
Image manually match:
当天， 星 州郡 数千 民众 集会， 抗议 在 当地 部署 “ 萨德 ”
(On that day, thousands of people gathered in Seongju to protest the local deployment of "THAAD".)

Figure 5: An example image with corresponding Chinese texts that different methods obtain.

5 Conclusion

This paper addresses an asynchronous MMS task, namely, how to use related text, audio and video information to generate a textual summary. We formulate the MMS task as an optimization problem with a budgeted maximization of submodular functions. To selectively use the transcription of audio, guidance strategies are designed using the graph model to effectively calculate the salience score for each text unit, leading to more readable and informative summaries. We investigate various approaches to identify the relevance between the image and texts, and find that the **image match** model performs best. The final experimental results obtained using our MMS corpus in both English and Chinese demonstrate that our system can benefit from multi-modal information.

Adding audio and video does not seem to improve dramatically over text only model, which indicates that better models are needed to capture the interactions between text and other modalities, especially for visual. We also plan to enlarge our MMS dataset, specifically to collect more videos.

Acknowledgments

The research work has been supported by the Natural Science Foundation of China under Grant No. 61333018 and No. 61403379.

References

Jingwen Bian, Yang Yang, and Tat-Seng Chua. 2013. Multimedia summarization for trending topics in microblogs. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1807–1812. ACM.

Jingwen Bian, Yang Yang, Hanwang Zhang, and Tat-Seng Chua. 2015. Multimedia summarization for social events in microblog stream. *IEEE Transactions on Multimedia*, 17(2):216–228.

Michael G Christel, Michael A Smith, C Roy Taylor, and David B Winkler. 1998. Evolving video skims into useful multimedia abstractions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 171–178. ACM Press/Addison-Wesley Publishing Co.

Serhan Dagtas and Mohamed Abdel-Mottaleb. 2001. Extraction of tv highlights using multimedia features. In *Multimedia Signal Processing, 2001 IEEE Fourth Workshop on*, pages 91–96. IEEE.

Manfred Del Fabro, Anita Sobe, and Laszlo Böszörmenyi. 2012. Summarization of real-life events based on community-contributed content. In *The Fourth International Conferences on Advances in Multimedia*, pages 119–126.

Gunes Erkan and Dragomir R. Radev. 2011. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Qiqihar Junior Teachers College*, 22:2004.

Berna Erol, D-S Lee, and Jonathan Hull. 2003. Multimodal summarization of meeting recordings. In *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 3, pages III–25. IEEE.

Georgios Evangelopoulos, Athanasia Zlatintsi, Alexandros Potamianos, Petros Maragos, Konstantinos Rapantzikos, Georgios Skoumas, and Yannis Avrithis. 2013. Multimodal saliency and fusion for movie summarization based on aural, visual, and textual attention. *IEEE Transactions on Multimedia*, 15(7):1553–1568.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics, Volume 28, Number 3, September 2002*.

Ralph Gross, Michael Bett, Hua Yu, Xiaojin Zhu, Yue Pan, Jie Yang, and Alex Waibel. 2000. Towards a multimodal meeting record. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 3, pages 1593–1596. IEEE.

Taufiq Hasan, Hynek Bořil, Abhijeet Sangwan, and John HL Hansen. 2013. Multi-modal highlight generation for sports videos using an information-theoretic excitability measure. *EURASIP Journal on Advances in Signal Processing*, 2013(1):173.

Samir Khuller, Anna Moss, and Joseph Seffi Naor. 1999. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45.

Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf. 2014. Fisher vectors derived from hybrid gaussian-laplacian mixture models for image annotation. *arXiv preprint arXiv:1411.7399*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

Roger Levy and Christopher D. Manning. 2003. Is it harder to parse chinese, or the chinese treebank? In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*.

Zechao Li, Jinhui Tang, Xueming Wang, Jing Liu, and Hanqing Lu. 2016. Multimedia news summarization in search. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(3):33.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920. Association for Computational Linguistics.

Ioannis Mademlis, Anastasios Tefas, Nikos Nikolaidis, and Ioannis Pitas. 2016. Multimodal stereoscopic movie summarization conforming to narrative characteristics. *IEEE Transactions on Image Processing*, 25(12):5828–5840.

Rada Mihalcea and Paul Tarau. 2004. *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, chapter TextRank: Bringing Order into Text.

Chris Quirk, Chris Brockett, and William Dolan. 2004. *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, chapter Monolingual Machine Translation for Paraphrase Generation.

Manos Schinas, Symeon Papadopoulos, Georgios Petkos, Yiannis Kompatsiaris, and Pericles A Mitkas. 2015. Multimodal graph-based event detection and summarization in social media streams. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 189–192. ACM.

Rajiv Ratn Shah, Anwar Dilawar Shaikh, Yi Yu, Wenjing Geng, Roger Zimmermann, and Gangshan Wu. 2015. Eventbuilder: Real-time multimedia event summarization by visualizing social media. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 185–188. ACM.

Rajiv Ratn Shah, Yi Yu, Akshay Verma, Suhua Tang, Anwar Dilawar Shaikh, and Roger Zimmermann. 2016. Leveraging multimodal information for event summarization and concept-level sentiment analysis. *Knowledge-Based Systems*, 108:102–109.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Dian Tjondronegoro, Xiaohui Tao, Johannes Sasongko, and Cher Han Lau. 2011. Multi-modal summarization of key events and top players in sports tournament videos. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 471–478. IEEE.

H. Tseng, P. Chang, G. Andrew, D. Jurafsky, and C. Manning. 2005. A conditional random field word segmenter.

Robin Valenza, Tony Robinson, Marianne Hickey, and Roger Tucker. 1999. Summarisation of spoken audio through information extraction. In *ESCA Tutorial and Research Workshop (ETRW) on Accessing Information in Spoken Audio*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2016. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE transactions on pattern analysis and machine intelligence*.

Xiaojun Wan and Jianwu Yang. 2006. Improved affinity graph based multi-document summarization. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*.

Dingding Wang, Tao Li, and Mitsunori Ogihara. 2012. Generating pictorial storylines via minimum-weight connected dominating set approximation in multiview graphs. In *AAAI*.

Liwei Wang, Yin Li, and Svetlana Lazebnik. 2016a. Learning deep structure-preserving image-text embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5005–5013.

Yang William Wang, Yashar Mehdad, R. Dragomir Radev, and Amanda Stent. 2016b. A low-rank approximation approach to learning joint embeddings of news stories and images for timeline summarization. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 58–68. Association for Computational Linguistics.

Haitong Yang and Chengqing Zong. 2014. Multi-predicate semantic role labeling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 363–373. Association for Computational Linguistics.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations. *Transactions of the Association of Computational Linguistics*, 2:67–78.

Jiajun Zhang, Yu Zhou, Chengqing Zong, Jiajun Zhang, Yu Zhou, Chengqing Zong, Jiajun Zhang, Chengqing Zong, and Yu Zhou. 2016. Abstractive cross-language summarization via translation model enhanced predicate argument structure fusing. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(10):1842–1853.

Yueting Zhuang, Yong Rui, Thomas S Huang, and Sharad Mehrotra. 1998. Adaptive key frame extraction using unsupervised clustering. In *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, volume 1, pages 866–870. IEEE.

Tensor Fusion Network for Multimodal Sentiment Analysis

Amir Zadeh[†], **Minghai Chen**[†]
Language Technologies Institute
Carnegie Mellon University
{abagherz,minghail}@cs.cmu.edu

Soujanya Poria
Temasek Laboratories,
NTU, Singapore
sporia@ntu.edu.sg

Erik Cambria
School of Computer Science and
Engineering, NTU, Singapore
cambria@ntu.edu.sg

Louis-Philippe Morency
Language Technologies Institute
Carnegie Mellon University
morency@cs.cmu.edu

Abstract

Multimodal sentiment analysis is an increasingly popular research area, which extends the conventional language-based definition of sentiment analysis to a multimodal setup where other relevant modalities accompany language. In this paper, we pose the problem of multimodal sentiment analysis as modeling *intra-modality* and *inter-modality* dynamics. We introduce a novel model, termed Tensor Fusion Network, which learns both such dynamics end-to-end. The proposed approach is tailored for the volatile nature of spoken language in online videos as well as accompanying gestures and voice. In the experiments, our model outperforms state-of-the-art approaches for both multimodal and unimodal sentiment analysis.

1 Introduction

Multimodal sentiment analysis (Morency et al., 2011; Zadeh et al., 2016b; Poria et al., 2015) is an increasingly popular area of affective computing research (Poria et al., 2017) that focuses on generalizing text-based sentiment analysis to opinionated videos, where three communicative modalities are present: language (spoken words), visual (gestures), and acoustic (voice).

This generalization is particularly vital to part of the NLP community dealing with opinion mining and sentiment analysis (Cambria et al., 2017) since there is a growing trend of sharing opinions in videos instead of text, specially in social media (Facebook, YouTube, etc.). The central challenge in multimodal sentiment analysis is to model the *inter-modality* dynamics: the interactions between language, visual and acoustic behaviors that change the perception of the expressed sentiment.

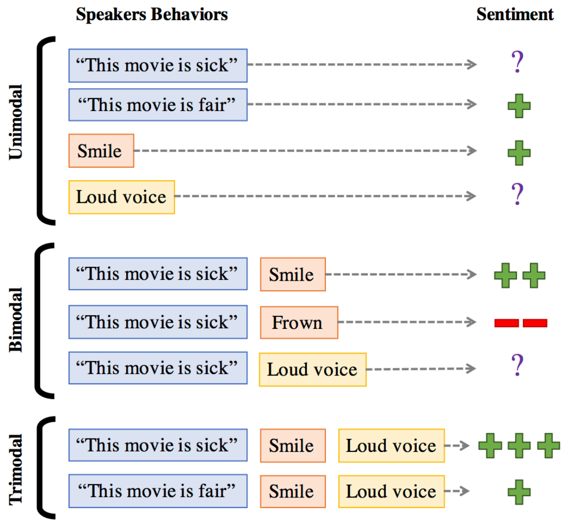


Figure 1: Unimodal, bimodal and trimodal interaction in multimodal sentiment analysis.

Figure 1 illustrates these complex inter-modality dynamics. The utterance "This movie is sick" can be ambiguous (either positive or negative) by itself, but if the speaker is also smiling at the same time, then it will be perceived as positive. On the other hand, the same utterance with a frown would be perceived negatively. A person speaking loudly "This movie is sick" would still be ambiguous. These examples are illustrating **bimodal** interactions. Examples of **trimodal** interactions are shown in Figure 1 when loud voice increases the sentiment to strongly positive. The complexity of inter-modality dynamics is shown in the second trimodal example where the utterance "This movie is fair" is still weakly positive, given the strong influence of the word "fair".

A second challenge in multimodal sentiment analysis is efficiently exploring *intra-modality* dynamics of a specific modality (**unimodal** interaction). Intra-modality dynamics are particularly

[†] means equal contribution

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1103–1114
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

challenging for the language analysis since multimodal sentiment analysis is performed on spoken language. A spoken opinion such as "I think it was alright . . . Hmmm . . . let me think . . . yeah . . . no . . . ok yeah" almost never happens in written text. This volatile nature of spoken opinions, where proper language structure is often ignored, complicates sentiment analysis. Visual and acoustic modalities also contain their own intra-modality dynamics which are expressed through both space and time.

Previous works in multimodal sentiment analysis does not account for both intra-modality and inter-modality dynamics directly, instead they either perform early fusion (a.k.a., feature-level fusion) or late fusion (a.k.a., decision-level fusion). Early fusion consists in simply concatenating multimodal features mostly at input level (Morency et al., 2011; Pérez-Rosas et al., 2013; Poria et al., 2016). This fusion approach does not allow the intra-modality dynamics to be efficiently modeled. This is due to the fact that inter-modality dynamics can be more complex at input level and can dominate the learning process or result in overfitting. Late fusion, instead, consists in training unimodal classifiers independently and performing decision voting (Wang et al., 2016; Zadeh et al., 2016a). This prevents the model from learning inter-modality dynamics in an efficient way by assuming that simple weighted averaging is a proper fusion approach.

In this paper, we introduce a new model, termed Tensor Fusion Network (TFN), which learns both the intra-modality and inter-modality dynamics end-to-end. Inter-modality dynamics are modeled with a new multimodal fusion approach, named Tensor Fusion, which explicitly aggregates unimodal, bimodal and trimodal interactions. Intra-modality dynamics are modeled through three Modality Embedding Subnetworks, for language, visual and acoustic modalities, respectively.

In our extensive set of experiments, we show (a) that TFN outperforms previous state-of-the-art approaches for multimodal sentiment analysis, (b) the characteristics and capabilities of our Tensor Fusion approach for multimodal sentiment analysis, and (c) that each of our three Modality Embedding Subnetworks (language, visual and acoustic) are also outperforming unimodal state-of-the-art unimodal sentiment analysis approaches.

2 Related Work

Sentiment Analysis is a well-studied research area in NLP (Pang et al., 2008). Various approaches have been proposed to model sentiment from language, including methods that focus on opinionated words (Hu and Liu, 2004; Taboada et al., 2011; Poria et al., 2014b; Cambria et al., 2016), n-grams and language models (Yang and Cardie, 2012), sentiment compositionality and dependency-based analysis (Socher et al., 2013; Poria et al., 2014a; Agarwal et al., 2015; Tai et al., 2015), and distributional representations for sentiment (Iyyer et al., 2015).

Multimodal Sentiment Analysis is an emerging research area that integrates verbal and nonverbal behaviors into the detection of user sentiment. There exist several multimodal datasets that include sentiment annotations, including the newly-introduced CMU-MOSI dataset (Zadeh et al., 2016b), as well as other datasets including ICT-MMMO (Wöllmer et al., 2013), YouTube (Morency et al., 2011), and MOUD (Pérez-Rosas et al., 2013), however CMU-MOSI is the only English dataset with utterance-level sentiment labels. The newest multimodal sentiment analysis approaches have used deep neural networks, including convolutional neural networks (CNNs) with multiple-kernel learning (Poria et al., 2015), SAL-CNN (Wang et al., 2016) which learns generalizable features across speakers, and support vector machines (SVMs) with a multimodal dictionary (Zadeh, 2015).

Audio-Visual Emotion Recognition is closely tied to multimodal sentiment analysis (Poria et al., 2017). Both audio and visual features have been shown to be useful in the recognition of emotions (Ghosh et al., 2016a). Using facial expressions and audio cues jointly has been the focus of many recent studies (Glodek et al., 2011; Valstar et al., 2016; Nojavanasghari et al., 2016).

Multimodal Machine Learning has been a growing trend in machine learning research that is closely tied to the studies in this paper. Creative and novel applications of using multiple modalities have been among successful recent research directions in machine learning (You et al., 2016; Donahue et al., 2015; Antol et al., 2015; Specia et al., 2016; Tong et al., 2017).

3 CMU-MOSI Dataset

Multimodal Opinion Sentiment Intensity (CMU-MOSI) dataset is an annotated dataset of video

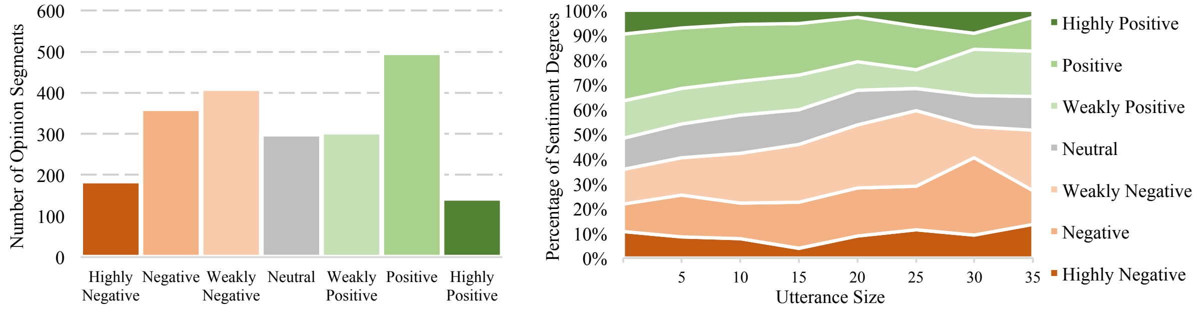


Figure 2: Distribution of sentiment across different opinions (left) and opinion sizes (right) in CMU-MOSI.

opinions from YouTube movie reviews (Zadeh et al., 2016a). Annotation of sentiment has closely followed the annotation scheme of the Stanford Sentiment Treebank (Socher et al., 2013), where sentiment is annotated on a seven-step Likert scale from very negative to very positive. However, whereas the Stanford Sentiment Treebank is segmented by sentence, the CMU-MOSI dataset is segmented by opinion utterances to accommodate spoken language where sentence boundaries are not as clear as text. There are 2199 opinion utterances for 93 distinct speakers in CMU-MOSI. There are an average 23.2 opinion segments in each video. Each video has an average length of 4.2 seconds. There are a total of 26,295 words in the opinion utterances. These utterance are annotated by five Mechanical Turk annotators for sentiment. The final agreement between the annotators is high in terms of Krippendorf's alpha $\alpha = 0.77$. Figure 2 shows the distribution of sentiment across different opinions and different opinion sizes. CMU-MOSI dataset facilitates three prediction tasks, each of which we address in our experiments: 1) *Binary Sentiment Classification* 2) *Five-Class Sentiment Classification* (similar to Stanford Sentiment Treebank fine-grained classification with seven scale being mapped to five) and 3) *Sentiment Regression* in range $[-3, 3]$. For sentiment regression, we report Mean-Absolute Error (lower is better) and correlation (higher is better) between the model predictions and regression ground truth.

4 Tensor Fusion Network

Our proposed TFN consists of three major components: 1) *Modality Embedding Subnetworks* take as input unimodal features, and output a rich modality embedding. 2) *Tensor Fusion Layer* explicitly models the unimodal, bimodal and trimodal interactions using a 3-fold Cartesian product from modality embeddings. 3) *Sentiment Inference Subnetwork* is a network conditioned on the output of the Tensor Fusion Layer and performs sentiment inference. Depending on the task from Section 3 the network output changes to accommodate binary classification, 5-class classification or regression. Input to the TFN is an opinion utterance which includes three modalities of language, visual and acoustic. The following three subsections describe the TFN subnetworks and their inputs in detail.

4.1 Modality Embedding Subnetworks

Spoken Language Embedding Subnetwork: Spoken text is different than written text (reviews, tweets) in compositionality and grammar. We revisit the spoken opinion: "I think it was alright . . . Hmmm . . . let me think . . . yeah . . . no . . . ok yeah". This form of opinion rarely happens in written language but variants of it are very common in spoken language. The first part conveys the actual message and the rest is speaker thinking out loud eventually agreeing with the first part. The key factor in dealing with this volatile nature of spoken language is to build models that are capable of operating in presence of unreliable and idiosyncratic speech traits by focusing on important parts of speech.

Our proposed approach to deal with challenges of spoken language is to learn a rich representation of spoken words at each word interval and use it as input to a fully connected deep network (Figure 3). This rich representation for ith word contains information from beginning of utterance through time, as well as ith word. This way as the model is discovering the meaning of the utterance through time, if it encounters unusable information in word $i + 1$ and arbitrary number of words after, the representation up until i is not diluted or lost. Also, if the model encounters usable information again, it can recover by embedding those in the long short-term memory (LSTM). The time-dependent

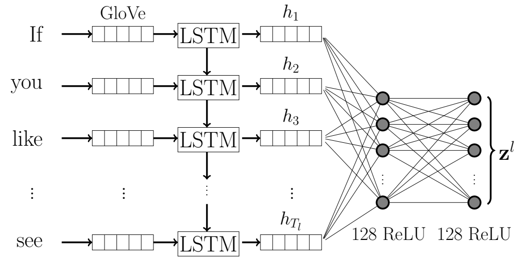


Figure 3: Spoken Language Embedding Subnetwork ($\mathcal{U}_l$)

encodings are usable by the rest of the pipeline by simply focusing on relevant parts using the non-linear affine transformation of time-dependent embeddings which can act as a dimension reducing attention mechanism. To formally define our proposed Spoken Language Embedding Subnetwork ($\mathcal{U}_l$), let $\mathbf{l} = \{l_1, l_2, l_3, \dots, l_{T_l}; l_t \in \mathbb{R}^{300}\}$, where T_l is the number of words in an utterance, be the set of spoken words represented as a sequence of 300-dimensional GloVe word vectors (Pennington et al., 2014).

A LSTM network (Hochreiter and Schmidhuber, 1997) with a forget gate (Gers et al., 2000) is used to learn time-dependent language representations $\mathbf{h_l} = \{h_1, h_2, h_3, \dots, h_{T_l}; h_t \in \mathbb{R}^{128}\}$ for words according to the following LSTM formulation.

$$\begin{pmatrix} i \\ f \\ o \\ m \end{pmatrix} = \begin{pmatrix} sigmoid \\ sigmoid \\ sigmoid \\ tanh \end{pmatrix} W_{l_d} \begin{pmatrix} X_t W_{l_e} \\ h_{t-1} \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot m$$

$$h_t = o \otimes tanh(c_t)$$

$$\mathbf{h_l} = [h_1; h_2; h_3; \dots; h_{T_l}]$$

$\mathbf{h_l}$ is a matrix of language representations formed from concatenation of $h_1, h_2, h_3, \dots h_{T_l}$. $\mathbf{h_l}$ is then used as input to a fully-connected network that generates language embedding $\mathbf{z}^l$:

$$\mathbf{z}^l = \mathcal{U}_l(\mathbf{l};\ W_l) \in \mathbb{R}^{128}$$

where W_l is the set of all weights in the $\mathcal{U}_l$ network (including W_{l_d}, W_{l_e},$W_{l_{fc}}$, and $b_{l_{fc}}$), σ is the sigmoid function.

Visual Embedding Subnetwork: Since opinion videos consist mostly of speakers talking to the audience through close-up camera, face is the most important source of visual information. The speaker's face is detected for each frame (sampled at 30Hz) and indicators of the seven basic emotions (anger, contempt, disgust, fear, joy, sadness, and surprise) and two advanced emotions (frustration and confusion) (Ekman, 1992) are extracted using FACET facial expression analysis framework[1]. A set of 20 Facial Action Units (Ekman et al., 1980), indicating detailed muscle movements on the face, are also extracted using FACET. Estimates of head position, head rotation, and 68 facial landmark locations also extracted per frame using OpenFace (Baltrušaitis et al., 2016; Zadeh et al., 2017).

Let the visual features $\hat{\mathbf{v}}_j = [v_j^1, v_j^2, v_j^3, \dots, v_j^p]$ for frame j of utterance video contain the set of p visual features, with T_v the number of total video frames in utterance. We perform mean pooling over the frames to obtain the expected visual features $\mathbf{v} = [\mathbb{E}[v^1], \mathbb{E}[v^2], \mathbb{E}[v^3], \dots, \mathbb{E}[v^l]]$. $\mathbf{v}$ is then used as input to the Visual Embedding Subnetwork $\mathcal{U}_v$. Since information extracted using FACET from videos is rich, using a deep neural network would be sufficient to produce meaningful embeddings of visual modality. We use a deep neural network with three hidden layers of 32 ReLU units and weights W_v. Empirically we observed that making the model deeper or increasing the number of neurons in each layer does not lead to better visual performance. The subnetwork output provides the visual embedding $\mathbf{z}^v$:

$$\mathbf{z}^v = \mathcal{U}_v(\mathbf{v};\ W_v) \in \mathbb{R}^{32}$$

Acoustic Embedding Subnetwork: For each opinion utterance audio, a set of acoustic features are extracted using COVAREP acoustic analysis framework (Degottex et al., 2014), including 12 MFCCs, pitch tracking and Voiced/UnVoiced segmenting features (using the additive noise robust *Summation of Residual Harmonics* (SRH) method (Drugman and Alwan, 2011)), glottal source parameters (estimated by glottal inverse filtering based on GCI synchronous IAIF (Drugman et al., 2012; Alku, 1992; Alku et al., 2002, 1997; Titze and Sundberg, 1992; Childers and Lee, 1991)), peak slope parameters (Degottex et al., 2014), maxima dispersion quotients (MDQ) (Kane and Gobl, 2013), and estimations of the R_d shape parameter of the Liljencrants-Fant (LF) glottal model (Fujisaki and Ljungqvist, 1986). These extracted features capture different characteristics of human voice and have been shown to be related to emotions (Ghosh et al., 2016b).

[1] http://goo.gl/1rh1JN

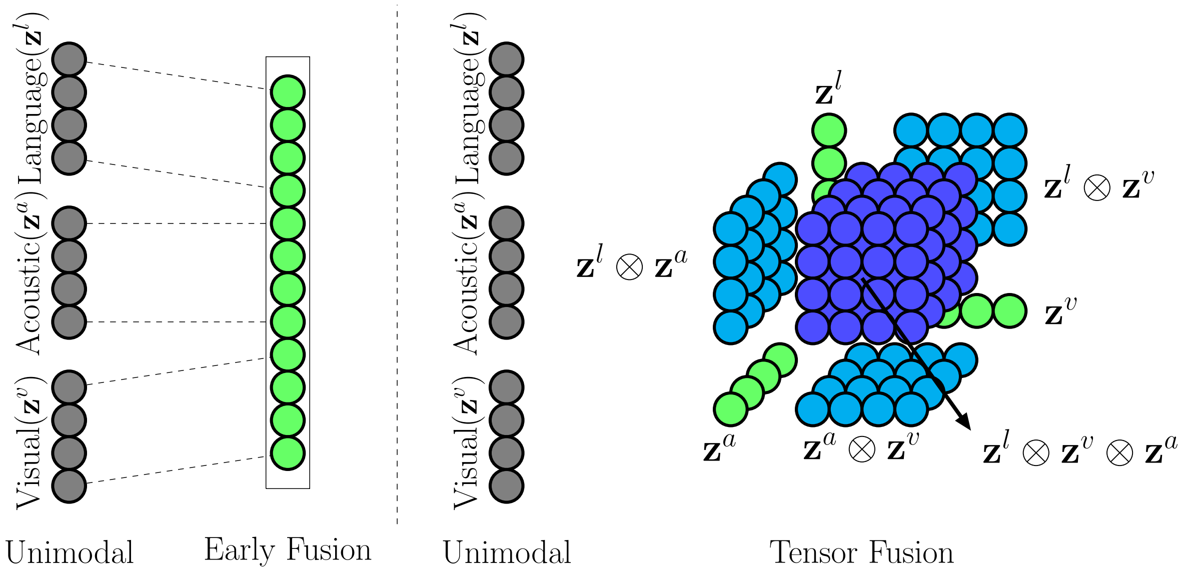


Figure 4: Left: Commonly used early fusion (multimodal concatenation). Right: Our proposed tensor fusion with three types of subtensors: unimodal, bimodal and trimodal.

For each opinion segment with T_a audio frames (sampled at 100Hz; i.e., 10ms), we extract the set of q acoustic features $\hat{\mathbf{a}}_j = [a_j^1, a_j^2, a_j^3, \dots, a_j^q]$ for audio frame j in utterance. We perform mean pooling per utterance on these extracted acoustic features to obtain the expected acoustic features $\mathbf{a} = [\mathbb{E}[a_1], \mathbb{E}[a_2], \mathbb{E}[a_3], \dots, \mathbb{E}[q]]$. Here, $\mathbf{a}$ is the input to the Audio Embedding Subnetwork $\mathcal{U}_a$. Since COVAREP also extracts rich features from audio, using a deep neural network is sufficient to model the acoustic modality. Similar to $\mathcal{U}_v, \mathcal{U}_a$ is a network with 3 layers of 32 ReLU units with weights W_a.

Here, we also empirically observed that making the model deeper or increasing the number of neurons in each layer does not lead to better performance. The subnetwork produces the audio embedding $\mathbf{z}^a$:

$$\mathbf{z}^a = \mathcal{U}_a(\mathbf{a}; W_a) \in \mathbb{R}^{32}$$

4.2 Tensor Fusion Layer

While previous works in multimodal research has used feature concatenation as an approach for multimodal fusion, we aim to build a fusion layer in TFN that disentangles unimodal, bimodal and trimodal dynamics by modeling each of them explicitly. We call this layer Tensor Fusion, which is defined as the following vector field using three-fold Cartesian product:

$$\left\{ (z^l, z^v, z^a) \mid z^l \in \begin{bmatrix} \mathbf{z}^l \\ 1 \end{bmatrix}, z^v \in \begin{bmatrix} \mathbf{z}^v \\ 1 \end{bmatrix}, z^a \in \begin{bmatrix} \mathbf{z}^a \\ 1 \end{bmatrix} \right\}$$

The extra constant dimension with value 1 generates the unimodal and bimodal dynamics. Each neural coordinate (z_l, z_v, z_a) can be seen as a 3-D point in the 3-fold Cartesian space defined by the language, visual, and acoustic embeddings dimensions $[\mathbf{z}^l 1]^T$, $[\mathbf{z}^v 1]^T$, and $[\mathbf{z}^a 1]^T$.

This definition is mathematically equivalent to a differentiable outer product between $\mathbf{z}^l$, the visual representation $\mathbf{z}^v$, and the acoustic representation $\mathbf{z}^a$.

$$\mathbf{z}^m = \begin{bmatrix} \mathbf{z}^l \\ 1 \end{bmatrix} \otimes \begin{bmatrix} \mathbf{z}^v \\ 1 \end{bmatrix} \otimes \begin{bmatrix} \mathbf{z}^a \\ 1 \end{bmatrix}$$

Here $\otimes$ indicates the outer product between vectors and $\mathbf{z}^m \in \mathbb{R}^{129 \times 33 \times 33}$ is the 3D cube of all possible combination of unimodal embeddings with seven semantically distinct subregions in Figure 4. The first three subregions $\mathbf{z}^l$, $\mathbf{z}^v$, and $\mathbf{z}^a$ are unimodal embeddings from Modality Embedding Subnetworks forming unimodal interactions in Tensor Fusion. Three subregions $\mathbf{z}^l \otimes \mathbf{z}^v$, $\mathbf{z}^l \otimes \mathbf{z}^a$, and $\mathbf{z}^v \otimes \mathbf{z}^a$ capture bimodal interactions in Tensor Fusion. Finally, $\mathbf{z}^l \otimes \mathbf{z}^v \otimes \mathbf{z}^a$ captures trimodal interactions.

Early fusion commonly used in multimodal research dealing with language, vision and audio, can be seen as a special case of Tensor Fusion with only unimodal interactions. Since Tensor Fusion is mathematically formed by an outer product, it has no learnable parameters and we empirically observed that although the output tensor is high dimensional, chances of overfitting are low.

We argue that this is due to the fact that the output neurons of Tensor Fusion are easy to interpret and semantically very meaningful (i.e., the manifold that they lie on is not complex but just high dimensional). Thus, it is easy for the subsequent layers of the network to decode the meaningful information.

4.3 Sentiment Inference Subnetwork

After Tensor Fusion layer, each opinion utterance can be represented as a multimodal tensor $\mathbf{z}^m$. We use a fully connected deep neural network called Sentiment Inference Subnetwork $\mathcal{U}_s$ with weights W_s conditioned on $\mathbf{z}^m$. The architecture of the network consists of two layers of 128 ReLU activation units connected to decision layer. The likelihood function of the Sentiment Inference Subnetwork is defined as follows, where ϕ is the sentiment prediction:

$$\underset{\phi}{\arg\max}\ \mathrm{p}(\phi \mid \mathbf{z}^m; W_s) = \underset{\phi}{\arg\max}\ \mathcal{U}_s(\mathbf{z}^m; W_s)$$

In our experiments, we use three variations of the $\mathcal{U}_s$ network. The first network is trained for binary sentiment classification, with a single sigmoid output neuron using binary cross-entropy loss. The second network is designed for five-class sentiment classification, and uses a softmax probability function using categorical cross-entropy loss. The third network uses a single sigmoid output, using mean-squarred error loss to perform sentiment regression.

5 Experiments

In this paper, we devise three sets of experiments each addressing a different research question:

Experiment 1: We compare our TFN with previous state-of-the-art approaches in multimodal sentiment analysis.

Experiment 2: We study the importance of the TFN subtensors and the impact of each individual modality (see Figure 4). We also compare with the commonly-used early fusion approach.

Experiment 3: We compare the performance of our three modality-specific networks (language, visual and acoustic) with state-of-the-art unimodal approaches.

Section 5.4 describes our experimental methodology which is kept constant across all experiments. Section 6 will discuss our results in more details with a qualitative analysis.

Multimodal Baseline	Binary		5-class	Regression	
	Acc(%)	F1	Acc(%)	MAE	r
Random	50.2	48.7	23.9	1.88	-
C-MKL	73.1	75.2	35.3	-	-
SAL-CNN	73.0	-	-	-	-
SVM-MD	71.6	72.3	32.0	1.10	0.53
RF	71.4	72.1	31.9	1.11	0.51
TFN	**77.1**	**77.9**	**42.0**	**0.87**	**0.70**
Human	85.7	87.5	53.9	0.71	0.82
Δ^{SOTA}	↑ 4.0	↑ 2.7	↑ 6.7	↓ 0.23	↑ 0.17

Table 1: Comparison with state-of-the-art approaches for multimodal sentiment analysis. TFN outperforms both neural and non-neural approaches as shown by Δ^{SOTA}.

5.1 E1: Multimodal Sentiment Analysis

In this section, we compare the performance of TFN model with previously proposed multimodal sentiment analysis models. We compare to the following baselines:

C-MKL (Poria et al., 2015) Convolutional MKL-based model is a multimodal sentiment classification model which uses a CNN to extract textual features and uses multiple kernel learning for sentiment analysis. It is current SOTA (state of the art) on CMU-MOSI.

SAL-CNN (Wang et al., 2016) Select-Additive Learning is a multimodal sentiment analysis model that attempts to prevent identity-dependent information from being learned in a deep neural network. We retrain the model for 5-fold cross-validation using the code provided by the authors on github.

SVM-MD (Zadeh et al., 2016b) is a SVM model trained on multimodal features using early fusion. The model used in (Morency et al., 2011) and (Pérez-Rosas et al., 2013) also similarly use SVM on multimodal concatenated features. We also present the results of Random Forest **RF-MD** to compare to another non-neural approach.

The results first experiment are reported in Table 1. TFN outperforms previously proposed neural and non-neural approaches. This difference is specifically visible in the case of 5-class classification.

5.2 E2: Tensor Fusion Evaluation

Table 4 shows the results of our ablation study. The first three rows are showing the performance of each modality, when no intermodality dynamics are modeled. From this first experiment, we observe that the language modality is the most predictive.

Baseline	Binary		5-class	Regression	
	Acc(%)	F1	Acc(%)	MAE	r
TFN$_{language}$	74.8	75.6	38.5	0.99	0.61
TFN$_{visual}$	66.8	70.4	30.4	1.13	0.48
TFN$_{acoustic}$	65.1	67.3	27.5	1.23	0.36
TFN$_{bimodal}$	75.2	76.0	39.6	0.92	0.65
TFN$_{trimodal}$	74.5	75.0	38.9	0.93	0.65
TFN$_{notrimodal}$	75.3	76.2	39.7	0.919	0.66
TFN	**77.1**	**77.9**	**42.0**	**0.87**	**0.70**
TFN$_{early}$	75.2	76.2	39.0	0.96	0.63

Table 2: Comparison of TFN with its subtensor variants. All the unimodal, bimodal and trimodal subtensors are important. TFN also outperforms early fusion.

As a second set of ablation experiments, we test our TFN approach when only the bimodal subtensors are used (TFN$_{bimodal}$) or when only the trimodal subtensor is used (TFN$_{bimodal}$). We observe that bimodal subtensors are more informative when used without other subtensors. The most interesting comparison is between our full TFN model and a variant (TFN$_{notrimodal}$) where the trimodal subtensor is removed (but all the unimodal and bimodal subtensors are present). We observe a big improvement for the full TFN model, confirming the importance of the trimodal dynamics and the need for all components of the full tensor.

We also perform a comparison with the early fusion approach (TFN$_{early}$) by simply concatenating all three modality embeddings $< z^l, z^a, z^v >$ and passing it directly as input to $\mathcal{U}_s$. This approach was depicted on the left side of Figure 4. When looking at Table 4 results, we see that our TFN approach outperforms the early fusion approach[2].

5.3 E3: Modality Embedding Subnetworks Evaluation

In this experiment, we compare the performance of our Modality Embedding Networks with state-of-the-art approaches for language-based, visual-based and acoustic-based sentiment analysis.

5.3.1 Language Sentiment Analysis

We selected the following state-of-the-art approaches to include variety in their techniques, based on dependency parsing (RNTN), distributional representation of text (DAN), and convolutional approaches (DynamicCNN). When possible, we retrain them on the CMU-MOSI dataset (performances of the original pre-trained models are shown in parenthesis in Table 3) and compare them to our language only TFN$_{language}$.

Language Baseline	Binary		5-class	Regression	
	Acc(%)	F1	Acc(%)	MAE	r
RNTN	- (73.7)	- (73.4)	- (35.2)	- (0.99)	- (0.59)
DAN	73.4 (68.8)	73.8 (68.4)	**39.2** (36.7)	- -	- -
D-CNN	65.5 (62.1)	66.9 (56.4)	32.0 (32.4)	- -	- -
CMKL-L	71.2	72.4	34.5	-	-
SAL-CNN-L	73.5	-	-	-	-
SVM-MD-L	70.6	71.2	33.1	1.18	0.46
TFN$_{language}$	**74.8**	**75.6**	38.5	**0.98**	**0.62**
$\Delta^{SOTA}_{language}$	↑ 1.1	↑ 1.8	↓ 0.7	↓ 0.01	↑ 0.03

Table 3: Language Sentiment Analysis. Comparison of with state-of-the-art approaches for language sentiment analysis. $\Delta^{SOTA}_{language}$ shows improvement.

RNTN (Socher et al., 2013)The Recursive Neural Tensor Network is among the most well-known sentiment analysis methods proposed for both binary and multi-class sentiment analysis that uses dependency structure.

DAN (Iyyer et al., 2015) The Deep Average Network approach is a simple but efficient sentiment analysis model that uses information only from distributional representation of the words and not from the compositionality of the sentences.

DynamicCNN (Kalchbrenner et al., 2014) DynamicCNN is among the state-of-the-art models in text-based sentiment analysis which uses a convolutional architecture adopted for the semantic modeling of sentences.

CMK-L, **SAL-CNN-L** and **SVM-MD-L** are multimodal models from section using only language modality 5.1.

Results in Table 3 show that our model using only language modality outperforms state-of-the-art approaches for the CMU-MOSI dataset. While previous models are well-studied and suitable models for sentiment analysis in written language, they underperform in modeling the sentiment in spoken language. We suspect that this underperformance is due to: RNTN and similar approaches rely heavily on dependency structure, which may not be present

[2] We also performed other comparisons with variants of the early fusion model TFN$_{early}$ where we increased the number of parameters and neurons to replicate the numbers from our TFN model. In all cases, the performances were similar to TFN$_{early}$ (and lower than our TFN model). Because of space constraints, we could not include them in this paper.

Visual Baseline	Binary		5-class	Regression	
	Acc(%)	F1	Acc(%)	MAE	r
3D-CNN	56.1	58.4	24.9	1.31	0.26
CNN-LSTM	60.7	61.2	25.1	1.27	0.30
LSTM-FA	62.1	63.7	26.2	1.23	0.33
CMKL-V	52.6	58.5	29.3	-	-
SAL-CNN-V	63.8	-	-	-	-
SVM-MD-V	59.2	60.1	25.6	1.24	0.36
TFN$_{visual}$	**69.4**	**71.4**	**31.0**	**1.12**	**0.50**
Δ^{SOTA}_{visual}	↑ 5.6	↑ 7.7	↑ 1.7	↓ 0.11	↑ 0.14

Table 4: Visual Sentiment Analysis. Comparison with state-of-the-art approaches for visual sentiment analysis and emotion recognition. Δ^{SOTA}_{visual} shows the improvement.

Acoustic Baseline	Binary		5-class	Regression	
	Acc(%)	F1	Acc(%)	MAE	r
HL-RNN	63.4	64.2	25.9	**1.21**	0.34
Adieu-Net	59.2	60.6	25.1	1.29	0.31
SER-LSTM	55.4	56.1	24.2	1.36	0.23
CMKL-A	52.6	58.5	**29.1**	-	-
SAL-CNN-A	62.1	-	-	-	-
SVM-MD-A	56.3	58.0	24.6	1.29	0.28
TFN$_{acoustic}$	**65.1**	**67.3**	27.5	1.23	**0.36**
$\Delta^{SOTA}_{acoustic}$	↑ 1.7	↑ 3.1	↓ 1.6	↑ 0.02	↑ 0.02

Table 5: Acoustic Sentiment Analysis. Comparison with state-of-the-art approaches for audio sentiment analysis and emotion recognition. $\Delta^{SOTA}_{acoustic}$ shows improvement.

in spoken language; DAN and similar sentence embeddings approaches can easily be diluted by words that may not relate directly to sentiment or meaning; D-CNN and similar convolutional approaches rely on spatial proximity of related words, which may not always be present in spoken language.

5.3.2 Visual Sentiment Analysis

We compare the performance of our models using visual information (TFN$_{visual}$) with the following well-known approaches in visual sentiment analysis and emotion recognition (retrained for sentiment analysis):

3DCNN (Byeon and Kwak, 2014) a network using 3D CNN is trained using the face of the speaker. Face of the speaker is extracted in every 6 frames and resized to 64×64 and used as the input to the proposed network.

CNN-LSTM (Ebrahimi Kahou et al., 2015) is a recurrent model that at each timestamp performs convolutions over facial region and uses output to an LSTM. Face processing is similar to 3DCNN.

LSTM-FA similar to both baselines above, information extracted by FACET is used every 6 frames as input to an LSTM with a memory dimension of 100 neurons.

SAL-CNN-V, **SVM-MD-V**, **CMKL-V**, **RF-V** use only visual modality in multimodal baselines from Section 5.1.

The results in Table 5 show that $\mathcal{U}_v$ is able to outperform state-of-the-art approaches on visual sentiment analysis.

5.3.3 Acoustic Sentiment Analysis

We compare the performance of our models using visual information (TFN$_{acoustic}$) with the following well-known approaches in audio sentiment analysis and emotion recognition (retrained for sentiment analysis):

HL-RNN (Lee and Tashev, 2015) uses an LSTM on high-level audio features. We use the same features extracted for $\mathcal{U}_a$ averaged over time slices of every 200 intervals.

Adieu-Net (Trigeorgis et al., 2016) is an end-to-end approach for emotion recognition in audio using directly PCM features.

SER-LSTM (Lim et al., 2016) is a model that uses recurrent neural networks on top of convolution operations on spectrogram of audio.

SAL-CNN-A, **SVM-MD-A**, **CMKL-A**, **RF-A** use only acoustic modality in multimodal baselines from Section 5.1.

5.4 Methodology

All the models in this paper are tested using five-fold cross-validation proposed by CMU-MOSI (Zadeh et al., 2016a). All of our experiments are performed independent of speaker identity, as no speaker is shared between train and test sets for generalizability of the model to unseen speakers in real-world. The best hyperparameters are chosen using grid search based on model performance on a validation set (using last 4 videos in train fold). The TFN model is trained using the Adam optimizer (Kingma and Ba, 2014) with the learning rate 5e4. $\mathcal{U}_v$ and $\mathcal{U}_a, \mathcal{U}_s$ subnetworks are regularized using dropout on all hidden layers with $p = 0.15$ and L2 norm coefficient 0.01. The train, test and validation folds are exactly the same for all baselines.

6 Qualitative Analysis

We analyze the impact of our proposed TFN multimodal fusion approach by comparing it with the

#	Spoken words + acoustic and visual behaviors	TFN- Acoustic	TFN- Visual	TFN- Language	TFN- Early	TFN	Ground Truth
1	“You can’t even tell funny jokes” + frowning expression	-0.375	-1.760	-0.558	-0.839	-1.661	-1.800
2	“I gave it a B” + smile expression + excited voice	1.967	1.245	0.438	0.467	1.215	1.400
3	“But I must say those are some pretty big shoes to fill so I thought maybe it has a chance” + headshake	-0.378	-1.034	1.734	1.385	0.608	0.400
4	“The only actor who can really sell their lines is Erin Eckart” + frown + low-energy voice	-0.970	-0.716	0.175	-0.031	-0.825	-1.000

Table 6: Examples from the CMU-MOSI dataset. The ground truth sentiment labels are between strongly negative (-3) and strongly positive (+3). For each example, we show the prediction output of the three unimodal models (TFN$_{acoustic}$, TFN$_{visual}$ and TFN$_{language}$), the early fusion model TFN$_{early}$ and our proposed TFN approach. TFN$_{early}$ seems to be mostly replicating language modality while our TFN approach successfully integrate intermodality dynamics to predict the sentiment level.

early fusion approach TFN$_{early}$ and the three unimodal models. Table 6 shows examples taken from the CMU-MOSI dataset. Each example is described with the spoken words as well as the acoustic and visual behaviors. The sentiment predictions and the ground truth labels range between strongly negative (-3) and strongly positive (+3).

As a first general observation, we observe that the early fusion model TFN$_{early}$ shows a strong preference for the language modality and seems to be neglecting the intermodality dynamics. We can see this trend by comparing it with the language unimodal model TFN$_{language}$. In comparison, our TFN approach seems to capture more complex interaction through bimodal and trimodal dynamics and thus performs better. Specifically, in the first example, the utterance is weakly negative where the speaker is referring to lack of funny jokes in the movie. This example contains a bimodal interaction where the visual modality shows a negative expression (frowning) which is correctly captured by our TFN approach.

In the second example, the spoken words are ambiguous since the model has no clue what a B is except a token, but the acoustic and visual modalities are bringing complementary evidences. Our TFN approach correctly identify this trimodal interaction and predicts a positive sentiment. The third example is interesting since it shows an interaction where language predicts a positive sentiment but the strong negative visual behaviors bring the final prediction of our TFN approach almost to a neutral sentiment. The fourth example shows how the acoustic modality is also influencing our TFN predictions.

7 Conclusion

We introduced a new end-to-end fusion method for sentiment analysis which explicitly represents unimodal, bimodal, and trimodal interactions between behaviors. Our experiments on the publicly-available CMU-MOSI dataset produced state-of-the-art performance when compared against both multimodal approaches. Furthermore, our approach brings state-of-the-art results for language-only, visual-only and acoustic-only multimodal sentiment analysis on CMU-MOSI.

Acknowledgments

This project was partially supported by Oculus research grant. We would like to thank the reviewers for their valuable feedback.

References

Basant Agarwal, Soujanya Poria, Namita Mittal, Alexander Gelbukh, and Amir Hussain. 2015. Concept-level sentiment analysis with dependency-based semantic parsing: a novel approach. *Cognitive Computation* 7(4):487–499.

Paavo Alku. 1992. Glottal wave analysis with pitch synchronous iterative adaptive inverse filtering. *Speech communication* 11(2-3):109–118.

Paavo Alku, Tom Bäckström, and Erkki Vilkman. 2002. Normalized amplitude quotient for parametrization of the glottal flow. *the Journal of the Acoustical Society of America* 112(2):701–710.

Paavo Alku, Helmer Strik, and Erkki Vilkman. 1997. Parabolic spectral parameter—a new method for quantification of the glottal flow. *Speech Communication* 22(1):67–79.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*. pages 2425–2433.

Tadas Baltrušaitis, Peter Robinson, and Louis-Philippe Morency. 2016. Openface: an open source facial behavior analysis toolkit. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. IEEE, pages 1–10.

Young-Hyen Byeon and Keun-Chang Kwak. 2014. Facial expression recognition using 3d convolutional neural network. *International Journal of Advanced Computer Science and Applications* 5(12).

Erik Cambria, Dipankar Das, Sivaji Bandyopadhyay, and Antonio Feraco. 2017. *A Practical Guide to Sentiment Analysis*. Springer, Cham, Switzerland.

Erik Cambria, Soujanya Poria, Rajiv Bajpai, and Björn Schuller. 2016. SenticNet 4: A semantic resource for sentiment analysis based on conceptual primitives. In *COLING*. pages 2666–2677.

Donald G Childers and CK Lee. 1991. Vocal quality factors: Analysis, synthesis, and perception. *the Journal of the Acoustical Society of America* 90(5):2394–2410.

Gilles Degottex, John Kane, Thomas Drugman, Tuomo Raitio, and Stefan Scherer. 2014. Covarep—a collaborative voice analysis repository for speech technologies. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, pages 960–964.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. pages 2625–2634.

Thomas Drugman and Abeer Alwan. 2011. Joint robust voicing detection and pitch estimation based on residual harmonics. In *Interspeech*. pages 1973–1976.

Thomas Drugman, Mark Thomas, Jon Gudnason, Patrick Naylor, and Thierry Dutoit. 2012. Detection of glottal closure instants from speech signals: A quantitative review. *IEEE Transactions on Audio, Speech, and Language Processing* 20(3):994–1006.

Samira Ebrahimi Kahou, Vincent Michalski, Kishore Konda, Roland Memisevic, and Christopher Pal. 2015. Recurrent neural networks for emotion recognition in video. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM, pages 467–474.

Paul Ekman. 1992. An argument for basic emotions. *Cognition & emotion* 6(3-4):169–200.

Paul Ekman, Wallace V Freisen, and Sonia Ancoli. 1980. Facial signs of emotional experience. *Journal of personality and social psychology* 39(6):1125–1134.

Hiroya Fujisaki and Mats Ljungqvist. 1986. Proposal and evaluation of models for the glottal source waveform. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'86.*. IEEE, volume 11, pages 1605–1608.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with lstm. *Neural computation* 12(10):2451–2471.

Sayan Ghosh, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2016a. Representation learning for speech emotion recognition. In *Interspeech 2016*. pages 3603–3607. https://doi.org/10.21437/Interspeech.2016-692.

Sayan Ghosh, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2016b. Representation learning for speech emotion recognition. *Interspeech 2016* pages 3603–3607.

Michael Glodek, Stephan Tschechne, Georg Layher, Martin Schels, Tobias Brosch, Stefan Scherer, Markus Kächele, Miriam Schmidt, Heiko Neumann, Günther Palm, et al. 2011. Multiple classifier systems for the classification of audio-visual emotional states. In *Affective Computing and Intelligent Interaction*, Springer, pages 359–368.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 168–177.

Mohit Iyyer, Varun Manjunatha, Jordan L Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL (1)*. pages 1681–1691.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences pages 656–666.

John Kane and Christer Gobl. 2013. Wavelet maxima dispersion for breathy to tense voice discrimination. *IEEE Transactions on Audio, Speech, and Language Processing* 21(6):1170–1179.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Jinkyu Lee and Ivan Tashev. 2015. High-level feature representation using recurrent neural network for speech emotion recognition. In *INTERSPEECH*. pages 1537–1540.

Wootaek Lim, Daeyoung Jang, and Taejin Lee. 2016. Speech emotion recognition using convolutional and recurrent neural networks. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2016 Asia-Pacific*. IEEE, pages 1–4.

Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. 2011. Towards multimodal sentiment analysis: Harvesting opinions from the web. In *Proceedings of the 13th international conference on multimodal interfaces*. ACM, pages 169–176.

Behnaz Nojavanasghari, Tadas Baltrušaitis, Charles E Hughes, and Louis-Philippe Morency. 2016. Emoreact: a multimodal approach and dataset for recognizing emotional responses in children. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*. ACM, pages 137–144.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval* 2(1–2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Verónica Pérez-Rosas, Rada Mihalcea, and Louis-Philippe Morency. 2013. Utterance-level multimodal sentiment analysis. In *ACL (1)*. pages 973–982.

Soujanya Poria, Basant Agarwal, Alexander Gelbukh, Amir Hussain, and Newton Howard. 2014a. Dependency-based semantic parsing for concept-level text analysis. In *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, pages 113–127.

Soujanya Poria, Erik Cambria, Rajiv Bajpai, and Amir Hussain. 2017. A review of affective computing: From unimodal analysis to multimodal fusion. *Information Fusion* 37:98–125.

Soujanya Poria, Erik Cambria, and Alexander F Gelbukh. 2015. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *EMNLP*. pages 2539–2544.

Soujanya Poria, Erik Cambria, Gregoire Winterstein, and Guang-Bin Huang. 2014b. Sentic patterns: Dependency-based rules for concept-level sentiment analysis. *Knowledge-Based Systems* 69:45–63.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Amir Hussain. 2016. Convolutional mkl based multimodal emotion recognition and sentiment analysis. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, pages 439–448.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. Citeseer, pages 1631–1642.

Lucia Specia, Stella Frank, Khalil Sima'an, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *Proceedings of the First Conference on Machine Translation, Berlin, Germany. Association for Computational Linguistics*.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics* 37(2):267–307.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks pages 1556–1566.

Ingo R Titze and Johan Sundberg. 1992. Vocal intensity in speakers and singers. *the Journal of the Acoustical Society of America* 91(5):2936–2946.

Edmund Tong, Amir Zadeh, and Louis-Philippe Morency. 2017. Combating human trafficking with deep multimodal models. In *Association for Computational Linguistics*.

George Trigeorgis, Fabien Ringeval, Raymond Brueckner, Erik Marchi, Mihalis A Nicolaou, Björn Schuller, and Stefanos Zafeiriou. 2016. Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, pages 5200–5204.

Michel Valstar, Jonathan Gratch, Björn Schuller, Fabien Ringeval, Dennis Lalanne, Mercedes Torres Torres, Stefan Scherer, Giota Stratou, Roddy Cowie, and Maja Pantic. 2016. Avec 2016: Depression, mood, and emotion recognition workshop and challenge. In *Proceedings of the 6th International Workshop on Audio/Visual Emotion Challenge*. ACM, pages 3–10.

Haohan Wang, Aaksha Meghawat, Louis-Philippe Morency, and Eric P Xing. 2016. Select-additive learning: Improving cross-individual generalization in multimodal sentiment analysis. *arXiv preprint arXiv:1609.05244* .

Martin Wöllmer, Felix Weninger, Tobias Knaup, Björn Schuller, Congkai Sun, Kenji Sagae, and Louis-Philippe Morency. 2013. Youtube movie reviews: Sentiment analysis in an audio-visual context. *IEEE Intelligent Systems* 28(3):46–53.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 1335–1345.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 4651–4659.

Amir Zadeh. 2015. Micro-opinion sentiment intensity analysis and summarization in online videos. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM, pages 587–591.

Amir Zadeh, Tadas Baltrušaitis, and Louis-Philippe Morency. 2017. Convolutional experts constrained local model for facial landmark detection. In *Computer Vision and Pattern Recognition Workshop (CVPRW)*. IEEE.

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016a. Mosi: Multimodal corpus of sentiment intensity and subjectivity analysis in online opinion videos. *arXiv preprint arXiv:1606.06259* .

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. 2016b. Multimodal sentiment intensity analysis in videos: Facial gestures and verbal messages. *IEEE Intelligent Systems* 31(6):82–88.

ConStance: Modeling Annotation Contexts to Improve Stance Classification

Kenneth Joseph[1] **Lisa Friedland**[1] **Will Hobbs**[1] **David Lazer**[1] **Oren Tsur**[1,2]
{k.joseph, l.friedland, w.hobbs}@northeastern.edu
d.lazer@neu.edu, orentsur@bgu.ac.il
[1]Network Science Institute
Northeastern University
[2]Software and Information Systems Engineering
Ben Gurion University of the Negev

Abstract

Manual annotations are a prerequisite for many applications of machine learning. However, weaknesses in the annotation process itself are easy to overlook. In particular, scholars often choose what information to give to annotators without examining these decisions empirically. For subjective tasks such as sentiment analysis, sarcasm, and stance detection, such choices can impact results. Here, for the task of political stance detection on Twitter, we show that providing too little context can result in noisy and uncertain annotations, whereas providing too strong a context may cause it to outweigh other signals. To characterize and reduce these biases, we develop ConStance, a general model for reasoning about annotations across information conditions. Given conflicting labels produced by multiple annotators seeing the same instances with different contexts, ConStance simultaneously estimates gold standard labels and also learns a classifier for new instances. We show that the classifier learned by ConStance outperforms a variety of baselines at predicting political stance, while the model's interpretable parameters shed light on the effects of each context.

1 Introduction

When annotators are asked for objective judgments about a text (e.g., POS tags), the broader context in which the text is situated is often irrelevant. However, many NLP tasks focus on inference of factors beyond words and syntax. For example, the present work addresses the task of detecting political stance on Twitter. We ask annotators to determine whether a given Twitter user supports Donald Trump or Hillary Clinton. However, inferring something about a *user* from a single tweet that she writes may prove difficult. Prior work on stance has relied on annotations collected this way (Mohammad et al., 2016b), but individual tweets do not always contain clear indicators.

One solution to this issue is to supply the annotator with more information about the user. For example, for the similar task of classifying a Twitter user's political affiliation, Cohen and Ruths (2013) display the user's last 10 tweets. Nguyen et al. (2013), studying gender and age, ask annotators to label users by leveraging all information available in their profile. Thus, researchers have provided a range of contexts (or more broadly, information conditions) to annotators in an attempt to balance annotators' exposure to the data needed for accuracy with reasonable costs in terms of time, money and cognitive load.

However, while scholars routinely make such decisions about what information to show annotators, they rarely examine how such decisions actually impact annotations. The first contribution of this paper (Section 3) is to show that, at least for political stance detection on Twitter, displaying different kinds of context to annotators yields significantly different annotations *for the same user*. As a result of these discrepancies, the accuracy of models trained on these annotations varies widely.

While it is possible one could select a "best" context for a given task, our results suggest that doing so *a priori* is difficult and that, moreover, different contexts provide complementary information. What we would prefer, instead, is a model that *learns* how contexts affect annotators and *combines* annotations from multiple contexts to create gold standard labels.

Fortunately, prior work suggests mechanisms for such a model. Typically in annotation tasks,

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1115–1124
Copenhagen, Denmark, September 7–11, 2017.

each item is judged by several annotators, and the resulting labels are aggregated, usually by majority vote, to create a gold standard. As an alternative to majority vote, Raykar et al. (2010) develop an elegant probabilistic approach for learning to aggregate labels produced by annotators of varying quality. Their model jointly estimates gold standard labels (in the form of probability scores), infers annotator error rates, and learns a classifier for use on out-of-sample data.

Our second contribution (Section 4) is an extension of Raykar et al.'s model to handle labels not only created by annotators of varying quality, but also produced under *information conditions* of varying quality. We call this model ConStance[1]. Like Raykar et al. (2010), who find that even low-quality annotators are useful, we find that low-quality contexts can be useful. Specifically, we find that the classifier produced from our model performs better than any classifier trained by majority vote from the same labels. Furthermore, the model provides an unsupervised method for comparing the information conditions by examining their respective error patterns.

Intuitively, ConStance performs a role analogous to boosting for annotations: for an arbitrary task, it permits collection of labels that capture different aspects of the instances at hand, then combines them automatically to determine which are more reliable and to produce a classifier that takes all this into account.

2 Annotating Political Stance

2.1 Political Stance Detection

Stance detection is defined as the task of determining whether an individual is in favor of, against, or neutral towards a target concept based on the content they have generated (Mohammad et al., 2016b). It is related to but distinct from sentiment analysis: a given document can have negative sentiment but a positive stance towards a particular target, or vice versa. Further, for stance detection, the target need not be explicitly mentioned. These points are best illustrated via example: the tweet "I hope that the Democrats get destroyed in this election!" has a negative sentiment (towards Democrats), and (therefore, most likely) implies a positive stance towards Donald Trump.

As a case study for how context impacts annotations, we focus on political stance detection on Twitter—specifically, determining stance towards Hillary Clinton and Donald Trump during the 2016 U.S. election season. This task illustrates the challenges of annotation, since individual tweets are often ambiguous with respect to stance, contexts on Twitter are inherently fractured, and differing contexts can make annotators lean in different directions.

Note that a user's stance, as we use the term in this paper, is a latent (and stable) property of the user. However, short of interviewing the user, we can never be completely certain of her stance. As such, the examples here and evaluations later rely on the authors' best estimates of stance, using all available information.

A user's tweets, in turn, may or may not reveal her stance. This means that, by our definitions, an annotator might accurately perceive no stance in a tweet, yet have their annotation be considered incorrect with respect to the user's true stance. We would consider this case an annotator error caused by lack of context.

As examples of the task, consider annotating the following three tweets: (i) "*crooked Hillary - #lockHerUp*," (ii) "*Lester thinks he can control the crowd when he can't even keep Trump on topic lmao*," and (iii) "*Settling in for #debatenight Hoping to hear an adult conversation*." In the case of (i), a passing familiarity with American politics gives us high confidence that the author is pro-Trump. The tweets in (ii) and (iii) are more ambiguous, but the authors' stances become clearer with access to varying forms of context. For (ii), a Pepe the frog image (a symbol used by the American alt-right movement) in the user profile reveals that the user is probably a Trump supporter. Similarly, for (iii), a profile description that reads "*Stereotypical Iowan who enjoys Hillary Clinton, progressive politics. Chair of CYDIWomen. Previously @HillaryForIA and @NARAL.*" suggests support for Clinton and distaste for Trump.

In order to explore the effects on annotation quality of providing these kinds of context to annotators, we crowd-source annotations for a set of tweets and vary the additional information provided to annotators. For ease of comparison with related work and within our own study, we associate each user with a single anchor tweet. Thus, both annotators and classifiers are asked to deter-

[1] Replication materials for this work, including code for ConStance, are available at https://github.com/kennyjoseph/constance. The paper's Supplementary Material can also be accessed there.

mine the stance of a user using data from one particular time window.

2.2 Data

We collected tweets during the general election season (7/29/2016–11/7/2016) from over 40,000 Twitter users we had previously matched to voter registration records. Matching Twitter users to voter registrations (using methods similar to Barberá, 2016; Hobbs et al., 2017) helps ensure that the accounts we study are controlled by humans, and it supplies additional demographic variables: gender, race and party registered with.

We identified as a political tweet any tweet that mentioned the official handle for Donald Trump (@realDonaldTrump) or Hillary Clinton (@HillaryClinton), or that contained one or more of the following terms or hashtags: Hillary, Clinton, Trump, Donald, #maga, #imwithher, #debatenight, #election2016, #electionnight. We removed all reply tweets, quote tweets and tweets that directly retweeted the candidates. Finally, we kept only those users who posted at least three political tweets.

From these users, we sampled 562 political tweets for crowd-sourced stance annotation, selecting at most one tweet per user. These tweets were all sampled from users who were registered Democrats or Republicans. Half the tweets were paired with Hillary Clinton as the target, the other half with Donald Trump. We also sampled and set aside an additional 250 + 318 tweet/target pairs to use as development and validation data, respectively (see Section 2.5).

2.3 Annotation Task

We used Amazon Mechanical Turk (AMT) for annotation. Annotators were presented a triplet of {tweet, target, context} and were asked to make their decisions on a 5-point Likert scale, ranging from “Definitely Opposes [target]” to “Definitely Supports [target]”. Both prior work (Mohammad et al., 2016b) and our pilot studies suggested confusion between options for a tweet’s irrelevance towards a target and the tweet’s neutrality towards the target, so we used the center of the scale for both options. For this paper, we use a narrower three-point scale formed by merging the “Definitely” and “Probably” options.

Further, while tweets were annotated with respect to different targets, we combine all annotations into a single task by assuming that “anti-Trump” means “pro-Clinton”, and vice-versa. This assumption seems reasonable given that the voting population was strongly polarized during the general (post-primary) election season, and it doubles the amount of data we can use to train the models. Thus, throughout this work the labels we use are taken from the set {“Support Trump / Oppose Clinton” $= -1$, “Neutral / I don’t know” $= 0$, “Oppose Trump / Support Clinton” $= 1$}.

Context	Displays the anchor tweet plus ...
No Context	No additional information
Partial Profile	Profile image, name, and handle
Full Profile	Author’s profile image, name, handle, and description
Previous Tweets	Author’s two most recent tweets *in general* prior to the anchor
(Previous) Political Tweets	Author’s two most recent *political* tweets prior to the anchor
Political Party	Political affiliation (if any) from the author’s voter registration

Table 1: Descriptions of the six contexts (information conditions) presented to the annotators.

2.4 Contexts Studied

Each of the 562 “anchor” tweets was annotated under six different *contexts* (also referred to as information conditions) described in Table 1. (The Supplementary Material provides visual examples of each.) We collected at least three annotations for each tweet/condition pair. Every AMT worker was shown 40 different tweets, one by one, randomly distributed across contexts. Two additional artificial tweets were used to control for task competency.

We selected the conditions in Table 1 based on two factors. First, we included conditions that varied in how much we expected them to impact annotations. For example, we expected the partial profile information to have a relatively small effect, and political party a larger one. Second, we restricted our options to sets of information that we believed would minimally impact task completion times. We confirmed this empirically by regressing the (logged) time to completion for each annotator on the number of tweets she saw for each context, finding no significant effects from any context.

2.5 Gold Standard Labels

Ideally, we would evaluate annotation quality and downstream performance by comparing to ground truth. Unfortunately, ground truth is difficult to characterize for tasks as subjective as stance detection or sentiment analysis (Passonneau and Carpenter, 2014; DiMaggio, 2015). In light of this, we constructed our own labels, using all available information about users, and we use them as an approximation of ground truth.

We constructed these labels in order to evaluate downstream classification performance, and they cover a set of users not shown to the AMT workers. Given our resource constraints and the numerous (at least 18), often conflicting labels already available for tweets shown to AMT workers, we did not create definitive labels for that set.

To create these "gold standard" (GS) labels, we considered all information found on the user's Twitter timeline, including everything AMT annotators could see, plus friend/following relationships, all of their previous tweets, demographics from the voter file, etc. Anecdotally, we found certain cases time-consuming to investigate, which argues for continuing to limit how much information we ask annotators to consider. All gold standard labels were agreed upon by at least two authors, who first labeled the data independently and then came together to discuss disagreements.

Our GS set consists of 318 users (with their associated anchor tweets). Each user is assigned a label from the tertiary Trump/Neutral/Clinton scale. Another 250 manually labeled accounts were used for model development but are not part of reported results. The GS is approximately equally divided among registered Democrats, registered Republicans, and people not registered with either party; the last category includes self-declared Independents and voters not affiliated with any party. We include this third set in order to ensure the models generalize beyond registered Democrats and Republicans.

3 Annotation Quality For Individual Contexts

In this section, we examine how annotator agreement varies depending on the context in which the labels were obtained, and how classifiers trained on majority-vote labels from each individual context, as well as on labels from all contexts combined, perform on the GS. First, we introduce the classifier and features used for the latter task, then discuss results for agreement and classifier performance.

3.1 Classifier, Labels, Features, & Evaluation

For each of the six contexts separately, we construct labels with which to train a classifier. Training labels are constructed using majority vote; we also tried weighting the training instances to match the distribution of labels, but it did not perform as well. We also construct a seventh set of labels using all annotations from all conditions. We then train a classifier on each set of labels. We use Random Forest models, as they outperformed regularized logistic regression and SVMs with linear kernels on the development set. Note that the only difference among the models in this section is the *labels* they are *trained* on.

The feature set used, shown in Table 2, is meant as a straightforward representation of the information seen by annotators; parts of it follow Ebrahimi et al. (2016). We construct three types of features for each tweet: text, sentiment and user features. For text features, we collapse the anchor tweet plus all additional textual context seen by any annotator into a single string, then compute various n-grams from it. For sentiment, we compute various scores from the anchor tweet alone. For user features, we include the user's race and gender, which annotators might have learned from the user's profile picture. Note that because we want models to generalize beyond registered Democrats or Republicans, we *do not* include a feature for political party.

Classifier performance on the GS is measured, following prior work (Mohammad et al., 2016a; Ebrahimi et al., 2016), on the average of the F1 scores on the two classes of interest ("Clinton" and "Trump"). Additionally, we report the average log-loss (the negative log-likelihood, according to the classifier, of the true label). Log-loss and F1 can be seen as complementary measures: whereas F1 evaluates the quality of the ranking of test instances, log-loss evaluates the quality of their individual probability estimates. To compute the probability estimate from a Random Forest, we compute mean class probabilities across all trees.

To assess the statistical significance of differences between two models, we first obtain probability estimates for all GS items. For log-loss, we use a Mann-Whitney test on the scores from the two models being compared. For F1, we create 1000 bootstrap iterations of the sample, compute the average F1 of each, and run a non-parametric difference-in-means test, using 95% confidence

Category	Data Source	Feature Representation
Text	Anchor tweet, previous (political) tweets, profile description	Character n-grams ($n \in [3, 5]$), word n-grams ($n \in [1, 3]$). Preprocessing: only use tokens appearing $\geq$ 10 times, apply tf-idf weighting.
Sentiment	Anchor tweet	VADER score (Hutto and Gilbert, 2014)
		Dictionary approach (Joseph et al., 2017): valence, dominance & arousal scores
User	Voter registration record	Race, gender

Table 2: Features used in classification.

Model	Agreement	Log-Loss	Avg F1
No Context	0.84	0.72	0.61
Partial Profile	0.83	0.71	0.68
Full Profile	0.82	0.69	0.62
Previous Tweets	0.84	0.65	**0.71**
Political Tweets	**0.88**	**0.61**	0.70
Political Party	**0.88**	0.63	0.68
All Combined	0.77	0.62	**0.71**

Table 3: Inter-annotator agreement, then performance of classifier trained on majority vote labels. (Best possible is 1 for agreement and F1, 0 for log-loss.)

intervals.

3.2 Effects of the Contexts

Before evaluating classification results, we consider annotator agreement within each context, calculated like Mohammad et al. (2016b) as the average, across tweets, of the percentage of annotations that match the majority vote. As shown in Table 3, annotators shown No Context achieve an agreement score of 0.84, similar to the 0.8185 reported by Mohammad et al. (2016b). Relative to this baseline, some contexts increase agreement more than others. As one might expect, Previous Political Tweets and Political Party show the strongest signals. Their effects are statistically ($p < .01$, Mann-Whitney test) and practically significant, increasing the number of labels having *full* agreement by 15% and 10%, respectively.

However, annotators shown different contexts did not necessarily converge to the same labels. Notice the low agreement for the All Combined condition: the majority labels held stronger majorities within any individual context than across all of them. In fact, if we look at the six majority vote labels for each tweet, only in 43% of the tweets are these labels in full agreement. At the end of Section 5, we return to the question of why agreement was so low across conditions, with the help of parameters estimated by ConStance.

In the classification task, the results in Table 3 further suggest that Previous Political Tweets serves as the strongest single context. There is a good case to be made for choosing this individual context, which is statistically significantly better than many others. For example, providing annotators with Previous Political Tweets provides a statistically significant increase in both average F1 scores and log-loss (with $p < .01$) over both the No Context and Full Profile conditions. Perhaps most noteworthy is that the All Combined classifier, created from the naive combination of all annotations, is no better than the classifiers from the individual conditions.

To summarize, results suggest that providing annotators with appropriate additional context can improve annotation quality, as measured via annotator agreement and downstream classification performance. However, it was not obvious in advance which context would be most helpful, and performing such an analysis as this requires the time-intensive construction of better "gold standard" labels against which to check the labels already being outsourced to annotators. In addition, the heterogeneity of the labels produced in different contexts suggests that the contexts provide diverse signals we might be able to leverage; however, simply combining all the annotations does not result in improvements.

4 ConStance: General Unified Model

The prior section thus suggests that it may be better to limit a priori decisions and instead to leverage multiple kinds of context during annotation. Like Raykar et al. (2010) assumes for annotators, we might expect (and indeed find) that even those contexts that turn out to be worse on some metrics still might be useful for other purposes. Here, we present a model for such an approach.

ConStance learns a classifier for *items*. For our purposes here, an item is a user together with their anchor tweet and the additional information from which features were derived (see Table 2); more broadly, it is whatever we choose to put into the feature vector. One could choose a differ-

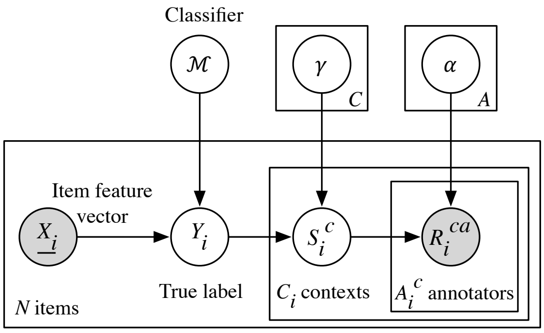


Figure 1: Graphical model for ConStance.

Var.	Meaning
$\underline{X_i}$	Feature vector of item i
Y_i	Latent true label of item i
S_i^c	Latent context-specific label of item i after noise from context c
R_i^{ca}	Label given by annotator a to item i in context c
V	Set of values for labels and annotations: $\{-1, 0, 1\}$
N	# of items, indexed by i
C	Set of contexts, indexed by c
A	Set of annotators, indexed by a
$\mathcal{M}$	Learned classifier
γ^c	$V \times V$ parameter matrix for context c
α^a	$V \times V$ parameter matrix for annotator a
$\mathcal{D}$	All observed data: all values of $\underline{X_i}$ and R_i^{ca}
Z	All latent variables: all values of Y_i and $\underline{S_i}$
θ	All model parameters: $\mathcal{M}, \gamma, \alpha$
T_i	All latent variables for item i: $(Y_i, \underline{S_i})$
$\tau_{i(y\underline{s})}$	Current estimate of all latent values for item i: $p(Y_i = y, \underline{S_i} = \underline{s} \mid \mathcal{D}, \theta)$

Table 4: Model variables.

ent setup; for example, an item could be a user and ten anchor tweets. However, the current arrangement allows for straightforward comparison to prior stance work on Twitter (Mohammad et al., 2016a).

Note that in general, the features need not be restricted to those annotators could have seen. Rather, they could include anything useful to a classifier. Note also that the feature set provided to ConStance is the same used by the baseline models; only the models themselves differ.

4.1 Overview

The model we develop is shown in Figure 1. There are N items to be labeled. Each item can be viewed in up to C different contexts. Finally, there are A total annotators labeling the items; each annotator sees multiple items. Each item can have a different number of annotations, produced by any assignment of annotators and information conditions to items. In our dataset, every item is labeled in 6 conditions (every $|C_i| = 6$), and within every context, every item is labeled by at least 3 annotators (every $|A_i^c| \geq 3$).

The model's generative story works as follows. Item i has feature vector $\underline{X_i}$ and a "true" label $Y_i \in V$. The relationship between $\underline{X_i}$ and Y_i can be described by some model $\mathcal{M}$, which we will ultimately learn. When the item is viewed with context c, the item's true label Y_i is transformed by noise into a "context-specific" label $S_i^c \in V$. In other words, the true label may appear differently when seen through the lens of each context. The variable S_i^c represents what an ideal annotator would say about item i given only as much information as is preserved by context c.

The "noise" introduced by context c is described by parameter γ^c. The parameter γ^c is a $V \times V$ matrix of transition probabilities from true labels to context-specific labels. These probabilities depend only on Y_i and γ^c, not on the item's features $\underline{X_i}$.

Importantly, annotators themselves are also imperfect. When annotator a sees item i, she may also distort the label she sees, S_i^c, into the observed annotation $R_i^{ca} \in V$. The annotator-specific noise process is described by parameter α^a, another $V \times V$ transition matrix.

For a better understanding of the role of γ^c (and by anology, α^a), consider the depictions in Figure 2. The matrix on the top left refers to the No Context condition. Its top row describes what an annotator with perfect judgment would think about a user whose true label is Trump [supporter], with no context. The top left cell, with a value around 0.65, is the probability the annotator would think Trump; the lighter middle cell, with a value around 0.35, is the probability she would think Neutral/Don't know; and the probability she would think Clinton is almost 0.

4.2 Learning

Like Raykar et al. (2010), we perform inference using Expectation Maximization (EM). A full derivation is provided in the Supplementary Material; here, we sketch the main steps.

The model's incomplete data likelihood function, Eq. (1), describes the joint probability, across all items, of Y_i, all values of S_i^c, and all values of R_i^{ca} assuming $\underline{X_i}$ is known and fixed. Uppercase denotes random variables; lowercase, specific values. In line (2), we substitute in the equivalent model parameters.

$$p(\mathcal{D}|\theta, X) = \prod_{i=1}^{N} \sum_{y}^{V} p(Y_i = y|\underline{x_i}, \mathcal{M}) \prod_{c}^{C_i} \sum_{s}^{V} p(S_i^c = s|y, \gamma) \prod_{a}^{A_i^c} p(r_i^{ca}|s, \alpha) \quad (1)$$

$$= \prod_{i=1}^{N} \sum_{y}^{V} \mathcal{M}_y(\underline{x_i}) \prod_{c}^{C_i} \sum_{s}^{V} \gamma_{ys}^c \prod_{a}^{A_i^c} \alpha_{sr}^a \quad (2)$$

The EM derivation is difficult because both Y_i and $\underline{S_i}$ are unobserved. Our solution is to treat the latent variables as a block, describing their joint configuration with a single term $T_i = (Y_i, \underline{S_i})$. In our data, since $|C_i| = 6$, T_i can take on $7^{|V|}$ possible values, a number small enough to enumerate over when we need to marginalize out T_i.

We define membership indicator variables $T_{i(y\underline{s})} \in \{0, 1\}$ such that $T_{i(y\underline{s})} = 1$ if T_i has the specific values $(y, \underline{s})$. During learning, we use analogous variables $\tau_{i(y\underline{s})} \in [0, 1]$ to represent the posterior probabilities of each configuration: $\tau_{i(y\underline{s})} = p(T_{i(y\underline{s})} = 1 \mid \mathcal{D}, \theta)$. The expected value of the complete data log-likelihood is:

$$\mathbb{E}_Z[\ell(\mathcal{D}, Z|\theta, X)] = \sum_{i=1}^{N} \sum_{y}^{V} \left(\sum_{s_i^1}^{V} \cdots \sum_{s_i^{C_i}}^{V} \right) \tau_{i(y\underline{s})}(\log p(T_{i(y\underline{s})} \mid \underline{x_i}, \mathcal{M}, \gamma) + \sum_{c}^{C_i} \sum_{a}^{A_i^c} \log \alpha_{sr}^a) \quad (3)$$

For the E step, we update the membership estimates $\tau_{i(y\underline{s})}$ using the current parameters θ. With Bayes' rule, each item's new value of $\tau_{i(y\underline{s})}$ is shown to be the full joint likelihood of item i (see Eq. (2)) when setting $Y_i = y$ and $\underline{S_i} = \underline{s}$, divided by the sum, over all possible settings of Y_i and $\underline{S_i}$, of that same joint likelihood.

For the M step, we update the model parameters using the current membership estimates. To update the classifier $\mathcal{M}$, following the guidance of Raykar et al. (2010), we retrain the classifier using the current estimates of Y_i as weights for items. The estimates of Y_i can be obtained from $\tau_{iy\underline{s}}$ by marginalizing out $\underline{S_i}$, thus $\mathbb{E}_Z[Y_i = y] = \sum_{s_i^1}^{V} \cdots \sum_{s_i^{C_i}}^{V} \tau_{iy\underline{s}}$. We then use sampling to construct a discrete set of labels for model training based on these weights.

Model	Log-Loss	Avg F1
Best baselines	*0.61*	*0.71*
ConStance	**0.57**	**0.77**
Ablations		
1. Only Political Tweets	0.59	*0.73*
2. Context Labels Masked	**0.57**	0.75
3. Annotator Labels Masked	*0.65*	0.75

Table 5: Classification performance of ConStance and model ablations. Boldface highlights best scores. Significance tests use the the $p < .05$ level for log-loss. Compared to the best baselines, all scores that appear better are statistically significant. Italics indicate the scores that are significantly worse than ConStance.

To update γ and α, we maximize them with respect to Eq. (3). For γ, the entry γ_{ys}^c (i.e., row y, column s of matrix γ^c) denotes $p(S_i^c = s \mid Y_i = y)$. Each matrix entry can be updated individually by taking the partial derivative of Eq. (3) and using, as a Lagrange multiplier term, the constraint that the row must sum to 1. The updated value for γ_{ys}^c turns out to be a fraction in which the numerator is the weighted (by τ) number of items having $Y_i = y$ and $S_i^c = s$, and the denominator is the weighted number of items having $Y_i = y$ (and any value for S_i^c). For α, a similar derivation yields the following update to α_{sr}^a: the weighted number of labels by annotator a, in any context, having $S_i^c = s$ and $R_i^{ca} = r$, divided by the weighted number of labels by annotator a, in any context, having $S_i^c = s$.

5 Model Results and Discussion

The top portion of Table 5 displays ConStance's performance compared to the best results from Section 3. Using the same experimental setup as Section 3—the model type and features, $\mathcal{M}$ and X respectively, are the same as in the baselines—ConStance improves over the best baseline models for each metric. This improvement is statistically significant for both metrics (at the $p < .05$ level for log-loss). Further, the model converges rapidly, within 5-7 iterations of the EM algorithm and 3-5 minutes on a single machine.[2]

In addition to comparing to the baselines provided in Section 3, we investigate which information the model is leveraging to be successful. We do so by exploring three ablations of the model. Variation #1 ("Only Political Tweets" in Table 5)

[2] As above, a development set is used for coarse hyperparameter tuning; see the Supplementary Material for details.

uses the full model, but only gives it the annotations from the Political Tweets condition. This tests whether simply modeling differences in annotators' error rates, as Raykar et al. (2010) do, with a single ("best") context is helpful. We find that it is: the performance of this variation is significantly better on both metrics than the Political Tweets baseline from Table 3.

In the second and third variations, we check whether the effectiveness of ConStance stems from modeling differences between annotators rather than differences in contexts, or vice versa. Variation #2 ("Context Labels Masked"), like #1, models only annotator effects; however, it instead uses the entire set of annotations, treating them as if from a single context (i.e., "masking" context information from the model). Variation #3 ("Annotator Labels Masked") is the complement of Variation #2: it models differences in contexts, and it uses the entire set of annotations, treating them as if from a single annotator.

The results of the model ablation experiments are three-fold. First, we see that each piece of the model on its own is effective in moving beyond baseline approaches that use only one context or naively combine labels across contexts and annotators (the "All Combined" baseline). All model variations achieve significantly higher Avg. F1 than the baselines, and Variations #1 and #2 improve on log-loss. Second, we see that modeling annotators alone is clearly better than not: not only does Variation #1 outperform the Political Tweets baseline (significantly), but also Variation #2 outperforms the All Combined baseline (significantly) and ConStance outperforms Variation #3 (with significance in one measure). Finally, the best results come from using the full model. Even if the differences between ConStance and the variations are not all statistically significant, modeling both annotators and contexts appears to be the most complete and effective approach.

In addition to model performance, we can also examine what ConStance has learned about the quality of labels from each context. Recall that the model produces a parameter matrix for each context, γ^c, which describes how a context distorts the "true" labels the model assumes. Each γ^c is a transition matrix, so a context that perfectly preserves true labels would show up as the identity matrix; off-diagonal entries show error patterns.

Figure 2 visualizes parameter estimates for γ.

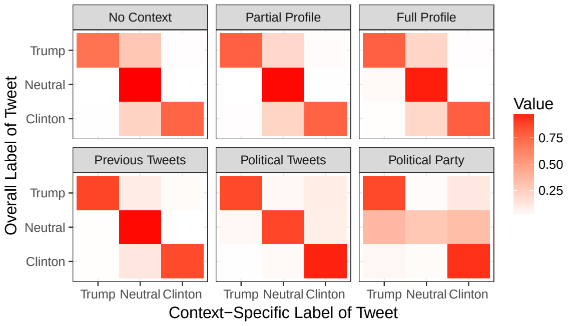


Figure 2: Parameter matrices γ^c learned by ConStance for each context. Darker shading indicates higher values.

We see that in the No Context, Partial Profile and Full Profile conditions, annotators often selected the "Neutral" option (x-axis) when the model inferred the true label was "Clinton" or "Trump" (y-axis). This finding is in line with intuitions; annotators who saw these conditions simply lacked enough information to determine any label.

On the other extreme, in the Political Party context, annotators selected "Trump" or "Clinton" too often when the model settled on the "Neutral" option. That is, even when a user's stance was not clear to annotators in other conditions, annotators who saw political party still inferred stance from the text. Here, one could argue annotators were shown "too much" or "too strong" a context—they saw stance even where the content produced by the user did not suggest one. Indeed, further manual inspection of 90 tweets on which annotations disagreed across contexts implies that annotators who saw political affiliation were often wrong because they focused too little on text content relative to the provided political affiliations.

In presenting these findings, a key point to highlight is that unlike the results of Section 3, Figure 2 was produced without access to any full information labels, which depend on a significant level of manual effort beyond annotations gathered on AMT.

6 Related Work

Recent work has shown that cognitive biases such as stereotypes (Carpenter et al., 2016) and anchoring (Berzak et al., 2016) can negatively impact text annotation and resulting models, even for objective tasks like POS tagging (Blodgett et al., 2016). Still, researchers often decide what context to show annotators without rigorously evaluating how their decisions will affect annotations, on tasks from gender identification to political leanings (Chen et al., 2015; Nguyen et al., 2014; Burger et al., 2011; Cohen and Ruths, 2013). Our work suggests an interesting avenue of development towards reducing annotation bias by explicitly modeling it and reducing the need for a priori decisions on which context is best for which particular task.

In doing so, we draw on a large body of work around improving annotation quality for NLP data. Our work aligns with efforts to improve task design (e.g. Schneider et al., 2013; Morstatter and Liu, 2016; Schneider, 2015), and to develop better models of annotation. With respect to the former and specific to Twitter, Frankenstein et al. (2016) show that for the task of labeling the sentiment of reply tweets, annotations vary depending on whether or not the original tweet (being replied to) is also shown. With respect to the latter, several recent models beyond Raykar et al. (2010) have been proposed (Guan et al., 2017; Tian and Zhu, 2012; Wauthier and Jordan, 2011; Passonneau and Carpenter, 2014). However, our work is most similar to efforts outside the domain of NLP, where Dai et al. (2013) have developed a method of switching between task workflows based on annotation quality for particular items, and Nguyen et al. (2016) have developed a Bayesian model similar to ours to study annotation quality for other kinds of slightly subjective tasks.

In a closely related vein, recent work has also considered how text annotations may vary in important ways based on the characteristics of annotators (rather than how the task is posed, as we study here) (Sen et al., 2015). An interesting avenue of future work is to understand the intersection between the design of NLP annotation tasks and the characteristics of the annotating population.

7 Conclusion and Future Work

Annotated data serves as a foundational layer for many NLP tasks. While some annotation tasks only require information from short texts, in many others, we can elicit higher-quality labels by providing annotators with additional contextual information. However, asking annotators to consider too much information would make their task slow and burdensome.

In this paper we demonstrate how exposing annotators to short contextual information leads to better labels and better classification results. However, different contexts lead to results of different quality, and it is not obvious a priori which context is best, nor—even given ground truth—how to combine labels produced across contexts to exploit the information present in each. We then propose ConStance, a generalizable model that learns the effects of both individual contexts and individual annotators on the labeling process. The model infers (probability estimates for) ground truth labels, plus learns a classifier that can be applied to new instances. We show that this classifier significantly improves classification of political stance compared to the standard practice of training models on majority vote labels.

The focus of this work is on improving both the annotation process for nuanced, context-dependent tasks and the use of the resulting labels. While ConStance's label estimation can be used in conjunction with any classification method, this paper does not address the optimization of the classifier itself. Thus, while we consider an assortment of contexts and use a rich feature representation, using additional contexts or different features may lead to better performance on stance detection. Finally, the model is versatile enough we could consider treating different tweets as different "contexts" for the same user, augmenting the extensively annotated tweets with other types of data, and, naturally, applying the same framework to other annotation tasks.

References

Pablo Barberá. 2016. Less is more? How demographic sample weights can improve public opinion estimates based on Twitter data. *Working Paper*.

Yevgeni Berzak, Yan Huang, Andrei Barbu, Anna Korhonen, and Boris Katz. 2016. Anchoring and Agreement in Syntactic Annotations. *arXiv preprint arXiv:1605.04481*.

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic dialectal variation in social media: A case study of African-American English. *EMNLP'16*.

John D. Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating Gender on Twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1301–1309, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jordan Carpenter, Daniel Preoţiuc-Pietro, Lucie Flekova, Salvatore Giorgi, Courtney Hagan, Margaret Kern, Anneke E. K. Buffone, Lyle Ungar, and Martin E. P. Seligman. 2016. Real Men Don't Say "Cute": Using Automatic Language Analysis to Isolate Inaccurate Aspects of Stereotypes. *Social Psychological and Personality Science*.

Xin Chen, Yu Wang, Eugene Agichtein, and Fusheng Wang. 2015. A Comparative Study of Demographic Attribute Inference in Twitter. *ICWSM*, 15:590–593.

Raviv Cohen and Derek Ruths. 2013. Classifying political orientation on Twitter: It's not easy! In *ICWSM*.

Peng Dai, Christopher H. Lin, Mausam, and Daniel S. Weld. 2013. POMDP-based control of workflows for crowdsourcing. *Artificial Intelligence*, 202:52–85.

Paul DiMaggio. 2015. Adapting computational text analysis to social science (and vice versa). *Big Data & Society*, 2(2).

Javid Ebrahimi, Dejing Dou, and Daniel Lowd. 2016. A Joint Sentiment-Target-Stance Model for Stance Classification in Tweets. *COLING'16*.

Will Frankenstein, Kenneth Joseph, and K. M. Carley. 2016. Contextualized Sentiment Analysis. In *Social Computing, Behavioral-Cultural Modeling and Prediction*, Washington, DC, USA.

Melody Y. Guan, Varun Gulshan, Andrew M. Dai, and Geoffrey E. Hinton. 2017. Who Said What: Modeling Individual Labelers Improves Classification. *arXiv:1703.08774 [cs]*.

William Hobbs, Lisa Friedland, Kenneth Joseph, Oren Tsur, Stefan Wojcik, and David Lazer. 2017. "Voters of the Year": 19 Voters Who Were Unintentional Election Poll Sensors on Twitter. In *ICWSM*.

C. J. Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International AAAI Conference on Weblogs and Social Media*.

Kenneth Joseph, Wei Wei, and Kathleen M. Carley. 2017. Girls rule, boys drool: Extracting semantic and affective stereotypes from Twitter. In *2017 ACM Conference on Computer Supported Cooperative Work.(CSCW)*.

Saif M. Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016a. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the International Workshop on Semantic Evaluation, SemEval*, volume 16.

Saif M. Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2016b. Stance and sentiment in tweets. *arXiv preprint arXiv:1605.01655*.

Fred Morstatter and Huan Liu. 2016. Replacing mechanical turkers? challenges in the evaluation of models with semantic properties. *Journal of Data and Information Quality (JDIQ)*, 7(4):15.

An Thanh Nguyen, Matthew Halpern, Byron C. Wallace, and Matthew Lease. 2016. Probabilistic modeling for crowdsourcing partially-subjective ratings. In *Proceedings of The Conference on Human Computation and Crowdsourcing (HCOMP)*, pages 149–158.

Dong Nguyen, Rilana Gravel, Dolf Trieschnigg, and Theo Meder. 2013. "How Old Do You Think I Am?": A Study of Language and Age in Twitter. In *Seventh International AAAI Conference on Weblogs and Social Media*.

Dong-Phuong Nguyen, R. B. Trieschnigg, A. Seza Doğruöz, Rilana Gravel, Mariët Theune, Theo Meder, and F. M. G. de Jong. 2014. Why gender and age prediction from tweets is hard: Lessons from a crowdsourcing experiment. In *Proceedings of COLING 2014*.

Rebecca J. Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics*, 2:311–326.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *Journal of Machine Learning Research*, 11(Apr):1297–1322.

Nathan Schneider. 2015. What I've learned about annotating informal text (and why you shouldn't take my word for it). In *The 9th Linguistic Annotation Workshop Held in Conjuncion with NAACL 2015*, page 152.

Nathan Schneider, Brendan O'Connor, Naomi Saphra, David Bamman, Manaal Faruqui, Noah A. Smith, Chris Dyer, and Jason Baldridge. 2013. A framework for (under) specifying dependency syntax without overloading annotators. *arXiv preprint arXiv:1306.2091*.

Shilad Sen, Isaac L. Johnson, Rebecca Harper, Huy Mai, Samuel Horlbeck Olsen, Benjamin Mathers, Laura Souza Vonessen, Matthew Wright, and Brent J. Hecht. 2015. Towards Domain-Specific Semantic Relatedness: A Case Study from Geography. In *IJCAI*, pages 2362–2370.

Yuandong Tian and Jun Zhu. 2012. Learning from crowds in the presence of schools of thought. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–234. ACM.

Fabian L. Wauthier and Michael I. Jordan. 2011. Bayesian bias mitigation for crowdsourcing. In *Advances in Neural Information Processing Systems*, pages 1800–1808.

Deeper Attention to Abusive User Content Moderation

John Pavlopoulos
Straintek, Athens, Greece
ip@straintek.com

Prodromos Malakasiotis
Straintek, Athens, Greece
mm@straintek.com

Ion Androutsopoulos
Athens University of Economics and Business, Greece
ion@aueb.gr

Abstract

Experimenting with a new dataset of 1.6M user comments from a news portal and an existing dataset of 115K Wikipedia talk page comments, we show that an RNN operating on word embeddings outpeforms the previous state of the art in moderation, which used logistic regression or an MLP classifier with character or word n-grams. We also compare against a CNN operating on word embeddings, and a word-list baseline. A novel, deep, classification-specific attention mechanism improves the performance of the RNN further, and can also highlight suspicious words for free, without including highlighted words in the training data. We consider both fully automatic and semi-automatic moderation.

1 Introduction

User comments play a central role in social media and online discussion fora. News portals and blogs often also allow their readers to comment to get feedback, engage their readers, and build customer loyalty.[1] User comments, however, and more generally user content can also be abusive (e.g., bullying, profanity, hate speech) (Cheng et al., 2015). Social media are under pressure to combat abusive content, but so far rely mostly on user reports and tools that detect frequent words and phrases of reported posts.[2] Wulczyn et al. (2017) estimated that only 17.9% of personal attacks in Wikipedia discussions were followed by moderator actions. News portals also suffer from abusive user comments, which damage their reputations and make them liable to fines, e.g., when hosting comments encouraging illegal actions. They often employ moderators, who are frequently overwhelmed, however, by the volume and abusiveness of comments.[3] Readers are disappointed when non-abusive comments do not appear quickly online because of moderation delays. Smaller news portals may be unable to employ moderators, and some are forced to shut down their comments sections entirely.

We examine how deep learning (Goodfellow et al., 2016; Goldberg, 2016, 2017) can be employed to moderate user comments. We experiment with a new dataset of approx. 1.6M manually moderated (accepted or rejected) user comments from a Greek sports news portal (called Gazzetta), which we make publicly available.[4] This is one of the largest publicly available datasets of moderated user comments. We also provide word embeddings pre-trained on 5.2M comments from the same portal. Furthermore, we experiment on the 'attacks' dataset of Wulczyn et al. (2017), approx. 115K English Wikipedia talk page comments labeled as containing personal attacks or not.

In a fully automatic scenario, there is no moderator and a system accepts or rejects comments. Although this scenario may be the only available one, e.g., when news portals cannot afford moderators, it is unrealistic to expect that fully automatic moderation will be perfect, because abusive comments may involve irony, sarcasm, harassment without profane phrases etc., which are particularly difficult for a machine to detect. When moderators are available, it is more realistic to develop semi-

[1] See, for example, http://niemanreports.org/articles/the-future-of-comments/.

[2] Consult, for example, https://www.facebook.com/help/131671940241729 and https://www.theguardian.com/technology/2017/feb/07/twitter-abuse-harassment-crackdown.

[3] See, e.g., https://www.wired.com/2017/04/zerochaos-google-ads-quality-raters and https://goo.gl/89M2bI.

[4] The portal is http://www.gazzetta.gr/. Instructions to download the dataset will become available at http://nlp.cs.aueb.gr/software.html.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1125–1135
Copenhagen, Denmark, September 7–11, 2017.

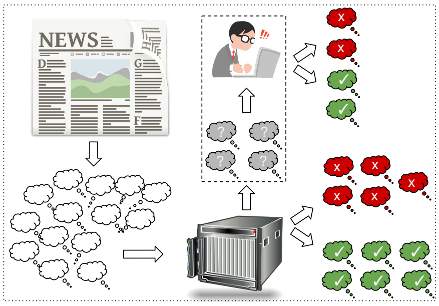


Figure 1: Semi-automatic moderation.

Dataset/Split	Accepted	Rejected	Total
G-TRAIN-L	960,378 (66%)	489,222 (34%)	1.45M
G-TRAIN-S	67,828 (68%)	32,172 (32%)	100,000
G-DEV	20,236 (68%)	9,464 (32%)	29,700
G-TEST-L	20,064 (68%)	9,636 (32%)	29,700
G-TEST-S	1,068 (71%)	432 (29%)	1,500
G-TEST-S-R	1,174 (78%)	326 (22%)	1,500
W-ATT-TRAIN	61,447 (88%)	8,079 (12%)	69,526
W-ATT-DEV	20,405 (88%)	2,755 (12%)	23,160
W-ATT-TEST	20,422 (88%)	2,756 (12%)	23,178

Table 1: Statistics of the datasets used.

automatic systems aiming to assist, rather than replace the moderators, a scenario that has not been considered in previous work. In this case, comments for which the system is uncertain (Fig. 1) are shown to a moderator to decide; all other comments are accepted or rejected by the system. We discuss how moderation systems can be tuned, depending on the availability and workload of the moderators. We also introduce additional evaluation measures for the semi-automatic scenario.

On both datasets (Gazzetta and Wikipedia comments) and for both scenarios (automatic, semi-automatic), we show that a recurrent neural network (RNN) outperforms the system of Wulczyn et al. (2017), the previous state of the art for comment moderation, which employed logistic regression or a multi-layer Perceptron (MLP), and represented each comment as a bag of (character or word) n-grams. We also propose an attention mechanism that improves the overall performance of the RNN. Our attention mechanism differs from most previous ones (Bahdanau et al., 2015; Luong et al., 2015) in that it is used in a classification setting, where there is no previously generated output subsequence to drive the attention, unlike sequence-to-sequence models (Sutskever et al., 2014). In that sense, our attention is similar to that of of Yang et al. (2016), but our attention mechanism is a deeper MLP and it is only applied to words, whereas Yang et al. also have a second attention mechanism that assigns attention scores to entire sentences. In effect, our attention detects the words of a comment that affect most the classification decision (accept, reject), by examining them in the context of the particular comment.

Although our attention mechanism does not always improve the performance of the RNN, it has the additional advantage of allowing the RNN to highlight suspicious words that a moderator could consider to decide more quickly if a comment should be accepted or rejected. The highlighting comes for free, i.e., the training data do not contain highlighted words. We also show that words highlighted by the attention mechanism correlate well with words that moderators would highlight.

Our main contributions are: (i) We release a dataset of 1.6M moderated user comments. (ii) We introduce a novel, deep, classification-specific attention mechanism and we show that an RNN with our attention mechanism outperforms the previous state of the art in user comment moderation. (iii) Unlike previous work, we also consider a semi-automatic scenario, along with threshold tuning and evaluation measures for it. (iv) We show that the attention mechanism can automatically highlight suspicious words for free, without manually highlighting words in the training data.

2 Datasets

We first discuss the datasets we used, to help acquaint the reader with the problem.

2.1 Gazzetta comments

There are approx. 1.45M training comments (covering Jan. 1, 2015 to Oct. 6, 2016) in the Gazzetta dataset; we call them G-TRAIN-L (Table 1). Some experiments use only the first 100K comments of G-TRAIN-L, called G-TRAIN-S. An additional set of 60,900 comments (Oct. 7 to Nov. 11, 2016) was split to development (G-DEV, 29,700 comments), large test (G-TEST-L, 29,700), and small test set (G-TEST-S, 1,500). Gazzetta's moderators (2 full-time, plus journalists occasionally helping) are occasionally instructed to be stricter (e.g., during violent events). To get a more accurate view of performance in normal situtations, we manually re-moderated (labeled as 'accept' or 'reject') the comments of G-TEST-S, producing G-TEST-S-R. The reject ratio is approx. 30% in all subsets, except for G-TEST-S-R where it drops to 22%, because there are no occasions where the moderators were instructed to be stricter in G-TEST-S-R.

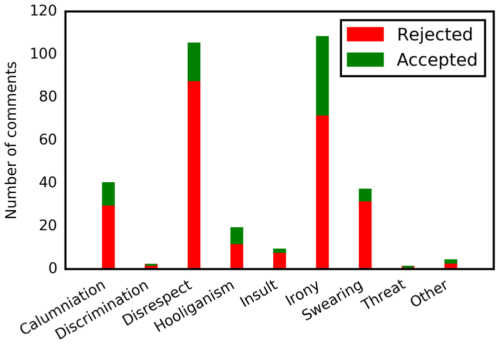


Figure 2: Re-moderated comments with at least one snippet of the corresponding category.

Each G-TEST-S-R comment was re-moderated by five annotators. Krippendorff's (2004) alpha was 0.4762, close to the value (0.45) reported by Wulczyn et al. (2017) for the Wikipedia 'attacks' dataset. Using Cohen's Kappa (Cohen, 1960), the mean pairwise agreement was 0.4749. The mean pairwise percentage of agreement (% of comments each pair of annotators agreed on) was 81.33%. Cohen's Kappa and Krippendorff's alpha lead to lower scores, because they account for agreement by chance, which is high when there is class imbalance (22% reject, 78% accept in G-TEST-S-R).

During the re-moderation of G-TEST-S-R, the annotators were also asked to highlight snippets they considered suspicious, i.e., words or phrases that could lead a moderator to consider rejecting each comment.[5] We also asked the annotators to classify each snippet into one of the following categories: calumniation (e.g., false accusations), discrimination (e.g., racism), disrespect (e.g., looking down at a profession), hooliganism (e.g., calling for violence), insult (e.g., making fun of appearance), irony, swearing, threat, other. Figure 2 shows how many comments of G-TEST-S-R contained at least one snippet of each category, according to the majority of annotators; e.g., a comment counts as containing irony if at least 3 annotators annotated it with an irony snippet (not necessarily the same). The gold class of each comment (accept or reject) is determined by the majority of the annotators. Irony and disrespect are particularly frequent in both classes, followed by calumniation, swearing, hooliganism, insults. Notice that comments that contain irony, disrespect etc. are not necessarily rejected. They are, however, more likely in the rejected class, considering that the accepted comments are 2.5 times more than the rejected ones (78% vs. 22%).

We also provide 300-dimensional word embeddings, pre-trained on approx. 5.2M comments (268M tokens) from Gazzetta using WORD2VEC (Mikolov et al., 2013a,b).[6] This larger dataset cannot be used to directly train classifiers, because most of its comments are from a period (before 2015) when Gazzetta did not employ moderators.

2.2 Wikipedia comments

The Wikipedia 'attacks' dataset (Wulczyn et al., 2017) contains approx. 115K English Wikipedia talk page comments, which were labeled as containing personal attacks or not. Each comment was labeled by at least 10 annotators. Inter-annotator agreement, measured on a random sample of 1K comments using Krippendorff's (2004) alpha, was 0.45. The gold label of each comment is determined by the majority of annotators, leading to *binary labels* (accept, reject). Alternatively, the gold label is the percentage of annotators that labeled the comment as 'accept' (or 'reject'), leading to *probabilistic labels*.[7] The dataset is split in three parts (Table 1): training (W-ATT-TRAIN, 69,526 comments), development (W-ATT-DEV, 23,160), and test (W-ATT-TEST, 23,178). In all three parts, the rejected comments are 12%, but this is an artificial ratio (Wulczyn et al. oversampled comments posted by banned users). By contrast, the ratio of rejected comments in all the Gazzetta subsets is the truly observed one. The Wikipedia comments are also longer (median length 38 tokens) compared to Gazzetta's (median length 25 tokens).

Wulczyn et al. (2017) also provide two additional datasets of English Wikipedia talk page comments, which are not used in this paper. The first one, called 'aggression' dataset, contains the same comments as the 'attacks' dataset, now labeled as 'aggressive' or not. The (probabilistic) labels of the 'attacks' and 'aggression' datasets are very highly correlated (0.8992 Spearman, 0.9718 Pearson) and we did not consider the aggression dataset any further. The second additional dataset, called 'toxicity' dataset, contains approx. 160K comments labeled as being toxic or not. Experiments we reported elsewhere (Pavlopoulos et al., 2017) show that results on the 'attacks' and 'toxicity' datasets are very similar; we do not include

[5] Treating snippet overlaps as agreements, the mean pairwise Dice coefficient for snippet highlighting was 50.03%.

[6] We used CBOW, window size 5, min. term freq. 5, negative sampling, obtaining a vocabulary size of approx. 478K.

[7] We also construct probabilistic labels for G-TEST-S-R, where there are five annotators.

results on the latter in this paper to save space.

3 Methods

We experimented with an RNN operating on word embeddings, the same RNN enhanced with our attention mechanism (a-RNN), a vanilla convolutional neural network (CNN) also operating on word embeddings, the DETOX system of Wulczyn et al. (2017), and a baseline that uses word lists.

3.1 DETOX

DETOX (Wulczyn et al., 2017) was the previous state of the art in comment moderation, in the sense that it had the best reported results on the Wikipedia datasets (Section 2.2), which were in turn the largest previous publicly available dataset of moderated user comments.[8] DETOX represents each comment as a bag of word n-grams ($n \leq 2$, each comment becomes a bag containing its 1-grams and 2-grams) or a bag of character n-grams ($n \leq 5$, each comment becomes a bag containing character 1-grams, …, 5-grams). DETOX can rely on a logistic regression (LR) or MLP classifier, and it can use binary or probabilistic gold labels (Section 2.2) during training.

We used the DETOX implementation provided by Wulczyn et al. and the same grid search (and code) to tune the hyper-parameters of DETOX that select word or character n-grams, classifier (LR or MLP), and gold labels (binary or probabilistic). For Gazzetta, only binary gold labels were possible, since G-TRAIN-L and G-TRAIN-S have a single gold label per comment. Unlike Wulczyn et al., we tuned the hyper-parameters by evaluating (computing AUC and Spearman, Section 4) on a random 2% of held-out comments of W-ATT-TRAIN or G-TRAIN-S, instead of the development subsets, to be able to obtain more realistic results from the development sets while developing the methods. For both Wikipedia and Gazzetta, the tuning selected character n-grams, as in the work of Wulczyn et al. Also, for both Wikipedia and Gazzetta, it preferred LR to MLP, whereas Wulczyn et al. reported slightly higher performance for the MLP on W-ATT-DEV.[9] The tuning also selected probabilistic labels for Wikipedia, as in the work of Wulczyn et al.

3.2 RNN-based methods

RNN: The RNN method is a chain of GRU cells (Cho et al., 2014) that transforms the tokens $w_1 \ldots, w_k$ of each comment to the hidden states $h_1 \ldots, h_k$, followed by an LR layer that uses h_k to classify the comment (accept, reject). Formally, given the vocabulary V, a matrix $E \in \mathbb{R}^{d \times |V|}$ containing d-dimensional word embeddings, an initial h_0, and a comment $c = \langle w_1, \ldots, w_k \rangle$, the RNN computes $h_1, \ldots, h_k$ as follows ($h_t \in \mathbb{R}^m$):

$$\begin{aligned}
\tilde{h}_t &= \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \\
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r)
\end{aligned}$$

where $\tilde{h}_t \in \mathbb{R}^m$ is the proposed hidden state at position t, obtained by considering the word embedding x_t of token w_t and the previous hidden state h_{t-1}; $\odot$ denotes element-wise multiplication; $r_t \in \mathbb{R}^m$ is the reset gate (for r_t all zeros, it allows the RNN to forget the previous state h_{t-1}); $z_t \in \mathbb{R}^m$ is the update gate (for z_t all zeros, it allows the RNN to ignore the new proposed $\tilde{h}_t$, hence also x_t, and copy h_{t-1} as h_t); σ is the sigmoid function; $W_h, W_z, W_r \in \mathbb{R}^{m \times d}$; $U_h, U_z, U_r \in \mathbb{R}^{m \times m}$; $b_h, b_z, b_r \in \mathbb{R}^m$. Once h_k has been computed, the LR layer estimates the probability that comment c should be rejected, with $W_p \in \mathbb{R}^{1 \times m}$, $b_p \in \mathbb{R}$:

$$P_{\text{RNN}}(\textit{reject}|c) \quad = \quad \sigma(W_p h_k + b_p)$$

***a*-RNN:** When the attention mechanism is added, the LR layer considers the weighted sum h_{sum} of all the hidden states, instead of just h_k (Fig. 3):[10]

$$\begin{aligned}
h_{sum} &= \sum_{t=1}^{k} a_t h_t \qquad (1) \\
P_{a-\text{RNN}}(\textit{reject}|c) &= \sigma(W_p h_{sum} + b_p)
\end{aligned}$$

The weights a_t are produced by an attention mech-

[8] Two of the co-authors of Wulczyn et al. (2017) are with Jigsaw, who recently announced Perspective, a system to detect ‘toxic’ comments. Perspective is not the same as DETOX (personal communication), but we were unable to obtain scientific articles describing it. An API for Perspective is available at `https://www.perspectiveapi.com/`, but we did not have access to the API at the time the experiments of this paper were carried out.

[9] We repeated the tuning by evaluating on W-ATT-DEV, and again character n-grams with LR were selected.

[10] We tried replacing the LR layer by a deeper classification MLP, and the RNN chain by a bidirectional RNN (Schuster and Paliwal, 1997), but there were no improvements.

anism, which is an MLP with l layers:

$$\begin{aligned} a_t^{(1)} &= \text{RELU}(W^{(1)} h_t + b^{(1)}) \qquad (2) \\ &\ldots \\ a_t^{(l-1)} &= \text{RELU}(W^{(l-1)} a_t^{(l-2)} + b^{(l-1)}) \\ a_t^{(l)} &= W^{(l)} a_t^{(l-1)} + b^{(l)} \\ a_t &= \mathtt{softmax}(a_t^{(l)}; a_1^{(l)}, \ldots, a_k^{(l)}) \ (3) \end{aligned}$$

where $a_t^{(1)}, \ldots, a_t^{(l-1)} \in \mathbb{R}^r$, $a_t^{(l)}, a_t \in \mathbb{R}$, $W^{(1)} \in \mathbb{R}^{r \times m}$, $W^{(2)}, \ldots, W^{(l-1)} \in \mathbb{R}^{r \times r}$, $W^{(l)} \in \mathbb{R}^{1 \times r}$, $b^{(1)}, \ldots, b^{(l-1)} \in \mathbb{R}^r$, $b^{(l)} \in \mathbb{R}$. The softmax operates across the $a_t^{(l)}$ ($t = 1, \ldots, k$), making the weights a_t sum to 1. Our attention mechanism differs from most previous ones (Mnih et al., 2014; Bahdanau et al., 2015; Xu et al., 2015; Luong et al., 2015) in that it is used in a classification setting, where there is no previously generated output subsequence (e.g., partly generated translation) to drive the attention (e.g., assign more weight to source words to translate next), unlike seq2seq models (Sutskever et al., 2014). It assigns larger weights a_t to hidden states h_t corresponding to positions where there is more evidence that the comment should be accepted or rejected.

Yang et al. (2016) use a similar attention mechanism, but ours is deeper. In effect they always set $l = 2$, whereas we allow l to be larger (tuning selects $l = 4$).[11] On the other hand, the attention mechanism of Yang et al. is part of a classification method for longer texts (e.g., product reviews). Their method uses two GRU RNNs, both bidirectional (Schuster and Paliwal, 1997), one turning the word embeddings of each sentence to a sentence embedding, and one turning the sentence embeddings to a document embedding, which is then fed to an LR layer. Yang et al. use their attention mechanism in both RNNs, to assign attention scores to words and sentences. We consider shorter texts (comments), we have a single RNN, and we assign attention scores to words only.[12]

***da*-CENT:** We also experiment with a variant of *a*-RNN, called *da*-CENT, which does not use the hidden states of the RNN. The input to the first layer of the attention mechanism is now directly the embedding x_t instead of h_t (cf. Eq. 2), and

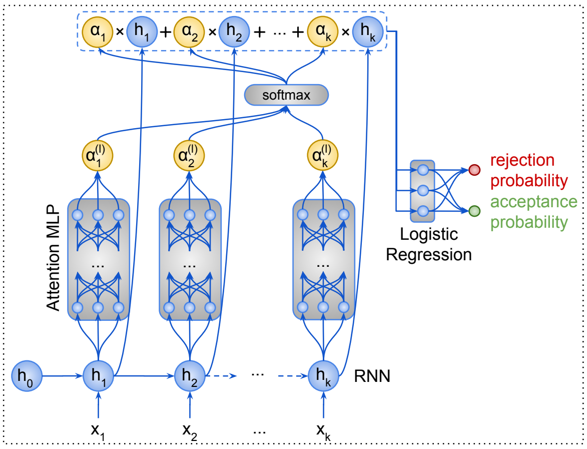


Figure 3: Illustration of *a*-RNN.

h_{sum} is now the weighted sum (centroid) of word embeddings $h_{sum} = \sum_{t=1}^{k} a_t x_t$ (cf. Eq. 1).[13]

We set $l = 4, d = 300, r = m = 128$, having tuned all hyper-parameters on the same 2% held-out comments of W-ATT-TRAIN or G-TRAIN-S that were used to tune DETOX. We use Glorot initialization (Glorot and Bengio, 2010), categorical cross-entropy loss, and Adam (Kingma and Ba, 2015).[14] Early stopping evaluates on the same held-out subsets. For Gazzetta, word embeddings are initialized to the WORD2VEC embeddings we provide (Section 2.1). For Wikipedia, they are initialized to GLOVE embeddings (Pennington et al., 2014).[15] In both cases, the embeddings are updated during backpropagation. Out of vocabulary (OOV) words, meaning words for which we have no initial embeddings, are mapped to a single randomly initialized embedding, also updated.

3.3 CNN

We also compare against a vanilla CNN operating on word embeddings. We describe the CNN only briefly, because it is very similar to that of of Kim (2014); see also Goldberg (2016) for an introduction to CNNs, and Zhang and Wallace (2015).

For Wikipedia comments, we use a 'narrow' convolution layer, with kernels sliding (stride 1) over (entire) embeddings of word n-grams of sizes $n = 1, \ldots, 4$. We use 300 kernels for each n value, a total of 1,200 kernels. The outputs of each kernel, obtained by applying the kernel to the different n-grams of a comment c, are then

[11] Yang et al. use tanh instead of RELU in Eq. 2, which works worse in our case, and no bias $b^{(l)}$ in the l-th layer.

[12] We tried a bidirectional instead of unidirectional GRU chain in our methods, also replacing the LR layer by a deeper classification MLP, but there were no improvements.

[13] For experiments with additional variants of *a*-RNN, consult Pavlopoulos et al. (2017).

[14] We implemented the methods of this sub-section using Keras (`keras.io`) and TensorFlow (`tensorflow.org`).

[15] See `https://nlp.stanford.edu/projects/glove/`. We use 'Common Crawl' (840B tokens).

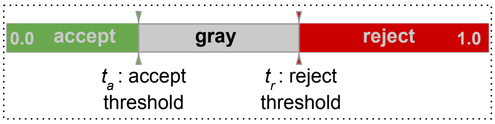


Figure 4: Illustration of threshold tuning.

max-pooled, leading to a single output per kernel. The resulting feature vector (1,200 max-pooled outputs) goes through a dropout layer (Hinton et al., 2012) ($p = 0.5$), and then to an LR layer, which provides $P_{\text{CNN}}(\textit{reject}|c)$. For Gazzetta, the CNN is the same, except that $n = 1, \dots, 5$, leading to 1,500 features per comment. All hyper-parameters were tuned on the 2% held-out comments of W-ATT-TRAIN or G-TRAIN-S that were used to tune the other methods. Again, we use 300-dimensional embeddings, which are now randomly initialized, since tuning indicated this was better than initializing to pre-trained embeddings. OOV words are treated as in the RNN-based methods. All embeddings are updated during back-propagation. Early stopping evaluates on the held-out subsets. Again, we use Glorot initialization, categorical cross-entropy loss, and Adam.[16]

3.4 LIST baseline

A baseline, called LIST, collects every word w that occurs in more than 10 (for W-ATT-TRAIN, G-TRAIN-S) or 100 comments (for G-TRAIN-L) in the training set, along with the precision of w, i.e., the ratio of rejected training comments containing w divided by the total number of training comments containing w. The resulting lists contain 10,423, 16,864, and 21,940 word types, when using W-ATT-TRAIN, G-TRAIN-S, G-TRAIN-L, respectively. For a comment c, LIST returns as $P_{\text{LIST}}(\textit{reject}|c)$ the maximum precision of all the words in c.

3.5 Tuning thresholds

All methods produce a $p = P(\textit{reject}|c)$ per comment c. In semi-automatic moderation (Fig. 1), a comment is directly rejected if its p is above a rejection theshold t_r, it is directly accepted if p is below an acceptance threshold t_a, and it is shown to a moderator if $t_a \leq p \leq t_r$ (gray zone of Fig. 4).

In our experience, moderators (or their employers) can easily specify the approximate percentage of comments they can afford to check manually (e.g., 20% daily) or, equivalently, the approximate percentage of comments the system should handle automatically. We call *coverage* the latter percentage; hence, $1 - \textit{coverage}$ is the approximate percentage of comments to be checked manually. By contrast, moderators are baffled when asked to tune t_r and t_a directly. Consequently, we ask them to specify the approximate desired coverage. We then sort the comments of the development set (G-DEV or W-ATT-DEV) by p, and slide t_a from 0.0 to 1.0 (Fig. 4). For each t_a value, we set t_r to the value that leaves a $1 - \textit{coverage}$ percentage of development comments in the gray zone ($t_a \leq p \leq t_r$). We then select the t_a (and t_r) that maximizes the weighted harmonic mean $F_\beta(P_{\textit{reject}}, P_{\textit{accept}})$ on the development set:

$$F_\beta(P_{\textit{reject}}, P_{\textit{accept}}) = \frac{(1+\beta^2) \cdot P_{\textit{reject}} \cdot P_{\textit{accept}}}{\beta^2 \cdot P_{\textit{reject}} + P_{\textit{accept}}}$$

where $P_{\textit{reject}}$ is the *rejection precision* (correctly rejected comments divided by rejected comments) and $P_{\textit{accept}}$ is the *acceptance precision* (correctly accepted divided by accepted). Intuitively, coverage sets the width of the gray zone, whereas $P_{\textit{reject}}$ and $P_{\textit{accept}}$ show how certain we can be that the red (reject) and green (accept) zones are free of misclassified comments. We set $\beta = 2$, emphasizing $P_{\textit{accept}}$, because moderators are more worried about wrongly accepting abusive comments than wrongly rejecting non-abusive ones.[17] The selected t_a, t_r (tuned on development data) are then used in experiments on test data. In fully automatic moderation, $\textit{coverage} = 100$ and $t_a = t_r$; otherwise, threshold tuning is identical.

4 Experimental results

4.1 Comment classification evaluation

Following Wulczyn et al. (2017), we report in Table 2 AUC scores (area under ROC curve), along with Spearman correlations between system-generated probabilities $P(\textit{accept}|c)$ and human probabilistic gold labels (Section 2.2) when probabilistic gold labels are available.[18] Wulczyn et al. reported DETOX results only on W-ATT-DEV, shown in brackets. Table 2 shows that RNN is

[16] We implemented the CNN directly in TensorFlow.

[17] More precisely, when computing F_β, we reorder the development comments by time posted, and split them into batches of 100. For each t_a (and t_r) value, we compute F_β per batch and macro-average across batches. The resulting thresholds lead to F_β scores that are more stable over time.

[18] When computing AUC, the gold label is the majority label of the annotators. When computing Spearman, the gold label is probabilistic (% of annotators that accepted the comment). The decisions of the systems are always probabilistic.

Training	Evaluation	Score	RNN	*a*-RNN	*da*-CENT	CNN	DETOX	LIST
G-TRAIN-S	G-DEV	AUC	75.75	**76.19**	74.91	70.97	72.50	61.47
	G-TEST-L	AUC	75.10	**76.15**	74.72	71.34	72.06	61.59
	G-TEST-S	AUC	74.40	**75.83**	73.79	70.88	71.59	61.26
	G-TEST-S-R	AUC	80.27	**80.41**	78.82	76.03	75.67	64.19
		Spearman	51.89	**52.51**	49.22	42.88	43.80	24.33
G-TRAIN-L	G-DEV	AUC	79.50	**79.64**	78.73	77.57	–	67.04
	G-TEST-L	AUC	79.41	**79.58**	78.64	77.35	–	67.06
	G-TEST-S	AUC	79.23	**79.67**	78.62	78.16	–	66.17
	G-TEST-S-R	AUC	84.17	**84.69**	83.53	83.98	–	69.51
		Spearman	59.31	**60.87**	57.82	55.90	–	33.61
W-ATT-TRAIN	W-ATT-DEV	AUC	97.39	**97.46**	96.58	96.91	96.26 (96.59)	93.05
		Spearman	**71.92**	71.59	68.59	70.06	67.75 (68.17)	55.39
	W-ATT-TEST	AUC	**97.71**	97.68	96.83	97.07	96.71	92.91
		Spearman	**72.79**	72.32	68.86	70.21	68.09	54.55

Table 2: Comment classification results. Scores reported by Wulczyn et al. (2017) are shown in brackets.

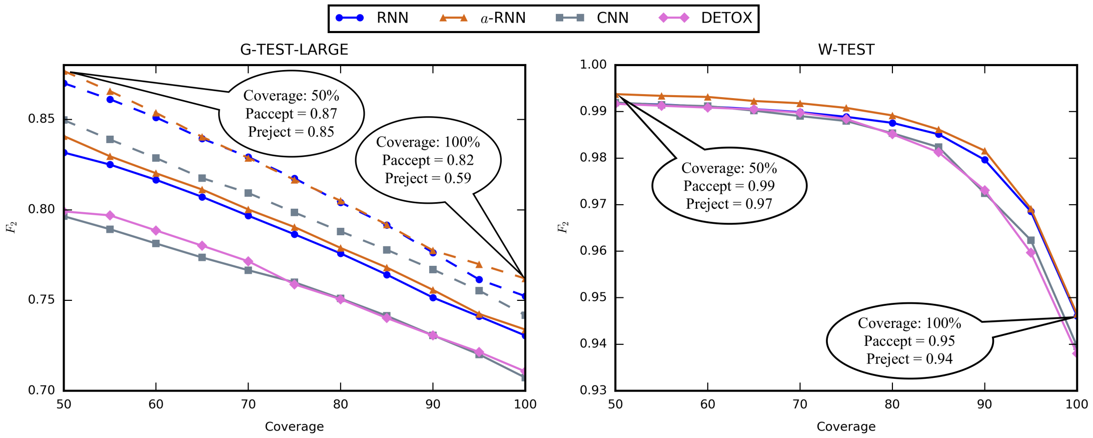
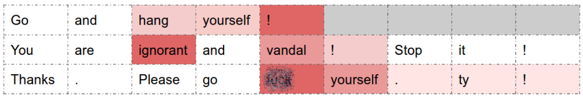


Figure 5: F_2 scores for varying coverage. Dotted lines were obtained using a larger training set.

always better than CNN and DETOX; there is no clear winner between CNN and DETOX. Furthermore, *a*-RNN is always better than RNN on Gazzetta comments, but not on Wikipedia comments, where RNN is overall slightly better according to Table 2. Also, *da*-CENT is always worse than *a*-RNN and RNN, confirming that the hidden states (intuitively, context-aware word embeddings) of the RNN chain are important, even with the attention mechanism. Increasing the size of the Gazzetta training set (G-TRAIN-S to G-TRAIN-L) significantly improves the performance of all methods. The implementation of DETOX could not handle the size of G-TRAIN-L, which is why we do not report DETOX results for G-TRAIN-L. Notice, also, that the Wikipedia dataset is easier than the Gazzetta one (all methods perform better on Wikipedia comments, compared to Gazzetta).

Figure 5 shows $F_2(P_{reject}, P_{accept})$ on G-TEST-L and W-ATT-TEST, when t_a, t_r are tuned on G-DEV, W-ATT-DEV for varying coverage. For G-TEST-L, we show results training on G-TRAIN-S (solid lines) and G-TRAIN-L (dotted). The differences between RNN and *a*-RNN are again small, but it is now easier to see that *a*-RNN is overall better. Again, *a*-RNN and RNN are better than CNN and DETOX. All three deep learning methods benefit from the larger training set (dotted). In Wikipedia, *a*-RNN obtains $P_{accept}, P_{reject} \geq 0.94$ for all coverages (Fig. 5, call-outs). On the more difficult Gazzetta dataset, *a*-RNN still obtains $P_{accept}, P_{reject} \geq 0.85$ when tuned for 50% coverage. When tuned for 100% coverage, comments for which the system is uncertain (gray zone) cannot be avoided and there are inevitably more misclassifications; the use of F_2 during threshold tuning places more emphasis on avoiding wrongly accepted comments, leading to high P_{accept} (0.82), at the expense of wrongly rejected comments, i.e., sacrificing P_{reject} (0.59). On the re-moderated G-TEST-S-R (similar diagrams, not shown), P_{accept}, P_{reject} become 0.96, 0.88 for coverage 50%, and 0.92, 0.48 for coverage 100%.

We also repeated the *annotator ensemble* experiment of Wulczyn et al. (2017) on 8K randomly chosen comments of W-ATT-TEST (4K comments



Go	and	hang	yourself	!				
You	are	ignorant	and	vandal	!	Stop	it	!
Thanks	.	Please	go	[redacted]	yourself	.	ty	!

Figure 6: Word highlighting by *a*-RNN.

from random users, 4K comments from banned users).[19] The decisions of 10 randomly chosen annotators (possibly different per comment) were used to construct the gold label of each comment. The gold labels were then compared to the decisions of the systems and the decisions of an ensemble of k other annotators, k ranging from 1 to 10. Table 3 shows the mean AUC and Spearman scores, averaged over 25 runs of the experiment, along with standard errrors (in brackets). We conclude that RNN and *a*-RNN are as good as an ensemble of 7 human annotators; CNN is as good as 4 annotators; DETOX is as good as 4 in AUC and 3 annotators in Spearman correlation, which is consistent with the results of Wulczyn et al. (2017).

k	AUC	Spearman
1	84.34 (0.64)	53.82 (0.77)
2	92.14 (0.42)	61.61 (0.51)
3	95.05 (0.41)	65.20 (0.55)
4	96.43 (0.31)	67.25 (0.52)
5	97.17 (0.25)	68.46 (0.49)
6	97.68 (0.22)	69.45 (0.40)
7	97.99 (0.21)	70.16 (0.33)
8	98.21 (0.18)	70.67 (0.31)
9	98.39 (0.15)	71.12 (0.32)
10	98.51 (0.14)	71.50 (0.34)
RNN	98.03 (0.13)	70.58 (0.27)
a-RNN	98.00 (0.13)	70.19 (0.30)
CNN	97.29 (0.14)	67.92 (0.32)
DETOX	97.00 (0.14)	66.21 (0.32)

Table 3: Comparing to an ensemble of k humans.

4.2 Snippet highlighting evaluation

To investigate if the attention scores of *a*-RNN can highlight suspicious words, we focused on G-TEST-S-R, the only dataset with suspicious snippets annotated by humans. We removed comments with no human-annotated snippets, leaving 841 comments (515 accepted, 326 rejected), a total of 40,572 tokens, of which 13,146 were inside a suspicious snippet of at least one annotator. In each remaining comment, each token was assigned a *gold suspiciousness* score, defined as the percentage of annotators that included it in their snippets.

We evaluated three methods that score each token w_t of a comment c for suspiciousness. The first one assigns to each w_t the attention score a_t (Eq. 3) of *a*-RNN (trained on G-TRAIN-L). The second method assigns to each w_t its precision, as computed by LIST (Section 3.4). The third method (RAND) assigns to each w_t a random (uniform distribution) score between 0 and 1. In the latter two methods, a `softmax` is applied to the scores of all the tokens per comment, as in *a*-RNN. Figure 6 shows three comments (from W-ATT-TEST) highlighted by *a*-RNN; heat corresponds to attention.[20]

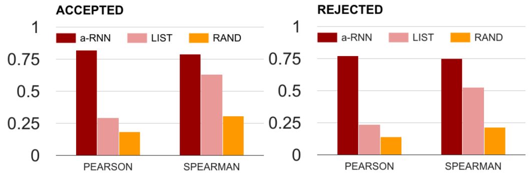


Figure 7: Suspicious snippet highlighting results.

We computed Pearson and Spearman correlations between the gold suspiciousness scores and the scores of the three methods on the 40,572 tokens. Figure 7 shows the correlations on comments that were accepted (left) and rejected (right) by the majority of moderators. In both cases, *a*-RNN performs better than LIST and RAND by both Pearson and Spearman correlations. The high Pearson correlations of *a*-RNN also show that its attention scores are to a large extent linearly related to the gold ones. By contrast, LIST performs reasonably well in terms of Spearman correlation, but much worse in terms of Pearson, indicating that its precision scores rank reasonably well the tokens from most to least suspicious ones, but are not linearly related to the gold scores.

5 Related work

Djuric et al. (2015) experimented with 952K manually moderated comments from Yahoo Finance, but their dataset is not publicly available. They convert each comment to a comment embedding using DOC2VEC (Le and Mikolov, 2014), which is then fed to an LR classifier. Nobata et al. (2016) experimented with approx. 3.3M manually moderated comments from Yahoo Finance and News; their data are also not available.[21] They used Vowpal Wabbit[22] with character n-grams ($n = 3, \ldots, 5$) and word n-grams ($n = 1, 2$), hand-crafted features (e.g., number of capitalized or black-listed words), features based on dependency

[19] We used the protocol, code, and data of Wulczyn et al.

[20] In innocent comments, *a*-RNN spreads its attention to all tokens, leading to quasi-uniform low color intensity.

[21] According to Nobata et al., their clean test dataset (2K comments) would be made available, but it is currently not.

[22] See http://hunch.net/~vw/.

trees, averages of WORD2VEC embeddings, and DOC2VEC-like embeddings. Character n-grams were the best, on their own outperforming Djuric et al. (2015). The best results, however, were obtained using all features. We use no hand-crafted features and parsers, making our methods more easily portable to other domains and languages.

Mehdad et al. (2016) train a (token or character-based) RNN language model per class (accept, reject), and use the probability ratio of the two models to accept or reject user comments. Experiments on the dataset of Djuric et al. (2015), however, showed that their method (RNNLMs) performed worse than a combination of SVM and Naive Bayes classifiers (NBSVM) that used character and token n-grams. An LR classifier operating on DOC2VEC-like comment embeddings (Le and Mikolov, 2014) also performed worse than NBSVM. To surpass NBSVM, Mehdad et al. used an SVM to combine features from their three other methods (RNNLMs, LR with DOC2VEC, NBSVM).

Wulczyn et al. (2017) experimented with character and word n-grams. We included their dataset and moderation system (DETOX) in our experiments. Waseem et al. (2016) used approx. 17K tweets annotated for hate speech. Their best results were obtained using an LR classifier with character n-grams ($n = 1, \dots, 4$), plus gender. Warner and Hirschberg (2012) aimed to detect anti-semitic speech, experimenting with 9K paragraphs and a linear SVM. Their features consider windows of at most 5 tokens, examining the tokens of each window, their order, POS tags, Brown clusters etc., following Yarowsky (1994).

Cheng et al. (2015) aimed to predict which users would be banned from on-line communities. Their best system used a random forest or LR classifier, with features examining readability, activity (e.g., number of posts daily), community and moderator reactions (e.g., up-votes, number of deleted posts).

Sood et al. (2012a; 2012b) experimented with 6.5K comments from Yahoo Buzz, moderated via crowdsourcing. They showed that a linear SVM, representing each comment as a bag of word bigrams and stems, performs better than word lists. Their best results were obtained by combining the SVM with a word list and edit distance.

Yin et al. (2009) used posts from chat rooms and discussion fora (<15K posts in total) to train an SVM to detect online harassment. They used TF-IDF, sentiment, and context features (e.g., similarity to other posts in a thread). Our methods might also benefit by considering threads, rather than individual comments. Yin at al. point out that unlike other abusive content, spam in comments or dicsussion fora (Mishne et al., 2005; Niu et al., 2007) is off-topic and serves a commercial purpose. Spam is unlikely in Wikipedia discussions and not an issue in the Gazzetta dataset (Fig. 2).

For a more extensive discussion of related work, consult Pavlopoulos et al. (2017).

6 Conclusions

We experimented with a new publicly available dataset of 1.6M moderated user comments from a Greek sports news portal and an existing dataset of 115K English Wikipedia talk page comments. We showed that a GRU RNN operating on word embeddings outpeforms the previous state of the art, which used an LR or MLP classifier with character or word n-gram features, also outperforming a vanilla CNN operating on word embeddings, and a baseline that uses an automatically constructed word list with precision scores. A novel, deep, classification-specific attention mechanism improves further the overall results of the RNN, and can also highlight suspicious words for free, without including highlighted words in the training data. We considered both fully automatic and semi-automatic moderation, along with threshold tuning and evaluation measures for both.

We plan to consider user-specific information (e.g., ratio of comments rejected in the past) (Cheng et al., 2015; Waseem and Hovy, 2016) and explore character-level RNNs or CNNs (Zhang et al., 2015), e.g., as a first layer to produce embeddings of unknown words from characters (dos Santos and Zadrozny, 2014; Ling et al., 2015), which would then be passed on to our current methods that operate on word embeddings.

Acknowledgments

This work was funded by Google's Digital News Initiative (project ML2P, contract 362826).[23] We are grateful to Gazzetta for the data they provided. We also thank Gazzetta's moderators for their feedback, insights, and advice.

[23] See https://digitalnewsinitiative.com/.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*. San Diego, CA, USA.

Justin Cheng, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2015. Antisocial behavior in online discussion communities. In *Proceedings of the 9th International AAAI Conference on Web and Social Media*. Oxford University, England, pages 61–70.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 1724–1734.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20(1):37–46.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*. Florence, Italy, pages 29–30.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning*. Beijing, China, pages 1818–1826.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*. Sardinia, Italy, pages 249–256.

Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research* 57:345–420.

Yoav Goldberg. 2017. *Neural Network Methods in Natural Language Processing*. Morgan and Claypool Publishers.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* abs/1207.0580.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*. San Diego, CA, USA.

Klaus Krippendorff. 2004. *Content Analysis: An Introduction to Its Methodology (2nd edition)*. Sage Publications.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*. Beijing, China, pages 1188–1196.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 1520–1530.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 1412–1421.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Los Angeles, CA, pages 299–303.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at International Conference on Learning Representations*. Scottsdale, AZ, USA.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, GA, pages 746–751.

Gilad Mishne, David Carmel, and Ronny Lempel. 2005. Blocking blog spam with language model disagreement. In *Proceedings of the 1st International Workshop on Adversarial Information Retrieval on the Web*. Chiba, Japan.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*. Montreal, Canada, pages 2204–2212.

Yuan Niu, Yi-Min Wang, Hao Chen, Ming Ma, and Francis Hsu. 2007. A quantitative study of forum spamming using context-based analysis. In *Proceedings of the 14th Annual Network and Distributed System Security Symposium*. San Diego, CA, pages 79–92.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*. Montreal, Canada, pages 145–153.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deep learning for user comment moderation. In *Proceedings of the 1st ACL Workshop on Abusive Language Online*. Vancouver, Canada.

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

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transacions of Signal Processing* 45(11):2673–2681.

Sara Sood, Judd Antin, and Elizabeth F. Churchill. 2012a. Profanity use in online communities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Austin, TX, USA, pages 1481–1490.

Sara Sood, Judd Antin, and Elizabeth F. Churchill. 2012b. Using crowdsourcing to improve profanity detection. In *AAAI Spring Symposium: Wisdom of the Crowd*. Stanford, CA, USA, pages 69–74.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. Montreal, Canada, pages 3104–3112.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the World Wide Web. In *Proceedings of the 2nd Workshop on Language in Social Media*. Montreal, Canada, pages 19–26.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*. San Diego, CA, USA, pages 88–93.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*. Perth, Australia, pages 1391–1399.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*. Lille, France, pages 2048–2057.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, CA, USA, pages 1480–1489.

David Yarowsky. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*. Las Cruces, NM, USA, pages 88–95.

Dawei Yin, Zhenzhen Xue, Liangjie Hong, Brian D Davison, April Kontostathis, and Lynne Edwards. 2009. Detection of harassment on Web 2.0. In *Proceedings of the WWW workshop on Content Analysis in the Web 2.0*. Madrid, Spain.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*. Montreal, Canada, pages 649–657.

Ye Zhang and Byron C. Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR* abs/1510.03820.

Outta Control: Laws of Semantic Change and Inherent Biases in Word Representation Models

Haim Dubossarsky[1], **Eitan Grossman**[2] and **Daphna Weinshall**[3]
[1] Edmond and Lily Safra Center for Brain Sciences
[2] Department of Linguistics
[3] School of Computer Science and Engineering
The Hebrew University of Jerusalem, 91904 Jerusalem, Israel
haim.dub@gmail.com, {eitan.grossman,daphna}@mail.huji.ac.il

Abstract

This article evaluates three proposed laws of semantic change. Our claim is that in order to validate a putative law of semantic change, the effect should be observed in the genuine condition but absent or reduced in a suitably matched control condition, in which no change can possibly have taken place. Our analysis shows that the effects reported in recent literature must be substantially revised: (i) the proposed negative correlation between meaning change and word frequency is shown to be largely an artefact of the models of word representation used; (ii) the proposed negative correlation between meaning change and prototypicality is shown to be much weaker than what has been claimed in prior art; and (iii) the proposed positive correlation between meaning change and polysemy is largely an artefact of word frequency. These empirical observations are corroborated by analytical proofs that show that count representations introduce an inherent dependence on word frequency, and thus word frequency cannot be evaluated as an independent factor with these representations.

1 Introduction

The increasing availability of digitized historical corpora, together with newly developed tools of computational analysis, make the quantitative study of language change possible on a larger scale than ever before. Thus, many important questions may now be addressed using a variety of NLP tools that were originally developed to study synchronic similarities between words. This has catalyzed the evolution of an exciting new field of *historical distributional semantics*, which has yielded findings that inform our understanding of the dynamic structure of language (Sagi et al., 2009; Wijaya and Yeniterzi, 2011; Mitra et al., 2014; Hilpert and Perek, 2015; Fremann and Lapata, 2016; Dubossarsky et al., 2016). Recent research has even proposed *laws of change* that predict the conditions under which the meaning of words is likely to change (Dubossarsky et al., 2015; Xu and Kemp, 2015; Hamilton et al., 2016). This is an important development, as traditional historical linguistics has generally been unable to provide predictive models of semantic change.

However, these preliminary results should be addressed with caution. To date, analyses of changes in words' meanings have relied on the comparison of word representations at different points in time. Thus any proposed change in meaning is contingent on a particular model of word representation and the method used to measure change. Distributional semantic models typically count words and their co-occurrence statistics (*explicit* models) or predict the embedding contexts of words (*implicit* models). In this paper, we show that the choice of model may introduce biases into the analysis. We therefore suggest that empirical findings may be used to support laws of semantic change only after a proper control can be shown to eliminate artefactual factors as the underlying cause of the empirical observations.

Regardless of the specific representation used, a frequent method of measuring the semantic change a word has undergone (Gulordava and Baroni, 2011; Jatowt and Duh, 2014; Kim et al., 2014; Dubossarsky et al., 2015; Kulkarni et al., 2015; Hamilton et al., 2016) is to compare the word's vector representations between two points in time using the cosine distance:

$$cosDist(x, y) = 1 - \frac{x \cdot y}{\|x\|_2 \|y\|_2} \quad (1)$$

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1136–1145
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

This choice naturally assumes that greater distances correspond to greater semantic changes. However, this measure introduces biases that may affect our interpretation of meaning change.

We examine various representations of word meaning, in order to identify inherent confounds when meaning change is evaluated using the cosine distance. In addition to the empirical evaluation, in Section 5 we provide an analytical account of the influence of word frequency on cosine distance scores when using these representations.

In our empirical investigation, we highlight the critical role of control conditions in the validation of experimental findings. Specifically, we argue that every observation about a change of meaning over time should be subjected to a control test. The control condition described in Section 2.1 is based on the construction of an artificially generated corpus, which resembles the historical corpus in most respects but where no change of meaning over time exists. In order to establish the validity of an observation about meaning change - and even more importantly, the validity of a law-like generalization about meaning change - the result obtained in a genuine experimental condition should be demonstrated to be lacking (or at least significantly diminished) in the control condition.

As we show in Section 4, some recently reported laws of historical meaning change do not survive this proposed test. In other words, similar results are obtained in the genuine and control conditions. These include the correlation of meaning change with word frequency, polysemy (the number of different meanings a word has), and prototypicality (how representative a word is of its category). These factors lie at the basis of the following proposed laws of semantic change:

- The Law of Conformity, according to which frequency is negatively correlated with semantic change (Hamilton et al., 2016).
- The Law of Innovation, according to which polysemy is positively correlated with semantic change (Hamilton et al., 2016).
- The Law of Prototypicality, according to which prototypicality is negatively correlated with semantic change (Dubossarsky et al., 2015).

Our analysis shows that these laws have only residual effects, suggesting that frequency and prototypicality may play a smaller role in semantic change than previously claimed. The main artefact underlying the emergence of the first two laws in both the genuine and control conditions may be due to the SVD step used for the embedding of the PPMI word representation (see Section 2.5).

2 Methods

The historical corpus used here is Google Books 5-grams of English fiction. Equally sized samples of 10 million 5-grams per year were randomly sampled for the period of 1900-1999 (Kim et al., 2014) to prevent the more prolific publication years from biasing the results, and were grouped into ten-year bins. Uncommon words were removed, keeping the 100,000 most frequent words as the vocabulary for subsequent model learning. All words were lowercased and stripped of punctuation.

This corpus served as the genuine condition, and was used to replicate and evaluate findings from previous studies. In this corpus, words are expected to change their meaning between decadal bins, as they do in a truly random sample of texts. According to the distributional hypothesis (Firth, 1957), one can extract a word's meaning from the contexts in which it appears. Therefore, if words' meanings change over time, as has been argued at least since Reisig (1839), it follows that the words' contexts should change accordingly, and this change should be detected by our model.

2.1 Control condition setup

Complementary to the genuine condition, a control condition was created where no change of meaning is expected. Therefore, any observed change in a word's meaning in the control condition can only stem from random “noise“, while changes in meaning in the genuine condition are attributed to “real“ semantic change in addition to “noise“. Two methods were used to construct the corpus in the control condition:

Chronologically shuffled corpus (shuffle): 5-grams were randomly shuffled between decadal bins, so that each bin contained 5-grams from all the decades evenly. This was chosen as a control condition for two reasons. First, this condition resembles the genuine condition in size of the vocabulary, size of the corpus, overall variance in words' usage, and size of the decadal bins. Second and crucially, words are not expected to show any apparent change in their meaning between decades in the control condition, because their various usage contexts are shuffled across decades.

One synchronous corpus (subsample): All 5-grams of the year 1999, which amount to 250 million 5-grams, were selected from Google Books English fiction. 10 million 5-grams were randomly subsampled from this selection, and this process was repeated 30 times. This is suggested as an additional control condition since the underlying assumption is always that words in the same year do not change their meaning. Again, unlike in the genuine condition, any changes that are observed based on these 30 subsamples can be attributed *only* to "noise" that stems from random sampling, rather than real change in meaning.

2.2 Measures of interest

Meaning change: Meaning change was evaluated as the cosine distance between vector representations of the same word in consecutive decades. This was done separately for each processing stage (see Section 2.5). For the subsample condition, this was defined as the average cosine distance between the vectors in all 30 samples.

Frequency: Words' frequencies were computed separately for each decadal bin as the number of times a word appeared divided by the total number of words in that decade. For the subsample control condition, it was computed as the number of times a word appeared among the 250 million 5-grams, divided by the total number of words.

2.3 Construct validity

To establish the adequacy of our control condition, we compared the meaning change scores (before log-transformation and standardization) between the genuine and the shuffled control conditions. Change scores were obtained by taking the average meaning change over all words in each decade using the representation of the final processing stage (SVD). An adequate control condition will exhibit a lower degree of change compared to the genuine condition, and is expected to show a fixed rate of change across decades (see 3a).

2.4 Statistical analysis

Following common practice (Hamilton et al., 2016), the 10k most frequent words, as measured by their average decadal bin frequencies, were used for the analysis of semantic change. Change scores and frequencies were log-transformed, and all variables were subsequently standardized.

A linear mixed effects model was used to evaluate meaning change in both the genuine and shuffled control conditions. Frequency was set as a fixed effect while random intercepts were set per word. The model attempts to account for semantic change scores using frequency, while controlling for the variability between words by assuming that each word's behavior is strongly correlated across decades and independent across words as follows:

$$\Delta w_i^{(t)} = \beta_0 + \beta_f freq_{w_i}^{(t)} + z_{w_i} + \varepsilon_{w_i}^{(t)} \quad (2)$$

Here $\Delta w_i^{(t)}$ is the semantic change score of the i'th word measured between two specific consecutive decades, β_0 is the model's intercept, β_f is the fixed-effect predictor coefficient for frequency, $z_{w_i} \sim N(0, \sigma)$ is a random intercept for the i'th word, and $\varepsilon_{w_i}^{(t)}$ is an error term associated with the i'th word. We report the predictor coefficient as well as the proportion of variance explained[1] by each model. Only statistically significant results ($p < .01$) are reported. All statistical tests are performed in R (lme4 and MuMln packages).

2.5 Word meaning representation

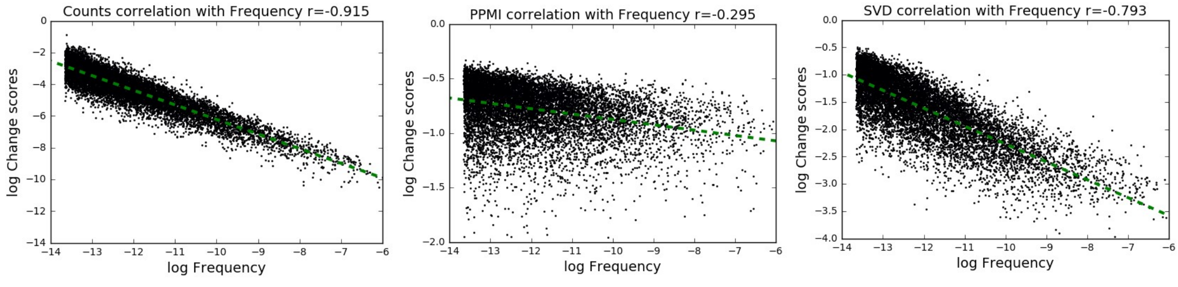
We used a cascade of processing stages based on the *explicit meaning* representation of words (i.e., word counts, PPMI, SVD, as explained below) as commonly practiced (Baroni et al., 2014; Levy et al., 2015). For each of these stages, we sought to evaluate the relationship between word frequency and meaning change, by computing the corresponding correlations between these two factors in the subsample control condition.

Counts: Co-occurrence counts were collected for all the words in the vocabulary per decade.

PPMI: Sparse square matrices of vocabulary size containing positive pointwise mutual information (PPMI) scores were constructed for each decade based on the co-occurrence counts. We used the context distribution smoothing parameter $\alpha = 0.75$, as recommended by (Levy et al., 2015), using the following procedure:

$$PPMI_\alpha(w, c) = \max\left(log\left(\frac{\hat{P}(w, c)}{\hat{P}(w)\hat{P}_\alpha(c)}\right), 0\right)$$

[1] R^2 for mixed linear models (Nakagawa and Schielzeth, 2013)



(a) (b) (c)

Figure 1: Correlations in the control condition between change scores in the year 1999 and word frequency for three word representation types, based on: (a) Counts, (b) PPMI, (c) SVD. Correlation coefficients are reported above each subplot. LS regression lines are shown in dashed green.

where $\hat{P}(w,c)$ denotes the probability that word c appears as a context word of w, while $\hat{P}(w)$ and $\hat{P}_\alpha(c) = \frac{\#(c)^\alpha}{\sum_C \#(c)^\alpha}$ denote the marginal probabilities of the word and its context, respectively.

SVD: Each PPMI matrix was approximated by a truncated singular value decomposition as described in (Levy et al., 2015). This embedding was shown to improve results on downstream tasks (Baroni et al., 2014; Bullinaria and Levy, 2012; Turney and Pantel, 2010). Specifically, the top 300 elements of the diagonal matrix of singular values Σ, denoted Σ_d, were retained to represent a new, dense embedding of the word vectors, using the truncated left hand orthonormal matrix U_d:

$$W_i^{SVD} = (U_d \cdot \Sigma_d)_i \tag{3}$$

These representations were subsequently aligned with the orthogonal Procrustes method following (Hamilton et al., 2016).

Relation to other models: (Levy and Goldberg) have shown that the Skip-Gram with Negative Sampling (SGNS) embedding model, e.g. word2vec (Mikolov et al., 2013) - perhaps the most popular model of word meaning representation, implicitly factorizes the values of the word-context PMI matrix. Hence, the optimization goal and the sources of information available to SGNS and our model are in fact very similar. We therefore hypothesize that conclusions similar to those reported below can be drawn for SGNS models.

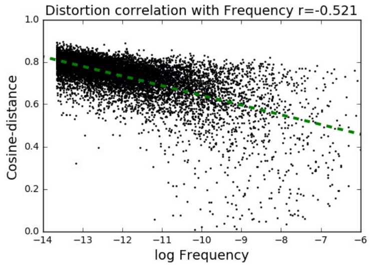


Figure 2: Cosine distances between PPMI and approximated PPMI representations (y-axis), plotted against frequency (x-axis). Correlation coefficient is reported above the plot.

3 Results

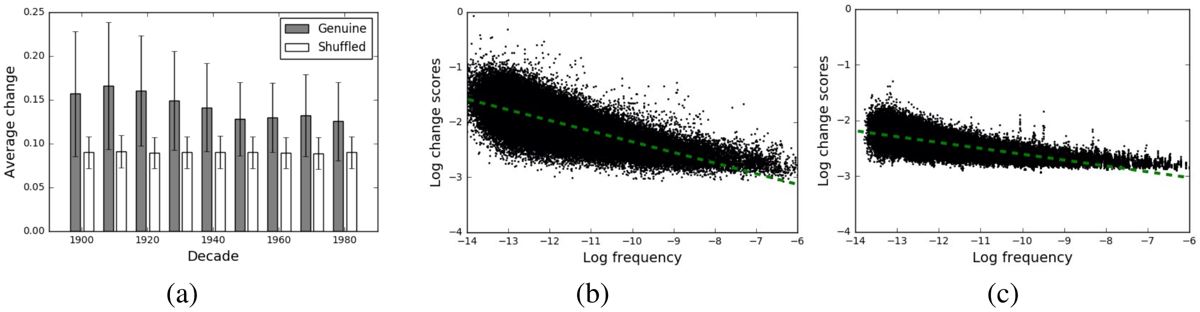
3.1 Confound of frequency

There are many factors that may confound the measurement of meaning change. Here we focus on frequency, and investigate the existence of an artefactual relation between frequency and meaning change. This is done by evaluating this relation in the subsample control condition. Any changes observed in this condition must be the consequence of inherent noise, since this control condition contains random samples from the same year (and the baseline assumption is that no change can be observed within the same year).

We first plotted the change scores that use the representation based on word count vs. word frequency. This resulted in a robust correlation ($r = -0.915$) between the two variables, as shown in Fig. 1a (see the analytical account in Section 5). We repeated the same procedure using the PPMI representation, which showed a much weaker correlation with frequency ($r = -0.295$), see Fig. 1b.

Finally, we repeated the same procedure using the final *explicit representation* after SVD embedding[2], see Fig. 1c. Surprisingly, the negative correlation with frequency was reinstated ($r = -0.793$). To investigate how this came about,

[2] Similar results were obtained for the implicit embedding (word2vec-SGNS) described in Section 2.5.



(a) (b) (c)

Figure 3: (a) Average change score per decade for the genuine and control conditions. Bars represent standard deviations. (b-c) Change scores (y-axis), relative to their frequency (x-axis): (b) genuine historical corpus, (c) chronologically shuffled historical corpus. LS regression lines are shown in dashed green.

we computed the change in the PPMI vectors before and after the low-rank SVD embedding using the cosine-distance. As apparent from Fig. 2, it turns out that the SVD procedure distorts data in an uneven manner - frequent words are distorted less than infrequent words. Thus we demonstrate that this reinstatement of correlation between frequency and change scores is merely an artefactual consequence of the truncated SVD factorization.

3.2 Construct validity

Potential confounding factors can be addressed by comparing any experimental finding to a validated control condition. Here we validate the use of the shuffled condition as a proper control. To this end, the average change scores of words per decade in both the genuine and shuffled conditions are compared within each processing stage. In the genuine condition, words appear in different usage contexts between decades, while in the shuffled condition they do not, because the random shuffling creates a homogeneous corpus. Therefore, the validity of the control condition is established if: (a) the change scores are diminished as compared to the genuine condition; (b) change scores are uniform across decades (since decades are shuffled); (c) the variance of change scores is smaller that in the genuine condition. As seen in Fig. 3a, all these requirements are met by the control condition. Note that the change scores in the shuffled condition are all significantly positive, namely, meaning change allegedly exists in this control condition. This supports the claim that any measurement is significantly affected by unrelated noise.

Thus, we have established that the shuffled condition is a suitable control for meaning change. While validity was established for each of the processing stages, the most robust effect was seen for the PPMI representation, following by SVD and word counts.

3.3 Accounting for the frequency confound

In Section 3.1 we used the subsample control condition to establish the confounding effect of frequency on meaning change. We now examine the extent to which this frequency confound exists in a historical corpus. We do so by comparing the frequency confound between the genuine historical corpus and the shuffled historical corpus.

To visualize the frequency confound in a manner comparable to the analysis presented in Section 3.1, we again plot change scores vs. frequency, ignoring the time dimension of the data. Fig. 3b presents this plot for the genuine condition. The same analysis is repeated in the shuffled condition, see Fig. 3c.

Both plots reveal a highly significant correlation between change scores and frequency. Furthermore, the fact that the correlation coefficients are virtually identical in the genuine and shuffled conditions, with $r = -0.748$ and $r = -0.747$ respectively, suggests that they are due to artefactual factors in both conditions and not to true change of meaning over time. In fact, this pattern of results is reminiscent of the spurious pattern we see in Fig. 1c.

The relation between frequency and meaning change can also be represented by a linear mixed effect model, with the benefit that this model enables the addition of more explanatory variables to the data. The regression model found frequency to have a negative influence on change scores,

		PPMI + SVD		PPMI	
		Genuine	Shuffled	Genuine	Shuffled
Frequency (one-predictor)	β	-0.91	-0.75	-0.29	0.06
	explained variance (σ^2)	67%	56%	8%	0%
Frequency + Polysemy (two-predictor)	β frequency	-1.22	-1.12	-0.69	0.53
	β polysemy	0.43	0.40	0.49	-0.52
	explained variance (σ^2)	68%	60%	9%	4%
Frequency + Prototypicality (two-predictor)	β frequency	-0.71	-0.70	-0.02	0.07
	β polysemy	0.22	0.21	0.12	0.02
	explained variance (σ^2)	65%	60%	2%	0%

Table 1: Results of one-predictor and two-predictor regression analysis in all conditions.

with β_f=-0.91 and β_f=-0.75, for the genuine and shuffled conditions respectively. Importantly, frequency accounted for 67% of the variance in the change scores in the genuine condition, and was only slightly diminished in the shuffled condition, accounting for 56% of the variance. Similar results were obtained for the PPMI representation (see Table 1).

4 Revisiting previous studies

We replicated three recent results that were affected by this frequency effect, since they all define change as the word's cosine distance relative to itself at two time points. These studies report laws of semantic change that measure the role of frequency in semantic change either directly (Law of Conformity), or indirectly through another linguistic variable that is dependent on frequency (Laws of Innovation and Prototypicality).

4.1 Laws of conformity and innovation

Continuing the work described in Section 3.1, we replicated the model and analysis procedure described in (Hamilton et al., 2016), where **two predictors** were used together to explain the change scores: frequency and polysemy. Polysemy, which describes the number of different senses a word has, naturally differs among words, where some words are more polysemous than others (compare *bank* and *date* to *wine*). Following (Hamilton et al., 2016), we defined polysemy as the words' secondary connections patterns - the connections between each word's co-occurring words (using the entries in the PPMI representation for that word). The more interconnected these secondary connections are, the less polysemic a word is, and vice versa. Polysemy scores were computed using the authors' provided code[3]. We then log-transformed and standardized the polysemy scores. Next, frequency and polysemy were set as two fixed effect predictors in a linear mixed effect model, like the one described in Section 2.4.

Thus we were able to replicate the results in the genuine condition as reported in (Hamilton et al., 2016). Interestingly, the same pattern of results emerged, again, in the shuffled condition (see Table 1). Importantly, the difference in effect size between conditions, as evaluated by the explained variance of frequency and polysemy together, showed a modest effect of 8% over the shuffled condition, pointing to the conclusion that the putative effects may indeed be real, but to a far lesser extent than had been claimed. We conclude that adding polysemy to the analysis contributed very little to the model's predictive power.

Since the PPMI representation (the *explicit representation* without dimensionality reduction with SVD) seems much less affected by spurious effects correlated with frequency (see Fig. 1b), we repeated the analysis of frequency described here and in Section 3.1 while using this representation. The results are listed in Table 1, showing a similar pattern of rather small frequency effect.

4.2 Prototypicality

Prototypicality is the degree to which a word is representative of the category of which it is a member (a *robin* is a more prototypical bird than a *parrot*). According to the proposed Law of Prototypicality, words with more prototypical meanings will show less semantic change, and vice versa. Following (Dubossarsky et al., 2015), we computed words' prototypicality scores for each decade as the cos-distance between a word's vec-

[3] https://github.com/williamleif/histwords

tor and its k-means cluster's centroid, and extended the analysis to encompass the entire 20th century. The previous regression model assumed independence between words, and therefore assigned words to a random effect variable. However, when modeling prototypicality, this assumption is invalid as relations between words are what inherently define prototypicality. We therefore designed a model in which decades, rather than words, are the random effect variable.

With this analysis the prototypicality effect seems to be substantiated in two ways. First, the addition of prototypicality explains an additional 5% of the variance. Second, the effect of prototypicality meets the more stringent requirement of being diminished in the shuffle condition (see Table 1). Nevertheless, here too the effect originally reported was found to be drastically reduced after being compared with the proper control.

5 Theoretical analysis

We show in Section 5.1 that the average cosine distance between two vectors representing the same word is equivalent to the variance of the population of vectors representing the same word in independent samples, and is therefore always positive. This is true for any word vector representation.

In Sections 5.2-5.3 we prove that the average cosines distance between two *count* vectors representing the same word is negatively correlated with the frequency of the word, and positively correlated with the polysemy score of the word.

5.1 Sampling variability and the cos distance

Lemma 1. *Assume two random variables x, y of length $\|x\|_2 = \|y\|_2 = 1$, distributed iid with expected value μ and covariance matrix Σ. The expected value of the cosine distance between them is equal to the sum of the diagonal elements of Σ.*

Proof.

$$\begin{aligned}E(x-y)^2 =& E(x-\mu)^2 + E(y-\mu)^2 + \\ & 2E(x-\mu)(y-\mu) \\ =& 2\sum E(x_i-\mu_i)^2 = 2\sum Var(x_i) \\ E(x-y)^2 =& E(x^2) + E(y^2) - 2E(x\cdot y) \\ =& 2 - 2E\left(\frac{x\cdot y}{\|x\|_2\|y\|_2}\right) \\ =& 2E(cosDist(x,y))\end{aligned}$$

It follows that

$$E(cosDist(x,y)) = \sum Var(x_i) \quad (4)$$

□

Implication: The average cosine distance between two samples of the same random variables is directly related to the variance of the variable, or the sampling noise. This variance should be measured empirically whenever cosine distance is used, since only distances that are larger than the empirical variance can be relied upon to support significant observations.

5.2 Cos distance of count vectors: frequency

Next, we analyze the cosine distance between 2 iid samples from a normalized multinomial random variable. This distribution models the distribution of the count vector representation. Let $k_i,\ 1 \le i \le m$ denote the number of times word i appeared in the context of word w, and let m denote the size of the dictionary not including w. Let $n = \sum k_i$ denote the number of words in the count vector of w; n determines the word's frequency score. Assume that the counts are sampled from the distribution Multinomial$(n, \vec{p})$, namely

$$Prob(k_1, \cdots, k_m) = \binom{n}{k_1 \cdots, k_m} p_1^{k_1} \cdots p_m^{k_m}$$

Lemma 2. *The expected value of the cosine distance between two count vectors x, y sampled iid from this distribution is monotonically decreasing with n.*

Proof. By definition, $1 - E[cosDist(x,y)]$ equals

$$E\left[\frac{x\cdot y}{\|x\|_2\|y\|_2}\right] = \sum_i \left[E\frac{x_i}{\|x\|_2}\right]^2 = \sum_i E_i^2 \quad (5)$$

We compute the expected value of E_i directly:

$$E_i = \sum_{(k_1,\cdots,k_m)} \frac{k_i}{\sqrt{\sum_j k_j^2}} \binom{n}{k_1 \cdots, k_m} p_1^{k_1} \cdots p_m^{k_m}$$

Using Taylor expansion:

$$\begin{aligned}\frac{k_i}{\sqrt{\sum_j k_j^2}} =& \frac{\frac{k_i}{n}}{\sqrt{(\sum_j \frac{k_j}{n})^2 - \sum_{l\neq j}\frac{k_j k_l}{n^2}}} \\ =& \frac{k_i}{n}\frac{1}{\sqrt{1 - \sum_{l\neq j}\frac{k_j k_l}{n^2}}} \\ =& \frac{k_i}{n}\left(1 + \frac{\varepsilon}{2} + O(\varepsilon^2)\right) \quad (6)\end{aligned}$$

where $\varepsilon = \sum_{l \neq j} \frac{k_j k_l}{n^2}$.

The expected value of the 0-order term with respect to ε in (6) equals p_i, which is independent of n. We conclude the proof by focusing on the first order term with respect to ε in (6), to be denoted f_1, showing that its expected value is monotonically decreasing with n. Specifically:

$$f_1 = \sum_{\vec{k}} \sum_{l \neq j} \frac{k_i}{n} \frac{k_j}{n} \frac{k_l}{n} \binom{n}{k_1 \cdots, k_m} p_1^{k_1} \cdots p_m^{k_m}$$

We switch the summation order and compute each expression in the external sum, considering two cases separately: when $l \neq j \neq i$

$$\sum_{(k_1, \cdots, k_m)} \frac{k_i}{n} \frac{k_j}{n} \frac{k_l}{n} \binom{n}{k_1 \cdots, k_m} p_1^{k_1} \cdots p_m^{k_m}$$
$$= \frac{n(n-1)(n-2)}{n^3} p_i p_j p_l$$

When $l \neq j = i$ w.l.g, we rewrite $k_i k_j = k_i(k_i - 1) + k_i$, and the sum above becomes $\frac{n(n-1)(n-2)}{n^3} p_i^2 p_l + \frac{n(n-1)}{n^2} p_i p_l$. Thus

$$f_1 = \frac{n-1}{n} p_i \left[\frac{n-2}{n} \sum_{l,j:l \neq j} p_j p_l + (1 - p_i) \right]$$

and it readily follows that f_1 is monotonically increasing with n.

Since n measures the frequency score of word w, it follows from (5) that the expected value of the cosine distance between two iid samples from the distribution of the count vector of w is monotonically decreasing with the word's frequency. □

5.3 Cos distance of count vectors: polysemy

We start our investigation of polysemy by modeling the distribution of the parameters of the multinomial distribution from which count vectors are sampled. A common prior distribution on the vector $\vec{p}^w$ in m-simplex, which defines the multinomial distribution generating the context of word w, is the Dirichlet distribution $f(\vec{p}^w; \vec{\alpha}^w) = f(p_1, \cdots, p_m; \alpha_1, \cdots, \alpha_m)$.

$\vec{\alpha}^w$ is a sparse vector of prior counts on all the words in the dictionary, by which the cooccurrence context of word w is modeled. We divide the set of none-zero indices of $\vec{\alpha}^w$ into two subsets: $i_1, \cdots, i_{m_0}$ correspond to the words which always appear in the context of w, while $j_1, \cdots, i_{m_1}$ correspond to the words which appear in the context of w in one given meaning. If w is polysemous and has two meanings, then there is a third set of indices $k_1, \cdots, k_{m_2}$ which correspond to the words appearing in the context of w in its second meaning. If w has more then two meanings, they can be modeled with additional sets of disjoint indices.

Lemma 3. *Under certain conditions specified in the proof, given two count vectors x, y sampled iid from the above distribution of w, the expected value of the cosine distance between them increases with the number of sets of disjoint indices which represent different meanings of w.*

Proof. We will prove that when w has two meanings, the expected value of the cosine distance is larger than in the case of a single meaning. The proof for the general case immediately follows.

Starting from (6) while keeping only the 0-order term in ε, it follows from the derivations in the proof of Lemma 2 that the expected cosine distance between two count vector samples of w, to be denoted M, is $1 - \sum p_i^2$. In our current model $\vec{p}$ is a random variable, and we shall compute the expected value of this random variable under the two conditions, when w has either one or two meanings.

We start by observing that, given the definition of the Dirichlet distribution, it follows that

$$E(p_i^2) = Var(p_i) + E(p_i)^2 = \frac{\alpha_i(1+\alpha_i)}{\alpha_0(1+\alpha_0)}$$
$$\alpha_o = \sum \alpha_i$$
$$\Longrightarrow M = \sum E(p_i^2) = \frac{\alpha_0 + \sum \alpha_i^2}{\alpha_0(1+\alpha_0)} \qquad (7)$$

Considering the different sets of indices in isolation, let $\varphi_o = \sum_{i=i_1}^{i_{m_0}} \alpha_i$, $\varphi_1 = \sum_{i=j_1}^{j_{m_1}} \alpha_i$, and $\varphi_2 = \sum_{i=k_1}^{k_{m_2}} \alpha_i$. Let $\psi_o = \sum_{i=i_1}^{i_{m_0}} \alpha_i^2$, $\psi_1 = \sum_{i=j_1}^{j_{m_1}} \alpha_i^2$, and $\psi_2 = \sum_{i=k_1}^{k_{m_2}} \alpha_i^2$.

We rewrite (7) for the two conditions:

1. w has one meaning:

$$M^{(1)} = \frac{\varphi_0 + \varphi_1 + \psi_0 + \psi_1}{(\varphi_0 + \varphi_1)(1 + \varphi_0 + \varphi_1)}$$

2. w has two meanings:

$$M^{(2)} = \frac{\varphi_0 + \varphi_1 + \varphi_2 + \psi_0 + \psi_1 + \psi_2}{(\varphi_0 + \varphi_1 + \varphi_2)(1 + \varphi_0 + \varphi_1 + \varphi_2)}$$

With some algebraic manipulations, it can be shown that $M^{(1)} > M^{(2)}$ if the following holds:

$$\begin{aligned}&(\varphi_0+\varphi_1)^2\varphi_2+(\psi_0+\psi_1)\varphi_2^2 \\ &+2(\psi_0+\psi_1)(\varphi_0+\varphi_1)\varphi_2+(\psi_0+\psi_1)\varphi_2 \\ &+(\varphi_0+\varphi_1)(\varphi_2^2-\psi_2) > \psi_2(\varphi_0+\varphi_1)^2\end{aligned} \tag{8}$$

Thus when (8) holds, the average cosine distance between two samples of a certain word w gets larger as w acquires more meanings. □

(8) readily holds under reasonable conditions, e.g., when the prior counts for each meaning are similar (as a set) and much bigger than the prior counts of the joint context words (i.e., $\varphi_0 = \psi_0 = \varepsilon$, $\varphi_1 = \varphi_2$, $\psi_1 = \psi_2$).

6 Conclusions and discussion

In this article we have shown that some reported laws of semantic change are largely spurious results of the word representation models on which they are based. While identifying such laws is probably within the reach of NLP analyses of massive digital corpora, we argued that a more stringent standard of proof is necessary in order to put them on a firm footing. Specifically, it is necessary to demonstrate that any proposed law of change has to be observable in the genuine condition, but to be diminished or absent in a control condition. We replicated previous studies claiming to establish such laws, which propose that semantic change is negatively correlated with frequency and prototypicality, and positively correlated with polysemy. None of these laws - at least in their strong versions - survived the more stringent standard of proof, since the observed correlations were found in the control conditions.

In our analysis, the Law of Conformity, which claims a negative correlation between word frequency and meaning change, was shown to have a much smaller effect size than previously claimed. This indicates that word frequency probably does play a role - but a small one - in semantic change. According to the Law of Innovation, polysemy was claimed to correlate positively with meaning change. However, our analysis showed that polysemy is highly collinear with frequency, and as such, did not demonstrate independent contribution to semantic change. For similar reasons, the alleged role of prototypicality was diminished.

These results may be more consonant than previous ones with the findings of historical linguistics, as it is commonly assumed that the factors leading to semantic change are more diverse than purely distributional factors. For example, sociocultural, political, and technological changes are known to impact semantic change (Bochkarev et al., 2014; Newman, 2015). Furthermore, some regularities of semantic change have been imputed to ‘channel bias‘, inherent biases of utterance production and interpretation on the part of speakers and listeners, e.g., (Moreton, 2008). As such, it would be surprising if word frequency, polysemy, and prototypicality were to capture *too high* a degree of variance. In other words, since semantic change may result from the interaction of many factors, small effects may be a priori more credible than large ones.

The results of our empirical analysis showed that the spurious effects of frequency were much weaker for the explicit PPMI representation unaugmented by SVD dimensionality reduction. We therefore conclude that the artefactual frequency effects reported are inherent to the type of word representations upon which these analyses are based. As the analytical proof in Section 5 demonstrates, it is count vectors that introduce an artefactual dependence on word frequency.

Intuitively, one might expect that the average value for the cosine distance between a given word's vector in any two samples would be 0. However, Lemma 1 above shows that this is not the case, and the average distance is the variance of the population of vectors representing the same word. This result is independent of the specific method used to represent words as vectors. Lemma 2 proves that the average cosine distance between two samples of the same word, when using count vector representations, is negatively correlated with the word's frequency. Thus, the role of frequency cannot be evaluated as an independent predictor in any model based on count vector representations. It remains for future research to establish whether other approaches to word representation, e.g. (Blei et al., 2003; Mikolov et al., 2013), have inherent biases.

While our findings may seem to be mainly negative, since they invalidate proposed laws of semantic change, we would like to point to the positive contribution made by articulating more stringent standards of proof and devising replicable control conditions for future research on language change based on distributional semantics representations.

References

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of ACL*, pages 238–247.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022.

Vladimir Bochkarev, Valery Solovyev, and Sören Wichmann. 2014. Universals versus historical contingencies in lexical evolution. *Journal of The Royal Society Interface*, 11:1–23.

John A Bullinaria and Joseph P Levy. 2012. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and svd. *Behavior research methods*, 44(3):890–907.

Haim Dubossarsky, Yulia Tsvetkov, Chris Dyer, and Eitan Grossman. 2015. A bottom up approach to category mapping and meaning change. In *NetWordS 2015 Word Knowledge and Word Usage*, pages 66–70.

Haim Dubossarsky, Daphna Weinshall, and Eitan Grossman. 2016. Verbs change more than nouns: A bottom up computational approach to semantic change. *Lingue e Linguaggio*, 1:5–25.

John Rupert Firth. 1957. *Papers in Linguistics 1934–1951*. Oxford University Press, London.

Lea Frermann and Mirella Lapata. 2016. A Bayesian model of diachronic meaning change. *TACL*, 4:31–45.

Kristina Gulordava and Marco Baroni. 2011. A distributional similarity approach to the detection of semantic change in the Google Books Ngram corpus. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 67–71. Association for Computational Linguistics.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of ACL*.

Martin Hilpert and Florent Perek. 2015. Meaning change in a petri dish: constructions, semantic vector spaces, and motion charts. *Linguistics Vanguard*, 1(1):339–350.

Adam Jatowt and Kevin Duh. 2014. A framework for analyzing semantic change of words across time. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 229–238.

Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. In *Proceedings of ACL*, pages 61–65.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, pages 625–635.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2177–2185.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee, and Pawan Goyal. 2014. That's sick dude!: Automatic identification of word sense change across different timescales. In *Proceedings of ACL*, pages 1020–1029.

Elliott Moreton. 2008. Analytic bias and phonological typology. *Phonology*, 25(1):83–127.

Shinichi Nakagawa and Holger Schielzeth. 2013. A general and simple method for obtaining r2 from generalized linear mixed-effects models. *Methods in Ecology and Evolution*, 4(2):133–142.

John Newman. 2015. Semantic shift. In Nick Rimer, editor, *The Routledge Handbook of Semantics*, pages 266–280. Routledge, New York.

Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2009. Semantic density analysis: Comparing word meaning across time and phonetic space. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 104–111. Association for Computational Linguistics.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188.

Derry Tanti Wijaya and Reyyan Yeniterzi. 2011. Understanding semantic change of words over centuries. In *Proceedings of the 2011 international workshop on DETecting and Exploiting Cultural diversiTy on the social web*, pages 35–40. ACM.

Yang Xu and Charles Kemp. 2015. A computational evaluation of two laws of semantic change. In *CogSci*.

Human Centered NLP with User-Factor Adaptation

Veronica E. Lynn, Youngseo Son, Vivek Kulkarni
Niranjan Balasubramanian and **H. Andrew Schwartz**
Stony Brook University
Stony Brook, NY
{velynn, yson, vvkulkarni, niranjan, has}@cs.stonybrook.edu

Abstract

We pose the general task of *user-factor adaptation* — adapting supervised learning models to real-valued user factors inferred from a background of their language, reflecting the idea that a piece of text should be understood within the context of the user that wrote it. We introduce a *continuous* adaptation technique, suited for real-valued user factors that are common in social science and bringing us closer to personalized NLP, adapting to each user uniquely. We apply this technique with known user factors including age, gender, and personality traits, as well as latent factors, evaluating over five tasks: POS tagging, PP-attachment, sentiment analysis, sarcasm detection, and stance detection. Adaptation provides statistically significant benefits for 3 of the 5 tasks: up to +1.2 points for PP-attachment, +3.4 points for sarcasm, and +3.0 points for stance.

1 Introduction

Language use is personal. Knowing who wrote a piece of text can help to better understand it. For instance, knowing the age and gender groups of authors has been shown to improve document classification (Hovy, 2015) and sentiment analysis (Volkova et al., 2013).

However, putting people into discrete groups (e.g. age groups, binary gender) often relies on arbitrary boundaries which may not correspond to meaningful changes in language use. A wealth of psychological research suggests people should not be characterized as discrete *types* (or domains) but rather as mixtures of continuous *factors* (McCrae and Costa Jr., 1989; Ruscio and Ruscio, 2000; Widiger and Samuel, 2005).

Here, we ask how one can adapt NLP models to real-valued human *factors* – continuous valued attributes that capture fine-grained differences between users (e.g. real-valued age, continuous gender scores). We refer to this problem as *user-factor adaptation*, and investigate a solution to it in the context of social media, a genre where language is generated by a particularly diverse set of users (Duggan and Smith, 2013). Importantly, *user-factor adaptation* brings us closer to personalized NLP in that with real-valued factors we can now adapt uniquely for each user.

Our approach composes user factor information with the linguistic features, similar to feature augmentation (Daumé III, 2007), a widely used domain adaptation technique which allows for easy integration with most feature-based learning models. Since relevant user information often is not explicitly available, we use a background of tweets from the user to infer user factors. We evaluate our approach over five tasks — POS tagging, PP-attachment, sentiment analysis, sarcasm detection, and stance detection — and with a variety of inferred user factors including (a) *known factors*: age, gender, and personality traits, as well as (b) *latent factors* derived from past user tweets.

Contributions. The main contributions of this work include (a) adaptation based simply on background language (e.g. past tweets; no required *a priori* user knowledge or "domain"), (b) a method for adapting models based on continuous variables, (c) adaptation to other user attributes beyond age and gender (personality and latent factors), and (d) empirical evidence that standard NLP models can often be improved by user-factor adaptation with a range of inferred factors.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1146–1155
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

2 User-Factor Adaptation

User-factor adaptation is especially critical for social media, where content is generated by a diverse user base (Duggan and Smith, 2013). Adaptation requires two components: 1) a user factor representation that captures salient traits indicative of language differences between users, and 2) an adaptation technique that uses this representation to modify learning appropriately.

User factors, even simple ones such as age and gender, may not always be readily available. The messages posted by users, however, are often public and can be used to infer many known linguistically relevant user traits including personality, as well as latent language factors (described next). Given this *background* information about users, the *user-factor adaptation problem* is to learn a single model that is sensitive to both the variations and commonalities in language across different users.

3 User Factors

The first step in our adaptation approach is to create a representation of users that relates to their language use. To this end, we explore two sets of factors: 1) inferred demographics and personality traits, and 2) latent language factors that directly capture language use variations among users.

Different from prior work, we model these human attributes as *real-valued factors*, as is common in psychology literature. Although they may refer to discrete classes such as cluster membership, a factor representation is able to capture more nuanced differences and characteristics that are best understood as a continuum (McCrae and Costa Jr., 1989; Ruscio and Ruscio, 2000; Widiger and Samuel, 2005). This is critical for our goal of moving beyond group-level adaptation toward personalization.

3.1 Demographic and Personality Factors

Many studies have linked language variations with demographic (Argamon et al., 2007; Cheshire, 2005), occupational (Preoţiuc-Pietro et al., 2016) and other psychosocial variables such as personality (Schwartz et al., 2013). We investigate the relevance of a subset of these social variables as user factors for adaptation.

However, we may not have direct access to such information. Unlike the tweets posted by a user, their demographic and personality traits are not always publicly available. We use automatic classification models for obtaining real-valued age and gender estimates (Sap et al., 2014) and personality traits (Park et al., 2015). In addition to being reasonably accurate (e.g. age prediction has a Pearson r of .83 with true age), language based estimation of factor scores may capture linguistic preferences more clearly. For instance, Bamman et al. (2014b) found that *perceived* gender was strongly linked to the gender makeup of a user's social network, and may be a better descriptor of linguistic preferences than self-reported gender.

3.2 Latent Language Factors

We also explore methods to derive latent factors that capture language use similarities and variations across users. The main idea is to derive a latent d-dimensional representation of each user using their background tweets. While there are many choices here, we explore a factorization technique (generative factor analysis), a clustering technique (k-means with TF-IDF), and a hybrid (word2vec with k-means).

Generative Factor Analysis. Factorization methods allow us to build latent representations of users by finding low-rank approximations of the original high-dimensional representations of their text. We use a general method called factor analysis (FA) (Lawley and Maxwell, 1971). Intuitively, FA seeks to capture the variability across correlated variables as a weighted linear combination of a given number of latent dimensions, thus allowing a low-dimensional representation of words[1].

Formally, let $\mathbf{M}_{|\mathcal{U}|\times|\mathcal{V}|}$ denote the user-term matrix, whose entries $\mathbf{M}_{ij}$ indicate the number of times word j is used by user i. FA factorizes this high-dimensional representation into two matrices $\mathbf{F}$ and $\mathbf{L}$ as follows: $\mathbf{M} = \mathbf{FL} + \mathbf{E}$ where $\mathbf{E}$ is an error matrix consisting of residual errors not captured by $\mathbf{FL}$ and where the residual noise is assumed to be Gaussian distributed with zero mean.

Clustering. We also explore commonly used text clustering-based methods to derive *latent factors* from the users' tweets. The idea is to cluster the users based on their tweets. In one case we use TF-IDF based representations, and in the other we use word2vec embeddings (Mikolov et al., 2013).

[1] In this sense FA is a more flexible method than singular value decomposition in that it allows factors to be correlated.

We produce a k-means clustering on this reduced dimensional space to create clusters of users who have similar language use. We derive real-valued factors from these clusters using the distance of the user to the centers of each cluster. Cluster membership yields the discrete representation. Refer to section 5.1 for implementation details.

4 Adaptation Models

Given a factor representation of each user, the adaptation task is to learn a model that is sensitive to both the differences and commonalities across all users. This is similar to the objective for domain adaptation tasks, where the task data is drawn from one or more underlying domains and learning needs to account for both the similarities and differences in the domains. We formulate user-factor adaptation as a domain adaptation technique based on feature augmentation (Daumé III, 2007) but rather than force users into discrete domains, we develop a continuous formulation that allows us to make good use of the real-valued user factors.

Here we first describe a direct discrete formulation of feature augmentation and then describe our proposed continuous formulation.

4.1 Discrete Adaptation

Feature augmentation uses domain information to transform instances into a new augmented space such that instances from the same domain have higher similarity in the augmented space compared to instances from different domains. A learner operating over this augmented space can now learn to model both domain-specific and domain-general influences of the features.

The discrete adaptation method is a direct application of this idea, where the training and test instances are mapped into domains based on some grouping that we induce from the user factors. For example, the user factor *age* induces three discrete *domains*: `low` (*age* < 24), `middle` ($24 <$ *age* $<$ 28), and `high` (*age* > 28).

User	Factor Classes	Augmented Instance $\Phi(\mathbf{x}, u)$
User 1	F_1	$\langle \mathbf{x}, \mathbf{x}, \mathbf{0}, \mathbf{0}, \cdots, 0 \rangle$
User 2	F_2	$\langle \mathbf{x}, \mathbf{0}, \mathbf{x}, \mathbf{0}, \cdots, 0 \rangle$
User 3	F_1, F_3	$\langle \mathbf{x}, \mathbf{x}, \mathbf{0}, \mathbf{x}, \cdots, 0 \rangle$
User 4	F_k	$\langle \mathbf{x}, \mathbf{0}, \mathbf{0}, \cdots, 0, \mathrm{x} \rangle$

Table 1: Discrete Factor Adaptation: Augmentations of an original instance vector $\mathbf{x}$ under different factor class mappings. With k domains the augmented feature vector is of length $n(k+1)$.

Given the instance domain mapping, feature augmentation transforms the instances based on their domain. Suppose the original instances have n features and suppose there are d discrete factor classes ($F_1, \cdots, F_d$) i.e., d domains. Given an instance which is mapped to a factor class F_i, augmentation creates a new feature vector that has $d+1$ feature sets of length n each. The original features are copied over to the first feature set for all instances regardless of their domain. For instances from domain i, the original features are copied over to feature set $i+1$. The other feature sets are zeroes. Table 1 shows some examples of this augmentation strategy for a single instance, $\mathbf{x}$, under different factor class mappings.

These augmented instances are used for training and testing without any further modifications to the original learning formulation.

4.2 Continuous Adaptation

Discrete adaptation ignores the continuous nature of user factors. Unlike the commonly considered domains, people don't fit neatly into discrete bins. Many psychological studies have shown the ineffectiveness of treating user factors as discrete types (McCrae and Costa Jr., 1989); we expect an adaptation method which does so to be similarly ineffective. For most factors the boundaries for determining classes is unclear, and such arbitrarily-drawn boundaries may not correspond to big changes in language use.

Figure 1 illustrates the advantage of continuous adaptation for a single feature — whether the current instance contains an intensifier — using sarcasm detection as an example. The colored shapes show the feature values for instances from four users, with green squares representing "sarcastic" tweets and yellow circles representing "not sarcastic" ones. The model is unable to distinguish between sarcastic and non-sarcastic tweets in the no adaptation and discrete adaptation case. While discrete adaptation could induce some separability, in this case it fails to account for the variations between differently-aged *over 30* users. On the other hand, if we use features values that are proportional to the actual age, it can result in a better

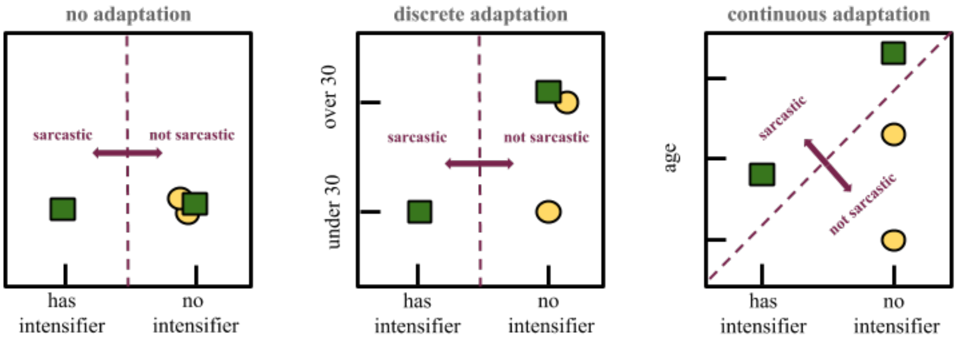


Figure 1: Comparison of feature augmentation under discrete and continuous adaptations. Each shape represents a particular observation (e.g. a tweet) to be classified, each from a different user. The x-axis represents a particular boolean feature: whether the tweet has an intensifier. The y-axis represents how the feature is augmented by the users' age using both discrete adaptation (middle) and continuous adaptation (right). Continuous adaptation allows us to distinguish observations where discrete may not.

separation as shown in the figure.

A compositional function c combines d user factor scores $f_{u,d}$ with original feature values $\mathbf{x}$:

$$\Phi(\mathbf{x}, u) = \langle \mathbf{x}, c(f_{u,1}, \mathbf{x}), c(f_{u,2}, \mathbf{x}), \cdots, c(f_{u,d}, \mathbf{x}) \rangle$$

Thus, a version of each feature exists with and without the factor information integrated. We will explore a simple multiplicative compositional function (i.e., $c(f_{u,d}, \mathbf{x}) = f_{u,d} \cdot \mathbf{x}$) but others can be imagined (e.g. additive, multiplicative with kernel functions).

Multiplicative composition has the property of reducing Φ to discrete adaptation when the factors are binary i.e., $c(f_{u,d}, \mathbf{x}) = \mathbf{x}$ when $u \in F_d$ and $c(f_{u,d}, \mathbf{x}) = 0$ otherwise. As with discrete adaptation, learning then proceeds unmodified with these augmented instances.

The *augmented training data* (train$_{aug}$) is thus associated with the features x of the tweet, the task labels y, and the user information u. Following the feature augmentation formulation, any supervised learning task of finding a parametrized function h_θ over the original labeled training data can now be specified in terms of the augmented training data along with the transformed instances:

$$\arg\min_{\theta} \sum_{(x,y,u) \in train_{aug}} loss \left(h_\theta \left(\Phi(x, u), y \right) \right)$$

For test instances we apply the same transformation function Φ before prediction.

5 Evaluation

We apply user-factor adaptation to five popular NLP tasks: part-of-speech tagging, prepositional-phrase attachment, sentiment analysis, sarcasm detection, and stance detection. These represent both syntactic and semantic tasks; include some of the key steps in an NLP application pipeline; and use different types of learning formulations including logistic regression, conditional random fields, and support vector machines.

We demonstrate the value of user-factor adaptation on strong baselines for each task. Table 2 provides the specific details for each task including the systems used and their configurations.

5.1 Implementation Details

We learn factors from a user's *background language*, or past tweets[2]. To do so, we collect up to 200 tweets per user; users with fewer than 20 tweets were excluded. Retweets were not included and all tweets were tokenized using the Happier Fun Tokenizer[3].

Demographics and Personality. We derive real-valued demographics and personality scores using the models introduced in section 3.1. For demographics, our model predicts continuous age and a gender score where higher values imply more "femaleness". For personality, these scores represent the Big Five personality traits: openness to experience, conscientiousness, extraversion, agreeableness, and neuroticism (Goldberg, 1990; McCrae and Costa Jr., 1997). Age, gender, and the five personality dimensions are each a single factor.

[2] Factor inference code is available at: https://stonybrooknlp.github.io/user-factor-adaptation/

[3] https://github.com/dlatk/happierfuntokenizing

Task	**POS Tagging**	**PP-Attachment**	**Sentiment**	**Sarcasm**	**Stance**
Output	POS tags	ranked attachments	positive, neutral, negative	sarcastic, not sarcastic	for, against, neutral
System	Owoputi et al. (2013)	variant on Belinkov et al. (2014)	Mohammad et al. (2013)	Bamman and Smith (2015)	Mohammad et al. (2016)
Features	Brown clusters, lexical features	n-grams, Treebank, WordNet	word/char n-grams, lexicon features	all tweet features	word/char n-grams
Learning Alg.	conditional random field	SVM-Rank (Joachims, 2006)	linear-SVM	logistic regression	SVM
Dataset	Owoputi et al. (2013)	Kong et al. (2014) + 986 new tweets	SemEval 2013 (Nakov et al., 2013)	Bamman and Smith (2015)	SemEval 2016 (Mohammad et al., 2016)
Eval	Train/Test, Accuracy	Cross-validation, Accuracy	Train/Test, F1	Cross-validation, F1	Train/Test, F1
Tweets	1544	1319	10339	17084	3021
Users	1541	1319	9917	10966	2349
Instances	22723	2365	10339	17084	3021

Table 2: Overview of the experimental setup for all tasks. Choices were dictated primarily by the literature on top performing systems for each task.

Latent Language Factors. We use three methods to derive latent factors: (1) **tf-idf**: The TF-IDF approach uses unigrams, bigrams, and trigrams occurring in more than 20% but fewer than 80% of documents. (2) **word2vec**: The skip-grams algorithm (Mikolov et al., 2013) was used to produce 50-dimensional word embeddings. (3) **user-embed**: d-dimensional user embeddings from generative factor analysis (Child, 1990) over relative frequencies of n-grams per user-background. The TF-IDF and word2vec representations are then clustered to produce a low-dimensional representation of the users. Each dimension is a single factor. We primarily report results for d=5 for all latent factors, although we explore alternate values in Section 5.3.

Discrete Adaptation. Each user is mapped to a single "domain" per factor. For inferred age, we select three equally-sized domains: $age < 24$, $24 < age < 28$, and $age > 28$. TF-IDF and word2vec define their domains based on cluster membership. Gender, personality, and user embeddings have two domains, above and below the mean, which is done on a per-dimension basis.

Continuous Adaptation. We apply transformations to the raw factor scores before using them for adaptation. For demographic and personality factors, we apply a min-max transformation. Because language often does not vary linearly with age (Pennebaker and Stone, 2003), we additionally use the square root of the predicted age. For the cluster based latent factors, we use the inverse of the Euclidean distance of the user-background from the cluster centroid, amplifying the power of those users who are closest to each cluster. User embeddings from factor analysis are used without any transforms since they naturally produce a Gaussian distribution.

5.2 Results

Table 3 presents the main adaptation results. We compare the performance of adaptation techniques against two baselines: no inclusion of additional factors or adaptation, and models with factors randomly drawn from a Gaussian distribution – a situation requiring learning the same number of parameters as our most augmented models. For the random factor baseline, we take the average performance across five iterations for both discrete and continuous adaptation. To establish significance of difference in error between adaptation results and the no-adaptation baseline, we use permutation testing for stance detection and McNemar's test for the others. Our findings follow. While these conclusions were drawn from our own experiments, we encourage future researchers to see what works best on their own tasks and datasets.

(i) Adaptation improves over unadapted baselines: The results show significant gains with adaptation for PP-attachment (+1.0), sentiment (+1.0), sarcasm (+3.4), and stance (+3.0). Adaptation yields better results for sarcasm and stance, semantic tasks where we'd expect user preferences to be an important factor. While prior studies have shown POS variations across demographic factors (Pennebaker and Stone, 2003; Schwartz

	pos tagging		**pp-attachment**		**sentiment**		**sarcasm**		**stance**	
eval measure	**acc.**		**acc.**		**F1**		**F1**		**F1**	
adaptation factors	**disc**	**cont**	**disc**	**cont**	**disc**	**cont**	**disc**	**cont**	**disc**	**cont**
baselines										
no adaptation	91.7	91.7	71.0	71.0	60.6	60.6	73.9	73.9	64.9	64.9
random factors	91.4	91.7	71.0	70.7	59.1	61.1	73.4	74.0	65.5	65.3
user-factor adaptation — known factors										
age	91.5	91.7	69.6	70.8	60.0	**61.4**	**74.9**†	**74.8**†	**66.3**	64.9
gender	91.6	**91.9**	69.7	70.7	**61.0**	**61.0**	**75.0**†	**75.1**†	**66.2**	**65.1**
personality	91.1	91.2	**71.3**	70.2	58.6	**61.2**	**74.3**	**75.6**†	**67.7**†	**66.3**
user-factor adaptation — latent factors										
user embed (*d*=5)	91.2	90.9	70.7	70.8	59.8	**60.7**	73.9	**77.3**†	64.6	**67.9**†
tf-idf (*d*=5)	91.4	91.5	70.5	**72.0**†	58.7	**61.6**	73.8	**74.7**†	**66.8**	64.9
word2vec (*d*=5)	91.6	90.7	70.3	**71.1**	56.30	60.5	**76.4**†	**76.9**†	**67.0**	**66.2**

Table 3: Results of user-factor adaptation across all tasks. Adaptation results are shown in comparison with baseline performance (1) without adaptation and (2) with adaptation using randomly-assigned factors. *disc* denotes discrete adaptation results, and *cont* denotes continuous adaptation results. † indicates statistically significant results at 0.05 level, in comparison to the no-adaptation baseline.

et al., 2013), we hypothesize that the ambiguity in POS reduces greatly when conditioning on local context compared to demographic preferences. This coupled with the ceiling effect in a strong baseline may explain the lack of improvements.

(ii) Continuous is better than discrete: For PP-attachment, sarcasm, and sentiment, continuous adaptation is better than discrete in all but three of the eighteen configurations. Binning people into discrete groups is hard and lossy; using continuous weights helps avoid this issue. Stance, however, is the one task where discrete works better for most factors. As we show in Section 5.4, demographics and personality scores are themselves highly predictive of stances on issues; we believe this makes the simpler binning approach more reliable than the continuous model.

(iii) Both known and latent factors are helpful: Sarcasm benefits from age, gender and personality based adaptations, while stance benefits from personality. The demographic and personality factors do not help PP-attachment. Language factors help all tasks except POS tagging.

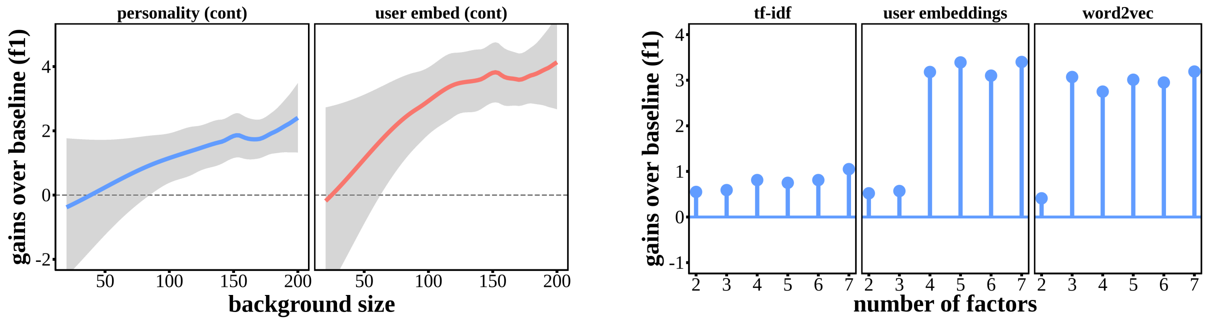
(iv) Latent factors provide best gains: The latent language factors provide the best observed gains for all of the tasks where we saw a significant improvement: PP-attachment, sentiment, sarcasm, and stance. The language factors model users directly in terms of the similarities (and differences) in their entire language use, whereas the inferred demographic and personality factors focus more on a subset of their language as it relates to the particular attribute.

(v) Expanding feature space alone is not enough: One possible explanation for the gains with factors are that the expanded feature space could somehow provide more capacity for learners to generalize. However, adapting to random factors typically lowered results, suggesting that models using more features but not more information naturally take a hit.

5.3 Background Size and Number of Factors

The amount of background available directly affects the factor measurement, which in turn may impact adaptation effectiveness. Figure 2a shows how varying the background size affects adaptation effectiveness for sarcasm. In general, larger backgrounds lead to bigger gains as expected. Even with small amounts of background (50 tweets) adaptation can provide gains, suggesting that even with imperfect predictions of the user attributes, there is still some benefit to adaptation.

Figure 2b compares how performance varies with the number of latent factors for sarcasm. We see gains for all *d* sizes we explored. Performance improves with *d* first and then tapers off; its best is +3.4 at *d*=5 and 7. As the number of factors increases, there is greater potential for a fine-grained characterization of language use differences. However, this is offset by the increased risk of overwhelming the learner with too many parameters to learn during adaptation. We also find that the impact of number of factors also varies with the type of task (e.g., for PP-attachment we find *d*=3 gives the best performance of 72.2, a +1.2 gain over the baseline).



(a) Background size effects for cases with large adaptation gains: sarcasm when using personality and user-embedding factors.

(b) Gains over unadapted baseline for sarcasm using TF-IDF, user embeddings, and word2vec with varying number of factors.

Figure 2: Adaptation performance compared against background size and number of factors.

5.4 Factors as Direct Features

One way to use the factors is to add them as direct features to the instances, without adaptation. Table 4 compares how the most beneficial known factor, personality, performs when added directly as a feature to the two tasks where it had the highest impact.

task	base	direct	adapt	best
sarcasm	73.9	**75.6**†	**75.6**†	**77.3**†
stance	64.9	**65.5**	**67.7**†	**67.9**†

Table 4: Effects of including personality scores as direct features, rather than doing adaptation. Other tasks had no benefit from direct features. Best column shows best result achieved with adaptation using any factor.

For sarcasm, adding personality as a direct feature itself leads to a strong improvement on par with using it for adaptation. For stance, however, we see that while there is an improvement over the baseline, it is not as large as that from adaptation. We observed little-to-no improvement for POS tagging, sentiment or PP-attachment when using personality as direct features. This reflects the relative complexity of the relationships between user factors and class labels for each task. Note that while direct features provide benefits, the overall possible gain with adaptation using any factor (shown in best column) is larger.

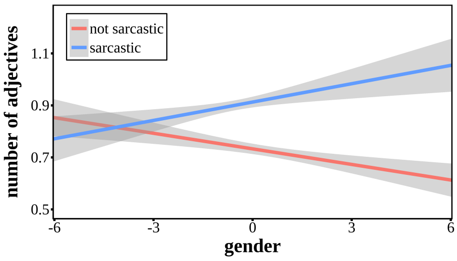


Figure 3: There is a positive correlation between gender and adjective use for sarcastic tweets, and a negative correlation for non-sarcastic tweets. Higher gender scores indicate a greater degree of "femaleness", whereas lower scores represent more "maleness" according to the gender prediction model.

Including user factors as direct features is beneficial when there is a linear relationship with the class label, such as with gender and sarcasm use. In contrast, user-factor adaptation can capture more complex relationships between user groups and their language expression. Figure 3, for instance, shows a three-way interaction between gender scores, adjective use and sarcasm. Increase in the number of adjectives is a positive indicator of sarcasm for women (high gender scores) but is a negative indicator for men (low gender scores). We observe similar trends for age: phrases such as "thanks" and "love it" tend to be meant sarcastically by younger users and sincerely by older ones. User-factor adaptation can model these interaction relationships when direct features alone may not.

5.5 Comparison of Latent Representations

To understand the advantage of continuous latent adaptation, we look at how well discrete and continuous factor representations capture meaningful information about users. To do so, we select two dimensions from the TF-IDF latent factors for stance detection and examine the extent to which they differentiate users based on their attributes (i.e. age) and posting behavior (i.e. URL use). This is shown in Figure 4. The top row gives the

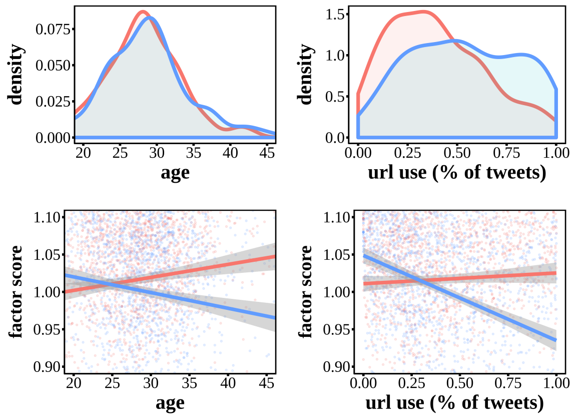


Figure 4: Kernel densities (top) and scatter plots (bottom) of users' age and use of URLs broken down by TF-IDF latent dimension. Colors represent each dimension and are consistent across plots. Lines in scatter plots represent best-fit linear regression. Shaded regions indicate standard error. On the left, discrete latent factors do not seem to distinguish by age (top) but come apart when treated continuously (bottom). On the other hand, discrete and continuous seem to partially capture a dimension of how often someone mentions URLs.

discrete representation: kernel density plots show age and URL use distributions for users binned into the two factor dimensions, shown here in red and blue. The bottom gives the continuous representation: scatter plots show the relationship between age and URL use and the factor score for each dimension.

In the discrete view, age distributions are similar for both factors; there is no apparent relationship between factor membership and age. However, in the continuous view there is a clear negative correlation for age with the factor score for blue and a positive one for red. This indicates that the factors are capturing meaningful information about user age: those with a high factor score for blue tend to be younger, whereas those with a low factor score are older. The reverse is true for red. The URL use shows some difference between the two dimensions in the discrete view, and again we see strong and differing linear relationships with the continuous view.

Overall, the latent factors appear to capture both user attributes and posting behavior, with the continuous version providing additional benefits in characterizing these relationships. The lack of a clear differentiation in the discrete case hints at the difficulty in capturing linguistic variance by splitting users into discrete groups.

6 Related Work

Modeling users has a long history of successful applications in providing personalized information access (Dou et al., 2007; Teevan et al., 2005) and recommendations (Guy et al., 2009; Li et al., 2010; Morales et al., 2012). In contrast, this work models users to better understand their content via language processing tasks following ideas from demographics-aware and domain adaptation.

User-level language variance affects lexical choices (Preoţiuc-Pietro et al., 2016) and even syntactic aspects of language (Johannsen et al., 2015). Such variations can result in demographics-based bias in low-level tasks such as POS tagging (Hovy and Søgaard, 2015) and can also impact high-level applications such as sentiment analysis (Volkova et al., 2013) and machine translation (Mirkin et al., 2015), motivating demographics and personality-based adaptations.

Consequently, recent works have explored demographics-aware NLP (Volkova et al., 2013; Bamman et al., 2014a; Kulkarni et al., 2016; Hovy, 2015; Yang and Eisenstein, 2015). Volkova et al. (2013) propose a gender-aware model and demonstrate superior performance over a gender-agnostic model on the task of sentiment analysis. Bamman et al. (2014a) and Kulkarni et al. (2016) analyze regional linguistic variation using region-specific word embeddings on online social media. Hovy (2015) advances this line of research further and learns group-specific word embeddings, showing improvements over general embeddings on three types of text classification tasks. When author demographics are not available, Yang and Eisenstein (2015) show that learning community-specific embeddings using social networks can help improve sentiment analysis. A similar approach with a social theory-based optimization also showed improvements for sentiment analysis (Hu et al., 2013). For sarcasm detection, historical information about the author and their past context (e.g. entities they discuss) have been shown to be helpful (Bamman and Smith, 2015; Khattri et al., 2015; Rajadesingan et al., 2015).

Our work builds on these ideas and explores the general task of user-factor adaptation. Compared to past work, our method (a) is more general – working with both continuous and discrete factors, (b) uses factors beyond demographics – characteristics like personality are known to influence language beyond demographics (Schwartz et al., 2013), and (c) only requires a background of language – by using inferred factors from a background of language, we require no *a priori* knowledge of user traits.

7 Conclusion

Language on social media reflects the diversity in its user base and motivates the need for robust models that can handle the resulting variations by user attributes. We have introduced user-factor adaptation, a method to adapt typical supervised language classifiers based on factors of the user authoring the language. Our approach requires nothing more than a background of language by the user and only needs access to the features used by the supervised learner. Since it requires no other modifications to the learner, our approach can be easily applied to many NLP tasks.

To the best of our knowledge, this represents the first work to use the idea of continuous-valued variables for language processing adaptation. Continuous adaptation to a variety of user factors brings us closer to personalized NLP and outperforms discrete adaptation over four different tasks: part-of-speech tagging, preposition-phrase attachment, sentiment analysis, and sarcasm detection. Adaptations with data-driven latent factors produced the largest gains. We see this work as part of a growing trend to put language not just within its document-wide context, but also within the context of the human being that wrote it.

Acknowledgments

This publication was made possible, in part, through the support of a grant from the Templeton Religion Trust – TRT0048. We wish to thank the following colleagues for their annotation help for the PP-attachment task: Chetan Naik, Heeyoung Kwon, Ibrahim Hammoud, Jun Kang, Masoud Rouhizadeh, Mohammadzaman Zamani, and Samuel Louvan.

References

Shlomo Argamon, Moshe Koppel, James W. Pennebaker, and Jonathan Schler. 2007. Mining the blogosphere: Age, gender and the varieties of self-expression. *First Monday*, 12(9).

David Bamman, Chris Dyer, and Noah A. Smith. 2014a. Distributed representations of geographically situated language. *Proceedings of ACL*, pages 828–834.

David Bamman, Jacob Eisenstein, and Tyler Schnoebelen. 2014b. Gender identity and lexical variation in social media. *Journal of Sociolinguistics*, 18(2):135–160.

David Bamman and Noah A Smith. 2015. Contextualized sarcasm detection on Twitter. In *Ninth International AAAI Conference on Web and Social Media*.

Yonatan Belinkov, Tao Lei, Regina Barzilay, and Amir Globerson. 2014. Exploring compositional architectures and word vector representations for prepositional phrase attachment. *TACL*, 2:561–572.

Jenny Cheshire. 2005. Syntactic variation and beyond: Gender and social class variation in the use of discourse-new markers. *Journal of Sociolinguistics*, 9(4):479–508.

Dennis Child. 1990. *The Essentials of Factor Analysis*. Cassell Educational.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of ACL*.

Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. 2007. A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of WWW*.

Maeve Duggan and Aaron Smith. 2013. Demographics of key social networking platforms. *Pew Research on Social Media*.

Lewis R. Goldberg. 1990. An alternative "description of personality": The Big-Five factor structure. *Journal of Personality and Social Psychology*, 59(6):1216.

Ido Guy, Naama Zwerdling, David Carmel, Inbal Ronen, Erel Uziel, Sivan Yogev, and Shila Ofek-Koifman. 2009. Personalized recommendation of social software items based on social relations. In *Proceedings of RecSys*.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of ACL*.

Dirk Hovy and Anders Søgaard. 2015. Tagging performance correlates with author age. In *Proceedings of ACL*.

Xia Hu, Lei Tang, Jiliang Tang, and Huan Liu. 2013. Exploiting social relations for sentiment analysis in microblogging. In *Proceedings of WSDM*.

Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of KDD*, pages 217–226. ACM.

Anders Johannsen, Dirk Hovy, and Anders Søgaard. 2015. Cross-lingual syntactic variation over age and gender. In *Proceedings of CONLL*.

Anupam Khattri, Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. 2015. Your sentiment precedes you: Using an author's historical tweets to predict sarcasm. In *Sixth Workshop* *on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*, page 25.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proceedings of EMNLP*.

Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. 2016. Freshman or fresher? Quantifying the geographic variation of language in online social media. In *Tenth International AAAI Conference on Web and Social Media*.

Derrick Norman Lawley and Albert Ernest Maxwell. 1971. *Factor analysis as a statistical method*, volume 18. JSTOR.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of WWW*.

Robert R. McCrae and Paul T. Costa Jr. 1989. Reinterpreting the Myers-Briggs type indicator from the perspective of the five-factor model of personality. *Journal of Personality*, 57(1):17–40.

Robert R. McCrae and Paul T. Costa Jr. 1997. Personality trait structure as a human universal. *American Psychologist*, 52(5):509.

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

Shachar Mirkin, Scott Nowson, Caroline Brun, and Julien Perez. 2015. Motivating personality-aware machine translation. In *Proceedings of EMNLP*.

Saif M. Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. SemEval-2016 task 6: Detecting stance in tweets. In *Proceedings of SemEval-2016*, volume 16.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of SemEval-2013*.

Gianmarco De Francisci Morales, Aristides Gionis, and Claudio Lucchese. 2012. From chatter to headlines: Harnessing the real-time web for personalized news recommendation. In *Proceedings of WSDM*.

Preslav Nakov, Zornitsa Kozareva, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Theresa Wilson. 2013. SemEval-2013 task 2: Sentiment analysis in Twitter.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL-HLT*.

Gregory Park, H. Andrew Schwartz, Johannes C. Eichstaedt, Margaret L. Kern, Michal Kosinski, David J. Stillwell, Lyle H. Ungar, and Martin E. P. Seligman. 2015. Automatic personality assessment through social media language. *Journal of Personality and Social Psychology*, 108(6):934.

James W. Pennebaker and Lori D. Stone. 2003. Words of wisdom: Language use over the life span. *Journal of Personality and Social Psychology*, 85(2):291.

Daniel Preoţiuc-Pietro, Wei Xu, and Lyle Ungar. 2016. Discovering user attribute stylistic differences via paraphrasing. In *Proceedings of AAAI*.

Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In *Proceedings of WSDM*.

John Ruscio and Ayelet Meron Ruscio. 2000. Informing the continuity controversy: A taxometric analysis of depression. *Journal of Abnormal Psychology*, 109(3):473–487.

Maarten Sap, Gregory J. Park, Johannes C. Eichstaedt, Margaret L. Kern, David Stillwell, Michal Kosinski, Lyle H. Ungar, and Hansen Andrew Schwartz. 2014. Developing age and gender predictive lexica over social media. In *Proceedings of EMNLP*.

H. Andrew Schwartz, Johannes C. Eichstaedt, Margaret L. Kern, Lukasz Dziurzynski, Stephanie M. Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin E. P. Seligman, and Lyle H. Ungar. 2013. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS ONE*, 8(9):e73791.

Jaime Teevan, Susan T. Dumais, and Eric Horvitz. 2005. Personalizing search via automated analysis of interests and activities. In *SIGIR*.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *Proceedings of EMNLP*.

Thomas A. Widiger and Douglas B. Samuel. 2005. Diagnostic categories or dimensions? A question for the Diagnostic and Statistical Manual of Mental Disorders—Fifth Edition. *Journal of Abnormal Psychology*, 114(4):494.

Yi Yang and Jacob Eisenstein. 2015. Putting things in context: Community-specific embedding projections for sentiment analysis. *CoRR*, abs/1511.06052.

Neural Sequence Learning Models for Word Sense Disambiguation

Alessandro Raganato, Claudio Delli Bovi and **Roberto Navigli**
Department of Computer Science
Sapienza University of Rome
{raganato,dellibovi,navigli}@di.uniroma1.it

Abstract

Word Sense Disambiguation models exist in many flavors. Even though supervised ones tend to perform best in terms of accuracy, they often lose ground to more flexible knowledge-based solutions, which do not require training by a word expert for every disambiguation target. To bridge this gap we adopt a different perspective and rely on sequence learning to frame the disambiguation problem: we propose and study in depth a series of end-to-end neural architectures directly tailored to the task, from bidirectional Long Short-Term Memory to encoder-decoder models. Our extensive evaluation over standard benchmarks and in multiple languages shows that sequence learning enables more versatile all-words models that consistently lead to state-of-the-art results, even against word experts with engineered features.

1 Introduction

As one of the long-standing challenges in Natural Language Processing (NLP), Word Sense Disambiguation (Navigli, 2009, WSD) has received considerable attention over recent years. Indeed, by dealing with lexical ambiguity an effective WSD model brings numerous benefits to a variety of downstream tasks and applications, from Information Retrieval and Extraction (Zhong and Ng, 2012; Delli Bovi et al., 2015) to Machine Translation (Carpuat and Wu, 2007; Xiong and Zhang, 2014; Neale et al., 2016). Recently, WSD has also been leveraged to build continuous vector representations for word senses (Chen et al., 2014; Iacobacci et al., 2015; Flekova and Gurevych, 2016).

Inasmuch as WSD is described as the task of associating words in context with the most suitable entries in a pre-defined sense inventory, the majority of WSD approaches to date can be grouped into two main categories: supervised (or semi-supervised) and knowledge-based. Supervised models have been shown to consistently outperform knowledge-based ones in all standard benchmarks (Raganato et al., 2017), at the expense, however, of harder training and limited flexibility. First of all, obtaining reliable sense-annotated corpora is highly expensive and especially difficult when non-expert annotators are involved (de Lacalle and Agirre, 2015), and as a consequence approaches based on unlabeled data and semi-supervised learning are emerging (Taghipour and Ng, 2015b; Başkaya and Jurgens, 2016; Yuan et al., 2016; Pasini and Navigli, 2017).

Apart from the shortage of training data, a crucial limitation of current supervised approaches is that a dedicated classifier (*word expert*) needs to be trained for every target lemma, making them less flexible and hampering their use within end-to-end applications. In contrast, knowledge-based systems do not require sense-annotated data and often draw upon the structural properties of lexico-semantic resources (Agirre et al., 2014; Moro et al., 2014; Weissenborn et al., 2015). Such systems construct a model based only on the underlying resource, which is then able to handle multiple target words at the same time and disambiguate them jointly, whereas word experts are forced to treat each disambiguation target in isolation.

In this paper our focus is on supervised WSD, but we depart from previous approaches and adopt a different perspective on the task: instead of framing a separate classification problem for each given word, we aim at modeling the joint disambiguation of the target text as a whole in terms of a sequence labeling problem. From this standpoint, WSD amounts to translating a sequence of words into a sequence of potentially sense-tagged tokens.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1156–1167
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

With this in mind, we design, analyze and compare experimentally various neural architectures of different complexities, ranging from a single bidirectional Long Short-Term Memory (Graves and Schmidhuber, 2005, LSTM) to a sequence-to-sequence approach (Sutskever et al., 2014). Each architecture reflects a particular way of modeling the disambiguation problem, but they all share some key features that set them apart from previous supervised approaches to WSD: they are trained end-to-end from sense-annotated text to sense labels, and learn a single all-words model from the training data, without fine tuning or explicit engineering of local features.

The contributions of this paper are twofold. First, we show that neural sequence learning represents a novel and effective alternative to the traditional way of modeling supervised WSD, enabling a single all-words model to compete with a pool of word experts and achieve state-of-the-art results, while also being easier to train, arguably more versatile to use within downstream applications, and directly adaptable to different languages without requiring additional sense-annotated data (as we show in Section 6.2); second, we carry out an extensive experimental evaluation where we compare various neural architectures designed for the task (and somehow left underinvestigated in previous literature), exploring different configurations and training procedures, and analyzing their strengths and weaknesses on all the standard benchmarks for all-words WSD.

2 Related Work

The literature on WSD is broad and comprehensive (Agirre and Edmonds, 2007; Navigli, 2009): new models are continuously being developed (Yuan et al., 2016; Tripodi and Pelillo, 2017; Butnaru et al., 2017) and tested over a wide variety of standard benchmarks (Edmonds and Cotton, 2001; Snyder and Palmer, 2004; Pradhan et al., 2007; Navigli et al., 2007, 2013; Moro and Navigli, 2015). Moreover, the field has been explored in depth from different angles by means of extensive empirical studies and evaluation frameworks (Pilehvar and Navigli, 2014; Iacobacci et al., 2016; McCarthy et al., 2016; Raganato et al., 2017).

As regards supervised WSD, traditional approaches are generally based on extracting local features from the words surrounding the target, and then training a classifier (Zhong and Ng, 2010; Shen et al., 2013) for each target lemma. In their latest developments, these models include more complex features based on word embeddings (Taghipour and Ng, 2015b; Rothe and Schütze, 2015; Iacobacci et al., 2016).

The recent upsurge of neural networks has also contributed to fueling WSD research: Yuan et al. (2016) rely on a powerful neural language model to obtain a latent representation for the whole sentence containing a target word w; their instance-based system then compares that representation with those of example sentences annotated with the candidate meanings of w. Similarly, Context2Vec (Melamud et al., 2016) makes use of a bidirectional LSTM architecture trained on an unlabeled corpus and learns a context vector for each sense annotation in the training data. Finally, Kågebäck and Salomonsson (2016) present a supervised classifier based on bidirectional LSTM for the lexical sample task (Kilgarriff, 2001; Mihalcea et al., 2004). All these contributions have shown that supervised neural models can achieve state-of-the-art performances without taking advantage of external resources or language-specific features. However, they all consider each target word as a separate classification problem and, to the best of our knowledge, very few attempts have been made to disambiguate a text jointly using sequence learning (Ciaramita and Altun, 2006).

Sequence learning, especially using LSTM (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005; Graves, 2013), has become a well-established standard in numerous NLP tasks (Zhou and Xu, 2015; Ma and Hovy, 2016; Wang and Chang, 2016). In particular, sequence-to-sequence models (Sutskever et al., 2014) have grown increasingly popular and are used extensively in, e.g., Machine Translation (Cho et al., 2014; Bahdanau et al., 2015), Sentence Representation (Kiros et al., 2015), Syntactic Parsing (Vinyals et al., 2015), Conversation Modeling (Vinyals and Le, 2015), Morphological Inflection (Faruqui et al., 2016) and Text Summarization (Gu et al., 2016). In line with this trend, we focus on the (so far unexplored) context of supervised WSD, and investigate state-of-the-art all-words approaches that are based on neural sequence learning and capable of disambiguating all target content words within an input text, a key feature in several knowledge-based approaches.

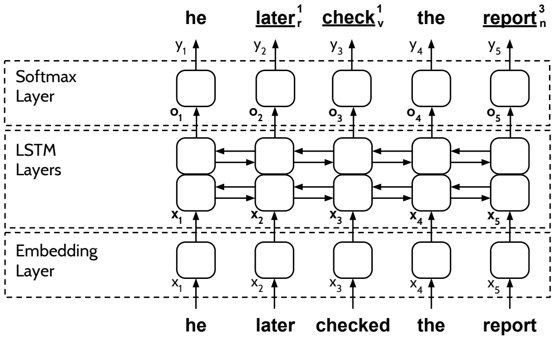


Figure 1: Bidirectional LSTM sequence labeling architecture for WSD (2 hidden layers). We use the notation of Navigli (2009) for word senses: w_p^i is the i-th sense of w with part of speech p.

3 Sequence Learning for Word Sense Disambiguation

In this section we define WSD in terms of a sequence learning problem. While in its classical formulation (Navigli, 2009) WSD is viewed as a classification problem for a given word w in context, with word senses of w being the class labels, here we consider a variable-length sequence of input symbols $\vec{x} = \langle x_1, ..., x_T \rangle$ and we aim at predicting a sequence of output symbols $\vec{y} = \langle y_1, ..., y_{T'} \rangle$.[1] Input symbols are word tokens drawn from a given vocabulary V.[2] Output symbols are either drawn from a pre-defined sense inventory S (if the corresponding input symbols are open-class content words, i.e., nouns, verbs, adjectives or adverbs), or from the same input vocabulary V (e.g., if the corresponding input symbols are function words, like prepositions or determiners). Hence, we can define a WSD model in terms of a function that maps sequences of symbols $x_i \in V$ into sequences of symbols $y_j \in O = S \cup V$.

Here all-words WSD is no longer broken down into a series of distinct and separate classification tasks (one per target word) but rather treated directly at the sequence level, with a single model handling all disambiguation decisions. In what follows, we describe three different models for accomplishing this: a traditional LSTM-based model (Section 3.1), a variant that incorporates an attention mechanism (Section 3.2), and an encoder-decoder architecture (Section 3.3).

3.1 Bidirectional LSTM Tagger

The most straightforward way of modeling WSD as formulated in Section 3 is that of considering a sequence labeling architecture that tags each symbol $x_i \in V$ in the input sequence with a label $y_j \in O$. Even though the formulation is rather general, previous contributions (Melamud et al., 2016; Kågebäck and Salomonsson, 2016) have already shown the effectiveness of recurrent neural networks for WSD. We follow the same line and employ a bidirectional LSTM architecture: in fact, important clues for disambiguating a target word could be located anywhere in the context (not necessarily before the target) and for a model to be effective it is crucial that it exploits information from the whole input sequence at every time step.

Architecture. A sketch of our bidirectional LSTM tagger is shown in Figure 1. It consists of:

- An embedding layer that converts each word $x_i \in \vec{x}$ into a real-valued d-dimensional vector $\mathbf{x}_i$ via the embedding matrix $\mathbf{W} \in \mathbb{R}^{d \times |V|}$;
- One or more stacked layers of bidirectional LSTM (Graves and Schmidhuber, 2005). The hidden state vectors $\mathbf{h}_i$ and output vectors $\mathbf{o}_i$ at the i^{th} time step are then obtained as the concatenations of the forward and backward pass vectors $\overrightarrow{\mathbf{h}}_i, \overrightarrow{\mathbf{o}}_i$ and $\overleftarrow{\mathbf{h}}_i, \overleftarrow{\mathbf{o}}_i$;
- A fully-connected layer with softmax activation that turns the output vector $\mathbf{o}_i$ at the i^{th} time step into a probability distribution over the output vocabulary O.

Training. The tagger is trained on a dataset of N labeled sequences $\{(\vec{x}_k, \vec{y}_k)\}_{k=1}^{N}$ directly obtained from the sentences of a sense-annotated corpus, where each $\vec{x}_k$ is a sequence of word tokens, and each $\vec{y}_k$ is a sequence containing both word tokens and sense labels. Ideally $\vec{y}_k$ is a copy of $\vec{x}_k$ where each content word is sense-tagged. This is, however, not the case in many real-world datasets, where only a subset of the content words is annotated; hence the architecture is designed to deal with both fully and partially annotated sentences. Apart from sentence splitting and tokenization, no preprocessing is required on the training data.

[1] In general $\vec{x}$ and $\vec{y}$ might have different lengths, e.g., if $\vec{x}$ contains a multi-word expression (*European Union*) which is mapped to a unique sense identifier ($\texttt{European Union}^1_n$).

[2] V generalizes traditional vocabularies used in WSD and includes both word lemmas and inflected forms.

3.2 Attentive Bidirectional LSTM Tagger

The bidirectional LSTM tagger of Section 3.1 exploits information from the whole input sequence $\vec{x}$, which is encoded in the hidden state $\mathbf{h}_i$. However, certain elements of $\vec{x}$ might be more discriminative than others in predicting the output label at a given time step (e.g., the syntactic subject and object when predicting the sense label of a verb).

We model this hunch by introducing an attention mechanism, already proven to be effective in other NLP tasks (Bahdanau et al., 2015; Vinyals et al., 2015), into the sequence labeling architecture of Section 3.1. The resulting *attentive* bidirectional LSTM tagger augments the original architecture with an attention layer, where a context vector $\mathbf{c}$ is computed from all the hidden states $\mathbf{h}_1, ..., \mathbf{h}_T$ of the bidirectional LSTM. The attentive tagger first reads the entire input sequence $\vec{x}$ to construct $\mathbf{c}$, and then exploits $\mathbf{c}$ to predict the output label y_j at each time step, by concatenating it with the output vector $\mathbf{o}_j$ of the bidirectional LSTM (Figure 2).

We follow previous work (Vinyals et al., 2015; Zhou et al., 2016) and compute $\mathbf{c}$ as the weighted sum of the hidden state vectors $\mathbf{h}_1, ..., \mathbf{h}_T$. Formally, let $H \in \mathbb{R}^{n \times T}$ be the matrix of hidden state vectors $[\mathbf{h}_1, ..., \mathbf{h}_T]$, where n is the hidden state dimension and T is the input sequence length (cf. Section 3). $\mathbf{c}$ is obtained as follows:

$$\begin{aligned}\mathbf{u} &= \omega^T \tanh(H)\\ \mathbf{a} &= softmax(\mathbf{u})\\ \mathbf{c} &= H\mathbf{a}^T \qquad (1)\end{aligned}$$

where $\omega \in \mathbb{R}^n$ is a parameter vector, and $\mathbf{a} \in \mathbb{R}^T$ is the vector of normalized attention weights.

3.3 Sequence-to-Sequence Model

The attentive tagger of Section 3.2 performs a two-pass procedure by first reading the input sequence $\vec{x}$ to construct the context vector $\mathbf{c}$, and then predicting an output label y_j for each element in $\vec{x}$. In this respect, the attentive architecture can effectively be viewed as an encoder for $\vec{x}$. A further generalization of this model would then be a complete encoder-decoder architecture (Sutskever et al., 2014) where WSD is treated as a sequence-to-sequence mapping (*sequence-to-sequence WSD*), i.e., as the "translation" of word sequences into sequences of potentially sense-tagged tokens.

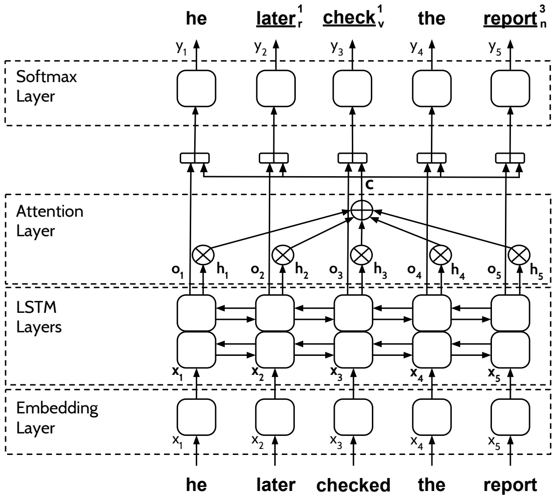


Figure 2: Attentive bidirectional LSTM sequence labeling architecture for WSD (2 hidden layers).

In the sequence-to-sequence framework, a variable-length sequence of input symbols $\vec{x}$ is represented as a sequence of vectors $\vec{\mathbf{x}} = \langle \mathbf{x}_1, ..., \mathbf{x}_T \rangle$ by converting each symbol $x_i \in \vec{x}$ into a real-valued vector $\mathbf{x}_i$ via an embedding layer, and then fed to an encoder, which generates a fixed-dimensional vector representation of the sequence. Traditionally, the encoder function is a Recurrent Neural Network (RNN) such that:

$$\begin{aligned}\mathbf{h}_t &= f(\mathbf{h}_{t-1}, \mathbf{x}_t)\\ \mathbf{c} &= q(\{\mathbf{h}_1, ..., \mathbf{h}_T\}) \qquad (2)\end{aligned}$$

where $\mathbf{h}_t \in \mathbb{R}^n$ is the n-dimensional hidden state vector at time t, $\mathbf{c} \in \mathbb{R}^n$ is a vector generated from the whole sequence of input states, and f and q are non-linear functions.[3] A decoder is then trained to predict the next output symbol y_t given the encoded input vector $\mathbf{c}$ and all the previously predicted output symbols $\langle y_1, ..., y_{t-1} \rangle$. More formally, the decoder defines a probability over the output sequence $\vec{y} = \langle y_1, ..., y_{T'} \rangle$ by decomposing the joint probability into ordered conditionals:

$$p(\vec{y} \mid \vec{x}) = \prod_{t=1}^{T'} p(y_t \mid \mathbf{c}, \langle y_1, ..., y_{t-1} \rangle) \qquad (3)$$

Typically a decoder RNN defines the hidden state at time t as $\mathbf{s}_t = g(\mathbf{s}_{t-1}, \{\mathbf{c}, y_{t-1}\})$ and then feeds $\mathbf{s}_t$ to a softmax layer in order to obtain a conditional probability over output symbols.

[3] For instance, Sutskever et al. (2014) used an LSTM as f, and $q(\{\mathbf{h}_1, ..., \mathbf{h}_T\}) = \mathbf{h}_T$.

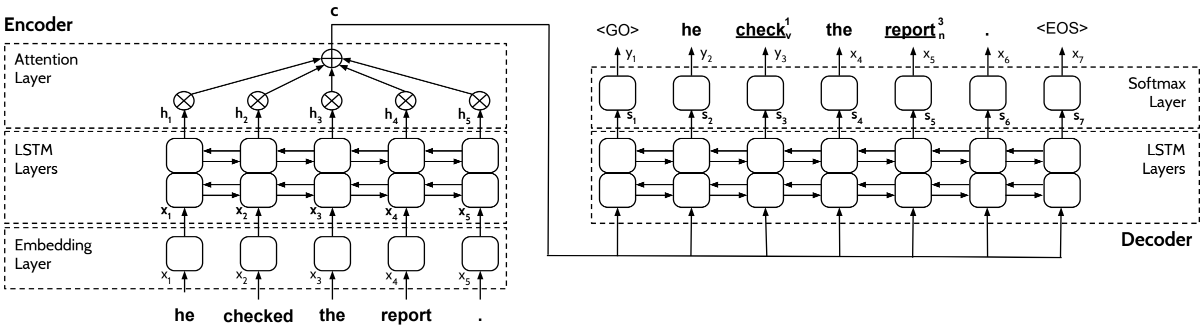


Figure 3: Encoder-decoder architecture for sequence-to-sequence WSD, with 2 bidirectional LSTM layers and an attention layer.

In the context of WSD framed as a sequence learning problem, a sequence-to-sequence model takes as input a training set of labeled sequences (cf. Section 3.1) and learns to replicate an input sequence $\vec{x}$ while replacing each content word with its most suitable word sense from S. In other words, sequence-to-sequence WSD can be viewed as the combination of two sub-tasks:

- A *memorization* task, where the model learns to replicate the input sequence token by token at decoding time;
- The actual *disambiguation* task where the model learns to replace content words across the input sequence with their most suitable senses from the sense inventory S.

In the latter stage, multi-word expressions (such as nominal entity mentions or phrasal verbs) are replaced by their sense identifiers, hence yielding an output sequence that might have a different length than $\vec{x}$.

Architecture. The encoder-decoder architecture generalizes over both the models in Sections 3.1 and 3.2. In particular, we include one or more bidirectional LSTM layers at the core of both the encoder and the decoder modules. The encoder utilizes an embedding layer (cf. Section 3.1) to convert input symbols into embedded representations, feeds it to the bidirectional LSTM layer, and then constructs the context vector $\mathbf{c}$, either by simply letting $\mathbf{c} = \mathbf{h}_T$ (i.e., the hidden state of the bidirectional LSTM layer after reading the whole input sequence), or by computing the weighted sum described in Section 3.2 (if an attention mechanism is employed). In either case, the context vector $\mathbf{c}$ is passed over to the decoder, which generates the output symbols sequentially based on $\mathbf{c}$ and the current hidden state $\mathbf{s}_t$, using one or more bidirectional LSTM layers as in the encoder module. Instead of feeding $\mathbf{c}$ to the decoder only at the first time step (Sutskever et al., 2014; Vinyals and Le, 2015), we condition each output symbol y_t on $\mathbf{c}$, allowing the decoder to peek into the input at every step, as in Cho et al. (2014). Finally, a fully-connected layer with softmax activation converts the current output vector of the last LSTM layer into a probability distribution over the output vocabulary O. The complete encoder-decoder architecture (including the attention mechanism) is shown in Figure 3.

4 Multitask Learning with Multiple Auxiliary Losses

Several recent contributions (Søgaard and Goldberg, 2016; Bjerva et al., 2016; Plank et al., 2016; Luong et al., 2016) have shown the effectiveness of *multitask learning* (Caruana, 1997, MTL) in a sequence learning scenario. In MTL the idea is that of improving generalization performance by leveraging training signals contained in related tasks, in order to exploit their commonalities and differences. MTL is typically carried out by training a single architecture using multiple loss functions and a shared representation, with the underlying intention of improving a main task by incorporating joint learning of one or more related auxiliary tasks. From a practical point of view, MTL works by including one task-specific output layer per additional task, usually at the outermost level of the architecture, while keeping the remaining hidden layers common across all tasks.

In line with previous approaches, and guided by the intuition that WSD is strongly linked to other NLP tasks at various levels, we also design and study experimentally a multitask augmentation of the models described in Section 3. In particular, we consider two auxiliary tasks:

- **Part-of-speech (POS) tagging**, a standard auxiliary task extensively studied in previous work (Søgaard and Goldberg, 2016; Plank et al., 2016). Predicting the part-of-speech tag for a given token can also be informative for word senses, and help in dealing with cross-POS lexical ambiguities (e.g., *book a flight* vs. *reading a good book*);

- **Coarse-grained semantic labels (LEX)** based on the WordNet (Miller et al., 1990) lexicographer files,[4] i.e., 45 coarse-grained semantic categories manually associated with all the synsets in WordNet on the basis of both syntactic and logical groupings (e.g., *noun.location*, or *verb.motion*). These very coarse semantic labels, recently employed in a multitask setting by Alonso and Plank (2017), group together related senses and help the model to generalize, especially over senses less covered at training time.

We follow previous work (Plank et al., 2016; Alonso and Plank, 2017) and define an auxiliary loss function for each additional task. The overall loss is then computed by summing the main loss (i.e., the one associated with word sense labels) and all the auxiliary losses taken into account.

As regards the architecture, we consider both the models described in Sections 3.2 and 3.3 and modify them by adding two softmax layers in addition to the one in the original architecture. Figure 4 illustrates this for the attentive tagger of Section 3.2, considering both POS and LEX as auxiliary tasks. At the j^{th} time step the model predicts a sense label y_j together with a part-of-speech tag POS_j and a coarse semantic label LEX_j.[5]

5 Experimental Setup

In this section we detail the setup of our experimental evaluation. We first describe the training corpus and all the standard benchmarks for all-words WSD; we then report technical details on the architecture and on the training process for all the models described throughout Section 3 and their multitask augmentations (Section 4).

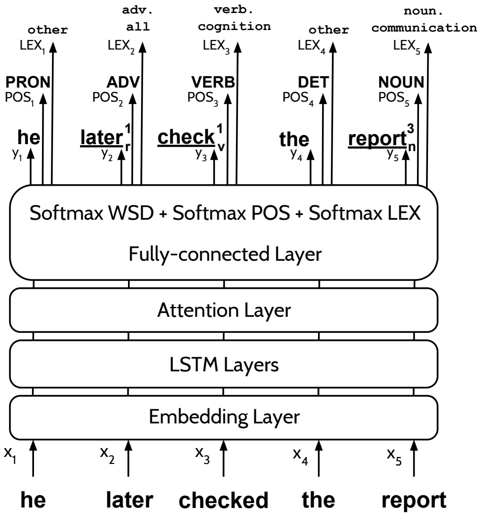


Figure 4: Multitask augmentation (with both POS and LEX as auxiliary tasks) for the attentive bidirectional LSTM tagger of Section 3.2.

Evaluation Benchmarks. We evaluated our models on the English all-words WSD task, considering both the fine-grained and coarse-grained benchmarks (Section 6.1). As regards fine-grained WSD, we relied on the evaluation framework of Raganato et al. (2017), which includes five standardized test sets from the Senseval/SemEval series: Senseval-2 (Edmonds and Cotton, 2001, **SE2**), Senseval-3 (Snyder and Palmer, 2004, **SE3**), SemEval-2007 (Pradhan et al., 2007, **SE07**), SemEval-2013 (Navigli et al., 2013, **SE13**) and SemEval-2015 (Moro and Navigli, 2015, **SE15**). Due to the lack of a reasonably large development set for our setup, we considered the smallest among these test sets, i.e., **SE07**, as development set and excluded it from the evaluation of Section 6.1. As for coarse-grained WSD, we used the SemEval-2007 task 7 test set (Navigli et al., 2007), which is not included in the standardized framework, and mapped the original sense inventory from WordNet 2.1 to WordNet 3.0.[6] Finally, we carried out an experiment on multilingual WSD using the Italian, German, French and Spanish data of **SE13**. For these benchmarks we relied on BabelNet (Navigli and Ponzetto, 2012)[7] as unified sense inventory.

[4] https://wordnet.princeton.edu/man/lexnames.5WN.html

[5] We use a dummy LEX label (other) for punctuation and function words.

[6] We utilized the original sense-key mappings available at http://wordnetcode.princeton.edu/3.0 for nouns and verbs, and the automatic mappings by Daudé et al. (2003) for the remaining parts of speech (not available in the original mappings).

[7] http://babelnet.org

	Dev	Test Datasets				Concatenation of All Test Datasets				
	SE07	**SE2**	**SE3**	**SE13**	**SE15**	**Nouns**	**Verbs**	**Adj.**	**Adv.**	**All**
BLSTM	61.8	71.4	68.8	65.6	69.2	70.2	56.3	75.2	**84.4**	68.9
BLSTM + att.	62.4	71.4	**70.2**	66.4	70.8	71.0	**58.4**	75.2	83.5	69.7
BLSTM + att. + LEX	63.7	**72.0**	69.4	66.4	**72.4**	**71.6**	57.1	**75.6**	83.2	**69.9**
BLSTM + att. + LEX + POS	**64.8**	**72.0**	69.1	**66.9**	71.5	71.5	57.5	75.0	83.8	**69.9**
Seq2Seq	60.9	68.5	67.9	65.3	67.0	68.7	54.5	74.0	81.2	67.3
Seq2Seq + att.	62.9	69.9	69.6	65.6	67.7	69.5	57.2	74.5	81.8	68.4
Seq2Seq + att. + LEX	64.6	70.6	67.8	66.5	68.7	70.4	55.7	73.3	82.9	68.5
Seq2Seq + att. + LEX + POS	63.1	70.1	68.5	66.5	69.2	70.1	55.2	75.1	84.4	68.6
IMS	61.3	70.9	69.3	65.3	69.5	70.5	55.8	75.6	82.9	68.9
IMS+emb	**62.6**	**72.2**	**70.4**	65.9	71.5	**71.9**	56.6	**75.9**	**84.7**	**70.1**
Context2Vec	61.3	71.8	69.1	65.6	**71.9**	71.2	**57.4**	75.2	82.7	69.6
Lesk$_{ext}$+emb	$\star$56.7	63.0	63.7	66.2	64.6	70.0	51.1	51.7	80.6	64.2
UKB$_{gloss}$ w2w	42.9	63.5	55.4	$\star$62.9	63.3	64.9	41.4	69.5	69.7	61.1
Babelfy	51.6	$\star$67.0	63.5	**66.4**	70.3	68.9	50.7	73.2	79.8	66.4
MFS	54.5	65.6	$\star$66.0	63.8	$\star$67.1	67.7	49.8	73.1	80.5	65.5

Table 1: F-scores (%) for English all-words fine-grained WSD on the test sets in the framework of Raganato et al. (2017) (including the development set **SE07**). The first system with a statistically significant difference from our best models is marked with $\star$ (unpaired t-test, $p < 0.05$).

At testing time, given a target word w, our models used the probability distribution over O, computed by the softmax layer at the corresponding time step, to rank the candidate senses of w; we then simply selected the top ranking candidate as output of the model.

Architecture Details. To set a level playing field with comparison systems on English all-words WSD, we followed Raganato et al. (2017) and, for all our models, we used a layer of word embeddings pre-trained[8] on the English ukWaC corpus (Baroni et al., 2009) as initialization, and kept them fixed during the training process. For all architectures we then employed 2 layers of bidirectional LSTM with 2048 hidden units (1024 units per direction).

As regards multilingual all-words WSD (Section 6.2), we experimented, instead, with two different configurations of the embedding layer: the pre-trained bilingual embeddings by Mrkšić et al. (2017) for all the language pairs of interest (EN-IT, EN-FR, EN-DE, and EN-ES), and the pre-trained multilingual 512-dimensional embeddings for 12 languages by Ammar et al. (2016).

Training. We used SemCor 3.0 (Miller et al., 1993) as training corpus for all our experiments. Widely known and utilized in the WSD literature, SemCor is one of the largest corpora annotated manually with word senses from the sense inventory of WordNet (Miller et al., 1990) for all open-class parts of speech. We used the standardized version of SemCor as provided in the evaluation framework[9] which also includes coarse-grained POS tags from the universal tagset. All models were trained for a fixed number of epochs $E = 40$ using Adadelta (Zeiler, 2012) with learning rate 1.0 and batch size 32. After each epoch we evaluated our models on the development set, and then compared the best iterations (E^*) on the development set with the reported state of the art in each benchmark.

6 Experimental Results

Throughout this section we identify the models based on the LSTM tagger (Sections 3.1-3.2) by the label **BLSTM**, and the sequence-to-sequence models (Section 3.3) by the label **Seq2Seq**.

6.1 English All-words WSD

Table 1 shows the performance of our models on the standardized benchmarks for all-words fine-grained WSD. We report the F1-score on each in-

[8] We followed Iacobacci et al. (2016) and used the Word2Vec (Mikolov et al., 2013) skip-gram model with 400 dimensions, 10 negative samples and a window size of 10.

[9] http://lcl.uniroma1.it/wsdeval

SemEval-2007 task 7			
BLSTM + att. + LEX	83.0	IMS	81.9
BLSTM + att. + LEX + POS	**83.1**	Chen et al. (2014)	82.6
Seq2Seq + att. + LEX	82.3	Yuan et al. (2016)	**82.8**
Seq2Seq + att. + LEX + POS	81.6	UKB w2w	80.1

Table 2: F-scores (%) for coarse-grained WSD.

	SemEval-2013 task 12			
	IT	**FR**	**DE**	**ES**
BLSTM (bilingual)	61.6	55.2	**69.2**	65.0
BLSTM (multilingual)	62.0	55.5	**69.2**	66.4
UMCC-DLSI	**65.8**	**60.5**	62.1	**71.0**
DAEBAK!	61.3	53.8	59.1	60.0
MFS	57.5	45.3	67.4	64.5

Table 3: F-scores (%) for multilingual WSD.

dividual test set, as well as the F1-score obtained on the concatenation of all four test sets, divided by part-of-speech tag.

We compared against the best supervised and knowledge-based systems evaluated on the same framework. As supervised systems, we considered **Context2Vec** (Melamud et al., 2016) and It Makes Sense (Zhong and Ng, 2010, **IMS**), both the original implementation and the best configuration reported by Iacobacci et al. (2016, **IMS+emb**), which also integrates word embeddings using exponential decay.[10] All these supervised systems were trained on the standardized version of SemCor. As knowledge-based systems we considered the embeddings-enhanced version of Lesk by Basile et al. (2014, **Lesk**$_{ext}$**+emb**), UKB (Agirre et al., 2014) (**UKB**$_{gloss}$ **w2w**) , and **Babelfy** (Moro et al., 2014). All these systems relied on the Most Frequent Sense (**MFS**) baseline as back-off strategy.[11] Overall, both **BLSTM** and **Seq2Seq** achieved results that are either state-of-the-art or statistically equivalent (unpaired t-test, $p < 0.05$) to the best supervised system in each benchmark, performing on par with word experts tuned over explicitly engineered features (Iacobacci et al., 2016). Interestingly enough, **BLSTM** models tended consistently to outperform their **Seq2Seq** counterparts, suggesting that an encoder-decoder architecture, despite being more powerful, might be suboptimal for WSD. Furthermore, introducing LEX (cf. Section 4) as auxiliary task was generally helpful; on the other hand, POS did not seem to help, corroborating previous findings (Alonso and Plank, 2017; Bingel and Søgaard, 2017).

The overall performance by part of speech was consistent with the above analysis, showing that our models outperformed all knowledge-based systems, while obtaining results that are superior or equivalent to the best supervised models. It is worth noting that RNN-based architectures outperformed classical supervised approaches (Zhong and Ng, 2010; Iacobacci et al., 2016) when dealing with verbs, which are shown to be highly ambiguous (Raganato et al., 2017).

The performance on coarse-grained WSD followed the same trend (Table 2). Both **BLSTM** and **Seq2Seq** outperformed UKB (Agirre et al., 2014) and IMS trained on SemCor (Taghipour and Ng, 2015a), as well as recent supervised approaches based on distributional semantics and neural architectures (Chen et al., 2014; Yuan et al., 2016).

6.2 Multilingual All-words WSD

All the neural architectures described in this paper can be readily adapted to work with different languages without adding sense-annotated data in the target language. In fact, as long as the first layer (cf. Figures 1-3) is equipped with *bilingual* or *multilingual* embeddings where word vectors in the training and target language are defined in the same space, the training process can be left unchanged, even if based only on English data. The underlying assumption is that words that are translations of each other (e.g., *house* in English and *casa* in Italian) are mapped to word embeddings that are as close as possible in the vector space.

In order to assess this, we considered one of our best models (**BLSTM+att.+LEX**) and replaced the monolingual embeddings with bilingual and multilingual embeddings (as specified in Section 5), leaving the rest of the architecture unchanged. We then trained these architectures on the same English training data, and ran the resulting models on the multilingual benchmarks of SemEval-2013 for Italian, French, German and Spanish. While doing this, we exploited BabelNet's inter-resource mappings to convert WordNet sense labels (used at training time) into BabelNet synsets compliant with the sense inventory of the task.

F-score figures (Table 3) show that bilingual and multilingual models, despite being trained only on English data, consistently outperformed the MFS

[10] We are not including Yuan et al. (2016), as their models are not available and not replicable on the standardized test sets, being based on proprietary data.

[11] Since each system always outputs an answer, F-score equals both precision and recall, and statistical significance can be expressed with respect to any of these measures.

baseline and achieved results that are competitive with the best participating systems in the task. We also note that the overall F-score performance did not change substantially (and slightly improved) when moving from bilingual to multilingual models, despite the increase in the number of target languages treated simultaneously.

6.3 Discussion and Error Analysis

All the neural models evaluated in Section 6.1 utilized the MFS back-off strategy for instances unseen at training time, which amounted to 9.4% overall for fine-grained WSD and 10.5% for coarse-grained WSD. Back-off strategy aside, 85% of the times the top candidate sense for a target instance lay within the 10 most probable entries in the probability distribution over O computed by the softmax layer.[12] In fact, our sequence models learned, on the one hand, to associate a target word with its candidate senses (something word experts are not required to learn, as they only deal with a single word type at a time); on the other, they tended to generate softmax distributions reflecting the semantics of the surronding context. For example, in the sentence:

(a) The two *justices* have been attending federalist society events for years,

our model correctly disambiguated *justices* with the WordNet sense $\texttt{justice}_n^3$ (public official) rather than $\texttt{justice}_n^1$ (the quality of being just), and the corresponding softmax distribution was heavily biased towards words and senses related to persons or groups (*commissioners*, *defendants*, *jury*, *cabinet*, *directors*). On the other hand, in the sentence:

(b) Xavi Hernandez, the player of Barcelona, has 106 *matches*,

the same model disambiguated *matches* with the wrong WordNet sense $\texttt{match}_n^1$ (tool for starting a fire). This suggests that the signal carried by discriminative words like *player* vanishes rather quickly. In order to enforce global coherence further, recent contributions have proposed more sophisticated models where recurrent architectures are combined with Conditional Random Fields (Huang et al., 2015; Ma and Hovy, 2016). Finally, a number of errors were connected to shorter sentences with limited context for disambiguation: in fact, we noted that the average precision of our model, without MFS back-off, increased by 6.2% (from 74.6% to 80.8%) on sentences with more than 20 word tokens.

7 Conclusion

In this paper we adopted a new perspective on supervised WSD, so far typically viewed as a classification problem at the word level, and framed it using neural sequence learning. To this aim we defined, analyzed and compared experimentally different end-to-end models of varying complexities, including augmentations based on an attention mechanism and multitask learning.

Unlike previous supervised approaches, where a dedicated model needs to be trained for every content word and each disambiguation target is treated in isolation, sequence learning approaches learn a single model in one pass from the training data, and then disambiguate jointly all target words within an input text. The resulting models consistently achieved state-of-the-art (or statistically equivalent) figures in all benchmarks for all-words WSD, both fine-grained and coarse-grained, effectively demonstrating that we can overcome the so far undisputed and long-standing word-expert assumption of supervised WSD, while retaining the accuracy of supervised word experts.

Furthermore, these models are sufficiently flexible to allow them, for the first time in WSD, to be readily adapted to languages different from the one used at training time, and still achieve competitive results (as shown in Section 6.2). This crucial feature could potentially pave the way for cross-lingual supervised WSD, and overcome the shortage of sense-annotated data in multiple languages that, to date, has prevented the development of supervised models for languages other than English.

As future work, we plan to extend our evaluation to larger sense-annotated corpora (Raganato et al., 2016) as well as to different sense inventories and different languages. We also plan to exploit the flexibility of our models by integrating them into downstream applications, such as Machine Translation and Information Extraction.

Acknowledgments

The authors gratefully acknowledge the support of the ERC Consolidator Grant MOUSSE No. 726487.

[12] We refer here to the same model considered in Section 6.2 (i.e., **BLSTM+att.+LEX**).

References

Eneko Agirre and Philip Edmonds. 2007. *Word sense disambiguation: Algorithms and applications*, volume 33. Springer Science & Business Media.

Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2014. Random Walks for Knowledge-Based Word Sense Disambiguation. *Comput. Linguist.*, 40(1):57–84.

Héctor Martínez Alonso and Barbara Plank. 2017. When is Multitask Learning Effective? Semantic Sequence Prediction under Varying Data Conditions. In *Proc. of ACL*, pages 44–53.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively Multilingual Word Embeddings. *CoRR*, abs/1602.01925.

Osman Başkaya and David Jurgens. 2016. Semi-supervised Learning with Induced Word Senses for State of the Art Word Sense Disambiguation. *J. Artif. Int. Res.*, 55(1):1025–1058.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR Workshop*.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2014. An Enhanced Lesk Word Sense Disambiguation Algorithm through a Distributional Semantic Model. In *Proc. of COLING*, pages 1591–1600.

Joachim Bingel and Anders Søgaard. 2017. Identifying Beneficial Task Relations for Multi-task Learning in Deep Neural Networks. In *Proc. of EACL*, pages 164–169.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic Tagging with Deep Residual Networks. In *Proc. of COLING*, pages 3531–3541.

Andrei Butnaru, Radu Tudor Ionescu, and Florentina Hristea. 2017. ShotgunWSD: an Unsupervised Algorithm for Global Word Sense Disambiguation Inspired by DNA Sequencing. In *Proc. of EACL*, pages 916–926.

Marine Carpuat and Dekai Wu. 2007. Improving Statistical Machine Translation using Word Sense Disambiguation. In *Proc. of EMNLP*, pages 61–72.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A Unified Model for Word Sense Representation and Disambiguation. In *Proc. of EMNLP*, pages 1025–1035.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proc. of EMNLP*, pages 1724–1734.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage Sense Disambiguation and Information Extraction with a Supersense Sequence Tagger. In *Proc. of EMNLP*, pages 594–602.

Jordi Daudé, Lluís Padró, and German Rigau. 2003. Validation and Tuning of WordNet Mapping Techniques. In *Proc. of RANLP*, pages 117–123.

Oier Lopez de Lacalle and Eneko Agirre. 2015. A Methodology for Word Sense Disambiguation at 90% based on large-scale Crowd Sourcing. In *Proc. of SEM*, pages 61–70.

Claudio Delli Bovi, Luca Telesca, and Roberto Navigli. 2015. Large-Scale Information Extraction from Textual Definitions through Deep Syntactic and Semantic Analysis. *TACL*, 3:529–543.

Philip Edmonds and Scott Cotton. 2001. Senseval-2: Overview. In *Proc. of SENSEVAL*, pages 1–5.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological Inflection Generation Using Character Sequence to Sequence Learning. In *Proc. of NAACL-HLT*, pages 634–643.

Lucie Flekova and Iryna Gurevych. 2016. Supersense embeddings: A unified model for supersense interpretation, prediction, and utilization. In *Proc. of ACL*, pages 2029–2041.

Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *CoRR*, abs/1308.0850.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proc. of ACL*, pages 1631–1640.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *CoRR*, abs/1508.01991.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning Sense Embeddings for Word and Relational Similarity. In *Proc. of ACL*, pages 95–105.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for Word Sense Disambiguation: An Evaluation Study. In *Proc. of ACL*, pages 897–907.

Mikael Kågebäck and Hans Salomonsson. 2016. Word Sense Disambiguation using a Bidirectional LSTM. In *Proceedings of CogALex*, pages 51–56.

Adam Kilgarriff. 2001. English Lexical Sample Task Description. In *Proc. of SENSEVAL*, pages 17–20.

Ryan Kiros, Yukun Zhu, Ruslan R. Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proc. of NIPS*, pages 3294–3302.

Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *ICLR Workshop*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proc. of ACL*, pages 1064–1074.

Diana McCarthy, Marianna Apidianaki, and Katrin Erk. 2016. Word Sense Clustering and Clusterability. *Comput. Linguist.*, 42(2):245–275.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning Generic Context Embedding with Bidirectional LSTM. In *Proc. of CoNLL*, pages 51–61.

Rada Mihalcea, Timothy Chklovski, and Adam Kilgarriff. 2004. The Senseval-3 English Lexical Sample Task. In *Proc. of SENSEVAL*, pages 25–28.

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

George A. Miller, R.T. Beckwith, Christiane D. Fellbaum, D. Gross, and K. Miller. 1990. WordNet: an online lexical database. *International Journal of Lexicography*, 3(4):235–244.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proc. of HLT*, pages 303–308.

Andrea Moro and Roberto Navigli. 2015. SemEval-2015 Task 13: Multilingual All-Words Sense Disambiguation and Entity Linking. In *Proc. of SemEval*, pages 288–297.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *TACL*, 2:231–244.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Roi Reichart, Ira Leviant, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic Specialisation of Distributional Word Vector Spaces using Monolingual and Cross-Lingual Constraints. *TACL*, 5.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2):10.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. SemEval-2013 Task 12: Multilingual Word Sense Disambiguation. In *Proc. of SemEval*, volume 2, pages 222–231.

Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves. 2007. SemEval-2007 Task 07: Coarse-grained English All-words Task. In *Proc. of SemEval*, pages 30–35.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. *Artificial Intelligence*, 193:217–250.

Steven Neale, Lus Gomes, Eneko Agirre, Oier Lopez de Lacalle, and Antnio Branco. 2016. Word Sense-Aware Machine Translation: Including Senses as Contextual Features for Improved Translation Models. In *Proc. of LREC*, pages 2777–2783.

Tommaso Pasini and Roberto Navigli. 2017. Train-O-Matic: Large-Scale Supervised Word Sense Disambiguation in Multiple Languages without Manual Training Data. In *Proc. of EMNLP*.

Mohammad Taher Pilehvar and Roberto Navigli. 2014. A Large-scale Pseudoword-based Evaluation Framework for State-of-the-art Word Sense Disambiguation. *Comput. Linguist.*, 40(4):837–881.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss. In *Proc. of ACL*, pages 412–418.

Sameer S. Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. SemEval-2007 Task 17: English Lexical Sample, SRL and All Words. In *Proc. of SemEval-2007*, pages 87–92.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2016. Automatic Construction and Evaluation of a Large Semantically Enriched Wikipedia. In *Proc. of IJCAI*, pages 2894–2900.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison. In *Proc. of EACL*, pages 99–110.

Sascha Rothe and Hinrich Schütze. 2015. AutoExtend: Extending Word Embeddings to Embeddings for Synsets and Lexemes. In *Proc. of ACL*, pages 1793–1803.

Hui Shen, Razvan Bunescu, and Rada Mihalcea. 2013. Coarse to Fine Grained Sense Disambiguation in Wikipedia. In *Proc. of SEM*, pages 22–31.

Benjamin Snyder and Martha Palmer. 2004. The english all-words task. In *Proc. of Senseval-3*, pages 41–43.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proc. of ACL*, pages 231–235.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proc. of NIPS*, pages 3104–3112.

Kaveh Taghipour and Hwee Tou Ng. 2015a. One Million Sense-Tagged Instances for Word Sense Disambiguation and Induction. In *Proc. of CoNLL*, pages 338–344.

Kaveh Taghipour and Hwee Tou Ng. 2015b. Semi-Supervised Word Sense Disambiguation Using Word Embeddings in General and Specific Domains. In *Proc. of NAACL-HLT*, pages 314–323.

Rocco Tripodi and Marcello Pelillo. 2017. A Game-Theoretic Approach to Word Sense Disambiguation. *Comput. Linguist.*, 43(1):31–70.

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proc. of NIPS*, pages 2773–2781.

Oriol Vinyals and Quoc V. Le. 2015. A Neural Conversational Model. In *Proc. of ICML, JMLR: W&CP*, volume 37.

Wenhui Wang and Baobao Chang. 2016. Graph-based Dependency Parsing with Bidirectional LSTM. In *Proc. of ACL*, pages 2306–2315.

Dirk Weissenborn, Leonhard Hennig, Feiyu Xu, and Hans Uszkoreit. 2015. Multi-Objective Optimization for the Joint Disambiguation of Nouns and Named Entities. In *Proc. of ACL*, pages 596–605.

Deyi Xiong and Min Zhang. 2014. A Sense-Based Translation Model for Statistical Machine Translation. In *Proc. of ACL*, pages 1459–1469.

Dayu Yuan, Ryan Doherty, Julian Richardson, Colin Evans, and Eric Altendorf. 2016. Semi-supervised Word Sense Disambiguation with Neural Models. In *Proc. of COLING*, pages 1374–1385.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701.

Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense: A Wide-Coverage Word Sense Disambiguation System for Free Text. In *Proc. of ACL: System Demonstrations*, pages 78–83.

Zhi Zhong and Hwee Tou Ng. 2012. Word Sense Disambiguation Improves Information Retrieval. In *Proc. of ACL*, pages 273–282.

Jie Zhou and Wei Xu. 2015. End-to-End Learning of Semantic Role Labeling using Recurrent Neural Networks. In *Proc. of ACL*, pages 1127–1137.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. In *Proc. of ACL*, pages 207–212.

Learning Word Relatedness over Time

Guy D. Rosin[1], **Eytan Adar**[2], **Kira Radinsky**[1,3]
[1]Technion – Israel Institute of Technology, Haifa, Israel
[2]University of Michigan, Ann Arbor, USA
[3]eBay Research, Israel
{guyrosin,kirar}@cs.technion.ac.il, eadar@umich.edu

Abstract

Search systems are often focused on providing relevant results for the "now", assuming both corpora and user needs that focus on the present. However, many corpora today reflect significant longitudinal collections ranging from 20 years of the Web to hundreds of years of digitized newspapers and books. Understanding the *temporal intent* of the user and retrieving the most relevant historical content has become a significant challenge. Common search features, such as query expansion, leverage the relationship between terms but cannot function well across all times when relationships vary temporally. In this work, we introduce a temporal relationship model that is extracted from longitudinal data collections. The model supports the task of identifying, given two words, *when* they relate to each other. We present an algorithmic framework for this task and show its application for the task of query expansion, achieving high gain.

1 Introduction

The focus of large-scale Web search engines is largely on providing the best access to present snapshots of text – what we call the "Now Web". The system constraints and motivating use cases of traditional information retrieval (IR) systems, coupled with the relatively short history of the Web, has meant that little attention has been paid to how search engines will function when search must scale not only to the number of documents but also temporally. Most IR systems assume fixed language models and lexicons. They focus only on the leading edge of query behavior (i.e., what does the user likely mean today when they type "Jaguar"). In this context, features as basic as disambiguation and spelling corrections are fixed to what is most likely today or within the past few years (Radinsky et al., 2013), query expansions and synonyms are weighted towards current information (Shokouhi and Radinsky, 2012), and results tend to include the most recent and popular content. While this problem would seem to be speculative in that it will be years until we need to address it, the reality is the rate of change (Adar et al., 2009) of the Web, language, and culture have simply compressed the time in which critical changes happen.

The "Now Web" assumptions are entirely reasonable for temporally coherent text collections and allow users (and search engines) to ignore the complexity of changing language and concentrate on a narrower (though by no means simpler) set of issues. The reality is that this serves a significant user population effectively. There are nonetheless a growing number of both corpora and users who require access not just to what is relevant at a particular instant (e.g., Hathitrust (Willis and Efron, 2013), historical news corpora, the Internet Archives, and even fast changing Twitter feeds). Within such contexts, a search engine will need to vary the way it functions (e.g., disambiguation) and interacts (e.g., suggested query expansions) depending on the period and temporal scale of documents being queried. This, of course, is further complicated by the fact that Web pages are constantly evolving and replaced.

Take for example the query "Prime Minister Ariel Sharon". When fed into a news archive search engine, the likely intent was finding results about *Sharon's* role as Israel's *Prime Minister*, a role held from *2001 to 2006*. Singh et al. (2016) refer to this as a *Historical Query Intent*. However, most popular search engines return results about Sharon's death in 2011, when he was no longer

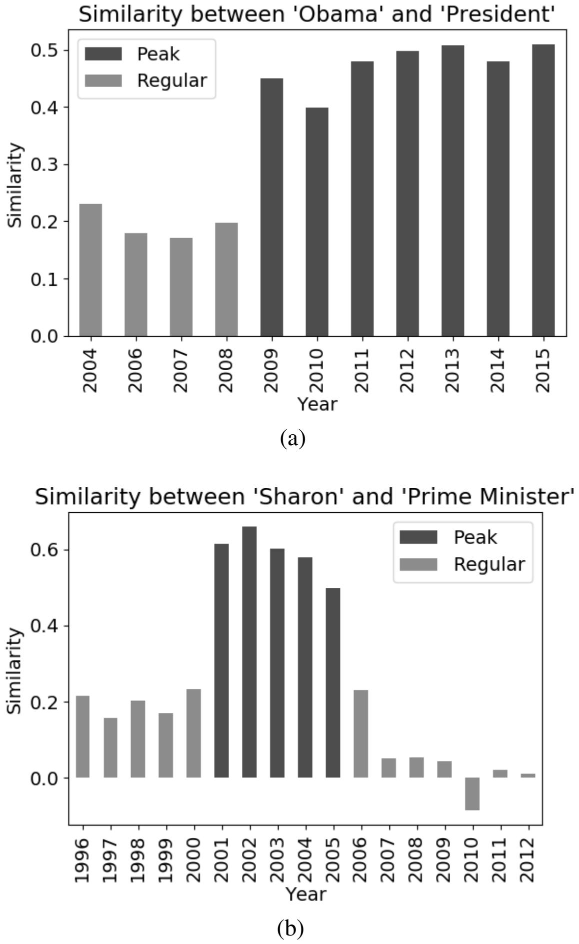
Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1168–1178
Copenhagen, Denmark, September 7–11, 2017.

prime minister. The searcher may take the additional step of filtering results to a time period, but this requires knowing what that period should be. Other query features are also unresponsive to temporal context. For example, the top query suggestions for this query focus on more recent events of his death: "Former prime minister Sharon dies at 85", "Former prime minister Sharon's condition worsens", etc. While these *might* satisfy the searcher if they are looking for the latest results, or the results most covered by the press, there are clearly other possible needs (Bingham, 2010).

In this paper, we focus on the task of measuring word relatedness over time. Specifically, we infer whether two words (tokens) relate to each other during a certain time range. This task is an essential building block of many temporal applications and we specifically target *time-sensitive query expansion (QE)*. Our focus is on *semantic relatedness* rather than *semantic similarity*. Relatedness assumes many different kinds of specific relations (e.g. meronymy, antonymy, functional association) and is often more useful for computational linguistics applications than the more narrow notion of similarity (Budanitsky and Hirst, 2006).

We present several temporal word-relatedness algorithms. Our method utilizes a large scale temporal corpus spanning over 150 years (The New York Times archive) to generate *temporal deep word embeddings*. We describe several algorithms to measure word relatedness over time using these temporal embeddings. Figure 1a presents the performance of one of those algorithms on the words "Obama" and "President". Note that the highest relatedness score for the words appears during the presidential term of Barack Obama. Similarly, Figure 1b shows a high score for "Ariel Sharon" and "prime minister" only during his term.

Using the approach above, we present a specific application – producing temporally appropriate query-expansions. For example, consider the query: "Trump Businessman". Figure 2 shows the non-temporal query expansion suggestions which focus heavily on the first entity (i.e., "Trump") and his current "state" (i.e., a focus on Donald Trump as President, rather than presenting suggestions about Trump's business activity as implied by the query). We present an empirical analysis presenting the strengths and weaknesses of the different temporal query-expansion algorithms and comparing them to current word-embeddings-based QE algorithms (Kuzi et al., 2016; Diaz et al., 2016).



(a)

(b)

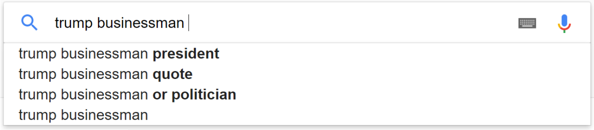
Figure 1: Similarity identified by our algorithms between words over time. Dark gray indicates high similarity whereas light gray indicates non-significant similarity.

In this paper we describe a novel problem of evaluating word relatedness over time and contribute our datasets to evaluate this task to the community[1]. Second, we present novel representations and algorithms for evaluating this task and show high performance. We share our code with the community as well. Finally, we present the application of this task to query-expansion and present several methods built on top of the temporal relatedness algorithms that show high performance for QE.

2 Related Work

Understanding the semantic change of words has become an active research topic (Section 2.1). Most work has focused on identifying semantic

[1] `https://github.com/guyrosin/learning-word-relatedness`



trump businessman

trump businessman **president**
trump businessman **quote**
trump businessman **or politician**
trump businessman

Figure 2: Query expansions for the terms "Trump Businessman". Most results are referring to his term as president and not to his business activity.

drifts and word meaning changes. A lot of effort has been made into analyzing texts temporally: several methods for temporal information extraction were recently proposed (Ling and Weld, 2010; Kuzey and Weikum, 2012; Talukdar et al., 2012), as well as publicly released knowledge bases, such as YAGO2 (Hoffart et al., 2013). These methods automatically extract temporal relational facts from free text or semi-structured data. In addition, Pustejovsky et al. (2003); UzZaman et al. (2012) and others annotated texts temporally, and extracted events as well as temporal expressions.

Work in Information Retrieval (IR, Section 2.2) has discussed the concept of 'time' as a contextual parameter for understanding user intent. Largely, this research utilizes query-log analysis with the time *of the query* as a context signal. In this work, we leverage the temporal variation in word relatedness to understand, and better accommodate, intent.

2.1 Word Dynamics

Continuous word embeddings (Mikolov et al., 2013) have been shown to effectively encapsulate relatedness between words. Radinsky et al. (2011) used temporal patterns of words from a large corpus for the task of word similarity. They showed that words that co-occur in history have a stronger relation. In our work, we focus on identifying *when* a relation holds. Numerous projects have studied the change of word meanings over time, and specifically focused on identification of the change itself. Sagi et al. (2009) used Latent Semantic Analysis for detecting changes in word meaning. Wijaya and Yeniterzi (2011) characterized 20 clusters to describe the nature of meaning change over time, whereas Mitra et al. (2014) used other clustering techniques to find changes in word senses. Mihalcea and Nastase (2012) identified changes in word usage over time by the change in their related part-of-speech. Others have investigated the use of word frequency to identify epochs (Popescu and Strapparava, 2013).

Jatowt and Duh (2014) represented a word embedding over the Google Books corpus (granularity of decades) and presented qualitative evaluation for several words. Hamilton et al. (2016) built Word2Vec embedding models on the Google Books corpus to detect known word shifts over 30 words and presented a dozen new shifts from the data. The authors presented two laws that govern the change of words – frequent words change more slowly and polysemous words change more quickly. Finally, Kenter et al. (2015) studied changes in meaning (represented by a few seed words), and monitored the changing set of words that are used to denote it.

In our work, we focus on learning relatedness of words over time. We evaluate the technique using a large scale analysis showing its prediction accuracy. Moreover, we define the task of identifying the temporality of relatedness between two words. We show that understanding the temporal behavior of entities improves performance on IR tasks.

2.2 Temporal Search

The temporal aspects of queries and ranking gained significant attention in IR literature. Some have focused on characterizing query behavior over time. For example, different queries change in popularity over time (Wang et al., 2003) and even by time of day (Beitzel et al., 2004). Jones and Diaz (2007) described queries to have three temporarilty patterns: atemporal, temporally unambiguous and temporally ambiguous. Others have leveraged temporal variance to look for indicators of query intent (Kulkarni et al., 2011). Several efforts (Shimshoni et al., 2009; Chien and Immorlica, 2005; M. Vlachos and Gunopulos, 2004; Zhao et al., 2006; Shokouhi, 2011; Radinsky et al., 2012) were done to not only characterize the temporal query behavior but also model it via time-series analysis. Radinsky and colleagues modeled changes in the frequency of clicked URLs, queries, and clicked query-URL pairs by using time-series analysis and show its application for improving ranking and query auto-suggestions (Radinsky et al., 2013; Shokouhi and Radinsky, 2012). Singh et al. (2016) focused on serving the specific needs of historians, and introduced the notion of a *Historical Query Intent* for this purpose.

Whereas prior work mainly focuses on query-log analysis and understanding the user's intent based on the time the query was issued or the document was changed, our work focuses on understanding the subtle changes of language over time that indicate a temporal intent. We show that understanding the temporal relatedness between words is a building block needed for understanding better query intent. Given two words or entities we identify when their relatedness was the strongest to help produce better query expansions.

3 Temporal Relatedness Dynamics and Semantics

To address the task of understanding temporal relatedness, our approach consists of three main steps: (1) Represent relatedness using word embeddings over time (Section 3.1). (2) Model relatedness *change* over time as a time series (Section 3.2). (3) Combine these to identify when relatedness relations hold temporally (Section 3.3).

3.1 Representing Relatedness using Word Embeddings

In our work we leverage the distributed representation framework of Word2Vec (Mikolov et al., 2013) (specifically skip-grams). Intuitively, given a word w_t, skip-grams attempt to predict surrounding words, e.g. $w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$ for a window size of $c = 2$.

Definition 3.1. Let C_i be the word context for a word w_i. In this work, we consider the context to be the surrounding words of a window size of n $C_i = \{w_{i-n}, w_{i-1}, w_{i+1}, w_{i+n}\}$. We define C_i^t to be the context for a word w_i in time period t, i.e. only in the documents written during time t.

Definition 3.2. We define the word context of a **specific embedding** of a year y for a word w_i, to be $\{w_c \in C_i^y\}$. Intuitively, a specific embedding represents the embedding of a word in a certain year. We denote the vector representation of a word w_i in a year y by v_i^y.

Definition 3.3. We define the word context of a **global embedding** for a word w_i to be C_i. Intuitively, a global embedding represents the embedding of a word over all time periods. We denote the vector representation of a word w_i by v_i.

Using Word2Vec, we approximate the semantic relatedness between two entities by the cosine similarity between their embeddings (Turney et al., 2010).

Definition 3.4. A **temporal relation** (e_1, e_2, t), where e_i are entities and t is a referenced time period, is said to be true if e_1, e_2 relate during t, i.e. their semantic relatedness during that time is relatively high. For example: *(Christopher Nolan, The Dark Knight, 2008)* is true, due to the fact that the movie, which was released in 2008, was directed by Nolan.

Definition 3.5. The **dynamics** of two entities e_1, e_2 is defined to be the time series of the semantic relatedness of e_1 and e_2:

$$Dynamics(e_1, e_2) = \left\langle cos(v_1^{t_1}, v_2^{t_1}), \dots, cos(v_1^{t_n}, v_2^{t_n}) \right\rangle \quad (1)$$

where $t_1, \dots, t_n$ are all the time periods.

We model entities' relatedness change over time by constructing their *dynamics*. Recall Figure 1a, which shows the semantic distance between the vector representations of *Barack Obama* and *President* over the years. Semantic distance is an accurate indicator of the time period when Obama was president. Therefore, our goal is to detect the time periods of high relatedness using peak detection.

3.2 Understanding Relatedness Dynamics

When considering a relationship between entities, detecting time periods of high relatedness enables us to reveal and identify what we call "periods of interest" – time periods in which the entities were the closest. In this section, we present an algorithm to identify these "periods of interest". Intuitively, when considering the *dynamics* of two entities, its peaks represent the lowest distances over time. More formally, given two entities e_1, e_2 we want to construct their *dynamics* and find peaks in it, i.e. the following sequence of time periods: $\left\{t_i \mid cos(v_1^{t_i}, v_2^{t_i}) \text{ is relatively high, and } t_i < t_j \text{ if } i < j\right\}$.

Traditional peak detection algorithms focus on detecting peaks that are sharp and isolated (i.e. not too many surrounding points have similar values) (Palshikar et al., 2009). In our case, relations often imply continuous periods of peaks, e.g. Obama was president for eight years. Therefore, we need to detect peaks, as well as periods of continuous peaks (i.e., steps).

Let $L = (t_1, v_1), (t_2, v_2), \dots, (t_n.v_n)$ be a list of tuples, where t_i are time periods and v_i are values. Given L, the algorithm returns a list of peak periods, i.e. $\{t \in L \mid t \text{ contains a peak}\}$.

The first step is to find relative maxima: we scan L and look for pairs (t_i, v_i) such that $v_{i-1} < v_i > v_{i+1}$, i.e. v_i is a local maximum. We apply two minimum thresholds in order to filter insignificant points: An absolute threshold, which below it we consider the peak not to be significant enough, and a relative threshold which facilitates removing points that are much lower than the highest maximum. The final step of the algorithm is to find plateaus which will also be considered part of the peak. For each peak point, the algorithm considers the point's surrounding neighbors. Using a threshold relative to the current peak, those identified as close in their value to the current peak are added to the peak list.

3.3 Learning Temporal Relatedness

We define the task of learning temporal relatedness: given two entities e_1, e_2, identify whether they relate to each other during a certain year y, i.e., whether the temporal relation (e_1, e_2, y) is true. For example, in Figure 1a, Obama was related to President in 2010 but not in 2005.

3.3.1 Specific Classifier

The first method we present to classify word relatedness, employs a classifier that receives as inputs v_i corresponding to the entities e_i and an additional feature of the year. In our evaluation (Section 4) we will use about 40 years of data, i.e. the year feature will have 40 possible values. The classifier will predict, given two entities, whether they relate to each other during a referenced year.

Let $Cl \colon \mathbb{R}^n \to \{0, 1\}$ be a classifier mapping a vector of n features $F = (f_1, \ldots, f_n)$ to a label $L \in \{0, 1\}$. Let our feature vector be of the form:

$$F = (v_1^y \| v_2^y \| y) \tag{2}$$

where v_1^y is the first entity's specific embedding (i.e. at time y), v_2^y is the second entity's specific embedding and y is the year. As a preliminary step, we need to train Cl on a diverse dataset of positive and negative examples, i.e. true and false temporal relations. For this purpose, we utilize our temporal relations dataset (Section 3.4.2). From each relation, we extract a temporal relation (e_1, e_2, y), calculate its feature vector and use it for training.

Given a new temporal relation, we apply Cl to predict whether it is true, i.e. whether its entities relate during the referenced time.

3.3.2 Temporal Classifier

Here we use a classifier similar to the one described in Section 3.3.1 (training is performed the same way), combined with input from the entities *dynamics*. First, we build the *dynamics*, i.e. build specific word embeddings of e_1 and e_2 for every year y, and calculate their cosine similarity $cos(v_1^y, v_2^y)$. We then apply our peak detection algorithm (Section 3.2) on the *dynamics* and use its output as one of the classifier's features (denoted by *isPeak*). As a result, our feature vector is:

$$F = (v_1^y \| v_2^y \| y \| isPeak) \tag{3}$$

3.4 Leveraging World Knowledge

We apply our techniques to two corpora: the first is a temporal corpora (Section 3.4.1), which we used for creating word embeddings, and the second is a relational corpora (Section 3.4.2), which we used for training our models and for evaluation.

3.4.1 Temporal Corpora

For constructing the corpora, we used The New York Times archive[2], with articles from 1981 to 2016 (9GB of text in total). Specific word embeddings were generated for every time period (i.e., year) using Word2Vec. Each year's data was used to create word embeddings using Word2Vec's skip-gram with negative sampling approach (Mikolov et al., 2013), with the Gensim library (Řehůřek and Sojka, 2010). We trained the models with the following parameters: window size of 5, learning rate of 0.05 and dimensionality of 140. We filtered out words with less than 30 occurrences during that year.

We observe both *ambiguity* (Apple, the company and the fruit) and *variability* (different phrases referring to the same entity, e.g., *President Obama*, *Barack H. Obama*, *Obama*). While such 'noise' may be problematic, both the scale of the data and stylistic standards of The New York Times help. Additionally, we ensure a connection to an entity database (Wikipedia) and perform additional cleaning methods (lemmatization, removing stopwords).

3.4.2 Relational Corpora

In this work, we use YAGO2 (Hoffart et al., 2013) as our relational corpora due to its temporal focus. The YAGO2 knowledge base contains millions of facts about entities, automatically extracted from

[2] http://spiderbites.nytimes.com/

Relation Type	Count	%
Directed	37713	47.42
HoldsPoliticalPosition	2765	3.48
IsMarriedTo	2210	2.78
PlaysFor	4376	5.50
Produced	23567	29.63
HappenedIn	8899	11.19
Total	79530	100

Table 1: Relations Dataset Composition

Wikipedia and other sources. We use relations from YAGO2 to build our own dataset of temporal relations, which we use in all of our algorithms and evaluation – as a source for temporal relations.

The dataset consists of temporal relations in the following format: *(entity*$_1$*, entity*$_2$*, year, type, class)*, where $entity_1$ and $entity_2$ are entities, $type$ is a relation type, and $class$ is true if the relation holds on $year$. For example, *(Tim Burton, Batman Returns, 1992, Directed, true)* and *(Battle Mogadishu, Somalia, 2010, HappenedIn, true)*. Table 1 shows the exact dataset composition. We built our dataset on all the relation types that have a temporal dimension in YAGO2: *Directed, HoldsPoliticalPosition, IsMarriedTo, Produced, PlaysFor, HappenedIn*. The dataset contains 80K of such relations.

4 Evaluation

4.1 Experimental Methodology

We compare the methods described in Section 3.3, where for Cl we chose to use a Support Vector Machine (SVM)[3], with an RBF kernel and C=1.0 (chosen empirically). Two baselines were used for comparison. The first is the common non-temporal model, i.e. a classifier that uses the global (all-time) word embeddings and the following features: the two entities' global embeddings, and a year. More formally,

$$F = (v_1||v_2||y) \tag{4}$$

Given a new temporal relation, the classifier predicts whether it is true during the referenced year, and we output the classifier's prediction. The second baseline we compare against is a standard text classifier that uses the global word embeddings as its only features, i.e. $F = (v_1||v_2)$.

The dataset on which we perform the evaluation is described in Section 4.2. The dataset is not balanced: it contains more negative examples than positive ones. Therefore, for evaluating the methods that involve a classifier we use stratified 10-fold cross validation. We remove relations from consideration if there is insufficient data in the corpora for that year (i.e., one of the entities was filtered out due to low incidence).

4.2 Dataset Construction

Recall that our relational corpora consists of 80K temporal relations in the following format: *(entity*$_1$*, entity*$_2$*, year, type, class)*, where $type$ is a relation type, and $class$ is true if the relation holds on $year$.

For training and evaluating our classifiers we need negative examples as well as positive examples. We generate negative examples in the following way: for every relation in the corpora, we randomly sample 10 negative examples. We exclude the years of the true examples from the dataset's year range, and then randomly choose years for the negative examples. To illustrate, let us observe the case of *Obama, President*: Obama was president from 2009-2016, so we sample negative examples from 1981 to 2008, such as *(Obama, President, 1990, HoldsPoliticalPosition, false)*. The resulting dataset contains 420K relations. We refer to it as the *Temporal Relations Dataset*[4].

4.3 Main Results

Table 2 presents the results of our experiments.

Baselines: The Global and Global+Year baselines produced an AUC of 0.55 and 0.57, respectively. They both performed much worse compared to our methods, with F1 of 0.13.

Specific Classifier produced an AUC of 0.72. It has the highest recall score of all methods (0.88), but its other scores are relatively low.

Temporal Classifier produced an AUC of 0.83. As reported in Table 2, it performed significantly better compared to all other methods, with $p < 0.05$. We applied the Wilcoxon signed-rank test to calculate statistical significance.

[3] We used the implementation by the scikit-learn library (Pedregosa et al., 2011).

[4] `https://github.com/guyrosin/learning-word-relatedness`

Algorithm	Acc.	Rec.	Pr.	F1	AUC
Global	0.67	0.26	0.08	0.13	0.57
Global+Year	0.52	0.39	0.08	0.13	0.55
Specific	0.52	**0.91**	0.32	0.47	0.73
Temporal	**0.81**	0.67	**0.58**	**0.62**	**0.84**

Table 2: Relatedness Learning Evaluation Results (Accuracy, Recall, Precision, F1, AUC)

4.4 Performance Analysis

We tuned Word2Vec parameters by empirically testing on a random subset of our dataset: we set the vector size to be 140 and used a minimum threshold of 30 occurrences (per year). We found that this balanced the removal of noisy data while ensuring that key entities were retained. For constructing the Word2Vec models, the amount of data is crucial or it may lead to unreliable (Hellrich and Hahn, 2016) or inaccurate results. We saw a clear correlation between accuracy and number of occurrences of a participating word. That drove our decision to evaluate our algorithms only on New York Times articles from 1981 onwards – where the number of articles per year is sufficiently large.

5 Task Example: Query Expansion

Temporal relatedness learning can be used for various NLP and IR-related tasks. For example, it is a common practice in IR to expand user queries to improve retrieval performance (Carpineto and Romano, 2012). Our technique allows us to produce temporally appropriate expansions. Specifically, given a query of n entities $Q = \{e_1, e_2, \ldots, e_n\}$, our task is to expand Q with additional search terms to add to it to improve retrieval of relevant documents.

For example, consider the query "Trump Businessman" (Figure 2). Current QE methods, which do not have a temporal aspect, focus on Donald Trump as President of the United States, a potentially erroneous result depending on the temporal focus of the searcher. A reasonable temporal expansion, might contain terms that relate to Donald Trump's business activity, such as "billionaire" or "real estate". Using our technique, the temporal focus of a query can be identified and appropriate expansions offered to the end-user. Specifically, we can analyze the relationship between the query entities to identify the "most relevant" time period – when those entities were strongly connected. Intuitively, the QE algorithms will identify the most relevant time period t for the query entities, and find semantically related terms from that time, to expand the query with.

To tackle this task, we use the algorithms described in Section 3.3. Several different algorithms can utilize the temporal relation models for the task of query expansion. Let us introduce the following definitions for our QE algorithms:

Definition 5.1. Let $NN_K^t(e)$ be the set of K terms that are the closest to an entity e in time t.

Definition 5.2. Let $NN_K(e)$ be the set of "globally" (all-time) closest terms to an entity e .

Definition 5.3. Mutual closeness between an entity x and a query Q is defined by the sum of cosine similarities between x and every $e \in Q$, i.e.

$$M_{cos}(x, Q) := \sum_{e \in Q} cos(x, e) \tag{5}$$

5.1 Query Expansion Algorithms

We describe alternative strategies to provide temporal query expansion ranging from a generic baseline to algorithms that leverage our embedding and classifiers. As a running example, we use the query: "Steven Spielberg, Saving Private Ryan" (Spielberg directed the movie in 1998). 'Reasonable' (temporally relevant) expansions for this query might include: actors who played in this movie, other Spielberg films or similar films from the same time and genre, etc.

5.1.1 Baseline

Following the results of Roy et al. (2016), we consider a baseline method that expands each entity separately, based on global Word2Vec similarity. We define the set of candidate expansion terms as

$$C = \bigcup_{e \in Q} NN_K(e) \tag{6}$$

i.e., for each entity, we choose the closest K global terms. For each $c \in C$, we compute the mutual closeness $M_{cos}(c, Q)$ and sort the terms in C on the basis of this value. The top K candidates are selected as the actual expansion terms. The baseline (poorly) expands the query "Steven Spielberg, Saving Private Ryan" with "Inglourious [Basterds], George Lucas". In some sense, one can see the relation – both are war movies, and

Lucas and Spielberg have worked together. However, Inglourious Basterds was created at 2009 and was directed by Quentin Tarantino.

5.1.2 Globally-Based Classifier

We use a heuristic and assume that the most relevant time period t for the entities of the query is the time when the entities were the closest. We use the classifier from our baseline method in Section 4.1, whose goal is to estimate how relevant a year is to a given set of n entities. Its features are the global word embeddings, as well as a year: $F = (v_1 \| v_2 \| \ldots \| v_n \| y)$.

We apply the classifier to every year y, and choose the one with the highest returned probability of the true label as the most relevant time t.

$$t = \arg\max_{y} \{Cl(v_1, v_2, \ldots, v_n, y)\} \tag{7}$$

We take as candidate-expansion terms the K closest terms to each entity from that year, separately:

$$C = \bigcup_{e \in Q} NN_K^t(e) \tag{8}$$

C is then filtered as described in the baseline.

To train the classifier, for each temporal relation in our temporal relations dataset, we calculate its feature vector and use it for training. Considering our example, this method wrongly chooses $t = 2004$. In that year the entity *Saving Private Ryan* does not exist, so we end up with a wrong expansion of “Francis Ford Coppola film”.

5.1.3 Temporal Classifier

As we have seen in the previous subsection, the globally-based classifier is limited in cases where time-specific knowledge might yield better results. Thus, in this method we use the specific classifier from Section 3.3.1. Its features are the entities’ specific embeddings, and a year: $F = (v_1^y \| v_2^y \| \ldots \| v_n^y \| y)$. We then continue as described in the previous method (find t, choose candidate terms and filter).

For our example, this method chooses correctly $t = 1998$, which is exactly the year of *Saving Private Ryan* release. Its expansion is “Tom Hanks, Movie”. Since Tom Hanks had a lead role in the movie, the expansion is reasonable. The next algorithm produces the same expansion as well.

5.1.4 Temporal Model Classifier

This method uses the temporal classifier from Section 3.3.2. Its feature vector is: $F = (v_1^y \| v_2^y \| \ldots \| v_n^y \| y \| isPeak)$. The rest is the same as described in Section 5.1.3.

5.2 Query Expansion Evaluation

Dataset. To evaluate temporal query expansion we use our temporal relations dataset, which will be made publicly available (described in Section 3.4.2). First, we evaluate on queries consisting of two entities ($n = 2$): for each relation, we create a distinct query that consists of its two entities concatenated. We search The New York Times corpus with this query[5]. We compare search performance when applying the various QE methods described in Section 5.1. To evaluate the methods that involve a classifier, we use stratified 10-fold cross validation, as the previous task was evaluated (Section 4.1). We use $K = 2$ for all methods, i.e. we generate two expansion terms per query.

In addition, we evaluate on queries consisting of three entities ($n = 3$): we created a new dataset, which contains triplets of entities instead of pairs, by merging every two related (true) relations from our relations dataset. Two relations are considered related if they share an entity, and their time periods overlap. We then generate negative relations as described in Section 4.2. In this new dataset, each temporal relation consists of three entities, a year and a binary classification.

Evaluation Metrics. Though a complete evaluation of QE is beyond the scope of this paper we describe here an evaluation suited for the temporal case. It should be noted that the technique we propose here would likely be used alongside established QE techniques (e.g., log mining).

First, when providing query expansions and suggestions we would like for them to not only retrieve relevant content, but *temporally*-relevant content. To test the latter we say that given a temporal relation, a retrieved article is considered “true” if it were published within the referenced time, and “false” otherwise. Additional manual validation was done to evaluate its relevance to the query. This metric, while not the most accurate one, allows us to distinguish between results from the most relevant time period and others. Precision of the top 10 retrieved documents (P@10) is used to evaluate the retrieval effectiveness.

Results. The results of the QE evaluation are reported in Table 3. We observe a consistent be-

[5] the archive was indexed using Solr (`https://lucene.apache.org/solr/`)

Method	P@10	
	$n = 2$	$n = 3$
Baseline (Roy et al., 2016)	14.0%	17.7%
Globally-Based Classifier	18.0%	22.7%
Temporal Classifier	27.1%	38.5%
Temporal Model Classifier	**29.4%**	**39%**

Table 3: Results of QE Algorithms Evaluation

havior for different query sizes ($n = 2, 3$). For our temporal classifiers, for $n = 3$ there is a 30% increase in precision, compared to $n = 2$.

All of our methods performed significantly better compared to the baseline (statistical significance testing has been performed using paired t-test with $p < 0.05$). This establishes our claim that utilizing temporal knowledge yields more temporal–promising results. The *Temporal Model Classifier* showed the best performance of all. This, too, suits our claim and fits to the results from the previous task (Section 4.3).

5.3 Textual Relevance

To validate results, we compared our query expansion algorithms' performance on different relation types and found big differences. On the relations *HoldsPoliticalPosition, HappenedIn* and *IsMarriedTo*, the temporal algorithms achieved around 50% accuracy, while on *Directed* and *Produced* they got only 20%. This difference is reasonable, as our models were built upon a news corpus.

Let us observe an example of using the QE algorithms with the query “Vicente Fox President” (Fox was president of Mexico from 2000 to 2006). The baseline expands with Mexico's two previous presidents (Zedillo and Salinas). This makes sense as the baseline doesn't take time into account. The globally-based classifier expands with Roh Moo-hyun, who was president of Korea during the same time period. *Temporal Classifier* expands with “Ricardo Lagos, National Action Party” (Lagos was president of Chile during that time. The latter is Fox's political party). The *Temporal Model Classifier* expands with ‘presidential’ and Francisco Labastida (the candidate who lost the elections to Fox).

Figure 3 shows the similarity between Apple and its top products since 1990. We can infer which products were the most significant at each time. Take as example the query “Apple Steve Jobs”. Using our technique, we can find the most relevant time period for this query, which is from Apple's foundation in 1976 until Jobs' death in 2011. Leveraging Figure 3, we can expand this query focusing on the most popular products of that time.

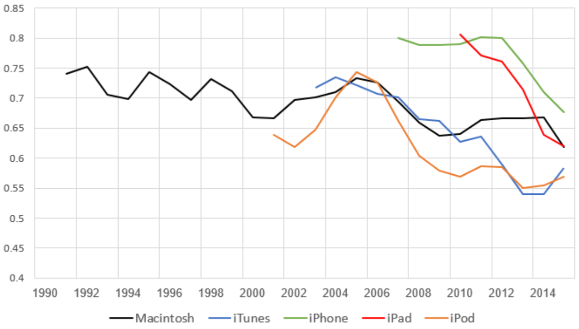


Figure 3: Similarity between Apple and its top products over time. The y-axis is cosine similarity.

6 Conclusions

We believe that as corpora evolve to include temporally-varying datasets, new techniques must be devised to support traditional and new IR methods. In this paper, we introduced a novel technique for extracting relations in temporal datasets. The technique is efficient at large scales and works in an unsupervised manner. Our experiments demonstrate the viability of the extraction technique as well as describing ways that it can be used in downstream applications. We specifically demonstrate a number of query expansion algorithms that can benefit from this technique.

References

Eytan Adar, Jaime Teevan, Susan T. Dumais, and Jonathan L. Elsas. 2009. The web changes everything: understanding the dynamics of web content. In *WSDM'09*.

S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. 2004. Hourly analysis of a very large topically categorized web query log. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*.

Adrian Bingham. 2010. the digitization of newspaper archives: Opportunities and challenges for historians. *Twentieth Century British History*, 21(2):225–231.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

Claudio Carpineto and Giovanni Romano. 2012. A survey of automatic query expansion in information retrieval. *ACM Comput. Surv.*, 44(1).

S. Chien and N. Immorlica. 2005. Semantic similarity between search engine queries using temporal correlation. In *WWW*, pages 2–11.

Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query expansion with locally-trained word embeddings. In *ACL'16*.

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *ACL'16*.

Johannes Hellrich and Udo Hahn. 2016. Bad company - neighborhoods in neural embedding spaces considered harmful. In *COLING*.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.

Adam Jatowt and Kevin Duh. 2014. A framework for analyzing semantic change of words across time. In *JCDL '14*.

Rosie Jones and Fernando Diaz. 2007. Temporal profiles of queries. *ACM Transactions on Information Systems*, 25.

Tom Kenter, Melvin Wevers, Pim Huijnen, and Maarten de Rijke. 2015. Ad hoc monitoring of vocabulary shifts over time. In *CIKM '15*.

Anagha Kulkarni, Jaime Teevan, Krysta M. Svore, and Susan T. Dumais. 2011. Understanding temporal query dynamics.

Erdal Kuzey and Gerhard Weikum. 2012. Extraction of temporal facts and events from wikipedia. In *2nd Temporal Web Analytics Workshop*, pages 25–32. ACM.

Saar Kuzi, Anna Shtok, and Oren Kurland. 2016. Query expansion using word embeddings. In *CIKM'16*.

Xiao Ling and Daniel S Weld. 2010. Temporal information extraction. In *AAAI*, volume 10, pages 1385–1390.

Z. Vagena M. Vlachos, C. Meek and D. Gunopulos. 2004. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD*.

Rada Mihalcea and Vivi Nastase. 2012. Word epoch disambiguation: Finding how words change over time. In *ACL'12*.

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

Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee, and Pawan Goyal. 2014. That's sick dude!: Automatic identification of word sense change across different timescales. In *ACL'14*.

G Palshikar et al. 2009. Simple algorithms for peak detection in time-series. In *Proc. 1st Int. Conf. Advanced Data Analysis, Business Analytics and Intelligence*, pages 1–13.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Octavian Popescu and Carlo Strapparava. 2013. Behind the times: Detecting epoch changes using large corpora. In *IJCNLP'13*.

James Pustejovsky, José M Castano, Robert Ingria, Roser Sauri, Robert J Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R Radev. 2003. Timeml: Robust specification of event and temporal expressions in text. *New directions in question answering*, 3:28–34.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of International World Wide Web Conference (WWW)*.

Kira Radinsky, Krysta Svore, Susan Dumais, Jaime Teevan, Alex Bocharov, and Eric Horvitz. 2012. Modeling and predicting behavioral dynamics on the web. In *Proceedings of International World Wide Web Conference (WWW)*.

Kira Radinsky, Krysta M. Svore, Susan T. Dumais, Milad Shokouhi, Jaime Teevan, Alex Bocharov, and Eric Horvitz. 2013. Behavioral dynamics on the web: Learning, modeling, and prediction. *ACM Transactions on Information Systems*, 31(3):16.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modeling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. 2016. Using word embeddings for automatic query expansion. *arXiv preprint arXiv:1606.07608*.

Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2009. Semantic density analysis: Comparing word meaning across time and phonetic space. In *Workshop on Geometrical Models of Natural Language Semantics*.

Yair Shimshoni, Niv Efron, and Yossi Matias. 2009. On the predictability of search trends. In *Technical Report*.

Milad Shokouhi. 2011. Detecting seasonal queries by time-series analysis. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*.

Milad Shokouhi and Kira Radinsky. 2012. Time-sensitive query auto-completion. In *Proceedings of the Special Interest Group on Information Retrieval (SIGIR)*.

Jaspreet Singh, Wolfgang Nejdl, and Avishek Anand. 2016. History by diversity: Helping historians search news archives. In *CHIIR'16*, pages 183–192.

Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. 2012. Coupled temporal scoping of relational facts. In *5th ACM international conference on Web search and data mining*, pages 73–82. ACM.

Peter D Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2012. Tempeval-3: Evaluating events, time expressions, and temporal relations. *arXiv preprint arXiv:1206.5333*.

P. Wang, M. W. Berry, and Y. Yang. 2003. Mining longitudinal web queries: trends and patterns. *Journal of the American Society for Information Science and Technology (JASIST)*, 54:743–758.

Derry Tanti Wijaya and Reyyan Yeniterzi. 2011. Understanding semantic change of words over centuries. In *2011 International Workshop on DETecting and Exploiting Cultural diversiTy on the Social Web*, DETECT '11.

Craig Willis and Miles Efron. 2013. Finding information in books: Characteristics of full-text searches in a collection of 10 million books. *Proceedings of the American Society for Information Science and Technology*, 50(1):1–10.

Qiankun Zhao, Steven C. H. Hoi, and Tie yan Liu. 2006. Time-dependent semantic similarity measure of queries using historical click-through data. In *Proceedings of International World Wide Web Conference (WWW)*.

Inter-Weighted Alignment Network for Sentence Pair Modeling

Gehui Shen Yunlun Yang Zhi-Hong Deng*
Key Laboratory of Machine Perception (Ministry of Education),
School of Electronics Engineering and Computer Science, Peking University,
Beijing 100871, China
{jueliangguke, incomparable-lun, zhdeng}@pku.edu.cn

Abstract

Sentence pair modeling is a crucial problem in the field of natural language processing. In this paper, we propose a model to measure the similarity of a sentence pair focusing on the interaction information. We utilize the word level similarity matrix to discover fine-grained alignment of two sentences. It should be emphasized that each word in a sentence has a different importance from the perspective of semantic composition, so we exploit two novel and efficient strategies to explicitly calculate a weight for each word. Although the proposed model only use a sequential LSTM for sentence modeling without any external resource such as syntactic parser tree and additional lexicon features, experimental results show that our model achieves state-of-the-art performance on three datasets of two tasks.

1 Introduction

Given two pieces of sentences S and T, sentence pair modeling (SPM) is a fundamental task whose applications include question answering (Lin, 2007), natural language inference (Bowman et al., 2015), paraphrase identification (Socher et al., 2011a) and sentence completion (Wan et al., 2016) and so on. In general, each of the two sentences are firstly mapped to a representation, and then a model is designed to determine the relation between them. Traditional methods use lexicon features such as Bag-of-Words(BOW) to map sentences. As we know, features design and selection are time-consuming and high dimensional features may suffer from sparsity because of the variation of linguistic. Recently, deep learning techniques have been applied to develop end-to-end models for NLP tasks, such as sentence modeling (Socher et al., 2011b; Kim, 2014), relation classification (Socher et al., 2012) and machine translation (Sutskever et al., 2014). These works show that deep learning models can be comparable with hand-crafted features based models and often outperform them.

Existing DNN models are based on pre-trained word embeddings which map each word to one low dimensional vector and compose word embeddings to represent sentence. Some models are developed directly from the sentence models. They obtain single vector representation for each sentence separately and then determine the relation based on two vectors (Huang et al., 2013; Qiu and Huang, 2015; Palangi et al., 2016). Because of the absence of interaction, these models can not achieve state-of-the-art performance.

Inspired by attention mechanism in computer vision and machine translation, some elaborate models have been proposed (Rocktäschel et al., 2016; Zhou et al., 2016; Wang and Jiang, 2016) which take interaction information into consideration. Meanwhile, to grasp the fine-grained information for semantic similarity, some prior works (Pang et al., 2016; He and Lin, 2016) firstly compute a word level similarity matrix according to word representation, and utilize multiple convolution layers and extract features from the similarity matrix in a perspective of image recognition.

In this paper, we focus on solving SPM problem by measuring semantic similarity between two sentences. We propose a new deep learning model based on two facts that previous works always neglected. As we know, in the aspect of semantic, each word in the sentence is of different importance. When calculating a sentence representation we should endow each word with a weight indicating its importance. Taking following sentences

*Corresponding author

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1179–1189
Copenhagen, Denmark, September 7–11, 2017.

as an example:

A: a man with a red helmet is riding a motorbike along a roadway.

B: a man is riding a motorbike along a roadway.

C: a man with a red helmet is riding a bicycle along a roadway.

We can see that sentence A is more similar with sentence B than with sentence C while a conventional model probably makes an opposite conclusion because the phrase ”with a red helmet” will bias the meaning of A to C meanwhile the difference between ”motorbike” and ”bicycle” is not large enough. If the model can realize that the phrase ”with a red helmet” has little effect on semantic composition, the mistake will be avoided. Since we have to analyse a pair of sentences, the weights should be related to not only the sentence itself, but also its partner. From this point, we propose a novel inter-weighted layer to measure the importance of each word.

On the other hand, the more similar two sentences are, the more probably we can align each word of sentence S with several words of sentence T, and vice versa. On account of the variety of expression, the position and length of two aligned parts are very likely different, so we apply soft-alignment mechanism and build an effective alignment layer.

In summary, our contributions are as follows:

1. We propose an Inter-Weighted Alignment Network (IWAN) for SPM, which builds an alignment layer to compute similarity score according to the degree of alignment.

2. Considering the importance of each word in a sentence is different, we argue that an inter-weighted layer for evaluating the weight of each word is crucial to semantic composition. We propose two strategies for calculating weights. Experimental results demonstrate their effectiveness.

3. Experimental results on semantic relatedness benchmark dataset SICK and two answer selection datasets show that proposed model achieves state-of-the-art performances without any external information.

2 Related Work

2.1 Sentence Models

For sentence modeling, RNN (Elman, 1990; Mikolov et al., 2010) and CNN (Kim, 2014) are both powerful and widely used. RNN models a sentence sequentially by updating the hidden state which represents context recurrently. As sentence length grows, RNN will suffer from gradient vanishing problem. However, gated mechanism, such as Long Short Term Memory(LSTM) (Hochreiter and Schmidhuber, 1997) is introduced to address it. RecNN exploits syntatic information and models sentences under a tree structure. Gated mechanism can also improve the performance of RecNN (Tai et al., 2015). CNN can extract and combine important local context meanwhile model sentences in a hierarchical way (Kim, 2014; Kalchbrenner et al., 2014). All of the above models can be adapted to SPM by modeling two sentences separately.

2.2 Attentive Models

Hermann et al. (2015) firstly introduces attention mechanism into question answering under an RNN architecture. Rocktäschel et al. (2016) applies a similar model to natural language inference which attends over the premise conditioned on the hypothesis. Zhou et al. (2016) combines attention mechanism with tree-structured RecNN encoder. Some prior works (Wang et al., 2016b; Parikh et al., 2016; Wang et al., 2017) compute soft-alignment representation for each word in sentences attentively with word level similarity and then compose the alignment representations to determine the relation. Our model is also under this framework however we focus on explicitly calculating weights for each word to get more reasonable semantic composition.

2.3 Similarity Matrix Based Models

Pang et al. (2016) adopts CNN on word level similarity matrix to extract fine-grained matching patterns from different text granularity. He and Lin (2016) uses a similar architecture with a 19-layer CNN in order to make full use of its power. Yin and Schütze (2015) proposes a hierarchical architecture to model different granularity representation and computes several similarity matrices for interaction.

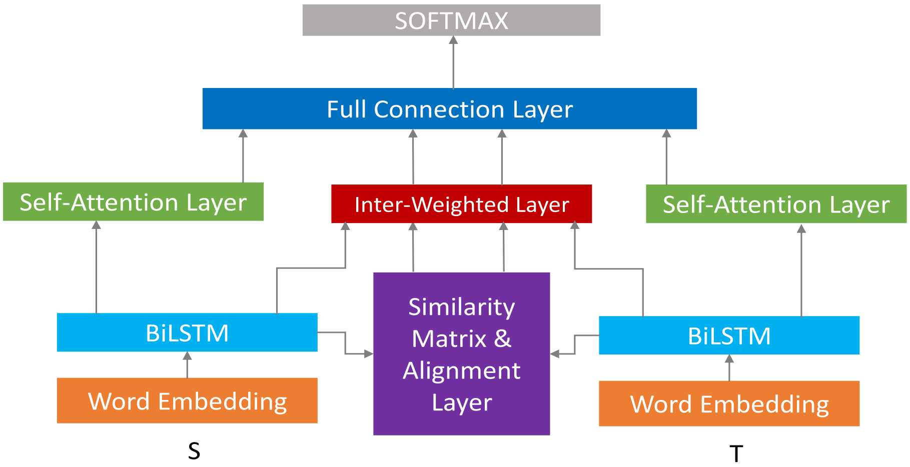


Figure 1: The architecture of IWAN. The blocks with same color have shared parameters.

3 Proposed Model

Given two sentences S and T, we aim to calculate a score to measure their similarity. Figure 1 shows the architecture of IWAN model. To learn representations with context information, we firstly use a bi-direction LSTM sentence model which takes word embeddings as inputs to obtain a context-aware representation for each position (Sec. 3.1). The context-aware representations are used to compute the word level similarity matrix (Sec. 3.2). Inspired by attention mechanism, we exploit soft-alignment to find semantic counterpart in one sentence for each position in the other and compute a weighted sum vector of one sentence as the alignment representation of each position of the other with an alignment layer (Sec. 3.3). Meanwhile, taking the context-aware representation of S and T as inputs, we apply an inter-weighted layer to compute a weight for each position in S and T. We argue that this weight can indicate the importance in semantic interaction and a weighted summation of the representations at each position is more interpretable than other composition method including max or average pooling and LSTM layer. We propose two strategies for computing those weights (Sec. 3.4). The weighted vectors are fed to full connection layers and a softmax layer is used to give the final prediction(Sec. 3.5).

As Figure 1 illustrates, our model is symmetric about S and T. So for simplicity, we only describe the left part of IWAN model which is mainly about modeling S from here. Right part is exactly same except the roles of S and T exchange.

3.1 BiLSTM Layer

With pre-trained d dimension word embedding, we can obtain sentence matrices $S_e = [s_e^1, \dots, s_e^m]$ and $T_e = [t_e^1, \dots, t_e^n]$ where $s_e^i \in \mathbb{R}^d$ is embedding of the i-th word in sentence S. m and n are the length of S and T respectively. In order to capture contextual information, we run a bi-direction LSTM (Hochreiter and Schmidhuber, 1997) on two matrices. Let hidden layer dimension of LSTM be u. Given the word embedding x_t at time step t, previous hidden vector h_{t-1} and cell state c_{t-1}, LSTM recurrently computes h_t and c_t as follows:

$$\begin{aligned}
g_t &= \phi(W_g x_t + V_g h_{t-1} + b_g), \\
i_t &= \sigma(W_i x_t + W_i h_{t-1} + b_i), \\
f_t &= \sigma(W_f x_t + W_f h_{t-1} + b_f), \\
o_t &= \sigma(W_o x_t + W_o h_{t-1} + b_o), \\
c_t &= g_t \odot i_t + c_{t-1} \odot f_t, \\
h_t &= c_t \odot o_t.
\end{aligned}$$

where all $W \in \mathbb{R}^{u \times d}$, $V \in \mathbb{R}^{u \times u}$ and $b \in \mathbb{R}^u$. σ is *sigmoid* function and ϕ is *tanh* function. $\odot$ indicates the element-wise multiplication of two vectors. The input gates i, forget gates f and output gates o control information flow self-adaptively, moreover cell state c_t can memorize long-distance information. h_t is regarded as the representation of time step t.

We feed S_e and T_e separately into a parameter shared LSTM sentence model. If we run an LSTM model on the sequence of S_e from left to right, we can get the forward hidden vector $S_{fh} = [s_{fh}^1, \dots, s_{fh}^m]$. For applying bi-direction LSTM, we also run another LSTM backward and get

$S_{bh} = [s_{bh}^1, \dots, s_{bh}^m]$. Then we concatenate them to one vector representation. So after bi-direction LSTM layer, we obtain $S_h = [s_h^1, \dots, s_h^m]$ and $T_h = [t_h^1, \dots, t_h^n]$ where $s_h^i = \begin{bmatrix} s_{fh}^i \\ s_{bh}^i \end{bmatrix}$.

3.2 Word Level Similarity Matrix

As mentioned above, the word level similarity matrix is crucial to making use of the interaction information. Pang et al. (2016) and Wang et al. (2016b) compute the similarity matrix between two word embeddings. We have argued that word embedding can not express the word meaning in context. From the view of RNN, s_{fh}^i contains the most semantic information about i-th word in S and less about the leftmost words, while s_{bh}^i also contains the most semantic information about i-th word in S and less about the rightmost words. Therefore, the hidden vector of BiLSTM keeps the most information of corresponding word as well as integrated with the context information. Computing similarity matrix between BiLSTM hidden vectors is expected to improve the interaction results. We regard the inner dot of two vectors as their similarity. For the similarity matrix M, its element M_{ij} indicates the similarity between s_h^i and t_h^j:

$$M_{ij} = s_h^{iT} \cdot t_h^j.$$

3.3 Alignment Layer

We design the alignment layer for an intuitive idea: more similar S and T are, more probably we can find semantic counterpart in T for each part in S, and vice versa. To some degree, people are likely to find semantic correspondences between two sentences and evaluate their similarity. He and Lin (2016) are also inspired by similar intuition, but they use deep CNN to recognize the alignment patterns implicitly. However, for each sentence, we explicitly calculate the alignment representation and alignment residual which we believe are good indicators of sentence pair similarity.

For calculating the alignment representation, we apply attention mechanism (Bahdanau et al., 2014) to conduct a soft-alignment. The original attention mechanism outputs the alignment weights from an extra full connection layer while we think the inner dot can represent the semantic relatedness adequately. Therefore, we consider the i-th row of M as the similarity between the i-th position of S and each position in T and normalize it as follows:

$$\alpha_{ij} = \frac{exp(M_{ij})}{\sum_{k=1}^{n} exp(M_{ik})}, \quad i = 1, \dots, m$$

while we also normalize each column of M for T counterpart. α_{ij} always belongs to $[0, 1]$ and can be regarded as weight. Then the alignment representation $S_a = [s_a^1, \dots, s_a^m]$ is computed as a weighted sum of $\{t_h^j\}$:

$$s_a^i = \sum_{k=1}^{n} \alpha_{ik} t_h^k, \quad i = 1, \dots, m$$

For T counterpart, the alignment representation is $T_a = [t_a^1, \dots, t_a^n]$.

In order to measure the gap between the alignment representation and original representation, a *direct* strategy is to compute the absolute value of their difference: $s_r^i = |s_h^i - s_a^i|$. We call $S_r = [s_r^1, \dots, s_r^m]$ alignment residual which is considered as alignment feature for subsequent processing.

We also utilize an *orthogonal decomposition* strategy which is first proposed by Wang et al. (2016b): the component s_p^i of s_h^i parallel to s_a^i represents the alignment component and component s_o^i orthogonal to s_a^i represents alignment residual. We compute these two component as follows:

$$s_p^i = \frac{s_h^i \cdot s_a^i}{s_a^i \cdot s_a^i} s_a^i, \quad parallel\ component$$

$$s_o^i = s_h^i - s_p^i, \quad orthogonal\ component$$

Then we can replace S_r with S_p and S_o to measure the degree of alignment where $S_p = [s_p^1, \dots, s_p^m]$ and $S_o = [s_o^1, \dots, s_o^m]$.

3.4 Inter-Weighted Layer

3.4.1 Inter-Attention Layer

(Lin et al., 2017) firstly proposes a self-attention sentence model which explicitly computes a weight for each word and uses the weighted summation of word representations as sentence embedding. Inspired by this work, we apply a full connection neural network to measure the importance to semantic interaction of every word. We extend the self-attention model to inter-attention layer in order to compute the weights combined with interaction information which composing the alignment representation benefits from. As the

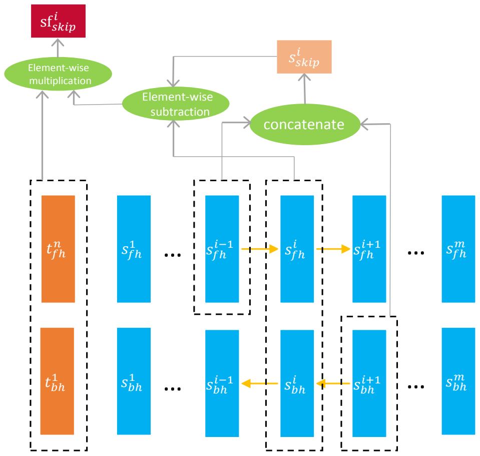


Figure 2: The illustration of computing sf^i_{skip}.

name suggests, these weights of S are not only dependent on S but also T and the parameters of the inter-attention layer are shared for S and T.

Formally, we take S_h and T_h as inputs and the inter-attention layer outputs a vector w_s with size m for S:

$$w_s = softmax(w_2 tanh(W_1 \begin{bmatrix} S_h \\ (t_{avg} \otimes e_m) \end{bmatrix})),$$

where $t_{avg} = \frac{1}{n}\sum_{k=1}^{n} t_h^k$ and $S_h \in \mathbb{R}^{2u \times m}$. We calculate the average of $\{t_h^k\}$ as the representation of T. We also try to replace average operator with a self-attention layer (Lin et al., 2017) but get a worse performance. e_m is a vector of 1s with size m and $\otimes$ represents outer product operator. We feed the concatenated matrix containing pairwise information into a 2-layer neural network. The parameter $W_1 \in \mathbb{R}^{s \times 4u}$ projects inputs into a hidden layer with s units. The output layer is parameterized by a vector w_2 with size s and a $softmax$ operator ensures all the element of output sum up to 1. Then we can use w_s to sum up S_r, S_p and S_o weightedly across the position dimension:

$$\begin{aligned} s_{wr} &= S_r * (w_s)^T, \\ s_{wp} &= S_p * (w_s)^T, \\ s_{wo} &= S_o * (w_s)^T. \end{aligned}$$

We can get t_{wr}, t_{wp} and t_{wo} in the same way. We call these inter-features for final prediction.

3.4.2 Inter-Skip Layer

We also explore another novel strategy to compute w_s from the intuition that if the i-th word in S has a low contribution to semantic composition, we will obtain a similar representation s^i_{skip} if we feed all word embeddings sequentially except s^i_e into BiLSTM. Unfortunately, the $O(m^2)$ complexity of running BiLSTM model m times is too high so we exploit an approximate method to compute $\{s^i_{skip}\}$:

$$s^i_{skip} = \begin{bmatrix} s^{i-1}_{fh} \\ s^{i+1}_{bh} \end{bmatrix}$$

Then we compute a skip feature as following:

$$sf^i_{skip} = (s^i_{skip} - S^i_h) \odot t_h,$$

where $t_h = \begin{bmatrix} t^n_{fh} \\ t^1_{bh} \end{bmatrix}$ is the BiLSTM hidden representation of T. Figure 2 illustrates how to compute sf^i_{skip}. We think the difference between s^i_{skip} and s^i_h approximately reflects the contribution the i-th word makes to semantic composition. On the one hand, if the difference is small or even close to zero, the importance of correspond word should be small. On the other hand, if the difference (a vector) is not similar to the representation of T, correspond word is probably of less importance in measuring semantic similarity. From these two points, we think $sf_{skip} = [sf^1_{skip}, \ldots, sf^m_{skip}]$ is a good feature to measure word importance. The process of computing w_s is similar:

$$w_s = softmax(w_2 tanh(W_1 sf_{skip})),$$

We can use w_s outputted by inter-skip layer to obtain inter-features in the same way.

3.5 Output Layer

For more rich information, we combine alignment information with sentence embeddings of S and T for final prediction. We run the simple but effective self-attention (Lin et al., 2017) model on S_h to obtain its embedding s_{wh}:

$$s_{wh} = S_h * (softmax(w_2' tanh(W_1' * S_h)))^T,$$

where W_1' and w_2' are trainable. We compute s_{wh} and t_{wh} with parameter shared self-attention layer which is similar with the inter-attention layer except inputs.

Following Tai et al. (2015), we compute their element-wise product $h_\times = s_{wh} \odot t_{wh}$ and their absolute difference $h_+ = |s_{wh} - t_{wh}|$ as self-features. If we use *direct strategy*, we combine the features as follows:

$$h_{di} = [h_\times^T; h_+^T; s_{wr}^T; t_{wr}^T]^T.$$

Strategy	SICK			TrecQA		WikiQA	
	r	ρ	MSE	MAP	MRR	MAP	MRR
DI	0.8774	0.8229	0.2374	0.815	0.882	0.724	0.739
OD	**0.8810**	**0.8261**	**0.2289**	**0.822**	**0.889**	**0.730**	**0.744**

Table 1: Performances of our model with different strategies in alignment layer on three datasets.

If we use *orthogonal decomposition strategy*, we combine the features as follows:

$$h_{od} = [h_{\times}^T; h_{+}^T; s_{wp}^T; t_{wp}^T; s_{wo}^T; t_{wo}^T]^T.$$

Following previous works, the sentence pair modeling problem can always be considered as a classification task, so we finally calculate a probability distribution with a 2-layer neural network:

$$\hat{p_\theta} = softmax(V_2 ReLU(V_1 h + b_1) + b_2),$$

where h can be h_{di} or h_{od} and the hidden size is l. We use rectified linear units (ReLU) as activation function.

4 Experimental Setup

4.1 Dataset and Evaluation Metric

To evaluate the proposed model, we conduct experiments on two tasks: semantic relatedness and answer selection.

For semantic relatedness task, we use the Sentences Involving Compositional Knowledge (SICK) dataset (Marelli et al., 2014), which consists of 9927 sentence pairs in a 4500/500/4927 train/dev/test split. The sentences are derived from existing image and video description and each sentence pair has a relatedness score $y \in [1, 5]$, where the larger score indicates more similarity between two sentences. As the goal of this task is to calculate sentence pair similarity, we can directly evaluate our model on SICK. Following previous works, we use Pearson's Correlation r, Spearman's Correlation ρ and mean square error (MSE) as evaluation metrics.

For answer selection task, we experiment on two datasets: TrecQA and WikiQA. The TrecQA dataset (Wang et al., 2007) from the Text Retrieval Conferences has been widely used for the answer selection task during the past decade. The original TrecQA train dataset consists of 1,229 questions with 53,417 question-answer pairs, 82 questions with 1,148 pairs in development set, and 100 questions with 1,517 pairs in test set. Recent works (dos Santos et al., 2016; Rao et al., 2016; Wang et al., 2016b) removed questions in development and test set with no answers or with only positive/negative answers, thus there are 65 questions with 1,117 pairs in *Clean version* development set and 68 questions with 1,442 pairs in *Clean version* test set. Rao et al. (2016) has showed the performances on Original TrecQA and *Clean version* TrecQA are not comparable. Therefore, for a fair comparison, we only display the results on *Clean version* TrecQA which are posted on the website of Wiki of the Association for Computational Linguistics[1]. The open domain question-answering WikiQA (Yang et al., 2015) is constructed from real queries of Bing and Wikipedia. We follow Yang et al. (2015) to remove all questions with no correct candidate answers. The excluded WikiQA has 873/126/243 questions and 8627/1130/2351 question-answer pairs for train/dev/test split. To adapt our model to this task, we use semantic similarity to measure the probability of matching between a question and a candidate answer. We evaluate models by mean average precision (MAP) and mean reciprocal rank (MRR).

4.2 Training Details

For experiments on SICK, we follows Tai et al. (2015) to transform the relatedness score y to a sparse target distribution p:

$$p_i = \begin{cases} y - \lfloor y \rfloor, & i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor + 1 - y, & i = \lfloor y \rfloor \\ 0, & otherwise \end{cases}$$

for $1 \leq i \leq 5$. The training objective is to minimize the KL-divergence loss between p and $\hat{p_\theta}$:

$$loss = \frac{1}{|D|} \sum_{k=1}^{|D|} KL(p^{(k)} \parallel \hat{p_\theta}^{(k)})$$

where $|D|$ is the number of training examples.

We regard the answer selection problem as "yes" or "no" binary classification and the training objective is to minimize the negative log-likelihood in training stage:

$$loss = -\frac{1}{|D|} \sum_{k=1}^{|D|} log\hat{p_\theta}^{(k)}(y^{(k)}|x^{(k)})$$

where $x^{(k)}$ represents a question-answer pair and $y^{(k)}$ indicates whether the candidate answer is cor-

[1] https://www.aclweb.org/aclwiki/index.php?title=Question_Answering_(State_of_the_art)

rect to the question. In test stage, we sort condidate answers for same question in descending order by probability of "yes" category and calculate MAP and MRR.

In all experiments, we use 300-dimension GloVe word embeddings[2] (Pennington et al., 2014) and fix the embeddings during training. The LSTM hidden size u is set to 150. The hidden size of inter-attention and self-attention layer s and full connection network l are both set to 50. The L2 regularization strength is set to 3×10^{-5}. We train the model with Adagrad (Duchi et al., 2011) optimization algorithm with a learning rate of 0.05. The minibatch size is always 25. We exploit early stopping strategy according to MSE on development set for SICK and MAP on development set for TrecQA and WikiQA.

Model	r	ρ	**MSE**
Meaning Factory (Jiménez et al., 2014)	0.8268	0.7721	0.3224
ECNU (Zhao et al., 2014)	0.8414	-	-
BiLSTM (Tai et al., 2015)	0.8567	0.7966	0.2736
Tree-LSTM (Tai et al., 2015)	0.8676	0.8083	0.2532
MPCNN (He et al., 2015)	0.8686	0.8047	0.2606
PWIM (He and Lin, 2016)	0.8784	0.8199	0.2329
Att Tree-LSTM (Zhou et al., 2016)	0.8730	0.8117	0.2426
Skip-thought+COCO* (Kiros et al., 2015)	0.8655	0.7995	0.2561
MaLSTM$^{*\circ}$ (Mueller and Thyagarajan, 2016)	0.8822	0.8345	0.2286
IWAN-att (Proposed)	0.8810	0.8261	0.2289
IWAN-skip (Proposed)	**0.8833**	**0.8263**	**0.2236**

Table 2: Test results on SICK. The symbol * indicates the models with pre-training. The symbol ° indicates the models with data augmentation strategy.

Model	**MAP**	**MRR**
Wang and Ittycheriah (2015)	0.746	0.820
QA-LSTM (Tan et al., 2015)	0.728	0.832
Att-pooling (dos Santos et al., 2016)	0.753	0.851
LDC (Wang et al., 2016b)	0.771	0.845
MPCNN (He et al., 2015)	0.777	0.836
PWIM (He and Lin, 2016)	0.738	0.827
NCE-CNN (Rao et al., 2016)	0.801	0.877
BiMPM (Wang et al., 2017)	0.802	0.875
IWAN-att (Proposed)	**0.822**	**0.889**
IWAN-skip (Proposed)	0.801	0.861

Table 3: Test results on *Clean version* TrecQA.

Model	**MAP**	**MRR**
NASM (Miao et al., 2016)	0.689	0.707
Att-pooling (dos Santos et al., 2016)	0.689	0.696
LDC (Wang et al., 2016b)	0.706	0.723
MPCNN (He et al., 2015)	0.693	0.709
PWIM (He and Lin, 2016)	0.709	0.723
NCE-CNN (Rao et al., 2016)	0.701	0.718
IARNN$^\circ$ (Wang et al., 2016a)	**0.734**	0.742
BiMPM (Wang et al., 2017)	0.718	0.731
IWAN-att (Proposed)	0.730	0.744
IWAN-skip (Proposed)	**0.733**	**0.750**

Table 4: Test results on WikiQA. The symbol ° indicates the models with data augmentation strategy.

4.3 Results

Firstly, we evaluate the effectiveness of two strategies in alignment layer. We use inter-attention model in inter-weighted layer and we find *orthogonal decomposition* (OD) strategy has a superior performance to *direct* (DI) strategy on all datasets. The comparison results are posted in Table 1. In following experiments, we always choose OD strategy in alignment layer.

Semantic Relatedness Table 3 shows the performances of our model and compared models on SICK dataset. IWAN-att and IWAN-skip represents our models using inter-attention layer and inter-skip layer respectively. IWAN-skip outperforms IWAN-att in all metrics by a small margin. The traditional feature engineering based models in first group have much poorer performances than deep learning models. MaLSTM (Mueller and Thyagarajan, 2016) benefits from the data argumentation strategy with Wordnet information and pre-training process. Ablation experiments (Mueller and Thyagarajan, 2016) illustrates a 0.04 degradation of Pearson's r without data argumentation strategy. Therefore it is unfair to compare with this model directly, but our models achieve a comparable performance with it. Our models both outperform all other deep learning models. IWAN-skip outperforms Attentive Tree-LSTM (Zhou et al., 2016) by 0.01 in Pearson's r, over 0.01 in Spearman's ρ and almost 0.02 in MSE, although it exploits syntactic parser information. Our model significantly outperforms sentence modeling based models with CNN or RNN which results from the absence of interaction information. He and Lin (2016) proposes Pairwise Word Interaction Model (PWIM) which constructs 19-layer CNN on similarity matrix to capture fine-grained interaction information and shows most competitive. However our model outperforms it in all metrics with much fewer parameters (about 0.65 million versus 1.7 million (He and

[2] http://nlp.stanford.edu/projects/glove/

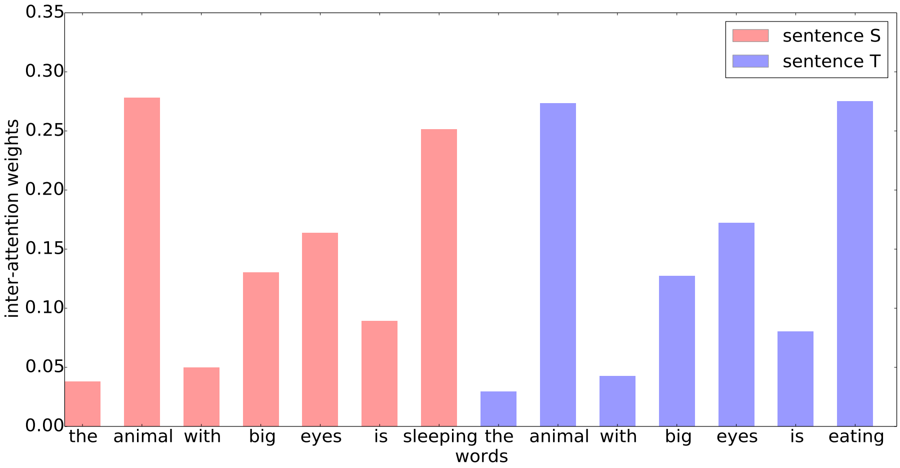


Figure 3: Visualization of weights outputted by inter-weighted layer of words in a sentence pair in SICK test set.

Lin, 2016)).

Answer Selection We compare our model with several state-of-the-art models on *Clean version* TrecQA and WikiQA in Table 3 and Table 4 respectively. Our two models both have a state-of-the-art performance on two datasets. IWAN-att outperform all previous works on TrecQA and make a significant improvement of state-of-the-art. IWAN-skip and IARNN (Wang et al., 2016a) which solves bias problem of attention mechanism beat all other models on WikiQA, while the latter is trained on an argumented dataset with negative sampling. Wang et al. (2016b) first proposes the orthogonal decomposition but their LDC model compute the similarity matrix between word embeddings which are lack of context information and IWAN-att outperforms it dramatically by 0.02-0.05 in MAP and MRR on both datasets. The PWIM (He and Lin, 2016) is still competitive on WikiQA but gets an inferior performance on TrecQA. However, our models both have state-of-the-art performances on three datasets which demonstrates our models have excellent generalization ability in different datasets.

4.4 Ablation Tests

Table 5 show the results of ablations tests on SICK dataset in r metric. From IWAN-att, we remove or replace one component at a time and evaluate performance of partial models. If removing inter-features, the r degrades with a 0.013 decline which proves the interaction information is crucial for sentence pair modeling. Whereas, the degradation from removing self-features is much smaller. We found a large decline when removing BiLSTM layer, which confirms our conjecture that context information is useful. It is worth mentioning that He and Lin (2016) posts the degradation of their model from removing BiLSTM is 0.1225 in r which is much larger than 0.0387 of us. Removing inter-attention layer means we perform a mean-pooling on inter-features instead of a weighted summation. A 0.0075 r degradation proves importance weighting can result in a significant improvement. If the weights are only about single sentence information, the performance still gets worse. The last two settings show both components from orthogonal decomposition are informative. More or less unexpected, parallel component is almost as useful as orthogonal component.

Ablation setting	**Pearson's r**
Full Model (IWAN-att)	0.8810
• w/o inter-features	-0.0130
• w/o self-features	-0.0070
• w/o BiLSTM layer	-0.0387
• w/o inter-attention layer	-0.0075
• Replace inter-attention weights with self-attention weights	-0.0063
• w/o parallel component	-0.0037
• w/o orthogonal component	-0.0046

Table 5: Ablation test on SICK dataset, removing each component separately.

4.5 Visualization of Inter-Weighted Layer

In order to illustrate the effect of inter-weighted layer in proposed model, we take a sentence pair in SICK test set as an example and display the weights outputted by inter-attention layer of each word in Figure 3. The ground truth of this pair is

3.2 and the prediction given by IWAN-att model is 3.507 which is much more accurate than 4.356 given by the model without inter-attention layer. We can find the inter-attention layer gives very high weights over 0.25 (while the average weight is about 0.14) to “sleeping” and “eating” which are the only difference between two sentences. Therefore, the difference will be attended in following processing. Meanwhile, the weights of the article “the” and the preposition “with” which are not as important as other real words in semantic composition are much lower. These prove the inter-weighted mechanism is reasonable and effective.

5 Conclusion

This work proposes a weighted alignment model for sentence pair modeling. We utilize an alignment layer to measure the similarity of sentence pairs according to their degree of alignment. Moreover, we propose an inter-weighted layer to measure the importance of different parts in sentences. Two strategies for this layer have been explored which are both effective. The composition of alignment features can benefit from the inter-weighted weights. Experiment results shows that proposed models achieve the state-of-the-art performance on three datasets. In the future work, we will improve the inter-weighted layer with more sophisticated module and evaluate our model on other large scale datasets.

Acknowledgments

This work is partially supported by the National High Technology Research and Development Program of China (Grant No. 2015AA015403) and the National Natural Science Foundation of China (Grant No. 61170091). We would also like to thank the anonymous reviewers for their helpful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Hua He, Kevin Gimpel, and Jimmy J. Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1576–1586.

Hua He and Jimmy J. Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 937–948.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.

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

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 2333–2338.

Sergio Jiménez, George Dueñas, Julia Baquero, and Alexander F. Gelbukh. 2014. UNAL-NLP: combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014.*, pages 732–742.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 655–665.

Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3294–3302.

Jimmy J. Lin. 2007. An exploration of the principles underlying redundancy-based factoid question answering. *ACM Trans. Inf. Syst.*, 25(2).

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of the International Conference on Learning Representations, ICLR 2017*.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014.*, pages 1–8.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1727–1736.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2786–2792.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab K. Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 24(4):694–707.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2793–2799.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2249–2255.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1305–1311.

Jinfeng Rao, Hua He, and Jimmy J. Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 1913–1916.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of the International Conference on Learning Representations, ICLR 2016*.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pages 801–809.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 1201–1211.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh,* *UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 151–161.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1556–1566.

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *CoRR*, abs/1511.04108.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2835–2841.

Bingning Wang, Kang Liu, and Jun Zhao. 2016a. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the jeopardy model? A quasi-synchronous grammar for QA. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, pages 22–32.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with LSTM. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1442–1451.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. *CoRR*, abs/1702.03814.

Zhiguo Wang and Abraham Ittycheriah. 2015. Faq-based question answering via word alignment. *CoRR*, abs/1507.02628.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016b. Sentence similarity learning by lexical decomposition and composition. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 1340–1349.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 2013–2018.

Wenpeng Yin and Hinrich Schütze. 2015. Multigrancnn: An architecture for general matching of text chunks on multiple levels of granularity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 63–73.

Jiang Zhao, Tiantian Zhu, and Man Lan. 2014. ECNU: one stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014.*, pages 271–277.

Yao Zhou, Cong Liu, and Yan Pan. 2016. Modelling sentence pairs with tree-structured attentive encoder. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2912–2922.

A Short Survey on Taxonomy Learning from Text Corpora: Issues, Resources and Recent Advances

Chengyu Wang, Xiaofeng He[*], Aoying Zhou
Shanghai Key Laboratory of Trustworthy Computing,
School of Computer Science and Software Engineering, East China Normal University
chywang2013@gmail.com, {xfhe,ayzhou}@sei.ecnu.edu.cn

Abstract

A taxonomy is a semantic hierarchy, consisting of concepts linked by is-a relations. While a large number of taxonomies have been constructed from human-compiled resources (e.g., Wikipedia), learning taxonomies from text corpora has received a growing interest and is essential for long-tailed and domain-specific knowledge acquisition. In this paper, we overview recent advances on taxonomy construction from free texts, reorganizing relevant subtasks into a complete framework. We also overview resources for evaluation and discuss challenges for future research.

1 Introduction

A taxonomy is a semantic hierarchy that organizes concepts by is-a relations, which exhibits the capability of improving many NLP and IR tasks, such as query understanding (Hua et al., 2017), personalized recommendation (Zhang et al., 2014), question answering (Yang et al., 2017), etc. It also supports a variety of real-world applications, including information management (Nickerson et al., 2013), e-commerce (Aanen et al., 2015) and biomedical systems (Köhler et al., 2014).

With massive Web data available, a number of taxonomies are constructed from human-compiled resources such as Wikipedia, Wikidata, etc (Suchanek et al., 2007; Ponzetto and Navigli, 2009; Flati et al., 2014; Mahdisoltani et al., 2015). But even large taxonomies may lack domain-specific and long-tailed knowledge. Recently, several methods have been developed to induce taxonomies from text corpora (Wu et al., 2012; Yang, 2012; Luu et al., 2014). However, this task is far from being solved for three reasons: i) Text corpora may vary in size, topic and quality. It is unlikely to develop a "one-size-fits-all" solution for all scenarios. For example, given an extremely large corpus, Hearst-pattern based method is employed to build Probase (Wu et al., 2012). For domain-specific corpora, learning hypernymy embedding is more suitable (Luu et al., 2016b). ii) The accuracy of free-text taxonomies is usually lower than many Wikipedia-based taxonomies because it is difficult to extract knowledge completely from texts; iii) The task of taxonomy learning is still insufficiently studied a) in emerging and specific domains and b) for non-English or under-resourced languages (Wei et al., 2014; Alfarone and Davis, 2015; Wang et al., 2015).

In this paper, we overview recent advances on taxonomy construction from text corpora, reorganizing relevant subtasks into a complete framework. The subtasks include hyponym acquisition, hypernym prediction, taxonomy induction, etc. We also summarize resources, evaluation metrics and state-of-the-art results. We also discuss issues and directions for future research.

2 Taxonomy Construction Techniques

Although workflows of different methods vary, a free text-based taxonomy construction system typically operates in two steps: i) extracting is-a relations using *pattern-based* (Sect. 2.1) or *distributional* methods (Sect. 2.2); ii) constructing a complete taxonomy from is-a relations (Sect. 2.3).

2.1 Pattern-based Methods

Traditional pattern based methods predict that there is an is-a relation between a term pair (x, y), if x and y appear in the same sentence and satisfy a particular pattern. The earliest and most influential work in this field is Hearst (1992), which handcrafts several lexical patterns to harvest is-a relations. A typical pattern is "[C] such as [E]", where

[*] Corresponding author.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1190–1203
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

[C] and [E] are placeholders of noun phrases that are regarded as the hypernym (class) y and the hyponym (entity) x respectively for an is-a relation (x, y). Based on Hearst patterns, Probase is constructed from billions of Web pages. It consists of 2.65 million concepts and 20.76 million is-a relations (Wu et al., 2012). Similar approaches are presented in Etzioni et al. (2004); Kozareva and Hovy (2010), which employ Hearst patterns to induce taxonomies from Web pages.

Despite the successful applications, these patterns are too specific to cover all linguistic circumstances, thus recall is sacrificed (Wu et al., 2012). Simple pattern matching is prone to error due to idiomatic expressions, parsing errors, incomplete/uninformative extractions and ambiguous issues (Kozareva et al., 2008; Etzioni et al., 2004). In the next part, we summarize key techniques to improve precision and recall for pattern-based methods. Note that a robust is-a relation extraction system may combine multiple techniques to achieve high precision and recall.

2.1.1 Methods Improving Recall

Pattern Generalization Several approaches either extend original Hearst patterns by linguistic rules or learn more generalized lexico-syntactic patterns. Ritter et al. (2009) increase recall by replacing the noun phrase "[E]" (i.e., candidate hyponym) in Hearst patterns with a list of k noun phrases. Luu et al. (2014) design more flexible patterns where a few words in such patterns are interchangeable. Automatic methods mine is-a patterns given a collection of seed instances as input. Snow et al. (2004) use the dependency path of two terms to represent the pattern, where both syntactic and lexical connections of two terms can be modeled. This practice is more resistent to noise than surface matching and is employed by a number of relation extraction systems (Snow et al., 2006; Banko et al., 2007; Shwartz et al., 2016).

The number of patterns generated from a text corpus is sufficiently large, causing the feature sparsity problem. Learning more abstract patterns from these "raw" patterns can improve the generality of these patterns, hence increases recall. Navigli and Velardi (2010) introduce the concept "star pattern" (which use wildcards to replace non-frequent words in sentences). More general patterns are created by clustering star patterns. In the PATTY system (Nakashole et al., 2012), a subset of words along the dependency path are replaced by their POS tags, ontological types or wildcards.

Iterative Extraction Incorrect relations are frequently extracted from overly generalized patterns due to language ambiguity and *semantic drift* (Carlson et al., 2010). In contrast to above-mentioned approaches, an opposite idea is to use extremely specific patterns. Kozareva et al. (2008) employ "doubly-anchored" patterns (e.g., "cars such as Ford and *") to harvest hyponyms for a particular hypernym and expand both hyponyms and hypernyms by a bootstrapping loop. It uses each pattern as a query and takes search engine results as a Web corpus. Another advantage is that the ambiguity of terms can be eliminated by "doubly-anchored" patterns. Similar to Kozareva and Hovy (2010); Carlson et al. (2010), new is-a relations and hypernym patterns are iteratively extracted in an automatic manner.

Hypernym Inference This type of methods overcome the limitation where x and y must appear in the same sentence. The idea of Ritter et al. (2009) is that if y is a hypernym of x and another term $x^{'}$ is sufficiently similar to x, there is a high probability that y is a hypernym of $x^{'}$. They train an HMM to learn a better similarity measure than vector-based approaches. In the Syntactic Contextual Subsumption (SCS) method (Luu et al., 2014), given a non-taxonomic relation r, denote $S_r(x)$ as the collection of objects such that for each $s \in S_r(x)$, x and s has the relation r. If $S_r(y)$ mostly contains $S_r(x)$ but not vice versa, we can infer y is a hypernym of x.

Syntactic inference on hyponym modifiers can generate additional is-a relations. For example, the machine can infer a grizzly bear is a bear based on the evidence that the head word of "grizzly bear" is "bear". In Taxify (Alfarone and Davis, 2015), the system adds the linguistic head of a multi-word term as its direct hypernym if the term is added to the taxonomy. Suchanek et al. (2007) link conceptual Wikipedia categories to WordNet synsets based on category head words. Gupta et al. (2016) introduce linguistic heuristics to derive is-a relations from Wikipedia category network. Besides English, a similar observation also holds for Chinese, as shown in Fu et al. (2013); Li et al. (2015).

2.1.2 Methods Improving Precision

Confidence Assessment After candidate is-a pairs (x, y) are extracted, statistical measures can be used to estimate confidence scores. Relations with low scores are discarded. In KnowItAll (Etzioni et al., 2004), the system estimates the pointwise mutual information (PMI) of x and y by search engine hit counts. Probase (Wu et al., 2012) employs the ratio of likelihood to determine the most possible hypernym y for a concept x, and reversely the most possible hyponym x for a concept y. Wu et al. (2012) further calculate the plausibility of extracted is-a pairs based on a Naive Bayes classifier. Besides statistics from extraction results, Luu et al. (2014, 2015) consider external factors, such as the inclusion of concepts in WordNet and dictionaries, as well as the trustworthiness of data sources (e.g. Web pages). The experience of building Google's Knowledge Vault (Dong et al., 2014) shows that assessing confidence scores is essential for acquiring and fusing knowledge from different extractors.

It is worth nothing that the negative evidence can be also employed to estimate confidence scores. A recent approach (Wang and He, 2016) uses statistics of both hypernym and co-hyponym patterns to give each pair a positive score and a negative score. Experiments show that using negative scores improves precision by discarding co-hyponym relations that are incorrectly predicted as is-a relations by their model.

Classification-based Validation These methods train a classifier f to predict the correctness of an extracted pair (x, y). Models of choice typically include SVM, logistic regression and neural nets. The features for f can be roughly divided into following categories: surface name, syntax, statistics, external resources, etc. In the literature, Snow et al. (2004, 2006) use the dependency paths between x and y as features in the corresponding lexico-syntactic patterns. Ritter et al. (2009) introduce a list of features based the frequency of matches between a pair and Hearst patterns, such as the number of matches for "x is a y" in a corpus. Surface name features (Bansal et al., 2014) consider the word formation of x and y, including whether x and y are capitalized, whether x ends with y. Bansal et al. (2014) further employ statistics derived from matches of Hearst patterns in the corpus and Wikipedia abstracts. This is because abstracts in Wikipedia contain definitions and summaries about concepts that can be used for inferring is-a relations.

Using both pattern-based and distributional representations of x and y can also enhance the performance of the classifier, as shown in Shwartz et al. (2016). This technique can be viewed as a combination of pattern-based and distributional methods, which will be discussed in details in Sect. 2.2.4.

2.2 Distributional Methods

Distributional methods predict is-a relations between terms based on their distributional representations, by either *unsupervised measures* (Sect. 2.2.2) or *supervised models* (Sect. 2.2.3). Because they directly *predict* is-a relations instead of *extracting* all is-a relations in a corpus, we briefly introduce how to obtain key terms to form term pairs as candidate is-a relations (Sect. 2.2.1).

2.2.1 Key Term Extraction

The first step for predicting is-a relations is to generate candidate hyponyms/hypernyms. For free texts, they are usually *key terms*, which are nouns, noun phases and/or named entities that frequently appear in the corpus. The key terms can be identified by applying POS tagging or NER tools to the corpora and then using rule-based extractors (Yang, 2012; Zhu et al., 2013; Luu et al., 2014). Existing keyword or key phrase extractors can be also used to recognize these terms automatically (Navigli et al., 2011; Qureshi et al., 2012; da Silva Conrado et al., 2013; Liu et al., 2015).

For learning domain-specific taxonomies, an important post-processing step after extracting key terms is *domain filtering*. This filters out terms not in the domain of interest, improving the taxonomy precision. A term can be filtered by statistics-based cuts, which include TF, TF-IDF, domain relevance, domain consensus (Navigli and Velardi, 2004; de Knijff et al., 2013) and domain specificity scores (Alfarone and Davis, 2015). To ensure the extracted terms are important concepts in a particular domain, several methods only harvest terms from domain definitive sentences (Navigli et al., 2011; Velardi et al., 2013; Anke et al., 2016b). Specially, Navigli et al. (2011) propose to use domain weights to select sentences that define unambiguous terms pertained to the domain of interest.

2.2.2 Unsupervised Measures

We first survey various unsupervised measures for is-a relation identification. After that, feature representations are introduced for these measures.

Distributional Similarity Measures Early work of *distributional similarity measures* mostly focuses on *symmetric* measures such as cosine, Jaccard, Jensen-Shannon divergence and the widely cited LIN measure (Lin, 1998):

$$\text{LIN}(x,y) = \frac{\sum_{f \in F_x \cap F_y} w_x(f) + w_y(f)}{\sum_{f \in F_x} w_x(f) + \sum_{f \in F_y} w_y(f)}$$

where x and y are candidate hyponyms and hypernyms respectively. F_x and F_y are features of x and y. $w_x(f)$ is the weight of feature f for word x. But they only learn semantic similarity of words.

Asymmetric measures model the *asymmetric* property of is-a relations, following the *Distributional Inclusion Hypothesis* (DIH) (Geffet and Dagan, 2005; Zhitomirsky-Geffet and Dagan, 2009). It assumes that a hyponym only appears in some of its hypernym's contexts, but a hypernym appears in all contexts of its hyponyms. For example, the concept "fruit" has a broader spectrum of contexts than its hyponyms, such as "apple", "banana" and "pear". As an example, Weeds et al. (2004) propose a simple measure *WeedsPrec* to compute the weighted inclusion of features of y within features of x:

$$\text{WeedsPrec}(x,y) = \frac{\sum_{f \in F_x \cap F_y} w_y(f)}{\sum_{f \in F_y} w_y(f)}$$

Other asymmetric measures are introduced in a variety of research, e.g., WeedsRec (Weeds et al., 2004), BalAPInc (Kotlerman et al., 2010), ClarkeDE (Clarke, 2009), cosWeeds, invCL (Lenci and Benotto, 2012), WeightedCosine (Rei and Briscoe, 2014). Detailed summarization of distributional similarity measures can be found in an early survey on vector space semantic models (Turney and Pantel, 2010).

More recently, several studies suggest that DIH is not correct for all the cases (Santus et al., 2014; Rimell, 2014). For example, "American" is a hypernym of "Barack Obama" but the (politics-related) contexts of "Barack Obama" cannot be covered by those of "American". Most contexts of a hypernym are less informative and more general than those of its hyponyms. To solve this problem, Santus et al. (2014) propose an entropy-based measure SLQS for hypernym detection. Roller et al. (2014) introduce the *Selective Distributional Inclusion Hypothesis*, which means the original DIH is correct, but only for relevant dimensions.

Features For each term x, a collection of features F_x are generated from the text corpus, where each feature $f \in F_x$ represents a contextual word with which x co-occurs (Lin, 1998; Weeds et al., 2004). In some work, f also specifies the syntactic relation between x and f (Lin, 1998). As stated in Padó and Lapata (2003), the usage of syntactic-based vector space model can better distinguish different lexical relations than the simple "Bag-of-Words" co-occurrence model. In addition, Schütze (1993) use the context word and the position relative to the target term as features. Baroni and Lenci (2010) propose a Distributional Memory framework to generate word-link-word features. Yamada et al. (2009) use raw verb-noun dependencies and cluster such dependencies to generate feature vectors

The value of each feature is determined by a weight function $w_x(f)$, which quantifies the statistical association between the feature f and the corresponding word x. Choices of $w_x(f)$ include the (point-wise) Mutual Information (PPMI) (Weeds et al., 2004), Local Mutual Information (LMI) (Evert, 2005). Dimension reduction methods such as SVD can be employed to create dense vectors (Roller and Erk, 2016).

2.2.3 Supervised Models

With training sets available, *classification/ranking methods* train a model to predict hypernymy based on the representations of a term pair (x, y). *Hypernym generation* approaches directly model how to "generate" hypernyms based on the representations of hyponyms in the embedding space.

Classification In classification methods, the most popular representations for x and y are word embeddings generated by pre-trained neural language models such as Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and ivLBL (Mnih and Kavukcuoglu, 2013). SensEmbed (Iacobacci et al., 2015) generates different embeddings for different senses of the same word.

The *concat* model combines term-pair vectors by $\vec{x} \oplus \vec{y}$ where $\vec{x}$ is the embedding vector of word x, then trains an off-the-shelf classifier such as SVM (Baroni et al., 2012). This model is regarded as a strong baseline in some papers (Kruszewski et al., 2015; Shwartz et al., 2016; Mirza and Tonelli, 2016). Recent work points out that it has a serious problem of *lexical memorization* (Roller et al., 2014; Levy et al., 2015; Weeds et al., 2014). It means that the classifier learns the semantics of terms rather than the relations between the terms.

Consequently, when the training sets and testing sets are significantly different, the model suffers from a poor performance.

To over this problem, the *diff* model uses vector offsets as features, represented as $\vec{y} - \vec{x}$ (Rimell, 2014; Weeds et al., 2014; Fu et al., 2014). The *asym* model is presented in Roller et al. (2014), using both vector difference and squared vector difference features. The *simDiff* model (Turney and Mohammad, 2015) employs the difference of two word-context matrices (i.e., domain matrix and function matrix) as features for relation classification. Additionally, other combinations of vectors are mentioned in the literature, such as vector sum $\vec{x} + \vec{y}$, and dot-product $\vec{x} \cdot \vec{y}$ (Shwartz et al., 2016). Roller and Erk (2016) exploit Hearst patterns in distributional vectors and introduce a PCA-like iterative procedure to learn *concat* classifiers. Kruszewski et al. (2015) learn mappings from distributional vectors to boolean-valued vectors, where the output vectors correspond to entailment between words.

In neural language models (Mikolov et al., 2013; Pennington et al., 2014), words that occur in similar contexts have similar embeddings. Yu et al. (2015) argue that this modeling technique is not strong enough to learn term embeddings for is-a relation prediction. For each word x, they learn two types of embeddings $\vec{x}_o$ and $\vec{x}_e$, representing the embeddings of x when x functions as a hyponym and a hypernym, respectively. The embeddings are generated by training a distance-margin based neural net. Luu et al. (2016b) further extend this approach by modeling the contexts between hypernyms and hyponyms in a dynamic weighting neural net. Li et al. (2016) design a joint model based on negative sampling to embed entities and categories jointly into the same semantic space. The high performance shows that using task-specific embeddings is more effective than general-purpose embeddings.

Hypernym Generation *Hypernym generation* approaches make prediction for a pair (x, y) by whether the model can map $\vec{x}$ to a vector close to $\vec{y}$. Fu et al. (2014) is a pioneer work in this field, which employs uniform linear projection and piecewise linear projection to map the embeddings of a hyponym to its hypernym. After that, three approaches (Wang and He, 2016; Yamane et al., 2016; Tan et al., 2015) have been proposed to extend Fu et al. (2014). Wang and He (2016) update transition matrices and extract new is-a relations iteratively. They improve the performance of the piecewise projection model when training sets and test sets have little overlap in the semantic space. Yamane et al. (2016) learn the number of clusters and transition matrices jointly by dynamically clustering is-a pairs. Tan et al. (2015) replace the transition matrix with the embedding of “is-a”. As shown in Yamane et al. (2016), these methods are comparable to state-of-the-art classification approaches in terms of F-measure. Additionally, by domain clustering, the approach (Fu et al., 2014) is modified to a transfer learning version that is sensitive to target data sources for domain adaptation (Anke et al., 2016a).

The negative sampling technique proves effective to enhance projection learning. This is because the representations of hypernymy relations are sometimes confused with synonymy, co-hyponymy and meronymy. In Ustalov et al. (2017), an additional regularization term is added to the model of Fu et al. (2014) to take the advantage of the semantics of not-is-a relations. Wang et al. (2017) explicitly learn the representations of not-is-a relations so that the true hypernymy relations are better distinguished. This method considers the representations of both is-a and not-is-a relations, hypernym-level similarity and linguistic rules in a transductive learning setting.

Ranking As an alternative approach, Fu et al. (2013) present a ranking model to select the most probable hypernym for an entity. Replacing a classification model with a ranking model is not a common practice for extracting is-a relations, due to its low recall. However, this method is specifically engineered for the Chinese language. As shown in (Fu et al., 2014; Wang et al., 2015; Li et al., 2015; Wang and He, 2016), learning Chinese is-a relations is fundamentally challenging due to flexible language expressions. Thus it is necessary to train a ranking model to extract Chinese is-a relations with high precision.

2.2.4 Discussion

In the literature, there are some disagreements on which methods are more effective for is-a relation prediction. For example, Shwartz et al. (2016) claim that distributional methods outperform pattern-based approaches, while Levy et al. (2015) hold the opinion that distributional methods do not even work. We overview major viewpoints in the research community and analyze pros and cons for both types of methods.

Pattern-based methods extract is-a relations (x, y) based on the lexico-syntactic paths connecting x and y in a corpus, which explicitly express the relation. The original Hearst patterns (Hearst, 1992) and more generalized patterns have been used in a large number of taxonomies (Wu et al., 2012; Navigli et al., 2011). A disadvantage is that using patterns as features may result in the sparsity of the feature space (Nakashole et al., 2012). Most methods require x and y to co-occur in the same sentence. Hence, the recall is limited. Besides, they are overly language-dependent and difficult to use if there are few Hearst-like patterns in other languages. For example, as shown in Fu et al. (2014); Wang and He (2016), they suffer from extremely low recall for the Chinese language.

In contrast, distributional approaches use word representations derived from contexts, independent of its hyponym or hypernym. The usage of word embeddings (Mikolov et al., 2013) allows machines to make predictions based on the entire corpus. However, distributional methods are less precise in detecting specific, strict is-a relations and tend to discover broad semantic similarity between terms (Shwartz et al., 2016; Yu et al., 2015). As Weeds et al. (2014) discover, some terms detected by distributional methods are co-hyponyms and meronyms, rather than hypernyms. Another drawback is that the representations are domain dependent and the models are heavily related to the training set (Roller et al., 2014). Yet a further criticism is pointed out by Levy et al. (2015). They find that supervised distributional methods actually learn whether y is a "prototypical hypernym", instead of the relation between x and y. This is because the features $\vec{x}$ and $\vec{y}$ are generated independently. They integrate the intra-pair similarity with the *diff* model by kernel functions but only achieves an incremental improvement.

Despite their own disadvantages, pattern-based and distributional methods have been considered complementary. The idea of integrating them has been proposed early (Mirkin et al., 2006; Kaji and Kitsuregawa, 2008) but have not drawn much attention over the years. Recently, the HyperNET system (Shwartz et al., 2016) represents a pair (x, y) by both distributional and pattern-based features. Each pattern is represented by a dependency path, and embedded by an LSTM model (Hochreiter and Schmidhuber, 1997). Experiments show that the joint representation improves the performance notably, having F1-scores of 0.901 and 0.700 over two large datasets. In contrast, the best pattern-based method (i.e., an extension of Snow et al. (2004)) has the performance of 0.761 and 0.660. The best distributional approach based on the *concat* model has the performance of 0.746 and 0.637. An extension of the previous model named LexNET (Shwartz and Dagan, 2016) is proposed to recognize multiple relations.

2.3 Taxonomy Induction

In this part, we summarize techniques for creating taxonomies from is-a relations.

Incremental Learning Several methods construct an entire taxonomy from a "seed" taxonomy via incremental learning. Snow et al. (2006) enrich WordNet by maximizing the probability of an extended taxonomy based on the evidence of is-a and cousin relations harvested from texts. They focus on extracting new entities and attaching them to the semantic hierarchy of WordNet. Shen et al. (2012) observe that the extracted terms can either refer to existing entities in the taxonomy or new ones, and propose a graph-based method to link these terms to the taxonomy or insert new entities into the taxonomy. While these methods rely heavily on existing taxonomies, Kozareva and Hovy (2010) take a root concept as input only and iteratively extract is-a relations to expand the taxonomy. Alfarone and Davis (2015) further consider the problem that a "seed" taxonomy cannot be obtained in a specific domain. They construct the "seed" taxonomy by Hearst pattern matching and heuristic rules.

Clustering Taxonomy learning can be modeled as a clustering problem where similar terms clustered together may share the same hypernym. Hierarchical clustering is employed to cluster similar terms into a taxonomy (Hjelm and Buitelaar, 2008; de Knijff et al., 2013; Meijer et al., 2014). Song et al. (2015) improve the hierarchical clustering technique by scalable Bayesian Rose Trees. A similar idea is also introduced in Alfarone and Davis (2015) where the lowest common ancestor of a collection of terms clustered by K-Medoids is inferred as their common hypernym. The SCS method (Luu et al., 2014) (see Sect. 2.1.1) is also related to clustering because it groups similar terms by non-taxonomic relations, and infer its hypernyms to improve the taxonomy coverage.

Graph-based Induction Graph-based approaches are naturally suitable for this task because taxonomies are generally graphs. Kozareva and Hovy (2010) derive the path from the root to a target term by finding the longest path in a raw graph where edges represent noisy is-a relations. Anke et al. (2016b) calculate the path weights by multiplying the domain pertinence scores of its edges. Another frequently used algorithm is the *optimal branching algorithm* (Velardi et al., 2013; Luu et al., 2014). It first assigns edge weights based on graph connectivity (e.g., in-degree, betweenness, etc.), and finds an optimal branching based on *Chu-Liu/Edmonds's algorithm* (Karp, 1971). After noisy edge removal, a rooted tree is constructed with maximum weights. Bansal et al. (2014) employ a factor graph model to represent terms and is-a relations. The learning of a taxonomy is regarded as a structured learning problem for the model, solved by loopy belief propagation.

Taxonomy Cleansing The final step of taxonomy learning is *taxonomy cleansing*, which removes wrong is-a relations to improve the quality. A recent study on Probase (Wu et al., 2012) shows that incorrect is-a relations may exist in taxonomies in the form of cycles (Liang et al., 2017a). By eliminating cycles, 74K wrong is-a relations are detected. This cycle removal process is also applied in Deshpande et al. (2013); Fu et al. (2014); Li et al. (2015).

Another issue is entity ambiguity. As discussed in Liang et al. (2017b), the transitivity property does not necessarily hold in automatically constructed taxonomies. For example, the facts "(Albert Einstein, is-a, professor)" and "(professor, is-a, position)" do not mean that "(Albert Einstein, is-a, position)". The ambiguity issue has been addressed in a few systems (Anke et al., 2016b; Wu et al., 2012; Ponzetto and Navigli, 2009) by word sense disambiguation. However, it is not fully solved. While learning multiple senses of the word "bank" (a financial institution or riverside) is easy nowadays, it is more challenging to distinguish whether the word "professor" refers to a particular person or a job title in the taxonomy learning process. Based on Liang et al. (2017b), we can safely conclude that there is a long way towards learning a fully-disambiguated taxonomy.

Contributor/Paper	#Positive	#Negative
Kotlerman et al. (2010)	1,068	2,704
Baroni and Lenci (2011)	1,337	13,210
Baroni et al. (2012)	1,385	1,385
Jurgens et al. (2012)	1,154	1,154
Levy et al. (2014)	945	11,657
Rei and Briscoe (2014)	3,074	-
Weeds et al. (2014)	2,564	3,771
Turney and Mohammad (2015)	920	772
Shwartz et al. (2016) (Lex)	5,659	22,636
Shwartz et al. (2016) (Rnd)	14,135	56,544

Table 1: Test sets for is-a relation prediction.

3 Resources and Analysis

In this section, we summarize resources and metrics for taxonomy learning. Results and recommendations for future research are also discussed.

3.1 Resources

There have been a variety of resources for the research of is-a relation prediction. The first type is high-quality taxonomies. knowledge bases and semantic networks. The knowledge in these systems can be used for generating training sets for distant supervised model learning. Typical English resources include WordNet (Miller, 1995), YAGO (Suchanek et al., 2007), WiBi (Flati et al., 2014) and DefIE (Bovi et al., 2015). For languages other than English, refer to multilingual systems (e.g., YAGO3 (Mahdisoltani et al., 2015), BabelNet (Navigli and Ponzetto, 2012), Multi-WiBi (Flati et al., 2016)). We need to point out that these systems are not necessarily all taxonomies but contain rich type hierarchical knowledge. We also summarize some recent (2010$\sim$) test sets and statistics in Table 1[1].

Two shared tasks are designed specifically for taxonomy learning, i.e., TExEval (SemEval-2015 Task 17) (Bordea et al., 2015) and TExEval-2 (SemEval-2016 Task 13) (Bordea et al., 2016)[2]. In TExEval, the goal is to construct taxonomies in four target domains (i.e. chemicals, equipment, food and science), each with gold-standard provided. The setting is expanded to cover four European languages (i.e., English, Dutch, French and

[1] Statistics combine training, validation and test sets. Papers that use subsets of these datasets are not listed. Dataset (Jurgens et al., 2012) is not directly capable of evaluating the task and is processed by Turney and Mohammad (2015).

[2] Due to the relatively large number of submissions, we do not provide citations to every system submitted to TExEval tracks. Readers can refer to the two reports for details.

Italian) in TExEval-2. In this task, the participants are encouraged to use the Wikipedia corpus as input but there are no restrictions on the data sources. In previous studies, several domain-specific corpora have also been employed as inputs for taxonomies, including AI papers (Velardi et al., 2013), biomedical corpora (Alfarone and Davis, 2015), Web pages related to animals, plants and vehicles (Kozareva et al., 2008) and MH370 (Luu et al., 2016a), terrorism reports (Luu et al., 2014), disease reports and emails (Luu et al., 2016a) and Wikipedia corpora related to specific topics.

3.2 Evaluation Metrics

Evaluating hypernymy prediction algorithms is by no means easy. Given a collection of is-a and not-is-a term pairs as ground truth, standard relation classification metrics such as Precision (P), Recall (R) and F-score (F) can be employed to make the comparison (Shwartz et al., 2016; Yu et al., 2015).

However, evaluating the quality of an entire taxonomy is a non-trivial task due to i) the large size of a taxonomy, ii) the difficulty of obtaining gold standard and iii) the existence of multiple aspects that should be considered such as topology, correctness and coverage. If gold standard taxonomies are available, denote $S = (V_S, E_S)$ and $G = (V_G, E_G)$ as the extracted and gold standard taxonomies where V_S and V_G are node sets, E_S and E_G are edge sets. Evaluation metrics introduced in the two shared tasks (Bordea et al., 2015, 2016) are briefly summarized as follows:

- Node coverage: $|V_S \cap V_G|$, $|V_S \cap V_G|/|V_G|$;
- Edge coverage: $|E_S \cap E_G|$, $|E_S \cap E_G|/|E_G|$, $(|E_S| - |E_S \cap E_G|)/|E_G|$;
- Edge correctness: $P = |E_S \cap E_G|/|E_S|$, $R = |E_S \cap E_G|/|E_G|$, $F = 2(P \cdot R)/(P + R)$;
- Cumulative Fowlkes&Mallows (Cumulative F&M) measure (Velardi et al., 2013).

The second type of metrics compares taxonomies generated by different methods. Size ($|V_S|$ and $|E_S|$) and quality are two factors to be considered. Human assessment is required to estimate the accuracy by sampling and labeling edges. Additionally, topological statistics, including the numbers of simple directed cycles, connected components and intermediate nodes, can check whether the taxonomy is a Direct Acyclic Graph (DAG) and well-structured.

3.3 Result Analysis and Discussion

While we have discussed is-a relation prediction in Sect. 2.2.4, we mostly focus on the overall performance for taxonomy learning in this part.

We first analyze results of the two shared tasks (Bordea et al., 2015, 2016) as they report performance of a variety of methods. In both task, two pattern-based methods (i.e., INRIASAC (Grefenstette, 2015) in TExEval and TAXI (Panchenko et al., 2016) in TExEval-2) consistently outperform others. INRIASAC uses frequency-based co-occurrence statistics, and substring inclusion heuristics to extract a set of hypernyms for hyponyms. TAXI crawls a domain-specific corpora and employs lexico-syntactic patterns and substrings for domain is-a relation extraction. However, the potential of distributional methods is not fully exploited as only one system uses such techniques. Besides, different systems may use their own corpora in these tasks and hence the results do not directly reflect the "goodness" of these algorithms. In multilingual tasks, there is a large decrease in performance w.r.t. other languages in TAXI. The research (Fu et al., 2014; Wang and He, 2016) shares a similar experience when several effective algorithms for English do not really work for Chinese. This phenomenon calls for language-specific algorithms for non-English language sources.

For other works, although knowledge sources and domains may differ, we notice that they suffer from the *low recall* problem. For example, recall values of Bansal et al. (2014); Luu et al. (2014); Navigli et al. (2011); Kozareva and Hovy (2010) are lower than 50% in most cases and domains. While improving precision is relatively easy by imposing constraints, increasing recall is more challenging because we aim to identify all is-a relations, no matter whether the relations are expressed explicitly or implicitly, in one or multiple sentences (Shwartz et al., 2016). This problem becomes severe when less focused and dynamic domains are considered (Velardi et al., 2013).

3.4 Our Recommendations

Based on our analysis, we discuss our recommendations to improve the performance of taxonomy learning that have not been sufficiently addressed.

Ensemble Representations and Deep Architectures Shwartz et al. (2016) show that combining pattern-based and distributional methods can improve the performance of is-a relation extraction. We suggest that the performance can be further improved by studying how the two types of approaches reinforce each other. Neural network models (Yu et al., 2015; Luu et al., 2016b) are effective to learn deeper representations of both features. In our opinion, It is also possible to solve the problem put forward by Levy et al. (2015) by adding the information of semantic relatedness between term pairs mined from patterns to distributed representations.

Another related topic is that despite several embedding learning methods mentioned above, there is only limited success of deep learning paradigms for the taxonomy induction. We believe this is mostly because it is difficult to design a single objective for neural networks to optimize for this task. Hence how to take advantage of the deep learning boom for taxonomy induction is worth researching in the future.

Benchmarks and Evaluation Benchmarks for taxonomy learning are crucial to quantify the performance of state-of-the-arts. Benchmarks should contain text corpora, gold standards and evaluation metrics. Bordea et al. (2015, 2016) have provided some gold standard taxonomies in several domains and languages but do not require all the systems to run over the same corpus. Other works use standard test sets and data sources, as we have summarized in Sect. 3.1.

Several issues in current benchmarks and methods have already been pointed out by previous works. Levy et al. (2015) show that supervised systems actually over-perform due to the lexical memorization problem. Shwartz et al. (2017) suggest that unsupervised approaches are more robust than supervised methods but supervised methods outperform unsupervised ones. Camacho-Collados (2017) discuss whether the hypernymy detection task is indeed an appropriate task for evaluating is-a relations in the context of taxonomy learning systems or their integration in downstream applications. We can see that more in-depth research should be done towards a complete, widely-accepted benchmark for evaluation.

Unambiguous and Canonicalized Terms For lexical taxonomies, a term may have multiple surface forms and senses. The ambiguity issue makes taxonomy-based applications prone to error (Liang et al., 2017b). It is desirable to construct taxonomies where each node represents an unambiguous term associated with its possible surface forms and their contexts. In this way, the taxonomy automatically supports entity linking and is beneficial for IR applications (e.g., Web search) (Shen et al., 2015; Hua et al., 2017).

Incorporating Domain Knowledge Domain knowledge is essential for term and relation extraction in domain-specific corpora but it is difficult to obtain from such limited corpora. By exploiting facts derived from domain knowledge bases, a domain taxonomy would be learned via distant supervision and have higher coverage than existing methods (Alfarone and Davis, 2015). Thus, it is an important task to construct a taxonomy based on a text corpus and a knowledge base of a specific domain.

Non-English and Under-resourced Languages The task addressed in this paper has not been extensively studied for under-resourced languages. Specifically, pattern-based methods, although effective for English, are language-dependent to a large extent. How to apply existing approaches to languages that are significantly different from English (e.g., Chinese, Arabic and Japanese) is worthy of research.

4 Conclusion

In this paper, we present a survey on taxonomy learning from text corpora. We overview pattern-based and distributional methods to learn hypernymy from texts, and discuss how to induce taxonomies from is-a relations. While there is significant success, this task is still far from being solved. By addressing the issues discussed in this paper, we suggest that high-quality taxonomies can be constructed in more domains and languages, having a greater influence in NLP and IR research.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. This work is partially supported by the National Key Research and Development Program of China under Grant No. 2016YFB1000904. Chengyu Wang would also like to thank the ECNU Outstanding Doctoral Dissertation Cultivation Plan of Action under Grant No. YB2016040 for the support of his research.

References

Steven S. Aanen, Damir Vandic, and Flavius Frasincar. 2015. Automated product taxonomy mapping in an e-commerce environment. *Expert Syst. Appl.* 42(3):1298–1313.

Daniele Alfarone and Jesse Davis. 2015. Unsupervised learning of an IS-A taxonomy from a limited domain-specific corpus. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*. pages 1434–1441.

Luis Espinosa Anke, José Camacho-Collados, Claudio Delli Bovi, and Horacio Saggion. 2016a. Supervised distributional hypernym discovery via domain adaptation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 424–435.

Luis Espinosa Anke, Horacio Saggion, Francesco Ronzano, and Roberto Navigli. 2016b. Extasem! extending, taxonomizing and semantifying domain terminologies. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. pages 2594–2600.

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. pages 2670–2676.

Mohit Bansal, David Burkett, Gerard de Melo, and Dan Klein. 2014. Structured learning for taxonomy induction with belief propagation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. pages 1041–1051.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. pages 23–32.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics* 36(4):673–721.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. page 110.

Georgeta Bordea, Paul Buitelaar, Stefano Faralli, and Roberto Navigli. 2015. Semeval-2015 task 17: Taxonomy extraction evaluation (texeval). In *Proceedings of the 9th International Workshop on Semantic Evaluation*. pages 902–910.

Georgeta Bordea, Els Lefever, and Paul Buitelaar. 2016. Semeval-2016 task 13: Taxonomy extraction evaluation (texeval-2). In *Proceedings of the 10th International Workshop on Semantic Evaluation*. pages 1081–1091.

Claudio Delli Bovi, Luca Telesca, and Roberto Navigli. 2015. Large-scale information extraction from textual definitions through deep syntactic and semantic analysis. *TACL* 3:529–543.

José Camacho-Collados. 2017. Why we have switched from building full-fledged taxonomies to simply detecting hypernymy relations. *CoRR* abs/1703.04178.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third International Conference on Web Search and Web Data Mining*. pages 101–110.

Daoud Clarke. 2009. Context-theoretic semantics for natural language: an overview. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*. page 112119.

Merley da Silva Conrado, Thiago Alexandre Salgueiro Pardo, and Solange Oliveira Rezende. 2013. A machine learning approach to automatic term extraction using a rich feature set. In *Proceedings of the 2013 Conference of the North American Chapter of the Association of Computational Linguistics*. pages 16–23.

Jeroen de Knijff, Flavius Frasincar, and Frederik Hogenboom. 2013. Domain taxonomy learning from text: The subsumption method versus hierarchical clustering. *Data Knowl. Eng.* 83:54–69.

Omkar Deshpande, Digvijay S. Lamba, Michel Tourn, Sanjib Das, Sri Subramaniam, Anand Rajaraman, Venky Harinarayan, and AnHai Doan. 2013. Building, maintaining, and using knowledge bases: a report from the trenches. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. pages 1209–1220.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pages 601–610.

Oren Etzioni, Michael J. Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th international conference on World Wide Web*. pages 100–110.

Stefan Evert. 2005. *The statistics of word cooccurrences: word pairs and collocations*. Ph.D. thesis, University of Stuttgart.

Tiziano Flati, Daniele Vannella, Tommaso Pasini, and Roberto Navigli. 2014. Two is bigger (and better) than one: the wikipedia bitaxonomy project. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. pages 945–955.

Tiziano Flati, Daniele Vannella, Tommaso Pasini, and Roberto Navigli. 2016. Multiwibi: The multilingual wikipedia bitaxonomy project. *Artif. Intell.* 241:66–102.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. pages 1199–1209.

Ruiji Fu, Bing Qin, and Ting Liu. 2013. Exploiting multiple sources for open-domain hypernym discovery. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 1224–1234.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*. pages 107–114.

Gregory Grefenstette. 2015. INRIASAC: simple hypernym extraction methods. In *Proceedings of the 9th International Workshop on Semantic Evaluation*. pages 911–914.

Amit Gupta, Francesco Piccinno, Mikhail Kozhevnikov, Marius Pasca, and Daniele Pighin. 2016. Revisiting taxonomy induction over wikipedia. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 2300–2309.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*. pages 539–545.

Hans Hjelm and Paul Buitelaar. 2008. Multilingual evidence improves clustering-based taxonomy extraction. In *Proceedings of the 18th European Conference on Artificial Intelligence*. pages 288–292.

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

Wen Hua, Zhongyuan Wang, Haixun Wang, Kai Zheng, and Xiaofang Zhou. 2017. Understand short texts by harvesting and analyzing semantic knowledge. *IEEE Trans. Knowl. Data Eng.* 29(3):499–512.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*. pages 95–105.

David Jurgens, Saif Mohammad, Peter D. Turney, and Keith J. Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation*. pages 356–364.

Nobuhiro Kaji and Masaru Kitsuregawa. 2008. Using hidden markov random fields to combine distributional and pattern-based word clustering. In *Proceedings of the 22nd International Conference on Computational Linguistics*. pages 401–408.

Richard M. Karp. 1971. A simple derivation of edmonds' algorithm for optimum branchings. *Networks* 1(3):265–272.

Sebastian Köhler, Sandra C. Doelken, Christopher J. Mungall, Sebastian Bauer, Helen V. Firth, Isabelle Bailleul-Forestier, Graeme C. M. Black, Danielle L. Brown, Michael Brudno, Jennifer Campbell, David R. FitzPatrick, Janan T. Eppig, Andrew P. Jackson, Kathleen Freson, Marta Gîrdea, Ingo Helbig, Jane A. Hurst, Johanna Jähn, Laird G. Jackson, Anne M. Kelly, David H. Ledbetter, Sahar Mansour, Christa L. Martin, Celia Moss, Andrew Mumford, Willem Ouwehand, Soo-Mi Park, Erin Rooney Riggs, Richard H. Scott, Sanjay Sisodiya, Steven Van Vooren, Ronald J. Wapner, Andrew O. M. Wilkie, Caroline F. Wright, Anneke T. Vulto-van Silfhout, Nicole de Leeuw, Bert B. A. de Vries, Nicole L. Washington, Cynthia L. Smith, Monte Westerfield, Paul N. Schofield, Barbara J. Ruef, Georgios V. Gkoutos, Melissa Haendel, Damian Smedley, Suzanna E. Lewis, and Peter N. Robinson. 2014. The human phenotype ontology project: linking molecular biology and disease through phenotype data. *Nucleic Acids Research* 42(Database-Issue):966–974.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering* 16(4):359–389.

Zornitsa Kozareva and Eduard H. Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. pages 1110–1118.

Zornitsa Kozareva, Ellen Riloff, and Eduard H. Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*. pages 1048–1056.

Germán Kruszewski, Denis Paperno, and Marco Baroni. 2015. Deriving boolean structures from distributional vectors. *TACL* 3:375–388.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*. pages 75–79.

Omer Levy, Ido Dagan, and Jacob Goldberger. 2014. Focused entailment graphs for open IE propositions. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. pages 87–97.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 970–976.

Jinyang Li, Chengyu Wang, Xiaofeng He, Rong Zhang, and Ming Gao. 2015. User generated content oriented chinese taxonomy construction. In *Proceedings of the 17th Asia-Pacific Web Conference*. pages 623–634.

Yuezhang Li, Ronghuo Zheng, Tian Tian, Zhiting Hu, Rahul Iyer, and Katia P. Sycara. 2016. Joint embedding of hierarchical categories and entities for concept categorization and dataless classification. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 2678–2688.

Jiaqing Liang, Yanghua Xiao, Yi Zhang, Seung-won Hwang, and Haixun Wang. 2017a. Graph-based wrong isa relation detection in a large-scale lexical taxonomy. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. pages 1178–1184.

Jiaqing Liang, Yi Zhang, Yanghua Xiao, Haixun Wang, Wei Wang, and Pinpin Zhu. 2017b. On the transitivity of hypernym-hyponym relations in data-driven lexical taxonomies. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. pages 1185–1191.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*. pages 296–304.

Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren, and Jiawei Han. 2015. Mining quality phrases from massive text corpora. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. pages 1729–1744.

Anh Tuan Luu, Siu Cheung Hui, and See-Kiong Ng. 2016a. Utilizing temporal information for taxonomy construction. *TACL* 4:551–564.

Anh Tuan Luu, Jung-jae Kim, and See-Kiong Ng. 2014. Taxonomy construction using syntactic contextual evidence. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pages 810–819.

Anh Tuan Luu, Jung-jae Kim, and See-Kiong Ng. 2015. Incorporating trustiness and collective synonym/contrastive evidence into taxonomy construction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1013–1022.

Anh Tuan Luu, Yi Tay, Siu Cheung Hui, and See-Kiong Ng. 2016b. Learning term embeddings for taxonomic relation identification using dynamic weighting neural network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 403–413.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. 2015. YAGO3: A knowledge base from multilingual wikipedias. In *Proceedings of the Seventh Biennial Conference on Innovative Data Systems Research*.

Kevin Meijer, Flavius Frasincar, and Frederik Hogenboom. 2014. A semantic approach for extracting domain taxonomies from text. *Decision Support Systems* 62:78–93.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*. pages 3111–3119.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM* 38(11):39–41.

Shachar Mirkin, Ido Dagan, and Maayan Geffet. 2006. Integrating pattern-based and distributional similarity methods for lexical entailment acquisition. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*.

Paramita Mirza and Sara Tonelli. 2016. On the contribution of word embeddings to temporal relation classification. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 2818–2828.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*. pages 2265–2273.

Ndapandula Nakashole, Gerhard Weikum, and Fabian M. Suchanek. 2012. PATTY: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 1135–1145.

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

Roberto Navigli and Paola Velardi. 2004. Learning domain ontologies from document warehouses and dedicated web sites. *Computational Linguistics* 30(2):151–179.

Roberto Navigli and Paola Velardi. 2010. Learning word-class lattices for definition and hypernym extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. pages 1318–1327.

Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. pages 1872–1877.

Robert C. Nickerson, Upkar Varshney, and Jan Muntermann. 2013. A method for taxonomy development and its application in information systems. *EJIS* 22(3):336–359.

Sebastian Padó and Mirella Lapata. 2003. Constructing semantic space models from parsed corpora. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. pages 128–135.

Alexander Panchenko, Stefano Faralli, Eugen Ruppert, Steffen Remus, Hubert Naets, Cédrick Fairon, Simone Paolo Ponzetto, and Chris Biemann. 2016. TAXI at semeval-2016 task 13: a taxonomy induction method based on lexico-syntactic patterns, substrings and focused crawling. In *Proceedings of the 10th International Workshop on Semantic Evaluation*. pages 1320–1327.

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

Simone Paolo Ponzetto and Roberto Navigli. 2009. Large-scale taxonomy mapping for restructuring and integrating wikipedia. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*. pages 2083–2088.

Muhammad Atif Qureshi, Colm O'Riordan, and Gabriella Pasi. 2012. Short-text domain specific key terms/phrases extraction using an n-gram model with wikipedia. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. pages 2515–2518.

Marek Rei and Ted Briscoe. 2014. Looking for hyponyms in vector space. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. pages 68–77.

Laura Rimell. 2014. Distributional lexical entailment by topic coherence. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. pages 511–519.

Alan Ritter, Stephen Soderland, and Oren Etzioni. 2009. What is this, anyway: Automatic hypernym discovery. In *Learning by Reading and Learning to Read, Proceedings of the 2009 AAAI Spring Symposium*. pages 88–93.

Stephen Roller and Katrin Erk. 2016. Relations such as hypernymy: Identifying and exploiting hearst patterns in distributional vectors for lexical entailment. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 2163–2172.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the 25th International Conference on Computational Linguistics*. pages 1025–1036.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. pages 38–42.

Hinrich Schütze. 1993. Part-of-speech induction from scratch. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*. pages 251–258.

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Trans. Knowl. Data Eng.* 27(2):443–460.

Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2012. A graph-based approach for ontology population with named entities. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. pages 345–354.

Vered Shwartz and Ido Dagan. 2016. The roles of path-based and distributional information in recognizing lexical semantic relations. *CoRR* abs/1608.05014.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. 2017. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. page 6575.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of the 17th Annual Conference on Neural Information Processing Systems*. pages 1297–1304.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*.

Yangqiu Song, Shixia Liu, Xueqing Liu, and Haixun Wang. 2015. Automatic taxonomy construction from keywords via scalable bayesian rose trees. *IEEE Trans. Knowl. Data Eng.* 27(7):1861–1874.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*. pages 697–706.

Liling Tan, Rohit Gupta, and Josef van Genabith. 2015. USAAR-WLV: hypernym generation with deep neural nets. In *Proceedings of the 9th International Workshop on Semantic Evaluation*. pages 932–937.

Peter D. Turney and Saif M. Mohammad. 2015. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering* 21(3):437–476.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Intell. Res. (JAIR)* 37:141–188.

Dmitry Ustalov, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. 2017. Negative sampling improves hypernymy extraction based on projection learning. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. page 543550.

Paola Velardi, Stefano Faralli, and Roberto Navigli. 2013. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics* 39(3):665–707.

Chengyu Wang, Ming Gao, Xiaofeng He, and Rong Zhang. 2015. Challenges in chinese knowledge graph construction. In *Proceedings of the 31st IEEE International Conference on Data Engineering Workshops*. pages 59–61.

Chengyu Wang and Xiaofeng He. 2016. Chinese hypernym-hyponym extraction from user generated categories. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 1350–1361.

Chengyu Wang, Junchi Yan, Aoying Zhou, and Xiaofeng He. 2017. Transductive non-linear learning for chinese hypernym prediction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David J. Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of the 25th International Conference on Computational Linguistics*. pages 2249–2259.

Julie Weeds, David J. Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Bifan Wei, Jun Liu, Jian Ma, Qinghua Zheng, Wei Zhang, and Boqin Feng. 2014. Motif-based hyponym relation extraction from wikipedia hyperlinks. *IEEE Trans. Knowl. Data Eng.* 26(10):2507–2519.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Qili Zhu. 2012. Probase: a probabilistic taxonomy for text understanding. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. pages 481–492.

Ichiro Yamada, Kentaro Torisawa, Jun'ichi Kazama, Kow Kuroda, Masaki Murata, Stijn De Saeger, Francis Bond, and Asuka Sumida. 2009. Hypernym discovery based on distributional similarity and hierarchical structures. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. pages 929–937.

Josuke Yamane, Tomoya Takatani, Hitoshi Yamada, Makoto Miwa, and Yutaka Sasaki. 2016. Distributional hypernym generation by jointly learning clusters and projections. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 1871–1879.

Hui Yang. 2012. Constructing task-specific taxonomies for document collection browsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 1278–1289.

Shuo Yang, Lei Zou, Zhongyuan Wang, Jun Yan, and Ji-Rong Wen. 2017. Efficiently answering technical questions - A knowledge graph approach. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. pages 3111–3118.

Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. 2015. Learning term embeddings for hypernymy identification. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*. pages 1390–1397.

Yuchen Zhang, Amr Ahmed, Vanja Josifovski, and Alexander J. Smola. 2014. Taxonomy discovery for personalized recommendation. In *Proceedings of the Seventh ACM International Conference on Web Search and Data Mining*. pages 243–252.

Maayan Zhitomirsky-Geffet and Ido Dagan. 2009. Bootstrapping distributional feature vector quality. *Computational Linguistics* 35(3):435–461.

Xingwei Zhu, Zhaoyan Ming, Xiaoyan Zhu, and Tat-Seng Chua. 2013. Topic hierarchy construction for the organization of multi-source user generated contents. In *Proceedings of the 36th International ACM SIGIR conference on research and development in Information Retrieval*. pages 233–242.

Idiom-Aware Compositional Distributed Semantics

Pengfei Liu Kaiyu Qian Xipeng Qiu[*] Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{pfliu14,kyqian, xpqiu,xjhuang}@fudan.edu.cn

Abstract

Idioms are peculiar linguistic constructions that impose great challenges for representing the semantics of language, especially in current prevailing end-to-end neural models, which assume that the semantics of a phrase or sentence can be literally composed from its constitutive words. In this paper, we propose an idiom-aware distributed semantic model to build representation of sentences on the basis of understanding their contained idioms. Our models are grounded in the literal-first psycholinguistic hypothesis, which can adaptively learn semantic compositionality of a phrase literally or idiomatically. To better evaluate our models, we also construct an idiom-enriched sentiment classification dataset with considerable scale and abundant peculiarities of idioms. The qualitative and quantitative experimental analyses demonstrate the efficacy of our models. The newly-introduced datasets are publicly available at `http://nlp.fudan.edu.cn/data/`

1 Introduction

Currently, neural network models have achieved great success for many natural language processing (NLP) tasks , such as text classification (Zhao et al., 2015; Liu et al., 2017), semantic matching (Liu et al., 2016a,b), and machine translation (Cho et al., 2014). The key factor of these neural models is how to compose a phrase or sentence representation from its constitutive words. Typically, a shared compositional function is used to compose word vectors recursively until obtaining the representation of the phrase or sentence. The form of compositional function involves many kinds of neural networks, such as recurrent neural networks (Hochreiter and Schmidhuber, 1997; Chung et al., 2014), convolutional neural networks (Collobert et al., 2011; Kalchbrenner et al., 2014), and recursive neural networks (Socher et al., 2013; Tai et al., 2015; Zhu et al., 2015).

However, these methods show an obvious defect in representing idiomatic phrases, whose semantics are not literal compositions of the individual words. For example, “`pulling my leg`” is idiomatic, and its meaning cannot be directly derived from a literal combination of its contained words. Due to its importance, some previous work focuses on automatic identification of idioms (Katz and Giesbrecht, 2006; Li and Sporleder, 2009; Fazly et al., 2009; Peng et al., 2014; Salton et al., 2016). However, challenge remains to take idioms into account to improve neural based semantic representations of phrases or sentences.

Motivated by the **literal-first** psycholinguistic hypothesis proposed by Bobrow and Bell (1973), in this paper, we propose an end-to-end neural model for idiom-aware distributed semantic representation, in which we adopt a neural architecture of recursive network (Socher et al., 2013; Tai et al., 2015; Zhu et al., 2015) to learn the compositional semantics over a constituent tree. More concretely, we introduce a neural idiom detector for each phrase in a sentence to adaptively determine their compositionality: literal or idiomatic manner. For the literal phrase, we compute its semantics from its constituents while for the idiomatic phrase, we design two different ways to learn representations of idioms grounded in two different linguistic views of idioms (Katz, 1963; Fraser, 1970; Nunberg et al., 1994).

To evaluate our models towards the ability to understand sentences with idioms, we conduct our

[*] Corresponding author.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1204–1213
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

experiments on sentiment classification task due to the following reasons: 1) Idioms typically imply an affective stance toward something and they are common in reviews and comments (Williams et al., 2015). 2) The error analysis of sentiment classification results reveals that a large number of errors are caused by idioms (Balahur et al., 2013).

The contributions of this work are summarized as follows:

- We grow the capacity of recursive neural network, enabling it to model idiomatic phrases and handle ubiquitous phenomenon of idiomatic variations when learning a sentential representation.
- We integrate idioms understanding into a real-world NLP task instead of evaluating idiom detection as a standalone task.
- We construct a new real-world dataset covering abundant idioms with original and variational forms. The elaborate qualitative and quantitative experimental analyses show the effectiveness of our models.

2 Linguistic Interpretation of Idioms

Recently, idioms have raised eyebrows among linguists, psycholinguists, and lexicographers due to their pervasiveness in daily discourse and their fascinating properties in linguistics literature (Villavicencio et al., 2005; Salton et al., 2014). As peculiar linguistic constructions (Villavicencio et al., 2005; Salton et al., 2014), idioms have three following properties:

Invisibility Idioms always disguise themselves as normal multi-words in sentences. It makes end-to-end training hard since we should detect idioms first, and then understand them.

Idiomaticity Idioms are semantically opaque, whose meanings cannot be derived from their constituent words. Existing compositional distributed approaches fail due to the hypothesis that the meaning of any phrase can be composed of the meanings of its constituents.

Flexibility While structurally fixed, idioms allow variations. The words of some idioms can be removed or substituted by other words.

Table 1 shows the three properties of idioms and the resulting challenges for distributed compositional semantics. To address these challenges, two different perspectives have been held for idiom comprehension.

Property	**Challenges**	**V1**	**V2**
Invisibility	End-to-end training is difficult	✓	✓
Idiomaticity	Difficult to predict meanings of idioms	×	✓
Flexibility	Hard to handle variation and generalize	✓	×

Table 1: The main properties of idioms and corresponding challenges. **V1** and **V2** represent two different linguistic perspectives towards idiom comprehension: arbitrary and compositional perspectives. ✓ indicates the perspective suffers from the corresponding challenge.

The first perspective treats idioms as long words whose meanings are stipulated arbitrarily and can not be predict from its constituent (Katz, 1963; Fraser, 1970). However, a lot of idioms have shown certain degree of flexibility in term of morphology and lexeme, so this kind of method handles variation badly and fails to generalize.

The second perspective considers idioms as linguistic expressions (Nunberg et al., 1994), whose meanings are determined by the meanings of their constituent parts and some compositional rules can be used to combine them. This fully compositional view may handle lexical variations, but it suffers from the idiomaticity problem, for the meanings of idioms are opaque.

3 Proposed Models

We propose an end-to-end neural model for idiom-aware distributed semantic representation. Specifically, in terms of invisibility, we introduce a neural idiom detector to adaptively distinguish literal and idiomatic meaning of each phrase when learning sentence representations. For the literal phrase, we compute its semantics from its constituents with Tree-structured LSTM (TreeLSTM) (Tai et al., 2015; Zhu et al., 2015). For the idiomatic phrase, we design two different ways to learn representations of idioms grounded in two different linguistic views of idioms, which considers the idiomaticity and flexibility properties of idioms. Figure 1 illustrates the framework of our proposed models, which consist of three modules: literal interpreter, idiom detector and idiomatic interpreter.

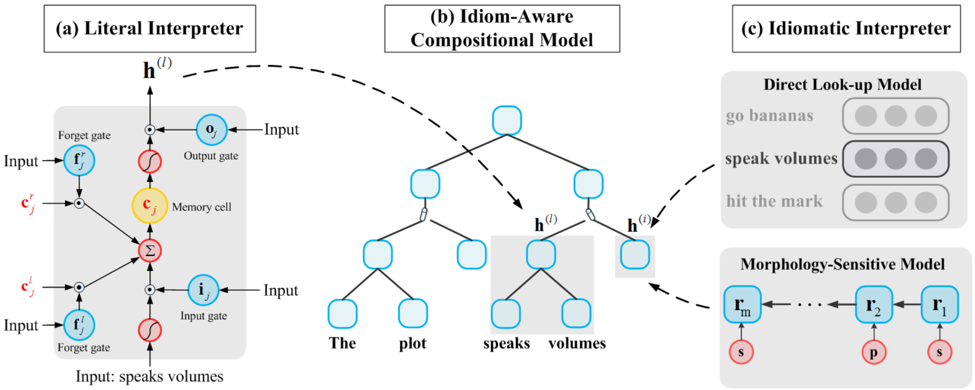


Figure 1: Illustration of different modules for our proposed idiom-aware composition network corresponding to the sentence “`The plot speaks volumes`”

3.1 Literal Interpreter

The literal interpreter is basically a compositional semantic model, in which the semantics of a phrase is composed by literal meanings of its constituent words. Several existing models could serve as literal interpreter. In this paper, we adopt TreeLSTM (Tai et al., 2015) due to its superior performance.

Formally, given a binary constituency tree T induced by a sentence, each non-leaf node corresponds to a phrase. We refer to $\mathbf{h}_j$ and $\mathbf{c}_j$ as the hidden state and memory cell of each node j. The transition equations of node j are as follows:

$$\begin{bmatrix} \tilde{\mathbf{c}}_j \\ \mathbf{o}_j \\ \mathbf{i}_j \\ \mathbf{f}_j^l \\ \mathbf{f}_j^r \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} T_{\mathbf{A},\mathbf{b}} \begin{bmatrix} \mathbf{x}_j \\ \mathbf{h}_j^l \\ \mathbf{h}_j^r \end{bmatrix}, \qquad (1)$$

$$\mathbf{c}_j = \tilde{\mathbf{c}}_j \odot \mathbf{i}_j + \mathbf{c}_j^l \odot \mathbf{f}_j^l + \mathbf{c}_j^r \odot \mathbf{f}_j^r, \qquad (2)$$

$$\mathbf{h}_j = \mathbf{o}_j \odot \tanh\left(\mathbf{c}_j\right), \qquad (3)$$

where $\mathbf{x_j}$ denotes the input vector and is non-zero if and only if it is a leaf node. The superscript l and r represent the left child and right child respectively. σ represents the logistic sigmoid function and $\odot$ denotes element-wise multiplication. $T_{\mathbf{A},\mathbf{b}}$ is an affine transformation which depends on parameters of the network $\mathbf{A}$ and $\mathbf{b}$. Figure 1-(a) gives an illustration of TreeLSTM unit.

3.2 Idiom Detector

Despite the success of TreeLSTM, there is still existing potential weakness of the hypothesis that the meaning of a phrase or a sentence can be composed from the meanings of its constituents. Previous neural sentence models are poor at learning the meanings of idiomatic phrases, not to mention modeling the idiomatic variations.

Therefore, we introduce a parameterized idiom detector, which is used for detecting the boundary between literal and idiomatic meanings. Specifically, if a compositional interpretation is nonsensical in the context of a sentence, then the detector is supposed to check whether an idiomatic sense should be taken and whether it makes sense. This **literal-first** model of idiom comprehension is motivated by the psycholinguistic hypothesis proposed by Bobrow and Bell (1973).

Due to ignoring the context information, TreeLSTM suffers from the problem of disambiguation. For example, the phrase “`in the bag`” is compositional in the sentence “`The dictionary is in the bag`” while it has idiomatic meaning in the sentence “`The election is in the bag unless the voters find out about my past`”. To address this problem, we explicitly model the context representation and integrate it into the process of sentence composition.

Context Representation More concretely, for each non-leaf node i and its corresponding phrase p_i, we define C as a word set which contains words surrounding the phrase p_i. Then the context representation s_i can be obtained as follow:

$$\mathbf{s}_i = f(c_1, c_2, ..., c_k; \theta) \qquad (4)$$

where f is a function with learnable parameter θ. Here, the function is implemented in two approaches, NBOW and LSTM.

Detector The detector outputs a scalar α to determine whether the meaning of a phrase is literal or idiomatic. Formally, for the phrase i (non-leaf node i) with its context information $\mathbf{s}_i$ and literal meaning $\mathbf{h}_i^{(l)}$, we compute the semantic compositional score α_i using a single layer multilayer perceptron.

$$\alpha_i = \sigma(\mathbf{v}_s^T \tanh(\mathbf{W}_s[\mathbf{h}_i^{(l)}, \mathbf{s}_i])), \tag{5}$$

where $\mathbf{W}_s \in \mathbb{R}^{d\times 2d}$ and $\mathbf{v}_s \in \mathbb{R}^d$ are learnable parameters.

3.3 Idiomatic Interpreter

Idiomatic phrases pose a clear challenge to current compositional models of language comprehension. However, until recently, there is little investigation of learning idiomatic phrases in a real-world task. Here, based on different views of idioms (Katz, 1963; Fraser, 1970; Nunberg et al., 1994), we propose two idiomatic interpreters to model them.

Direct Look-Up Model Inspired by the direct access theory for idiom comprehension (Glucksberg, 1993), in this model, once a phrase p is detected as an idiom, it will be regarded as a long word like a key, and then their meanings can be directly retrieved from an external memory M, which is a table and stores idiomatic information for each idiom as depicted in the top subfigure in Figure 1-(c). Formally, the idiomatic meaning for a phrase can be obtained as:

$$\mathbf{h}^{(i)} = \mathbf{M}[k] \tag{6}$$

where k denotes the index of the corresponding phrase p.

Morphology-Sensitive Model Since most idioms enjoy certain flexibility in morphology, lexicon, syntax, the above model suffers from the problem of idiom variations. To remedy this, inspired by the compositional view of idioms (Nunberg et al., 1994) and recent success of character-based models (Kim et al., 2016; Lample et al., 2016; Chung et al., 2016), we propose to use CharLSTM to directly encode the meaning of a phrase in an idiomatic space and generate an idiomatic representation, which is not contaminated by its literal meaning and sensitive to different variations.

Formally, for each non-leaf node i and its corresponding phrase p_i in a constituency tree, we apply charLSTM to phrase p_i as depicted in the bottom subfigure in Figure 1-(c) and utilize the emitted hidden states $\mathbf{r}_j$ to represent the idiomatic meaning of the phrase.

$$\mathbf{r}_j = \textbf{Char-LSTM}(\mathbf{r}_{j-1}, \mathbf{c}_{j-1}, \mathbf{x}_j) \tag{7}$$

where $j = 1, 2, \cdots, m$ and m represents the length of the input phrase.

Then, we can obtain the idiomatic representation:

$$\mathbf{h}^{(i)} = \mathbf{r}_m \tag{8}$$

After obtaining the literal or idiomatic meanings, we can compute the final representation for phrase p_i:

$$\mathbf{h}_i = \alpha_i \mathbf{h}_i^{(i)} + (1 - \alpha_i)\mathbf{h}_i^{(l)} \tag{9}$$

3.4 Analysis of Two Proposed Idiomatic Interpreters

Given a phrase, both interpreters can generate a corresponding semantic representation, which is not contaminated by its literal meaning. The difference is that Look-Up Model takes a totally non-compositional view that the meanings of idioms can be directly accessed from an external dictionary. This straightforward retrieval mechanism is more efficient and can introduce external prior knowledge by utilizing pre-trained external dictionary.

By contrast, Morphology-Sensitive Model holds the idea that idiomatic meanings can still be composed in an idiomatic space, which allows this model to understand idioms better in terms of flexibility. Besides, this kind of model does not require an extra dictionary.

4 iSent: A Benchmark for Idiom-Enriched Sentiment Classification Dataset

To evaluate our models, we need a task that heavily depends on the understanding of idioms. In this paper, we choose sentiment classification task due to following reasons: 1) Idioms typically imply an affective stance toward something and they are common in reviews and comments (Williams et al., 2015). 2) The error analysis of sentiment classification results reveals that a large number of errors are caused by idioms (Balahur et al., 2013).

In this section, we will first give a brief description of the most commonly used datasets for sentiment classification so as to motivate the need for a new benchmark dataset.

Dataset	Train	Dev.	Test	Class	L_{avg}	$\lvert\mathcal{V}\rvert$
MR	9596	-	1066	2	22	21K
SST-1	8544	1101	2210	5	19	18K
SST-2	6920	872	1821	2	18	15K
SUBJ	9000	-	1000	2	21	21K

Table 2: Statistics of the four mainstream datasets for sentiment classification. L_{avg} denotes the average length of documents; $|\mathcal{V}|$ denotes the size of vocabulary.

4.1 Mainstream Datasets for Sentiment Classification

We list four kinds of datasets which are most commonly used for sentiment classification in NLP community. Additionally, we also evaluate our models on these datasets to make a comparison with many recent proposed models. Each dataset is briefly described as follows.

- **SST-1** The movie reviews with five classes (negative, somewhat negative, neutral, somewhat positive, positive) in the Stanford Sentiment Treebank[1] (Socher et al., 2013).
- **SST-2** The movie reviews with binary classes. It is also derived from the Stanford Sentiment Treebank.
- **MR** The movie reviews with two classes [2](Pang and Lee, 2005).
- **SUBJ** Subjectivity data set where the goal is to classify each instance (snippet) as being subjective or objective. (Pang and Lee, 2004)

The detailed statistics about these four datasets are listed in Table 2.

4.2 Reasons for a New Dataset

Differing from previous work, which evaluating idiom detection as a standalone task, we want to integrate idiom understanding into sentiment classification task. However, most of existing sentiment datasets do not cover enough idioms or related linguistic phenomenon. To better evaluate our models on idiom understanding task, we proposed an idiom-enriched sentiment classification dataset, in which each sentence contains at least one idiom.

Additionally, considering most idioms have certain flexibility in morphology, lexicon and syntax, we enrich our dataset by introducing different types of idiom variations so that we can further evaluate the ability that the model handle different idiomatic variations. As shown in Table 3, we sum up two types of phenomena towards idiom variations and, for each variation, we obtain several corresponding sentences from a large corpora.

4.3 Data collection

We crawl the website `rottentomatoes.com` to excerpt movie reviews with corresponding scores and collect the idioms from dictionary (Flavell and Flavell, 2006). The idiom dictionary contains lexical variations while has no morphology variations. To address this problem, we manually annotate the morphological variation of each idiom in term of verb(tense), noun(plural or singular).

Then we filter these movie reviews with idioms ensuring that each sentence covers at least one idiom. After that, we obtain nearly 15K movie reviews covering 1K idioms. To further improve the quality of these idiom-enriched sentences, we take some strategies to filter the dataset and finally construct 13K idiom-enriched sentences.

- If the occurrence of an idiom in all the reviews is less than 3, we threw this idiom and corresponding reviews. [3]
- We find some "idioms" in sentences are movie names rather than expressing idiomatic meanings and we filtered this kind of noise.

4.4 Statistics

The iSent dataset finally contains 9752 training samples , 1020 development samples and 2003 test samples. Besides, the development and test sets cover different types of idiom variations allowing us to test the model's generalization. Table 4 shows the detailed statistics and Figure 2 shows the distribution of the number of reviews over different lengths.

5 Experiment

We first evaluate our proposed models on four popular sentiment datasets, so that we can make a comparison with varieties of competitors. And then, we use the newly-introduced dataset to make more detailed experiment analyses.

[1] `http://nlp.stanford.edu/sentiment.`

[2] `https://www.cs.cornell.edu/people/pabo/movie-review-data/.`

[3] The reason is that, for some idioms, we should split their corresponding reviews into train/dev/test sets.

Variations		Examples
Morp.	Verb Noun	go bananas → went bananas in a nutshell → in nutshells
Lexical	Add. Sub.	in the lurch → in the big lurch on the same page → on different pages

Table 3: Idiom variations at morphological and lexical level. Add. and Sub. refer to lexical addition and substitution respectively.

	Train	Dev			Test		
		O	M	L	O	M	L
Idiom	1247	124	21	40	124	21	40
Sent.	9752	720	200	100	1403	400	200

Table 4: Key statistics for the idioms and sentences in iSent dataset. O(Original) denotes the idioms in dev/test sets are in original forms and have appeared in training set. M(Morphology) and L(Lexical) represent the morphology and lexical idiom variations respectively and they are unseen in training set.

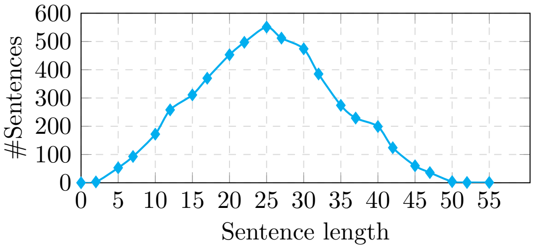


Figure 2: The distribution of the number of reviews over different lengths.

5.1 Experimental Settings

Loss Function Given a sentence and its label, the output of neural network is the probabilities of the different classes. The parameters of the network are trained to minimise the cross-entropy of the predicted and true label distributions. To minimize the objective, we use stochastic gradient descent with the diagonal variant of AdaGrad (Duchi et al., 2011).

Initialization and Hyperparameters In all of our experiments, the word embeddings for all of the models are initialized with the GloVe vectors (Pennington et al., 2014). The other parameters are initialized by randomly sampling from uniform distribution in $[-0.1, 0.1]$.

For each task, we take the hyperparameters which achieve the best performance on the development set via a small grid search over combinations of the initial learning rate $[0.1, 0.01, 0.001]$, l_2 regularization $[0.0, 5E{-}5, 1E{-}5]$ The final hyper-parameters are as follows. The initial learning rate is 0.1. The regularization weight of the parameters is $1E{-}5$.

For all the sentences from the five datasets, we parse them with constituency parser (Klein and Manning, 2003) to obtain the trees for our and some competitor models.

5.2 Competitor Models

We give some descriptions about the setting of our models and several baseline models.

- CharLSTM: Character level LSTM.
- TLSTM: Vanilla tree-based LSTM, proposed by Tai et al. (2015).
- Cont-TLSTM: Context-dependent tree-based LSTM, introduced by Bowman et al. (2016).
- iTLSTM-Lo: Proposed model with Look-Up idiomatic interpreter.
- iTLSTM-Mo: Proposed model with Morphology-Sensitive interpreter.

5.3 Evaluation over Mainstream Datasets

The experimental results are shown in Table 5. We can see Cont-TLSTM outperforms TLSTM on all four tasks, showing the importance of context-sensitive composition. Besides, both iTLSTM-Lo and iTLSTM-Mo achieve better results than TLSTM and Cont-LSTM, which indicates the effectiveness of our introduced modules (detector and idiomatic interpreter). Additionally, compared with iTLSTM-Lo, iTLSTM-Mo behaves better, suggesting its char-based idiomatic interpreter is more powerful.

Although four mainstream datasets are not rich in idioms, we could also observe substantial improvement gained from our models. We attribute this success to the power of introduced detector in identifying other non-compositional collocations besides idioms. We will discuss about this later.

5.4 Evaluation over iSent Dataset

Since iSent is a newly-introduced dataset, there is no existing baselines. Nevertheless, we provide several strong baselines implemented by ourselves as shown in Table 6, and we can observe that:

- Differing from the improvement achieved on mainstream datasets, proposed models have shown their advantages on idiom-enriched sentences. They obtain more significant improvements.

Model	MR	SST-1	SST-2	SUBJ
NBOW	77.2	42.4	80.5	91.3
RAE (Socher et al., 2011)	77.7	43.2	82.4	-
MV-RNN (Socher et al., 2012)	79.0	44.4	82.9	-
RNTN (Socher et al., 2013)	-	45.7	85.4	-
DCNN (Kalchbrenner et al., 2014)	-	48.5	86.8	-
CNN-multichannel (Kim, 2014)	81.5	47.4	88.1	93.2
CharLSTM	75.2	44.0	85.2	90.2
TLSTM	78.7	48.5	86.1	91.0
Cont-TLSTM	79.5	48.9	86.4	91.7
iTLSTM-Lo	81.6	49.9	87.7	93.2
iTLSTM-Mo	**82.5**	**51.2**	**88.2**	**94.5**

Table 5: Accuracies of our models on four datasets against state-of-the-art neural models.

Model	Train	Dev.	Test
NBOW	80.9	77.1	74.5
LSTM	87.5	76.9	75.0
BiLSTM	93.4	76.8	76.3
CharLSTM	92.4	75.1	74.4
TLSTM	88.2	75.3	74.9
Cont-TLSTM	90.8	76.2	75.5
iTLSTM-Lo	88.9	79.6	78.1
iTLSTM-Mo	91.3	81.1	80.0

Table 6: Accuracies of our models on iSent dataset against typical baselines. BiLSTM represents bidirectional LSTM.

- Additionally, iTLSTM-Lo performs worse than iTLSTM-Mo while still surpasses baseline models, which also indicates the variation-sensitive model (iTLSTM-Mo) of idioms could further improve the performance.

5.5 Analysis

In this section, we will provide more detailed quantitative and qualitative analysis in terms of three properties of idioms described in Table 1: flexibility, invisibility and idiomaticity.

Flexibility Besides the overall accuracies on the test set, we also list the performance achieved by different models over the different parts of test set: original, morphological and lexical, which represents different types of variations and have been described in Table 4.

We can see from Figure 3, both idiom-aware models achieve better performance than Cont-TLSTM by a large margin on the original part of test set, which indicates the importance of understanding idiomatic phrases during sentence modelling. Additionally, iTLSTM+Mo outperforms the other two models on the test set, suggesting the effectiveness of morphology-based model for modeling idiom variations.

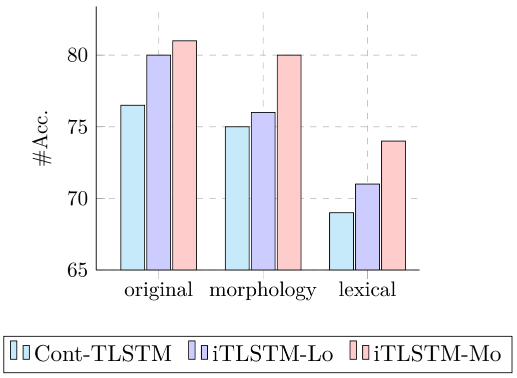


Figure 3: Performances achieved by different models are subdivided into three parts. Original, Morphology, Lexical represents accuracies achieved by corresponding part of test data.

Invisibility and Idiomaticity Previous experimental results have shown the effectiveness of our models. Here, we want to know how the introduced idiom detector contributes to the improvement of performance. Toward this end, we analyze all the 157 samples which our model predicts correctly while baseline model (Cont-TLSTM) fails on iSent, and find more than 120 sentences are given wrong sentiment by Cont-TLSTM due to ignoring the figurative meanings of idioms. For example, as shown in Figure 4, we randomly sample a sentence and analyze the changes of the predicted sentiment score at different nodes of the tree.

The sentence "`The movie enable my friends to blow a gasket`" has negative sentiment. Cont-TLSTM gives a wrong prediction due to ignoring the information expressed by the idiomatic phrase "`blow a gasket`". By contrast, our model correctly detects this idiom, whose meaning plays a major role in final sentiment prediction.

Non-compositional Phrases Detection Besides idioms, we find the introduced detector can also pick up other types of non-compositional phrases[4]. We roughly sum up these non-

[4] An idiom itself is a non-compositional phrase.

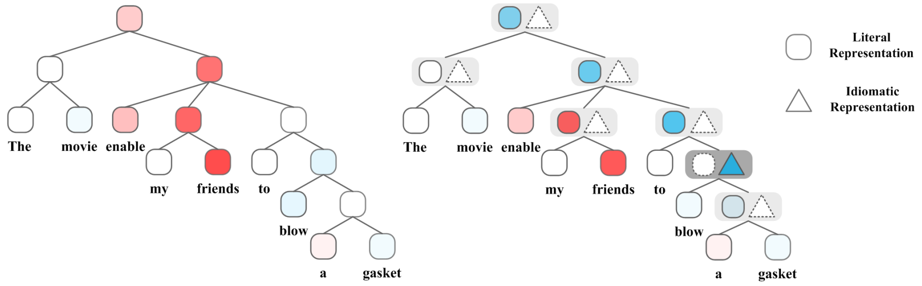


Figure 4: The change of the predicted sentiment score at different nodes of the tree. The red and blue color represent positive and negative sentiment respectively, where darker indicates higher confidence. Dashed triangle or square box denote not being selected by detector.

PN	VP	NP	AP
Notting Hill	*let alone*	*short cuts*	*on earth*
Holly Bolly	*take cover*	*deja vu*	*at once*
star wars saga	*rips off*	*black comedy*	*all in all*
Barry Skolnick	*thumb down*	*femme fatale*	*not only*
Apollo 13	*fall apart*	*no problem*	*at times*

Table 7: PN, VP, NP and AP represent proper noun, noun phrase, verb phrase and adverbial phrase respectively.

compositional phrases picked up by introduced detector from all the five development sets and list them in Table 7.

From the table, we can see that most of these phrases either imply an affective stance toward something: "`thumbs down`", or are critical to the understanding of sentences such as the "Verb Phrases" and "Adverb Phrases". For example, the sentence "`More often than not, this mixed bag hit its mark`" has a positive sentiment. Cont-TLSTM pays much more attention to the word "`not`" without realizing that it belongs to the collocation "`more often than not`", which expresses neutral emotion. In comparison, our model regards this collocation as a whole with neutral sentiment, which is crucial for the final prediction.

6 Related Work

Previous work related to idioms focused on their identification, which falls in two kinds of paradigms: idiom type classification (Gedigian et al., 2006; Shutova et al., 2010) and idiom token classification (Katz and Giesbrecht, 2006; Li and Sporleder, 2009; Fazly et al., 2009; Peng et al., 2014; Salton et al., 2016). Different with these work, we integrate idioms understanding into a real-world task and consider different peculiarities of idioms in an end-to-end trainable framework.

Recently, there are some work exploring the compositionality of various types of phrases (Kartsaklis et al., 2012; Muraoka et al., 2014; Hermann, 2014; Hashimoto and Tsuruoka, 2016). Compared with these work, we focus on how to properly model idioms under the context of sentence representations.

More recently, Zhu et al. (2016) propose a DAG-structured LSTM to incorporate external semantics including non-compositional or holistically learned semantics. Its key characteristic is that a DAG needs be built in advance, which merges some detected n-grams as the non-compositional phrases based on external knowledge. Different from this work, we focus on how to integrate detection and understanding of idioms into a unified end-to-end model, in which an idiomatic detector is introduced to adaptively control the semantic compositionality. Particularly, in the whole process no extra information is given to tell which phrases should be regarded as non-compositional.

7 Conclusion and Future Work

In this paper, we lay idioms understanding in the context of sentence-level semantic representation based on two linguistic perspectives. To apply our model into the real-world task, we introduce a sizeable idiom-enriched sentiment classification dataset, which covers abundant peculiarities of idioms. We make an elaborate experiment design and case analysis to evaluate the effectiveness of our proposed models.

In future work, we would like to investigate more complicated idiom-enriched NLP tasks, such as machine translation.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011 and 61672162), Shanghai Municipal Science and Technology Commission (No. 16JC1420401), and Young Elite Scientists Sponsorship Program By CAST (2015QNRC001).

References

Alexandra Balahur, Ralf Steinberger, Mijail Kabadjov, Vanni Zavarella, Erik Van Der Goot, Matina Halkia, Bruno Pouliquen, and Jenya Belyaeva. 2013. Sentiment analysis in the news. *arXiv preprint arXiv:1309.6202* .

Samuel A Bobrow and Susan M Bell. 1973. On catching on to idiomatic expressions. *Memory & Cognition* 1(3):343–346.

Samuel R Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. *arXiv preprint arXiv:1603.06021* .

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147* .

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The JMLR* 12:2493–2537.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The JMLR* 12:2121–2159.

Afsaneh Fazly, Paul Cook, and Suzanne Stevenson. 2009. Unsupervised type and token identification of idiomatic expressions. *Computational Linguistics* 35(1):61–103.

Linda Flavell and Roger Flavell. 2006. *Dictionary of idioms and their origins*. Kyle Cathie Limited.

Bruce Fraser. 1970. Idioms within a transformational grammar. *Foundations of language* pages 22–42.

Matt Gedigian, John Bryant, Srini Narayanan, and Branimir Ciric. 2006. Catching metaphors. In *Proceedings of the Third Workshop on Scalable Natural Language Understanding*. Association for Computational Linguistics, pages 41–48.

Sam Glucksberg. 1993. Idiom meanings and allusional content. *Idioms: Processing, structure, and interpretation* pages 3–26.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2016. Adaptive joint learning of compositional and non-compositional phrase embeddings. *arXiv preprint arXiv:1603.06067* .

Karl Moritz Hermann. 2014. Distributed representations for compositional semantics. *arXiv preprint arXiv:1411.3146* .

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

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of ACL*.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2012. A unified sentence space for categorical distributional-compositional semantics: Theory and experiments. In *In Proceedings of COLING: Posters*. Citeseer.

Graham Katz and Eugenie Giesbrecht. 2006. Automatic identification of non-compositional multiword expressions using latent semantic analysis. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*. Association for Computational Linguistics, pages 12–19.

Jerrold J Katz. 1963. *Semantic interpretation of idioms and sentences containing them*. Research Laboratory of Electronics, Massachusetts Institute of Technology.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. pages 423–430.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* .

Linlin Li and Caroline Sporleder. 2009. Classifier combination for contextual idiom detection without labelled data. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, pages 315–323.

Pengfe Liu, Xipeng Qiu, Jifan Chen, and Xuanjing Huang. 2016a. Deep fusion LSTMs for text semantic matching. In *Proceedings of ACL*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. *arXiv preprint arXiv:1704.05742* .

Pengfei Liu, Xipeng Qiu, Yaqian Zhou, Jifan Chen, and Xuanjing Huang. 2016b. Modelling interaction of sentence pair with coupled-lstms. In *Proceedings of the 2016 Conference on EMNLP*.

Masayasu Muraoka, Sonse Shimaoka, Kazeto Yamamoto, Yotaro Watanabe, Naoaki Okazaki, and Kentaro Inui. 2014. Finding the best model among representative compositional models. In *Proceedings of the 28th Pacific Asia Conference on Language, Information, and Computation (PACLIC 2014)*. pages 65–74.

Geoffrey Nunberg, Ivan A Sag, and Thomas Wasow. 1994. Idioms. *Language* pages 491–538.

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

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 115–124.

Jing Peng, Anna Feldman, and Ekaterina Vylomova. 2014. Classifying idiomatic and literal expressions using topic models and intensity of emotions. In *EMNLP*. pages 2019–2027.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the EMNLP* 12:1532–1543.

Giancarlo Salton, Robert Ross, and John Kelleher. 2014. An empirical study of the impact of idioms on phrase based statistical machine translation of english to brazilian-portuguese .

Giancarlo D Salton, Robert J Ross, and John D Kelleher. 2016. Idiom token classification using sentential distributed semantics. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. pages 194–204.

Ekaterina Shutova, Lin Sun, and Anna Korhonen. 2010. Metaphor identification using verb and noun clustering. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 1002–1010.

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

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*.

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

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Aline Villavicencio, Francis Bond, Anna Korhonen, and Diana McCarthy. 2005. Introduction to the special issue on multiword expressions: Having a crack at a hard nut.

Lowri Williams, Christian Bannister, Michael Arribas-Ayllon, Alun Preece, and Irena Spasić. 2015. The role of idioms in sentiment analysis. *Expert Systems with Applications* 42(21):7375–7385.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. *arXiv preprint arXiv:1504.05070* .

Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *ICML*. pages 1604–1612.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2016. Dag-structured long short-term memory for semantic compositionality. In *Proceedings of NAACL-HLT*. pages 917–926.

Macro Grammars and Holistic Triggering for Efficient Semantic Parsing

Yuchen Zhang and **Panupong Pasupat** and **Percy Liang**
{zhangyuc,ppasupat,pliang}@cs.stanford.edu
Computer Science Department, Stanford University

Abstract

To learn a semantic parser from denotations, a learning algorithm must search over a combinatorially large space of logical forms for ones consistent with the annotated denotations. We propose a new online learning algorithm that searches faster as training progresses. The two key ideas are using *macro grammars* to cache the abstract patterns of useful logical forms found thus far, and *holistic triggering* to efficiently retrieve the most relevant patterns based on sentence similarity. On the WikiTableQuestions dataset, we first expand the search space of an existing model to improve the state-of-the-art accuracy from 38.7% to 42.7%, and then use macro grammars and holistic triggering to achieve an 11x speedup and an accuracy of 43.7%.

1 Introduction

We consider the task of learning a semantic parser for question answering from question-answer pairs (Clarke et al., 2010; Liang et al., 2011; Berant et al., 2013; Artzi and Zettlemoyer, 2013; Pasupat and Liang, 2015). To train such a parser, the learning algorithm must somehow search for *consistent* logical forms (i.e., logical forms that execute to the correct answer denotation). Typically, the search space is defined by a compositional grammar over logical forms (e.g., a context-free grammar), which we will refer to as the *base grammar*.

To cover logical forms that answer complex questions, the base grammar must be quite general and compositional, leading to a huge search space that contains many useless logical forms. For example, the parser of Pasupat and Liang (2015) on Wikipedia table questions (with beam size 100) generates and featurizes an average of 8,400 partial logical forms per example. Searching for consistent logical forms is thus a major computational bottleneck.

	Rank	**Nation**	**Gold**	**Silver**	**Bronze**
r_1 :	1	France	3	1	1
r_2 :	2	Ukraine	2	1	2
r_3 :	3	Turkey	2	0	1
r_4 :	4	Sweden	2	0	0
r_5 :	5	Iran	1	2	1

Table 1: A knowledge base for the question x = "*Who ranked right after Turkey?*". The target denotation is $y = \{\texttt{Sweden}\}$.

In this paper, we propose *macro grammars* to bias the search towards structurally sensible logical forms. To illustrate the key idea, suppose we managed to parse the utterance "*Who ranked right after Turkey?*" in the context of Table 1 into the following consistent logical form (in lambda DCS) (Section 2.1):

$$\mathbf{R}[\texttt{Nation}].\mathbf{R}[\texttt{Next}].\texttt{Nation}.\texttt{Turkey},$$

which identifies the cell under the Nation column in the row after Turkey. From this logical form, we can abstract out all relations and entities to produce the following *macro*:

$$\mathbf{R}[\{\mathit{Rel}\#1\}].\mathbf{R}[\texttt{Next}].\{\mathit{Rel}\#1\}.\{\mathit{Ent}\#2\},$$

which represents the abstract computation: "identify the cell under the $\{\mathit{Rel}\#1\}$ column in the row after $\{\mathit{Ent}\#2\}$." More generally, macros capture the overall shape of computations in a way that generalizes across different utterances and knowledge bases. Given the consistent logical forms of utterances parsed so far, we extract a set of macro rules. The resulting macro grammar consisting of these rules generates only logical forms conforming to these macros, which is a much smaller and

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1214–1223
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

higher precision set compared to the base grammar.

Though the space of logical forms defined by the macro grammar is smaller, it is still expensive to parse with them as the number of macro rules grows with the number of training examples. To address this, we introduce *holistic triggering*: for a new utterance, we find the K most similar utterances and only use the macro rules induced from any of their consistent logical forms. Parsing now becomes efficient as only a small subset of macro rules are triggered for any utterance. Holistic triggering can be contrasted with the norm in semantic parsing, in which logical forms are either triggered by specific phrases (anchored) or can be triggered in any context (floating).

Based on the two ideas above, we propose an online algorithm for jointly inducing a macro grammar and learning the parameters of a semantic parser. For each training example, the algorithm first attempts to find consistent logical forms using holistic triggering on the current macro grammar. If it succeeds, the algorithm uses the consistent logical forms found to update model parameters. Otherwise, it applies the base grammar for a more exhaustive search to enrich the macro grammar. At test time, we only use the learned macro grammar.

We evaluate our approach on the WIKITABLEQUESTIONS dataset (Pasupat and Liang, 2015), which features a semantic parsing task with open-domain knowledge bases and complex questions. We first extend the model in Pasupat and Liang (2015) to achieve a new state-of-the-art test accuracy of 42.7%, representing a 10% relative improvement over the best reported result (Haug et al., 2017). We then show that training with macro grammars yields an 11x speedup compared to training with only the base grammar. At test time, using the learned macro grammar achieves a slightly better accuracy of 43.7% with a 16x run time speedup over using the base grammar.

2 Background

We base our exposition on the task of question answering on a knowledge base. Given a natural language utterance x, a semantic parser maps the utterance to a logical form z. The logical form is executed on a knowledge base w to produce denotation $[\![z]\!]_w$. The goal is to train a semantic parser from a training set of utterance-denotation pairs.

2.1 Knowledge base and logical forms

A knowledge base refers to a collection of entities and relations. For the running example "*Who ranked right after Turkey?*", we use Table 1 from Wikipedia as the knowledge base. Table cells (e.g., `Turkey`) and rows (e.g., r_3 = the 3rd row) are treated as entities. Relations connect entities: for example, the relation `Nation` maps r_3 to `Turkey`, and a special relation `Next` maps r_3 to r_4.

A logical form z is a small program that can be executed on the knowledge base. We use lambda DCS (Liang, 2013) as the language of logical forms. The smallest units of lambda DCS are entities (e.g., `Turkey`) and relations (e.g., `Nation`). Larger logical forms are composed from smaller ones, and the denotation of the new logical form can be computed from denotations of its constituents. For example, applying the join operation on `Nation` and `Turkey` gives `Nation.Turkey`, whose denotation is $[\![\texttt{Nation.Turkey}]\!]_w = \{r_3\}$, which corresponds to the 3rd row of the table. The partial logical form `Nation.Turkey` can then be used to construct a larger logical form:

$$z = \mathbf{R}[\texttt{Nation}].\mathbf{R}[\texttt{Next}].\texttt{Nation}.\texttt{Turkey}, \quad (1)$$

where $\mathbf{R}[\cdot]$ represents the reverse of a relation. The denotation of the logical form z with respect to the knowledge base w is equal to $[\![z]\!]_w = \{\texttt{Sweden}\}$. See Liang (2013) for more details about the semantics of lambda DCS.

2.2 Grammar rules

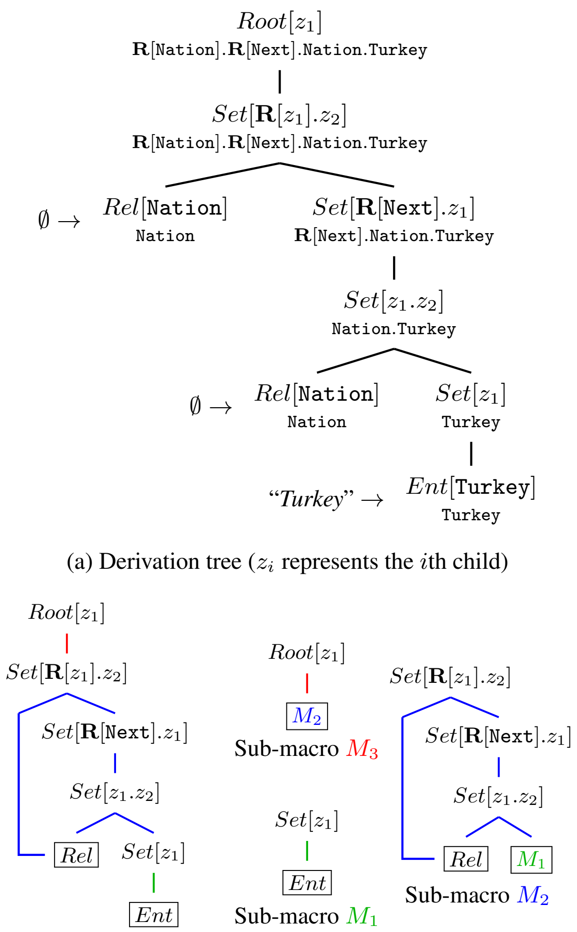
The space of logical forms is defined recursively by grammar rules. In this setting, each constructed logical form belongs to a category (e.g., $Entity$, Rel, Set), with a special category $Root$ for complete logical forms. A rule specifies the categories of the arguments, category of the resulting logical form, and how the logical form is constructed from the arguments. For instance, the rule

$$Rel[z_1] + Set[z_2] \rightarrow Set[z_1.z_2] \quad (2)$$

specifies that a partial logical form z_1 of category Rel and z_2 of category Set can be combined into $z_1.z_2$ of category Set. With this rule, we can construct `Nation.Turkey` if we have constructed `Nation` of type Rel and `Turkey` of type Set.

We consider the rules used by Pasupat and Liang (2015) for their floating parser.[1] The rules

[1] Their grammar and our implementation use more fine-grained categories ($Atomic$, $Values$, $Records$) instead of Set. We use the coarser category here for simplicity.



(a) Derivation tree (z_i represents the ith child)

(b) Macro

(c) Atomic sub-macros

Figure 1: From the derivation tree (a), we extract a macro (b), which can be further decomposed into atomic sub-macros (c). Each sub-macro is converted into a macro rule.

are divided into *compositional rules* and *terminal rules*. Rule (2) above is an example of a compositional rule, which combines one or more partial logical forms together. A terminal rule has one of the following forms:

$$TokenSpan[span] \to c[f(span)] \qquad (3)$$

$$\emptyset \to c[f(\emptyset)] \qquad (4)$$

where c is a category. A rule with the form (3) converts an utterance token span (e.g., "*Turkey*") into a partial logical form (e.g., `Turkey`). A rule with the form (4) generates a partial logical form without any trigger. This allows us to generate logical predicates that do not correspond to any part of the utterance (e.g., `Nation`).

A complete logical form is generated by recursively applying rules. We can represent the derivation process by a *derivation tree* such as in Figure 1a. Every node of the derivation tree corresponds to one rule. The leaf nodes correspond to terminal rules, and the intermediate nodes correspond to compositional rules.

2.3 Learning a semantic parser

Parameters of the semantic parser are learned from training data $\{(x_i, w_i, y_i)\}_{i=1}^n$. Given a training example with an utterance x, a knowledge base w, and a target denotation y, the learning algorithm constructs a set of candidate logical forms indicated by $\mathcal{Z}$. It then extracts a feature vector $\phi(x, w, z)$ for each $z \in \mathcal{Z}$, and defines a log-linear distribution over the candidates z:

$$p_\theta(z \mid x, w) \propto \exp(\theta^\top \phi(x, w, z)), \qquad (5)$$

where θ is a parameter vector. The straightforward way to construct $\mathcal{Z}$ is to enumerate all possible logical forms induced by the grammar. When the search space is prohibitively large, it is a common practice to use beam search. More precisely, the algorithm constructs partial logical forms recursively by the rules, but for each category and each search depth, it keeps only the B highest-scoring logical forms according to the model probability (5).

During training, the parameter θ is learned by maximizing the regularized log-likelihood of the correct denotations:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(y_i \mid x_i, w_i) - \lambda \|\theta\|_1, \qquad (6)$$

where the probability $p_\theta(y_i \mid x_i, w_i)$ marginalizes over the space of candidate logical forms:

$$p_\theta(y_i \mid x_i, w_i) = \sum_{z \in \mathcal{Z}_i : [\![z]\!]_{w_i} = y_i} p_\theta(z \mid x_i, w_i).$$

The objective is optimized using AdaGrad (Duchi et al., 2010). At test time, the algorithm selects a logical form $z \in \mathcal{Z}$ with the highest model probability (5), and then executes it on the knowledge base w to predict the denotation $[\![z]\!]_w$.

3 Learning a macro grammar

The base grammar usually defines a large search space containing many irrelevant logical forms. For example, the grammar in Pasupat and Liang (2015) can generate long chains of join operations (e.g., $\mathbf{R}$[`Silver`].`Rank`.$\mathbf{R}$[`Gold`].`Bronze`.`2`) that rarely express meaningful computations.

Algorithm 1: Processing a training example

```
Data: example $(x, w, y)$, macro grammar,
      base grammar with terminal rules $\mathcal{T}$
Select a set $\mathcal{R}$ of macro rules (Section 3.4);
Generate a set $\mathcal{Z}$ of candidate logical forms
  from rules $\mathcal{R} \cup \mathcal{T}$ (Section 2.3);
if $\mathcal{Z}$ contains consistent logical forms then
    Update model parameters (Section 3.5);
else
    Apply the base grammar to search for a
      consistent logical form (Section 2.3);
    Augment the macro grammar
      (Section 3.6);
end
Associate utterance $x$ with the highest-
  scoring consistent logical form found;
```

The main contribution of this paper is a new algorithm to speed up the search based on previous searches. At a high-level, we incrementally build a *macro grammar* which encodes useful logical form macros discovered during training. Algorithm 1 describes how our learning algorithm processes each training example. It first tries to use an appropriate subset of rules in the macro grammar to search for logical forms. If the search succeeds, then the semantic parser parameters are updated as usual. Otherwise, it falls back to the base grammar, and then add new rules to the macro grammar based on the consistent logical form found. Only the macro grammar is used at test time.

We first describe macro rules and how they are generated from a consistent logical form. Then we explain the steps of the training algorithm in detail.

3.1 Logical form macros

A *macro* characterizes an abstract logical form structure. We define the macro for any given logical form z by transforming its derivation tree as illustrated in Figure 1b. First, for each terminal rule (leaf node), we substitute the rule by a placeholder, and name it with the category on the right-hand side of the rule. Then we merge leaf nodes that represent the same partial logical form. For example, the logical form (1) uses the relation `Nation` twice, so in Figure 1b, we merge the two leaf nodes to impose such a constraint.

While the resulting macro may not be tree-like, we call each node *root* or *leaf* if it is a root node or a leaf node of the associated derivation tree.

3.2 Constructing macro rules from macros

For any given macro M, we can construct a set of *macro rules* that, when combined with terminal rules from the base grammar, generates exactly the logical forms that satisfy the macro M. The straightforward approach is to associate a unique rule with each macro: assuming that its k leaf nodes contain categories $c_1, \ldots, c_k$, we can define a rule:

$$c_1[z_1] + \cdots + c_k[z_k] \rightarrow \mathit{Root}[f(z_1, \ldots, z_k)], \tag{7}$$

where f substitutes $z_1, \ldots, z_k$ into the corresponding leaf nodes of macro M. For example, the rule for the macro in Figure 1b is

$$\mathit{Rel}[z_1]+\mathit{Ent}[z_2] \rightarrow \mathit{Root}[\mathbf{R}[z_1].\mathbf{R}[\texttt{Next}].z_1.z_2].$$

3.3 Decomposed macro rules

Defining a unique rule for each macro is computationally suboptimal since the common structures shared among macros are not being exploited. For example, while $\texttt{max}(\mathbf{R}[\texttt{Rank}].\texttt{Gold.Num.2})$ and $\mathbf{R}[\texttt{Nation}].\texttt{argmin}(\texttt{Gold.Num.2}, \texttt{Index})$ belong to different macros, the partial logical form `Gold.Num.2` is shared, and we wish to avoid generating and featurizing it more than once.

In order to reuse such shared parts, we decompose macros into *sub-macros* and define rules based on them. A subgraph M' of M is a sub-macro if (1) M' contains at least one non-leaf node; and (2) M' connects to the rest of the macro $M \backslash M'$ only through one node (the root of M'). A macro M is called *atomic* if the only sub-macro of M is itself.

Given a non-atomic macro M, we can find an atomic sub-macro M' of M. For example, from Figure 1b, we first find sub-macro $M' = M_1$. We detach M' from M and define a macro rule:

$$c'_1[z_1] + \cdots + c'_k[z_k] \rightarrow c'_{\text{out}}[f(z_1, \ldots, z_k)], \tag{8}$$

where $c'_1, \ldots, c'_k$ are categories of the leaf nodes of M', and f substitutes $z_1, \ldots, z_k$ into the sub-macro M'. The category c'_{out} is computed by serializing M' as a string; this way, if the sub-macro M' appears in a different macro, the category name will be shared. Next, we substitute the subgraph M' in M by a placeholder node with name c'_{out}. The procedure is repeated on the new graph until the remaining macro is atomic. Finally, we define a single rule for the atomic macro. The macro grammar uses the decomposed macro rules in replacement of Rule (7).

For example, the macro in Figure 1b is decomposed into three macro rules:

$$\begin{aligned} \mathit{Ent}[z_1] &\rightarrow M_1[z_1], \\ \mathit{Rel}[z_1] + M_1[z_2] &\rightarrow M_2[\mathbf{R}[z_1].\mathbf{R}[\texttt{Next}].z_1.z_2], \\ M_2[z_1] &\rightarrow \mathit{Root}[z_1]. \end{aligned}$$

These correspond to the three atomic sub-macros M_1, M_2 and M_3 in Figure 1c. The first and the second macro rules can be reused by other macros.

Having defined macro rules, we now describe how Algorithm 1 uses and updates the macro grammar when processing each training example.

3.4 Triggering macro rules

Throughout training, we keep track of a set $\mathcal{S}$ of training utterances that have been associated with a consistent logical form. (The set $\mathcal{S}$ is updated by Step 9 of Algorithm 1.) Then, given a training utterance x, we compute its K-nearest neighbor utterances in $\mathcal{S}$, and select all macro rules that were extracted from their associated logical forms. These macro rules are used to parse utterance x.

We use token-level Levenshtein distance as the distance metric for computing nearest neighbors. More precisely, every utterance is written as a sequence of lemmatized tokens $x = (x^{(1)}, \ldots, x^{(m)})$. After removing all determiners and infrequent nouns that appear in less than 2% of the training utterances, the distance between two utterances x and x' is defined as the Levenshtein distance between the two sequences. When computing the distance, we treat each word token as an atomic element. For example, the distance between "highest score" and "best score" is 1. Despite its simplicity, the Levenshtein distance does a good job in capturing the structural similarity between utterances. Table 2 shows that nearest neighbor utterances often map to consistent logical forms with the same macro.

In order to compute the nearest neighbors efficiently, we pre-compute a sorted list of $K_{\text{max}} = 100$ nearest neighbors for every utterance before training starts. During training, calculating the intersection of this sorted list with the set $\mathcal{S}$ gives the nearest neighbors required. For our experiments, the preprocessing time is negligible compared to the overall training time (less than 3%), but if computing nearest neighbors is expensive, then parallelization or approximate algorithms (e.g., Indyk, 2004) could be used.

Who *ranked* **right after** *Turkey?* **Who** *took office* **right after** *Uriah Forrest?*
How many more *passengers flew to Los Angeles* **than** *to Saskatoon* **in** *2013?* **How many more** *Hungarians live in the Serbian Banat region* **than** *Romanians* **in** *1910?*
Which is *deeper*, *Lake Tuz* **or** *Lake Palas Tuzla?* **Which** *peak* **is** *higher*, *Mont Blanc* **or** *Monte Rosa?*

Table 2: Examples of nearest neighbor utterances in the WIKITABLEQUESTIONS dataset.

3.5 Updating model parameters

Having computed the triggered macro rules $\mathcal{R}$, we combine them with the terminal rules $\mathcal{T}$ from the base grammar (e.g., for building Ent and Rel) to create a per-example grammar $\mathcal{R} \cup \mathcal{T}$ for the utterance x. We use this grammar to generate logical forms using standard beam search. We follow Section 2.3 to generate a set of candidate logical forms $\mathcal{Z}$ and update model parameters.

However, we deviate from Section 2.3 in one way. Given a set $\mathcal{Z}$ of candidate logical forms for some training example (x_i, w_i, y_i), we pick the logical form z_i^+ with the highest model probability among consistent logical forms, and pick z_i^- with the highest model probability among inconsistent logical forms, then perform a gradient update on the objective function:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \left[\log \frac{p_i^+}{p_i^+ + p_i^-} \right] - \lambda \|\theta\|_1, \quad (9)$$

$$\text{where} \quad \begin{aligned} p_i^+ &= p_\theta(z_i^+ \mid x_i, w_i) \\ p_i^- &= p_\theta(z_i^- \mid x_i, w_i). \end{aligned}$$

Compared to (6), this objective function only considers the top consistent and inconsistent logical forms for each example instead of all candidate logical forms. Empirically, we found that optimizing (9) gives a 2% gain in prediction accuracy compared to optimizing (6).

3.6 Updating the macro grammar

If the triggered macro rules fail to find a consistent logical form, we fall back to performing a beam search on the base grammar. For efficiency, we stop the search either when a consistent logical form is found, or when the total number of generated logical forms exceeds a threshold T. The two stopping criteria prevent the search algorithm from spending too much time on a complex example. We might miss consistent logical forms on such examples, but because the base grammar is only used for generating macro rules, not for updating model parameters, we might be able to induce the same macro rules from other examples. For instance, if an example has an utterance phrase that matches too many knowledge base entries, it would be more efficient to skip the example; the macro that would have been extracted from this example can be extracted from less ambiguous examples with the same question type. Such omissions are not completely disastrous, and can speed up training significantly.

When the algorithm succeeds in finding a consistent logical form z using the base grammar, we derive its macro M following Section 3.1, then construct macro rules following Section 3.3. These macro rules are added to the macro grammar. We also associate the utterance x with the consistent logical form z, so that the macro rules that generate z can be triggered by other examples. Parameters of the semantic parser are not updated in this case.

3.7 Prediction

At test time, we follow Steps 1–2 of Algorithm 1 to generate a set $\mathcal{Z}$ of candidate logical forms from the triggered macro rules, and then output the highest-scoring logical form in $\mathcal{Z}$. Since the base grammar is never used at test time, prediction is generally faster than training.

4 Experiments

We report experiments on the WIKITABLEQUESTIONS dataset (Pasupat and Liang, 2015). Our algorithm is compared with the parser trained only with the base grammar, the floating parser of Pasupat and Liang (2015) (PL15), the Neural Programmer parser (Neelakantan et al., 2016) and the Neural Multi-Step Reasoning parser (Haug et al., 2017). Our algorithm not only outperforms the others, but also achieves an order-of-magnitude speedup over the parser trained with the base grammar and the parser in PL15.

4.1 Setup

The dataset contains 22,033 complex questions on 2,108 Wikipedia tables. Each question comes with a table, and the tables during evaluation are disjoint from the ones during training. The training and test sets contain 14,152 and 4,344 examples respectively.[2] Following PL15, the development accuracy is averaged over the first three 80-20 training data splits given in the dataset package. The test accuracy is reported on the train-test data split.

"Which driver appears the most?" $\texttt{argmax}(\mathbf{R}[\texttt{Driver}].\texttt{Type.Row}, \mathbf{R}[\lambda\texttt{x.count}(\texttt{Driver.x})])$
"What language was spoken more during the Olympic oath, English or French?" $\texttt{argmax}(\texttt{English} \sqcup \texttt{French}, \mathbf{R}[\lambda\texttt{x.count}(\texttt{Language.x})])$
"Who is taller, Rose or Tim?" $\texttt{argmax}(\texttt{Rose} \sqcup \texttt{Tim}, \mathbf{R}[\lambda\texttt{x}.\mathbf{R}[\texttt{Num}].\mathbf{R}[\texttt{Height}].\texttt{Name.x})])$

Table 3: Several example logical forms our grammar can generate that are not covered by PL15.

We use the same features and logical form pruning strategies as PL15, but generalize their base grammar. To control the search space, the actual system in PL15 restricts the superlative operators `argmax` and `argmin` to be applied only on the set of table rows. We allow these operators to be applied on the set of tables cells as well, so that the grammar captures certain logical forms that are not covered by PL15 (see Table 3). Additionally, for terminal rule (3), we allow $f(span)$ to produce entities that approximately match the token span in addition to exact matches. For example, the phrase "Greenville" can trigger both entities `Greenville_Ohio` and `Greensville`.

We chose hyperparameters using the first train-dev split. The beam size B of beam search is chosen to be $B = 100$. The K-nearest neighbor parameter is chosen as $K = 40$. Like PL15, our algorithm takes 3 passes over the dataset for training. The maximum number of logical forms generated in step 6 of Algorithm 1 is set to $T = 5{,}000$ for the first pass. For subsequent passes, we set $T = 0$ (i.e., never fall back to the base grammar) so that we stop augmenting the macro grammar. During the first pass, Algorithm 1 falls back to the base grammar on roughly 30% of the training examples.

For training the baseline parser that only relies on the base grammar, we use the same beam size $B = 100$, and take 3 passes over the dataset for training. There is no maximum constraint on the

[2] The remaining 3,537 examples were not included in the original data split.

	Dev	Test
Pasupat and Liang (2015)	37.0%	37.1%
Neelakantan et al. (2016)	37.5%	37.7%
Haug et al. (2017)	-	38.7%
This paper: base grammar	**40.6%**	42.7%
This paper: macro grammar	40.4%	**43.7%**

Table 4: Results on WIKITABLEQUESTIONS.

number of logical forms that can be generated for each example.

4.2 Coverage of the macro grammar

With the base grammar, our parser generates 13,700 partial logical forms on average for each training example, and hits consistent logical forms on 81.0% of the training examples. With the macro rules from holistic triggering, these numbers become 1,300 and 75.6%. The macro rules generate much fewer partial logical forms, but at the cost of slightly lower coverage.

However, these coverage numbers are computed based on finding any logical form that executes to the correct denotation. This includes spurious logical forms, which do not reflect the semantics of the question but are coincidentally consistent with the correct denotation. (For example, the question "*Who got the same number of silvers as France?*" on Table 1 might be spuriously parsed as $\mathbf{R}[\texttt{Nation}].\mathbf{R}[\texttt{Next}].\texttt{Nation}.\texttt{France}$, which represents the nation listed after France.) To evaluate the "true" coverage, we sample 300 training examples and manually label their logical forms. We find that on 48.7% of these examples, the top consistent logical form produced by the base grammar is semantically correct. For the macro grammar, this ratio is also 48.7%, meaning that the macro grammar's effective coverage is as good as the base grammar.

The macro grammar extracts 123 macros in total. Among the 75.6% examples that were covered by the macro grammar, the top 34 macros cover 90% of consistent logical forms. By examining the top 34 macros, we discover explicit semantic meanings for 29 of them, which are described in detail in the supplementary material.

4.3 Accuracy and speedup

We report prediction accuracies in Table 4. With a more general base grammar (additional superlatives and approximate matching), and by optimizing the objective function (9), our base parser outperforms PL15 (42.7% vs 37.1%). Learning a macro grammar slightly improves the accuracy to 43.7% on the test set. On the three train-dev splits, the averaged accuracy achieved by the base grammar and the macro grammar are close (40.6% vs 40.4%).

		Time (ms/ex)	
	Acc.	Train	Pred
PL15	37.0%	619	645
Ours: base grammar	40.6%	1,117	1,150
Ours: macro grammar	40.4%	99	70
no holistic triggering	40.1%	361	369
no macro decomposition	40.3%	177	159

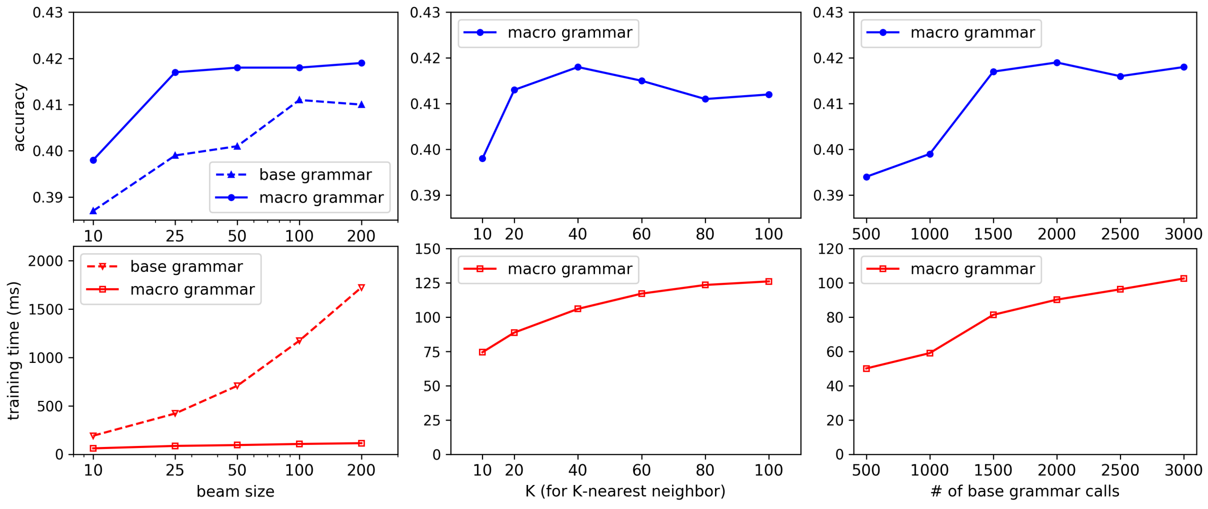
Table 5: Comparison and ablation study: the columns report averaged prediction accuracy, training time, and prediction time (milliseconds per example) on the three train-dev splits.

In Table 5, we compare the training and prediction time of PL15 as well as our parsers. For a fair comparison, we trained all parsers using the SEMPRE toolkit (Berant et al., 2013) on a machine with Xeon 2.6GHz CPU and 128GB memory without parallelization. The time for constructing the macro grammar is included as part of the training time. Table 5 shows that our parser with the base grammar is more expensive to train than PL15. However, training with the macro grammar is substantially more efficient than training with only the base grammar— it achieves 11x speedup for training and 16x speedup for test time prediction.

We run two ablations of our algorithm to evaluate the utility of holistic triggering and macro decomposition. The first ablation triggers all macro rules for parsing every utterance without holistic triggering, while the second ablation constructs Rule (7) for every macro without decomposing it into smaller rules. Table 5 shows that both variants result in decreased efficiency. This is because holistic triggering effectively prunes irrelevant macro rules, while macro decomposition is important for efficient beam search and featurization.

4.4 Influence of hyperparameters

Figure 2a shows that for all beam sizes, training with the macro grammar is more efficient than training with the base grammar, and the speedup rate grows with the beam size. The test time ac-



(a) Varying beam size

(b) Varying neighbor size

(c) Varying base grammar usage count

Figure 2: Prediction accuracy and training time (per example) with various hyperparameter choices, reported on the first train-dev split.

curacy of the macro grammar is robust to varying beam sizes as long as $B \geq 25$.

Figure 2b shows the influence of the neighbor size K. A smaller neighborhood triggers fewer macro rules, leading to faster computation. The accuracy peaks at $K = 40$ then decreases slightly for large K. We conjecture that the smaller number of neighbors acts as a regularizer.

Figure 2c reports an experiment where we limit the number of fallback calls to the base grammar to m. After the limit is reached, subsequent training examples that require fallback calls are simply skipped. This limit means that the macro grammar will get augmented at most m times during training. We find that for small m, the prediction accuracy grows with m, implying that building a richer macro grammar improves the accuracy. For larger m, however, the accuracies hardly change. According to the plot, a competitive macro grammar can be built by calling the base grammar on less than 15% of the training data.

Based on Figure 2, we can trade accuracy for speed by choosing smaller values of (B, K, m). With $B = 50$, $K = 40$ and $m = 2000$, the macro grammar achieves a slightly lower averaged development accuracy (40.2% rather than 40.4%), but with an increased speedup of 15x (versus 11x) for training and 20x (versus 16x) for prediction.

5 Related work and discussion

A traditional semantic parser maps natural language phrases into partial logical forms and composes these partial logical forms into complete logical forms. Parsers define composition based on a grammar formalism such as Combinatory Categorial Grammar (CCG) (Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2011, 2013; Kushman and Barzilay, 2013; Krishnamurthy and Kollar, 2013), Synchronous CFG (Wong and Mooney, 2007), and CFG (Kate and Mooney, 2006; Chen and Mooney, 2011; Berant et al., 2013; Desai et al., 2016), while others use the syntactic structure of the utterance to guide composition (Poon and Domingos, 2009; Reddy et al., 2016). Recent neural semantic parsers allow any sequence of logical tokens to be generated (Dong and Lapata, 2016; Jia and Liang, 2016; Kociský et al., 2016; Neelakantan et al., 2016; Liang et al., 2017; Guu et al., 2017). The flexibility of these composition methods allows arbitrary logical forms to be generated, but at the cost of a vastly increased search space.

Whether we have annotated logical forms or not has dramatic implications on what type of approach will work. When logical forms are available, one can perform grammar induction to mine grammar rules *without search* (Kwiatkowski et al., 2010). When only annotated denotations are available, as in our setting, one must use a base grammar to define the output space of logical forms. Usually these base grammars come with many restrictions to guard against combinatorial explosion (Pasupat and Liang, 2015).

Previous work on higher-order unification for lexicon induction (Kwiatkowski et al., 2010) using factored lexicons (Kwiatkowski et al., 2011) also learns logical form macros with an online algorithm. The result is a lexicon where each entry contains a logical form template and a set of possible phrases for triggering the template. In contrast, we have avoided binding grammar rules to particular phrases in order to handle lexical variations. Instead, we use a more flexible mechanism—holistic triggering—to determine which rules to fire. This allows us to generate logical forms for utterances containing unseen lexical paraphrases or where the triggering is spread throughout the sentence. For example, the question "*Who is X, John or Y*" can still trigger the correct macro extracted from the last example in Table 3 even when *X* and *Y* are unknown words.

Our macro grammars bears some resemblance to adaptor grammars (Johnson et al., 2006) and fragment grammars (O'Donnell, 2011), which are also based on the idea of caching useful chunks of outputs. These generative approaches aim to solve the modeling problem of assigning higher probability mass to outputs that use reoccurring parts. In contrast, our learning algorithm uses caching as a way to constrain the search space for computational efficiency; the probabilities of the candidate outputs are assigned by a separate discriminative model. That said, the use of macro grammars does have a small positive modeling contribution, as it increases test accuracy from 42.7% to 43.7%.

An orthogonal approach for improving search efficiency is to adaptively choose which part of the search space to explore. For example, Berant and Liang (2015) uses imitation learning to strategically search for logical forms. Our holistic triggering method, which selects macro rules based on the similarity of input utterances, is related to the use of paraphrases (Berant and Liang, 2014; Fader et al., 2013) or string kernels (Kate and Mooney, 2006) to train semantic parsers. While the input similarity measure is critical for scoring logical forms in these previous works, we use the measure only to retrieve candidate rules, while scoring is done by a separate model. The retrieval bar means that our similarity metric can be quite crude.

6 Summary

We have presented a method for speeding up semantic parsing via macro grammars. The main source of efficiency is the decreased size of the logical form space. By performing beam search on a few macro rules associated with the K-nearest neighbor utterances via holistic triggering, we have restricted the search space to semantically relevant logical forms. At the same time, we still maintain coverage over the base logical form space by occasionally falling back to the base grammar and using the consistent logical forms found to enrich the macro grammar. The higher efficiency allows us expand the base grammar without having to worry much about speed: our model achieves a state-of-the-art accuracy while also enjoying an order magnitude speedup.

Acknowledgements. We gratefully acknowledge Tencent for their support on this project.

Reproducibility. Code, data, and experiments for this paper are available on CodaLab platform: https://worksheets.codalab.org/worksheets/0x4d6dbfc5ec7f44a6a4da4ca2a9334d6e/.

References

Y. Artzi and L. Zettlemoyer. 2013. UW SPF: The University of Washington semantic parsing framework. *arXiv preprint arXiv:1311.3011*.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Berant and P. Liang. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*.

J. Berant and P. Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics (TACL)*, 3:545–558.

D. L. Chen and R. J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 859–865.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Computational Natural Language Learning (CoNLL)*, pages 18–27.

A. Desai, S. Gulwani, V. Hingorani, N. Jain, A. Karkare, M. Marron, S. R, and S. Roy. 2016. Program synthesis using natural language. In *International Conference on Software Engineering (ICSE)*, pages 345–356.

L. Dong and M. Lapata. 2016. Language to logical form with neural attention. In *Association for Computational Linguistics (ACL)*.

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

A. Fader, L. Zettlemoyer, and O. Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Association for Computational Linguistics (ACL)*.

K. Guu, P. Pasupat, E. Z. Liu, and P. Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Association for Computational Linguistics (ACL)*.

T. Haug, O. Ganea, and P. Grnarova. 2017. Neural multi-step reasoning for question answering on semi-structured tables. *arXiv preprint arXiv:1702.06589*.

P. Indyk. 2004. Approximate nearest neighbor under edit distance via product metrics. In *Symposium on Discrete Algorithms (SODA)*, pages 646–650.

R. Jia and P. Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*.

M. Johnson, T. Griffiths, and S. Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 641–648.

R. J. Kate and R. J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*, pages 913–920.

T. Kociský, G. Melis, E. Grefenstette, C. Dyer, W. Ling, P. Blunsom, and K. M. Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1078–1087.

J. Krishnamurthy and T. Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics (TACL)*, 1:193–206.

N. Kushman and R. Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, pages 826–836.

T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Empirical Methods in Natural Language Processing (EMNLP)*.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1223–1233.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1512–1523.

C. Liang, J. Berant, Q. Le, and K. D. F. N. Lao. 2017. Neural symbolic machines: Learning semantic parsers on Freebase with weak supervision. In *Association for Computational Linguistics (ACL)*.

P. Liang. 2013. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408*.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599.

A. Neelakantan, Q. V. Le, and I. Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *International Conference on Learning Representations (ICLR)*.

T. J. O'Donnell. 2011. *Productivity and Reuse in Language*. Ph.D. thesis, Massachusetts Institute of Technology.

P. Pasupat and P. Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Association for Computational Linguistics (ACL)*.

H. Poon and P. Domingos. 2009. Unsupervised semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*.

S. Reddy, O. Täckström, M. Collins, T. Kwiatkowski, D. Das, M. Steedman, and M. Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. In *Association for Computational Linguistics (ACL)*, pages 127–140.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for Computational Linguistics (ACL)*, pages 960–967.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 678–687.

A Continuously Growing Dataset of Sentential Paraphrases

Wuwei Lan[1], Siyu Qiu[2], Hua He[3], Wei Xu[1]
[1] Department of Computer Science and Engineering
Ohio State University
{lan.105, xu.1265}@osu.edu
[2] Computer and Information Science Department
University of Pennsylvania
siqiu@seas.upenn.edu
[3] Department of Computer Science
University of Maryland, College Park
huah@umd.edu

Abstract

A major challenge in paraphrase research is the lack of parallel corpora. In this paper, we present a new method to collect large-scale sentential paraphrases from Twitter by linking tweets through shared URLs. The main advantage of our method is its simplicity, as it gets rid of the classifier or human in the loop needed to select data before annotation and subsequent application of paraphrase identification algorithms in the previous work. We present the largest human-labeled paraphrase corpus to date of 51,524 sentence pairs and the first cross-domain benchmarking for automatic paraphrase identification. In addition, we show that more than 30,000 new sentential paraphrases can be easily and continuously captured every month at ∼70% precision, and demonstrate their utility for downstream NLP tasks through phrasal paraphrase extraction. We make our code and data freely available.[1]

1 Introduction

A paraphrase is a restatement of meaning using different expressions (Bhagat and Hovy, 2013). It is a fundamental semantic relation in human language, as formalized in the Meaning-Text linguistic theory which defines meaning as 'invariant of paraphrases' (Milićević, 2006). Researchers have shown benefits of using paraphrases in a wide range of applications (Madnani and Dorr, 2010), including question answering (Fader et al., 2013), semantic parsing (Berant and Liang, 2014), information extraction (Sekine, 2006; Zhang et al., 2015), machine translation (Mehdizadeh Seraj et al., 2015), textual entailment (Dagan et al., 2006; Bjerva et al., 2014; Marelli et al., 2014; Izadinia et al., 2015), vector semantics (Faruqui et al., 2015; Wieting et al., 2015), and semantic textual similarity (Agirre et al., 2015; Li and Srikumar, 2016). Studying paraphrases in Twitter can also help track unfolding events (Vosoughi and Roy, 2016) or the spread of information (Bakshy et al., 2011) on social networks.

In this paper, we address a major challenge in paraphrase research — the lack of parallel corpora. There are **only two** publicly available datasets of naturally occurring sentential paraphrases and non-paraphrases:[2] the MSRP corpus derived from clustered news articles (Dolan and Brockett, 2005) and the PIT-2015 corpus from Twitter trending topics (Xu et al., 2014, 2015). Our goal is not only to create a new annotated paraphrase corpus, but to identify a new data source and method that can narrow down the search space of paraphrases without using the classifier-biased or human-in-the-loop data selection as in MSRP and PIT-2015. This is so that sentential paraphrases can be conveniently and continuously harvested in large quantities to benefit downstream applications.

We present an effective method to collect sentential paraphrases from tweets that refer to the same URL and contribute a new gold-standard annotated corpus of 51,524 sentence pairs, which is the largest to date (Table 1). We show the different characteristics of this new dataset contrasting the two existing corpora through the first system-

[1] The code and data can be obtained from the first and last author's websites.

[2] Meaningful non-paraphrases (pairs of sentences that have similar wordings or topics but different meanings, and that are not randomly or artificially generated) have been very difficult to obtain but are very important, because they serve as necessary distractors in training and evaluation.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1224–1234
Copenhagen, Denmark, September 7–11, 2017.

Name	Genre	Size	Sentence Length	Multi-Ref.	Non-Para.
MSR Paraphrase Corpus (MSRP)	news	5801 pairs	18.9 words	no	yes
Twitter Paraphrase Corpus (PIT-2015)	Twitter	18,762 pairs	11.9 words	some	yes
Twitter News URL Corpus (this work)	Twitter	44,365 pairs	14.8 words	yes	yes
MSR Video Description Corpus	YouTube	123,626 sentences	7.03 words	yes	no

Table 1: Summary of publicly available large sentential paraphrase corpora with manual quality assurance. Our Twitter News URL Corpus has the advantages of including both meaningful non-paraphrases (Non-Para.) and multiple references (Multi-Ref.), which are important for training paraphrase identification and evaluating paraphrase generation, respectively.

atic study of paraphrase identification across multiple datasets. Our new corpus is complementary to previous work, as the corpus contains multiple references of both formal well-edited and informal user-generated texts. This is also the first work that provides a continuously growing collection, with more than 30,000 new sentential paraphrases per month automatically labeled at ∼70% precision. We demonstrate that up-to-date phrasal paraphrases can then be extracted via word alignment (see examples in Table 2). We plan to continue collecting paraphrases using our method and release a constantly updating paraphrase resource.

a 15-year-old girl, a 15yr old, a 15 y/o girl
fetuses, fetal tissue, miscarried fetuses
responsible for, guilty of, held liable for, liable for
UVA administrator, UVa official, U-Va. dean, University of Virginia dean
FBI Director backs CIA finding, FBI agrees with CIA, FBI backs CIA view, FBI finally backs CIA view, FBI now backs CIA view, FBI supports CIA assertion, FBI-Clapper back CIA's view, The FBI backs the CIA's assessment, FBI Backs CIA,
Donald Trump, DJT, Mr Trump, Donald @realTrump, D*nald Tr*mp, Comrade #Trump, GOPTrump, Pres-elect Trump, President-Elect Trump, President-elect Donald J. Trump, PEOTUS Trump, He-Who-Must-Not-Be-Named[3]

Table 2: Up-to-date phrasal paraphrases automatically extracted from Twitter with our new method.

2 Existing Paraphrase Corpora and Their Limitations

To date, there exist only two publicly available corpora of both sentential paraphrases and non-paraphrases:

MSR Paraphrase Corpus [MSRP] (Dolan et al., 2004; Dolan and Brockett, 2005) This corpus contains 5,801 pairs of sentences from news articles, with 4,076 for training and the remaining 1,725 for testing. It was created from clustered news articles by using an SVM classifier (using features including string similarity and WordNet synonyms) to gather likely paraphrases, then annotated by human on semantic equivalence. The MSRP corpus has a known deficiency skewed toward over-identification (Das and Smith, 2009), because the "purpose was not to evaluate the potential effectiveness of the classifier itself, but to identify a reasonably large set of both positive and plausible 'near-miss' negative examples" (Dolan and Brockett, 2005). It contains a large portion of sentence pairs with many ngrams shared in common.

Twitter Paraphrase Corpus [PIT-2015] (Xu et al., 2014, 2015) This corpus was derived from Twitter's trending topic data. The training set contains 13,063 sentence pairs on 400 distinct topics, and the test set contains 972 sentence pairs on 20 topics. As numerous Twitter users spontaneously talk about varied topics, this dataset contains many lexically divergent paraphrases. However, this method requires a manual step of selecting topics to ensure the quality of collected paraphrases, because many topics detected automatically are either incorrect or too broad. For example, the topic "New York" relates to tweets with a wide range of information and cannot narrow the search space down enough for human annotation and the subsequent application of classification algorithms.

3 Constructing the Twitter URL Paraphrase Corpus

For paraphrase acquisition, it has been crucial to find a simple and effective way to locate paraphrase candidates (see related work in Section 6). We show the efficacy of tracking URLs in Twitter. This method does not rely on automatic news clustering as in MSRP or topic detection as in PIT-2015, but it keeps collecting good candidate paraphrase pairs in large quantities.

[3] Another 12 name variations are omitted in the paper due to their offensive nature.

Twitter News URL Corpus	
Original Tweet	Samsung halts production of its Galaxy Note 7 as battery problems linger
Paraphrase	#Samsung temporarily suspended production of its Galaxy #Note7 devices following reports
	News hit that @Samsung is temporarily halting production of the #GalaxyNote7.
	Samsung still having problems with their Note 7 battery overheating. Completely halt production.
	SAMSUNG HALTS PRODUCTS OF GALAXY NOTE 7 . THE BATTERIES ARE * STILL * EXPLODING .
Non-Paraphrase	in which a phone bonfire in 1995–a real one–is a metaphor for samsung's current note 7 problems
	samsung decides, "if we don't build it, it won't explode."
	Samsung's Galaxy Note 7 Phones AND replacement phones have been going up in flames due to the defective batteries

Table 3: A representative set of tweets linked by a URL originated from news agencies (this work).

Twitter Streaming URL Data	1 dasviness louistomlinson overhears harrystyles on the phone
	when she likes tall guys ??? ??? vine by justjamiie
	shineeasvines jonghyun when he wears shoe lifts
	idaliaorellana kimmvanny ladyfea 21 hahaha if he does it he needs heels

Table 4: A representative set of tweets linked by a URL in streaming data (generally poor readability).

3.1 Data Source: News Tweets vs. Streaming

We extracted the embedded URL in each tweet and used Twitter's Search API to retrieve all tweets that contain the same URL. Some tweets use shortened URLs, which we resolve as full URLs. We tracked 22 English news accounts in Twitter to create the paraphrase corpus in this paper (see examples in Table 3). We will extend the corpus to include other languages and domains in future work.

As shown in Table 5, nearly all the tweets posted by news agencies have embedded URLs. About 51.17% of posts contain two URLs, usually one pointing to a news article and the other to media such as a photo or video. Although close to half of the tweets in Twitter streaming data[4] contain at least one URL, most of them are very hard to read (see examples in Table 4).

Data Source	**tweets w/o url**	**avg #url per tweet**	**avg # url (news) per tweet**
Streaming Data	55.8%	0.52	
@nytimes	1.2%	1.31	0.988
@cnnbrk	0.0%	1.17	1
@BBCBreaking	1.0%	1.32	0.99
@CNN	0.0%	1.85	1
@ABC	1.7%	1.26	0.983
@NBCNews	1.1%	1.63	0.989

Table 5: Statistics of tweets in Twitter's streaming data and news account data. Many tweets contain more than one URL because media such as photo or video is also represented by URLs.

3.2 Filtering of Retweets

Retweeting is an important feature in Twitter. There are two types: automatic and manual retweets. An automatic retweet is done by clicking the retweet button on Twitter and is easy to remove using the Twitter API. A manual retweet occurs when the user creates a new tweet by copying and pasting the original tweet and possibly adding some extras, such as hashtags, usernames or comments. It is crucial to remove these redundant tweets with minor variations, which otherwise represent a significant portion of the data (Table 6). We preprocessed the tweets using a tokenizer[5] (Gimpel et al., 2011) and an in-house sentence splitter. We then filtered out manual retweets using a set of rules, checking if one tweet was a sub- string of the other, or if it only differed in punctuation, or the contents of the "twitter:title" or "twitter:description" tag in the linked HTML file of the news article.

Table 6 shows the effectiveness of the filtering. We used PINC, a standard paraphrase metric, to measure ngram-based dissimilarity (Chen and Dolan, 2011), and Jaccard metric to measure token-based string similarity (Jaccard, 1912). After filtering, the dataset contains tweets with more significant rephrasing as indicated by higher PINC and lower Jaccard scores.

	avg #tweets (STD)	**avg PINC**	**avg Jaccard**
before filtering	205.51 (219.66)	0.6153	0.3635
after filtering	74.75 (94.39)	0.7603	0.2515

Table 6: Impact of filtering of manual retweets.

[4] We used Twitters Streaming API which provided a real-time stream of public tweets posted on Twitter.

[5] `http://www.cs.cmu.edu/~ark/TweetNLP/`

3.3 Gold Standard Corpus

To get the gold-standard paraphrase corpus, we obtained human labels on Amazon Mechanical Turk. We showed annotators an original sentence, and asked them to select sentences with the same meaning from 10 candidate sentences. For each question, we recruited 6 annotators and paid $0.03 to each worker.[6] On average, each question took about 53 seconds to finish. For each sentence pair, we aggregated the paraphrase and non-paraphrase labels using the majority vote.

We constructed the largest gold standard paraphrase corpus to date, with 42,200 tweets of 4,272 distinct URLs annotated in the training set and 9,324 tweets of 915 distinct URLs in the test set. The training data was collected between 10/10/2016 and 11/22/2016, and testing data between 01/09/2017 and 01/19/2017. In Section 4, we contrast the characteristics of our data against existing paraphrase corpora.

Quality Control We evaluated the annotation quality of each worker using Cohen's kappa agreement (Artstein and Poesio, 2008) against the majority vote of other workers. We asked the best workers (the top 528 out of 876) to label more data by republishing the questions done by workers with low reliability (Cohen's kappa <0.4).

Inter-Annotator Agreement In addition, we had 300 sampled sentence pairs independently annotated by an expert. The annotated agreement is 0.739 by Cohen's kappa between the expert and the majority vote of 6 crowdsourcing workers. If we assume the expert annotation is gold, the precision of worker vote is 0.871, the recall is 0.787, and F1 is 0.827, similar to those of PIT-2015.

3.4 Continuous Harvesting of Sentential Paraphrases

Since our method directly applies to raw tweets, it can continuously extract sentential paraphrases from Twitter. In Section 4, we show that this approach can produce a silver-standard paraphrase corpus at about 70% precision that grows by more than 30,000 new sentential paraphrases *per month*. Section 5 presents experiments demonstrating the utility of these automatically identified sentential paraphrases.

4 Comparison of Paraphrase Corpora

Though paraphrasing has been widely studied, supporting analyses and experiments have thus far often only been conducted on a single dataset. In this section, we present a comparative analysis of our newly constructed gold-standard corpus with two existing corpora by 1) individually examining the instances of paraphrase phenomena and 2) benchmarking a range of automatic paraphrase identification approaches.

4.1 Paraphrase Phenomena

In order to show the differences across these three datasets, we sampled 100 sentential paraphrases from each training set and counted occurrences of each phenomenon in the following categories: Elaboration (textual pairs can differ in total information content, such as *Trump's ex-wife Ivana* and *Ivana Trump*), Phrasal (alternates of phrases, such as *taking over* and *replaces*), Spelling (spelling variants, such as *Trump* and *Trumpf*), Synonym (such as *said* and *told*), Anaphora (a full noun phrase in one sentence that corresponds to the counterpart, such as *@MarkKirk* and *Kirk*) and Reordering (when a word, phrase or the whole sentence reorders, or even logically reordered, such as *Matthew Fishbein questioned him* and *under questioning by Matthew Fishbein*). We report the average number of occurrences of each paraphrase type per sentence pair for each corpus in Table 7. As sentences tend to be longer in MSRP and shorter in PIT-2015, we also normalized the numbers by the length of sentences to be more comparable to the URL dataset.

These three datasets exhibit distinct and complementary compositions of paraphrase phenom-

per sentence	**MSRP**	**PIT-2015**	**URL**
Elaboration	0.60	0.23	**0.79**
Spelling	0.17	0.13	**0.35**
Synonym	**0.26**	0.10	0.13
Phrasal	0.42	**0.56**	0.35
Anaphora	0.27	0.08	**0.33**
Reordering	**0.53**	0.33	0.49
adjusted by sent length*	**MSRP***	**PIT-2015***	**URL**
Elaboration	0.42	0.36	**0.79**
Spelling	0.12	0.21	**0.35**
Synonym	**0.18**	0.16	0.13
Phrasal	0.29	**0.89**	0.35
Anaphora	0.19	0.13	**0.33**
Reordering	0.37	**0.52**	0.49

Table 7: Mean number of instances of paraphrase phenomena per sentence pair across three corpora.

[6] The low pricing helps to not attract spammers to this easy-to-finish task. We gave bonus to workers based on quality and the average hourly pay for each worker is about $7.

ena. MSRP has more synonyms, because authors of different news articles may use different and rather sophisticated words. PIT-2015 contains many phrasal paraphrases, probably due to the fact that most tweets under the same trending topic are written spontaneously and independently. Our URL dataset shows more elaboration, spelling and anaphora paraphrase phenomena, showing that many URL-embedded tweets are created by users with a conscious intention to rephrase the original news headline.

4.2 Automatic Paraphrase Identification

We provide a benchmark on paraphrase identification to better understand various models, as well as the characteristics of our new corpus compared to the existing ones. We focus on binary classification of paraphrase/non-paraphrase, and report the maximum F1 measure of any point on the precision-recall curve.

4.2.1 Models

We chose several representative technical approaches for automatic paraphrase identification:

GloVe (Pennington et al., 2014) This is a word representation model trained on aggregated global word-word co-occurrence statistics from a corpus. We used 300-dimensional word vectors trained on Common Crawl and Twitter, summed the vectors for each sentence, and computed the cosine similarity.

LR The logistic regression (LR) model incorporates 18 features based on 1-3 gram overlaps between two sentences (s_1 and s_2) (Das and Smith, 2009). The features are of the form precision$_n$ (number of n-gram matches divided by the number of n-grams in s_1), recall$_n$ (number of n-gram matches divided by the number of n-grams in s_2), and F$_n$ (harmonic mean of recall and precision). The model also includes lemmatized versions of these features.

WMF/OrMF Weighted Matrix Factorization (WMF) (Guo and Diab, 2012) is an unsupervised latent space model. The unobserved words are carefully handled, which results in more robust embeddings for short texts. Orthogonal Matrix Factorization (OrMF) (Guo et al., 2014) is the extension of WMF, with an additional objective to obtain nearly orthogonal dimensions in matrix factorization to discount redundant information.

Method	F1	Precision	Recall
Random	0.799	0.665	1.0
Edit Distance	0.799	0.666	1.0
GloVe	0.812	0.707	0.952
LR	0.829	0.741	0.941
WMF (vec)	0.817	0.713	0.956
LEX-WMF (vec)	**0.836**	**0.751**	**0.943**
OrMF (vec)	0.820	0.733	0.930
LEX-OrMF (vec)	**0.833**	**0.741**	**0.950**
WMF (sim)	0.812	0.728	0.918
LEX-WMF (sim)	0.831	0.732	0.962
OrMF (sim)	0.815	0.699	0.976
LEX-OrMF (sim)	0.832	0.735	0.958
MultiP	0.800	0.667	0.998
DeepPairwiseWord	**0.834**	**0.763**	**0.919**

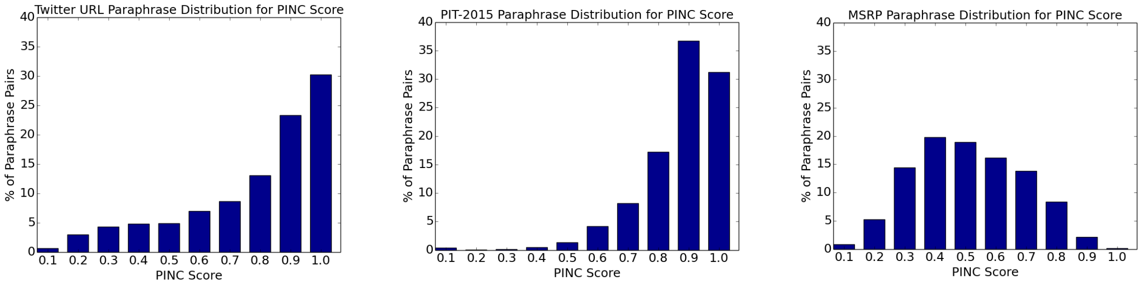
Table 8: Paraphrase models in the MSR Paraphrase Corpus (MSRP). The bold font in the table represents top three models in the dataset.

Method	F1	Precision	Recall
Random	0.346	0.209	1.0
Edit Distance	0.363	0.236	0.789
GloVe	0.484	0.396	0.617
LR	0.645	0.669	0.623
WMF (vec)	0.594	0.681	0.526
LEX-WMF (vec)	0.635	0.655	0.617
OrMF (vec)	0.594	0.681	0.526
LEX-OrMF (vec)	0.638	0.579	0.709
WMF (sim)	0.553	0.570	0.537
LEX-WMF (sim)	**0.651**	**0.657**	**0.646**
OrMF (sim)	0.563	0.591	0.537
LEX-OrMF (sim)	0.644	0.632	0.657
MultiP	**0.721**	**0.705**	**0.737**
DeepPairwiseWord	**0.667**	**0.725**	**0.617**

Table 9: Paraphrase models in the Twitter Paraphrase Corpus (PIT-2015).

Method	F1	Precision	Recall
Random	0.327	0.195	1.000
Edit Distance	0.526	0.650	0.442
GloVe	0.583	0.607	0.560
LR	0.683	0.669	0.698
WMF (vec)	0.660	0.640	0.680
LEX-WMF (vec)	**0.693**	**0.687**	**0.698**
OrMF (vec)	0.662	0.625	0.703
LEX-OrMF (vec)	**0.691**	**0.709**	**0.674**
WMF (sim)	0.659	0.595	0.738
LEX-WMF (sim)	0.688	0.632	0.754
OrMF (sim)	0.660	0.690	0.632
LEX-OrMF (sim)	0.688	0.630	0.758
MultiP	0.536	0.386	0.875
DeepPairwiseWord	**0.749**	**0.803**	**0.702**

Table 10: Paraphrase models in Twitter URL Corpus (this work).



(a) Twitter URL

(b) PIT-2015

(c) MSRP

Figure 1: Comparison of ngram dissimilarity (PINC score) in sentential paraphrases across three corpora. The MSRP contains sentential paraphrases with more ngram overlaps (low PINC). Our URL corpus and PIT-2015 contain more lexically divergent paraphrases (high PINC).

Specifically, for the **(vec)** version, vectors of a pair of sentences $\vec{v_1}$ and $\vec{v_2}$ are converted into one feature vector, $[\vec{v_1} + \vec{v_2}, |\vec{v_1} - \vec{v_2}|]$, by concatenating the element-wise sum $\vec{v_1} + \vec{v_2}$ and absolute difference $|\vec{v_1} - \vec{v_2}|$. We also provide the **(sim)** variation, which directly uses the single cosine similarity score between two sentence vectors.

LEX-WMF/LEX-OrMF This is an open-sourced adaptation (Xu et al., 2014) of LEXDISCRIM (Ji and Eisenstein, 2013) that have shown comparable performance. It combines WMF/OrMF with n-gram overlapping features to train a LR classifier.

MultiP MultiP (Xu et al., 2014) is a multi-instance learning model suited for short messages on Twitter. The at-least-one-anchor assumption in this model looks for two sentences that have a topical phrase in common, plus at least one pair of anchor words that carry a similar key meaning. This model achieved the best performance in the PIT-2015 (Xu et al., 2014) dataset.

DeepPairwiseWord He et al. (2016) developed a deep neural network model that focuses on important pairwise word interactions across input sentences. This model innovates in proposing a similarity focus layer and a 19-layer very deep convolutional neural network to guide model attention to important word pairs. It has shown state-of-the-art performance on several textual similarity measurement datasets.

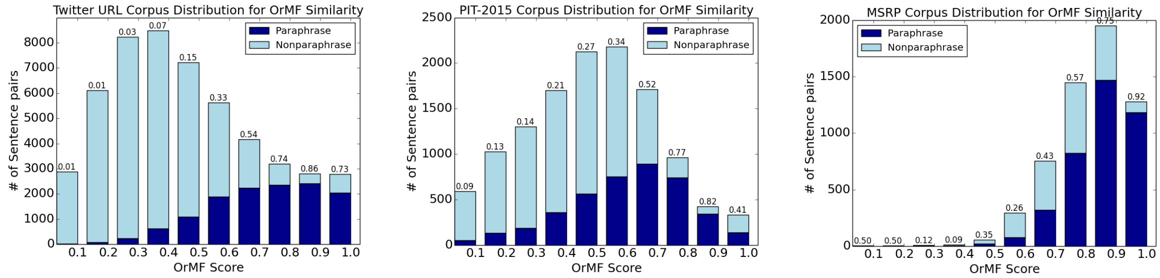
4.2.2 Model Performance and Dataset Difference

The results on three benchmark paraphrase corpora are shown in Table 8, 9 and 10. The random baseline reflects that close to 80% sentence pairs are paraphrases in the MSPR corpus. This is atypical in the real-world text data and may cause falsely positive predictions.

Both the edit distance and the LR models exploit surface word features. In particular, the LR model that uses lemmatization and ngram overlap features achieves very competitive performance on all datasets. Figure 1 shows a closer look at ngram differences across datasets measured by the PINC metric (Chen and Dolan, 2011), which is the opposite of BLEU (Papineni et al., 2002). MSRP consists of paraphrases with more ngram overlap (lower PINC), while PIT-2015 contains shorter and more lexically dissimilar sentences. Our new URL corpus is in between the two, and is more similar to PIT-2015. It includes user's intentional rephrasing of an original tweet from a news agency with some words untouched, as well as some dramatic paraphrases that are challenging for any automatic identification methods, such as *CO2 levels mark 'new era' in the world's changing climate* and *CO2 levels haven't been this high for 3 to 5 million years*.

MultiP exploits a restrictive constraint that the candidate sentence pairs share a same topical phrase. It achieves the best performance on PIT-2015, which naturally contains such phrases. For MSRP and URL datasets, we uses the named entity tagged with the longest span as an approximation of a shared topic phrase and thus suffered a performance drop.

Both Glove and WMT/OrMF utilize the underlying co-occurrence statistics of the text corpus. WMT/OrMF use global matrix factorization to project sentences into lower dimension and show great advantages on measuring sentence-level semantic similarities over Glove, which focuses on



(a) Twitter URL

(b) PIT-2015

(c) MSRP

Figure 2: Comparison of OrMF-based distributional semantic similarity across three paraphrase corpora.

word representations. Figure 2 shows that the fine-grained distribution of the OrMF-based cosine similarities and that the URL-linked Twitter data works well with OrMF to yield sentential paraphrases. Once combined with ngram overlap features, LEX-WMF and LEX-OrMF show consistently high performance across different datasets, close to the more complicated DeepPairwiseWord. The similarity focus mechanism on important pairwise word interactions in DeepPairwiseWord is more helpful for the two Twitter datasets, due to the fact that they contain lexically divergent paraphrases while MSRP has an artificial bias toward sentences with high n-gram overlap.

5 Extracting Phrasal Paraphrases

We can apply paraphrase identification models trained on our gold standard corpus to unlabeled Twitter data and continuously harvest sentential paraphrases in large quantities. We used the open-sourced LEX-OrMF model and obtained 114,025 sentential paraphrases (system predicted probability $\geq$ 0.5 and average precision = 69.08%) from raw 1% free Twitter data between 10/10/2016 and 01/10/2017. To demonstrate the utility, we show that we can extract up-to-date lexical and phrasal paraphrases from this data.

5.1 Phrase Extraction and Ranking

One of the most successful ideas to obtain lexical and phrasal paraphrases in large quantities is through word alignment, then ranking for better quality. This approach was proposed by Bannard (Bannard and Callison-Burch, 2005) and previously applied to bilingual parallel data to create PPDB (Ganitkevitch et al., 2013; Pavlick et al., 2015). There has been little previous work utilizing monolingual parallel data to learn paraphrases since it is not as naturally available as bitexts.

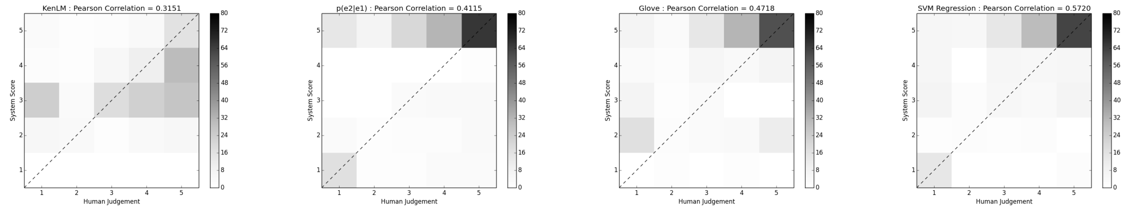
We used the GIZA++ word aligner in the Moses machine translation toolkit (Koehn et al., 2007) and extracted 245,686 phrasal paraphrases. Some examples are shown in Table 2. We additionally explored two supervised monolingual aligners: Jacana aligner (Yao et al., 2013) and Md Sultan's aligner (Sultan et al., 2014). We ranked the phrase pairs using four different scores:

- **Language Model Score** Let $w_{-2}w_{-1}pw_1w_2$ be the context of the phrase p. We considered a phrase p' to be a good substitute for p if $w_{-2}w_{-1}p'w_1w_2$ is a likely sequence according to a language model (Heafield, 2011) trained on Twitter data.
- **Translation Score** Moses provides translation probabilities $\varphi(p|p')$.
- **Glove Score** We used Glove (Pennington et al., 2014) pretrained 100-dimensional Twitter word vectors and cosine similarity.
- **Our Score** We trained a supervised SVM regression model using 500 phrase pairs with human ratings. We used the language model, translation, and glove scores as features, and additionally used the inverse phrase translation probability $\varphi(p'|p)$, lexical weighting $lex(p|p')$, and $lex(p'|p)$ from Moses.

Figure 3 compares the different ranking methods against the human judgments on 200 phrase pairs randomly sampled from GIZA++.

5.2 Paraphrase Quality Evaluation

We compared the quality of paraphrases extracted by our method with the closest previous work (BUCC-2013) (Xu et al., 2013), in which a similar phrase table was created using Moses from monolingual parallel tweets that contain the same named entity and calendar date. We randomly



(a) Language Model Score ($\rho = 0.3151$)

(b) Translation Score ($\rho = 0.4115$)

(c) Glove Score ($\rho = 0.4718$)

(d) Our Score ($\rho = 0.5720$)

Figure 3: Correlation between automatic scores (vertical axis) and 5-point human scores (horizontal axis) for ranking phrasal paraphrases. The darker squares along the diagonal line indicate a higher ranking.

sampled 500 phrase pairs from each phrase table and collected human judgements on a 5-point Likert scale, as described in Callison-Burch (Callison-Burch, 2008). Table 11 shows the evaluation results. We focused on the highest-quality paraphrases that rated as 5 (“all of the meaning of the original phrase is retained, and nothing is added”) and their presence among all extracted paraphrases sorted by ranking scores.

We were also interested in how these phrasal paraphrases compared with those in PPDB. We sampled an equal amount of 420 paraphrase pairs from our phrase tables and PPDB, and then checked what percentage out of the total 840 could be found in our phrase tables and PPDB, respectively. As shown in Table 12, there is little overlap between URL data and PPDB, only 1.3% (51.3-50%) plus 0.8% (50.8-50%). Our Twitter URL data complements well with the existing paraphrase resources, such as PPDB, which are primarily derived from well-edited texts.

Top Rankings	BUCC 2013	GIZA++	Jacana	Sultan
10%	76.0	85.5	90.0	90.0
20%	65.6	86.5	91.0	91.0
30%	62.7	79.2	86.0	88.0
40%	56.6	73.2	85.5	84.5
50%	52.1	68.1	83.4	84.8
100% (all)	36.3	49.8	75.8	77.2

Table 11: Percentage of high-quality phrasal paraphrases extracted from Twitter URL data (this work) by GIZA++, Jacana, Sultan aligners , comparing to the previous work (BUCC-2013).

6 Related Work

Sentential Paraphrase Data Researchers have found several data sources from which to collect sentential paraphrases: multiple news agencies reporting the same event (MSRP) (Dolan et al., 2004; Dolan and Brockett, 2005), multiple translated versions of a foreign novel (Barzilay and Elhadad, 2003; Barzilay and Lee, 2003) or other texts (Cohn et al., 2008), multiple definitions of the same concept (Hashimoto et al., 2011), descriptions of the same video clip from multiple workers (Chen and Dolan, 2011) or rephrased sentences (Burrows et al., 2013; Toutanova et al., 2016). However, all these data collection methods are incapable of obtaining sentential paraphrases on a large scale (i.e. limited number of news agencies or books with multiple translated versions), and/or lack meaningful negative examples. Both of these properties are crucial for developing machine learning models that identify paraphrases and measure semantic similarities.

	PPDB	URL	GIZA++	Jacana	Sultan
Sample Size	50%	50%	16.7%	16.7%	16.7%
Coverage	51.3%	50.8%	18.7%	32.1%	34.4%

Table 12: Coverage comparison of phrasal paraphrases extracted from Twitter URL data (sampled 1:1:1 from GIZA++, Jacana and Sultan’s aligner outputs) and the PPDB (Ganitkevitch et al., 2013).

Non-sentential Paraphrase Data There are other phrasal and syntactic paraphrase data, such as DIRT (Lin and Pantel, 2001), POLY (Grycner et al., 2016), PATTY (Nakashole et al., 2012), DEFIE (Bovi et al., 2015), and PPDB (Ganitkevitch et al., 2013; Pavlick et al., 2015). Most of these works focus on news or web data. Other earlier works on Twitter paraphrase extraction used unsupervised approaches (Xu et al., 2013; Wang et al., 2013) or small datasets (Zanzotto et al., 2011; Antoniak et al., 2015).

7 Conclusion and Future Work

In this paper, we show how a simple method can effectively and continuously collect large-scale sentential paraphrases from Twitter. We rigorously evaluated our data with automatic identification classification models and various measurements. We will share our new dataset with the research community; this dataset includes 51,524 sentence pairs manually labeled and a monthly growth of 30,000 sentential paraphrases automatically labeled. Future work could include expanding into many different languages present in social media and developing language-independent automatic paraphrase identification models.

Acknowledgments

We would like to thank Chris Callison-Burch, Weiwei Guo and Mike White for valuable discussions, as well as the anonymous reviewers for helpful feedback.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, et al. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval)*.

Maria Antoniak, Eric Bell, and Fei Xia. 2015. Leveraging paraphrase labels to extract synonyms from twitter. In *Proceedings of the 28th Florida Artificial Intelligence Research Society Conference (FLAIRS)*.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics* 34(4):555–596.

Eytan Bakshy, Jake M Hofman, Winter A Mason, and Duncan J Watts. 2011. Everyone's an influencer: quantifying influence on twitter. In *Proceedings of the fourth ACM international conference on Web search and data mining (WSDM)*.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL)*.

Regina Barzilay and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT)*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Rahul Bhagat and Eduard Hovy. 2013. What is a paraphrase? *Computational Linguistics* 39(3).

Johannes Bjerva, Johan Bos, Rob Van der Goot, and Malvina Nissim. 2014. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval)*.

Claudio Delli Bovi, Luca Telesca, and Roberto Navigli. 2015. Large-scale information extraction from textual definitions through deep syntactic and semantic analysis. *Transactions of the Association for Computational Linguistics (TACL)* pages 529–543.

Steven Burrows, Martin Potthast, and Benno Stein. 2013. Paraphrase acquisition via crowdsourcing and machine learning. *ACM Transactions on Intelligent Systems and Technology (TIST)* 4(3):43.

Chris Callison-Burch. 2008. Syntactic constraints on paraphrases extracted from parallel corpora. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

David L. Chen and William B. Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Trevor Cohn, Chris Callison-Burch, and Mirella Lapata. 2008. Constructing corpora for the development and evaluation of paraphrase systems. *Computational Linguistics* 34(4):597–614.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Proceedings of the First international conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment (MLCW)*.

Dipanjan Das and Noah A Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP)*.

William B. Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International*

Conference on Computational Linguistics (COLING).

Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL).*

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT).*

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-HLT).*

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT).*

Adam Grycner, Saarland Informatics Campus, and Gerhard Weikum. 2016. POLY: Mining relational paraphrases from multilingual sentences. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL).*

Weiwei Guo, Wei Liu, and Mona Diab. 2014. Fast tweet retrieval with compact binary codes. In *Proceedings of the 30th International Conference on Computational Linguistics (COLING).*

Chikara Hashimoto, Kentaro Torisawa, Stijn De Saeger, Jun'ichi Kazama, and Sadao Kurohashi. 2011. Extracting paraphrases from definition sentences on the web. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT).*

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT).*

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation (WMT).*

Hamid Izadinia, Fereshteh Sadeghi, Santosh K Divvala, Hannaneh Hajishirzi, Yejin Choi, and Ali Farhadi. 2015. Segment-phrase table for semantic segmentation, visual entailment and paraphrasing. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV).*

P. Jaccard. 1912. The distribution of the flora in the alpine zone. *New Phytologist* 11(2):37–50.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions.*

Tao Li and Vivek Srikumar. 2016. Exploiting sentence similarities for better alignments. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Dekang Lin and Patrick Pantel. 2001. DIRT – Discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD).*

Nitin Madnani and Bonnie J. Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics* 36(3).

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval).*

Ramtin Mehdizadeh Seraj, Maryam Siahbani, and Anoop Sarkar. 2015. Improving statistical machine translation with a multilingual paraphrase database. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Jasmina Milićević. 2006. A short guide to the meaning-text linguistic theory. *Journal of Koralex* 8:187–233.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: a taxonomy of relational patterns with semantic types. In *Proceedings of* *the 2012 Joint Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevich, and Chris Callison-Burch Ben Van Durme. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

Satoshi Sekine. 2006. On-demand information extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *Transactions of the Association for Computational Linguistics (TACL)* 2:219–230.

Kristina Toutanova, Chris Brockett, Ke M. Tran, and Saleema Amershi. 2016. A dataset and evaluation metrics for abstractive compression of sentences and short paragraphs. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Soroush Vosoughi and Deb Roy. 2016. A semi-automatic method for efficient detection of stories on social media. In *Tenth International AAAI Conference on Web and Social Media (ICWSM)*.

Ling Wang, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2013. Paraphrasing 4 microblog normalization. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

J. Wieting, M. Bansal, K. Gimpel, K. Livescu, and D. Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics (TACL)* 3:345–358.

Wei Xu, Chris Callison-Burch, and William B. Dolan. 2015. SemEval-2015 Task 1: Paraphrase and semantic similarity in Twitter (PIT). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval)*.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from Twitter. *Transactions of the Association for Computational Linguistics (TACL)* 2:435–448.

Wei Xu, Alan Ritter, and Ralph Grishman. 2013. Gathering and generating paraphrases from twitter with application to normalization. In *Proceedings of the Sixth Workshop on Building and Using Comparable Corpora (BUCC)*.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. A lightweight and high performance monolingual word aligner. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Kostas Tsioutsiouliklis. 2011. Linguistic redundancy in Twitter. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Congle Zhang, Stephen Soderland, and Daniel S Weld. 2015. Exploiting parallel news streams for unsupervised event extraction. *Transactions of the Association for Computational Linguistics (TACL)* 3:117–129.

Cross-domain Semantic Parsing via Paraphrasing

Yu Su
Department of Computer Science
University of California, Santa Barbara
ysu@cs.ucsb.edu

Xifeng Yan
Department of Computer Science
University of California, Santa Barbara
xyan@cs.ucsb.edu

Abstract

Existing studies on semantic parsing mainly focus on the in-domain setting. We formulate cross-domain semantic parsing as a domain adaptation problem: train a semantic parser on some source domains and then adapt it to the target domain. Due to the diversity of logical forms in different domains, this problem presents unique and intriguing challenges. By converting logical forms into canonical utterances in natural language, we reduce semantic parsing to paraphrasing, and develop an attentive sequence-to-sequence paraphrase model that is general and flexible to adapt to different domains. We discover two problems, *small micro variance* and *large macro variance*, of pretrained word embeddings that hinder their direct use in neural networks, and propose standardization techniques as a remedy. On the popular OVERNIGHT dataset, which contains eight domains, we show that both cross-domain training and standardized pre-trained word embedding can bring significant improvement.

1 Introduction

Semantic parsing, which maps natural language utterances into computer-understandable logical forms, has drawn substantial attention recently as a promising direction for developing natural language interfaces to computers. Semantic parsing has been applied in many domains, including querying data/knowledge bases (Woods, 1973; Zelle and Mooney, 1996; Berant et al., 2013), controlling IoT devices (Campagna et al., 2017), and communicating with robots (Chen and Mooney, 2011; Tellex et al., 2011; Artzi and Zettlemoyer, 2013; Bisk et al., 2016).

Despite the wide applications, studies on semantic parsing have mainly focused on the *in-domain* setting, where both training and testing data are drawn from the same domain. How to build semantic parsers that can learn across domains remains an under-addressed problem. In this work, we study *cross-domain semantic parsing*. We model it as a domain adaptation problem (Daumé III and Marcu, 2006), where we are given some *source* domains and a *target* domain, and the core task is to adapt a semantic parser trained on the source domains to the target domain (Figure 1). The benefits are two-fold: (1) by training on the source domains, the cost of collecting training data for the target domain can be reduced, and (2) the data of source domains may provide information complementary to the data collected for the target domain, leading to better performance on the target domain.

This is a very challenging task. Traditional domain adaptation (Daumé III and Marcu, 2006; Blitzer et al., 2006) only concerns natural languages, while semantic parsing concerns both natural and formal languages. Different domains often involve different predicates. In Figure 1, from the source BASKETBALL domain a semantic parser can learn the semantic mapping from natural language to predicates like `team` and `season`, but in the target SOCIAL domain it needs to handle predicates like `employer` instead. Worse still, even for the same predicate, it is legitimate to use arbitrarily different predicate symbols, e.g., other symbols like `hired_by` or even `predicate1` can also be used for the `employer` predicate, reminiscent of the symbol grounding problem (Harnad, 1990). Therefore, directly transferring the mapping from natural language to predicate symbols learned from source domains to the target domain may not be much beneficial.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1235–1246
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

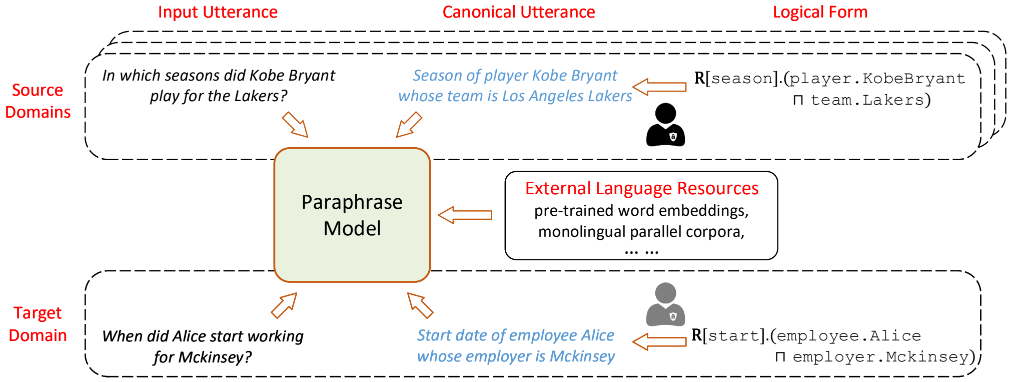

Input Utterance
Canonical Utterance
Logical Form
Source Domains
In which seasons did Kobe Bryant play for the Lakers?
Season of player Kobe Bryant whose team is Los Angeles Lakers
R[season].(player.KobeBryant ⊓ team.Lakers)
Paraphrase Model
External Language Resources
pre-trained word embeddings,
monolingual parallel corpora,
… …
Target Domain
When did Alice start working for Mckinsey?
Start date of employee Alice whose employer is Mckinsey
R[start].(employee.Alice ⊓ employer.Mckinsey)

Figure 1: Cross-domain semantic parsing via paraphrasing framework. In a deterministic way, logical forms are first converted into canonical utterances in natural language. A paraphrase model then learns from the source domains and adapts to the target domain. External language resources can be incorporated in a consistent way across domains.

Inspired by the recent success of paraphrasing based semantic parsing (Berant and Liang, 2014; Wang et al., 2015), we propose to use natural language as an intermediate representation for cross-domain semantic parsing. As shown in Figure 1, logical forms are converted into canonical utterances in natural language, and semantic parsing is reduced to paraphrasing. It is the knowledge of paraphrasing, at lexical, syntactic, and semantic levels, that will be transferred across domains.

Still, adapting a paraphrase model to a new domain is a challenging and under-addressed problem. To give some idea of the difficulty, for each of the eight domains in the popular OVERNIGHT (Wang et al., 2015) dataset, 30% to 55% of the words never occur in any of the other domains, a similar problem observed in domain adaptation for machine translation (Daumé III and Jagarlamudi, 2011). The paraphrase model therefore can get little knowledge for a substantial portion of the target domain from the source domains. We introduce pre-trained word embeddings such as WORD2VEC (Mikolov et al., 2013) to combat the vocabulary variety across domains. Based on recent studies on neural network initialization, we conduct a statistical analysis of pre-trained word embeddings and discover two problems that may hinder their direct use in neural networks: *small micro variance*, which hurts optimization, and *large macro variance*, which hurts generalization. We propose to *standardize* pre-trained word embeddings, and show its advantages both analytically and experimentally.

On the OVERNIGHT dataset, we show that cross-domain training under the proposed framework can significantly improve model performance. We also show that, compared with directly using pre-trained word embeddings or normalization as in previous work, the proposed standardization technique can lead to about 10% absolute improvement in accuracy.

2 Cross-domain Semantic Parsing

2.1 Problem Definition

Unless otherwise stated, we will use u to denote input utterance, c for canonical utterance, and z for logical form. We denote $\mathcal{U}$ as the set of all possible utterances. For a domain, suppose $\mathcal{Z}$ is the set of logical forms, a semantic parser is a mapping $f: \mathcal{U} \rightarrow \mathcal{Z}$ that maps every input utterance to a logical form (a `null` logical form can be included in $\mathcal{Z}$ to reject out-of-domain utterances).

In cross-domain semantic parsing, we assume there are a set of K source domains $\{\mathcal{Z}_i\}_{i=1}^{K}$, each with a set of training examples $\{(u_j^i, z_j^i)\}_{j=1}^{N_i}$. It is in principle advantageous to model the source domains separately (Daumé III and Marcu, 2006), which retains the possibility of separating domain-general information from domain-specific information, and only transferring the former to the target domain. For simplicity, here we merge the source domains into a single domain $\mathcal{Z}_s$ with training data $\{(u_i, z_i)\}_{i=1}^{N_s}$. The task is to learn a semantic parser $f: \mathcal{U} \rightarrow \mathcal{Z}_t$ for a target domain $\mathcal{Z}_t$, for which we have a set of training examples $\{(u_i, z_i)\}_{i=1}^{N_t}$. Some characteristics can be summarized as follows:

- $\mathcal{Z}_t$ and $\mathcal{Z}_s$ can be totally disjoint.
- The input utterance distribution of the source and the target domains can be independent and differ remarkably.

- Typically $N_t \ll N_s$.

In the most general and challenging case, $\mathcal{Z}_t$ and $\mathcal{Z}_s$ can be defined using different formal languages. Because of the lack of relevant datasets, here we restrain ourselves to the case where $\mathcal{Z}_t$ and $\mathcal{Z}_s$ are defined using the same formal language, e.g., λ-DCS (Liang, 2013) as in the OVERNIGHT dataset.

2.2 Framework

Our framework follows the research line of semantic parsing via paraphrasing (Berant and Liang, 2014; Wang et al., 2015). While previous work focuses on the in-domain setting, we discuss its applicability and advantages in the cross-domain setting, and develop techniques to address the emerging challenges in the new setting.

Canonical utterance. We assume a *one-to-one* mapping $g\colon \mathcal{Z} \rightarrow \mathcal{C}$, where $\mathcal{C} \subset \mathcal{U}$ is the set of canonical utterances. In other words, every logical form will be converted into a unique canonical utterance deterministically (Figure 1). Previous work (Wang et al., 2015) has demonstrated how to design such a mapping, where a domain-general grammar and a domain-specific lexicon are constructed to automatically convert every logical form to a canonical utterance. In this work, we assume the mapping is given[1], and focus on the subsequent paraphrasing and domain adaptation problems.

This design choice is worth some discussion. The grammar, or at least the lexicon for mapping predicates to natural language, needs to be provided by domain administrators. This indeed brings an additional cost, but we believe it is reasonable and even necessary for three reasons: (1) Only domain administrators know the predicate semantics the best, so it has to be them to reveal that by grounding the predicates to natural language (the symbol grounding problem (Harnad, 1990)). (2) Otherwise, predicate semantics can only be learned from supervised training data of each domain, bringing a significant cost on data collection. (3) Canonical utterances are understandable by average users, and thus can also be used for training data collection via crowdsourcing (Wang et al., 2015; Su et al., 2016), which can amortize the cost.

Take comparatives as an example. In logical forms, comparatives can be legitimately defined using arbitrarily different predicates in different domains, e.g., `<`, `smallerInSize`, or even predicates with an ambiguous surface form, like `lt`. When converting logical form to canonical utterance, however, domain administrators have to choose common natural language expressions like "*less than*" and "*smaller*", providing a shared ground for cross-domain semantic parsing.

Paraphrase model. In the previous work based on paraphrasing (Berant and Liang, 2014; Wang et al., 2015), semantic parsers are implemented as log-linear models with hand-engineered domain-specific features (including paraphrase features). Considering the recent success of representation learning for domain adaptation (Glorot et al., 2011; Chen et al., 2012), we propose a paraphrase model based on the sequence-to-sequence (Seq2Seq) model (Sutskever et al., 2014), which can be trained end to end without feature engineering. We show that it outperforms the previous log-linear models by a large margin in the in-domain setting, and can easily adapt to new domains.

Pre-trained word embeddings. An advantage of reducing semantic parsing to paraphrasing is that external language resources become easier to incorporate. Observing the vocabulary variety across domains, we introduce pre-trained word embeddings to facilitate domain adaptation. For the example in Figure 1, the paraphrase model may have learned the mapping from "play for" to "whose team is" in a source domain. By acquiring word similarities ("play"-"work" and "team"-"employer") from pre-trained word embeddings, it can establish the mapping from "work for" to "whose employer is" in the target domain, even without in-domain training data. We analyze statistical characteristics of the pre-trained word embeddings, and propose standardization techniques to remedy some undesired characteristics that may bring a negative effect to neural models.

Domain adaptation protocol. We will use the following protocol: (1) train a paraphrase model using the data of the source domain, (2) use the learned parameters to initialize a model in the target domain, and (3) fine-tune the model using the training data of the target domain.

[1] In the experiments we use the provided canonical utterances of the OVERNIGHT dataset.

2.3 Prior Work

While most studies on semantic parsing so far have focused on the in-domain setting, there are a number of studies of particular relevance to this work. In the recent efforts of scaling semantic parsing to large knowledge bases like Freebase (Bollacker et al., 2008), researchers have explored several ways to infer the semantics of knowledge base relations unseen in training, which are often based on at least one (often both) of the following assumptions: (1) *Distant supervision*. Freebase entities can be linked to external text corpora, and serve as anchors for seeking semantics of Freebase relations from text. For example, Cai and Alexander (2013), among others (Berant et al., 2013; Xu et al., 2016), use sentences from Wikipedia that contain any entity pair of a Freebase relation as the support set of the relation. (2) *Self-explaining predicate symbols*. Most Freebase relations are described using a carefully chosen symbol (surface form), e.g., `place_of_birth`, which provides strong cues for their semantics. For example, Yih et al. (2015) directly compute the similarity of input utterance and the surface form of Freebase relations via a convolutional neural network. Kwiatkowski et al. (2013) also extract lexical features from input utterance and the surface form of entities and relations. They have actually evaluated their model on Freebase subdomains not covered in training, and have shown impressive results. However, in the more general setting of cross-domain semantic parsing, we may have neither of these luxuries. Distant supervision may not be available (e.g., IoT devices involving no entities but actions), and predicate symbols may not provide enough cues (e.g., `predicate1`). In this case, seeking additional inputs from domain administrators is probably necessary.

In parallel of this work, Herzig and Berant (2017) have explored another direction of semantic parsing with multiple domains, where they use all the domains to train a single semantic parser, and attach a domain-specific encoding to the training data of each domain to help the semantic parser differentiate between domains. We pursue a different direction: we train a semantic parser on some source domains and adapt it to the target domain. Another difference is that their work directly maps utterances to logical forms, while ours is based on paraphrasing.

Cross-domain semantic parsing can be seen as a way to reduce the cost of training data collection, which resonates with the recent trend in semantic parsing. Berant et al. (2013) propose to learn from utterance-denotation pairs instead of utterance-logical form pairs, while Wang et al. (2015) and Su et al. (2016) manage to employ crowd workers with no linguistic expertise for data collection. Jia and Liang (2016) propose an interesting form of data augmentation. They learn a grammar from existing training data, and generate new examples from the grammar by recombining segments from different examples.

We use natural language as an intermediate representation to transfer knowledge across domains, and assume the mapping from the intermediate representation (canonical utterance) to logical form can be done deterministically. Several other intermediate representations have also been used, such as combinatory categorial grammar (Kwiatkowski et al., 2013; Reddy et al., 2014), dependency tree (Reddy et al., 2016, 2017), and semantic role structure (Goldwasser and Roth, 2013). But their main aim is to better represent input utterances with a richer structure. A separate ontology matching step is needed to map the intermediate representation to logical form, which requires domain-dependent training.

A number of other related studies have also used paraphrasing. For example, Fader et al. (2013) leverage question paraphrases to for question answering, while Narayan et al. (2016) generate paraphrases as a way of data augmentation.

Cross-domain semantic parsing can greatly benefit from the rich literature of domain adaptation and transfer learning (Daumé III and Marcu, 2006; Blitzer et al., 2006; Pan and Yang, 2010; Glorot et al., 2011). For example, Chelba and Acero (2004) use parameters trained in the source domain as prior to regularize parameters in the target domain. The feature augmentation technique from Daumé III (2009) can be very helpful when there are multiple source domains. We expect to see many of these ideas to be applied in the future.

3 Paraphrase Model

In this section we propose a paraphrase model based on the Seq2Seq model (Sutskever et al., 2014) with soft attention. Similar models have been used in semantic parsing (Jia and Liang, 2016; Dong and Lapata, 2016) but for directly mapping utterances to logical forms. We demonstrate that it can also be used as a paraphrase model for semantic parsing. Several other neural models have been proposed for paraphrasing (Socher et al., 2011; Hu et al., 2014; Yin and Schütze, 2015), but it is not the focus of this work to compare all the alternatives.

For an input utterance $u = (u_1, u_2, \ldots, u_m)$ and an output canonical utterance $c = (c_1, c_2, \ldots, c_n)$, the model estimates the conditional probability $p(c|u) = \prod_{j=1}^{n} p(c_j|u, c_{1:j-1})$. The tokens are first converted into vectors via a word embedding layer ϕ. The initialization of the word embedding layer is critical for domain adaptation, which we will further discuss in Section 4.

The *encoder*, which is implemented as a bi-directional recurrent neural network (RNN), first encodes u into a sequence of state vectors $(h_1, h_2, \ldots, h_m)$. The state vectors of the forward RNN and the backward RNN are respectively computed as:

$$\overrightarrow{h}_i = GRU_{fw}(\phi(u_i), \overrightarrow{h}_{i-1})$$
$$\overleftarrow{h}_i = GRU_{bw}(\phi(u_i), \overleftarrow{h}_{i+1})$$

where gated recurrent unit (GRU) as defined in (Cho et al., 2014) is used as the recurrence. We then concatenate the forward and backward state vectors, $h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i], i = 1, \ldots, m$.

We use an attentive RNN as the *decoder*, which will generate the output tokens one at a time. We denote the state vectors of the decoder RNN as $(d_1, d_2, \ldots, d_n)$. The attention takes a form similar to (Vinyals et al., 2015). For the decoding step j, the decoder is defined as follows:

$$\begin{aligned} d_0 &= \tanh(W_0[\overrightarrow{h}_m, \overleftarrow{h}_1]) \\ u_{ji} &= v^T \tanh(W_1 h_i + W_2 d_j) \\ \alpha_{ji} &= \frac{u_{ji}}{\sum_{i'=1}^{m} u_{ji'}} \\ h'_j &= \sum_{i=1}^{m} \alpha_{ji} h_i \\ d_{j+1} &= GRU([\phi(c_j), h'_j], d_j) \\ p(c_j|u, c_{1:j-1}) &\propto \exp(U[d_j, h'_j]) \end{aligned}$$

where W_0, W_1, W_2, v and U are model parameters. The decoder first calculates normalized attention weights α_{ji} over encoder states, and get a summary state h'_j. The summary state is then used to calculate the next decoder state d_{j+1} and the output probability distribution $p(c_j|u, c_{1:j-1})$.

Training. Given a set of training examples $\{(u_i, c_i)\}_{i=1}^{N}$, we minimize the cross-entropy loss $-\frac{1}{N}\sum_{i=1}^{N} \log p(c_i|u_i)$, which maximizes the log probability of the correct canonical utterances. We apply dropout (Hinton et al., 2012) on both input and output of the GRU cells to prevent overfitting.

Testing. Given a domain $\{\mathcal{Z}, \mathcal{C}\}$, there are two ways to use a trained model. One is to use it to *generate* the most likely output utterance u' given an input utterance u (Sutskever et al., 2014),

$$u' = \arg\max_{u' \in \mathcal{U}} p(u'|u).$$

In this case u' can be any utterance permissable by the output vocabulary, and may not necessarily be a legitimate canonical utterance in $\mathcal{C}$. This is more suitable for large domains with a lot of logical forms, like Freebase. An alternative way is to use the model to *rank* the legitimate canonical utterances (Kannan et al., 2016):

$$c = \arg\max_{c \in \mathcal{C}} p(c|u),$$

which is more suitable for small domains having a limited number of logical forms, like the ones in the OVERNIGHT dataset. We will adopt the second strategy. It is also very challenging; random guessing leads to almost no success. It is also possible to first find a smaller set of candidates to rank via beam search (Berant et al., 2013; Wang et al., 2015).

4 Pre-trained Word Embedding for Domain Adaptation

Pre-trained word embeddings like WORD2VEC have a great potential to combat the vocabulary variety across domains. For example, we can use pre-trained WORD2VEC vectors to initialize the word embedding layer of the source domain, with the hope that the other parameters in the model will co-adapt with the word vectors during training in the source domain, and generalize better to the out-of-vocabulary words (but covered by WORD2VEC) in the target domain. However, deep neural networks are very sensitive to initialization (Erhan et al., 2010), and a statistical analysis of the pre-trained WORD2VEC vectors reveals some characteristics that may not be desired for initializing deep neural networks. In this section we present the analysis and propose a standardization technique to remedy the undesired characteristics.

Initialization	L2 norm	Micro Variance	Cosine Sim.
Random	17.3 ± 0.45	1.00 ± 0.05	0.00 ± 0.06
word2vec	2.04 ± 1.08	0.02 ± 0.02	0.13 ± 0.11
word2vec + ES	17.3 ± 0.05	1.00 ± 0.00	0.13 ± 0.11
word2vec + FS	16.0 ± 8.47	1.09 ± 1.31	0.12 ± 0.10
word2vec + EN	1.00 ± 0.00	0.01 ± 0.00	0.13 ± 0.11

Table 1: Statistics of the word embedding matrix with different initialization strategies. Random: random sampling from $U(-\sqrt{3}, \sqrt{3})$, thus unit variance. word2vec: raw word2vec vectors. ES: per-example standardization. FS: per-feature standardization. EN: per-example normalization. Cosine similarity is computed on a randomly selected (but fixed) set of 1M word pairs.

Analysis. Our analysis will be based on the 300-dimensional word2vec vectors trained on the 100B-word Google News corpus[2]. It contains 3 million words, leading to a 3M-by-300 word embedding matrix. The "rule of thumb" to randomly initialize word embedding in neural networks is to sample from a uniform or Gaussian distribution with *unit variance*, which works well for a wide range of neural network models in general. We therefore use it as a reference to compare different word embedding initialization strategies. Given a word embedding matrix, we compute the L2 norm of each row and report the mean and the standard deviation. Similarly, we also report the variance of each row (denoted as *micro variance*), which indicates how far the numbers in the row spread out, and pair-wise cosine similarity, which indicates the word similarity captured by word2vec.

The statistics of the word embedding matrix with different initialization strategies are shown in Table 1. Compared with random initialization, two characteristics of the word2vec vectors stand out: (1) *Small micro variance*. Both the L2 norm and the micro variance of the word2vec vectors are much smaller. (2) *Large macro variance*. The variance of different word2vec vectors, reflected by the standard deviation of L2 norm, is much larger (e.g., the maximum and the minimum L2 norm are 21.1 and 0.015, respectively). Small micro variance can make the variance of neuron activations starts off too small[3], implying a poor starting point in the parameter space. On the other hand, because of the magnitude difference, large macro variance may make a model hard to generalize to words unseen in training.

Standardization. Based on the above analysis, we propose to do *unit variance standardization* (standardization for short) on pre-trained word embeddings. There are two possible ways, *per-example standardization*, which standardizes each row of the embedding matrix to unit variance by simply dividing by the standard deviation of the row, and *per-feature standardization*, which standardizes each column instead. We do not make the rows or columns zero mean. Per-example standardization enjoys the goodness of both random initialization and pre-trained word embeddings: it fixes the small micro variance problem as well as the large macro variance problem of pre-trained word embeddings, while still preserving cosine similarity, i.e., word similarity. Per-feature standardization does not preserve cosine similarity, nor does it fix the large macro variance problem. However, it enjoys the benefit of *global* statistics, in contrast to the *local* statistics of individual word vectors used in per-example standardization. Therefore, in problems where the testing and training vocabularies are similar, per-feature standardization may be more advantageous. Both standardizations lose vector magnitude information. Levy et al. (2015) have suggested *per-example normalization*[4] of pre-trained word embeddings for lexical tasks like word similarity and analogy, which do no involve deep neural networks. Making the word vectors unit length alleviates the large macro variance problem, but the small micro variance problem remains (Table 1).

Discussion. This is indeed a pretty simple trick, and per-feature standardization (with zero mean) is also a standard data preprocessing method. However, it is not self-evident that this kind of standardization shall be applied on pre-trained word embeddings before using them in deep neural networks, especially with the obvious downside of rendering the word embedding algorithm's loss function sub-optimal.

We expect this to be less of a issue for large-scale problems with a large vocabulary and abundant training examples. For example, Vinyals et al. (2015) have found that directly using the word2vec vectors for initialization can bring a

[2] https://code.google.com/archive/p/word2vec/

[3] Under some conditions, including using Xavier initialization (also introduced in that paper and now widely used) for weights, Glorot and Bengio (2010) have shown that the activation variances in a feedforward neural network will be roughly the same as the input variances (word embedding here) at the beginning of training.

[4] It can also be found in the implementation of Glove (Pennington et al., 2014): https://github.com/stanfordnlp/GloVe

consistent, though small, improvement in neural constituency parsing. However, for smaller-scale problems (e.g., an application domain of semantic parsing can have a vocabulary size of only a few hundreds), this issue becomes more critical. Initialized with the raw pre-trained vectors, a model may quickly fall into a poor local optimum and may not have enough signal to escape. Because of the large macro variance problem, standardization can be critical for domain adaptation, which needs to generalize to many words unseen in training.

The proposed standardization technique appears in a similar spirit to batch normalization (Ioffe and Szegedy, 2015). We notice two computational differences, that ours is applied on the inputs while batch normalization is applied on internal neuron activations, and that ours standardizes the whole word embedding matrix beforehand while batch normalization standardizes each mini-batch on the fly. In terms of motivation, the proposed technique aims to remedy some undesired characteristics of pre-trained word embeddings, and batch normalization aims to reduce the internal covariate shift. It is of interest to study the combination of the two in future work.

5 Evaluation

5.1 Data Analysis

The OVERNIGHT dataset (Wang et al., 2015) contains 8 different domains. Each domain is based on a separate knowledge base, with logical forms written in λ-DCS (Liang, 2013). Logical forms are converted into canonical utterances via a simple grammar, and the input utterances are collected by asking crowd workers to paraphrase the canonical utterances. Different domains are designed to stress different types of linguistic phenomena. For example, the CALENDAR domain requires a semantic parser to handle temporal language like "*meetings that start after 10 am*", while the BLOCKS domain features spatial language like "*which block is above block 1*".

Vocabularies vary remarkably across domains (Table 2). For each domain, only 45% to 70% of the words are covered by any of the other 7 domains. A model has to learn the out-of-vocabulary words from scratch using in-domain training data. The pre-trained WORD2VEC embedding covers most of the words of each domain, and thus can connect the domains to facilitate domain adaptation. Words that are still missing are mainly stop words and typos, e.g., "*ealiest*".

5.2 Experiment Setup

We compare our model with all the previous methods evaluated on the OVERNIGHT dataset. Wang et al. (2015) use a log-linear model with a rich set of features, including paraphrase features derived from PPDB (Ganitkevitch et al., 2013), to rank logical forms. Xiao et al. (2016) use a multi-layer perceptron to encode the unigrams and bigrams of the input utterance, and then use a RNN to predict the derivation sequence of a logical form under a grammar. Similar to ours, Jia and Liang (2016) also use a Seq2Seq model with bi-directional RNN encoder and attentive decoder, but it is used to predict linearized logical forms. They also propose a data augmentation technique, which further improves the average accuracy to 77.5%. But it is orthogonal to this work and can be incorporated in any model including ours, therefore not included.

The above methods are all based on the in-domain setting, where a separate parser is trained for each domain. In parallel of this work, Herzig and Berant (2017) have explored another direction of cross-domain training: they use all of the domains to train a single parser, with a special domain encoding to help differentiate between domains. We instead model it as a domain adaptation problem, where training on the source and the target domains are separate. Their model is the same as Jia and Liang (2016). It is the current best-performing method on the OVERNIGHT dataset.

We use the standard 80%/20% split of training and testing, and randomly hold out 20% of training for validation. In cross-domain experiments, for each target domain, all the other domains are combined as the source domain. Hyper-parameters are selected based on the validation set. State size of both the encoder and the decoder are set to 100, and word embedding size is set to 300. Input and output dropout rate of the GRU cells are 0.7 and 0.5, respectively, and mini-batch size is 512. We use Adam with the default parameters suggested in the paper for optimization. We use gradient clipping with a cap for global norm at 5.0 to alleviate the exploding gradients problem of recurrent neural networks. Early stopping based on the validation set is used to decide when to stop training. The selected model is retrained using the whole training set (training + validation). The

Metric	CALENDAR	BLOCKS	HOUSING	RESTAURANTS	PUBLICATIONS	RECIPES	SOCIAL	BASKETBALL
# of example (N)	837	1995	941	1657	801	1080	4419	1952
# of logical form ($\lvert\mathcal{Z}\rvert$, $\lvert\mathcal{C}\rvert$)	196	469	231	339	149	124	624	252
vocab. size ($\lvert\mathcal{V}\rvert$)	228	227	318	342	203	256	533	360
% $\in$ other domains	71.1	61.7	60.7	55.8	65.6	71.9	46.0	45.6
% $\in$ WORD2VEC	91.2	91.6	88.4	88.6	91.1	93.8	86.9	86.9
% $\in$ other domains + WORD2VEC	**93.9**	**93.8**	**90.9**	**90.4**	**95.6**	**97.3**	**89.3**	**89.4**

Table 2: Statistics of the domains in the OVERNIGHT dataset. Pre-trained WORD2VEC embedding covers most of the words in each domain, paving a way for domain adaptation.

Method	CALENDAR	BLOCKS	HOUSING	RESTAURANTS	PUBLICATIONS	RECIPES	SOCIAL	BASKETBALL	Avg.
Previous Methods									
Wang et al. (2015)	74.4	41.9	54.0	75.9	59.0	70.8	48.2	46.3	58.8
Xiao et al. (2016)	75.0	55.6	61.9	80.1	75.8	–	80.0	80.5	72.7
Jia and Liang (2016)	78.0	58.1	71.4	76.2	76.4	79.6	81.4	85.2	75.8
Herzig and Berant (2017)	82.1	**62.7**	78.3	82.2	**80.7**	82.9	81.7	86.2	79.6
Our Methods									
Random + I	75.6	60.2	67.2	77.7	77.6	80.1	80.7	86.5	75.7
Random + X	79.2	54.9	74.1	76.2	78.5	82.4	82.5	86.7	76.9
WORD2VEC + I	67.9	59.4	52.4	75.0	64.0	73.2	77.0	87.5	69.5
WORD2VEC + X	78.0	54.4	63.0	81.3	74.5	83.3	81.5	83.1	74.9
WORD2VEC + EN + I	63.1	56.1	60.3	75.3	65.2	69.0	76.4	81.8	68.4
WORD2VEC + EN + X	78.0	52.6	63.5	74.7	65.2	80.6	79.9	80.8	71.2
WORD2VEC + FS + I	78.6	62.2	67.7	78.6	75.8	85.7	81.3	86.7	77.1
WORD2VEC + FS + X	**82.7**	59.4	75.1	80.4	78.9	85.2	81.8	87.2	78.9
WORD2VEC + ES + I	79.8	60.2	71.4	81.6	78.9	84.7	82.9	86.2	78.2
WORD2VEC + ES + X	82.1	62.2	**78.8**	**83.7**	80.1	**86.1**	**83.1**	**88.2**	**80.6**

Table 3: Main experiment results. We combine the proposed paraphrase model with different word embedding initializations. I: in-domain, X: cross-domain, EN: per-example normalization, FS: per-feature standardization, ES: per-example standardization.

evaluation metric is accuracy, i.e., the proportion of testing examples for which the top prediction yields the correct denotation. Our model is implemented in Tensorflow (Abadi et al., 2016), and the code can be found at `https://github.com/ysu1989/CrossSemparse`.

5.3 Experiment Results

5.3.1 Comparison with Previous Methods

The main experiment results are shown in Table 3. Our base model (Random + I) achieves an accuracy comparable to the previous best in-domain model (Jia and Liang, 2016). With our main novelties, cross-domain training and word embedding standardization, our full model is able to outperform the previous best model, and achieve the best accuracy on 6 out of the 8 domains. Next we examine the novelties separately.

5.3.2 Word Embedding Initialization

The in-domain results clearly show the sensitivity of model performance to word embedding initialization. Directly using the raw WORD2VEC vectors or with per-example normalization, the performance is significantly worse than random initialization (6.2% and 7.3%, respectively). Based on the previous analysis, however, one should not be too surprised. The small micro variance problem hurts optimization. In sharp contrast, both of the proposed standardization techniques lead to better in-domain performance than random initialization (1.4% and 2.5%, respectively), setting a new best in-domain accuracy (78.2%) on OVERNIGHT. The results show that the pre-trained WORD2VEC vectors can indeed provide useful information, but only when they are properly standardized.

5.3.3 Cross-domain Training

A consistent improvement from cross-domain training is observed across all word embedding initialization strategies. Even for raw WORD2VEC embedding or per-example normalization, cross-domain training helps the model escape the poor initialization, though still inferior to the alternative initializations. The best results are again obtained with standardization, with per-example standardization bringing a slightly larger improvement than per-feature standardization. We observe that the improvement from cross-domain training is correlated with the abundance of the in-domain training data of the target domain. To further examine this observation, we use the ratio between the number of examples (N) and the vocabulary size ($\lvert\mathcal{V}\rvert$) to indicate the data abundance of a domain (the higher, the more abundant), and compute the Pearson correlation coefficient between data abundance and accuracy improvement from cross-domain training (X$-$I). The results in Ta-

Word Embedding Initialization	Correlation
Random	−0.698
WORD2VEC	−0.730
WORD2VEC + EN	−0.461
WORD2VEC + FS	−0.770
WORD2VEC + ES	−0.514

Table 4: Correlation between in-domain data abundance and improvement from cross-domain training. The gain of cross-domain training is more significant when in-domain training data is less abundant.

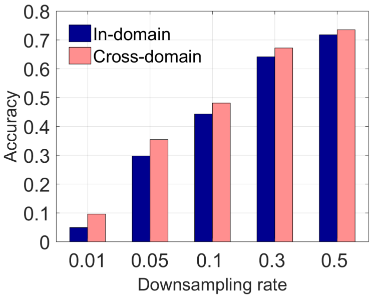


Figure 2: Results with downsampled in-domain training data. The experiment with each downsampling rate is repeated for 3 times and average results are reported. For simplicity, we only report the average accuracy over all domains. Pre-trained word embedding with per-example standardization is used in both settings.

ble 4 show a consistent, moderate to strong negative correlation between the two variables. In other words, cross-domain training is more beneficial when in-domain training data is less abundant, which is reasonable because in that case the model can learn more from the source domain data that is missing in the training data of the target domain.

5.3.4 Using Downsampled Training Data

Compared with the vocabulary size and the number of logical forms, the in-domain training data in the OVERNIGHT dataset is indeed abundant. In cross-domain semantic parsing, we are more interested in the scenario where there is insufficient training data for the target domain. To emulate this scenario, we downsample the in-domain training data of each target domain, but still use all training data from the source domain (thus $N_t \ll N_s$). The results are shown in Figure 2. The gain of cross-domain training is most significant when in-domain training data is scarce. As we collect more in-domain training data, the gain becomes smaller, which is expected. These results reinforce those from Table 4. It is worth noting that the effect of downsampling varies across domains. For domains with quite abundant training data like SOCIAL, using only 30% of the in-domain training data, the model can achieve an accuracy almost as good as when using all the data.

6 Discussion

Scalability, including *vertical scalability*, i.e., how to scale up to handle more complex inputs and logical constructs, and *horizontal scalability*, i.e., how to scale out to handle more domains, is one of the most critical challenges semantic parsing is facing today. In this work, we took an early step towards horizontal scalability, and proposed a paraphrasing based framework for cross-domain semantic parsing. With a sequence-to-sequence paraphrase model, we showed that cross-domain training of semantic parsing can be quite effective under a domain adaptation setting. We also studied how to properly standardize pre-trained word embeddings in neural networks, especially for domain adaptation.

This work opens up a number of future directions. As discussed in Section 2.3, many conventional domain adaptation and representation learning ideas can find application in cross-domain semantic parsing. In addition to pre-trained word embeddings, other language resources like paraphrase corpora (Ganitkevitch et al., 2013) can be incorporated into the paraphrase model to further facilitate domain adaptation. In this work we require a full mapping from logical form to canonical utterance, which could be costly for large domains. It is of practical interest to study the case where only a lexicon for mapping schema items to natural language is available. We have restrained ourselves to the case where domains are defined using the same formal language, and we look forward to evaluating the framework on domains of different formal languages when such datasets with canonical utterances become available.

Acknowledgments

The authors would like to thank the anonymous reviewers for their thoughtful comments. This research was sponsored in part by the Army Research Laboratory under cooperative agreements W911NF09-2-0053 and NSF IIS 1528175. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467 [cs.DC]*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Yonatan Bisk, Deniz Yuret, and Daniel Marcu. 2016. Natural language communication with robots. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International conference on Management of data*.

Qingqing Cai and Alexander Yates. 2013. Semantic parsing freebase: Towards open-domain semantic parsing. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*.

Giovanni Campagna, Rakesh Ramesh, Silei Xu, Michael Fischer, and Monica S Lam. 2017. Almond: The architecture of an open, crowdsourced, privacy-preserving, programmable virtual assistant. In *Proceedings of the International Conference on World Wide Web*, pages 341–350. International World Wide Web Conferences Steering Committee.

Ciprian Chelba and Alex Acero. 2004. Adaptation of maximum entropy capitalizer: Little data can help a lot. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

David L Chen and Raymond J Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the International Conference on Machine Learning*.

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

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv:0907.1815 [cs.LG]*.

Hal Daumé III and Jagadeesh Jagarlamudi. 2011. Domain adaptation for machine translation by mining unseen words. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the International Conference on Machine Learning*.

Dan Goldwasser and Dan Roth. 2013. Leveraging domain-independent information in semantic parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346.

Jonathan Herzig and Jonathan Berant. 2017. Neural semantic parsing over multiple knowledge-bases. *arXiv:1702.01569 [cs.CL]*.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580 [cs.NE]*.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning*, pages 448–456.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, László Lukács, Marina Ganea, Peter Young, et al. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

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

Percy Liang. 2013. Lambda dependency-based compositional semantics. *arXiv:1309.4408 [cs.AI]*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

Shashi Narayan, Siva Reddy, and Shay B Cohen. 2016. Paraphrase generation from latent-variable PCFGs for semantic parsing. *arXiv:1601.06068 [cs.CL]*.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

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

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Siva Reddy, Oscar Täckström, Slav Petrov, Mark Steedman, and Mirella Lapata. 2017. Universal semantic parsing. *arXiv:1702.03196 [cs.CL]*.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gür, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for QA evaluation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

Stefanie A Tellex, Thomas Fleming Kollar, Steven R Dickerson, Matthew R Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

William A Woods. 1973. Progress in natural language understanding: an application to lunar geology. In *Proceedings of the American Federation of Information Processing Societies Conference*.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Wenpeng Yin and Hinrich Schütze. 2015. Multi-GranCNN: An architecture for general matching of text chunks on multiple levels of granularity. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

John M Zelle and Raymon J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

A Joint Sequential and Relational Model for Frame-Semantic Parsing

Bishan Yang
Machine Learning Department
Carnegie Mellon University
bishan@cs.cmu.edu

Tom Mitchell
Machine Learning Department
Carnegie Mellon University
tom.mitchell@cs.cmu.edu

Abstract

We introduce a new method for frame-semantic parsing that significantly improves the prior state of the art. Our model leverages the advantages of a deep bidirectional LSTM network which predicts semantic role labels word by word and a relational network which predicts semantic roles for individual text expressions in relation to a predicate. The two networks are integrated into a single model via knowledge distillation, and a unified graphical model is employed to jointly decode frames and semantic roles during inference. Experiments on the standard FrameNet data show that our model significantly outperforms existing neural and non-neural approaches, achieving a 5.7 F1 gain over the current state of the art, for full frame structure extraction.

1 Introduction

One way to represent meaning is through organization of semantic structures. Consider the following sentences "John sells Marry a car." and "Mary buys a car from John.". While having different syntactic structures, they express the same type of event that involves a buyer, a seller, and goods. Such meaning can be represented using semantic frames – structured representations that characterize events, scenarios, and the participants. Researchers have developed FrameNet (Baker et al., 1998; Fillmore et al., 2003), a large lexical database of English that comes with sentences annotated with semantic frames. It has been considered a valuable resource for Natural Language Processing and useful for studying tasks such as information extraction, machine translation, and question answering (Surdeanu et al., 2003; Shen and Lapata, 2007; Liu and Gildea, 2010).

Here we consider the task of automatic extraction of semantic frames as defined in FrameNet. This include *target identification* – identifying frame-evoking predicates, *frame identification* – identifying which frame each predicate evokes, and *semantic role labeling* (SRL) – identifying phrasal arguments of each evoked frame and labeling them with the frame's semantic roles. Consider the sentence "We decided to treat the patient with combination chemotherapy.". Here "decided" evokes the DECIDING frame and "treat" evokes the CURE frame. Each frame takes a set of arguments that fill the semantic roles of the frame, as illustrated below:

> [We$_{\text{COGNIZER}}$] **decided** [to treat the patient with combination chemotherapy$_{\text{DECISION}}$].
>
> [We$_{\text{HEALER}}$] decided to **treat** [the patient$_{\text{PATIENT}}$] [with combination chemotherapy$_{\text{TREATMENT}}$].

We address frame identification and semantic role labeling in this work.[1] Frame identification can be addressed as a word sense disambiguation problem, while semantic role labeling can be formulated as a structured prediction problem. We train different neural network models for these two problems, and interpret their outputs as factors in a graphical model for performing joint inference over the distribution of frames and semantic roles.

Specifically, our frame identification model is a simple multi-layer neural network that learns ap-

[1] We do not consider target identification due to the lack of consistent labeled data (Das et al., 2014).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1247–1256
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

propriate feature representations for frame disambiguation. Our SRL model is an integrated model of an LSTM-based network that learns to predict semantic roles on a word-by-word basis and a multi-layer network that learns to directly predict semantic roles for individual text spans in relation to a given predicate. The sequential neural network is powerful for modeling sentence-level information while the relational neural network is good at capturing span-level dependencies between predicate and arguments. To leverage the power of these two networks, we transfer the knowledge in the sequential model, encoded as its predictive distributions. Specifically, we do this by training a single relational model with an objective that measures both its prediction accuracy with respect to the true semantic role labels, and its match to the probability distributions provided by the sequential model.

We evaluate our models for frame identification, SRL, and full structure extraction on the FrameNet 1.5 data. Our full model achieves 76.6 F1, a 5.7 absolute gain over the prior state of the art. We also evaluate our SRL model on CoNLL 2005. It demonstrates strong performance that is close to the best published results. Error analysis further confirms the benefits of integrating sequential and relational models and performing joint inference over frames and semantic roles.

2 Related Work

Research on automatic semantic structure extraction has been widely studied since the pioneering work of Gildea and Jurafsky (2002). This work focuses on extracting semantic frames defined in FrameNet (Baker et al., 1998), which includes predicting frame types and frame-specific semantic roles. Our model can be easily adapted to predict PropBank-style semantic roles (Palmer et al., 2005), where role labels are generic instead of frame-specific.

The core problem in semantic frame extraction is semantic role labeling (SRL). Earlier SRL systems employ linear classifiers which rely heavily on hand-engineered feature templates to represent argument structures (Johansson and Nugues, 2007; Das et al., 2010; Das, 2014). Recent work has exploited neural networks to learn better feature representations. Roth and Woodsend (2014) improves the feature-based system by adding word embeddings as features. Roth and Lapata (2016) further includes dependency path embeddings as features. FitzGerald et al. (2015) embeds the standard SRL features into a low-dimensional vector space using a feed-forward neural network and demonstrates state-of-the-art results on FrameNet.

Different neural network architectures have also been explored for SRL. Collobert et al. (2011) first applies a convolutional neural network to extract features from a window of words. Zhou and Xu (2015) employs a deep bi-directional LSTM (DB-LSTM) network and achieves state-of-the-art results on PropBank-style SRL. Recently, Swayamdipta et al. (2016) employs stack LSTMs (Dyer et al., 2015) for joint syntactic-semantic dependency parsing. He et al. (2017) recently proposed further improvements to the DB-LSTM architecture which significantly improve the state of the art results on PropBank SRL.

In order to enforce structural consistency, most existing work applies different types of structural constraints during inference. The inference problem are typically solved via Integer Linear Programming (ILP) (Punyakanok et al., 2008). Täckström et al. (2015) improves the inference efficiency with a dynamic programming algorithm that encodes tractable global constraints. Recently, Belanger et al. (2017) models SRL using end-to-end structured prediction energy networks and demonstrates benefits of accounting for complex structural dependencies during training. In this work, we explicitly encode structural constraints as factors in a graphical model, and adopt the Alternating Directions Dual Decomposition (AD^3) algorithm (Martins et al., 2011) for efficient inference.

3 Overview

We aim to extract frame-semantic structures from text. Each semantic frame contains a frame-evoking predicate, its frame type, the arguments of the predicate, and their semantic roles.

Both FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2005) provide sentences annotated with predicates and the semantic roles of arguments of the predicates, but there are some differences. In FrameNet, a semantic frame can be evoked by a set of lexical units. For example, the COMMERCE_BUY frame can be evoked by *buy.v*, *purchase.n*, and *purchase.v*. Each frame is also associated with a set of roles, some of which are core roles (necessary components) of the frame.

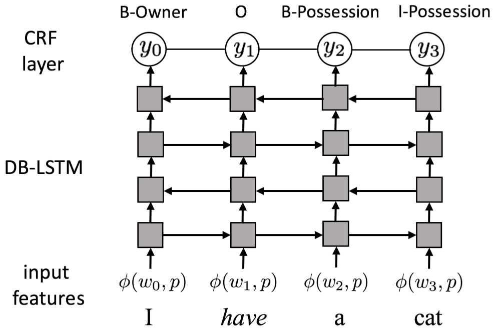


Figure 1: DB-LSTM network (four layers) with a CRF prediction layer. The network learns to predict a sequence of argument role labels given a sentence (e.g., "I have a cat") and a predicate (e.g., "have").

For example, the COMMERCE_BUY frame contains core roles such as BUYER and GOODS, and non-core roles such as MONEY and MEANS. In PropBank, a semantic frame is corresponding to a verb senses. Each verb sense is associated with a set of semantic roles. For example, the verb sense *buy.01* is associated with roles A0 (i.e., agent), A1 (i.e., patient), A2 (i.e., instrument), etc. The semantic roles in PropBank use generic labels. There are about 30 different role labels in total (vs $\sim 10^3$ role labels in FrameNet). Among them 7 are core role labels (A0-A5 and AA) and the rest are non-core (modifier) roles (e.g., the locative role LOC and the temporal role TMP).

In the rest of the paper, we first describe our models for SRL (§ 4), including a sequential neural model, a relational neural model, and the integration of the two. Then, we present our frame identification model (§ 5), followed by a joint inference algorithm for full frame-semantic structure extraction that enforces structural constraints among predicates and arguments (§ 6).

4 Semantic Role Labeling

Given a predicate and its frame, we seek to identify arguments of the predicate and their semantic roles in relation to the predicate's frame. Denote a predicate as p, its frame as f, and a sentence as x. We want to output a set of argument spans $A = \{a_1, ..., a_k\}$, where each a_i is labeled with a semantic role that takes values from a set of role labels $\mathcal{R}_f$ with respect to the frame f.

4.1 Sequential Neural Model

The SRL task can be formulated as a sequence labeling problem, where the semantic role labels are encoded using the "IOB" tagging scheme, as in (Collobert et al., 2011; Zhou and Xu, 2015), where "I" indicates the inside of a chunk, "B" indicates the beginning of a chunk, and "O" indicates being outside of a chunk.

We employ DB-LSTM, a deep bidirectional Long Short-Term Memory neural network with a Condidtional Random Field (CRF) layer introduced by Zhou and Xu (2015) for PropBank-style SRL. The architecture is illustrated in Figure 1. In this work, we adapt it to perform both FrameNet-style and PropBank-style SRL.

At each time step t, the DB-LSTM network is provided with a set of input features $\phi(w_t, p)$, including the current word w_t, the predicate word p, and a position mark that denotes whether the current word is in the neighborhood of the predicate (within a window of 5 words)[2]. Each word feature is associated with a parameter vector which is initialized using the pre-trained paraphrastic word embeddings (Wieting et al., 2015). The input representation at time step t is the concatenation of the above features. As proposed in (Zhou and Xu, 2015), we stack 8 layers of the LSTM unit to produce the hidden representation for each time step. Then, we employ a CRF layer on top to estimate the sequence-level label distributions.

During training, we minimize the negative conditional log-likelihood of N training examples. Each example consists of a sentence x, a predicate p, and a label sequence $\mathbf{y} = \{y_1, ..., y_n\}$, where n is the length of the sentence. The conditional probability is given by:

$$P_{seq}(\mathbf{y} \mid p, f; \boldsymbol{\theta}) = \frac{1}{Z_f} \exp\left(\sum_{t=1}^{n} C_{t,y_t} + \sum_{t=0}^{n} T_{y_t, y_{t+1}}\right) \tag{1}$$

where Z_f is a normalization constant depending on the frame f, as we only normalize over role label sequences that are compatible with the frame. For PropBank-style SRL, we simply drop the dependency on f and compute normalization over all possible role label sequences. C_{t,y_t} is the score output by DB-LSTM for assigning the t-th word

[2] We did not use the predicate context features as in Zhou and Xu (2015) since they did not improve performance in our implementation.

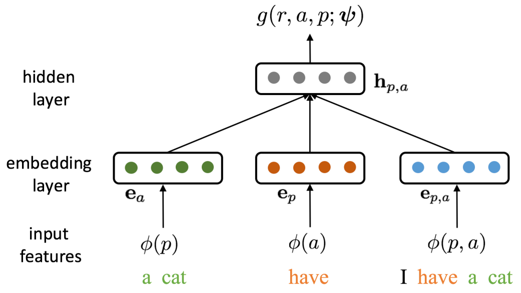


Figure 2: A relational network architecture. The network learns to predict a relation between a predicate p and an argument a given the predicate-argument pair and the sentence that contains it.

with label y_t and $T_{y_t,y_{t+1}}$ is the score of transitioning from label y_t to y_{t+1}. $\boldsymbol{\theta}$ denotes the model parameters, including the DB-LSTM parameters and the transition matrix T.

4.2 Relational Neural Model

An alternative way to formulate the SRL problem is to enumerate all possible argument spans for a given predicate and employ multi-class classification on every argument span. We describe how to obtain candidate argument spans in Section §7.2.

Denote a set of candidate argument spans as $\tilde{\mathcal{A}}$. For each argument span $a \in \tilde{\mathcal{A}}$, we seek to estimate the conditional probability given by:

$$P_{rel}(r \mid a, p, f; \boldsymbol{\psi}) = \frac{\exp(g(r, a, p; \boldsymbol{\psi}))}{\sum_{r' \in \mathcal{R}_f \cup \varnothing} \exp(g(r', a, p; \boldsymbol{\psi}))} \quad (2)$$

where $g(r, a, p; \boldsymbol{\psi})$ is a potential function for scoring the assignment of semantic role r to an argument span a with respect to predicate p, $\boldsymbol{\psi}$ denotes the model parameters, $\mathcal{R}_f$ is a set of valid semantic roles with respect to frame f and $\varnothing$ is an empty class that indicates invalid semantic roles.

We estimate g using a neural network as depicted in Figure 2. The inputs to the network are discrete features: $\phi(a)$ denotes argument-specific features, which include words within the argument span, the dependents of the argument's head, and their dependency labels; $\phi(p)$ denotes predicate-specific features, which include the predicate word, its dependents, and their dependency labels; $\phi(p, a)$ denotes predicate-argument relation features, which include the words between p and a and the lexicalized shortest dependency path.

We then map these features into a low dimensional space. Specifically, we compute an embedding of the argument features: $\mathbf{e}_a = [\bar{\mathbf{v}}_w^a; \bar{\mathbf{v}}_d^a; \bar{\mathbf{v}}_l^a]$, where $\bar{\mathbf{v}}_w^a \in \mathbb{R}^k$ is the average of argument word embeddings, $\bar{\mathbf{v}}_d^a \in \mathbb{R}^k$ is the average embedding of the argument's dependents, and $\bar{\mathbf{v}}_l^a \in \mathbb{R}^k$ is the average embedding of the corresponding dependency labels. Similarly, the embedding of the predicate features is: $\mathbf{e}_p = [\bar{\mathbf{v}}_w^p; \bar{\mathbf{v}}_d^p; \bar{\mathbf{v}}_l^p]$, which is the concatenation of the average embeddings for the predicate words, the predicate's dependents, and their dependency labels. For the relational features, we have $\mathbf{e}_{p,a} = [\bar{\mathbf{v}}_w^{pa}; \mathbf{v}_{path}]$, where $\bar{\mathbf{v}}_w^{pa} \in \mathbb{R}^k$ is the average embedding for words between p and a, and $\mathbf{v}_{path} \in \mathbb{R}^k$ is a dependency path embedding, which is the final hidden state of an LSTM network that operates over the dependency path between p and a, with the input at each time step being the concatenation of a dependency label embedding and a word embedding.

The feature embeddings are then integrated through a non-linear hidden layer:

$$\mathbf{h}_{p,a} = \text{ReLu}(\mathbf{W}_{p,a} \cdot [\mathbf{e}_a; \mathbf{e}_p; \mathbf{e}_{p,a}]) \quad (3)$$

where $\mathbf{W}_{p,a}$ is an $m \times 8k$ matrix and $\text{ReLu}(x) = \max(0, x)$. Finally, we compute the potential function: $g(r, a, p; \boldsymbol{\psi}) = \mathbf{w}_r^T \mathbf{h}_{p,a}$, where $\mathbf{w}_r \in \mathbb{R}^m$ is a weight vector to be learned.

During training, we minimize the negative conditional log-likelihood of the training examples, with the conditional probability for each example given by Eq. 2.

4.3 An Integrated Model

Our integrated model is essentially a relational neural model that is learned using the knowledge distilled from the sequential model.

Note that the sequential model estimates probabilities for semantic role label sequences over words instead of over text spans. These learned probabilities carry important information about how the sequential model learns to generalize. We identify them as the learned knowledge of the sequential model. To make use of such knowledge in the relational model, we first transform the sequence distributions into span-based distributions. Specifically, we derive the marginal distribution for any given span $a = (w_s, ..., w_t)$, $1 \leq s \leq t < n$, and a non-empty semantic role label r as:

$$\begin{aligned} &P_{seq}(r \mid a) = \\ &P_{seq}(y_s = B_r, ..., y_t = I_r, y_{t+1} \neq I_r \mid a) \end{aligned} \quad (4)$$

Here we drop the dependency on p, and f for brevity. B_r, I_r, and O denote the beginning, the inside, and the outside of the filler of role r respectively. The probability for an empty role is:

$$P_{seq}(r = \varnothing \mid a) = 1 - \sum_{r \in \mathcal{R}_f} P_{seq}(r \mid a) \quad (5)$$

After obtaining the span-based role distributions, we incorporate them into the training objective of a relational model $\tilde{P}_{rel}$ by adding a regularization term that minimizes the KL divergence:

$$L = -\log \tilde{P}_{rel}(r \mid a) + \beta KL(P_{seq} || \tilde{P}_{rel}) \quad (6)$$

which is equivalent to minimizing

$$-\log \tilde{P}_{rel}(r \mid a) + \beta \sum_{r} P_{seq}(r \mid a) \log \tilde{P}_{rel}(r \mid a)$$

where β is a weight parameter. We refer to $\tilde{P}_{rel}$ as the integrated model. At inference time, it computes the predictive distributions of semantic roles in the same way as a vanilla relational model.

5 Frame Identification

Our semantic role labeling model is conditioned on a predicate and its frame. We now describe how to estimate the probabilities of a frame f given a predicate p.

Denote $\mathcal{F}$ as a set of semantic frames, we learn to estimate the probability:

$$P_f(f \mid p) = \frac{\exp\left(u(f, p; \boldsymbol{\lambda})\right)}{\sum_{f' \in \mathcal{F}} \exp\left(u(f', p; \boldsymbol{\lambda})\right)} \quad (7)$$

The potential function $u(f, p; \boldsymbol{\lambda})$ is computed using a multi-layer neural network, whose architecture is similar to Figure 2. The input features are $\phi(p)$ as defined in §4.2. The embedding layer computes $\mathbf{e}_p$ as described above, and the hidden layer computes:

$$\mathbf{h}_p = \text{ReLu}(\mathbf{W}_p \cdot \mathbf{e}_p)$$

where $\mathbf{W}_p$ is an $m \times 3k$ matrix. The potential function is then estimated as $u(f, p; \boldsymbol{\lambda}) = \mathbf{w}_f^T \mathbf{h}_p$, where $\mathbf{w}_f \in \mathbb{R}^m$ is a weight vector to be learned. Training is done by minimizing the negative conditional log-likelihood of the training examples where the conditional probability for each example is given by Eq. 7.

6 Joint Inference

Finally, we want to jointly assign frames and roles to all predicates and their arguments.

Given a set of predicates $\mathcal{P} = \{p_1, ..., p_N\}$ and a set of candidate argument spans $\tilde{\mathcal{A}} = \{a_1, ..., a_M\}$, we optimize the following objective:

$$\underset{\text{s.t. } (\mathbf{f}, \mathbf{r}) \in \mathcal{Q}}{\arg\max} \sum_{j=1}^{N} P_f(f_j \mid p_j) \sum_{i=1}^{M} \tilde{P}_{rel}(r_i \mid a_i, p_j, f_j) \quad (8)$$

where $\mathbf{f}$ is a vector of frame assignments, $\mathbf{r}$ is a vector of role assignments, and $\mathcal{Q}$ is a constrained set of frame and role assignments.

We employ the standard structural constraints for SRL, including avoiding non-overlapping argument spans and repeated core roles for each frame. In addition, we introduce two constraints: one encodes the compatibility between frame types and semantic roles, for example, INSTRUMENT is not a valid role for the frame COMMERCIAL_TRANSACTION, and the other encodes type consistencies of semantic role fillers of different frames, e.g., the same named entity cannot play both a PERSON role and a VEHICLE role. We consider six common entity types (that are mutually exclusive): PERSON, LOCATION, WEAPON, VEHICLE, VALUE, and TIME.[3]

We solve the inference problem (8) using the AD^3 algorithm (Martins et al., 2011), which allows for more efficient constrained optimization than generic Integer Linear Programming solvers.

7 Experiment

7.1 Datasets

We evaluate our approach on semantic frame extraction using the FrameNet 1.5 release[4]. We use the same train/development/test split of the fully-annotated text documents as in previous work. We also include the partially-annotated exemplar sentences (i.e., each exemplar has only one annotated frame.) in FrameNet as training data.[5] We use the standard evaluation script that measures frame

[3] We simply check if the role name contains any of the entity type names like "person", "location". We plan to incorporate an automatic semantic typing model into our framework in future work.

[4] http://framenet.icsi.berkeley.edu

[5] Existing work also makes use of the exemplars, but mainly as a lexicon. We found that adding the exemplar sentences generally introduces a 3-4 F1 gain for FrameNet SRL.

structure extraction precision, recall and F1[6].

For PropBank-style SRL, we use the CoNLL2005 data set (Carreras and Màrquez, 2005) with the official scripts[7] for evaluation. It contains section 2-21 of WallStreet Journal (WSJ) data as training set, section 24 as development set and section 23 of WSJ concatenated with 3 sections from Brown corpus as the test set.

For data pre-processing, we parse all the sentences with the part-of-speech tagger and the dependency parser provided in the Stanford CoreNLP toolkit (Manning et al., 2014).

7.2 Argument candidate extraction

Existing work relied on either constituency syntax (Xue and Palmer, 2004) or dependency syntax (Täckström et al., 2015) to derive heuristic rules for extracting candidate arguments. Instead, we extract candidate arguments using a pre-trained sequential SRL model (described in § 4.1). Specifically, we extract the argument spans from the K-best semantic role label sequences output by the sequential model. We choose K from {5,10,20,50}. Increasing K will increase the recall of unlabeled arguments but lower the precision. We tune K based on the argument extraction performance of our relational model (in § 4.2) using the development data. In all our experiments, we set $K = 10$, which gives an unlabeled argument recall/precision of 89.6%/24.8% on FrameNet and 92.4%/29.4% on CoNLL2005.

7.3 Implementation details

All of our models are implemented using Theano on a single GPU. We set the embedding dimension k to 300 and the hidden dimension m to 100. We initialize the word embeddings using the pre-trained word embeddings from (Wieting et al., 2015) while randomly initializing the embeddings for out-of-vocabulary words and the embeddings for the dependency labels within $(-0.01, 0.01)$. All these embeddings are updated during the training process. We apply dropout to the embedding layer with rate 0.5, and train using Adam with default settings (Kingma and Ba, 2014). The weight parameter β in Eq. 6 is set to 1 in our experiments. All the models are trained for 50 epochs with early stopping based on development results.

Model	All	Ambiguous
Log-Linear Words	87.3	70.5
Log-Linear Embedding	86.7	70.3
Wsabie Embedding	**88.4**	73.1
Ours (Frame Only)	88.2	**75.7**

Table 1: Accuracy results on frame identification, including results on *all* predicates and *ambiguous* predicates in the FrameNet lexicon.

For all our experimental results, we perform statistical significance tests using the paired bootstrap test (Efron and Tibshirani, 1994) with 1000 bootstrap samples of the evaluated examples, and use $*$ to indicate statistical significance ($p < 0.05$) of the differences between our best model and our second best model.

7.4 FrameNet Results

Frame Identification. We first evaluate our frame identification model in § 5. For baselines, we consider the prior state-of-the-art approach Wsabie Embedding (Hermann et al., 2014), which learns feature representations based on word embeddings and dependency path embeddings using the Wsabie algorithm (Weston et al., 2011). We also include two strong baselines implemented in Hermann et al. (2014): Log-Linear Words and Log-Linear Embeddings, which are both log-linear models, one with standard linguistic features and one with embedding features. Table 1 shows the results.[8] We can see that our model in general gives competitive performance and it outperforms all the baselines on predicting frames for ambiguous predicates (i.e., seen with more than one possible frames in the FrameNet lexicon).

Semantic Role Labeling. Next, we evaluate our SRL models with gold-standard frames, so that we can focus on the performance for argument identification. Our SRL models include the sequential model described in § 4.1 (denoted as *Seq*); the relational model described in § 4.2 (denoted as *Rel*); and the integrated model described in § 4.3 (denoted as *Seq+Rel*).

Table 2 shows the results for argument span ex-

[6] http://www.cs.cmu.edu/~ark/SEMAFOR/eval/

[7] http://www.lsi.upc.edu/~srlconll/srl-eval.pl

[8] We consider the Full Lexicon evaluation setting and copy the results from the updated version of the paper from the author's website http://www.dipanjandas.com/pages/papers. Note that the set of ambiguous predicates we consider is different from the set used by Hermann et al. (2014). This is because we process the lexical units with the Stanford POS tagger.

Model	Prec.	Rec.	F1
SEMAFOR	65.6	53.8	59.1
SEMAFOR (best)	66.0	60.4	63.1
Ours (Seq)	63.4	**66.4**	64.9
Ours (Rel)	**71.8**	57.7	64.0
Ours (Seq+Rel)	70.2	60.2	**65.5***

Table 2: Argument only evaluation results on the FrameNet test set in comparison to the results in Kshirsagar et al. (2015).

Model	Prec.	Rec.	F1
SEMAFOR	78.4	73.1	75.7
Framat	80.3	71.7	75.8
Framat+context	80.4	73.0	76.5
Ours (Seq)	78.5	**79.9**	79.2
Ours (Rel)	**84.8**	75.5	80.0
Ours (Seq+Rel)	84.2	77.1	**80.5***

Table 3: Full structure extraction results on the FrameNet test set (with gold frames) in comparison to the results in Roth and Lapata (2015).

traction. Our baselines include SEMAFOR (Das et al., 2014)[9], a widely used frame-semantic parser for English, and SEMAFOR (BEST), an improved SEMAFOR system that is trained with heterogeneous resources (Kshirsagar et al., 2015). We can see that all of our models outperform these two systems in terms of F1, especially, our sequential model provides the best recall, our relation model provides the best precision, and our integrated model gives the best F1 score.

Table 3 shows results for full structure extraction (i.e., the accuracies of the frame-argument structure as a whole). We compare to the results reported in Roth and Lapata (2015). *Framat* is an open-source semantic role labeling tool provided by mate-tools (Björkelund et al., 2010), and *Framat+context* is an extension of *Framat* that uses additional context features. All of our models significantly outperform the baselines in F1. In particular, our integrated model achieves the best F1 score of 80.5%.

Full Semantic Structure Extraction. We now evaluate our models on full semantic frame extraction. Previous work implements the task in a two-stage pipeline: first apply a frame identification model to assign a frame to each predicate, and then apply a SRL model to assign a frame-specific role label or ∅ to each candidate argument span. We compare with previous work using four model variants: three are pipeline models that combine our frame identification model with each of our SRL models and *JointAll* is the joint model that simultaneously predicts frames and roles as described in § 6.

Model	Prec.	Rec.	F1
SEMAFOR	69.2	65.1	67.1
Framat	71.1	63.7	67.2
Framat+context	71.1	64.8	67.8
Hermann	74.3	66.0	69.9
Täckström (Struct.)	75.4	65.8	70.3
FitzGerald (Struct.)	74.8	65.5	69.9
FitzGerald (Struct., PoE)	74.6	66.3	70.2
FitzGerald (Local, PoE, Joint)	75.0	67.3	70.9
Ours (Seq)	69.6	70.9	70.2
Ours (Rel)	77.1	68.7	72.7
Ours (Seq+Rel)	77.3	71.2	74.1
Ours (JointAll)	**78.8**	**74.5**	**76.6***

Table 4: Full structure extraction results on the FrameNet test set in comparison to the previously published results.

Table 4 compares our models with previously published results. The first block shows results from Roth and Lapata (2015) and the second block shows results from FitzGerald et al. (2015). All these previous methods implements a pipeline of frame identification and semantic role labeling. The first block uses SEMAFOR for frame identification and the second block uses the WSABIE model from Hermann et al. (2014). For the semantic role labeling step, *Hermann* is a standard log-linear classification model used in Hermann et al. (2014); *Täckström (Struct.)* is a graphical model with global factors (Täckström et al., 2015); *FitzGerald (Struct.)* is an improved version of the graphical model with non-linear potential functions instead of linear ones; *FitzGerald (Struct., PoE)* further employs an ensemble with the product-of-experts (PoE) (Hinton, 2002); and *FitzGerald (Local, PoE, Joint)* indicates the best reported results in FitzGerald et al. (2015) which uses local factors and additional training data from CoNLL 2005. We can see that our sequential model alone is already close to the state of the art. Our relational model demonstrates superior performance on precision, which confirms the benefit of modeling predicate-argument interactions at the span level. The integrated model further improves over the relational model in both precision and recall. Finally, by

[9] http://www.cs.cmu.edu/~ark/SEMAFOR/

Method	Dev	WSJ	Brown
Surdeanu (Ensemble)	-	80.6	70.1
Toutanova (Ensemble)	78.6	80.3	68.8
Punyakanok (Ensemble)	77.4	79.4	67.8
Zhou (DB-LSTM)	79.6	**82.8**	69.4
Täckström (Struct.)	78.6	79.9	71.3
FitzGerald (Struct.)	78.3	79.4	71.2
FitzGerald (Struct., PoE)	78.9	80.3	**72.2**
Ours (Seq)	78.5	80.5	70.8
Ours (Rel)	79.2	81.4	71.3
Ours (Seq+Rel)	**80.3**	81.9	72.0*

Table 5: Semantic role labeling results on CoNLL 2005.

joint inference of both frames and semantic roles, our model performs even better, achieving a 5.7 absolute F1 gain over the prior state of the art.

7.5 CoNLL Results

Table 5 shows the results of our SRL models on the CoNLL 2005 data. Our baselines include the best feature-based systems of Surdeanu et al. (2007), Toutanova et al. (2008), and Punyakanok et al. (2008), the recurrent neural network model (DB-LSTM) (Zhou and Xu, 2015), the graphical model with global factors (Täckström et al., 2015) and the improved versions that use neural network factors (FitzGerald et al., 2015). Note that our sequential model in this setting is essentially the same as the DB-LSTM model (Zhou and Xu, 2015) since all the frame-specific constraints are removed, except that we use simpler input features.[10] We observe a similar performance trend among our models. However, the performance gain introduced by our integrated model is relatively small compared to our FrameNet results. Note that the argument structures in CoNLL 2005 is much simpler and less diverse than the ones in FrameNet. This may lead to less complementary information captured by the sequential model and the relational model. Overall, our integrated model achieves comparable performance to the previously published results.

7.6 Analysis

We perform further analysis of our results on FrameNet to better understand our models.

We first look at how well our models perform on sentences of different lengths. In general, longer sentences tend to have more predicates and are more likely to contain complex long-range predicate-argument dependencies. We divide the FrameNet test set into 7 bins based on sentence lengths, each with length increased by 10, and the last bin contains sentences of length > 60. Figure 3 shows the F1 scores for full structure extraction for each bin. For all our models, performance tends to degrade as sentence length increases. Interestingly, our relational model consistently outperforms our sequential model at different sentence lengths, which demonstrates its robustness of handling relations of different ranges. The combination of the two models leads to consistent performance gains, and our final joint model performs the best across different sentence lengths.

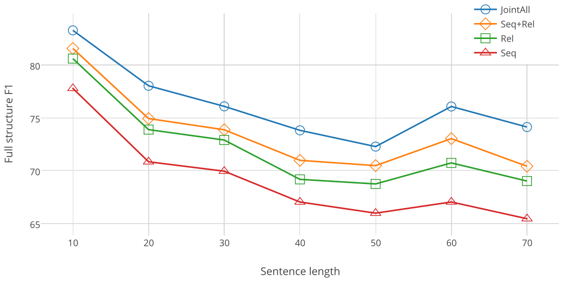


Figure 3: Full structure F1 on the FrameNet test set by the sentence length.

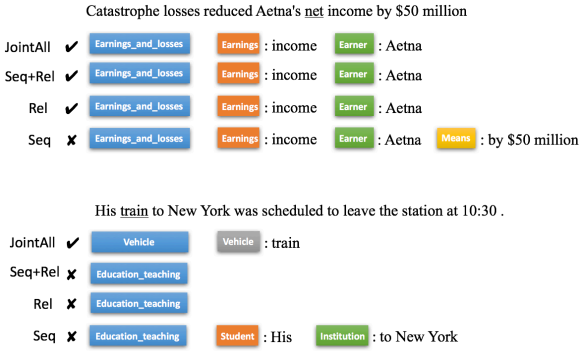

Catastrophe losses reduced Aetna's net income by $50 million
JointAll ✔ Earnings_and_losses Earnings : income Earner : Aetna
Seq+Rel ✔ Earnings_and_losses Earnings : income Earner : Aetna
Rel ✔ Earnings_and_losses Earnings : income Earner : Aetna
Seq ✘ Earnings_and_losses Earnings : income Earner : Aetna Means : by $50 million
His train to New York was scheduled to leave the station at 10:30 .
JointAll ✔ Vehicle Vehicle : train
Seq+Rel ✘ Education_teaching
Rel ✘ Education_teaching
Seq ✘ Education_teaching Student : His Institution : to New York

Figure 4: Examples of semantic frames output by different models.

Next, we analyze the errors made by different models. In general, our sequential model produces higher recall than the relational model and the integrated model, but it has lower precision. For example, for the first sentence in Figure 4, the sequential model mistakenly predicts "by $50 million" as a means to earn while both the relational and integrated models avoid this mistake. This

[10] Our reimplementation using the same feature set as Zhou and Xu (2015) did not achieve the same performance, see § 4.1 for details.

shows that performing sequential predictions over individual words has limitations. Although our relational models are good at reducing precision errors, they can be affected by frame identification errors if they are used in a pipeline. This is demonstrated by the second sentence in Figure 4, where only the *JointAll* model correctly predicts that the word "train" triggers a "Vehicle" frame. All the pipeline approaches mistakenly predict the "Education_teaching" frame in the first stage. In the second stage, the sequential model further extracts wrong semantic roles "Student" and "Institution". While the relational model and the integrated model extract no semantic roles, the frame prediction mistake remains.

8 Conclusion

We presented a new method for frame-semantic parsing that achieves the new state of the art results on standard FrameNet data. Our model integrates a sequential neural network into the learning of a relational neural network for more accurate span-based semantic role labeling. During inference, it jointly predicts frames and semantic roles using a graphical model with neural network factors. Empirical results demonstrate that our approach significantly outperforms existing neural and non-neural approaches on FrameNet data. Our model can also be adapted to perform PropBank-style SRL and it demonstrates comparable performance with the state of the art on CoNLL 2005 data.

Acknowledgments

This research was supported in part by DARPA under contract number FA8750-13-2-0005, and by NSF grants IIS-1065251 and IIS-1247489. We also gratefully acknowledge the support of the Microsoft Azure for Research program and the AWS Cloud Credits for Research program. In addition, we would like to thank anonymous reviewers for their helpful comments.

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*. pages 86–90.

David Belanger, Bishan Yang, and Andrew McCallum. 2017. End-to-end learning for structured prediction energy networks. In *ICML*.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*. Association for Computational Linguistics, pages 33–36.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *CoNLL*.

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

Dipanjan Das. 2014. Statistical models for frame-semantic parsing. In *Proceedings of Frame Semantics in NLP: A Workshop in Honor of Chuck Fillmore*. volume 1929, pages 26–29.

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational linguistics* 40(1):9–56.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A Smith. 2010. Probabilistic frame-semantic parsing. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*. Association for Computational Linguistics, pages 948–956.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL*.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Charles J Fillmore, Christopher R Johnson, and Miriam RL Petruck. 2003. Background to framenet. *International journal of lexicography* 16(3):235–250.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *EMNLP*. pages 960–970.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics* 28(3):245–288.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and whats next. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *ACL*.

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation* 14(8):1771–1800.

Richard Johansson and Pierre Nugues. 2007. Lth: semantic structure extraction using nonprojective dependency trees. In *Proceedings of the 4th international workshop on semantic evaluations*. Association for Computational Linguistics, pages 227–230.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *ICLR*.

Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime Carbonell, Noah A Smith, and Chris Dyer. 2015. Frame-semantic role labeling with heterogeneous annotations. In *ACL*.

Ding Liu and Daniel Gildea. 2010. Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 716–724.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*. pages 55–60.

André FT Martins, Mario AT Figeuiredo, Pedro MQ Aguiar, Noah A Smith, and Eric P Xing. 2011. An augmented lagrangian approach to constrained map inference. In *ICML*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics* 31(1):71–106.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics* 34(2):257–287.

Michael Roth and Mirella Lapata. 2015. Context-aware frame-semantic role labeling. *Transactions of the Association for Computational Linguistics* 3:449–460.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *ACL*.

Michael Roth and Kristian Woodsend. 2014. Composition of word representations improves semantic role labelling. In *EMNLP*. Citeseer, pages 407–413.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *EMNLP-CoNLL*. pages 12–21.

Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 8–15.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research* 29:105–151.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2016. Greedy, joint syntactic-semantic parsing with stack lstms. In *CoNLL*.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics* 3:29–41.

Kristina Toutanova, Aria Haghighi, and Christopher D Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics* 34(2):161–191.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*. volume 11, pages 2764–2770.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. In *ICLR*.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *EMNLP*. pages 88–94.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *ACL (1)*. pages 1127–1137.

Getting the Most out of AMR Parsing

Chuan Wang and **Nianwen Xue**
Brandeis University
{cwang24,xuen}@brandeis.edu

Abstract

This paper proposes to tackle the AMR parsing bottleneck by improving two components of an AMR parser: concept identification and alignment. We first build a Bidirectional LSTM based concept identifier that is able to incorporate richer contextual information to learn sparse AMR concept labels. We then extend an HMM-based word-to-concept alignment model with graph distance distortion and a rescoring method during decoding to incorporate the structural information in the AMR graph. We show integrating the two components into an existing AMR parser results in consistently better performance over the state of the art on various datasets.

1 Introduction

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a semantic representation where the meaning of a sentence is encoded as a rooted, directed graph. A number of AMR parsers have been developed in recent years (Flanigan et al., 2014; Wang et al., 2015b; Artzi et al., 2015; Pust et al., 2015; Peng et al., 2015; Zhou et al., 2016; Goodman et al., 2016a), and the initial benefit of AMR parsing has been demonstrated in various downstream applications such as Information Extraction (Pan et al., 2015; Huang et al., 2016), Machine Comprehension (Sachan and Xing, 2016), and Language Generation (Flanigan et al., 2016b; Butler, 2016). However, AMR parsing parsing accuracy is still in the high 60%, as measured by the SMatch score (Cai and Knight, 2013), and a significant improvement is needed in order for it to positively impact a larger number of applications.

Previous research has shown that concept identification is the bottleneck to further improvement of AMR parsing. For example, JAMR (Flanigan et al., 2014), the first AMR parser, is able to achieve an F-score of 80% (close to the inter-annotator agreement of 83) if gold concepts are provided. Its parsing accuracy drops sharply to 62.3% when the concepts are identified automatically.

One of the challenges in AMR concept identification is data sparsity. A large portion of AMR's concepts are either word lemmas or sense-disambiguated lemmas drawn from Propbank (Palmer et al., 2005). Since the AMR Bank is relatively small, many of the concept labels in the development or test set only occur a few times or never appear in the training set. Werling et al. (2015) addresses this problem by defining a set of *generative* actions that maps words in the sentence to their AMR concepts and use a local classifier to learn these actions. Given such sparse data, making full use of contextual information is crucial to accurate concept labeling. Bidirectional LSTM has shown its success on many sequence labeling tasks since it is able to combine contextual information from both directions and avoid manual feature engineering. However, it is non-trivial to formalize concept identification as a sequence labeling problem because of the large concept label set. Inspired by Foland and Martin (2016; 2017), who first apply the Bidirectional LSTM to AMR concept identification by categorizing the large labels into a finite set of predefined types, we propose to address concept identification using Bidirectional LSTM with **Factored Concept Labels (FCL)**, where we re-group the concept label set based on their shared graph structure. This makes it possible for different concepts to be represented by one common label that captures the shared semantics of these concepts.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1257–1268
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Accurate concept identification also crucially depends on the word-to-AMR-concept alignment. Since there is no manual alignment in the AMR annotation, typically either a rule-based or unsupervised aligner is applied to the training data to extract the mapping between words and concepts. This mapping will then be used as reference data to train concept identification models. The JAMR aligner (Flanigan et al., 2014) greedily aligns a span of words to graph fragments using a set of heuristics. While it can easily incorporate information from additional linguistic sources such as WordNet, it is not adaptable to other domains. Unsupervised aligners borrow techniques from Machine Translation and treat sentence-to-AMR alignment as a word alignment problem between a source sentence and its linearized AMR graph (Pourdamghani et al., 2014) and solve it with IBM word alignment models (Brown et al., 1993). However, the distortion model in the IBM models is based on the linear distance between source side words while the linear order of the AMR concepts has no linguistic significance, unlike word order in natural language. A more appropriate sentence-to-AMR alignment model should be one that takes the hierarchical structure of the AMR into account. We develop a Hidden Markov Model (HMM)-based sentence-to-AMR alignment method with a novel **Graph Distance** distortion model to take advantage of the structural information in AMR, and apply a structural constraint to re-score the posterior during decoding time.

We present experimental results that show incorporating these two improvements to CAMR (Wang et al., 2016), a state-of-the-art transition-based AMR parser, results in consistently better Smatch scores over the state of the art on various datasets. The rest of paper is organized as follows. Section 2 describes related work on AMR parsing. Section 3 describes our improved LSTM based concept identification model, and Section 4 describes our alignment method. We present experimental results in Section 5, and conclude in Section 6.

2 Related Work

Existing AMR parsers are either transition-based or graph-based. Transition-based AMR parsers (Wang et al., 2015b,a; Goodman et al., 2016a,b), focus on modeling the correspondence between the dependency tree and the AMR graph of a sentence by designing a small set of actions that transform the dependency tree into the AMR graph. Pust et al. (2015) formulates AMR parsing as a machine translation problem in which the sentence is the source language input and the AMR is the target language output. AMR parsing systems that focus on modeling the graph aspect of the AMR includes JAMR (Flanigan et al., 2014, 2016a; Zhou et al., 2016), which treats AMR parsing as a procedure for searching for the Maximum Spanning Connected Subgraphs (MSCGs) from an edge-labeled, directed graph of all possible relations. Parsers based on Hyperedge Replacement Grammars (HRG) (Chiang et al., 2013; Björklund et al., 2016; Groschwitz et al., 2015) put more emphasis on modeling the formal properties of the AMR graph. One practical implementation of HRG-based parsing is that of (Peng et al., 2015; Peng and Gildea, 2016). The adoption of Combinatory Categorical Grammar (CCG) in AMR parsing has also been explored in (Artzi et al., 2015; Misra and Artzi, 2016), where a number of extensions have been proposed to enable CCG to work on the broad-coverage AMR corpus.

More recently, Foland and Martin (2016; 2017) describe a neural network based model that decomposes the AMR parsing task into a series of subproblems. Their system first identifies the concepts using a Bidirectional LSTM Recurrent Neural Network (Hochreiter and Schmidhuber, 1997), and then locates and labels the arguments and attributes for each predicate, and finally constructs the AMR using the concepts and relations identified in previous steps. (Barzdins and Gosko, 2016) first applies the sequence-to-sequence model (Sutskever et al., 2014) typically used in neural machine translation to AMR parsing by simply treating the pre-order traversal of AMR as foreign language strings. (Peng et al., 2017) also adopts the sequence-to-sequence model for neural AMR parsing and focuses on reducing data sparsity in neural AMR parsing with categorization of the concept and relation labels. In contrast, (Konstas et al., 2017) adopts a different approach and tackles the data sparsity problem with a self-training procedure that can utilize a large set of unannotated external corpus. (Buys and Blunsom, 2017) design a generic transition-based system for semantic graph parsing and apply sequence-to-sequence framework to learn the transformation from natural language sequences to action sequences.

3 Concept Identification with Bidirectional LSTM

In this section, we first introduce how we categorize AMR concepts using **Factored Concept Labels**. We then integrate character-level information into a Bidirectional LSTM through Convolutional Neural Network (CNN)-based embeddings.

3.1 Background and Notation

Given a pair of AMR graph G and English sentence S, a look-up table M is first generated which maps a span of tokens to concepts using an aligner. Although there are differences among results generated by different aligners, in general, the aligned AMR concepts can be classified into the following types:

- PREDICATE. Concepts with sense tags, which are frames borrowed from Propbank, belong to this case. Most of the tokens aligned to this type are verbs and nouns that have their own argument structures.
- NON-PREDICATE. This type of concepts are mostly lemmatized word tokens from the original English sentences.
- CONST. Most of the numerical expressions in English sentences are aligned to this type, where AMR concepts are normalized numerical expressions.
- MULTICONCEPT. In this type, one or more word tokens in an English sentence are aligned to multiple concepts that form a sub-structure in an AMR graph. The most frequent case is named entity subgraphs. For example, in Figure 1, *"Mr. Vinken"* is aligned to subgraph `(p / person :name (m / name :op1 "Mr." :op2 "Vinken")`.

3.2 Factored Concept Labels

To be able to fit AMR's large concept label space into a sequence labeling framework, redefining the label set is necessary in order to make the learning process feasible. While it is trivial to categorize the PREDICATE, NON-PREDICATE, CONST cases, there is no straightforward way to deal with the MULTICONCEPT type. Foland and Martin (2016) only handle named entities, which constitute the majority of the MULTICONCEPT cases, where they adopt BIOES tags to detect the boundary and use an additional Bidirectional LSTM to learn the fine-grained named entity concept types. For other MULTICONCEPT cases, they only use the leaf concepts and ignore other parts of the subgraphs. Figure 2 shows the concept label distribution on development set of LDC2015E86, where we can see nearly half of the MULTICONCEPTcases are not named entities.

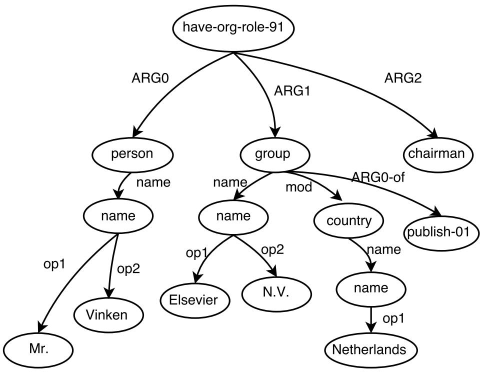


Figure 1: An example AMR graph for sentence: "*Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.*".

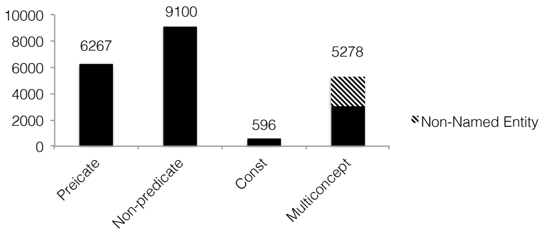


Figure 2: AMR concept label distribution for development set of LDC2015E86

Based on the observation that many of the MULTICONCEPT cases are actually similarly structured subgraphs that only differ in the lexical items, we choose to factor the lexical items out of the subgraph fragments and use the skeletal structure as the fine-grained labels, which we refer as **Factored Concept Label (FCL)**.

Figure 4 shows that although English words "visitor" and "worker" have been aligned to different subgraph fragments, after replacing the lexical items, in this case the leaf concepts `visit-01` and `work-01` with a placeholder "x", we are able to arrive at the same FCL. The strategy for determining the FCL for a word is simple: for each English word w and the subgraph s it aligns to, if the length of the longest overlapping substring be-

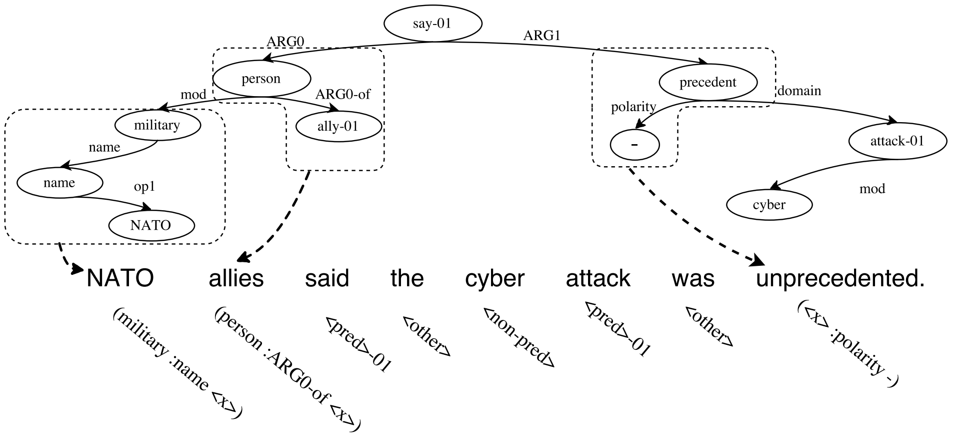


Figure 3: One example of generating FCL for sentence "*NATO allies said the cyber attack was unprecedented.*"

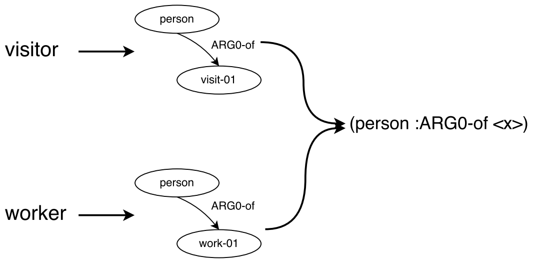


Figure 4: One example of generating FCL

tween w and the leaf concept c of s is greater than 4, we replace c with a placeholder.

Despite this simple strategy, our results show that it can capture a wide range of MULTICONCEPT cases while keeping the new label space manageable. While the named entity can be easily categorized using FCL, it can also cover some other common cases such as morphological expressions of negation (e.g., "inadequate") and comparatives (e.g., "happier"). Setting a frequency threshold to prune out the noisy labels, we are able to extract 91 canonical FCL labels on the training set. Our empirical results show that this canonical FCL label set can cover 96% of the MULTICONCEPT cases on the development set. Figure 3 gives one full example of FCLs generated for one sentence. For the PREDICATE cases, following (Foland and Martin, 2016), we only use the sense tag as its label.

We use label $\langle other\rangle$ to label stop words that do not map to AMR concepts. The MULTICONCEPT cases are handled by FCL. The FCL label set generated by this procedure can be treated as an abstraction of the original AMR concept label space, where it groups concepts that have similar AMR subgraphs into the same category.

3.3 CNN-based Character-level Embedding

After constructing the new label set with FCL, we set up a baseline Bidirectional LSTM using the concatenation of word and NER embeddings as the input. For each input word w and its NER tag t, their embeddings e_{w} and e_{t} are extracted from a word embedding matrix $W_{\text{wd}} \in \mathbb{R}^{d_{\text{wd}}\times|V_{\text{wd}}|}$ and a NER tag embedding matrix $W_{\text{t}} \in \mathbb{R}^{d_{\text{t}}\times|V_{\text{t}}|}$ respectively, where d_{wd} and d_{t} are the dimensions of the word and NER tag embedding matricies, $|V_{\text{wd}}|$ and $|V_{\text{t}}|$ are the sizes of the word and NER tag vocabulary.

Although this architecture is able to capture long-range contextual information, it fails to extract information originating from the word form itself. As we have discussed above, in some of the MULTICONCEPT cases the concepts are associated with the word forms themselves and won't benefit from its contextual information. For example, in "*unprecedented*", the prefix "*un*" itself already gives enough information to predict the FCL label $\langle x\rangle$: $polarity$ −, which indicates negative polarity. In order to incorporate such morphological and shape information, we choose to add a convolutional layer to extract character-level representations. A similar technique has been applied to Named Entity Recognition (Santos and Guimaraes, 2015; Chiu and Nichols, 2015) and we only provide a brief description of the architecture here. For a word w composed of characters $\{c_1, c_2, \ldots, c_l\}$, where l is the length of word w, we learn a character embedding matrix $W_{\text{c}} \in \mathbb{R}^{d_{\text{c}}\times|V_{\text{c}}|}$, where d_{c} is the character embedding dimension defined by the user and V_{c} is character vocabulary size. After retrieving the character embedding ch_i for each character c_{i} in word w, we obtain a sequence of vectors $\{ch_1, ch_2, \dots, ch_l\}$. This serves as the input to convolutional layer.

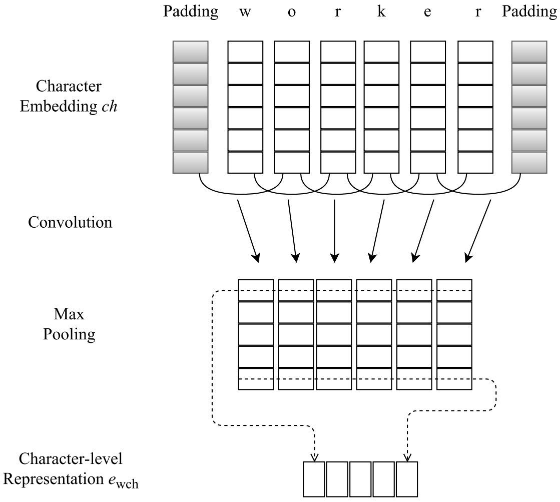


Figure 5: The architecture of the CNN-based character-level embedding.

The convolutional layer applies a linear transformation to the local context of a character in the input sequence, where the local context is parameterized by window size k. Here we define the local context of the character embedding ch_i to be:

$$f_i = (ch_{i-(k-1)/2}, \dots, ch_{i+(k-1)/2})^\top \quad (1)$$

The j-th element of the convolutional layer output vector e_{wch} is computed by element-wise max-pooling (Ranzato et al., 2007):

$$[e_{\text{wch}}]_j = \max_{1 \le i \le l} [W^0 f_i + b^0]_j \quad (2)$$

W^0 and b^0 are the parameters of the convolutional layer. And the output vector e_{wch} is the character level representation of the word w. The architecture of the model is shown in Figure 5.

The final input to the Bidirectional LSTM is the concatenation of three embeddings $[e_w, e_t, e_{\text{wch}}]$ for each word position.

4 Aligning an English Sentence to its AMR graph

Given an AMR graph G and English sentence $\boldsymbol{e} = \{e_1, e_2, \dots, e_i, \dots, e_I\}$, in order to fit them into the traditional word alignment framework, the AMR graph G is normally linearized using depth first search by printing each node as soon as it it visited. The re-entrance node is printed but not expanded to preserve the multiple mentions of concept. The relation (also called AMR role token) between concepts are preserved in the unsupervised aligner (Pourdamghani et al., 2014) because they also try to align relations to English words. We ignore the relations here since we focus on aligning concepts. Therefore the linearized concept sequences can be represented as $\boldsymbol{g} = \{g_1, g_2, \dots, g_j, \dots, g_J\}$. However, although this configuration makes it easy to adopt existing word alignment models, it also ignores the structural information in the AMR graph.

In this section, we proposes a method that incorporates the structural information in the AMR graph through a distortion model inside an HMM-based word aligner. We then further improve the model with a re-scoring method during decoding time.

4.1 HMM-based Aligner with Graph Distance Distortion

Given a sequence pair $(\boldsymbol{e}, \boldsymbol{g})$, the HMM-based word alignment model assumes that each source word is assigned to exactly one target word, and defines an asymmetric alignment for the sentence pair as $\boldsymbol{a} = \{a_1, a_2, \dots, a_i, \dots, a_I\}$, where each $a_i \in [0, J]$ is an alignment from source position i to target position a_i, $a_i = 0$ means that e_i is not aligned to any target words. Note that in the AMR to English alignment context, both the alignment and the graph structure is asymmetric, since we only have AMR graph annotation on in linearized AMR sequence g. Unlike the traditional word alignment for machine translation, here we will have different formulas for each translation direction. In this section, we only discuss the translation from English (source) to linearized AMR concepts (target) and we will discuss the AMR to English direction in the following section.

The HMM-based model breaks the generative alignment process into two factors:

$$P(\boldsymbol{e}, \boldsymbol{a} \mid \boldsymbol{g}) = \prod_{i=1}^{I} P_d(a_i \mid a_{i-1}, J) P_t(e_i | g_{a_i}) \quad (3)$$

where P_d is the distortion model and P_t is the translation model. Traditionally, the distortion probability $P_d(j \mid j', J)$ is modeled to depend only on the jump width $(j - j')$ (Vogel et al., 1996) and is defined as:

$$P_d(j \mid j', J) = \frac{ct(j - j')}{\sum_{j''=1}^{J} ct(j'' - j')} \quad (4)$$

where $ct(j - j')$ is the count of jump width. This formula simultaneously satisfies the normalization constraint and captures the *locality* assumption that words that are adjacent in the source sentence tend to align to words that are closer in the target sentence.

As the linear *locality* assumption does not hold among linearized AMR concepts, we choose instead to encode the distortion probability through graph distance, which is given by:

$$P_{\text{gd}}(j \mid j', G) = \frac{ct(d(j,j'))}{\sum_{j''} ct(d(j'',j'))} \tag{5}$$

The graph distance $d(j, j')$ is the length of shortest path on AMR graph G from concept j to concept j'. Note that we have to artificially normalize $P_{\text{gd}}(j \mid j', G)$, because unlike the linear distance between word tokens in a sentence, there can be multiple concepts that can have the same distance from the j'-th concept in the AMR graph, as pointed out in (Kondo et al., 2013).

During training, just like the original HMM-based aligner, an EM algorithm can be applied to update the parameters of the model.

4.2 Improved Decoding with Posterior Rescoring

So far, we have integrated the graph structure information into the forward direction (English to AMR). To also improve the reverse direction model (AMR to English), we choose to use the graph structure to rescore the posterior during decoding time.

Compared with Viterbi decoding, posterior thresholding has shown better results in word alignment tasks (Liang et al., 2006). Given threshold γ, for all possible alignments, we select the final alignment based on the following criteria:

$$\boldsymbol{a} = \{(i,j) : p(a_j = i \mid \boldsymbol{g}, \boldsymbol{e}) > \gamma\} \tag{6}$$

where the state probability $p(a_j = i \mid \boldsymbol{g}, \boldsymbol{e})$ is computed using the forward-backward algorithm. The forward algorithm is defined as:

$$\alpha_{j,i} \\ = \sum_{i'} \alpha_{j-1,i'} p(a_j = i \mid a_{j-1} = i') p(g_j \mid e_{a_j}) \tag{7}$$

To incorporate the graph structure, we rescale the distortion probability in reverse direction model as:

$$p_{\text{new}}(a_j = i \, | a_{j-1} = i') \\ = p(a_j = i \mid a_{j-1} = i')^{e^{\Delta d}} \tag{8}$$

where the scaling factor $\Delta d = d_j - d_{j-1}$ is the graph depth difference between the adjacent AMR concepts g_j and g_{j-1}. We also apply the same procedure for the backward computation. Note that since the model is in reverse direction, the distortion $p(a_j = i \mid a_{j-1} = i')$ here is still based on English word distance, jump width.

This rescaling procedure is based on the intuition that after we have processed the last concept g_{j-1} in some subgraph, the next concept g_j's aligned English position i is not necessarily related to the last aligned English position i'. Figure 6 illustrates this phenomenon: Although `we` and `current` are adjacent concepts in linearized AMR sequence, they are actually far away from each other in the graph (with a graph depth difference of -2). However, the distortion based on the English word distance mostly tends to choose the closer word, which may yield a very low probability for our correct answer here (the jump width between "Currently" and "our" is -6). By applying the exponential scaling factor, we are able to reduce the differences between different distortion probabilities. On the contrary, when the distortion probability is reliable (the absolute value of the graph depth difference is small), the model chooses to trust the distortion and picks the closer English word.

The rescaling factor can be viewed as a selection filter for decoding, where it relies on the graph depth difference Δd to control the effect of learned distortion probability. Note that after the rescaling, the resulting distortion probability no longer satisfies the normalization constraint. However, we only apply this during decoding time and experiments show that the typical threshold $\gamma = 0.5$ still works well for our case.

4.3 Combining Both Directions

Empirical results show that combining alignments from both directions improve the alignment quality (DeNero and Klein, 2007; Och and Ney, 2003; Liang et al., 2006). To combine the alignments, we adopt a slightly modified version of posterior thresholding, *competitive thresholding*, as proposed in (DeNero and Klein, 2007), which tends to select alignments that form a contiguous span.

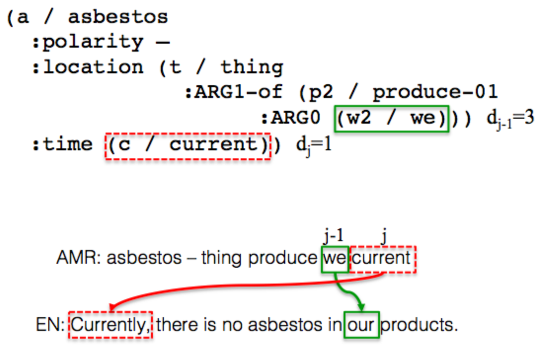

(a / asbestos
:polarity –
:location (t / thing
:ARG1-of (p2 / produce-01
:ARG0 (w2 / we))) d_{j-1}=3
:time (c / current)) d_j=1
j-1
j
AMR: asbestos – thing produce we current
EN: Currently, there is no asbestos in our products.

Figure 6: AMR graph annotation, linearized concepts for sentence "*Currently, there is no asbestos in our products*". The concept `we` in solid line is the $(j-1)$-th token in linearized AMR. It is aligned to English word "our" and its depth in graph d_{j-1} is 3. While the word distance-based distortion prefers an alignment near "our", the correct alignment needs a longer distortion.

5 Experiments

We first test the performance of our Bidirectional LSTM concept identifier and HMM-based aligner as standalone tasks, where we investigate the effectiveness of each component in AMR parsing. Then we report the final results by incorporating both components to CAMR (Wang et al., 2016). At the model development stage, we mainly use the dataset LDC2015E86 used in the SemEval Shared Task (May, 2016). Note that this dataset includes `:wiki` relations where every named entity concept is linked to its wikipedia entry. We remove this information in the training data throughout the development of our models. At the final testing stage, we add wikification using an off-the-shelf AMR wikifier (Pan et al., 2015) as a post-processing step. All AMR parsing results are evaluated using the Smatch (Cai and Knight, 2013) scorer.

5.1 Bidirectional LSTM Concept Identification Evaluation

In order to isolate the effects of our concept identifier, we first use the official alignments provided by SemEval. The alignment is generated by the unsupervised aligner described in (Pourdamghani et al., 2014). After getting the alignment table, we generate our FCL label set by filtering out noisy FCL labels that occur fewer than 30 times in the training data. The remaining FCL labels account for 96% of the MULTICONCEPT cases in the development set. Adding other labels that include PREDICATE, NON-PREDICATE and CONST gives us 116 canonical labels. UNK label is added to handle the unseen concepts.

In the Bidirectional LSTM, the hyperparameter settings are as follows: word embedding dimension $d_{wd} = 128$, NER tag embedding dimension $d_t = 8$, character embedding dimension $d_c = 50$, character level embedding dimension $d_{wch} = 50$, convolutional layer window size $k = 2$.

Input	P	R	F_1	Acc
word,NER	81.2	80.6	80.9	85.4
word,NER,CNN	83.3	82.7	83.0	87.0

Table 1: Performance of Bidirectional LSTM with different input.

Table 1 shows the performance on the development set of LDC2015E86, where the precision, recall and F-score are computed by treating $\langle other \rangle$ as the negative label and accuracy is calculated using all labels. We include accuracy here since correctly predicting words that don't invoke concepts is also important. We can see that utilizing CNN-based character level embedding yields an improvement of around 2 percentage points absolute for both F-score and accuracy, which indicates that morphological and word shape information is important for concept identification.

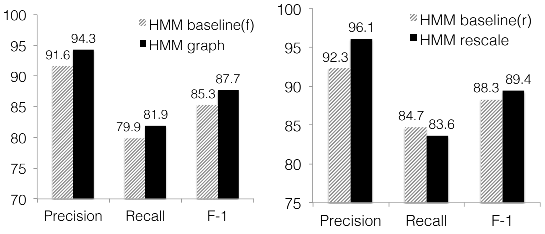
Impact on AMR Parsing In order to test the impact of our concept identification component on AMR parsing, we add the predicted concept labels as features to CAMR. Here is the detailed feature set we add to CAMR's feature templates. To clarify the notation, we refer the concept labels predicted by our concept identifier as c_{pred} and the candidate concept labels in CAMR as c_{cand}:

- `pred_label`. c_{pred} used directly as a feature.
- `is_eq_sense`. A binary feature of whether a c_{pred} and c_{cand} have the same sense (if applicable).

One reason why we choose to add the concept label and sense as features to predict the concept rather than using the predicted label to recover the concept directly is that the latter is not a straightforward process. For example, since we generalize all the predicates to a compact form `<pred-xx>`, for irregular verbs like "*became*" ⇒ `become-01`, simply stemming the inflected verb form will not give us the correct concept even if the sense is predicted correctly. However, since CAMR uses the alignment table to store all possible concept candidates for a word, adding our predicated label as a feature could potentially help the parser to choose the correct concept. In order to take full advantage of the predicted concept labels, we also extend CAMR so that it can discover candidate concepts outside of the alignment table. To achieve this, during the FCL label generation process, we first store the string-to-concept mapping as a template. For example, when we generate the FCL label `(person :ARG0-of <x>-01)` from “worker”, we also store the template `<x>er -> (person :ARG0-of <x>-01)`. Then during decoding time, we would enumerate every template and try to use the left hand side of the template (which is `<x>er`) as a regular expression to match current word. Once we find a match in all the template entries, we would substitute the placeholder in right hand side with the matched substring to get the candidate concept label. As a result, even we haven’t seen “teacher”, by matching teacher with the regular expression `(.⋆)er`, we could generate the correct answer `(person :ARG0-of teach-01)`. We refer this process as *unknown concept generation*. Table 2 summarizes the impact of our proposed methods on development set of LDC2015E86. We can see that by utilizing the unknown concept generation and features derived from c_{pred} , both precision and recall improve by about 1 percentage point, which indicates that the new feature brings richer information to the concept prediction model to help correctly score candidate concepts from the alignment table.

Parsers	P	R	F_1
CAMR (Wang et al., 2016)	72.3	61.4	66.5
CAMR-*gen*	72.1	62.0	66.6
CAMR-*gen*-c_{pred}	**73.6**	**62.6**	**67.6**

Table 2: Performance of AMR parsing with c_{pred} features without *wikification* on dev set of LDC2015E86. The first row is performance of the baseline parser. The second row adds unknown concept generation and the last row additionally extends the baseline parser with c_{pred} features.



(a) HMM forward

(b) HMM reverse

Figure 7: Our improved forward (graph) and reverse(rescale) model compared with HMM baseline on hand aligned development set.

5.2 HMM-based AMR-to-English Aligner Evaluation

To validate the effectiveness of our proposed alignment methods, we first evaluate our forward (English-to-AMR) and reverse (AMR-to-English) aligners against the baseline HMM word alignment model, which is the Berkeley aligner toolkit (DeNero and Klein, 2007). Then we combine the forward and reverse alignment results using competitive thresholding. We set the threshold γ to be 0.5 in the following experiments. To evaluate the alignment quality, we use 200 hand-aligned sentences from (Pourdamghani et al., 2014) split equally as the development and test sets. We process the English sentences by removing stop words, following similar procedure as in (Pourdamghani et al., 2014). When linearizing AMR graphs, we instead remove all the relations and only keep the concepts. For all models, we run 5 iterations of IBM Model 1 and 2 iterations of HMM on the whole dataset.

From Figure 7a, we can see that our graph-distance based model improves both the precision and recall by a large margin, which indicates the graph distance distortion better fits the English-to-AMR alignment task. For the reverse model, although our HMM rescaling model loses accuracy in recall, it is able to improve the precision by around 4 percentage points, which confirms our intuition that the rescoring factor is able to keep reliable alignments and penalize unreliable ones. We then combine our forward and reverse alignment result using competitive thresholding. Table 3 shows the combined result against hand-aligned dev and test sets.

Datasets	P	R	F_1
dev	97.7	84.3	90.5
test	96.9	84.6	90.3

Table 3: Combined HMM alignment result evaluation.

Dataset	Aligner	P	R	F_1
Dev	JAMR	73.6	62.6	67.6
	ISI	72.8	**64.6**	**68.4**
	Our HMM	**73.6**	63.6	68.2
Test	JAMR	71.6	61.3	66.0
	ISI	70.6	**62.7**	66.4
	Our HMM	**72.1**	62.5	**67.0**

Table 4: AMR parsing result (without *wikification*) with different aligner on development and test of LDC2015E86, where JAMR is the rule-based aligner, ISI is the modified IBM Model 4 aligner

Impact on AMR Parsing To investigate our aligner's contribution to AMR parsing, we replace the alignment table generated by the best performing aligner (the forward and reverse combined) in the previous section and re-train CAMR with the predicted concept label features included.

From Table 4, we can see that the unsupervised aligner (ISI and HMM) generally outperforms the JAMR rule-based aligner, and our improved HMM aligner is more consistent than the ISI aligner (Pourdamghani et al., 2014), which is a modified version of IBM Model 4.

5.3 Comparison with other Parsers

We first add the *wikification* information to the parser output using the off-the-shelf AMR wikifier (Pan et al., 2015) and compare results with the state-of-the-art parsers in 2016 SemEval Shared Task. We also report our result on the previous release (LDC2014T12), AMR annotation Release 1.0, which is another popular dataset that most of the existing parsers report results on. Note that the Release 1.0 annotation doesn't include *wiki* information.

Dataset	Parsers	P	R	F_1
SemEval Test	CAMR	70.3	63.1	66.5
	RIGA	-	-	67.2
	JAMR(2016a)	70	65	67
	Our parser	**71.7**	**64.9**	**68.1**
SemEval Blind Test	CAMR	67.4	57.3	62.0
	RIGA	68.0	57.0	62.0
	JAMR(2016a)	-	-	56
	Our parser	**68.2**	**59.5**	**63.6**

Table 5: Comparison with the winning systems in SemEval (**with** *wikification*) on test and blind test sets

CAMR and RIGA (Barzdins and Gosko, 2016) are the two best performing parsers that participated in SemEval 2016 shared task. While we use CAMR as our baseline system, the parser from RIGA is also based on a version of CAMR extended with a error-correction wrapper and an ensemble with a character-level neural sequence-to-sequence model. Our parser outperforms both systems by around 1.5 percentage points, where the improvement in recall is more significant, at around 2 percentage points.

Parsers	P	R	F_1
CAMR	71.3	62.2	66.5
(Zhou et al., 2016)	70	62	66
(Pust et al., 2015)	-	-	65.8
Our parser	**72.7**	**64.0**	**68.07**

Table 6: Comparison with the existing parsers on full test set of LDC2014T12

Table 6 shows the performance of our parser on the full test set of LDC2014T12. We include the previous best results on this dataset. The parser proposed in (Zhou et al., 2016) jointly learns the concept and relation through an incremental joint model. We also include the AMR parser by (Pust et al., 2015) that models AMR parsing as a machine translation task and incorporates various external resources. Our parser still achieves the best result without incorporating external resources other than the NER information.

6 Conclusion

In this paper, we presents work that improves AMR parsing performance by focusing on two components of the parser: concept identification and alignment. We first build a Bidirectional LSTM based concept identifier which is able to incorporate richer context and learn sparse concept labels. Then we extend the HMM-based word alignment model with a graph distance distortion and a rescoring method during decoding to incorporate the graph structure information. By integrating the two components into an existing AMR parser, our parser is able to outperform state-of-the-art AMR parsers and establish a new state of the art.

Acknowledgments

We want to thank the anonymous reviewers for their suggestions and detailed error checking.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Guntis Barzdins and Didzis Gosko. 2016. Riga at semeval-2016 task 8: Impact of smatch extensions and character-level neural translation on amr parsing accuracy. *arXiv preprint arXiv:1604.01278*.

Henrik Björklund, Frank Drewes, and Petter Ericson. 2016. Between a rock and a hard place–uniform parsing for hyperedge replacement dag grammars. In *International Conference on Language and Automata Theory and Applications*, pages 521–532. Springer.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Alastair Butler. 2016. Deterministic natural language generation from meaning representations for machine translation. In *Proceedings of the 2nd Workshop on Semantics-Driven Machine Translation (SedMT 2016)*, pages 1–9. Association for Computational Linguistics.

Jan Buys and Phil Blunsom. 2017. Robust incremental neural semantic graph parsing. *CoRR*, abs/1704.07092.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752. Association for Computational Linguistics.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proceedings of the 51st Meeting of the Association of Computational Linguistics*, Sofia, Bulgaria.

Jason P. C. Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *CoRR*, abs/1511.08308.

John DeNero and Dan Klein. 2007. Tailoring Word Alignments to Syntactic Machine Translation. *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 17–24.

Jeffrey Flanigan, Chris Dyer, A. Noah Smith, and Jaime Carbonell. 2016a. CMU at SemEval-2016 task 8: Graph-based AMR Parsing with Infinite Ramp Loss. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1202–1206. Association for Computational Linguistics.

Jeffrey Flanigan, Chris Dyer, A. Noah Smith, and Jaime Carbonell. 2016b. Generation from Abstract Meaning Representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739. Association for Computational Linguistics.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland. Association for Computational Linguistics.

William Foland and H. James Martin. 2016. CU-NLP at SemEval-2016 Task 8: AMR parsing using LSTM-based recurrent neural networks. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1197–1201. Association for Computational Linguistics.

William Foland and James Martin. 2017. Abstract meaning representation parsing using lstm recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association of the Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016a. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–11. Association for Computational Linguistics.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016b. UCL+Sheffield at SemEval-2016 task 8: Imitation learning for amr parsing with an alpha-bound. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1167–1172. Association for Computational Linguistics.

Jonas Groschwitz, Alexander Koller, Christoph Teichmann, et al. 2015. Graph parsing with s-graph grammars. In *ACL (1)*, pages 1481–1490.

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

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, R. Clare Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 258–268. Association for Computational Linguistics.

Shuhei Kondo, Kevin Duh, and Yuji Matsumoto. 2013. Hidden markov tree model for word alignment. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 503–511.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: sequence-to-sequence models for parsing and generation. *CoRR*, abs/1704.08381.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 104–111. Association for Computational Linguistics.

Jonathan May. 2016. Semeval-2016 task 8: Meaning representation parsing. *Proceedings of SemEval*, pages 1063–1073.

Dipendra Kumar Misra and Yoav Artzi. 2016. Neural shift-reduce ccg semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1775–1786, Austin, Texas. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1130–1139.

Xiaochang Peng and Daniel Gildea. 2016. UofR at SemEval-2016 Task 8: Learning synchronous hyperedge replacement grammar for AMR parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1185–1189. Association for Computational Linguistics.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for AMR parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*.

Xiaochang Peng, Chuan Wang, Daniel Gildea, and Nianwen Xue. 2017. Addressing the data sparsity issue in neural amr parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 366–375, Valencia, Spain. Association for Computational Linguistics.

Nima Pourdamghani, Yang Gao, Ulf Hermjakob, and Kevin Knight. 2014. Aligning English strings with abstract meaning representation graphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 425–429.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing English into abstract meaning representation using syntax-based machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1143–1154, Lisbon, Portugal. Association for Computational Linguistics.

M. Ranzato, F. J. Huang, Y. L. Boureau, and Y. LeCun. 2007. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Mrinmaya Sachan and Eric Xing. 2016. Machine comprehension using rich semantic representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 486–492. Association for Computational Linguistics.

Cicero Nogueira dos Santos and Victor Guimaraes. 2015. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 836–841. Association for Computational Linguistics.

Chuan Wang, Sameer Pradhan, Xiaoman Pan, Heng Ji, and Nianwen Xue. 2016. CAMR at semeval-2016 task 8: An extended transition-based AMR parser. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1173–1178, San Diego, California. Association for Computational Linguistics.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, pages 857–862.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375, Denver, Colorado. Association for Computational Linguistics.

Keenon Werling, Gabor Angeli, and Christopher Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. *arXiv preprint arXiv:1506.03139*.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu. 2016. AMR Parsing with an Incremental Joint Model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 680–689, Austin, Texas. Association for Computational Linguistics.

AMR Parsing using Stack-LSTMs

Miguel Ballesteros **Yaser Al-Onaizan**
IBM T.J Watson Research Center,
1101 Kitchawan Road, Route 134 Yorktown Heights, NY 10598. U.S
miguel.ballesteros@ibm.com, onaizan@us.ibm.com

Abstract

We present a transition-based AMR parser that directly generates AMR parses from plain text. We use Stack-LSTMs to represent our parser state and make decisions greedily. In our experiments, we show that our parser achieves very competitive scores on English using only AMR training data. Adding additional information, such as POS tags and dependency trees, improves the results further.

1 Introduction

Transition-based algorithms for natural language parsing (Yamada and Matsumoto, 2003; Nivre, 2003, 2004, 2008) are formulated as a series of decisions that read words from a buffer and incrementally combine them to form syntactic structures in a stack. Apart from dependency parsing, these models, also known as shift-reduce algorithms, have been successfully applied to tasks like phrase-structure parsing (Zhang and Clark, 2011; Dyer et al., 2016), named entity recognition (Lample et al., 2016), CCG parsing (Misra and Artzi, 2016) joint syntactic and semantic parsing (Henderson et al., 2013; Swayamdipta et al., 2016) and even abstract-meaning representation parsing (Wang et al., 2015b,a; Damonte et al., 2016).

AMR parsing requires solving several natural language processing tasks; mainly named entity recognition, word sense disambiguation and joint syntactic and semantic role labeling.[1] Given the difficulty of building an end-to-end system, most prior work is based on pipelines or heavily dependent on precalculated features (Flanigan et al., 2014; Zhou et al., 2016; Werling et al., 2015; Wang et al., 2015b, inter-alia).

Inspired by Wang et al. (2015b,a); Goodman et al. (2016); Damonte et al. (2016) and Dyer et al. (2015), we present a shift-reduce algorithm that produces AMR graphs directly from plain text. Wang et al. (2015b,a); Zhou et al. (2016); Goodman et al. (2016) presented transition-based tree-to-graph transducers that traverse a dependency tree and transforms it to an AMR graph. Damonte et al. (2016)'s input is a sentence and it is therefore more similar (with a different parsing algorithm) to our approach, but their parser relies on external tools, such as dependency parsing, semantic role labeling or named entity recognition.

The input of our parser is plain text sentences and, through rich word representations, it predicts all actions (in a single algorithm) needed to generate an AMR graph representation for an input sentence; it handles the detection and annotation of named entities, word sense disambiguation and it makes connections between the nodes detected towards building a predicate argument structure. Even though the system that runs with just words is very competitive, we further improve the results incorporating POS tags and dependency trees into our model.

Stack-LSTMs[2] have proven to be useful in tasks related to syntactic and semantic parsing (Dyer et al., 2015, 2016; Swayamdipta et al., 2016) and named entity recognition (Lample et al., 2016). In this paper, we demonstrate that they can be effectively used for AMR parsing as well.

2 Parsing Algorithm

Our parsing algorithm makes use of a STACK (that stores AMR nodes and/or words) and a BUFFER that contains the words that have yet to be processed. The parsing algorithm is inspired from

[1] Check (Banarescu et al., 2013) for a complete description of AMR graphs.

[2] We use the dynamic framework of Neubig et al. (2017) to implement our parser.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1269–1275
Copenhagen, Denmark, September 7–11, 2017.

the semantic actions presented by Henderson et al. (2013), the transition-based NER algorithm by Lample et al. (2016) and the arc-standard algorithm (Nivre, 2004). As in (Ballesteros and Nivre, 2013) the buffer starts with the root symbol at the end of the sequence. Figure 2 shows a running example. The transition inventory is the following:

- SHIFT: pops the front of the BUFFER and push it to the STACK.
- CONFIRM: calls a subroutine that predicts the AMR node corresponding to the top of the STACK. It then pops the word from the STACK and pushes the AMR node to the STACK. An example is the prediction of a propbank sense: From *occured* **to** *occur-01*.
- REDUCE: pops the top of the STACK. It occurs when the word/node at the top of the stack is complete (no more actions can be applied to it). Note that it can also be applied to words that do not appear in the final output graph, and thus they are directly discarded.
- MERGE: pops the two nodes at the top of the STACK and then it merges them, it then pushes the resulting node to the top of STACK. Note that this can be applied recursively. This action serves to get multiword named entities (e.g. *New York City*).
- ENTITY(label): labels the node at the top of the STACK with an entity label. This action serves to label named entities, such as *New York City* or *Madrid* and it is normally run after MERGE when it is a multi-word named entity, or after SHIFT if it is a single-word named entity.
- DEPENDENT(label,node): creates a new node in the AMR graph that is dependent on the node at the top of the STACK. An example is the introduction of a negative *polarity* to a given node: From *illegal* **to** (*legal, polarity* -).
- LA(label) and RA(label): create a left/right arc with the top two nodes at the top of the STACK. They keep both the head and the dependent in the stack to allow reentrancies (multiple incoming edges). The head is now a composition of the head and the dependent. They are enriched with the AMR label.
- SWAP: pops the two top items at the top of the STACK, pushes the second node to the front of the BUFFER, and pushes the first one back into the STACK. This action allows non-projective arcs as in (Nivre, 2009) but it also helps to introduce reentrancies. At oracle time, SWAP is produced when the word at the top of the stack is blocking actions that may happen between the second element at the top of the stack and any of the words in the buffer.

Figure 1 shows the parser actions and the effect on the parser state (contents of the stack, buffer) and how the graph is changed after applying the actions.

We implemented an oracle that produces the sequence of actions that leads to the gold (or close to gold) AMR graph. In order to map words in the sentences to nodes in the AMR graph we need to align them. We use the JAMR aligner provided by Flanigan et al. (2014).[3] It is important to mention that even though the aligner is quite accurate, it is not perfect, producing a F1 score of around 0.90. This means that most sentences have at least one alignment error which implies that our oracle is not capable of perfectly reproducing all AMR graphs. This has a direct impact on the accuracy of the parser described in the next section since it is trained on sequences of actions that are not perfect. The oracle achieves 0.895 F1 Smatch score (Cai and Knight, 2013) when it is run on the development set of the LDC2014T12.

The algorithm allows a set of different constraints that varies from the basic ones (not allowing impossible actions such as SHIFT when the buffer is empty or not generating arcs when the words have not yet been CONFIRMed and thus transformed to nodes) to more complicated ones based on the propbank candidates and number of arguments. We choose to constrain the parser to the basic ones and let it learn the more complicated ones.

3 Parsing Model

In this section, we revisit Stack-LSTMs, our parsing model and our word representations.

[3] We used the latest version of the aligner (Flanigan et al., 2016)

Stack$_t$	**Buffer**$_t$	**Action**	**Stack**$_{t+1}$	**Buffer**$_{t+1}$	**Graph**
S	u, B	SHIFT	u, S	B	—
u, S	B	CONFIRM	n, S	B	—
u, S	B	REDUCE	S	B	—
u, v, S	B	MERGE	$(u, v), S$	B	—
u, S	B	ENTITY(l)	$(u : l), S$	B	—
u, S	B	DEPENDENT(r, d)	u, S	B	$u \xrightarrow{r} d$
u, v, S	B	RA(r)	u, v, S	B	$u \xrightarrow{r} v$
u, v, S	B	LA(r)	u, v, S	B	$u \xleftarrow{r} v$
u, v, S	B	SWAP	u, S	v, B	—

Figure 1: Parser transitions indicating the action applied to the stack and buffer and the resulting state.

ACTION	STACK	BUFFER
INIT		It, should, be, vigorously, advocated, R
SHIFT	it	should, be, vigorously, advocated, R
CONFIRM	it	should, be, vigorously, advocated, R
SHIFT	should, it	be, vigorously, advocated, , R
CONFIRM	recommend-01, it	be, vigorously, advocated, R
SWAP	recommend-01	it, be, vigorously, advocated, R
SHIFT	it, recommend-01	be, vigorously, advocated, R
SHIFT	be, it, recommend-01	vigorously, advocated, R
REDUCE	it, recommend-01	vigorously, advocated, R
SHIFT	vigorously, it, recommend-01	advocated, R
CONFIRM	vigorous, it, recommend-01	advocated, R
SWAP	vigorous, recommend-01	it, advocated, R
SWAP	vigorous	recommend-01, it, advocated, R
SHIFT	recommend-01, vigorous	it, advocated, R
SHIFT	it, recommend-01, vigorous	advocated , R
SHIFT	it, recommend-01, vigorous	advocated, R
SHIFT	advocated, it, recommend-01, vigorous	R
CONFIRM	advocate-01, it, recommend-01, vigorous	R
LA(ARG1)	advocate-01, it, recommend-01, vigorous	R
SWAP	advocate-01, recommend-01, vigorous	it R
SHIFT	it, advocate-01, recommend-01, vigorous	R
REDUCE	advocate-01, recommend-01, vigorous	R
RA(ARG1)	advocate-01, recommend-01, vigorous	R
SWAP	advocate-01, vigorous	recommend-01, R
SHIFT	recommend01, advocate-01, vigorous	R
SHIFT	R, recommend01, advocate-01, vigorous	
LA(root)	R, recommend01, advocate-01, vigorous	
REDUCE	recommend01, advocate-01, vigorous	
REDUCE	advocate-01, vigorous	
LA(manner)	advocate-01, vigorous	
REDUCE	vigorous	
REDUCE		

```
(r / recommend-01
    :ARG1 (a / advocate-01
          :ARG1 (i / it)
          :manner (v / vigorous)))
```

Figure 2: Transition sequence for the sentence *It should be vigorously advocated.* R represents the root symbol

3.1 Stack-LSTMs

The **stack LSTM** is an augmented LSTM (Hochreiter and Schmidhuber, 1997; Graves, 2013) that allows adding new inputs in the same way as LSTMs but it also provides a POP operation that moves a pointer to the previous element. The output vector of the LSTM will consider the stack pointer instead of the rightmost position of the sequence.[4]

3.2 Representing the State and Making Parsing Decisions

The state of the algorithm presented in Section 2 is represented by the contents of the STACK, BUFFER and a list with the history of actions (which are encoded as Stack-LSTMs).[5] All of this forms the vector $\mathbf{s}_t$ that represents the state which s calculated as follows:

$$\mathbf{s}_t = \max\left\{\mathbf{0}, \mathbf{W}[\mathbf{st}_t; \mathbf{b}_t; \mathbf{a}_t] + \mathbf{d}\right\},$$

where $\mathbf{W}$ is a learned parameter matrix, $\mathbf{d}$ is a bias term and $\mathbf{st}_t$, $\mathbf{b}_t$,$\mathbf{a}_t$ represent the output vector of the Stack-LSTMs at time t.

Predicting the Actions: Our model then uses the vector $\mathbf{s}_t$ for each timestep t to compute the probability of the next action as:

$$p(z_t \mid \mathbf{s}_t) = \frac{\exp\left(\mathbf{g}_{z_t}^\top \mathbf{s}_t + q_{z_t}\right)}{\sum_{z' \in \mathcal{A}} \exp\left(\mathbf{g}_{z'}^\top \mathbf{s}_t + q_{z'}\right)} \quad (1)$$

where $\mathbf{g}_z$ is a column vector representing the (output) embedding of the action z, and q_z is a bias term for action z. The set $\mathcal{A}$ represents the actions listed in Section 2. Note that due to parsing constraints the set of possible actions may vary. The total number of actions (in the LDC2014T12 dataset) is 478; note that they include all possible labels (in the case of LA and RA) and the different dependent nodes for the DEPENDENT action

Predicting the Nodes: When the model selects the action CONFIRM, the model needs to decide the AMR node[6] that corresponds to the word at

[4] We refer interested readers to (Dyer et al., 2015) for further details.

[5] Word representations, input and hidden representations have 100 dimensions, action and label representations are of size 20.

[6] When the word at the top of stack is an out of vocabulary word, the system directly outputs the word itself as AMR node.

the top of the STACK, by using $\mathbf{s}_t$, as follows:

$$p(e_t \mid \mathbf{s}_t) = \frac{\exp\left(\mathbf{g}_{e_t}^{\top}\mathbf{s}_t + q_{e_t}\right)}{\sum_{e' \in \mathcal{N}} \exp\left(\mathbf{g}_{e'}^{\top}\mathbf{s}_t + q_{e'}\right)} \quad (2)$$

where $\mathcal{N}$ is the set of possible candidate nodes for the word at the top of the STACK. $\mathbf{g}_e$ is a column vector representing the (output) embedding of the node e, and q_e is a bias term for the node e. It is important to mention that this implies finding a propbank sense or a lemma. For that, we rely entirely on the AMR training set instead of using additional resources.

Given that the system runs two softmax operations, one to predict the action to take and the second one to predict the corresponding AMR node, and they both share LSTMs to make predictions, we include an additional layer with a *tanh* nonlinearity after $\mathbf{s}_t$ for each softmax.

3.3 Word Representations

We use character-based representations of words using bidirectional LSTMs (Ling et al., 2015b; Ballesteros et al., 2015). They learn representations for words that are orthographically similar. Note that they are updated with the updates to the model. Ballesteros et al. (2015) and Lample et al. (2016) demonstrated that it is possible to achieve high results in syntactic parsing and named entity recognition by just using character-based word representations (not even POS tags, in fact, in some cases the results with just character-based representations outperform those that used explicit POS tags since they provide similar vectors for words with similar/same morphosyntactic tag (Ballesteros et al., 2015)); in this paper we show a similar result given that both syntactic parsing and named-entity recognition play a central role in AMR parsing.

These are concatenated with pretrained word embeddings. We use a variant of the skip n-gram model provided by Ling et al. (2015a) with the LDC English Gigaword corpus (version 5). These embeddings encode the syntactic behavior of the words (see (Ling et al., 2015a)).

More formally, to represent each input token, we concatenate two vectors: a learned character-based representation ($\tilde{\mathbf{w}}_{\mathrm{C}}$); and a fixed vector representation from a neural language model ($\tilde{\mathbf{w}}_{\mathrm{LM}}$). A linear map ($\mathbf{V}$) is applied to the resulting vector and passed through a component-wise ReLU,

$$\mathbf{x} = \max\left\{\mathbf{0}, \mathbf{V}[\tilde{\mathbf{w}}_{\mathrm{C}}; \tilde{\mathbf{w}}_{\mathrm{LM}}] + \mathbf{b}\right\}.$$

where $\mathbf{V}$ is a learned parameter matrix, b is a bias term and $\mathbf{w}_{\mathrm{C}}$ is the character-based learned representation for each word, $\tilde{\mathbf{w}}_{\mathrm{LM}}$ is the pretrained word representation.

3.4 POS Tagging and Dependency Parsing

We may include preprocessed POS tags or dependency parses to incorporate more information into our model. For the POS tags we use the Stanford tagger (Toutanova et al., 2003) while we use the Dyer et al. (2015)'s Stack-LSTM parser trained on the English CoNLL 2009 dataset (Hajič et al., 2009) to get the dependencies.

POS tags: The POS tags are preprocessed and a learned representation $\mathbf{tag}$ is concatenated with the word representations. This is the same setting as (Dyer et al., 2015).

Dependency Trees: We use them in the same way as POS tags by concatenating a learned representation $\mathbf{dep}$ of the dependency label to the parent with the word representation. Additionally, we enrich the state representation $\mathbf{s}_t$, presented in Section 3.2. If the two words at the top of the STACK have a dependency between them, $\mathbf{s}_t$ is enriched with a learned representation that indicates that and the direction; otherwise $\mathbf{s}_t$ remains unchanged. $\mathbf{s}_t$ is calculated as follows:

$$\mathbf{s}_t = \max\left\{\mathbf{0}, \mathbf{W}[\mathbf{st}_t; \mathbf{b}_t; \mathbf{a}_t; \mathbf{dep}_t] + \mathbf{d}\right\},$$

where $\mathbf{dep}_t$ is the learned vector that represents that there is an arc between the two top words at the top of the stack.

4 Experiments and Results

We use the LDC2014T12 dataset[7] for our experiments. Table 1 shows results, including comparison with prior work that are also evaluated on the same dataset.[8]

[7] This dataset is a standard for comparison and has been used for evaluation in recent papers like (Wang et al., 2015a; Goodman et al., 2016; Zhou et al., 2016). We use the standard training/development/test split: 10,312 sentences for training, 1,368 sentences for development and 1,371 sentences held-out for testing.

[8] The first entry for Damonte et al. is calculated using a pretrained LDC2015 model, available at https://github.com/mdtux89/amr-eager, but evaluated on the LDC2014 dataset. This means that the score is not directly comparable with the rest. The second entry (0.64) for Damonte et al. is calculated by training their parser with the LDC2014 training set which makes it directly comparable with the rest of the parsers.

Model	$\mathbf{F_1}$(Newswire)	$\mathbf{F_1}$(ALL)
Flanigan et al. (2014)* (POS, DEP)	0.59	0.58
Flanigan et al. (2016)* (POS, DEP, NER)	0.62	0.59
Werling et al. (2015)* (POS, DEP, NER)	0.62	–
Damonte et al. (2016)[8](POS, DEP, NER, SRL)	–	0.61
Damonte et al. (2016)[8](POS, DEP, NER, SRL)	–	0.64
Artzi et al. (2015) (POS, CCG)	0.66	–
Goodman et al. (2016)* (POS, DEP, NER)	0.70	–
Zhou et al. (2016)* (POS, DEP, NER, SRL)	**0.71**	**0.66**
Pust et al. (2015) (LM, NER)	–	0.61
Pust et al. (2015) (Wordnet, LM, NER)	–	**0.66**
Wang et al. (2015b)* (POS, DEP, NER)	0.63	0.59
Wang et al. (2015a)* (POS, DEP, NER, SRL)	0.70	**0.66**
OUR PARSER (NO PRETRAINED-NO CHARS)	0.64	0.59
OUR PARSER (NO PRETRAINED-WITH CHARS)	0.66	0.61
OUR PARSER (WITH PRETRAINED-NO CHARS)	0.66	0.62
OUR PARSER	0.68	0.63
OUR PARSER (POS)	0.68	0.63
OUR PARSER (POS, DEP)	0.69	0.64

Table 1: AMR results on the LDC2014T12 dataset; Newsire section (left) and full (right). Rows labeled with OUR-PARSER show our results. POS indicates that the system uses preprocessed POS tags, DEP indicates that it uses preprocessed dependency trees, SRL indicates that it uses preprocessed semantic roles, NER indicates that it uses preprocessed named entitites. LM indicates that it uses a LM trained on AMR data and WordNet indicates that it uses WordNet to predict the concepts. Systems marked with * are pipeline systems that require a dependency parse as input. (WITH PRETRAINED-NO CHARS) shows the results of our parser without character-based representations. (NO PRETRAINED-WITH CHARS) shows results without pretrained word embeddings. (NO PRETRAINED-NO CHARS) shows results without character-based representations and without pretrained word embeddings. The rest of our results include both pretrained embeddings and character-based representations.

Our model achieves 0.68 F1 in the newswire section of the test set just by using character-based representations of words and pretrained word embeddings. All prior work uses lemmatizers, POS taggers, dependency parsers, named entity recognizers and semantic role labelers that use additional training data while we achieve competitive scores without that. Pust et al. (2015) reports 0.66 F1 in the full test by using WordNet for concept identification, but their performance drops to 0.61 without WordNet. It is worth noting that we achieved 0.64 in the same test set without WordNet. Wang et al. (2015b,a) without SRL (via Propbank) achieves only 0.63 in the newswire test set while we achieved 0.69 without SRL (and 0.68 without dependency trees).

In order to see whether pretrained word embeddings and character-based embeddings are useful we carried out an ablation study by showing the results of our parser with and without character-based representations (replaced by standard lookup table learned embeddings) and with and without pretrained word embeddings. By looking at the results of the parser without character-based embeddings but with pretrained word embeddings we observe that the character-based representation of words are useful since they help to achieve 2 points better in the Newswire dataset and 1 point more in the full test set. The parser with character-based embeddings but without pretrained word embeddings, the parser has more difficulty to learn and only achieves 0.61 in the full test set. Finally, the model that does not use neither character-based embeddings nor pretrained word embeddings is the worst achieving only 0.59 in the full test set, note that this model has no explicity way of getting any syntactic information through the word embeddings nor a smart way to handle out of vocabulary words.

All the systems marked with * require that the input is a dependency tree, which means that they solve a transduction task between a dependency tree and an AMR graph. Even though our parser starts from plain text sentences when we incorporate more information into our model, we achieve further improvements. POS tags provide small improvements (0.6801 without POS tags vs 0.6822 for the model that runs with POS tags). Dependency trees help a bit more achieving 0.6920.

5 Conclusions and Future Work

We present a new transition-based algorithm for AMR parsing and we implement it using Stack-LSTMS and a greedy decoder. We present competitive results, without any additional resources and external tools. Just by looking at the words, we achieve 0.68 F1 (and 0.69 by preprocessing dependency trees) in the standard dataset used for evaluation.

Acknowledgments

We thank Marco Damonte, Shay Cohen and Giorgio Satta for running and evaluating their parser in the LDC2014T12 dataset. We also thank Chuan Wang for useful discussions.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal. Association for Computational Linguistics.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal. Association for Computational Linguistics.

Miguel Ballesteros and Joakim Nivre. 2013. Going to the roots of dependency parsing. *Computational Linguistics*, 39(1).

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *ACL (2)*, pages 748–752.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2016. An incremental parser for abstract meaning representation. *CoRR*, abs/1608.06111.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah Smith. 2016. Recurrent neural network grammars. In *Proceedings of NAACL-HLT 2016*.

Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime Carbonell. 2016. Cmu at semeval-2016 task 8: Graph-based amr parsing with infinite ramp loss. *Proceedings of SemEval*, pages 1202–1206.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1426–1436. Association for Computational Linguistics.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for abstract meaning representation parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–11, Berlin, Germany. Association for Computational Linguistics.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 1–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4):949–998.

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

Guillaume Lample, Miguel Ballesteros, Kazuya Kawakami, Sandeep Subramanian, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT 2016*.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015a. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015b. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Dipendra Kumar Misra and Yoav Artzi. 2016. Neural shift-reduce ccg semantic parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1775–1786, Austin, Texas. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Joakim Nivre. 2003. An Efficient Algorithm for Projective Dependency Parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34:4:513–553. MIT Press.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 351–359. Association for Computational Linguistics.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Parsing english into abstract meaning representation using syntax-based machine translation. In *Proc. EMNLP*, Lisbon, Portugal.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack lstms. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 187–197.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings NAACL*.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based AMR parsing with refined actions and auxiliary analyzers. In *Proc. of , ACL 2015*, pages 857–862.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for amr parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375, Denver, Colorado. Association for Computational Linguistics.

Keenon Werling, Gabor Angeli, and Christopher D. Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 982–991, Beijing, China. Association for Computational Linguistics.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.

Junsheng Zhou, Feiyu Xu, Hans Uszkoreit, Weiguang QU, Ran Li, and Yanhui Gu. 2016. Amr parsing with an incremental joint model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 680–689, Austin, Texas. Association for Computational Linguistics.

An End-to-End Deep Framework for Answer Triggering with a Novel Group-Level Objective

Jie Zhao
The Ohio State University
zhao.1359@osu.edu

Yu Su
University of California, Santa Babara
ysu@cs.ucsb.edu

Ziyu Guan
Northwest University, China
ziyuguan@nwu.edu.cn

Huan Sun
The Ohio State University
sun.397@osu.edu

Abstract

Given a question and a set of answer candidates, answer triggering determines whether the candidate set contains any correct answers. If yes, it then outputs a correct one. In contrast to existing pipeline methods which first consider *individual* candidate answers separately and then make a prediction based on a threshold, we propose an end-to-end deep neural network framework, which is trained by a novel *group-level* objective function that directly optimizes the answer triggering performance. Our objective function penalizes three potential types of error and allows training the framework in an end-to-end manner. Experimental results on the WikiQA benchmark show that our framework outperforms the state of the arts by a 6.6% absolute gain under F_1 measure[1].

1 Introduction

Question Answering (QA) aims at automatically responding to natural language questions with direct answers (Heilman and Smith, 2010; Severyn and Moschitti, 2013; Yao et al., 2013; Berant and Liang, 2014; Yih et al., 2015; Sun et al., 2015; Miller et al., 2016; Sun et al., 2016). Most existing QA systems always output an answer for any question, no matter whether their answer candidate set contains correct answers or not (Feng et al., 2015; Severyn and Moschitti, 2015; Yang et al., 2016; Rao et al., 2016). In practice, however, this can greatly hurt user experience, especially when it is hard for users to judge answer correctness. In this paper, we study the critical yet under-addressed *Answer Triggering* (Yang et al., 2015) problem: Given a question and a set of answer candidates, determine whether the candidate set contains any correct answer, and if so, select a correct answer as system output.

The answer triggering problem can be logically divided into two sub-problems: $\mathbf{P}_1$: Build an *individual-level* model to rank answer candidates so that a correct one (if it exists) gets the highest score. $\mathbf{P}_2$: Make a *group-level* binary prediction on the existence of correct answers within the candidate set. Previous work (Yang et al., 2015; Jurczyk et al., 2016) attack the problem via a pipeline approach: First solve P_1 as a ranking task and then solve P_2 by choosing an optimal threshold upon the previous step's highest ranking score. However, the yielded answer triggering performance is far from satisfactory, with F_1 between 32% and 36%. An alternative pipeline approach is to first solve P_2 and then P_1, i.e., first determine whether there's a correct answer in the candidate set and then rank all candidates to find a correct one. However, as we will show using state-of-the-art Multiple Instance Learning (MIL) algorithms in Section 4, P_2 by itself is currently a very challenging task, partly because of the difficulty of extracting features from a set of candidate answers that are effective for answer triggering. Because both P_1 and P_2 performances are far from perfect, the above pipeline approaches also suffer from error propagation (Finkel et al., 2006; Zeng et al., 2015).

We propose Group-level Answer Triggering (GAT), an end-to-end framework for jointly optimizing P_1 and P_2. Our key contribution in GAT is a novel group-level objective function, which aggregates individual-level information and penalizes three potential error types in answer triggering as a group-level task. By optimizing this objective function, we can directly back-propagate the

[1] Our code is available at https://github.com/jiez-osu/answer-triggering.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1276–1282
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

final answer triggering errors to the entire framework and learn all the parameters simultaneously. We conduct evaluation using the same dataset and measure as in previous work (Yang et al., 2015; Jurczyk et al., 2016), and our framework improves the F_1 score by **6.6%** (from 36.65% to 43.27%), compared with the state of the art.

2 Framework

Notations. Let i and j respectively be the index of question and answer candidate, $l_{i,j}$ be the binary label of the j-th answer candidate for question q_i, and l_i be the group label of the answer candidate set of q_i (1 if it contains any correct answer; 0 otherwise). $m_{i,j}$ denotes an individual-level matching score, measuring how likely question q_i can be correctly addressed by its j-th answer candidate.

The GAT framework is illustrated in Figure 1, which consists of three components: (1) ***Encoder***. Two separate encoders process questions and answer candidates respectively, mapping them from token sequences into two different vector spaces. (2) ***QA Matching***. For each question and answer candidate pair, we concatenate their encoded vectors, and pass it through a feed forward neural network with a binary softmax output layer. The output is an *individual-level* matching score, i.e., $m_{i,j}$. (3) ***Signed Max Pooling***. Max pooling is applied on all the matching scores in a candidate set. During training when each candidate is positively/negatively labeled on whether they can answer the question or not, we use the labels to divide the scores into two disjoint subsets and perform max pooling separately:

$$m_i^+ = \max_{j:l_{i,j}=1} m_{i,j}, \qquad m_i^- = \max_{j:l_{i,j}=0} m_{i,j},$$

where m_i^+ is the maximum score among correct answers (if there's any) and m_i^- is that among wrong ones. At testing time when labels are unavailable, it reduces to normal max pooling and pools a single score $m_i = \max_j m_{i,j}$. The answer triggering prediction is then made by comparing m_i with a predefined threshold (0.5) to decide whether to return the top-scored answer candidate to the user.

The GAT framework design is generic in that the Encoder component can be instantiated with different network architectures. In this paper, we implement it with Bidirectional RNNs (Bi-RNN) (Schuster and Paliwal, 1997) with GRU cells (Cho et al., 2014), and use the temporal average pooling over the hidden states as the encoding representation. We choose Bi-RNN mainly because of its good performance in many QA problems (Wang and Nyberg, 2015; Wang et al., 2016).

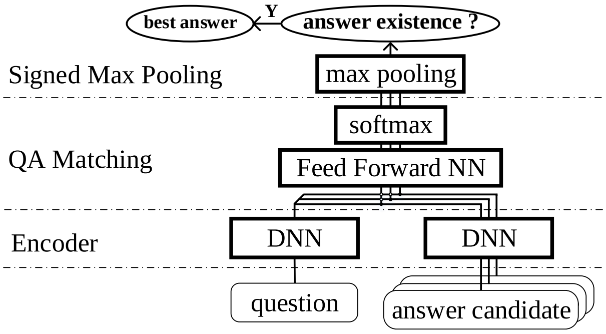


Figure 1: GAT: An end-to-end deep framework to be trained with a novel group-level objective function. Rounded rectangles at the bottom represent input data.

2.1 Learning

The cost function for negative groups (answer candidate sets without correct answers) and positive groups (those with correct answers) are treated differently. For each negative group, the highest QA matching score is penalized by a hinge loss:

$$O_1 = \frac{1}{N_{\text{neg}}} \sum_{i:\, l_i=0} \max(0, d^- - (0.5 - m_i^-)),$$

where the maximum matching score m_i^- is compared with 0.5, a fixed threshold for our framework. The variable d^- here, as well as d^+ and $d^\pm$ that will appear shortly after, are all margin hyperparameters. O_1 is normalized by N_{neg}, which is the number of negative groups (with $l_i = 0$). We use O_1 to reduce false-positive answer existence predictions by penalizing the top matching score that is not safely below the 0.5 threshold.

For a positive group, it is more complicated because answer triggering prediction can have the following two error types: (1) the top matching score is below the threshold, or (2) the top ranked answer candidate is a wrong answer. We design loss terms O_2 and O_3 to penalize these two types of error, respectively. O_2 is a hinge loss that penalizes the case where the highest score among the correct answers in a group is not large enough to signify answer existence. O_3 is to penalize the case where the highest score is obtained by an incorrect candidate answer. Formally:

$$O_2 = \frac{1}{N_{\text{pos}}} \sum_{i:\, l_i=1} \max(0, d^+ - (m_i^+ - 0.5))$$

$$O_3 = \frac{1}{N_{\text{pos}}} \sum_{i:\, l_i=1} \max(0, d^\pm - (m_i^+ - m_i^-))$$

Finally, the overall objective function in Equation 1 is a linear combination of the three loss terms and a standard ℓ_2-regularization. Θ denotes all the trainable parameters in the framework. α, β and λ are hyper-parameters.

$$\mathcal{O} = O_1 + \alpha O_2 + \beta O_3 + \lambda\|\Theta\|^2 \qquad (1)$$

2.2 A Naive Objective Baseline

For comparison, we provide an alternative objective formulation, which equivalently treats positive and negative groups, and does not explicitly penalize cases where an incorrect candidate answer obtains the highest QA matching score in a positive group.

$$\begin{aligned} O_2^* &= \frac{1}{N_{\text{pos}}} \sum_{i:\, l_i=1} \max(0, d^+ - (m_i - 0.5)) \\ O_1^* &= O_1; \quad \mathcal{O}^* = O_1^* + \alpha^* O_2^* + \lambda^* \|\Theta\|^2 \end{aligned} \qquad (2)$$

Here d^+ is a margin and α^*, λ^* are weights. We hypothesize this formulation will work worse than the objective in Equation 1, and will use experiments to verify it.

3 Experiments

3.1 Dataset

We use the WikiQA dataset (Yang et al., 2015) for evaluation. It contains 3,047 questions from Bing query logs, each associated with a group of candidate answer sentences from Wikipedia and manually labeled via crowdsourcing. Several intuitive features are also included in WikiQA: two word matching features (IDF-weighted and unweighted word-overlapping counts between questions and candidate answers, denoted as `Cnt`), the length of a question (`QLen`), and the length of a candidate answer (`SLen`). As in previous works, we also test the effect of these features, by combining them with other features as input into the Softmax layer in our framework. We use the standard 70% (train), 10% (dev), and 20% (test) split of WikiQA. We also use the same data pre-processing steps for fair comparison: Truncate questions and sentences to a maximum of 40-token long and initialize the 300-dimensional word vectors using pretrained word2vec embedding (Mikolov et al., 2013).

3.2 Implementation Details

We implement our full framework using TensorFlow (Abadi et al., 2016) and train it using the AdaDelta optimizer (Zeiler, 2012) with learning rate 0.1 and decay factor 0.95. Dropout is used during training to prevent overfitting. The default threshold in Signed Max Pooling is set at 0.5. We select the hyper-parameters using the dev set and set α=1.2, β=1.0, d^+=0.2, d^-=0.3, $d^\pm$=0.5, λ=$1e^{-4}$. The RNN's hidden state size is 200 in both directions. The feed-forward network in *QA Matching* has two layers of 400 hidden units.

3.3 Evaluation Metrics

We use precision, recall, and F_1, defined in the same way as in previous work. A question is treated as a positive case only if it contains one or more correct answers in its candidate set. For the prediction of a question, only the candidate with the highest matching score is considered. A true positive prediction shall meet two criteria: (1) the score is above a threshold (0.5 for our framework; tuned on dev set in other work), and (2) the candidate is labeled as a correct answer to the question.

3.4 Results

a. Comparison with Baselines

We evaluate the effectiveness of the proposed GAT framework by comparing with several baseline models. To the best of our knowledge, there has only been limited work so far on answer triggering, and they are the first two baselines below. (1) Yang et al. (2015) propose CNN-Cnt, which is a combination of the CNN model from Yu et al. (2014) and two `Cnt` features. We use their best reported result which is achieved when CNN-Cnt is combined with `QLen` features. (2) Jurczyk et al. (2016) extend the previous work with various network structures and add some more sophisticated features. Here we compare with their best model on WikiQA, which is a CNN model combined with carefully designed tree-matching features, extracted from expensive dependency parsing results. (3) We include a third **Naive** baseline where the objective function in Equation 2 is used to train our architecture in Figure 1. Due to space limits, we show its best result obtained among various feature combinations.

The results are summarized in Table 1.

We can see that GAT combined with `Cnt` features improves the F_1 score from Yang et al. (2015) and Jurczyk et al. (2016) by around **11.1%** and **6.6%** (from 32.17 and 36.65 to 43.27), which shows the effectiveness of our framework. We denote this configuration as our *full* framework. Through the comparison between Naive and GAT,

Model	Prec	Rec	F1
(Yang et al., 2015)	27.96	37.86	32.17
(Jurczyk et al., 2016)	29.43	48.56	36.65
Naive +`Cnt`	27.36	48.84	35.07
GAT	32.70	48.59	39.09
GAT +`Cnt` (Full)	33.54	60.92	**43.27**
GAT +`Cnt`+`QLen`	33.12	59.09	42.45
GAT +`Cnt`+`SLen`	28.03	64.60	39.10
GAT +`All`	31.35	58.82	40.90

Table 1: Results on the test set.

we can see that our proposed objective function has a great advantage over the Naive one which does not model the complexity of answer triggering for positive candidate sets. Different from Yang et al. (2015)'s results, combining with the `QLen` feature does not further improve the performance in our case, possibly because we choose Bi-RNN as our encoder, which may capture some question characteristics better than a length feature.

b. Framework Breakdown

Now we conduct further analysis in order to better understand the contribution of each component in our full framework. Since the code from (Yang et al., 2015) is available, we use it (rather than (Jurczyk et al., 2016)) to assist our analysis.

We first test a variant of our full framework by replacing the Encoder and QA Matching component with the CNN based model from (Yang et al., 2015)[2], denoted as **GAT** w/ CNN, and train it with our objective. From the first two rows in Table 2, we observe that: (1) Using our current design Bi-RNN and feed-foward NN improves from 35.03% to 43.27%, in comparison with the CNN based model, partly because their CNN only consists of one convolution layer and one average pooling layer. However, we leave more advanced encoder and QA matching design for future work, and anticipate that more complex CNN based models can achieve similar or better results than our current design, as in many other QA-related work (Hu et al., 2014; He and Lin, 2016). (2) Compared with the best result from (Yang et al., 2015) in Table 1, training the CNN based model end-to-end using our objective improves from 32.17% to 35.03%. *This directly shows an end-to-end learning strategy works better than the pipeline approach in (Yang et al., 2015).*

Now we detach the Encoder component **ENC** from our end-to-end full framework. To obtain semantic vectors of questions and candidate answers as input to the subsequent QA Matching component, we leverage Yang et al.(2015)'s released code to train the Encoder component (with CNN) through their well-tuned individual-level optimization, and use their learnt semantic vectors. Then our framework without ENC, i.e.,
-ENC, is trained and tested as before. We further detach the QA matching component **QAM** in a similar way: We directly use the matching score between a question and a candidate answer obtained by Yang et al. (2015), and concatenate it with `Cnt` features as input to the Softmax layer, which is our framework without ENC or QAM, denoted as **-ENC -QAM**, and trained by our group-level objective. By comparing them with our end-to-end frameworks on both dev and test sets, we can see that it is beneficial to jointly train the entire framework.

	Framework	F_1 score	
		dev	test
End-to-End	Full	44.63	**43.27**
	GAT w/ CNN	39.67	35.03
Pipeline	**-ENC**	39.13	33.42
	-ENC -QAM	38.69	33.20

Table 2: GAT framework breakdown. All variants are trained with our proposed objective function (Equation 1).

3.5 Error Analysis

We now demonstrate some typical mistake types made by our framework to inspire future improvements.

Q: *What city was the convention when Gerald Ford was nominated?*
A: *Held in Kemper arena in Kansas City , Missouri , the convention nominated president Gerald Ford for a full term, but only after narrowly defeating a strong challenge from former California governor Ronald Reagan.*

In this case, **A** is correct, but our framework made a false negative prediction. Although already being the highest ranked in a set of 4 candidate answers, **A** only got a score of 0.134, possibly due to its complicated semantic structure (attribute clause) and the extra irrelevant information (defeating Reagan).

Q: *What can SQL 2005 do?*
A1: *Microsoft SQL server is a relational database management system developed by Microsoft.*

[2] Where the QA matching score is obtained first through CNN encoding and then a bilinear model.

A2: *As a database , it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a [TRUNCATED END]*

The incorrect answer **A1** is ranked higher than the correct answer **A2**, both with scores above 0.5. This is a false positive case, with incorrect ranking as well. Possible reasons are that the detailed functionality of SQL explained in **A2** is hard to be captured and related to the question, and **A2** gets truncated to 40 tokens long in our experiments. On the other hand, the "database management system" phrase in **A1** sounds close to an explanation of functionality, if not carefully distinguished.

Both cases above show that the semantic relation between a question and its answer is hard to capture. For future research, more advanced models can be incorporated in the *Encoder* and *QA Matching* components of our framework.

4 Related Work

Answer Selection. Answer selection (a.k.a., answer sentence selection) is the task of assigning answer candidates with individual-level ranking scores given a question, which is similar to P1 defined in Section 1. Existing QA systems based on answer selection just select the top-scored candidate as answer, without considering the possibility that the true answer doesn't even exist. However, many neural network models recently explored in the answer existence literature (Hu et al., 2014; Wang and Nyberg, 2015; Feng et al., 2015) could be utilized for answer selection as well in the future. For example, Tan et al. (2016) explore the respective advantages of different network architectures such as Long Short-Term Memory Networks (LSTMs) and CNNs. They also develop hybrid models for answer selection. Various attention mechanisms have been proposed such as (Wang et al., 2016) for RNNs and (Yin et al., 2015; dos Santos et al., 2016) for CNNs. Answer selection is also formulated as a sentence similarity measurement problem (He and Lin, 2016; He et al., 2015) or a pairwise ranking problem as in (Severyn and Moschitti, 2015; Yang et al., 2016; Rao et al., 2016).

Multiple Instance Learning We have briefly mentioned MIL (Babenko et al., 2011; Amores, 2013; Cheplygina et al., 2015) in Section 1. Many MIL algorithms can not be directly applied for answer triggering, because individual-level annotations and predictions are often assumed unavailable and unnecessary in MIL (Maron and Lozano-Pérez, 1998; Babenko et al., 2011; Amores, 2013; Cheplygina et al., 2015), but not in the answer triggering setting, where the correctness of each answer candidate is annotated during training and needs to be predicted during testing. We experimented with two popular MIL algorithms that explicitly discriminate individual-level labels: MI-SVM(Andrews et al., 2003) and SbMIL (Bunescu and Mooney, 2007) implemented in one of the state-of-the-art MIL toolkits (Doran and Ray, 2014), where we represented each question/answer with encoder vectors as in Section 3.4. Unfortunately, both algorithms predict no correct answer exists for any question, possibly because the training data are biased towards negative groups and the input features are not effective enough. This indicates that using MIL for answer triggering is challenging and still open for future research.

5 Conclusion

In conclusion, we address the critical answer triggering challenge with an effective framework based on deep neural networks. We propose a novel objective function to optimize the entire framework end-to-end, where we focus more on the group-level prediction and take into account multiple important factors. In particular, the objective function explicitly penalizes three potential errors in answer triggering: (1) false-positive and (2) false-negative predictions of the existence of a correct answer, as well as (3) ranking incorrect answers higher than correct ones. We experimented with different objective function settings and show that our GAT framework outperforms the previous state of the arts by a remarkable margin.

Acknowledgments

We would like to thank the anonymous reviewers for valuable comments. The computing resources in this work are supported by Ohio Supercomputer Center (Center, 1987) and the National Science Foundation under Grant No. CNS-1513120. The third author is supported by the National Natural Science Foundation of China (Grant Nos. 61522206, 61672409, 61373118), the Major Basic Research Project of Shaanxi Province (Grant No. 2017ZDJC-31) and the Science and Technology Plan Program in Shaanxi Province of China (Grant No. 2017KJXX-80).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Jaume Amores. 2013. Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, 201:81–105.

Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. 2003. Support vector machines for multiple-instance learning. *Advances in neural information processing systems*, pages 577–584.

Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. 2011. Robust object tracking with online multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1619–1632.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *ACL*, pages 1415–1425.

Razvan C Bunescu and Raymond J Mooney. 2007. Multiple instance learning for sparse positive bags. In *Proceedings of the 24th international conference on Machine learning*, pages 105–112. ACM.

Ohio Supercomputer Center. 1987. Ohio supercomputer center. `http://osc.edu/ark:/19495/f5s1ph73`.

Veronika Cheplygina, David MJ Tax, and Marco Loog. 2015. Multiple instance learning with bag dissimilarities. *Pattern Recognition*, 48(1):264–275.

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

Gary Doran and Soumya Ray. 2014. A theoretical and empirical analysis of support vector machine methods for multiple-instance classification. *Machine Learning*, 97(1-2):79–102.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 813–820. IEEE.

Jenny Rose Finkel, Christopher D Manning, and Andrew Y Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626. Association for Computational Linguistics.

Hua He, Kevin Gimpel, and Jimmy J Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*, pages 1576–1586.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of NAACL-HLT*, pages 937–948.

Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019. Association for Computational Linguistics.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050.

Tomasz Jurczyk, Michael Zhai, and Jinho D Choi. 2016. Selqa: A new benchmark for selection-based question answering. *arXiv preprint arXiv:1606.08513*.

Oded Maron and Tomás Lozano-Pérez. 1998. A framework for multiple-instance learning. *Advances in neural information processing systems*, pages 570–576.

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

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1913–1916. ACM.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *EMNLP*, volume 13, pages 458–467.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382. ACM.

Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. 2016. Table cell search for question answering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 771–782. International World Wide Web Conferences Steering Committee.

Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. 2015. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1045–1055. ACM.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Bingning Wang, Kang Liu, and Jun Zhao. 2016. Inner attention based recurrent neural networks for answer selection. In *The Annual Meeting of the Association for Computational Linguistics*.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *ACL*, pages 707–712.

Liu Yang, Qingyao Ai, Jiafeng Guo, and W Bruce Croft. 2016. anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 287–296. ACM.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of EMNLP*, pages 2013–2018. Citeseer.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *HLT-NAACL*, pages 858–867. Citeseer.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*, pages 1753–1762.

Predicting Word Association Strengths

Andrew Cattle **Xiaojuan Ma**
Hong Kong University of Science and Technology
Department of Computer Science and Engineering
Clear Water Bay, Hong Kong
{acattle,mxj}@cse.ust.hk

Abstract

This paper looks at the task of predicting word association strengths across three datasets; WordNet Evocation (Boyd-Graber et al., 2006), University of Southern Florida Free Association norms (Nelson et al., 2004), and Edinburgh Associative Thesaurus (Kiss et al., 1973). We achieve results of $r = 0.357$ and $\rho = 0.379$, $r = 0.344$ and $\rho = 0.300$, an $\rho = 0.292$ and $\rho = 0.363$, respectively. We find Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) cosine similarities, as well as vector offsets, to be the highest performing features. Furthermore, we examine the usefulness of Gaussian embeddings (Vilnis and McCallum, 2014) for predicting word association strength, the first work to do so.

1 Introduction

Word embeddings such as Word2Vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014) have received increasing attention in the world of natural language processing and computational linguistics. Under such embeddings, the semantic relatedness of two words is generally taken to be the cosine similarity of their word vectors. Although this approach performs well for variety of applications, it is not without its limitations. First, it defines "relatedness" quite narrowly as the extent to which the two words appear in similar contexts. Second, it fails to capture how humans internally represent words (De Deyne et al., 2016b).

Word associations offer a more flexible view of semantic relatedness by leveraging "lexical knowledge acquired through world experience" (Nelson et al., 2004). While word embeddings capture distributional relationships, word associations are able to capture more nuanced relationships "which are based on human perception and experiences [and] are not reflected in common language usage." (Ma, 2013) For example, "yellow" is so closely associated with "banana" that many people would only specify a banana's colour if it is not yellow. This is backed up by De Deyne et al. (2016b) which found word associations performed better than word embeddings across a variety of semantic relatedness tasks.

Furthermore, word associations, unlike cosine similarities, are asymmetric; when presented with the word "beer", many people think of the word "glass" but when presented with the word "glass", few people think of the word "beer" (Ma, 2013). This directionality allows for more fine-grained exploration of semantic links, with applications in word similarity (Jabeen et al., 2013) and computational humour (Cattle and Ma, 2016).

Although several word association datasets exist, such as the Edinburgh Associative Thesaurus (EAT, Kiss et al., 1973), the University of South Florida Free Association Norms (USF, Nelson et al., 2004), or WordNet Evocation (Evocation, Boyd-Graber et al., 2006), their reliance on human annotations mean they all suffer from coverage issues relating to limited vocabularies or sparse connectivity (Cattle and Ma, 2016; De Deyne et al., 2016b). Although these issues would be somewhat alleviated by the creation of larger datasets, collecting human judgments for all possible word pairs is impractical. Therefore, the ability to predict association strengths between arbitrary word pairs represents the best solution to these coverage issues (Boyd-Graber et al., 2006).

Although the prediction of Evocation ratings has attracted some attention (Boyd-Graber et al., 2006; Hayashi, 2016), to the best of our knowledge this is the first work to focus on the prediction of USF or EAT strengths. As described in Sec-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1283–1288
Copenhagen, Denmark, September 7–11, 2017.

tion 2, USF and EAT have several advantages over Evocation, such as the ability to work with ambiguous words instead of WordNet synsets. Following Hayashi (2016)'s work on Evocation prediction, we frame word association prediction as a supervised regression task and introduce several new and modified features, including the first use of Gaussian embeddings (Vilnis and McCallum, 2014) to better capture the asymmetric nature of word associations.

2 Previous Work

Word association has been used in psychological and psycholinguistic experiments for well over 100 years (Boyd-Graber et al., 2006; De Deyne and Storms, 2008). Word association datasets such as USF or EAT have typically framed word association as "a task that requires participants to produce the first word to come to mind that is related in a specified way to a presented cue" (Nelson et al., 2000). These datasets use *forward strengths*, the proportion of participants who produce a specific response, to "index the relative accessibility of related words in memory [for a given cue]" (Nelson et al., 2004).

This cue/response framework has several drawbacks. First, since forward strengths are relative, comparing strengths across different cue words is difficult. Second, both cues and responses are ambiguous, with each participant's responses being greatly influenced by how they chose to interpret a given cue. For example, someone responding to the cue "brother" with "monk" is considering a different sense of "brother" than someone who responds "sister" (Ma, 2013). As such, forward strengths are biased toward responses which presume more readily apparent cue word senses. Third, limiting participants to a single response can lead to weaker associations being underreported or omitted entirely.

Evocation solves the ambiguity issue by focusing on associations between WordNet synsets. Boyd-Graber et al. (2006) presented participants with randomly selected synset pairs and asked them to score how much the first synset *evoked* (i.e. brought to mind) the second. Unlike forward strengths, these Evocation ratings are absolute, meaning they can be directly compared across different cues. While randomly selecting synset pairs ensured that weaker associations would not be underreported, it did have the disadvantage that 67% of pairs were unanimously rated as having no connection (Boyd-Graber et al., 2006).

Despite attempts to address this spareness issue by expanding Evocation with data gathered from Amazon Mechanical Turk[1] (Nikolova et al., 2009) or word-sense disambiguated USF cue/response pairs (Ma, 2013), obtaining human judgments for all possible synset pairs is impractical. As such, the prediction of Evocation ratings presents the most promising solution to this coverage issue. Boyd-Graber et al. (2006) detailed a simple Evocation estimator which used a combination of WordNet structure-based features, WordNet definition-based features, and corpus-based word co-occurrence features. However, this approach is somewhat limited in that it frames Evocation prediction as a classification task, considering only five Evocation levels.

The main drawback of Evocation prediction as a classification task is that it is too coarse-grained to deal with very weak associations, such as those in remote triads (De Deyne et al., 2016a), or very slight variations in association strength, such as those useful for computational humour (Cattle and Ma, 2016). To this end, Hayashi (2016) framed Evocation prediction as a supervised regression task. They employed a combination of WordNet structure-based features, word embedding-based features, and lexical features and found that vector offsets, i.e. the mathematical difference between vectors, were a strong indicator of Evocation ratings.

While Evocation's use of unambiguous synsets is useful for many applications, it is not without its own drawbacks. First, it requires texts to be word sense disambiguated; a non-trivial task. Second, since humans do not conceptualize words as a discrete set of independent word senses, Evocation is unable to capture natural associations owing to homography, homophony, or polysemy (Ma, 2013). As such, despite their drawbacks, word associations may provide a more flexible, more holistic view of mental semantics.

By allowing participants to record more than one response, De Deyne and Storms (2008), and their derivative works De Deyne et al. (2013) and De Deyne et al. (2016b), were able to better represent weaker associations. However, this introduced its own set of problems as great care had to be taken to avoid chaining, i.e. responding to a

[1] https://mturk.com/

previous response instead of the cue, and retrieval inhibition. De Deyne and Storms (2008) frames word association collection as a continuous task, meaning not only that the vocabulary is ever growing but also that changes in associations over time can be observed and tracked. But despite the steps taken to improve the size and quality of their association dataset, practicality dictates that coverage issues cannot be completely eliminated.

3 System Definition

Our word association prediction system extends the method in Hayashi (2016) with several modifications to make it better suited to the USF and EAT datasets.

First, we modify Hayashi (2016)'s *lexVector*. Hayashi (2016) represent each word's part-of-speech (POS) using a one-hot encoded five dimensional vector (one of each POS in WordNet). Similarly, they represent each word's lexical category using a one-hot encoded 45 dimensional vector (one for each WordNet lexicographer file). This results in a 100 dimensional vector representing the POS and lexical categories of both the cue and the response. Since words in USF and EAT can be associated with multiple synsets and we want to be able to capture associations related to polysemy, instead using a one-hot encoding we employ count vectors specifying the number of synsets from each POS/lexical category each word belongs to.

Second, instead of computing Wu-Palmer similarity (WUP, Wu and Palmer, 1994) between a single synset pair, we compute it for all cue synset/response synset pairs and record the maximum and average values. Following Boyd-Graber et al. (2006) and Ma (2013), we also explored the use of path and Leacock-Chodorow (Leacock and Chodorow, 1998) similarities but found they did not add any advantage over WUP alone. We take a similar approach for adapting load and betweenness centralities (Barthelemy, 2004) as well as AutoExtend (AutoEx, Rothe and Schütze, 2015) similarity.

Third, we extend the notion of *dirRel*, introduced in Hayashi (2016) to leverage the semantic network structure of WordNet. Given a graph where nodes represent synsets and arcs represent WordNet relations such as hypernym/hyponym and holonym/meronym, *dirRel(s,t,k)* is the proportion of k-step neighbours of s that are also k-step neighbours of t. In the original formula, s and t are nodes representing a single synset. We instead consider a set of nodes S and a set of nodes T representing the set of synsets associated with the cue and response words, respectively, as shown in Equation 1. This may increase the probability that $|nb(S,k) \cap nb(T,k)| > 0$, a shortcoming of the original *dirRel* due to WordNet's "relatively sparse connective structure" (Hayashi, 2016).

$$dirRel(S,T,k) = \frac{|nb(S,k) \cap nb(T,k)|}{|nb(S,k)|} \quad (1)$$

Fourth, in addition to the Word2Vec (w2v) cosine similarity between cue/response pairs calculated using Google's pre-trained 300 dimension Word2Vec embeddings[2]. We also examine the effectiveness of Stanford's pre-trained 300 dimension GloVe embeddings[3].

Fifth, in order to better capture asymmetric word associations, we propose using Gaussian embeddings. Gaussian embeddings (Vilnis and McCallum, 2014) represent words not as a fixed point in vector space but as "potential functions", continuous densities in latent space; therefore, they are more suitable for capturing asymmetric relationships. More specifically, for each cue/response pair, we calculate both the KL-divergence and cosine similarities of their Gaussian embeddings. The embeddings have a dimensionality of 300 and are trained on English Wikipedia using the Word2Gauss[4] (w2g) and the hyperparameters reported by the developer[5]

Sixth, since cue and response words are not associated with a single synset, the AutoEx embeddings employed in Hayashi (2016) to compute vector offsets are not well suited for our task. Instead, we experiment with offsets calculated using w2v, GloVe, and w2g embeddings.

Finally, our 300 topic LDA model (Blei et al., 2003) was trained using Gensim[6] on full English Wikipedia instead of the subset of English Wikipedia used in Hayashi (2016).

Using the above features, we trained a multilayer perceptron for each of our three datasets; Evocation, USF, and EAT. In the case of Evocation, we discarded any synset information and

[2] https://code.google.com/archive/p/word2vec/

[3] https://nlp.stanford.edu/projects/glove/

[4] https://github.com/seomoz/word2gauss

[5] https://github.com/seomoz/word2gauss/issues/18#issuecomment-286203006

[6] https://radimrehurek.com/gensim/

	Evocation		USF		EAT	
Feature	r	ρ	r	ρ	r	ρ
Hayashi (2016)	0.374	0.401	—	—	—	—
All (w/ w2v offsets)	0.357	0.379	0.344	0.300	0.292	0.363
betweenness (max)	−0.000	0.004	0.008	0.019	0.035	0.112
betweenness (avg)	−0.002	−0.001	0.012	0.004	0.002	0.021
load (max)	−0.009	−0.010	0.017	0.025	0.039	0.118
load (avg)	−0.007	−0.006	0.002	0.004	0.002	0.017
WUP sim (max)	0.098	0.136	0.092	0.111	0.049	−0.026
WUP sim (avg)	0.051	0.062	0.045	0.051	0.033	0.014
lexVector	0.115	0.117	0.091	0.077	0.105	0.249
dirRel	0.177	0.149	0.152	0.130	0.124	0.049
LDA cos sim	0.129	0.033	0.054	0.040	0.046	0.007
AutoEx cos sim (max)	0.135	0.144	0.124	0.132	0.054	−0.034
AutoEx cos sim (avg)	0.148	0.174	0.082	0.089	0.045	0.019
w2v cos sim	**0.265**	**0.264**	**0.229**	0.226	0.150	0.094
GloVe cos sim	0.239	0.262	0.222	**0.232**	0.117	−0.010
w2g cos sim	0.227	0.246	0.173	0.185	0.109	0.046
w2g KL-divergence	0.110	0.185	−0.013	−0.011	0.086	0.205
w2v offsets	0.010	0.009	0.092	0.076	0.144	0.299
GloVe offsets	0.007	0.009	0.127	0.098	**0.162**	**0.344**
w2g offsets	−0.005	−0.003	0.073	0.065	0.111	0.186

Table 1: Individual feature performance after 50 epochs

	Evocation		USF		EAT	
Feature	r	ρ	r	ρ	r	ρ
All (w/ w2v offsets)	0.357	0.379	0.344	0.300	0.292	0.363
- betweenness (max)	0.360	0.383	0.341	0.301	0.270	0.360
- betweenness (avg)	0.357	0.376	0.331	0.298	0.284	0.353
- load (max)	0.358	0.382	0.339	0.299	0.290	0.375
- load (avg)	0.360	0.381	0.340	0.304	0.279	0.353
- WUP sim (max)	0.367	0.376	0.333	0.294	0.283	0.367
- WUP sim (avg)	0.355	0.374	0.335	0.296	0.291	0.364
- lexVector	0.351	0.365	0.336	0.300	0.275	0.339
- dirRel	0.356	0.375	0.334	0.293	0.283	0.362
- LDA cos sim	0.357	0.377	0.340	0.299	0.291	0.361
- AutoEx cos sim (max)	0.362	0.382	0.347	0.299	0.280	0.358
- AutoEx cos sim (avg)	0.358	0.377	0.346	0.305	0.278	0.357
- w2v cos sim	0.352	0.377	0.331	0.294	0.280	0.345
- GloVe cos sim	0.352	0.367	0.332	0.292	0.284	0.360
- w2v and GloVe sims	**0.329**	**0.342**	**0.284**	**0.255**	0.261	0.353
- w2g cos sim	0.358	0.378	0.348	0.304	0.284	0.357
- w2g KL-divergence	0.351	0.356	0.344	0.299	0.286	0.348
- w2v offsets	0.361	0.386	0.303	0.280	**0.239**	**0.271**

Table 2: Ablation performance after 50 epochs

simply use each synset's headword (e.g. given the sysnet *entity.n.01*, we only considered the word *entity*). Following the setup used in Hayashi (2016), all neural networks are trained using the Chainer[7] Python library with rectified linear units, dropout, and two hidden layers, each with 50% of the number of units in the input layer. All were trained on 80% of their respective dataset, with 20% held out for testing. Mean squared error was used as a loss function and optimization was performed using Adam algorithm (Kingma and Ba, 2014). To act as a baseline, we also reimplemented the system described in Hayashi (2016) and trained it on the same 80/20 split of Evocation as our system. In addition to the reported results, we also performed feature selection experiments using 20% of the training sets as validation.

4 Results and Discussion

The performance of individual features are reported in Table 1 while the results of our ablation experiments are reported in Table 2. For all experiments we report both the Pearson correlation coefficient (as r) and Spearman's rank correlation coefficient (as ρ).

The best performing single feature on Evocation and USF is w2v cosine similarity. However, its removal in the ablation test had little effect. This is likely due to redundancy between w2v and GloVe; not only does GloVe perform similarly to w2v but removing both features at the same time produced the largest drop in performance. It is unclear why word embedding cosine similarities in general performed relatively poorly on EAT. While the USF and EAT datasets are very similar, EAT does seem to contain a greater number of multi-word cues/responses which would not be in the word embedding vocabularies. In such cases, perhaps a multi-word embedding like Doc2Vec (Le and Mikolov, 2014) would be more appropriate. However, if this were indeed the issue, one would expect vector offsets to perform equally poorly. This is not the case, with GloVe offsets being the best performing single feature on EAT and the removal of w2v offsets causing the greatest drop in performance in the EAT ablation tests.

The results of our Hayashi (2016) implementation are roughly comparable to those reported in the original paper ($r = 0.374$, $\rho = 0.401$ compared to $r = 0.439$, $\rho = 0.400$). Our slightly lower Pearson's R may be due to differences in way we split our training and test data as well as due to randomness in the training process itself.

On Evocation, our system does not perform as well as Hayashi (2016). This is expected as we explicitly ignore any synset information and instead attempt to predict association strengths between word-sense ambiguous words. Despite this, our performance is not appreciably lower, indicating the fitness of our system.

The fact that we perform better on Evocation than either USF or EAT is quite interesting considering our system was designed with USF and EAT in mind. There are several possible explanations for this. First, as mentioned in Section 2, 67% of cue/response pairs in Evocation have a strength of zero. This uniformity in Evocation strengths may make them easier to predict. Second, due to the way USF and EAT were collected, there are no true zeros in the datasets. This lack of grounding

[7] http://chainer.org/

may skew the predictions. Third, this may be an indication that predicting associations in a word-sense ambiguous context is a harder task than predicting them in a word-sense disambiguated one. Boyd-Graber et al. (2006) explicitly told annotators to ignore associations based on polysemy or rhyme. It could be the case that these effects are more difficult to identify.

Another possible explanation for this relatively lower performance is a lack of bespoke features. For example, we heavily rely on WordNet-based features which make sense in a word-sense disambiguated context but are less suited for ambiguous contexts. In fact, removal of several of these features, such as betweenness or AutoEx similarity, seem to slightly improve performance. One explanation is that, despite noting in Section2 that word association strengths are influenced by word-sense frequencies, our system instead implicitly assumes all synsets are equally likely.

The most surprising finding was the poor performance of Gaussian embeddings overall, and KL-divergence in particular. Given the asymmetric nature of word associations, KL-divergence seemed to be a natural fit. However, it is vastly outperformed by even cosine similarity on the same set of embeddings. Despite this, the usefulness of Gaussian embeddings cannot be ruled out. While we used pre-trained embeddings for Word2Vec and GloVe, we had to train our own Gaussian embedding model. Not only were Word2Vec and GloVe trained on much larger corpora than Gaussian embedding's English Wikipedia, but the pre-trained embeddings likely underwent a greater degree of hyperparameter tuning.

5 Conclusions and Future Works

In this paper we explored the effectiveness of various features for predicting Evocation, USF, and EAT association strengths, finding GloVe and Word2Vec cosine similarities as well as vector offsets to be the most useful features. We also examined the effectiveness of Gaussian embeddings for capturing the asymmetric nature of word embeddings but found it to be less effective than traditional word embeddings.

Although we report a lower performance than that in Hayashi (2016), potentially indicating that predicting association strengths in word-sense ambiguous contexts is a harder task, we believe our results are a promising start. Training Gaussian embeddings on a larger corpus may lead to improved effectiveness. Future works should also consider incorporating word-sense frequencies or developing word-sense agnostic features, with a particular focus on asymmetric features.

References

Marc Barthelemy. 2004. Betweenness centrality in large complex networks. *The European Physical Journal B-Condensed Matter and Complex Systems* 38(2):163–168.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Jordan Boyd-Graber, Christiane Fellbaum, Daniel Osherson, and Robert Schapire. 2006. Adding dense, weighted connections to wordnet."in: Proceedings of the third global wordnet meeting, jeju island, korea, january 2006 .

Andrew Cattle and Xiaojuan Ma. 2016. Effects of semantic relatedness between setups and punchlines in twitter hashtag games. *PEOPLES 2016* page 70.

Simon De Deyne, Daniel J Navarro, Amy Perfors, and Gert Storms. 2016a. Structure at every scale: A semantic network account of the similarities between unrelated concepts. *Journal of Experimental Psychology: General* 145(9):1228.

Simon De Deyne, Daniel J Navarro, and Gert Storms. 2013. Better explanations of lexical and semantic cognition using networks derived from continued rather than single-word associations. *Behavior Research Methods* 45(2):480–498.

Simon De Deyne, Amy Perfors, and J. Daniel Navarro. 2016b. Predicting human similarity judgments with distributional models: The value of word associations. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, pages 1861–1870. http://aclweb.org/anthology/C16-1175.

Simon De Deyne and Gert Storms. 2008. Word associations: Norms for 1,424 dutch words in a continuous task. *Behavior Research Methods* 40(1):198–205.

Yoshihiko Hayashi. 2016. Predicting the evocation relation between lexicalized concepts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 1657–1668. http://aclweb.org/anthology/C16-1156.

Shahida Jabeen, Xiaoying Gao, and Peter Andreae. 2013. Directional context helps: Guiding semantic relatedness computation by asymmetric word associations. In *WISE (1)*. pages 92–101.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

George R Kiss, Christine Armstrong, Robert Milroy, and James Piper. 1973. An associative thesaurus of english and its computer analysis. *The computer and literary studies* pages 153–165.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pages 1188–1196.

Claudia Leacock and Martin Chodorow. 1998. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database* 49(2):265–283.

Xiaojuan Ma. 2013. Evocation: analyzing and propagating a semantic link based on free word association. *Language resources and evaluation* 47(3):819–837.

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

Douglas L Nelson, Cathy L McEvoy, and Simon Dennis. 2000. What is free association and what does it measure? *Memory & cognition* 28(6):887–899.

Douglas L Nelson, Cathy L McEvoy, and Thomas A Schreiber. 2004. The university of south florida free association, rhyme, and word fragment norms. *Behavior Research Methods, Instruments, & Computers* 36(3):402–407.

Sonya Nikolova, Jordan Boyd-Graber, Christiane Fellbaum, and Perry Cook. 2009. Better vocabularies for assistive communication aids: connecting terms using semantic networks and untrained annotators. In *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*. ACM, pages 171–178.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the ACL*.

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

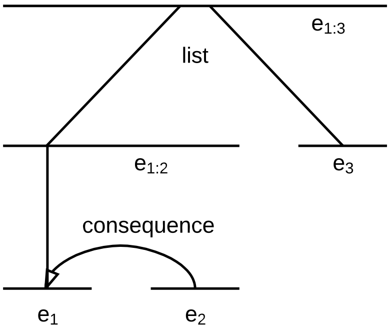
Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 133–138.

Learning Contextually Informed Representations for Linear-Time Discourse Parsing

Yang Liu and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
Yang.Liu2@ed.ac.uk,mlap@inf.ed.ac.uk

Abstract

Recent advances in RST discourse parsing have focused on two modeling paradigms: (a) high order parsers which jointly predict the tree structure of the discourse and the relations it encodes; or (b) linear-time parsers which are efficient but mostly based on local features. In this work, we propose a linear-time parser with a novel way of representing discourse constituents based on neural networks which takes into account global contextual information and is able to capture long-distance dependencies. Experimental results show that our parser obtains state-of-the art performance on benchmark datasets, while being efficient (with time complexity linear in the number of sentences in the document) and requiring minimal feature engineering.



[Only a few months ago, the 124-year-old securities firm seemed to be on the verge of a meltdown,]*e1* **[racked by internal squabbles and defections.]***e2* **[Its relationship with parent General Electric Co. had been frayed since a big Kidder insider-trading scandal two years ago.]***e3*

Figure 1: Example text (bottom) composed of two sentences (three EDUs) and its RST discourse tree representation (top).

1 Introduction

The computational treatment of discourse phenomena has recently attracted much attention, due to their increasing importance for potential applications. Knowing how text units can be composed into a coherent document and how they relate to each other e.g., whether they express contrast, cause, or elaboration, can usefully aid downstream tasks such summarization (Yoshida et al., 2014), question answering (Chai and Jin, 2004), and sentiment analysis (Somasundaran, 2010).

Rhetorical Structure Theory (RST, Mann and Thompson 1988), one of the most influential frameworks in discourse processing, represents texts by trees whose leaves correspond to Elementary Discourse Units (EDUs) and whose nodes specify how these and larger units (e.g., multi-sentence segments) are linked to each other by rhetorical relations. Discourse units are further characterized in terms of their importance in text: nuclei denote central segments, whereas satellites denote peripheral ones. Figure 1 shows an example of a discourse tree representing two sentences with three EDUs (e_1, e_2, and e_3). EDUs e_1 and e_2 are connected with a mononuclear relation (i.e., *Consequence*), where e_1 is the nucleus and e_2 the satellite (indicated by the left pointing arrow in the figure). Span $e_{1:2}$ is related to e_3 via *List*, a multi-nuclear relation, expressing the fact that both spans are equally important and therefore both nucleus.

Given such tree-based representations of discourse structure, it is not surprising that RST-style document analysis is often viewed as a parsing task. State-of-the-art performance on RST parsing is achieved by cubic-time parsers (Li, Li, and Hovy, 2014; Li, Li, and Chang, 2016), with $O(n^3)$ time complexity (where n denotes the number of sen-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1289–1298
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

tences in the document). These systems model the relations between all possible adjacent discourse segments and use a CKY-style algorithm to generate a global optimal tree. The high order complexity renders such parsers inefficient in practice, especially when processing large documents. As a result, more efficient linear-time discourse parsers have been proposed (Feng and Hirst, 2014; Ji and Eisenstein, 2014) which make local decisions and model the structure of the discourse and its relations separately. In this case, features are extracted from a local context (i.e., a small window of discourse constituents) without considering document-level information, which has been previously found useful in discourse analysis (Feng and Hirst, 2012).

In this paper, we propose a simple and efficient linear-time discourse parser with a novel way of learning contextual representations for discourse constituents. To guarantee linear-time complexity, we use a two-stage approach: we first parse each sentence in a document into a tree whose leaves correspond to EDUs, and then parse the document into a tree whose leaves correspond to already preprocessed sentences. The feature learning process for both stages is based on neural network models. At the sentence level, Long-Short Term Memory Networks (LSTMs; Hochreiter and Schmidhuber 1997) learn representations for EDUs and larger constituents, whereas at the document level, LSTMs learn representations for entire sentences. Treating a sentence as a sequence of EDUs and a document as a sequence of sentences allows to incorporate important contextual information on both levels capturing long-distance dependencies.

Recurrent neural networks excel at modeling sequences, but cannot capture hierarchical structure which is important when analyzing multi-sentential discourse. We therefore adopt a more structure-aware representation at the document level which we argue is complementary to the flat representations obtained from the LSTM. We represent documents as trees using recursive neural networks (Socher et al., 2012). Experimental evaluation on the RST Treebank shows that our parser yields comparable performance to previous linear-time systems, without requiring extensive manual feature engineering and improves upon related neural models (Li et al., 2014, 2016) on discourse relation classification, while being more efficient.

The rest of this paper is organized as follows. We overview related work in the following section. We describe the general flow of our parser in Section 3 and provide details on our parsing algorithm and feature learning method in Section 4. Experimental results are reported in Section 5. Section 7 concludes the paper.

2 Related Work

Recent advances in discourse modeling have greatly benefited from the availability of resources annotated with discourse-level information such as the RST Discourse Treebank (RST-DT; Carlson et al. 2003) and the Penn Discourse Treebank (PDTB, Prasad et al. 2008). In this work, we focus on RST-style discourse parsing, where a tree representation is derived for an entire document. In PDTB, discourse relations are annotated mostly between adjacent sentences and no global tree structure is provided.

Early approaches to discourse parsing (Marcu, 2000; LeThanh et al., 2004) have primarily focused on overt discourse markers (or cue words) and used a series of rules to derive the discourse tree structure. Soricut and Marcu (2003) employed a standard bottom-up chart parsing algorithm with syntactic and lexical features to conduct sentence-level parsing. Baldridge and Lascarides (2005) and Sagae (2009) used probabilistic head-driven parsing techniques. Subba and Di Eugenio (2009) were the first to incorporate rich compositional semantics into sentence- and document-level discourse parsing.

HILDA (Hernault et al., 2010) has been one of the most influential document-level discourse parsers paving the way for many machine learning-based models. HILDA parses a document pre-segmented into EDUs with two support vector machine classifiers working iteratively in a pipeline. At each iteration, a binary SVM predicts which adjacent units should be merged and then a multi-class SVM predicts their discourse relation. Subsequent work (Feng and Hirst, 2014; Joty et al., 2013) has shown that two-stage systems are not only efficient but can also achieve competitive performance. CKY-based parsers which guarantee globally optimal results have also been developed (Joty et al., 2013; Li et al., 2014).

Ji and Eisenstein (2014) were the first to apply neural network models to RST discourse parsing; their shift-reduce parser uses a feedforward neural network to learn the representations of the transition stack and queue. Li et al. (2014) proposed a CKY-based parser which uses recursive neural networks to learn representations for EDUs and their composition during the tree-building process. More recently, Li et al. (2016) designed a CKY-based parser which uses LSTMs to model text spans and a tensor-based transformation to compose adjacent spans.

Our own work joins others (Feng and Hirst, 2014; Joty et al., 2013) in adopting a two-stage architecture for our discourse parser. However, rather than considering each discourse constituent independently, we learn contextually-aware representations capturing long-range dependencies across sentences and documents. Discourse constituents at all levels are modeled with recurrent neural networks adopting a relatively simple, yet efficient, architecture compared to previously proposed neural systems (Li et al., 2016). Finally, we experimentally assess whether sequence-based representations are expressive enough by comparing them to those obtained (with recursive neural networks) from structured inputs.

3 Discourse Parser Overview

In RST discourse parsing, a document is first segmented into EDUs and a parser then builds a discourse tree with the EDUs as leaves. The first subtask is considered relatively easy with state-of-art accuracy at above 90% (Hernault et al., 2010). As a result, recent research focuses on the second subtask and often uses manual EDU segmentation.

Joty et al. (2013) found that a two-stage parsing strategy, which separates intra-sentential from multi-sentential parsing, has some advantages for document-level discourse parsing, since the distribution of discourse relations and useful features are different in the two stages. Based on their findings, our model also follows a two-stage approach and is composed by two components: an *intra-sentential parser* and a *multi-sentential parser*. Given a document pre-segmented into EDUs, our intra-sentential parser first builds a sentence-level discourse tree for individual sentences. Then, our multi-sentential parser creates a discourse tree for the entire document.

To guarantee linear-time complexity, both parsers adopt a greedy bottom-up tree-building process (Hernault et al., 2010) and are based on two conditional random field (CRF) models, one for creating discourse structure and another one for assigning relations. The intra-sentential parser considers adjacent EDUs and decides whether they should be connected (based on the scores predicted by the first CRF) and their relation (based on predictions of the second CRF). The multi-sentential parser follows the same procedure while operating over sentences.

The CRFs employ feature representations which we obtain using neural networks. Specifically, discourse constituents at all levels are modeled with recurrent neural networks (see Section 4.2 for details). In addition, for inter-sentential constituents, we complement the flat text span representation with recursive neural networks (see Section 4.4). We argue that the combination is advantageous; a sequential text span representation is in principle unsuitable for capturing hierarchical discourse structure, whereas a tree-based representation can be more precise, albeit less robust (due to the accumulation of errors from the recursive tree-building process).

4 Parsing Model

4.1 Intra-sentential Parser

To parse a sentence, we start with EDUs, which can be viewed as discourse constituents at the first level. As mentioned earlier, our intra-sentential parser is based on two linear-chain CRFs, the structure CRF decides which pair of constituents should be merged at the current level and the relation CRF assigns discourse relations to non-leaf constituents. For example, let $C_1 = \{e_1, e_2, \cdots, e_m\}$ denote a sentence with a sequence of EDUs, where e_i is the i^{th} EDU in the sentence. Suppose the structure CRF decides to merge together e_2 and e_3, then the next-level sequence is $C_2 = \{e_1, e_{2:3}, e_4, \cdots, e_m\}$ and the relation CRF will assign a discourse relation to $e_{2:3}$, the only non-leaf constituent so far. This process iterates until all EDUs are merged and a discourse subtree is generated for the entire sentence.

A linear-chain CRF for intra-sentential discourse parsing is shown in Figure 2. Here, $C = \{c_0, \cdots, c_t, \cdots, c_n\}$ are observed discourse constituents and $L = \{l_1, \cdots, l_t, \cdots, l_n\}$ are hidden structure nodes; label $l_t \in \{1, 0\}$ denotes whether constituents c_t and c_{t+1} should be connected. Our model differs from standard linear-chain CRFs in that the score between adjacent hidden nodes is not calculated based on a transition matrix, but is learned from observations instead. Specifically,

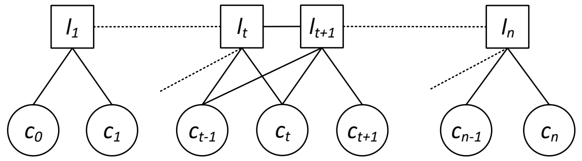


Figure 2: Intra-sentential structure CRF with pairwise modeling.

given a constituent sequence C, the probability of forming the structure sequence Y^{str} is written as:

$$p(Y^{str}|C) = \frac{1}{Z}\prod_{t=1}^{n} exp(u_t^{str} + b_{t,t+1}^{str}) \quad (1)$$

$$u_t^{str} = \theta(c_t, c_{t+1}, y_t) \quad (2)$$

$$b_{t,t+1}^{str} = \gamma(c_{t-1}, c_t, c_{t+1}, y_t, y_{t+1}) \quad (3)$$

where the u_t^{str} represents the unary potential score of structure node $l_t = y_t$, depending on constituents c_t and c_{t+1}. The binary potential score $b_{t,t+1}^{str}$ for $s_t = y_t, s_{t+1} = y_{t+1}$ is calculated based on c_{t-1}, c_t and c_{t+1}. The binary potential provides information for discriminating between $s_t = 0, s_{t+1} = 1$ and $s_t = 1, s_{t+1} = 0$. Also, we impose the constraint that one constituent can be merged with at most one other adjacent constituent.

Figure 3 depicts the relation CRF for intra-sentential parsing. Similar to the structure CRF, $C = \{c_0, \cdots, c_t, \cdots, c_n\}$ are observed constituents and $R = \{r_1, \cdots, r_t, \cdots, r_n\}$ hidden nodes, corresponding to discourse relations. The CRF will assign a relation to a non-leaf constituent c_t based on its right and left children, $c_{t,L}$ and $c_{t,R}$, respectively. If a constituent is a single EDU, we force its hidden node to be the special label *LEAF*, whereas hidden nodes for constituents which are the product of merging cannot be *LEAF*. Given a sequence of constituents C, the probability of the relation label sequence Y^{rel} can be written as:

$$p(Y^{rel}|C) = \frac{1}{Z}\prod_{t=1}^{n} exp(u_t^{rel} + b_{t,t+1}^{rel}) \quad (4)$$

$$u_t^{rel} = \theta(c_t, y_t) \quad (5)$$

$$b_{t,t+1}^{rel} = \gamma(c_t, c_{t+1}, y_t, y_{t+1}) \quad (6)$$

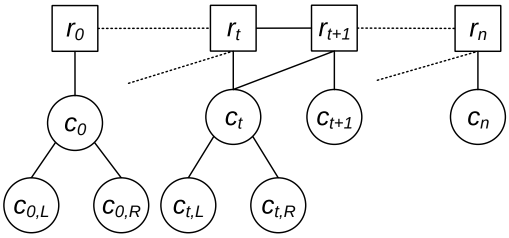


Figure 3: Intra-sentential relation CRF with pairwise modeling.

4.2 Intra-sentential Feature Learning

Instead of adopting a high order parsing model, we use neural networks to capture contextual information and recover the meaning of discourse constituents. Aside from modeling long distance dependencies, our representation learning process alleviates the need for elaborate feature engineering and selection. Our approach is based on LSTMs (Hochreiter and Schmidhuber, 1997) which have recently emerged as a popular architecture for modeling sequences and have been successfully applied to a variety of tasks ranging from machine translation (Sutskever et al., 2014), to speech recognition (Graves et al., 2013), and image description generation (Vinyals et al., 2015b). LSTMs have also been incorporated into syntactic parsing in a variety of ways (Vinyals et al. 2015a; Kiperwasser and Goldberg 2016; Dyer et al. 2015, *inter alia*). Of particular relevance to this work is LSTM-minus, a method for learning embeddings of text spans, which has achieved competitive performance in both dependency and constituency parsing (Wang and Chang, 2016; Cross and Huang, 2016). We describe below how we extend this method which is based on subtraction between LSTM hidden vectors to discourse parsing.

We represent each sentence as a sequence of word embeddings $[\boldsymbol{w}_{sos}, \boldsymbol{w}_1, \cdots, \boldsymbol{w}_i, \cdots, \boldsymbol{w}_n, \boldsymbol{w}_{eos}]$ and insert a special embedding w_E to indicate the boundaries of EDUs. We run a bidirectional LSTM over the sentence and obtain the output vector sequence $[\boldsymbol{h}_0, \cdots, \boldsymbol{h}_i, \cdots, \boldsymbol{h}_t]$, where $\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i, \overleftarrow{\boldsymbol{h}}_i]$ is the output vector for the i^{th} word, and $\overrightarrow{\boldsymbol{h}}_i$ and $\overleftarrow{\boldsymbol{h}}_i$ are the output vectors from the forward and backward directions, respectively. We represent a constituent c from position a to b with a span vector $\boldsymbol{sp}$ which is the concatenation of the vector differences $\overrightarrow{\boldsymbol{h}}_{b+1} - \overrightarrow{\boldsymbol{h}}_a$ and $\overleftarrow{\boldsymbol{h}}_{a-1} - \overleftarrow{\boldsymbol{h}}_b$:

$$\boldsymbol{sp} = [\overrightarrow{\boldsymbol{h}}_{b+1} - \overrightarrow{\boldsymbol{h}}_a, \overleftarrow{\boldsymbol{h}}_{a-1} - \overleftarrow{\boldsymbol{h}}_b] \quad (7)$$

As illustrated in Figure 4, spans are represented using output from both backward and forward LSTM components. Intuitively, this allows to obtain representations for EDUs and larger constituents in context, as embeddings are learned based on in-

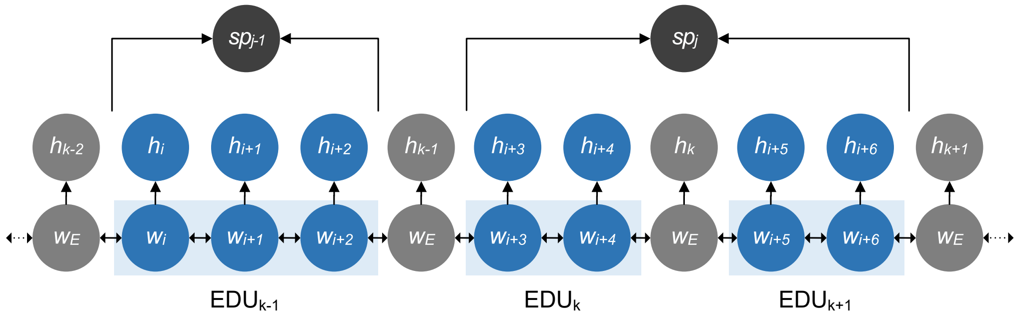


Figure 4: Modeling discourse constituents by LSTM-minus features. The feature vector $\boldsymbol{sp}_j$ for a constituent covering EDU$_k$ and EDU$_{k+1}$ is $[\overrightarrow{\boldsymbol{h}}_{k+1} - \overrightarrow{\boldsymbol{h}}_{i+3}, \overleftarrow{\boldsymbol{h}}_{k-1} - \overleftarrow{\boldsymbol{h}}_{i+6}]$. Blue nodes indicate word embeddings and LSTM outputs for words, while gray nodes represent EDU separators. Black nodes are learned LSTM-minus features for constituents.

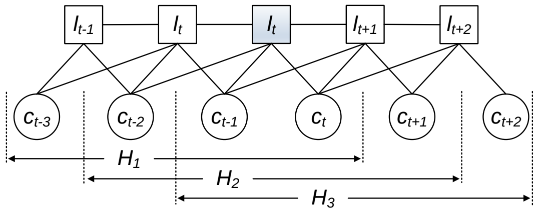


Figure 5: Multi-sentential structure CRF with sliding-window. The window size is 3, H_1, H_2, and H_3 denote the windows for predicting s_t, which is highlighted by the shaded rectangle.

formation from the span itself and the words surrounding it.

The structure CRF model calculates the unary potential score u_t^{str} and the binary potential score $b_{t,t+1}^{str}$ based on the span vector as follows:

$$\boldsymbol{u}_t^{str} = \boldsymbol{W}_u^{str}[\boldsymbol{sp}_j, \boldsymbol{sp}_{j+1}] \tag{8}$$

$$\boldsymbol{b}_{t,t+1}^{str} = \boldsymbol{W}_b^{str}[\boldsymbol{sp}_{j-1}, \boldsymbol{sp}_j, \boldsymbol{sp}_{j+1}] \tag{9}$$

where $\boldsymbol{sp}_j$ is the span vector for the j^{th} constituent in the current sequence; and $\boldsymbol{W}_u^{str} \in \mathbb{R}^{d\times 2}, \boldsymbol{W}_b^{str} \in \mathbb{R}^{d\times 4}$ are weight matrices. $\boldsymbol{b}_t^{str}$ is reshaped into a 2×2 matrix, where the $(i, j)^{th}$ entry indicates the score of the transition from label i to label j between constituent c_t and c_{t+1}.

Analogously, unary and binary potential scores for the relation CRF are calculated as:

$$\boldsymbol{u}_t^{rel} = \boldsymbol{W}_u^{rel}[\boldsymbol{sp}_{j,L}, \boldsymbol{sp}_{j,R}] \tag{10}$$

$$\boldsymbol{b}_{t,t+1}^{rel} = \boldsymbol{W}_b^{rel}[\boldsymbol{sp}_{j,L}, \boldsymbol{sp}_{j,R}, \boldsymbol{sp}_{j+1,L}, \boldsymbol{sp}_{j+1,R}] \tag{11}$$

where $\boldsymbol{sp}_{j,L}$ and $\boldsymbol{sp}_{j,R}$ are span vector representations for the left and right child of the j^{th} constituent; $\boldsymbol{W}_u^{rel} \in \mathbb{R}^{d\times n_r}, \boldsymbol{W}_b^{rel} \in \mathbb{R}^{d\times n_r^2}$, and n_r is the number of discourse relations; $\boldsymbol{b}_t^{rel}$ is also reshaped into a $n_r \times n_r$ matrix. For a constituent that is a single EDU, $\boldsymbol{sp}_{j,R}$ and $\boldsymbol{sp}_{j+1,R}$ are special vectors $\boldsymbol{w}_{LEAF_L}$ and $\boldsymbol{w}_{LEAF_R}$.

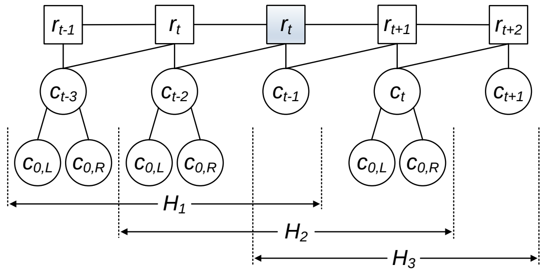


Figure 6: Multi-sentential relation CRF with sliding-window. The window size is 3, H_1, H_2 and H_3 denote the windows for predicting r_t, which highlighted by the shaded rectangle.

4.3 Multi-sentential Parser

The multi-sentential discourse parser treats sentences as the smallest possible discourse units, following a process similar to the intra-sentential model. Unfortunately, it is practically unfeasible to model all constituents with one CRF when processing entire documents. The forward-backward algorithm for calculating the CRF normalization factor on a sequence with T units has time complexity $O(TM^2)$, where M is the number of labels, leading to $O(T^2M^2)$ time for parsing a document.

We therefore modify the two CRFs into sliding-window versions. Specifically, at each level, when decoding a constituent sequence, for each hidden structure node l_i or relation node r_i, we find all windows of constituents that contain the hidden node, and set the hidden node's label according to the window with the maximum joint probability.

We present the modified sliding-window CRFs in Figure 5 (structure) and Figure 6 (relation).

4.4 Multi-sentential Feature Learning

For multi-sentential parsing, we also use the LSTM-minus method to model constituents, however, the minimum units here are sentences rather than EDUs. We represent individual sentences with the same bidirectional LSTM used for intra-sentential parsing. For sentence s_i, the LSTM output vector of the first token $\boldsymbol{h}_0$ and the last token $\boldsymbol{h}_n$ are taken to form the sentence representation $\boldsymbol{v}_i$:

$$\boldsymbol{v}_i = [\boldsymbol{h}_0, \boldsymbol{h}_n] \tag{12}$$

In order to capture additional contextual information, we then build another bidirectional LSTM over a sequence of sentence vectors. We learn representations for constituents (aka text spans within a document) with the LSTM-minus method and use $\boldsymbol{fl}$ to denote the resulting vectors. Although this flat representation is relatively straightforward to obtain, it is perhaps overly simplistic for modeling documents where the number of sentences can be relative large. Moreover, it is not clear that it is discriminating enough for modeling relations between constituents. Such relations follow structural regularities (e.g., they can be asymmetric or symmetric, left- or right-branching) which cannot be captured when adopting a sequence-based document view.

To inject structural knowledge in our representation, we also model constituents as subtrees with recursive neural networks (Socher et al., 2012). The latter operate over tree structures which we obtain during training from the RST Discourse Treebank (Carlson et al., 2003). The representation for each parent is computed based on its children iteratively, in a bottom-up fashion. More formally, let vectors $\boldsymbol{h}_L$ and $\boldsymbol{h}_R$ denote the left and right children of constituent c and dis their rhetorical relation. The vector for parent c is:

$$\boldsymbol{h}_c = \tanh(\boldsymbol{W}_{dis}[\boldsymbol{h}_L, \boldsymbol{h}_R] + \boldsymbol{b}_{dis}) \tag{13}$$

where $[\boldsymbol{h}_L, \boldsymbol{h}_R]$ is the concatenation of the children representations $\boldsymbol{h}_L \in \mathbb{R}^d$ and $\boldsymbol{h}_R \in \mathbb{R}^d$, $\boldsymbol{W}_{dis} \in \mathbb{R}^{2d\times d}$, and $\boldsymbol{b}_{dis} \in \mathbb{R}^d$ is the bias vector. We henceforth use the term tree vector $\boldsymbol{tr}$ to refer to the representation in Equation (13) since constituents are now subtrees in a document-wide discourse tree.

In multi-sentential parsing, the span vector $\boldsymbol{sp}$ now becomes the concatenation of the flat vector $\boldsymbol{fl}$ and the tree vector $\boldsymbol{tr}$:

$$\boldsymbol{sp} = [\boldsymbol{fl}, \boldsymbol{tr}] \tag{14}$$

The representation above, attempts to capture richer semantic features during the tree building process while benefiting from the robustness afforded by the flat LSTM-based text spans. An example of this representation is given in Figure 7.

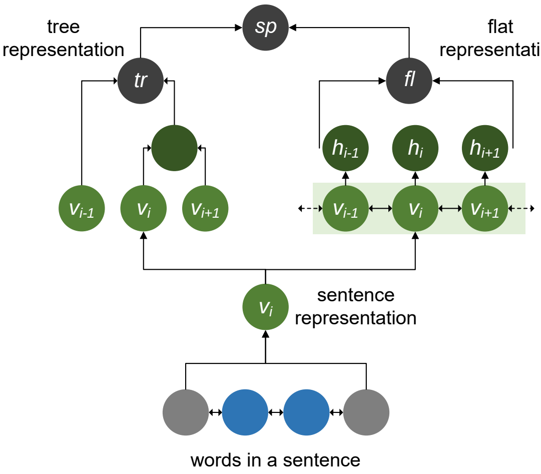


Figure 7: Document-level constituents with LSTM-minus features. Blue nodes are LSTM outputs for words, light green nodes represent vectors for sentences, dark green nodes are LSTM outputs for sentences, and black nodes are learned constituent representations.

Unary and binary potential scores for the structure and relation CRFs are calculated as in intra-sentential parsing (see Section 4.2).

4.5 Training

We first train the intra-sentential parser and use the learned LSTM as a component of the multi-sentential parser. During training, we maximize the log-likelihood of the correct label sequence Y given a constituent sequence C, which is $p(Y^{str}|C)$ for the structure CRFs and $p(Y^{rel}|C)$ for the relation CRFs. Stochastic gradient descent with momentum is used to update the parameters of the network. In our experiments, the momentum is set to 0.9 and the learning rate is 0.001. The LSTMs in our paper have one hidden layer.

5 Experimental Setup

In this section we present our experimental setup for assessing the performance of the discourse parser described above. We give details on the datasets we used, evaluation protocol, and model training.

Evaluation We evaluated our model on the RST Discourse Treebank (RST-DT; Carlson et al. 2003), which is partitioned into 347 documents for training and 38 documents for testing. Following previous work (Joty et al., 2013; Li et al., 2016), we converted non-binary relations into a cascade of right-branching binary relations.

Predicted RST-trees are typically evaluated by computing F1 against gold standard trees (Marcu, 2000). Evaluation metrics for RST-style discourse parsing include: (a) *span* (S) which measures whether the predicted subtrees match the goldstandard; (b) *nucleus* (N) which measures whether subtrees have the same nucleus as in the goldstandard and (c) *relation* (R) which measures whether discourse relations have been identified correctly. The three metrics are interdependent, errors on the span metric propagate to the nuclearity metric, and in turn to the relation metric. Following other RST-style discourse parsing systems (Hernault et al., 2010), we evaluate the relation metric using 18 coarse-grained relation classes, and with nuclearity attached, we have a total of 41 distinct relations.[1] Since EDU segmentation falls outside the scope of this work, we evaluate our system on gold-standard EDUs. Comparison systems are also assessed in the same setting.

Training Details Word embeddings were pretrained with the Gensim[2] implementation of word2vec (Mikolov et al., 2013) on the English GigaWord corpus (with case left intact). The dimensionality of the word embeddings was set to 50. Following Li et al. (2016), the embeddings were fine-tuned using a mapping matrix $\boldsymbol{W} \in \mathbb{R}^{50\times 50}$ trained with the following criterion:

$$\min_{\boldsymbol{W},\boldsymbol{b}} \|\boldsymbol{LT}_{tuned} - \boldsymbol{LT}_{pre}\boldsymbol{W} + \boldsymbol{b}\| \qquad (15)$$

where $\boldsymbol{LT}_{tuned}$, and $\boldsymbol{LT}_{pre}$ are lookup tables for fine-tuned and pre-trained word embeddings in the training set. Matrix W can be subsequently used to to estimate fine-tuned embeddings for words in the test set.

Tokenization, POS tagging and sentence splitting were performed using the Stanford CoreNLP toolkit (Manning et al., 2014). All neural network parameters were initialized randomly with Xavier's initialization (Glorot and Bengio, 2010). The hyper-parameters are tuned by cross-validation on the training set.

CIDER	S	N	R
Tree Span	79.5	68.1	56.6
Flat Span (−minus)	82.7	69.3	55.6
Flat Span (+minus)	83.6	70.1	55.4
Tree + Flat Span (+minus)	83.6	71.1	57.3

Table 1: CIDER performance using different constituent representations (RST-DT test set).

Additional Features Most existing state-of-the-art systems rely heavily on handcrafted features (Hernault et al., 2010; Feng and Hirst, 2014; Joty et al., 2013) some of which have been also proved helpful in neural network models (Li et al., 2014, 2016). In our experiments, we use the following basic features which have been widely adopted in various discourse parsing models: (1) the first three words and the last two words of each constituent; (2) the POS tags of the first three words and the last two words of each constituent; (3) the number of EDUs; and (4) the number of tokens of each constituent. We concatenate these features with the constituent vectors learned by our neural networks, and train new CRF models.

6 Results

In this paper we have presented two views for modeling discourse constituents, namely as trees or sequences. We experimentally assessed whether these two views are overlapping or complementary. Table 1 reports the performance of our parser which we call CIDER (as a shorthand for **C**ontextually **I**nformed **D**iscourse Pars**er**) without the additional features introduced in Section 5. The first row presents a version of CIDER based solely on tree span representations. In the second row, CIDER uses flat representations without LSTM minus features. Specifically, each constituent is represented as the average of the LSTM outputs within it. In the third row, CIDER's representations are computed with the LSTM minus method, while the fourth row shows results for the full system.

As can be seen, on the span (S) metric, CIDER with flat span representations is much better than CIDER with tree span representations; on the nuclearity (N) metric tree representations are still inferior to flat representations but the performance

[1] For calculating the binary potential scores, 41 relations will lead to a large number of parameters; to avoid this, we only use 18 relations without nuclearity.

[2] `https://radimrehurek.com/gensim/`

Discourse Parsers	S	N	R	Speed
Ji and Eisenstein (2014)	82.1	71.1	61.6	0.21
Feng and Hirst (2014)	85.7	71.0	58.2	9.88
Heilman and Sagae (2015)	83.5	68.1	55.1	0.40
CIDER (−AF)	83.6	71.1	57.3	3.80
CIDER (+AF)	85.0	71.1	59.0	3.80
Li et al. (2014)	84.0	70.8	58.6	26.00
Li et al. (2016)	85.8	71.7	58.9	–
Human	88.7	77.7	65.8	–

Table 2: Comparison with state-of-the-art systems (RST-DT test set). Speed indicates the average number seconds taken to parse a document.

gap is narrower, whereas on the the relation (R) metric, tree span representations are superior. We believe this can be explained by the fact that relation identification relies on more semantic information while span identification relies on more shallow features, like the beginning and the end of the span (Ji and Eisenstein, 2014). Span features based on the LSTM minus method bring improvements over vanilla LSTM representations on the span and nuclearity metrics. Perhaps unsurprisingly, the combination of span and tree representations achieves the best results overall.

In Table 2, we compare our system with several state-of-the-art discourse parsers, which can be classified in two groups depending on their time complexity. Linear-time systems (first block in the table) include two transition-based parsers (Ji and Eisenstein, 2014; Heilman and Sagae, 2015) and one CRF-based parser (Feng and Hirst, 2014), whereas cubic-time parsers (second block) include two neural network models (Li, Li, and Hovy, 2014; Li, Li, and Chang, 2016). CIDER falls in the first group as it is a liner-time parser, while it shares with parsers in the second group the use of neural architectures for automated feature extraction. We report CIDER scores with and without the additional features (AF) discussed in the previous section. As an upper bound, we also report inter-annotator agreement on the discourse parsing task (last row in the table).

Amongst linear-time systems, our parser achieves comparable results on the span and relation metric, and best performance on the nuclearity metric. Note that the three metrics evaluate different aspects of a discourse parser, and CIDER achieves the most balanced results across all metrics. As far as other comparison systems are concerned, Ji and Eisenstein (2014) employ a shift-reduce discourse parser. They represent EDUs with word-count vectors and use a projection matrix to combine them into text spans. A support vector machine classifier is used to decide the actions of the parser. Heilman and Sagae (2015) also adopt a shift-reduce approach and use multi-class logistic regression to select the best parsing action. Their classifier considers a variety of lexical, syntactic, and positional features. Feng and Hirst's (2014) system is closest to ours in their use of linear-chain CRFs, but their features are mainly extracted from local constituents. Furthermore, they adopt a post-editing method which modifies the discourse trees their parser creates with height features.

With regard to previously proposed cubic-time systems, CIDER outperforms Li et al. (2014) across all metrics. Their CYK-based parser adopts a recursive deep model for composing EDUs hierarchically together with several additional features to boost performance. CIDER performs slightly worse on span and nuclearity compared to Li et al. (2016), but is better at identifying relations. Their system uses an attention-based hierarchical neural network for modeling text spans and a tensor-based transformation for combining two spans. A CKY-like algorithm is used to generate the discourse tree structure. In comparison, CIDER is conceptually simpler, and more efficient.

We used paired bootstrap re-sampling (Efron and Tibshirani, 1993) to assess whether differences in performance are statistically significant. CIDER is significantly better than Feng and Hirst's 2014 system on the relation metric ($p < 0.05$); it is also significantly better ($p < 0.05$) than Heilman and Sagae (2015) on all three metrics and better than Ji and Eisenstein (2014) on the span metric. Compared to Li et al. (2014), CIDER is significantly better on the span and relation metrics ($p < 0.05$). Unfortunately, we cannot perform significance tests against Li et al. (2016) as we do not have access to the output of their system.

We also evaluated the speed of CIDER and comparison discourse parsers on a platform with Intel Core-i5-7200U CPU at 2.50GHz. We report the average number of seconds taken to parse a document in the RST-DT test set. The times shown in Table 2 do not include pre-processing, which for CIDER is only part-of-speech tagging, whereas all other linear-time systems rely on a syntactic parser. As can be seen CIDER is quite efficient compared to related systems (Feng and Hirst, 2014;

Heilman and Sagae, 2015) whilst requiring less feature engineering.

7 Conclusions

In this paper we described CIDER, a simple and efficient discourse parser which adopts a two-stage parsing strategy, whilst exploiting a more global feature space. We proposed a novel way to learn contextually informed representations of constituents with the LSTM minus method, at the sentence and document level. We also demonstrated that flat representations of text spans can be usefully complemented with tree-based ones leading to a more accurate characterization of discourse relations. Experimental results showed that CIDER performs on par with the state of the art (Li et al., 2016), despite the greedy parsing algorithm and relatively simple neural architecture. In the future, we would like to improve parsing accuracy by leveraging unlabeled text rather than relying exclusively on human annotated training data.

Acknowledgments The authors gratefully acknowledge the support of the European Research Council (award number 681760).

References

Jason Baldridge and Alex Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *Proceedings of the CoNLL conference*. pages 96–103.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Current and new directions in discourse and dialogue*, Springer, pages 85–112.

Joyce Y. Chai and Rong Jin. 2004. Discourse structure for context question answering. In *HLT-NAACL 2004: Workshop on Pragmatics of Question Answering*. pages 23–30.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the EMNLP conference*. pages 1–11.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the ACL conference*. pages 334–343.

Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction ot the Bootstrap*. Chapman & Hall, New York, NY.

Vanessa Wei Feng and Graeme Hirst. 2012. Text-level discourse parsing with rich linguistic features. In *Proceedings of the ACL conference*. pages 60–68.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the ACL conference*. pages 511–521.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. volume 9, pages 249–256.

Alex Graves, Jaitly Navdeep, and A-R Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *IEEE Workshop on Automatic Speech Recognition and Understanding*. pages 293–278.

Michael Heilman and Kenji Sagae. 2015. Fast rhetorical structure theory discourse parsing. *arXiv preprint arXiv:1505.02425* .

Hugo Hernault, Helmut Prendinger, David A DuVerle, Mitsuru Ishizuka, and Tim Paek. 2010. Hilda: a discourse parser using support vector machine classification. *Dialogue and Discourse* 1(3):1–33.

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

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the ACL conference*. pages 13–24.

Shafiq Joty, Giuseppe Carenini, Raymond Ng, and Yashar Mehdad. 2013. Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In *Proceedings of the ACL conference*. pages 486–496.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327.

Huong LeThanh, Geetha Abeysinghe, and Christian Huyck. 2004. Generating discourse structures for written texts. In *Proceedings of the 20th international conference on Computational Linguistics*. page 329.

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014. Recursive deep models for discourse parsing. In *Proceedings of the EMNLP conference*. pages 2061–2069.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Discourse parsing with attention-based hierarchical neural networks. In *Proceedings of the EMNLP conference*. pages 362–371.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse* 8(3):243–281.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the ACL conference: System Demonstrations*. pages 55–60.

Daniel Marcu. 2000. The rhetorical parsing of unrestricted texts: A surface-based approach. *Computational linguistics* 26(3):395–448.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The Penn discourse treebank 2.0. In *Proceedings of LREC*.

Kenji Sagae. 2009. Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In *Proceedings of the 11th International Conference on Parsing Technologies*. pages 81–84.

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

Swapna Somasundaran. 2010. *Discourse-level Relation for Opinion Analysis*. Ph.D. thesis, University of Pittsburgh.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the NAACL conference*. pages 149–156.

Rajen Subba and Barbara Di Eugenio. 2009. An effective discourse parser that uses rich linguistic information. In *Proceedings of the ACL conference*. pages 566–574.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the NIPS conference*. pages 3104–3112.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015a. Grammar as a foreign language. In *Proceedings of the NIPS conference*. pages 2773–2781.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015b. Show and tell: A neural image caption generator. In *Proceedings of the CVPR conference*. pages 3156–3164.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proceedings of the ACL conference*. pages 2306–2315.

Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata. 2014. Dependency-based discourse parser for single-document summarization. In *Proceedings of the EMNLP conference*. pages 1834–1839.

Multi-task Attention-based Neural Networks for Implicit Discourse Relationship Representation and Identification

Man Lan[1,2], **Jianxiang Wang**[1,3*], **Yuanbin Wu**[1,2*], **Zheng-Yu Niu**[3*] , **Haifeng Wang**[3]
[1] School of Computer Science and Software Engineering, East China Normal University
[2] Shanghai Key Laboratory of Multidimensional Information Processing, P.R.China
[3] Baidu Inc., Beijing, P.R.China
{mlan,ybwu}@cs.ecnu.edu.cn
{wangjianxiang01,niuzhengyu,wanghaifeng}@baidu.com

Abstract

We present a novel multi-task attention-based neural network model to address implicit discourse relationship representation and identification through two types of representation learning, an attention-based neural network for learning discourse relationship representation with two arguments and a multi-task framework for learning knowledge from annotated and unannotated corpora. The extensive experiments have been performed on two benchmark corpora (i.e., PDTB and CoNLL-2016 datasets). Experimental results show that our proposed model outperforms the state-of-the-art systems on benchmark corpora.

1 Introduction

The task of implicit discourse relation (or rhetorical relation) identification is to recognize how two adjacent text spans without explicit discourse marker (i.e., connective, e.g., *because* or *but*) between them are logically connected to one another (e.g., *cause* or *contrast*). It is considered to be a crucial step for discourse analysis and language generation and helpful to many downstream NLP applications, e.g., QA, MT, sentiment analysis, machine comprehension, etc.

With the release of PDTB 2.0 (Prasad et al., 2008), lots of work has been done for discourse relation identification on natural (i.e., genuine) discourse data (Pitler et al., 2009; Lin et al., 2009; Wang et al., 2010; Zhou et al., 2010; Braud and Denis, 2015; Fisher and Simmons, 2015) with the use of traditional NLP linguistically informed features and machine learning algorithms. Recently, more and more researchers resorted to neural networks for implicit discourse recognition (Zhang et al., 2015; Chen et al., 2016; Liu et al., 2016b; Qin et al., 2016a; Liu and Li, 2016; Braud and Denis, 2016; Wu et al., 2016). Meanwhile, to alleviate the shortage of labeled data, researchers explored multi-task learning with the aid of unannotated data for implicit discourse recognition either in traditional machine learning framework (Collobert and Weston, 2008; Lan et al., 2013) or recently in neural network framework (Wu et al., 2016; Liu et al., 2016b).

In this work, we present a novel multi-task attention-based neural network to address implicit discourse relationship representation and recognition. It performs two types of representation learning at the same time. An attention-based neural network conducts discourse relationship representation learning through interaction between two discourse arguments. Meanwhile, a multi-task learning framework leverages knowledge from auxiliary task to enhance the performance of main task. Furthermore, these two types of learning are integrated into one neural network framework and work together to maximize the overall performance.

The contributions of this work are listed as follows.

- We propose a multi-task attention-based neural network model to address implicit discourse relationship representation and recognition, which benefits from both the interaction between discourse arguments and the interaction between different learning tasks;
- Our method achieves the best results on two benchmark corpora in comparison with the state-of-the-art systems so far.

The organization of this work is as follows. Section 2 describes the proposed novel multi-task neural network. Section 3 introduces the exper-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1299–1308
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

imental settings in detail. Section 4 reports the comprehensive experimental results on two benchmark corpora. Section 5 summarized related work. Finally, Section 6 concludes this work.

2 Multi-task Attention-based Neural Networks Models

2.1 Motivation

The idea of learning two types of interactive knowledge from arguments and from multi-tasks is motivated by the following observations and analysis.

On the one hand, to recognize the discourse relationships, our system needs to understand the meaning of each argument and infer the discourse sense transferred between two arguments (denoted as *Arg*-1 and *Arg*-2). Learning the semantic representation of each argument (sentence) has been studied with the use of many neural network models and their variants (e.g., CNN, RNN, LSTM, Bi-LSTM, ect). However, learning the complicated and various types of discourse relationships between arguments may not be performed by simply summing up or concatenating two argument representations. We analyze the discourse with contrast relationship and find that the contrast information may result from different parts of sentence, e.g., tenses (e.g., previous vs. now), entities (their vs our), or even the whole arguments, etc. Therefore, in order to learn the relationship representation between two arguments, we propose an attention mechanism that can select out the most important part from two arguments and perform the information interaction between two arguments.

On the other hand, one common issue involved in implicit discourse relationship identification is the lack of labeled data. In this work, we state that the relevant information from unlabelled data might be helpful and we present a novel multi-task learning framework. In contrast with previous multi-task learning framework in traditional machine learning, we improve multi-task learning framework with representation learning for better discourse relationship representation.

Inspired by the above considerations, we present a novel multi-task attention-based neural network model by integrating attention mechanism with multi-task learning for information interaction between arguments and between tasks.

2.2 Learning Discourse Representation

To learn the semantic representation of each argument in discourse, a lot of neural network models and their variants have been proposed, such as, convolutional neural network (CNN), recurrent neural network (RNN) and so on. As a variant of RNN, long-short term memory (LSTM) neural network specifically addresses the issue of learning long-term dependencies and is good at modeling over a sequence of words with consideration of the contextual information. Therefore, in this work we adopt LSTM to model discourse argument.

2.2.1 LSTM for Argument Representation

Figure 1 shows the traditional LSTM model for representation learning of arguments. First of all, through the embedding layer, we associate each word w in the vocabulary with a vector representation $\boldsymbol{x}_w \in \mathbb{R}^{d_w}$. Let $\boldsymbol{x}_i^1$ ($\boldsymbol{x}_i^2$) be the i-th word vector in *Arg*-1 (*Arg*-2), then these two discourse arguments are represented as:

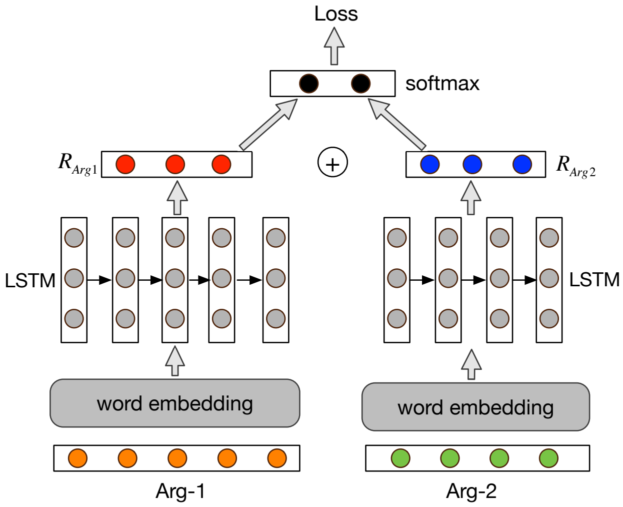


Figure 1: LSTM for discourse argument pair representation learning.

$$Arg\text{-}1\text{: } [\boldsymbol{x}_1^1, \boldsymbol{x}_2^1, \cdots, \boldsymbol{x}_{L_1}^1] \quad (1)$$

$$Arg\text{-}2\text{: } [\boldsymbol{x}_1^2, \boldsymbol{x}_2^2, \cdots, \boldsymbol{x}_{L_2}^2] \quad (2)$$

where *Arg*-1 (*Arg*-2) has L_1 (L_2) words.

Given the word representations of the argument $[\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_L]$ as the input sequence, an LSTM computes the state sequence $[\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_L]$ for each time step i using the following formulation:

$$\boldsymbol{i}_i = \sigma(\boldsymbol{W}_i[\boldsymbol{x}_i, \boldsymbol{h}_{i-1}] + \boldsymbol{b}_i) \tag{3}$$
$$\boldsymbol{f}_i = \sigma(\boldsymbol{W}_f[\boldsymbol{x}_i, \boldsymbol{h}_{i-1}] + \boldsymbol{b}_f) \tag{4}$$
$$\boldsymbol{o}_i = \sigma(\boldsymbol{W}_o[\boldsymbol{x}_i, \boldsymbol{h}_{i-1}] + \boldsymbol{b}_o) \tag{5}$$
$$\tilde{\boldsymbol{c}}_i = tanh(\boldsymbol{W}_c[\boldsymbol{x}_i, \boldsymbol{h}_{i-1}] + \boldsymbol{b}_c) \tag{6}$$
$$\boldsymbol{c}_i = \boldsymbol{i}_i \odot \tilde{\boldsymbol{c}}_i + \boldsymbol{f}_i \odot \boldsymbol{c}_{i-1} \tag{7}$$
$$\boldsymbol{h}_i = \boldsymbol{o}_i \odot tanh(\boldsymbol{c}_i) \tag{8}$$

where [ ] means the concatenation operation of vectors, σ denotes the sigmoid function and $\odot$ denotes element-wise product. Besides, $\boldsymbol{i}_i$, $\boldsymbol{f}_i$, $\boldsymbol{o}_i$ and $\boldsymbol{c}_i$ denote the input gate, forget gate, output gate and memory cell, respectively. Moreover, we also use bidirectional LSTM (Bi-LSTM) which is able to capture the context from both past and future rather than LSTM which only considers the context information from the past. Therefore, at each position i of the sequence, we obtain two states $\overrightarrow{\boldsymbol{h}}_i$ and $\overleftarrow{\boldsymbol{h}}_i$, where $\overrightarrow{\boldsymbol{h}}_i, \overleftarrow{\boldsymbol{h}}_i \in \mathbb{R}^{d_h}$. Then we concatenate them to get the intermediate state, i.e. $\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i, \overleftarrow{\boldsymbol{h}}_i]$. After that, we sum up the sequence states $[\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_L]$ to get the representations of *Arg*-1 and *Arg*-2 respectively as follows:

$$\boldsymbol{R}_{Arg_1} = \sum_{i=1}^{L_1} \boldsymbol{h}_i^1 \tag{9}$$

$$\boldsymbol{R}_{Arg_2} = \sum_{i=1}^{L_2} \boldsymbol{h}_i^2 \tag{10}$$

Finally we concatenate the two argument representations $\boldsymbol{R}_{Arg_1}$ and $\boldsymbol{R}_{Arg_2}$ as the argument pair representation, i.e., $\boldsymbol{R}_{pair} = [\boldsymbol{R}_{Arg_1}, \boldsymbol{R}_{Arg_2}]$.

Clearly, in this way, there is no any correlation and interaction between the two arguments. That is, whatever the types of discourse relationship they hold, the argument pair representation $\boldsymbol{R}_{pair}$ is independent from $\boldsymbol{R}_{Arg_1}$ or $\boldsymbol{R}_{Arg_2}$.

2.2.2 Attention Neural Network for Relationship Representation

In order to effectively capture the complicated and various types of relationships between arguments, we proposed a novel attention-based neural network model shown in Figure 2.

To do it, we first compute the match between $\boldsymbol{R}_{Arg_1}$ ($\boldsymbol{R}_{Arg_2}$) and each state $\boldsymbol{h}_i^2$ ($\boldsymbol{h}_i^1$) of *Arg*-2 (*Arg*-1) by taking the inner product followed by a softmax as follows:

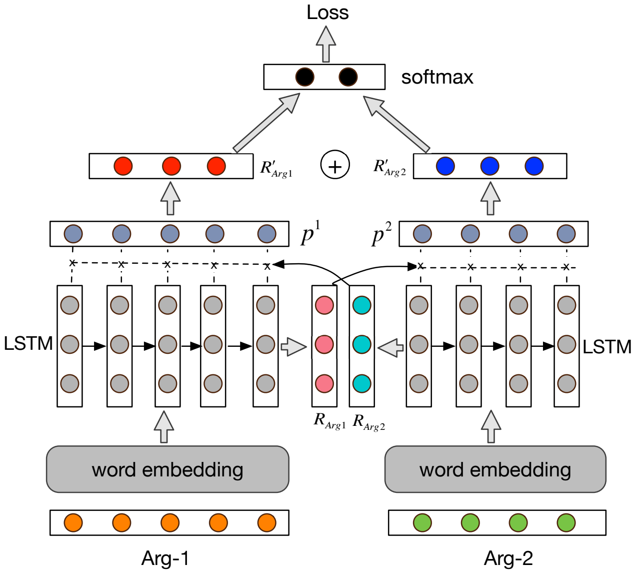


Figure 2: Attention Neural Network for representation learning of arguments.

$$p_i^1 = \text{Softmax}(\boldsymbol{R}_{Arg_2}^T \boldsymbol{h}_i^1) \tag{11}$$
$$p_i^2 = \text{Softmax}(\boldsymbol{R}_{Arg_1}^T \boldsymbol{h}_i^2) \tag{12}$$

where $\text{Softmax}(z_i) = e^{z_i} / \sum_j e^{z_i}$. Here $\boldsymbol{p}$ is an attention (probability) vector over the inputs and can be viewed as the weights of the words measuring to what degree our model should pay attention to. It is worth noting that $\boldsymbol{p^1}$ and $\boldsymbol{p^2}$ are determined by $\boldsymbol{R}_{Arg_2}$ and $\boldsymbol{R}_{Arg_1}$ respectively, which means the representation of one argument depends on the representation of the other.

Next, we sum over the state $\boldsymbol{h}_i$ weighted by the attention vector $\boldsymbol{p}$ to compute the new representations for *Arg*-1 and *Arg*-2 respectively as below:

$$\boldsymbol{R}'_{Arg_1} = \sum_{i=0}^{L_1} \boldsymbol{h}_i^1 p_i^1 \tag{13}$$

$$\boldsymbol{R}'_{Arg_2} = \sum_{i=0}^{L_2} \boldsymbol{h}_i^2 p_i^2 \tag{14}$$

The representation of *Arg*-2 ($\boldsymbol{R}_{Arg_2}$) is used to compute the weights of words in *Arg*-1 (i.e., $\boldsymbol{p^1}$) and $\boldsymbol{R}_{Arg_1}$ is used to compute the weights of words in *Arg*-2 (i.e., $\boldsymbol{p^2}$). In this way, the new representations of the two arguments interact with each other. Therefore, this attention mechanism enables our model to focus on specific spans in the two arguments, which is crucial to recognize the discourse relations. We then concatenate $\boldsymbol{R}'_{Arg_1}$

and $\boldsymbol{R}'_{Arg_2}$ to get the argument pair representation $\boldsymbol{R}_{pair} = [\boldsymbol{R}'_{Arg_1}, \boldsymbol{R}'_{Arg_2}]$.

Finally, we feed the argument pair vector $\boldsymbol{R}_{pair}$ to a fully-connected softmax layer which outputs the probabilities of different classes for the classification task. Here we choose the cross-entropy loss between the outputs of the softmax layer and the ground-truth class labels as our loss function.

2.3 Multi-task Attention-based Neural Networks

The model presented in Section 2.2 can perform implicit discourse relation recognition in itself. However, similar with many models in deep learning, one big issue is the lack of labeled data. Therefore, we propose a multi-task attention-based neural network by integrating the aforementioned model into a multi-task learning framework to address the implicit discourse relation recognition with the aid of large amount of unlabelled data. Figure 3 shows the general framework of our proposed multi-task attention-based neural network model.

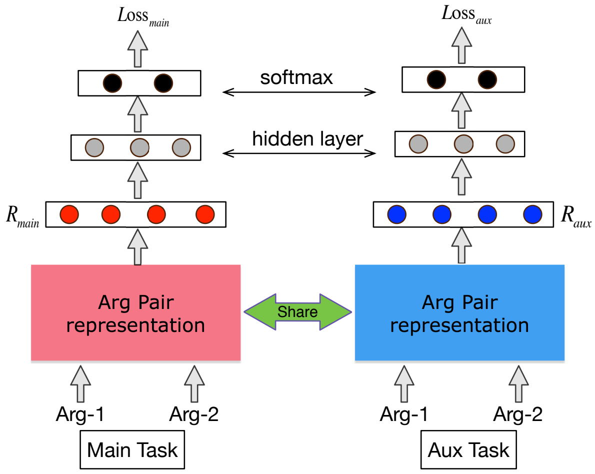


Figure 3: The framework of our proposed multi-task attention-based neural network model.

We use the aforementioned attention-based neural network to map the argument pair into a low-dimensional vector ($\boldsymbol{R}_{pair}$) denoted as `Arg Pair representation` component in Figure 3. Under the multi-task learning framework, the parameters of the `Arg Pair representation` components are shared between the main task and the auxiliary tasks. We denote $\boldsymbol{R}_{main}$ and $\boldsymbol{R}_{aux}$ as the representations of argument pair for main and auxiliary tasks, respectively. And we add a hidden layer after $\boldsymbol{R}_{main}$ and $\boldsymbol{R}_{aux}$ to learn the task-specific representations followed by the softmax layers used to compute the loss of the main task ($Loss_{main}$) and the loss of the auxiliary task ($Loss_{aux}$), respectively.

Regarding the strategy of sharing knowledge learnt from auxiliary to main task, we propose the following three methods.

2.3.1 Equal Share

A simple and straightforward way is to equally share the knowledge learned from main task and auxiliary task. Therefore, the total loss of the multi-task neural network is calculated as:

$$Loss = Loss_{main} + Loss_{aux} \tag{15}$$

where $Loss_{aux}$ has the same weight as $Loss_{main}$.

2.3.2 Weighted Share

Another method is to give different weights to the main and auxiliary task as below:

$$Loss = Loss_{main} + w * Loss_{aux} \tag{16}$$

where $w \in (0, 1]$ is a weight parameter. Clearly, a lower value of w means less importance of auxiliary task.

2.3.3 Sigmoid (Gated) Interaction

The above two ways of sharing knowledge actually have no deep interaction between the main and auxiliary tasks. They only share equal or weighted contributions from tasks to final result. Therefore, we propose a model that can perform interaction between tasks, which is shown in Figure 4.

We introduce two important parameters $\boldsymbol{W}_{inter} \in \mathbb{R}^{d_{pair} \times d_{pair}}$ and $\boldsymbol{b}_{inter} \in \mathbb{R}^{d_{pair}}$ (d_{pair} is the length of the argument pair representation vector $\boldsymbol{R}_{pair}$) to fulfil the interaction between main and auxiliary tasks. As shown in the following Formula (17) and (18), the new representation of argument pair $\boldsymbol{R}'_{main}$ is updated by the combination of $\boldsymbol{W}_{inter}$ and $\boldsymbol{R}_{aux}$ using a Sigmoid function.

$$\boldsymbol{R}'_{main} = \boldsymbol{R}_{main} \odot \sigma(\boldsymbol{W}_{inter}\boldsymbol{R}_{aux} + \boldsymbol{b}_{inter}) \tag{17}$$

$$\boldsymbol{R}'_{aux} = \boldsymbol{R}_{aux} \odot \sigma(\boldsymbol{W}_{inter}\boldsymbol{R}_{main} + \boldsymbol{b}_{inter}) \tag{18}$$

$\boldsymbol{W}_{inter}$ and the Sigmoid function (σ) work together to make information interacted between two tasks and select useful relevant information out of the opposite tasks as well. Clearly, $\boldsymbol{W}_{inter}$ is

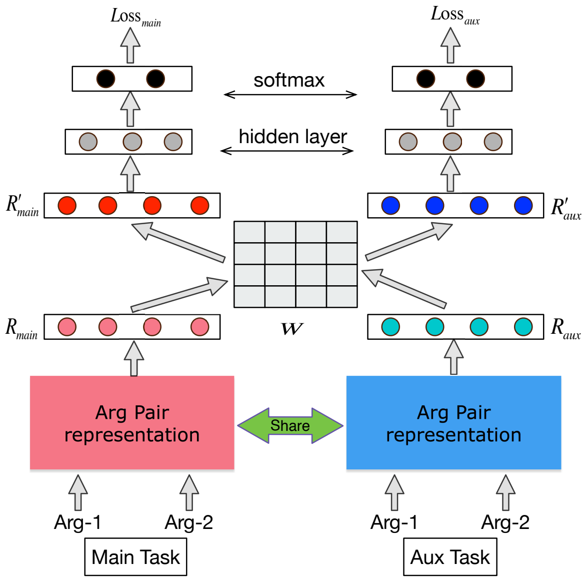


Figure 4: Sigmoid (Gated) interaction shared in multi-task framework (GShare).

a parameter to be trained. This mechanism acts as a gate to determine how much the information would pass through to the final result. Therefore, under the framework of multi-task and gated mechanism, the main and auxiliary tasks are capable of not only sharing the parameters of learning argument pair representation but also interacting the representations learning from each other.

2.4 Parameter Learning

We tried various settings of word embeddings trained on the BLLIP corpus with different dimensions d_{WE} = [50, 100, 150, 200] by *word2vec tool*[1] and finally set dimensionality as 50 based on the results on development set. we also explored the hidden state d_h = [50, 100, 150, 200] and the size of hidden layer in multi-task framework $d_{multi-task}$ = [50, 80, 120, 150]. Finally, for binary classification and four way classification on PDTB, we chose $d_h = 50$ and $d_{multi-task} = 80$. For multi-class classification on CoNLL-2016, we chose $d_h = 100$ and $d_{multi-task} = 120$. We applied dropout to the penultimate layer and set the dropout rate as 0.5. These parameters remain the same in experiments except the share weight w varies which will be discussed later. We chose the cross-entropy loss as loss function and adopted AdaGrad (Duchi et al., 2011) with a learning rate of 0.001 and a minibatch size of 64 to train the model.

3 Experiment Settings

3.1 Datasets

We adopted three corpora: PDTB 2.0 and CoNLL-2016 datasets are annotated for discourse relation recognition evaluation, and the BLLIP corpus is unlabeled and used as auxiliary task.

PDTB 2.0 is the largest annotated corpus of discourse relations, which contains 2, 312 Wall Street Journal (WSJ) articles. The sense label of discourse relations is hierarchically with three levels, i.e., class, type and sub-type. The top level contains four major semantic classes: Comparison (denoted as Comp.), Contingency (Cont.), Expansion (Exp.) and Temporal (Temp.). For each class, a set of types is used to refine relation sense. The set of subtypes is to further specify the semantic contribution of each argument. We focus on the top level (class) relations. Following (Pitler et al., 2009), we used sections 2-20 as training set, sections 21-22 as test set, and sections 0-1 as development set. Table 1 summarizes the statistics of four top level implicit discourse relations in PDTB.

Relation	Train	Dev	Test
Comp.	1942	197	152
Cont.	3342	295	279
Exp.	7004	671	574
Temp.	760	64	85

Table 1: The statistics of four top level implicit discourse relations in PDTB 2.0.

The CoNLL-2016 Shared Task focuses on shallow discourse parsing, which provides two test datasets, i.e., one from PDTB section 23 denoted as CoNLL-Test set, and the other from a similar source and domain (English Wikinews[2]) denoted as CoNLL-Blind test set. Different from the sense labels in PDTB, the CoNLL-Test set has three sense levels and the EntRel label. Moreover, it merges several labels in the original annotation to reduce some sparsity without losing too much of the utility and the semantics of the sense.

BLLIP The North American News Text (Complete) is used as unlabeled data source to generate synthetic labeled data for auxiliary task. We remove the explicit discourse connectives from raw texts and grant the explicit relations as the synthetic implicit relations. We obtain a resulting corpus with 100, 000 implicit relations by random sampling.

[1] http://www.code.google.com/p/word2vec

[2] https://en.wikinews.org/

3.2 Evaluation Measures

We adopt precision (*P*), recall (*R*) and their harmonic mean, i.e., F_1 for performance evaluation. We also report accuracy for direct comparison with previous works.

4 Results and Discussion

4.1 Results on PDTB in multiple binary classification

To be consistent with previous work, we first perform multiple binary classification (one-versus-other) on the four top level classes in PDTB. Several previous studies merged EntRel with Expansion, which is also explored in our study and noted as Exp+. Table 2 shows the results of our proposed three models in terms of F_1 (%) on PDTB using multiple binary classification, where STL means single task learning, *Eshare*, *Wshare* and *Gshare* denote the equal share, weighted share and gated interaction share under multi-task framework respectively, *Imp* denotes the standard implicit relations dataset in PDTB (similarly, *Imp* denotes standard implicit relations dataset in the CoNLL dataset when we perform experiments on the CoNLL dataset) used for training, *Exp* denotes all explicit relations in sections 00-24 in PDTB (similarly, all explicit relations in the CoNLL dataset when we perform experiments on the CoNLL dataset), and *BLLIP* denotes the synthetic implicit relations extracted from BLLIP. For example, *Imp* + *BLLIP* indicates that *Imp* is used for main task and *BLLIP* is for auxiliary task.

The first three rows in Table 2 list the results of LSTM, Bi-LSTM and attention neural network in the single task learning (STL) framework, which act as baselines for comparison with multi-task learning. We see that Bi-LSTM achieve slightly better performance than LSTM, which is consistent with previous work as Bi-LSTM considers the forward and backward direction contextual information while LSTM only considers the forward information. Compared with LSTM and Bi-LSTM, the attention neural network achieves much better performance. This indicates the effectiveness of attention mechanism for capturing the interaction between discourse arguments, which is crucial for relationship representation.

Generally, under the multi-task neural network framework, the three proposed multi-task neural networks, i.e., *Eshare*, *Wshare* and *Gshare*, outperform the single task learning methods. Comparing with *Eshare* and *Wshare*, we see that using a low value of w is able to boost the performance and reduce the negative influence brought by auxiliary task. We then use the best w value in *Wshare* to construct the loss of *Gshare* and the *Gshare* achieves the best performance among all methods through information interaction between main and auxiliary tasks.

Comparing *Imp* + *Exp* with *Imp* + *BLLIP*, we see that using *Exp* as auxiliary task achieves lower performance than using *BLLIP* and even hurts the performance compared with the single task. The possible reasons may result from (1) there is difference between explicit and implicit discourse relations and (2) the size of *Exp* dataset is much smaller than that of *BLLIP* and thus it is not large enough to boost the performance.

4.2 Results on PDTB and CoNLL-2016 in multi-class classification

We also perform multi-class classification on PDTB and CoNLL-2016. That is, a four-way classification on the four top-level classes in PDTB and a 15-way classification on the 15 sense labels in CoNLL dataset. Table 3 shows the results of multi-class classification on PDTB and CoNLL-2016 corpora in terms of accuracy (%) and macro-averaged F_1 (%).

The results of multi-class classification are consistent with the results of binary classification. First, the attention neural network achieves better performance than LSTM and Bi-LSTM. Second, the multi-task learning methods outperform the single-task learning method. Thrid, the *Gshare* method achieves the best performance.

4.3 Comparison with the state-of-the-art Systems

Table 4 lists the performance of our best model with the reported state-of-the-art systems on PDTB and CoNLL-2016. We see that our model achieves F_1 improvements of 1.64% on Cont., 0.97% on Exp., and 1.35% on Exp.+ against the best reported systems in binary classification. And in multi-class classification, our model also achieves the best performance of F_1 in four-way classification and accuracy in CoNLL-2016 Blind test set, which indicates that our model has good generality.

Specially, (Liu et al., 2016b) and (Liu and Li, 2016) listed in Table 4, which adopted neural network-based multi-task framework, are quite

		Comp.	Cont.	Exp.	Exp+	Temp
STL	LSTM	33.50	52.09	67.51	76.12	27.88
	Bi-LSTM	33.82	52.30	67.47	76.36	29.01
	Attention	**38.15**	**56.07**	**70.53**	**79.80**	**36.72**
Eshare	*Imp + Exp*	35.07	54.62	69.97	79.15	34.57
	Imp + BLLIP	37.67	56.82	70.81	80.43	35.48
Wshare	*Imp + Exp*	37.51 (w=0.1)	55.83 (w=0.2)	70.37 (w=0.3)	80.22(w=0.2)	35.71 (w=0.3)
	Imp + BLLIP	39.13 (w=0.2)	57.78(w=0.2)	71.88(w=0.1)	80.84 (w=0.3)	37.76(w=0.3)
Gshare	*Imp + Exp*	38.91	56.91	71.41	80.02	36.92
	Imp + BLLIP	**40.73**	**58.96**	**72.47**	**81.36**	**38.50**

Table 2: Performance of multiple binary classification on the top level classes in PDTB corpus in terms of F_1 (%).

		PDTB (Four way)	CoNLL-Test (*Acc*)	CoNLL-Blind (*Acc*)
STL	LSTM	F_1: 36.16; *Acc*: 56.12	34.45	35.07
	Bi-LSTM	F_1: 36.54; *Acc*: 54.30	34.85	35.83
	Attention	F_1: **45.57**; *Acc*: **57.55**	**37.41**	**38.36**
Eshare	*Imp + Exp*	F_1: 44.17; *Acc*: 55.65	35.56	37.06
	Imp + BLLIP	F_1: 44.57; *Acc*: 55.85	36.66	38.28
Wshare	*Imp + Exp*	F_1: 45.03; *Acc*: 56.21 (w=0.3)	36.24 (w=0.2)	37.34 (w=0.3)
	Imp + BLLIP	F_1: 45.80; *Acc*: **58.95** (w=0.2)	38.13 (w=0.1)	39.14 (w=0.4)
Gshare	*Imp + Exp*	F_1: 45.70; *Acc*: 57.17	37.84	38.10
	Imp + BLLIP	F_1: **47.80**; *Acc*: 57.39	**39.40**	**40.12**

Table 3: Performance of multi-class classification on PDTB and CoNLL-2016 in terms of accuracy (*Acc*) (%) and macro-averaged F_1 (%).

	Binary Classification (F_1)					Multi-class Classification (*Acc*)		
	Comp.	Cont.	Exp.	Exp+	Temp	PDTB (Four way)	CoNLL-Test(*Acc*)	CoNLL-Blind(*Acc*)
(Chen et al., 2016)	40.17	54.76	-	80.62	31.32	-	-	-
(Qin et al., 2016b)	**41.55**	57.32	71.50	80.96	35.43	-	-	-
(Liu and Li, 2016)	39.86	54.48	70.43	80.86	**38.84**	F_1: 46.29; *Acc*: **57.57**	-	-
(Wu et al., 2016)	-	-	-	-	-	F_1: 42.50; *Acc*: -	-	-
(Qin et al., 2016a)	38.67	54.91	-	80.66	32.76	-	-	-
(Liu et al., 2016b)	37.91	55.88	69.97	-	37.17	F_1: 44.98; *Acc*: 57.27	-	-
(Lan et al., 2013)	31.53	47.52	70.01	-	29.51	-	-	-
(Wang and Lan, 2016)	-	-	-	-	-	-	**40.91**	34.20
(Rutherford and Xue, 2016)	-	-	-	-	-	-	36.13	37.67
Our model	40.73	**58.96**	**72.47**	**81.36**	38.50	F_1: **47.80**; *Acc*: 57.39	39.40	**40.12**

Table 4: Comparison with the state-of-the-art systems reported on PDTB and CoNLL-2016, where - means N.A.

relevant to this work. (Liu et al., 2016b) presented a multi-task neural network, which considered information sharing between the main and auxiliary task. Different from their work, our work integrates the attention-based interaction between arguments and the multi-task based interaction between tasks into the final model. This is the main reason why our model achieves better performance in all types of relations, which shows the effectiveness of integrating gated mechanism into multi-task framework. Besides, (Liu and Li, 2016) used a complicated multi-level attention mechanism and the performance of our attention neural network in the single task is comparable to their results. Our multi-task attention model achieves better performance in most types with the aid of multi-task framework.

Besides, our previous work in (Lan et al., 2013) listed in Table 4, also presented a multi-task framework with traditional machine learning method to address implicit discourse recognition using BLLIP to obtain synthetic data. Clearly, under neural network-based multi-task framework, the attention and gated mechanism significantly improved the results and outperformed traditional machine learning method in all types of relations.

4.4 Effects of parameters w

Figure 5 shows the performance of four binary classification on four top level classes influenced by different share weights w in *Wshare* multi-task framework. We see that the best performance is achieved when we use a lower value of w. This

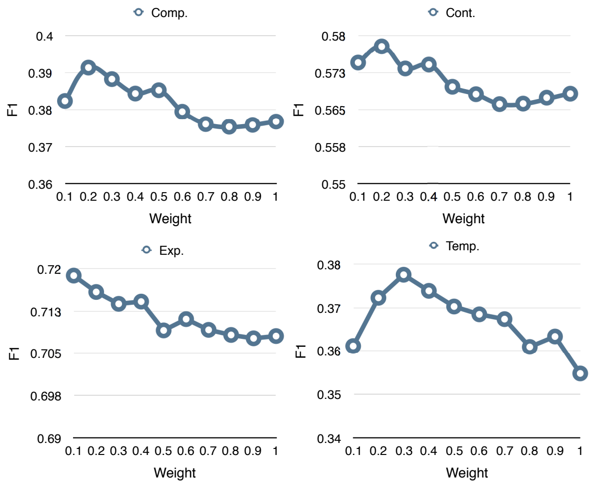


Figure 5: Results of top level implicit discourse relations in PDTB 2.0 with different weights w.

indicates that a low value of w can boost performance and reduce the negative influence brought by auxiliary task and enable our model to pay more attention to the main task.

5 Related Work

5.1 Implicit Discourse

With the release of PDTB 2.0, a number of studies performed discourse relation recognition on natural (i.e., genuine) discourse data with the use of traditional NLP techniques to extract linguistically informed features and traditional machine learning algorithms (Pitler et al., 2009; Lin et al., 2009; Wang et al., 2010; Braud and Denis, 2015; Fisher and Simmons, 2015).

Later, to make a full use of unlabelled data, several studies performed multi-task or unsupervised learning methods (Lan et al., 2013; Braud and Denis, 2015; Fisher and Simmons, 2015; Rutherford and Xue, 2015).

Recently, with the development of deep learning, researchers resorted to neural networks methods (Ji and Eisenstein, 2015; Zhang et al., 2015; Chen et al., 2016; Liu et al., 2016b; Qin et al., 2016a; Liu and Li, 2016; Braud and Denis, 2016; Wu et al., 2016).

5.2 Multi-task learning

Multi-task learning framework adopts traditional machine learning with human-selected effective knowledge and the shared part is integrated into the cost function to prefer the main task learning. (Collobert and Weston, 2008) proposed a multi-task neural network trained jointly on the relevant tasks using weight-sharing (sharing the word embeddings with tasks). (Liu et al., 2016a) proposed the multi-task neural network by modifying the recurrent neural network for text classification tasks. (Lan et al., 2013) present a multi-task learning based system which can effectively use synthetic data for implicit discourse relation recognition. (Wu et al., 2016) use bilingually-constrained synthetic implicit data for implicit discourse relation recognition a multi-task neural network. (Liu et al., 2016b) propose a convolutional neural network embedded multi-task learning system to improve the performance of implicit discourse identification.

5.3 Deep learning with Attention

Recently deep learning with attention has been widely adopted by NLP researchers. (Zhou et al., 2016) proposed an attention-based Bi-LSTM for relation classification. (Wang et al., 2016c) proposed an attention-based LSTM for aspect-level sentiment classification. (Tan et al., 2016) proposed a attentive LSTMs for Question Answer Matching. (Wang et al., 2016a) proposed an inner attention based RNN (add attention information before RNN hidden representation) for Answer Selection in QA. (Wang et al., 2016b) proposed multi-level attention CNNs for relation classification. (Yin et al., 2016) proposed an attentive convolutional neural network for QA.

6 Concluding Remarks

We present a novel multi-task attention-based neural network model for implicit discourse relationship representation and identification. Our method captures both the discourse relationships through interactions between discourse arguments and the complementary knowledge through interactions between annotated and unannotated data. The experimental results showed that our proposed model outperforms the state-of-the-art systems on two benchmark corpora.

Acknowledgments

This work is supported by grants from Science and Technology Commission of Shanghai Municipality (14DZ2260800 and 15ZR1410700), Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (ZF1213) and NSFC (61402175).

References

Chloé Braud and Pascal Denis. 2015. Comparing word representations for implicit discourse relation classification. In *Empirical Methods in Natural Language Processing (EMNLP 2015)*.

Chloé Braud and Pascal Denis. 2016. Learning connective-based word representations for implicit discourse relation identification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 203–213, Austin, Texas. Association for Computational Linguistics.

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, and X-uanjing Huang. 2016. Implicit discourse relation detection via a deep architecture with gated relevance network. In *Proceedings of ACL*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Robert Fisher and Reid Simmons. 2015. Spectral semi-supervised discourse relation classification. *Volume 2: Short Papers*.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association for Computational Linguistics*, 3:329–344.

Man Lan, Yu Xu, and Zheng-Yu Niu. 2013. Leveraging synthetic discourse data via multi-task learning for implicit discourse relation recognition. In *ACL (1)*, pages 476–485. Citeseer.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 343–351. Association for Computational Linguistics.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016a. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Yang Liu and Sujian Li. 2016. Recognizing implicit discourse relations via repeated reading: Neural networks with multi-level attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1233, Austin, Texas. Association for Computational Linguistics.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016b. Implicit discourse relation classification via a multi-task neural networks. *arXiv preprint arXiv:1603.02776*.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 683–691. Association for Computational Linguistics.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The Penn Discourse TreeBank 2.0. In *LREC*. Citeseer.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016a. Implicit discourse relation recognition with contextaware character-enhanced embeddings. In *the 26th International Conference on Computational Linguistics (COLING), Osaka, Japan, December*.

Lianhui Qin, Zhisong Zhang, and Hai Zhao. 2016b. A stacking gated neural architecture for implicit discourse relation classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), Austin, USA, November*.

Attapol Rutherford and Nianwen Xue. 2015. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *Proceedings of the NAACL-HLT*.

Attapol T Rutherford and Nianwen Xue. 2016. Robust non-explicit neural discourse parser in english and chinese. *ACL 2016*, page 55.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 464–473, Berlin, Germany. Association for Computational Linguistics.

Bingning Wang, Kang Liu, and Jun Zhao. 2016a. Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1288–1297, Berlin, Germany. Association for Computational Linguistics.

Jianxiang Wang and Man Lan. 2016. Two end-to-end shallow discourse parsers for english and chinese in conll-2016 shared task. *Proceedings of the CoNLL-16 shared task*, pages 33–40.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016b. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1298–1307, Berlin, Germany. Association for Computational Linguistics.

WenTing Wang, Jian Su, and Chew Lim Tan. 2010. Kernel based discourse relation recognition with temporal ordering information. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 710–719. Association for Computational Linguistics.

Yequan Wang, Minlie Huang, xiaoyan zhu, and Li Zhao. 2016c. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas. Association for Computational Linguistics.

Changxing Wu, xiaodong shi, Yidong Chen, Yanzhou Huang, and jinsong su. 2016. Bilingually-constrained synthetic data for implicit discourse relation recognition. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2306–2312, Austin, Texas. Association for Computational Linguistics.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1746–1756, Osaka, Japan. The COLING 2016 Organizing Committee.

Biao Zhang, Jinsong Su, Deyi Xiong, Yaojie Lu, Hong Duan, and Junfeng Yao. 2015. Shallow convolutional neural network for implicit discourse relation recognition. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2230–2235.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212, Berlin, Germany. Association for Computational Linguistics.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1507–1514, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chinese Zero Pronoun Resolution with Deep Memory Network

Qingyu Yin, Yu Zhang, Weinan Zhang, Ting Liu[*]
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{qyyin, yzhang, wnzhang, tliu}@ir.hit.edu.cn

Abstract

Existing approaches for Chinese zero pronoun resolution typically utilize only syntactical and lexical features while ignoring semantic information. The fundamental reason is that zero pronouns have no descriptive information, which brings difficulty in explicitly capturing their semantic similarities with antecedents. Meanwhile, representing zero pronouns is challenging since they are merely gaps that convey no actual content. In this paper, we address this issue by building a deep memory network that is capable of encoding zero pronouns into vector representations with information obtained from their contexts and potential antecedents. Consequently, our resolver takes advantage of semantic information by using these continuous distributed representations. Experiments on the OntoNotes 5.0 dataset show that the proposed memory network could substantially outperform the state-of-the-art systems in various experimental settings.

1 Introduction

A zero pronoun (ZP) is a gap in a sentence, which refers to an entity that supplies the necessary information for interpreting the gap (Zhao and Ng, 2007). A ZP can be either anaphoric if it corefers to one or more preceding noun phrases (antecedents) in the associated text, or non-anaphoric if there are no such noun phrases. Below is an example of ZPs and their antecedents, where "ϕ" denotes the ZP.

[警方] 表示 他们 自杀 的 可能性 很高，不过 ϕ_1 也 不 排除 ϕ_2 有 他杀 的 可能。

([The police] said that they are more likely to commit suicide, but ϕ_1 could not rule out ϕ_2 the possibility of homicide.)

In this example, the ZP "ϕ_1" is an anaphoric ZP that refers to the antecedent "警方/The police" while the ZP "ϕ_2" is non-anaphoric. Unlike overt pronouns, ZPs lack grammatical attributes such as gender and number that have been proven to be essential in pronoun resolution (Chen and Ng, 2014a), which makes ZP resolution a more challenging task than overt pronoun resolution.

Automatic Chinese ZP resolution is typically composed of two steps, i.e., anaphoric zero pronoun (AZP) identification that identifies whether a ZP is anaphoric; and AZP resolution, which determines antecedents for AZPs. For AZP identification, state-of-the-art resolvers use machine learning algorithms to build AZP classifiers in a supervised manner (Chen and Ng, 2013, 2016). For AZP resolution, literature approaches include unsupervised methods (Chen and Ng, 2014b, 2015), feature-based supervised models (Zhao and Ng, 2007; Kong and Zhou, 2010), and neural network models (Chen and Ng, 2016). Neural network models for AZP resolution are of growing interest for their capacity to learn task-specific representations without extensive feature engineering and to effectively exploit lexical information for ZPs and their candidate antecedents in a more scalable manner than feature-based models.

Despite these advantages, existing supervised approaches (Zhao and Ng, 2007; Chen and Ng, 2013, 2016) for AZP resolution typically utilize only syntactical and lexical information through features. They overlook semantic information that is regarded as an important factor in the resolution of common noun phrases (Ng, 2007). The fundamental reason is that ZPs have no descriptive information, which results in difficulty in calculating semantic similarities and relatedness scores

[*] Email corresponding.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1309–1318
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

between the ZPs and their antecedents. Therefore, the proper representations of ZPs are required so as to take advantage of semantic information when resolving ZPs. However, representing ZPs is challenging because they are merely gaps that convey no actual content.

One straightforward method to address this issue is to represent ZPs with supplemental information provided by some available components, such as contexts and candidate antecedents. Motivated by Chen and Ng (2016) who encode a ZP's lexical contexts by utilizing its preceding word and governing verb, we notice that a ZP's context can help to describe the ZP itself. As an example of its usefulness, given the sentence "ϕ taste spicy", people may resolve the ZP "ϕ" to the candidate antecedent "red peppers", but can hardly regard "my shoes" as its antecedent, because they naturally look at the ZP's context "taste spicy" to resolve it ("my shoes" cannot "taste spicy"). Meanwhile, considering that the antecedents of a ZP provide the necessary information for interpreting the gap (ZP), it is a natural way to express a ZP by its potential antecedents. However, only some subsets of candidate antecedents are needed to represent a ZP[1]. To achieve this goal, a desirable solution should be capable of explicitly capturing the importance of each candidate antecedent and using them to build up the representation for the ZP.

In this paper, inspired by the recent success of computational models with attention mechanism and explicit memory (Sukhbaatar et al., 2015; Tang et al., 2016; Kumar et al., 2015), we focus on AZP resolution, proposing the zero pronoun-specific memory network (**ZPMN**) that is competent for representing a ZP with information obtained from its contexts and candidate antecedents. These representations provide our system with an ability to take advantage of semantic information when resolving ZPs. Our **ZPMN** consists of multiple computational layers with shared parameters. With the underlying intuition that not all candidate antecedents are equally relevant for representing the ZP, we develop each computational layer as an attention-based model, which first learns the importance of each candidate antecedent and then utilizes this information to calculate the continuous distributed representation of the ZP. The attention weights over candidate antecedents with respect to the ZP's representation obtained by the last layer are regarded as the ZP coreference classification result. Given that every component is differentiable, the entire model could be efficiently trained end-to-end with gradient descent.

We evaluate our method on the Chinese portions of the OntoNotes 5.0 corpus by comparing with the baseline systems in different experimental settings. Results show that our approach significantly outperforms the baseline algorithms and achieves state-of-the-art performance.

2 Zero Pronoun-specific Memory Network

We describe our deep memory network approach for AZP resolution in this section. We first give an overview of our model and then describe its components. Finally, we present the training and initialization details.

2.1 An Overview of the Method

In this part, we present an overview of the zero pronoun-specific memory network (**ZPMN**) for AZP resolution. Given an AZP zp, we first extract a set of candidate antecedents. Following Chen and Ng (2016), we regard all and only those maximal or modifier noun phrases (NPs) that precede zp in the associated text and are at most two sentences away from it, to be its candidate antecedents. Suppose k candidate antecedents are extracted, our task is to determine the correct antecedent of zp from its candidate antecedent set $\mathcal{A}(zp) = \{c_1, c_2, ..., c_k\}$.

Specifically, these candidate antecedents are represented in form of vectors $\{v_{c_1}, v_{c_2}, ..., v_{c_k}\}$, which are stacked and regarded as the external memory $mem \in \mathbb{R}^{l \times k}$, where l is the dimension of v_c. Meanwhile, we represent each word as a continuous and real-valued vector, which is known as word embedding (Bengio et al., 2003). These word vectors can be randomly initialized, or be pre-trained from text corpus with learning algorithms (Mikolov et al., 2013; Pennington et al., 2014). In this work, we adopt the latter strategy since it can better exploit the semantics of words. All the word vectors are stacked in a word embedding matrix $L_w \in \mathbb{R}^{d \times |V|}$, where d is the dimension of the word vector and $|V|$ is the size of the word vocabulary. The embedding of word w is

[1] A common way to do this task is to first extract a set of candidate antecedents, and then select antecedents from the candidate set. Therefore, only those candidates who are possibly the correct antecedent of the given ZP are suitable for interpreting it.

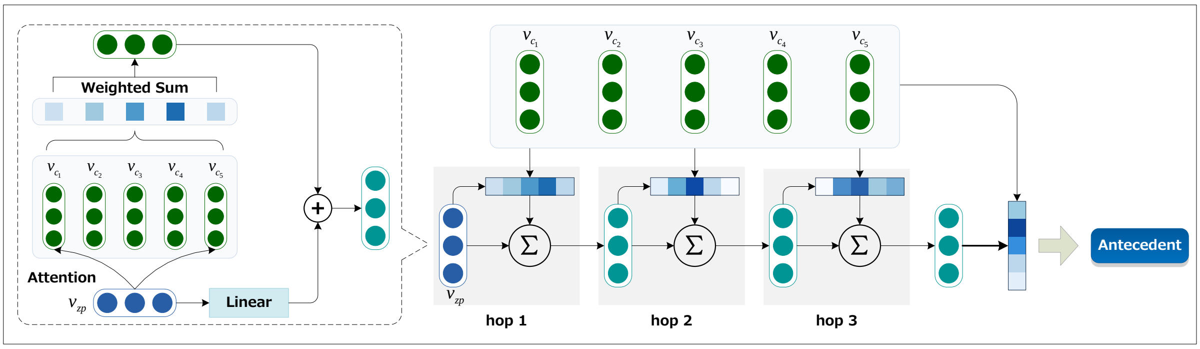


Figure 1: Illustration of the zero pronoun-specific memory network with three computational layers (hops). v_{zp} and v_c denote the vector representation of an AZP and its candidate antecedents. The left part in dashed box shows the details of the first hop.

notated as $e \in \mathbb{R}^{d \times 1}$, which is the column in L_w.

An illustration of **ZPMN** is given in Figure 1, which is inspired by the memory network utilized in question answering (Sukhbaatar et al., 2015). Our model consists of multiple computational layers, each of which contains an attention layer and a linear layer. First, we represent the AZP zp by utilizing its contextual information, that is, proposing the ZP-centered LSTM that encodes zp into its distributed vector representation (i.e. v_{zp} in Figure 1). We then regard v_{zp} as the initial representation of zp, and feed it as the input to the first computational layer (hop 1). In the first computational layer, we calculate the attention weight across the AZP for each candidate antecedent, by which our model adaptively selects important information from the external memory (candidate antecedents). The output of the attention layer and the linear transformation of v_{zp} are summed together as the input of to the next layer (hop 2).

We stack multiple hops by repeating the same process for multiple times in a similar manner. We call the abstractive information obtained from the external memory the "*key extension*" of the AZP. Note that the attention and linear layer parameters are shared in different hops. Regardless of the number of hops the model employs, they utilize the same number of parameters. Finally, after going through all the hops, we regard the attention weight of each candidate antecedent with respect to the AZP representation generated by the last hop as the probability that the candidate antecedent is the correct antecedent, and predict the highest-scoring (most probable) one to be the antecedent of the given AZP.

2.2 Modeling Zero Pronouns by Contexts

A vector representation of AZP is required when computing the **ZPMN**. As aforementioned, a ZP contains no actual content, it is therefore needed to employ some supplemental information to generate its initial representation. To achieve this goal, we develop the ZP-centered LSTM that encodes an AZP into a vector representation by utilizing its contextual information.

Admittedly, one efficient method to model a variable-length sequence of words (context words) is to utilize a recurrent neural network (Elman, 1991). A recurrent neural network (RNN) stores the sequence history in a real-valued history vector, which captures information of the whole sequence. LSTM (Hochreiter and Schmidhuber, 1997) is one of the classical variations of RNN that mitigate the gradient vanish problem of RNN. Assuming $x = \{x_1, x_2, ..., x_n\}$ is an input sequence, each time step t has an input x_t and a hidden state h_t. The internal mechanics of the LSTM is defined by:

$$i_t = \sigma(W^{(i)} \cdot [x_t; h_{t-1}] + b^{(i)}) \quad (1)$$

$$f_t = \sigma(W^{(f)} \cdot [x_t; h_{t-1}] + b^{(f)}) \quad (2)$$

$$o_t = \sigma(W^{(o)} \cdot [x_t; h_{t-1}] + b^{(o)}) \quad (3)$$

$$\tilde{C}_t = tanh(W^{(c)} \cdot [x_t; h_{t-1}] + b^{(c)}) \quad (4)$$

$$C_t = i_t \odot \tilde{C}_t + f_t \odot C_{t-1} \quad (5)$$

$$h_t = o_t \odot tanh(C_t) \quad (6)$$

where $\odot$ is an element-wise product and $W^{(i)}$, $b^{(i)}$, $W^{(f)}$, $b^{(f)}$, $W^{(o)}$, $b^{(o)}$, $W^{(c)}$, and $b^{(c)}$ are the parameters of the LSTM network.

Intuitively, the words near an AZP generally contain richer information to express it. To bet-

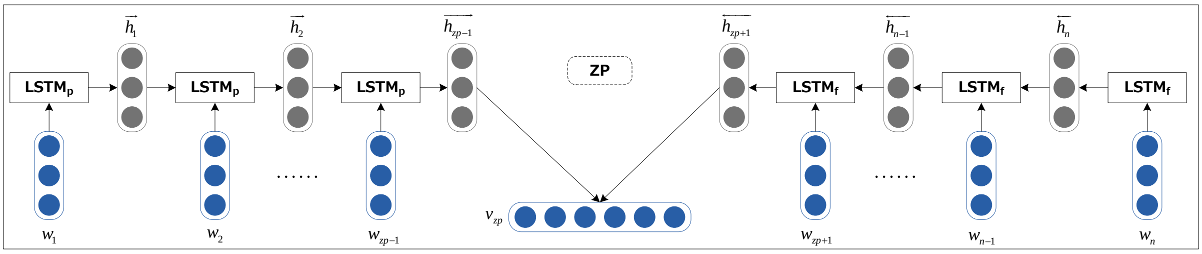


Figure 2: ZP-centered LSTM for encoding the AZP by its context words. w_i means the i-th word in the sentence, w_{zp-i} is the i-th last word before the ZP and w_{zp+i} is the i-th word behind the ZP.

ter utilize the information of words surrounding the AZP, on the basis of the traditional LSTM, we propose the ZP-centered LSTM to encode the AZPs. A graphical representation of this model is displayed in Figure 2. Specifically, the ZP-centered LSTM contains two standard LSTM neural networks, i.e., the LSTM$_p$ that encodes the preceding context of the AZP in a left-to-right manner, and the LSTM$_f$ that models the following context in the reverse direction. Ideally, the ZP-centered LSTM models the preceding and following contexts of the AZP separately, so that the words near the AZP are regarded as the last hidden units and could contribute more in representing the AZP. Afterward, we obtain the representation of the AZP by concatenating the last hidden vectors of LSTM$_p$ and LSTM$_f$, which summarizes the useful contextual information centered around the AZP. Averaging or summing the last hidden vectors of LSTM$_p$ and LSTM$_f$ could also be attempted as alternatives. We regard it as the initial vector representation of the AZP and feed it to the first computational layer to go through the remaining procedures of our system.

2.3 Generating the External Memory

We describe our method for generating the external memory in this subsection. For a given AZP, a set of noun phrases (NPs) is extracted as its candidate antecedents. Specifically, we generate the external memory by utilizing these candidate antecedents. One way to encode an NP candidate is to utilize its head word embedding (Chen and Ng, 2016). However, this method has a major drawback of not utilizing contextual information that is essential for representing a phrase. Besides, some approaches (Socher et al., 2013; Sun et al., 2015) encode a phrase by utilizing the average word embedding it contains. We argue that such an averaging operation simply treats all the words in a phrase equally, which is inaccurate because some words might be more informative than others.

A helpful property of LSTM is that it could keep useful history information in the memory cell by exploiting input, output and forget gates to decide how to utilize and update the memory of previous information. Given a sequence of words $\{w_1, w_2, ..., w_n\}$, previous research (Sutskever et al., 2014) utilizes the last hidden vector of LSTM to represent the information of the whole sequence. For word w_t in a sequence, its corresponding hidden vector h_t can capture useful information before and including w_t.

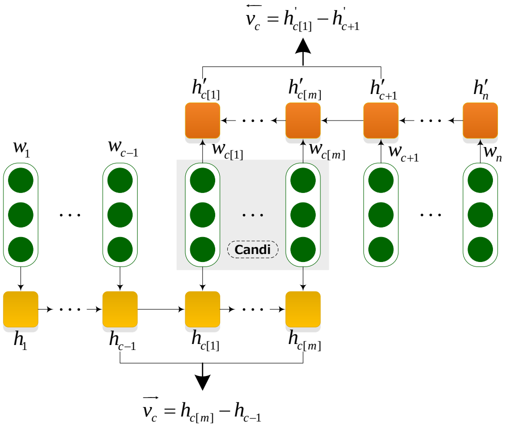


Figure 3: Illustration for modeling a candidate antecedent through its context and content words. *Candi* represents the candidate antecedent. Suppose the candidate antecedent contains m words, $w_{c[j]}$ denotes its j-th word. w_i is the i-th word in the sentence, and $w_{c+1(-1)}$ is the word appears immediately after (before) the candidate antecedent.

Inspired by this, we propose a novel method to produce representations of the candidate antecedents by utilizing both their contexts and content words. Specifically, we use the subtraction between LSTM hidden vectors to encode the candidate antecedents, as illustrated in Figure 3. Given a candidate antecedent c with m words, two standard LSTM neural networks are employed for encoding c in the forward and backward direction, respectively. For the forward LSTM, we extract a sequence of words related with c in a left-to-right manner, i.e., $\{w_1, w_2, ..., w_{c-1}, w_{c[1]}, ..., w_{c[m]}\}$. Subsequently, the forward vector representation of c can be calculated as $\overrightarrow{v_c} = h_{c[m]} - h_{c-1}$, where $h_{c[m]}$ and h_{c-1} indicate the hidden vectors of the forward LSTM corresponding to $w_{c[m]}$ and w_{c-1}, respectively. Meanwhile, the backward LSTM models a sequence of words that are extracted in the reverse direction, that is, $\{w_n, w_{n-1}, ..., w_{c+1}, w_{c[m]}, ..., w_{c[1]}\}$. We then perform the similar operation, computing the backward representation of c as $\overleftarrow{v_c} = h'_{c[1]} - h'_{c+1}$, where $h'_{c[1]}$ and h'_{c+1} indicate the hidden vectors of the backward LSTM corresponding to $w_{c[1]}$ and w_{c+1}. Finally, we concatenate these two vectors together as the ultimate vector representation of c, $v_c = \overrightarrow{v_c} || \overleftarrow{v_c}$.

This method enables our model to encode a candidate antecedent by the information both outside and inside the phrase, which provides our model a strong ability to access to sentence-level information when modeling the candidate antecedents. In this manner, we generate the vector representations of the candidate antecedents, and regard them as the external memory, i.e., $mem = \{v_{c_1}, v_{c_2}, ..., v_{c_k}\}$.

2.4 Attention Mechanism

In this part, we introduce our attention mechanism. This strategy has been widely used in many nature language processing tasks, such as factoid question answering (Hermann et al., 2015), entailment (Rocktäschel et al., 2015) and disfluency detection (Wang et al., 2016). The basic idea of attention mechanism is that it assigns a weight/importance to each lower position when computing an upper-level representation (Bahdanau et al., 2015). With the underlying intuition that not all candidate antecedents are equally relevant for representing the AZP, we employ the attention mechanism as to dynamically align the more informative candidate antecedents from the external memory, $mem = \{v_{c_1}, v_{c_2}, ..., v_{c_k}\}$ with regard to the given AZP, and use them to build up the representation of the AZP.

As shown in Chen and Ng (2016), traditional hand-crafted features are crucial for the resolver's success since they capture the syntactic, positional and other relationships between an AZP and its candidate antecedents. Therefore, to evaluate the importance of each candidate antecedent in a comprehensive manner, following Chen and Ng (2016) who encode hand-crafted features as inputs to their network, we integrate a set of features that are utilized in Chen and Ng (2016), in the form of vector ($v^{(feature)}$) into our attention model. For each multi-valued feature, we convert it into a corresponding set of binary-valued features[2].

Specifically, for the t-th candidate antecedent in the memory, v_{c_t}, taking the vector representation of the AZP v_{zp} and the corresponding feature vector $v_t^{(feature)}$ as inputs, we compute the attention score as $\alpha_t = G(v_{c_t}, v_{zp}, v_t^{(feature)})$. The scoring function G is defined by:

$$s_t = tanh(W^{(att)} \cdot [v_{c_t}; v_{zp}; v_t^{(feature)}] + b^{(att)}) \tag{7}$$

$$\alpha_t = \frac{exp(s_t)}{\sum_{t'=1}^{k} exp(s_{t'})} \tag{8}$$

where $W^{(att)}$ and $b^{(att)}$ are the attention parameters and k indicates the number of candidate antecedents. After obtaining the attention scores for all the candidate antecedents $\{a_1, a_2, ..., a_k\}$, our attention layer outputs a continuous vector vec that is computed as the weighted sum of each piece of memory in mem:

$$vec = \sum_{i=1}^{k} \alpha_i v_{c_i} \tag{9}$$

2.5 Training Details

We initialize our word embeddings with 100 dimensional ones produced by the $word2vec$ toolkit (Mikolov et al., 2013) on the Chinese portion of the training data from the OntoNotes 5.0 corpus. We randomly initialize the parameters from a uniform distribution $U(-0.03, 0.03)$ and minimize the training objective using stochastic gradient descent with learning rate equals to 0.01. In addition, to regularize the network, we apply L2 regularization to the network weights and dropout with a rate of 0.5 on the output of each hidden layer.

[2] If one feature has k different values, we will convert it into k binary features.

The model is trained in a supervised manner by minimizing the cross-entropy error of ZP coreference classification. Suppose the training set contains N AZPs $\{zp_1, zp_2, ..., zp_N\}$. Let $\mathcal{A}(zp_i)$ denote the set of candidate antecedents of an AZP zp_i, and $P(c|zp_i)$ represents the probability of predicting candidate c as the antecedent of zp_i (i.e., the attention weight of candidate antecedent c with respect to the AZP representation generated by the last hop), the loss is given by:

$$loss = -\sum_{i=1}^{N} \sum_{c \in \mathcal{A}(zp_i)} \delta(zp_i, c) log(P(c|zp_i)) \tag{10}$$

where $\delta(zp, c)$ is 1 or 0, indicating whether zp and c are coreferent.

3 Experiments

3.1 Experimental Setup

Datasets: Following Chen and Ng (2016, 2015), we run experiments on the Chinese portion of the OntoNotes Release 5.0 dataset[3] used in the CoNLL 2012 Shared Task (Pradhan et al., 2012). The dataset consists of three parts, i.e., a training set, a development set and a test set. Since only the training set and the development set contain ZP coreference annotations, we train our model on the training set and utilize the development set for testing purposes. Meanwhile, we reserve 20% of the training set as a held-out development set for tuning the hyperparameters of our network. The same experimental data setting is utilized in the baseline system (Chen and Ng, 2016). Table 1 shows the statistics of our corpus. Besides, documents in the datasets come from six sources, i.e., broadcast news (BN), newswires (NW), broadcast conversations (BC), telephone conversations (TC), web blogs (WB) and magazines (MZ).

	Documents	Sentences	Words	AZPs
Training	1,391	36,487	756K	12,111
Test	172	6,083	110K	1,713

Table 1: Statistics on the training and test corpus.

Evaluation metrics: Same as previous studies on Chinese ZP resolution (Zhao and Ng, 2007; Chen and Ng, 2016), we use three metrics to evaluate the quality of our model: recall, precision and F-score (denoted as R, P and F, respectively).

Experimental settings: We employ three Chinese ZP resolution systems as our baselines, i.e., Zhao and Ng (2007); Chen and Ng (2015, 2016). Consistent with Chen and Ng (2015, 2016), three experimental settings are designed to evaluate our approach. In Setting 1, we directly employ the gold syntactic parse trees and gold AZPs that are obtained from the OntoNotes dataset. In Setting 2, we utilize gold syntactic parse trees and system (automatically identified) AZPs[4]. In Setting 3, we employ system AZP and system syntactic parse trees that obtained through the Berkeley parser[5], which is the state-of-the-art parsing model.

3.2 Experimental Results

Table 2 shows the experimental results of the baseline systems and our model on entire test set. Our approach is abbreviated to ZPMN (k), where k indicates the number of hops. The best methods in each of the three experimental settings are in **bold** text. From Table 2, we can observe that our approach outperforms all previous baseline systems by a substantial margin. Meanwhile, among all our models from single hop to six hops, using more computational layers could generally lead to better performance. The best performance is achieved by the model with six hops under experimental Setting 1 and 2, and with four hops in experimental Setting 3. Furthermore, the **ZPMN** (with six hops) significantly outperforms the state-of-the-art baseline system (Chen and Ng, 2016) under three experimental settings by 2.7%, 2.7%, and 3.9% in terms of overall F-score[6], respectively. In all words, our model is an extremely strong performer and substantially outperforms baseline methods, which demonstrate the efficiency of the proposed zero pronoun-specific memory network.

It is well accepted that computational models that are composed of multiple processing layers could learn representations of data with multiple levels of abstraction (LeCun et al., 2015). In our approach, multiple computation layers allow the model to learn representations of AZPs with multiple levels of abstraction generated by candidate antecedents. Each layer/hop retrieves important candidate antecedents, and transforms the repre-

[3] http://catalog.ldc.upenn.edu/LDC2013T19

[4] In this study, we adopt the learning-based method utilized in (Chen and Ng, 2016) to identify system AZPs, including the location and identification of AZPs.

[5] https://github.com/slavpetrov/berkeleyparser

[6] All significance tests are paired t-tests, with $p < 0.05$.

	Setting 1 Gold Parse + Gold AZP			Setting 2 Gold Parse + System AZP			Setting 3 System Parse + System AZP		
	R	P	F	R	P	F	R	P	F
Zhao and Ng (2007)	41.5	41.5	41.5	22.4	24.4	23.3	12.7	14.2	13.4
Chen and Ng (2015)	50.0	50.4	50.2	35.7	26.2	30.3	19.6	15.5	17.3
Chen and Ng (2016)	51.8	52.5	52.2	**39.6**	27.0	32.1	21.9	15.8	18.4
ZPMN (1)	53.0	53.3	53.1	37.9	30.0	33.4	27.8	17.4	21.4
ZPMN (2)	53.7	54.0	53.9	38.8	30.6	34.0	28.1	18.2	22.1
ZPMN (3)	53.9	54.2	54.1	38.6	30.4	34.2	28.2	17.7	21.7
ZPMN (4)	54.4	54.7	54.5	39.0	30.7	34.3	**29.3**	**18.5**	**22.7**
ZPMN (5)	54.1	54.4	54.3	38.8	30.6	34.2	28.6	17.8	22.0
ZPMN (6)	**54.8**	**55.1**	**54.9**	39.4	**31.1**	**34.8**	28.9	18.2	22.3

Table 2: Experimental results on the test data. ZPMN represents the proposed zero pronoun-specific memory network model, and the number beside ZPMN in each row denotes the number of hops.

	Setting 1: Gold Parse + Gold AZP						Setting 2: Gold Parse + System AZP						Setting 3: System Parse + System AZP					
	Baseline			**ZPMN**			Baseline			**ZPMN**			Baseline			**ZPMN**		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
NW	48.8	48.8	48.8	48.8	48.8	**48.8**	34.5	26.4	29.9	39.5	34.3	**36.7**	11.9	12.8	12.3	21.0	19.9	**20.5**
MZ	41.4	41.6	41.5	46.3	46.3	**46.3**	34.0	22.4	27.0	34.6	35.0	**34.8**	9.3	7.3	8.2	17.1	15.7	**16.4**
WB	56.3	56.3	56.3	59.8	59.8	**59.8**	44.7	25.1	32.2	41.2	28.7	**33.8**	23.9	16.1	19.2	31.3	17.6	**22.6**
BN	55.4	55.4	55.4	58.2	58.6	**58.4**	36.9	31.9	34.2	43.8	30.0	**35.6**	22.1	23.2	22.6	35.1	20.7	**26.1**
BC	50.4	51.3	50.8	52.9	53.6	**53.2**	37.6	25.6	30.5	35.6	29.4	**32.2**	21.2	14.6	17.3	25.6	15.6	**19.4**
TC	51.9	54.2	53.1	54.8	54.8	**54.8**	46.3	29.0	**35.6**	36.9	32.9	34.8	31.4	15.9	21.1	33.2	21.0	**25.8**

Table 3: Experimental results on each source of test data. The strongest F-score in each row is in **bold**.

sentation at previous level into a representation at a higher, slightly more abstract level. We regard this representation as the "*key extension*" of the AZP, by which our model learns to encode the AZP in an efficient manner.

For per-source results, we conduct experiments by comparing the **ZPMN** (with six hops) with the state-of-the-art baseline system (Chen and Ng, 2016) on six sources of test data, as shown in Table 3. The rows in Table 3 are the experimental results from different sources under the three experimental settings. In experimental Settings 1 and 3, **ZPMN** improves results further across all the six sources of data. Under experimental Setting 2, our model outperforms the baseline system in five of the six sources of data, only slightly underperforms in source TC. All these prove that our approach achieves a considerable improvement in Chinese ZP resolution.

Moreover, to evaluate the effectiveness of our methods for modeling the AZP and candidate antecedents proposed in Section 2.2 and 2.3, we compare with three models that are all simplified versions of the **ZPMN**, namely, **ZPContextFree** where an AZP is initially represented by its governing verb and preceding word; **AntContentAvg** where the candidate antecedents are encoded by their averaged content word embeddings; and **AntContentHead** where each candidate antecedent is represented by the embedding of its head word. To make comparison as fair as possible, we keep the other parts of these models unchanged from the **ZPMN** with six computational layers (hop 6). To minimize the external influence, we run experiments under experimental Setting 1 (gold parse and gold AZPs). Table 4 shows the results.

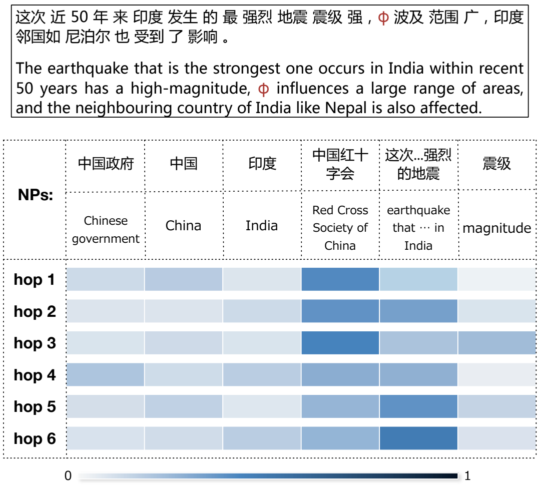
	R	P	F
ZPContextFree	53.5	53.8	53.6
AntContentAvg	52.6	52.9	52.7
AntContentHead	53.8	54.1	53.9
ZPMN (hop 6)	54.8	55.1	**54.9**

Table 4: Experimental results of different models.

With an intuition that contexts of an AZP provide more sufficient information than only a few specific of words in expressing the AZP, the performance of **ZPContextFree** is unsurprisingly worse than that of the **ZPMN**, which reflects the effects of the ZP-centered LSTM proposed to generate the initial representation for the AZP. In addition, the performance of **AntContentAvg** is relatively low. We attribute this to the model assigning the same importance to all the content words in a phrase, which causes difficulty for the model to capture informative words in a candidate antecedent. Meanwhile, **AntContent-**

Head only models limited information when encoding candidate antecedents, thereby underperforms the **ZPMN** whose external memory contains sentence-level information both outside and inside the candidate antecedents. These demonstrate the utility of the method for modeling candidate antecedents.

3.3 Attention Model Visualization



这次 近 50 年 来 印度 发生 的 最 强烈 地震 震级 强，φ 波及 范围 广，印度 邻国如 尼泊尔 也 受到 了 影响 。

The earthquake that is the strongest one occurs in India within recent 50 years has a high-magnitude, φ influences a large range of areas, and the neighbouring country of India like Nepal is also affected.

Figure 4: Example of attention weights in different hops. ZP is denoted as ϕ. The rows show the attention weights of candidates in each hop. Darker color means higher weight.

To obtain a better understanding of our deep memory network, we visualize the attention weights of the **ZPMN**, as is shown in Figure 4. We can observe that in the first three hops, the fourth candidate "中国红十字会/Red Cross Society of China" gains a higher attention weight than the others. Nevertheless, in hop 5 and 6, the attention weight of "这次...强烈的地震/the earthquake that ... in India" increases and the model finally predicts it correctly as the antecedent. This case illustrates the effects of multiple hops.

4 Related Work

4.1 Zero Pronoun Resolution

Chinese zero pronoun resolution. Early studies utilize heuristic rules to resolve ZPs in Chinese (Converse, 2006; Yeh and Chen, 2007). More recently, supervised approaches have been vastly explored. Zhao and Ng (2007) first present a machine learning approach to identify and resolve ZPs. By employing the J48 decision tree algorithm, various kinds of features are integrated into their model. Kong and Zhou (2010) develop a kernel-based approach, employing context-sensitive convolution tree kernels to model syntactic information. Chen and Ng (2013) further extend the study of Zhao and Ng (2007) by proposing several novel features and introducing the coreference links between ZPs. Despite the effectiveness of feature engineering, it is labor intensive and highly relies on annotated corpus. To handle these weaknesses, Chen and Ng (2014b) propose an unsupervised method. They first recover each ZP into ten overt pronouns and then apply a ranking model to rank the antecedents. Chen and Ng (2015) propose an end-to-end unsupervised probabilistic model, utilizing a salience model to capture discourse information. In recent years, Chen and Ng (2016) develop a deep neural network approach to learn useful task-specific representations and effectively exploit lexical features through word embeddings. Different from previous studies, in this work, we propose a novel memory network to perform the task. By encoding ZPs and candidate antecedents through the composition of texts based on the representation of words, our model benefits from the semantic information when resolving the ZPs.

Zero pronoun resolution for other languages. There have been various studies on ZP resolution for other languages besides Chinese. Ferrández and Peral (2000) propose a set of hand-crafted rules for resolving ZPs in Spanish texts. Recently, supervised approaches have been widely exploited for ZP resolution in Korean (Han, 2006), Italian (Iida and Poesio, 2011) and Japanese (Isozaki and Hirao, 2003; Iida et al., 2006, 2007; Imamura et al., 2009; Sasano and Kurohashi, 2011; Iida and Poesio, 2011; Iida et al., 2015). Iida et al. (2016) propose a multi-column convolutional neural network for Japanese intra-sentential subject zero anaphora resolution, where both the surface word sequence and dependency tree of a target sentence are exploited as clues in their model.

4.2 Attention and Memory Network

Attention mechanisms have been widely used in many studies and have achieved promising performances on a variety of NLP tasks (Rocktäschel et al., 2015; Rush et al., 2015; Liu et al., 2017). Recently, the memory network has been proposed and applied to question answering task (Weston et al., 2014), which is defined to have four components: input (I), generalization (G), output (O) and response (R). After then, memory networks have been adopted in many other NLP tasks, such as aspect sentiment classification (Tang et al., 2016), dialog systems (Dodge et al., 2015), and information extraction (Xiaocheng et al., 2017).

5 Conclusion

In this study, we propose a novel zero pronoun-specific memory network that is capable of encoding zero pronouns into the vector representations with supplemental information obtained from their contexts and candidate antecedents. Consequently, these continuous distributed vectors provide our model with an ability to take advantage of the semantic information when resolving zero pronouns. We evaluate our method on the Chinese portion of OntoNotes 5.0 dataset and report substantial improvements over the state-of-the-art systems in various experimental settings.

Acknowledgments

We greatly thank Yiming Cui and Xuxiang Wang for their tremendously helpful discussions. We also thank the anonymous reviewers for their valuable comments. This work was supported by the National High Technology Development 863 Program of China (No.2015AA015407), National Natural Science Foundation of China (No.61472105 and No.61472107).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Yoshua Bengio, Rejean Ducharme, and Pascal Vincent. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Chen Chen and Vincent Ng. 2013. Chinese zero pronoun resolution: Some recent advances. In *EMNLP*, pages 1360–1365.

Chen Chen and Vincent Ng. 2014a. Chinese overt pronoun resolution: A bilingual approach. In *AAAI*, pages 1615–1621.

Chen Chen and Vincent Ng. 2014b. Chinese zero pronoun resolution: An unsupervised approach combining ranking and integer linear programming. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Chen Chen and Vincent Ng. 2015. Chinese zero pronoun resolution: A joint unsupervised discourse-aware model rivaling state-of-the-art resolvers. In *Proceedings of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, page 320.

Chen Chen and Vincent Ng. 2016. Chinese zero pronoun resolution with deep neural networks. In *Proceedings of the 54rd Annual Meeting of the ACL.*

Susan P Converse. 2006. Pronominal anaphora resolution in chinese.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2015. Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931*.

Jeffrey L Elman. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2-3):195–225.

Antonio Ferrández and Jesús Peral. 2000. A computational approach to zero-pronouns in spanish. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 166–172. Association for Computational Linguistics.

Na-Rae Han. 2006. *Korean zero pronouns: analysis and resolution*. Ph.D. thesis, Citeseer.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

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

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploiting syntactic patterns as clues in zero-anaphora resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 625–632. Association for Computational Linguistics.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007. Zero-anaphora resolution by learning rich syntactic pattern features. *ACM Transactions on Asian Language Information Processing (TALIP)*, 6(4):1.

Ryu Iida and Massimo Poesio. 2011. A cross-lingual ilp solution to zero anaphora resolution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 804–813. Association for Computational Linguistics.

Ryu Iida, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, and Julien Kloetzer. 2015. Intra-sentential zero anaphora resolution using subject sharing recognition. *Proceedings of EMNLP'15*, pages 2179–2189.

Ryu Iida, Kentaro Torisawa, Jong-Hoon Oh, Canasai Kruengkrai, and Julien Kloetzer. 2016. Intra-sentential subject zero anaphora resolution using multi-column convolutional neural network. In *Proceedings of EMNLP*.

Kenji Imamura, Kuniko Saito, and Tomoko Izumi. 2009. Discriminative approach to predicate-argument structure analysis with zero-anaphora resolution. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 85–88. Association for Computational Linguistics.

Hideki Isozaki and Tsutomu Hirao. 2003. Japanese zero pronoun resolution based on ranking rules and machine learning. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 184–191. Association for Computational Linguistics.

Fang Kong and Guodong Zhou. 2010. A tree kernel-based unified framework for chinese zero anaphora resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 882–891. Association for Computational Linguistics.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

Ting Liu, Yiming Cui, Qingyu Yin, Shijin Wang, Weinan Zhang, and Guoping Hu. 2017. Effective deep memory networks for distant supervised relation extraction. In *ACL*.

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

Vincent Ng. 2007. Semantic class induction and coreference resolution. In *AcL*, pages 536–543.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40. Association for Computational Linguistics.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to japanese zero anaphora resolution with large-scale lexicalized case frames. In *IJCNLP*, pages 758–766.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *IJCAI*, pages 1333–1339.

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

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *EMNLP*.

Shaolei Wang, Wanxiang Che, and Ting Liu. 2016. A neural attention model for disfluency detection. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 278–287, Osaka, Japan. The COLING 2016 Organizing Committee.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Feng Xiaocheng, Guo Jiang, Qin Bing, Liu Ting, and Liu Yongjie. 2017. Effective deep memory networks for distant supervised relation extraction. In *IJCAI*.

Ching-Long Yeh and Yi-Chun Chen. 2007. Zero anaphora resolution in chinese with shallow parsing. *Journal of Chinese Language and Computing*, 17(1):41–56.

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of chinese zero pronouns: A machine learning approach. In *EMNLP-CoNLL*, volume 2007, pages 541–550.

How much progress have we made on RST discourse parsing? A replication study of recent results on the RST-DT

Mathieu Morey and **Philippe Muller** and **Nicholas Asher**
IRIT, Université Toulouse & CNRS, Univ. Paul Sabatier
{mathieu.morey, philippe.muller, nicholas.asher}@irit.fr

Abstract

This article evaluates purported progress over the past years in RST discourse parsing. Several studies report a relative error reduction of 24 to 51% on all metrics that authors attribute to the introduction of distributed representations of discourse units. We replicate the standard evaluation of 9 parsers, 5 of which use distributed representations, from 8 studies published between 2013 and 2017, using their predictions on the test set of the RST-DT. Our main finding is that most recently reported increases in RST discourse parser performance are an artefact of differences in implementations of the evaluation procedure. We evaluate all these parsers with the standard Parseval procedure to provide a more accurate picture of the actual RST discourse parsers performance in standard evaluation settings. Under this more stringent procedure, the gains attributable to distributed representations represent at most a 16% relative error reduction on fully-labelled structures.

1 Introduction

While several theories of discourse structure for text exist, discourse parsing work has largely concentrated on Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) and the RST Discourse Treebank (RST-DT) (Carlson et al., 2003), which is the largest corpus of texts annotated with full discourse structures. The RST-DT, annotated in the style of RST, consists of 385 news articles from the Penn Treebank, split into a training and test sets of 347 and 38 documents.The standard evaluation procedure for RST discourse parsing, RST-Parseval, proposed by Marcu (2000), adapts the Parseval procedure for syntactic parsing (Black et al., 1991). RST-Parseval computes scores on discourse structures with no label (S for Span) or labelled with nuclearity (N), relation (R) or both (F for Full). The semantic nature of discourse relations makes discourse parsing a difficult task. However, the recent introduction of distributed representations of discourse units has seemingly led to significant improvements, with a claimed relative error reduction of 51% on fully labelled structures. As part of a broader study of methods and evaluation metrics for discourse parsing, we collected predictions from nine RST discourse parsers and reimplemented RST-Parseval. In section 2, we present these RST parsers and report their published scores on RST-Parseval. In section 3, we replicate their evaluation and show that most of the heterogeneity in performance across RST parsers arises from differences in their evaluation procedures. In section 4, we replace RST-Parseval with the standard Parseval procedure and obtain a more accurate picture of the actual performance of RST parsers.

2 A sample of RST discourse parsers

Almost all RST discourse parsers are evaluated on the test section of the RST-DT using manually segmented Elementary Discourse Units (EDUs). We contacted by email the main or corresponding author of each recently (2013–2017) published, text-level RST discourse parser evaluated in this setting and asked the authors to provide us with the predictions they used in their study or a procedure that would enable us to reproduce identical or at least similar predictions. When our attempts were unsuccessful we tried to reproduce similar predictions from published materiel (source code, binaries, model). We managed to obtain or reproduce predictions for 9 parsers from 8 studies. The first parser, denoted **HHN16 HILDA**, is a reimplementation (Hayashi et al., 2016) of the classic, bottom-up, greedy HILDA parser with a linear SVM model (Hernault et al., 2010) ; this parser serves as a reference point to evaluate the progress made by more recent parsers. **SHV15**

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1319–1324
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

D is a variant of the HILDA parser with different models (perceptron for attachment of discourse units, logistic regression for relation labelling) and a slightly different feature set adapted to use predicted syntactic dependency trees (Surdeanu et al., 2015). **JCN15 1S-1S** is a two stage (sentence-then document-level) CKY chart parser with Dynamic Conditional Random Field (DCRF) models, in its 1 sentence - 1 subtree (1S-1S) variant that builds a document-level RST tree on top of sentence-level subtrees built for each sentence independently (Joty et al., 2013, 2015). **FH14** ***g*****CRF** is a two stage (sentence- then document-level) bottom-up, greedy parser with linear-chain CRF models (Feng and Hirst, 2014). We use the version of the parser available on the author's webpage, that lacks post-editing and contextual features. **BPS16** is a sequence-to-sequence parser, heuristically constrained to build trees, with a hierarchical neural network model (hierarchical bi-LSTM) (Braud et al., 2016). **LLC16** is a CKY chart parser with a hierarchical neural network model (attention-based hierarchical bi-LSTM) (Li et al., 2016). **BCS17 mono, BCS17 cross+dev** are two variants of a transition-based parser that uses a feed-forward neural network model (Braud et al., 2017). **JE14 DPLP** is a shift-reduce parser that uses an SVM model (Ji and Eisenstein, 2014). We use predictions provided by the author, from an improved, unpublished version of the parser.

The first four parsers (HHN16 HILDA, SHV15 D, JCN15 1S-1S, FH14 *g*CRF) use, as features, only localist representations of the input and parsing state, i.e. surface-form and syntactic information: length of discourse units (DUs), distance between DUs, n-grams of words and POS tags, relations of syntactic dominance between DUs... The last five parsers (BPS16, LLC16, BCS17 mono and cross+dev, JE14 DPLP concat) build distributed representations of DUs, complemented with a subset of localist representations.

The authors used various implementations of RST-Parseval, but all applied a right-heavy binarization procedure to the reference RST trees: Each node of arity greater than 2 is replaced with a right-branching cascade of binary nodes. In the publications, the tables of results provide a unique score for labeled structures, corresponding to either the R or F metric, with no explicit distinction. The F_1 scores published in the literature for the parsers in our sample are reported in Table 1, where an en-dash (–) indicates missing scores. We also report the scores of human agreement, computed and reported by Joty (2015), over the doubly annotated subset of the RST-DT consisting of 53 documents (48 from train, 5 from test).

parser	**S**	**N**	**R** or **F**
HHN16 HILDA	82.6	66.6	54.2
SHV15 D	–	–	55.2
JCN15 1S-1S	82.6	68.3	55.8
FH14 *g*CRF	84.9	69.9	57.2
BPS16	83.6	69.8	55.1
LLC16	**85.8**	71.1	58.9
BCS17 mono	85.0	72.3	60.1
BCS17 cross+dev	85.1	**73.1**	61.4
JE14 DPLP concat	82.1	71.1	**61.6**
human	88.7	77.7	65.8

Table 1: Published F_1 scores.

The parsers in the second group seem to perform markedly better than the parsers in the first group, especially on the hardest subtasks of predicting (partly or fully) labelled structures (N and R or F). Collectively, the parsers in the second group claim absolute improvements over the parsers in the first group by 0.9, 3.2 and 4.2 points, corresponding to a relative error reduction of 24% on S, 41% on N and 51% on R or F, compared to human agreement. While discourse parsing is a difficult, semantic task with relatively little annotated training data, authors attribute these significant gains to the capacity of distributed representations to capture latent semantic information and generalize over a long tail of alternative surface forms. As a preliminary step towards probing these claims, we replicated the evaluation of these parsers' predictions.

3 Evaluation

We collected or reproduced predictions from each parser and replicated the evaluation procedure [1]. The predictions came in various formats: bracketed strings as in the RST-DT, lists of span descriptions, trees or lists of attachment decisions. We wrote custom functions to load and normalize the predictions from each parser into RST trees. While we favor evaluating against the original,

[1] The source code and material are available at https://github.com/irit-melodi/rst-eval

non binarized reference RST trees, we conformed in this replicative study to the de facto standard in the RST parsing literature: We transformed the reference RST trees into right-branching binary trees and used these binary trees as reference in all our evaluation procedures. We also examined the source code from the evaluation procedures provided by the authors to determine whether the published scores corresponded to the R or F metric. In so doing we noticed a potentially important discrepancy in the various implementations of the RST-Parseval procedure: the implementations used to evaluate the parsers in the first group compute **micro-averaged** F_1 scores, as is standard practice in the syntactic parsing community, whereas the implementations used to evaluate the parsers in the second group compute **macro-averaged** F_1 scores across documents. The micro-averaged F_1 score is computed globally over the predicted and reference spans from all documents ; the macro-averaged F_1 score across documents is the average of F_1 scores computed independently for each document.

We implemented both strategies and report the corresponding scores in two separate tables. Parsers originally evaluated with micro-averaging scores are in the top half of each table, parsers originally evaluated with macro-averaged scores in the bottom half. An asterisk (*) marks parsers for which we reproduced predictions using code and material made available by the authors, although the experimental settings are not guaranteed to match exactly those from the original study. A double asterisk (**) marks a parser for which we used predictions generated by the author using an improved, unpublished version of the parser posterior to the original study. Lines with no asterisk in Tables 2 to 4 correspond to parsers whose authors sent us their original predictions. Replicated scores expected to match scores in Table 1 are underlined.

parser	S	N	R	F
HHN16 HILDA	82.6	66.6	54.6	54.3
SHV15 D *	82.6	67.1	55.4	54.9
JCN15 1S-1S	82.6	68.3	55.8	55.4
FH14 *g*CRF *	**84.3**	**69.4**	56.9	56.2
BPS16	79.7	63.6	47.7	47.5
LLC16	82.2	66.5	51.4	50.6
BCS17 mono	81.0	67.7	55.7	55.3
BCS17 cross+dev	81.3	68.1	56.3	56.0
JE14 DPLP **	82.0	68.2	**57.8**	**57.6**
human	88.3	77.3	65.4	64.7

Table 2: Micro-averaged F_1 scores.

Table 2 contains the micro-averaged F_1 scores on each metric (S, N, R, F). As expected, parsers in the first group obtain micro-averaged scores equal or close to their published scores reported in Table 1. More strikingly, the micro-averaged scores for the parsers in the second group are much lower than their published scores [2] and most of their claimed advantages over the parsers in the first group has vanished. On S and N, parsers in the second group do not improve over parsers in the first group ; on R and F the best parser in the second group provides an absolute improvement of 0.9 and 1.4 points. This improvement corresponds to a relative error reduction of 11% on R and 16% on F, much lower than the 51% claimed in the literature. [3]

parser	S	N	R	F
HHN16 HILDA	85.9	72.1	60.0	59.4
SHV15 D *	85.1	71.1	59.8	59.1
JCN15 1S-1S	85.7	73.0	60.9	60.2
FH14 *g*CRF *	**87.0**	**74.1**	61.1	60.5
BPS16	83.6	69.8	55.4	55.1
LLC16	85.4	70.8	58.4	57.6
BCS17 mono	85.0	72.3	60.8	60.1
BCS17 cross+dev	85.1	73.1	61.6	61.4
JE14 DPLP **	85.0	71.6	**62.0**	**61.9**
human	89.6	78.3	66.7	65.3

Table 3: Macro-averaged F_1 scores.

Table 3 contains the macro-averaged F_1 scores. Parsers in the first group obtain macro-averaged scores markedly higher than the micro-averaged scores from Table 2. Parsers in the second group obtain macro-averaged scores that are equal or close to the published scores reported in Table 1, which confirms our analysis of the source code of their evaluation procedures. The global picture on

[2] The milder decrease of the DPLP scores, especially on S, is directly attributable to improvements in the latest, unpublished version of the parser.

[3] Our replicated scores for human agreement are 0.4 points lower than those published on S, N, R, possibly due to different approaches in handling divergences in EDU segmentation on the doubly annotated subset of documents.

macro-averaged scores is consistent with that on micro-averaged scores: On S and N, parsers in the second group do not improve over parsers the first group and the best parser brings an absolute improvement of 0.9 and 1.4 points on R and F. On each metric, the two lowest scores are obtained by parsers from the second group.

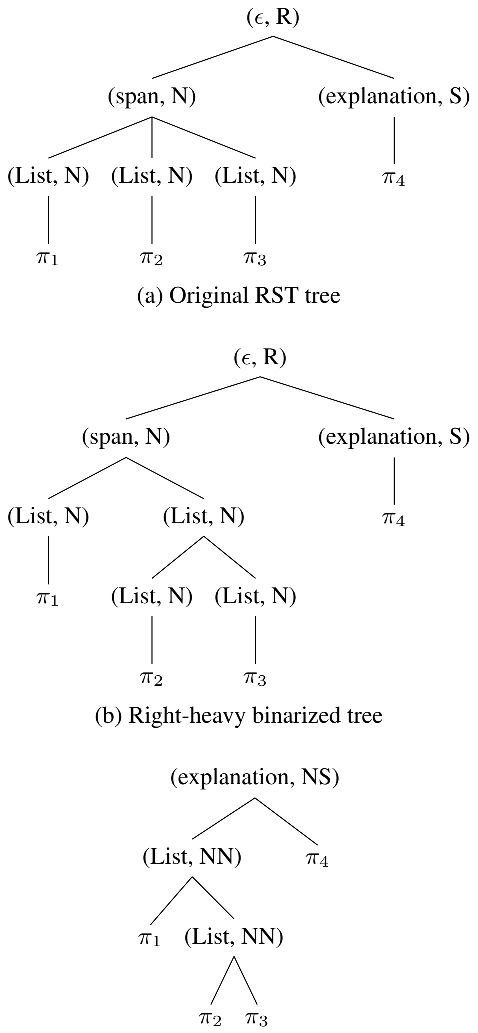
To sum up, parsers in the first group have identical scores in Tables 1 and 2, except for slight differences between our evaluation procedure and the authors', or between the predictions used in our evaluation compared to the original study. The second group of parsers have identical scores in Tables 1 and 3, modulo the same factors. The (exactly or nearly) matching entries between Tables 1, 2 and 3, underlined in Tables 2 and 3, are evidence of the two averaging strategies (micro in Table 2, macro in Table 3) used by the authors in their publications (Table 1). A comparison between Tables 2 and 3 reveals that the averaging strategy similarly affects both groups of parsers. As a result, the performance level among recent RST discourse parsers is much more homogeneous than the situation depicted in the literature. The distributed representations of DUs computed and used in JE14 DPLP (Ji and Eisenstein, 2014) and possibly BCS17 cross+dev (Braud et al., 2017) plausibly capture semantic information that helps with predicting discourse relations and structure, but the current experimental results do not provide a similarly strong support for BPS16 (Braud et al., 2016), LLC16 (Li et al., 2016) and BCS17 mono (Braud et al., 2017).

More generally, it is important that authors compute and report scores that accord with standard practice, unless duly motivated. The standard practice in syntactic parsing is to report micro-averaged scores for overall performance, often complemented with macro-averaged scores over classes to gain valuable insight into the average performance of parsers across labels, especially infrequent ones. Early work in RST discourse parsing follows this practice, reporting micro-averaged scores for global performance, plus distinct scores for each relation class or macro-averaged scores over all relation classes (Hernault et al., 2010; Feng and Hirst, 2014). The latter should not be confused with the scores published for BPS16, LLC16, BCS17 (mono, cross+dev) and JE14 DPLP, which are macro-averaged over documents.

4 Elements for a fairer evaluation

RST-Parseval crucially relies on an encoding of RST trees into constituency trees such that the rhetorical relation names are placed on the children nodes, and the nuclei of mononuclear relations are conventionally labelled SPAN. RST-Parseval resembles the original Parseval, except it considers a larger set of nodes to collect all nuclearity and relation labels in this encoding: the root node (whose label and nuclearity are fixed by convention) is excluded and the leaves, the EDUs, are included. On the one hand, RST-Parseval can handle discourse units of arity greater than 2, in particular those consisting of a nucleus independently modified by two satellites through distinct mononuclear relations. This avoids introducing discourse units that were not part of the original annotation, which a preliminary binarization of trees would have induced. On the other hand, RST-Parseval considers approximately twice as many nodes as the original Parseval would on binarized trees (at most $2n - 2$ nodes for n EDUs, compared to $n - 1$ attachments in a binary tree), and the relation labels of most nuclei are redundant with the nuclearity of a node and its sister (SPAN for a nucleus whose sisters are satellites, and the same label as its sisters for a nucleus whose sisters are nuclei). Both aspects artificially raise the level of agreement between RST trees, especially when using manual EDU segmentation.

However, all the parsers in our sample except (Sagae, 2009; Heilman and Sagae, 2015) predict binary trees over manually segmented EDUs and evaluate them against right-heavy binarized reference trees. In this setting, Marcu's encoding of RST trees RST-Parseval are no longer motivated. We can thus revert to using the standard Parseval procedure on a representation of binary RST trees where each internal node is a labelled attachment decision to obtain a more accurate evaluation of RST parser performance. Figure 1 represents (a) an original RST tree using Marcu's encoding, (b) its right-heavy binarized version, (c) the tree of labelled attachment decisions for the right-heavy binarized tree. To the best of our knowledge, we are the first to explicitly use an evaluation procedure for RST parsing closer to the original Parseval for syntax, although the trees of labelled attachment decisions we use directly correspond to the trees built by many RST parsers, eg. shift-reduce parsers. Table 4 provides the micro-averaged F_1



(a) Original RST tree

(b) Right-heavy binarized tree

(c) Labelled attachment decisions for the right-heavy binarized tree

Figure 1: Original RST tree, right-heavy binarization and labelled attachment decisions

scores for the parsers in our sample, using Parseval.

Parseval is more stringent than RST-Parseval, with the best system obtaining 46.3 on fully labelled structures (F). Parsers in the first group are competitive with parsers in the second group, outperforming them on S and to a lesser extent on N. Parsers in the second group reduce relative error by 8% on R and 16% on F, much lower than the published figures in the literature.

5 Conclusion

We replicated standard evaluation procedures in RST discourse parsing for 9 parsers and showed that most gains reported in recent publications are an artefact of implicit differences in evaluation procedures. We also showed how to use the standard Parseval procedure instead of Marcu's adaptation RST-Parseval, which artificially raises scores. Overall, the recent gains attributable to distributed representations represent at most a relative error reduction of 16%. Our study reveals an urgent need for the RST discourse parsing community to re-examine and standardize their evaluation procedures.

parser	S	N	R	F
HHN16 HILDA	65.1	54.6	44.7	44.1
SHV15 D *	65.3	54.2	45.1	44.2
JCN15 1S-1S	65.1	55.5	45.1	44.3
FH14 *g*CRF *	**68.6**	**55.9**	45.8	44.6
BPS16	59.5	47.2	34.7	34.3
LLC16	64.5	54.0	38.1	36.6
BCS17 mono	61.9	53.4	44.5	44.0
BCS17 cross+dev	62.7	54.5	45.5	45.1
JE14 DPLP **	64.1	54.2	**46.8**	**46.3**
human	78.7	66.8	57.1	55.0

Table 4: Micro-averaged F_1 scores on labelled attachment decisions (original Parseval).

Acknowledgments

This research was supported by ERC Advanced Grant n. 269427. We thank the authors of the parsers who made their predictions and evaluation scripts available, and the reviewers for helpful comments.

References

E. Black, S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. Procedure for quantitatively comparing the syntactic coverage of english grammars. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, pages 306–311. Association for Computational Linguistics.

Chloé Braud, Maximin Coavoux, and Anders Søgaard. 2017. Cross-lingual rst discourse parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 292–304, Valencia, Spain. Association for Computational Linguistics.

Chloé Braud, Barbara Plank, and Anders Søgaard.

2016. Multi-view and multi-task training of rst discourse parsers. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1903–1913, Osaka, Japan. The COLING 2016 Organizing Committee.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2003. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In Jan van Kuppevelt and Ronnie Smith, editors, *Current Directions in Discourse and Dialogue*, pages 85–112. Kluwer Academic Publishers.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 511–521, Baltimore, Maryland. Association for Computational Linguistics.

Katsuhiko Hayashi, Tsutomu Hirao, and Masaaki Nagata. 2016. Empirical comparison of dependency conversions for rst discourse trees. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 128–136, Los Angeles. Association for Computational Linguistics.

Michael Heilman and Kenji Sagae. 2015. Fast Rhetorical Structure Theory Discourse Parsing. *arXiv preprint*, arXiv:1505.02425.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010. HILDA: A Discourse Parser Using Support Vector Machine Classification. *Dialogue and Discourse*, 1(3):1–33.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24, Baltimore, Maryland. Association for Computational Linguistics.

Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. 2015. Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41(3):385–435.

Shafiq R Joty, Giuseppe Carenini, Raymond T Ng, and Yashar Mehdad. 2013. Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. In *ACL (1)*, pages 486–496.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Discourse parsing with attention-based hierarchical neural networks. In *EMNLP*, pages 362–371.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Towards a Functional Theory of Text Organization. *Text*, 8(3):243–281.

Daniel Marcu. 2000. *The theory and practice of discourse parsing and summarization*. MIT press.

Kenji Sagae. 2009. Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In *Proceedings of the 11th International Conference on Parsing Technologies*, IWPT '09, pages 81–84, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mihai Surdeanu, Thomas Hicks, and Marco A Valenzuela-Escárcega. 2015. Two practical rhetorical structure theory parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 1–5.

What is it? Disambiguating the different readings of the pronoun 'it'

Sharid Loáiciga
Uppsala University
Dept. of Linguistics & Philology
Uppsala, Sweden
sharid.loaiciga@lingfil.uu.se

Liane Guillou
University of Edinburgh
School of Informatics
Scotland, United Kingdom
lguillou@inf.ed.ac.uk

Christian Hardmeier
Uppsala University
Dept. of Linguistics & Philology
Uppsala, Sweden
christian.hardmeier@lingfil.uu.se

Abstract

In this paper, we address the problem of predicting one of three functions for the English pronoun 'it': anaphoric, event reference or pleonastic. This disambiguation is valuable in the context of machine translation and coreference resolution. We present experiments using a MAXENT classifier trained on gold-standard data and self-training experiments of an RNN trained on silver-standard data, annotated using the MAXENT classifier. Lastly, we report on an analysis of the strengths of these two models.

1 Introduction

We address the problem of disambiguating the English pronoun 'it', which may function as a pleonastic, anaphoric, or event reference pronoun. As an *anaphoric* pronoun, 'it' corefers with a noun phrase (called the *antecedent*), as in example (1):

(1) I have a bicycle. It is red.

Pleonastic pronouns, in contrast, do not refer to anything but are required to fill the subject position in many languages, including English, French and German:

(2) It is raining.

Event reference pronouns are anaphoric, but instead of referring to a noun phrase, they refer to a verb, verb phrase, clause or even an entire sentence, as in example (3):

(3) He lost his job. It came as a total surprise.

We propose the identification of the three usage types of *it*, namely anaphoric, event reference, and pleonastic, with a single system. We present several classification experiments which rely on information from the current and previous sentences, as well as on the output of external tools.

2 Related Work

Due to its difficulty, proposals for the identification and the subsequent resolution of abstract anaphora (i.e., event reference) are scarce (Eckert and Strube, 2000; Byron, 2002; Navarretta, 2004; Müller, 2007). The automatic detection of instances of pleonastic 'it', on the other hand, has been addressed by the non-referential 'it' detector NADA (Bergsma and Yarowsky, 2011), and also in the context of several coreference resolution systems, including the Stanford sieve-based coreference resolution system (Lee et al., 2011).

The coreference resolution task focuses on the resolution of nominal anaphoric pronouns, de facto grouping our event and pleonastic categories together and discarding both of them. The coreference resolution task can be seen as a two-step problem: *mention* identification followed by *antecedent* identification. Identifying instances of pleonastic 'it' typically takes place in the mention identification step. The recognition of event reference 'it' is, however, to our knowledge not currently included in any such systems, although from a linguistic point of view, event instances are also referential (Boyd et al., 2005). As suggested by Lee et al., (2016), it would be advantageous to incorporate event reference resolution in the second step.

In the context of machine translation, work by Le Nagard and Koehn (2010); Novák et al. (2013); Guillou (2015) and Loáiciga et al. (2016) have also considered disambiguating the function of the pronoun 'it' in the interest of improving pronoun translation into different languages.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1325–1331
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

3 Disambiguating ‘it’

3.1 Labeled Data

The ParCor corpus (Guillou et al., 2014) and *DiscoMT2015.test* dataset (Hardmeier et al., 2016) were used as gold-standard data. Under the ParCor annotation scheme, which was used to annotate both corpora, pronouns are manually labeled according to their function: anaphoric, event reference, pleonastic, etc. For all instances of ‘it’ in the corpora, we extracted the sentence-internal position of the pronoun, the sentence itself, and the two previous sentences. All examples were shuffled before the corpus was divided, ensuring a balanced distribution of the classes (Table 1).

The pronouns ‘this’ and ‘that’, when used as event reference pronouns, may often be used interchangeably with the pronoun ‘it’ (Guillou, 2016). We therefore automatically substituted all instances of event reference ‘this’ and ‘that’ with ‘it’ to increase the number of training examples.

Data set	Event	Anaphoric	Pleonastic	Total
Training	504	779	221	1,504
Dev	157	252	92	501
Test	169	270	62	501
Total	830	1,301	375	2,506

Table 1: Distribution of classes in the data.

3.2 Baselines

We provide two different baselines (MC and LM Baseline in Table 2). The first is a setting in which all instances are assigned to the majority class *it-anaphoric*. The second baseline system is a 3-gram language model built using KenLM (Heafield, 2011) and trained on a modified version of the annotated corpus in which every instance of ‘it’ is concatenated with its function (e.g. ‘it-event’). At test time, the ‘it’ position is filled with each of the three it-function labels in turn, the language model is queried, and the highest scoring option is chosen.

3.3 Features

We designed features to capture not only the token context, but also the syntactic and semantic context preceding the pronouns and, where appropriate, their antecedents/referents, as well as the pronoun head. We used the output of the POS tagger and dependency parser of Bohnet et al. (2013)[1], and of the TreeTagger lemmatizer (Schmid, 1994) to extract the following information for each training example:

Token context (tok) **1.** Previous three tokens and next two tokens. This includes words, punctuation and the tokens in the previous sentence when the ‘it’ occupies the first position of the current sentence. **2.** Lemmas of the next two tokens.

Pronoun head (hea) **3.** Head word and its lemma. Most of the time the head word is a verb. **4.** If the head verb is copular, we include its complement head and not the verb itself (for the verbs *be*, *appear*, *seem*, *look*, *sound*, *smell*, *taste*, *feel*, *become* and *get*). **5.** Whether the head word takes a ‘that’ complement (verbs only). **6.** Tense of head word (verbs only), computed as described by Loáiciga et al. (2014).

Syntactic context (syn) **7.** Whether a ‘that’ complement appears in the previous sentence. **8.** Closest NP head to the left and to the right. **9.** Presence or absence of extraposed sentential subjects as in ‘*So it’s difficult to attack malaria from inside malarious societies, [...].* **10.** Closest adjective to the right.

Semantic context (sem) **11.** VerbNet selectional restrictions of the verb. VerbNet (Kipper et al., 2008) specifies 36 types of argument that verbs can take. We limited ourselves to the values of *abstract*, *concrete* and *unknown*. **12.** Likelihood of head word taking an event subject (verbs only). An estimate of the likelihood of a verb taking a event subject was computed over the Annotated English Gigaword v.5 corpus (Napoles et al., 2012). We considered two cases favouring event subjects that may be identified by exploiting the parse annotation of the Gigaword corpus. The first case is when the subject is a gerund and the second case is composed of ‘this’ pronoun subjects. **13.** Non-referential probability assigned to the instance of ‘it’ by NADA (Bergsma and Yarowsky, 2011).

3.4 MaxEnt

The MaxEnt classifier is trained using the Stanford Maximum Entropy package (Manning and Klein, 2003) with all of the features described above. We also experimented with other features and options. For features 1 and 2, a window

[1] We used the pre-trained models for English that are available online https://code.google.com/p/mate-tools/downloads/list

	Dev-set				Test-set			
MC Baseline	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
it-anaphoric	0.539	1	0.700	(252/501)	0.503	1	0.669	(270/501)
				0.503				0.539
LM Baseline	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
it-anaphoric	0.613	0.290	0.394	(166/501)	0.732	0.263	0.387	(163/501)
it-pleonastic	0.169	0.523	0.255	0.331	0.139	0.694	0.231	0.325
it-event	0.459	0.287	0.353		0.521	0.290	0.373	
MaxEnt	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
it-anaphoric	0.685	**0.758**	**0.719**	(326/501)	0.716	**0.756**	**0.735**	(344/501)
it-pleonastic	**0.884**	**0.543**	**0.633**	0.651	**0.750**	0.726	**0.738**	0.687
it-event	**0.545**	0.541	**0.543**		**0.564**	0.521	0.542	
RNN-Gold	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
it-anaphoric	0.544	0.560	0.552	(221/501)	0.595	0.659	0.626	(250/501)
it-pleonastic	0.274	0.217	0.242	0.441	0.177	0.177	0.177	0.499
it-event	0.355	0.382	0.368		0.436	0.361	0.394	
RNN-Silver	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
it-anaphoric	0.661	0.611	0.635	(286/501)	0.706	0.552	0.620	(286/501)
it-pleonastic	0.725	0.402	0.517	0.571	0.542	0.516	0.529	0.571
it-event	0.438	0.605	0.508		0.455	0.621	0.525	
RNN-Combined	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
it-anaphoric	**0.697**	0.492	0.577	(280/501)	**0.794**	0.530	0.636	(315/501)
it-pleonastic	0.633	**0.543**	0.585	0.559	0.582	**0.742**	0.652	0.629
it-event	0.434	**0.675**	0.529		0.520	**0.746**	**0.613**	

Table 2: Comparison of baselines and classification results.

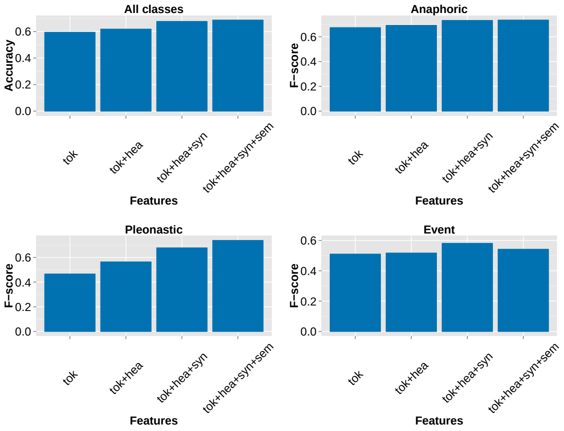


Figure 1: Feature ablation – MaxEnt system.

of three tokens showed a degradation in performance. For feature 8, adding one of the 26 WordNet (Princeton University, 2010) types of nouns had no effect. The feature combination of noun and adjectives to the left or right also had no effect. Feature ablation tests revealed that while combining all features is beneficial for the prediction of the anaphoric and pleonastic classes, the same is not true for the event class. In particular, the inclusion of semantic features, which we designed as indicators of event*ness*, appears to be harmful (Figure 1).

3.5 Unlabeled Data

Given the small size of the gold-standard data, and with the aim of gaining insight from unstructured and unseen data, we used the MaxEnt classifier to label additional data from the pronoun prediction shared task at WMT16 (Guillou et al., 2016). This new *silver-standard* training corpus comprises 1,101,922 sentences taken from the Europarl (3,752,440 sentences), News (344,805 sentences) and TED talks (380,072 sentences) sections of the shared task training data.

3.6 RNN

Our second system is a bidirectional recurrent neural network (RNN) which reads the context words and then makes a decision based on the representations that it builds. Concretely, it consists on word-level embeddings of size 90, two layers of Gated

REFERENCE RELATIONSHIP	MAXENT	RNN-COMBINED
(1) NP antecedent in previous 2 sentences *e.g. The infectious disease that's killed more humans than any other is* ***malaria****. It's carried in the bites of infected mosquitos, and* ***it****'s probably our oldest scourge.*	***(191/248)** **0.770**	(136/248) 0.548
(2) VP antecedent in previous 2 sentences *e.g. And there's hope in this next section, of this brain section of somebody else with M.S., because what it illustrates is, amazingly, the brain can* ***repair itself****. It just doesn't do* ***it*** *well enough.*	(25/38) 0.658	**(27/38)** **0.711**
(3) NP or VP antecedent further away in the text (not in snippet) *e.g. It has spread. It has more ways to evade attack than we know.* ***It****'s a shape-shifter, for one thing.*	(28/47) 0.596	(28/47) 0.596
(4) Sentential or clausal antecedent e.g. ***Pension systems have a hugely important economic and social role and are affected by a great variety of factors. It*** *has been reflected in EU policy on pensions, which has become increasingly comprehensive over the years.*	(52/88) 0.591	***(66/88)** **0.750**
(5) Pleonastic constructions *e.g. And* ***it*** *seemed to me that there were three levels of acceptance that needed to take place.*	(43/59) 0.729	(42/59) 0.728
(6) Ambiguous between event and anaphoric *e.g. Today, multimedia is a desktop or living room experience, because the apparatus is so clunky .* ***It*** *will change dramatically with small, bright, thin, high-resolution displays.*	(3/12) 0.250	**(7/12)** **0.583**
(7) Ambiguous between event and pleonastic *e.g. I did some research on how much it cost, and I just became a bit obsessed with transportation systems. And* ***it*** *began the idea of an automated car.*	**(2/5)** **0.400**	(1/5) 0.200
(8) Annotation errors *e.g. Youth unemployment is particularly worrying in* ***it*** *context, as the lost opportunity for jobless young people to develop professional skills is likely to translate into lower productivity and lower earnings over a longer period of time.*	(0/4) –	(0/4) –

Table 3: Accuracy scores of the systems in different portions of the test-set. For each category, we test whether MAXENT is better or worse than RNN-COMBINED. A * indicates significance at $p < 0.001$ using McNemar's χ^2 test.

Recurrent Units (GRUs) of size 90 as well, and a final softmax layer to make the predictions. The network uses a context window of 50 tokens both to the left and right of the 'it' to be predicted. The features described above are also fed to the network in the form of one-hot vectors. The system uses the *adam* optimizer and the categorical cross-entropy loss function. We chose this architecture following the example of Luotolahti et al. (2016), who built a system for the related task of cross-lingual pronoun prediction.

4 Discussion

We report all of the results in Table 2. MAXENT and RNN-GOLD are trained on the gold-standard data only. RNN-SILVER is trained on the silver-standard data (annotated using the MAXENT classifier). RNN-COMBINED is trained on both the silver-standard and gold-standard data.

The MAXENT and RNN models show improvements, albeit small for the it-event class, over the baseline systems. Since they are trained on the same gold-standard data, one would expect RNN-GOLD to perform similarly to MAXENT. However, in the case of the RNN-gold, the 50 tokens window may actually not have enough words to be filled with, because the gold-standard data is composed of the sentence with the it-pronoun and the three previous sentences, which in addition tend to be short. For the RNN-SILVER system this is not a problem, since the sentences of interest have not been taken out of their original context, fully exploiting the RNN capacity to learn the entirety of the context window they are presented with, even if the data is noisy. As expected, RNN-COMBINED performs better than RNN-GOLD and RNN-SILVER. Although it does not perform overwhelmingly better than MAXENT, there are gains in precision for the it-anaphoric class, and in recall for the it-pleonastic and it-event classes, suggesting that the system benefits from the inclusion of gold-standard data.

With the two-fold goal of gaining a better understanding of the difficulties of the task and strengths of the systems, we re-classified the test set in a stratified manner. We present the systems with seven scenarios reflecting the different types of reference relationships observed in the corpora (Table 3). Our scenarios are exhaustive, thus some only have few examples. The analysis reveals that the MAXENT is a better choice for nominal reference (case (1), mostly *it-anaphoric*) whereas the RNN-COMBINED system is better at identifying difficult antecedents such as cases (4) and (6). RNN-COMBINED performs slightly better at detecting verbal antecedents, case (2), while both systems perform similarly at learning pleonastic instances (5) or when the antecedent is not in the snippet (3). Finally, we found 4 instances of annotation errors (8). These correspond to some of the automatically substituted cases of 'this'/'that' with 'it', for which the 'this'/'that' should not have been marked as a pronoun by the human annotator in the first place. Case (8) is not taken into account in the evaluation.

Taking the complete test set, we found that the MAXENT system performs better than the RNN-COMBINED system in absolute terms ($\chi^2 = 50.8891, p < 0.001$), but this is because case (1) is the most frequent one, which is also the case the MAXENT system is strongest at.

5 Conclusions and Future Work

We have shown that distinguishing between nominal anaphoric and event reference realizations of 'it' is a complex task. Our results are promising, but there is room for improvement. The self-training experiment demonstrated the benefit of combining gold-standard and silver-standard data.

We also found that the RNN-COMBINED system is better at handling difficult and ambiguous referring relationships, while the MAXENT performed better for the nominal anaphoric case, when the antecedent is close. Since the two models have different strengths, in future work we plan to enrich the training data with re-training instances from the silver data where the two systems agree, in order to reduce the amount of noise, following the example of Jiang et al. (2016).

Ultimately, we aim towards integrating the it-prediction system within a full machine translation pipeline and a coreference resolution system. In the first case, the different translations of pronoun 'it' can be constrained according to their function. In the second case, the performance of a coreference resolution system vs a modified version using the three-way distinction can be measured.

Acknowledgments

This work was supported by the Swedish Research Council under project 2012-916 *Discourse-Oriented Statistical Machine Translation*. We used computing resources on the Abel cluster, owned by the University of Oslo and the Norwegian metacenter for High Performance Computing (NOTUR) and on the Taito-CSC cluster provided through the Nordic Language Processing Laboratory (NLPL) (www.nlpl.eu). SL was also supported by the Swiss National Science Foundation under grant no. P1GEP1_161877.

References

Shane Bergsma and David Yarowsky. 2011. NADA: A robust system for non-referential pronoun detection. In Iris Hendrickx, Sobha Lalitha Devi, António Branco, and Ruslan Mitkov, editors, *Anaphora Processing and Applications: 8th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC)*, Lecture Notes in Artificial Intelligence, pages 12–23. Springer, Faro, Portugal.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajič. 2013. Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, 1:415–428.

Adriane Boyd, Whitney Gegg-Harrison, and Donna K. Byron. 2005. Identifying non-referential *it*: a machine learning approach incorporating linguistically motivated patterns. In *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, pages 40–47, Ann Arbor, Michigan. Association for Computational Linguistics.

Donna K. Byron. 2002. Resolving pronominal reference to abstract entities. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, ACL 2002, pages 80–87, Philadelphia. Association for Computational Linguistics.

Miriam Eckert and Michael Strube. 2000. Dialogue acts, synchronising units and anaphora resolution. *Journal of Semantics*, 17(1):51–89.

Liane Guillou. 2015. Automatic post-editing for the DiscoMT pronoun translation task. In *Proceedings of the Second Workshop on Discourse in Machine Translation*, DiscoMT 2015, pages 65–71, Lisbon, Portugal. Association for Computational Linguistics.

Liane Guillou. 2016. *Incorporating Pronoun Function into Statistical Machine Translation*. Ph.D. thesis, University of Edinburgh, Scotland, UK.

Liane Guillou, Christian Hardmeier, Preslav Nakov, Sara Stymne, Jörg Tiedemann, Yannick Versley, Mauro Cettolo, Bonnie Webber, and Andrei Popescu-Belis. 2016. Findings of the 2016 WMT shared task on cross-lingual pronoun prediction. In *Proceedings of the First Conference on Machine Translation*, WMT16, pages 525–542, Berlin, Germany. Association for Computational Linguistics.

Liane Guillou, Christian Hardmeier, Aaron Smith, Jörg Tiedemann, and Bonnie Webber. 2014. ParCor 1.0: A parallel pronoun-coreference corpus to support statistical MT. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, LREC 2014, pages 3191–3198, Reykjavik, Iceland. European Language Resources Association (ELRA).

Christian Hardmeier, Jörg Tiedemann, Preslav Nakov, Sara Stymne, and Yannick Versely. 2016. DiscoMT 2015 Shared Task on Pronoun Translation. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT11, pages 187–197, Edinburgh, UK. Association for Computational Linguistics.

Kailang Jiang, Giuseppe Carenini, and Raymond Ng. 2016. Training data enrichment for infrequent discourse relations. In *Proceedings the 26th International Conference on Computational Linguistics: Technical Papers*, COLING 2016, pages 2603–2614, Osaka, Japan. The COLING 2016 Organizing Committee.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large scale classification of English verbs. *Language Resources and Evaluation*, 42(1):21–40.

Ronan Le Nagard and Philipp Koehn. 2010. Aiding pronoun translation with co-reference resolution. In *Proceedings of the Joint 5th Workshop on Statistical Machine Translation*, WMT10, pages 258–267, Uppsala, Sweden. Association for Computational Linguistics.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, CONLL 2011, pages 28–34, Portland, Oregon. Association for Computational Linguistics.

Timothy Lee, Alex Lutz, and Jinho D. Choi. 2016. QA-It: classifying non-referential it for question answer pairs. In *Proceedings of the ACL 2016 Student Research Workshop*, pages 132–137, Berlin, Germany. Association for Computational Linguistics.

Sharid Loáiciga, Liane Guillou, and Christian Hardmeier. 2016. It-disambiguation and source-aware language models for cross-lingual pronoun prediction. In *Proceedings of the First Conference on Machine Translation*, WMT16, pages 581–588, Berlin, Germany. Association for Computational Linguistics.

Sharid Loáiciga, Thomas Meyer, and Andrei Popescu-Belis. 2014. English-French verb phrase alignment in Europarl for tense translation modeling. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, LREC 2014, pages 674–681, Reykjavik, Iceland. European Language Resources Association (ELRA).

Juhani Luotolahti, Jenna Kanerva, and Filip Ginter. 2016. Cross-lingual pronoun prediction with deep recurrent neural networks. In *Proceedings of the First Conference on Machine Translation*, WMT16, pages 596–601, Berlin, Germany. Association for Computational Linguistics.

Christopher Manning and Dan Klein. 2003. MaxEnt models, conditional estimation, and optimization without magic. Tutorial at HLT-NAACL and 41st ACL conferences.

Christoph Müller. 2007. Resolving it, this, and that in unrestricted multi-party dialog. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, ACL 2007, pages 816–823, Prague, Czech Republic. Association for Computational Linguistics.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX, pages 95–100, Montreal, Canada. Association for Computational Linguistics.

Costanza Navarretta. 2004. Resolving individual and abstract anaphora in texts and dialogues. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING 2004, pages 233–239, Geneva, Switzerland. Association for Computational Linguistics.

Michal Novák, Anna Nedoluzhko, and Zdeněk Žabokrtský. 2013. Translation of "It" in a deep syntax framework. In *Proceedings of the Workshop on Discourse in Machine Translation*, DiscoMT 2015, pages 51–59, Sofia, Bulgaria. Association for Computational Linguistics.

Princeton University. 2010. WordNet.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK.

Revisiting Selectional Preferences for Coreference Resolution

Benjamin Heinzerling[*]
AIPHES
Heidelberg Institute for
Theoretical Studies
benjamin.heinzerling@h-its.org

Nafise Sadat Moosavi[*]
Heidelberg Institute for
Theoretical Studies
nafise.moosavi@h-its.org

Michael Strube
Heidelberg Institute for
Theoretical Studies
michael.strube@h-its.org

Abstract

Selectional preferences have long been claimed to be essential for coreference resolution. However, they are mainly modeled only implicitly by current coreference resolvers. We propose a dependency-based embedding model of selectional preferences which allows fine-grained compatibility judgments with high coverage. We show that the incorporation of our model improves coreference resolution performance on the CoNLL dataset, matching the state-of-the-art results of a more complex system. However, it comes with a cost that makes it debatable how worthwhile such improvements are.

1 Introduction

Selectional preferences have long been claimed to be useful for coreference resolution. In his seminal work on "Resolving Pronominal References" Hobbs (1978) proposed a semantic approach that requires reasoning about the "demands the predicate makes on its arguments." For example, selectional preferences allow resolving the pronoun *it* in the text *"The Titanic hit an iceberg. It sank quickly."* Here, the predicate *sink* 'prefers' certain subject arguments over others: It is plausible that a ship sinks, but implausible that an iceberg does.

Work on the automatic acquisition of selectional preferences has shown considerable progress (Dagan and Itai, 1990; Resnik, 1993; Agirre and Martinez, 2001; Pantel et al., 2007; Erk, 2007; Ritter et al., 2010; Van de Cruys, 2014). However, today's coreference resolvers (Martschat and Strube, 2015; Wiseman et al., 2016; Clark and Manning, 2016a, i.a.) capture selectional preferences only implicitly at best, e.g., via a given mention's dependency governor and other contextual features.

Since negative results do not often get reported, there is no clear evidence in the literature regarding the non-utility of particular knowledge sources. Consequently, an absence of the explicit modeling of selectional preferences in the recent literature is an indicator that incorporating this knowledge source has not been very successful for coreference resolution.

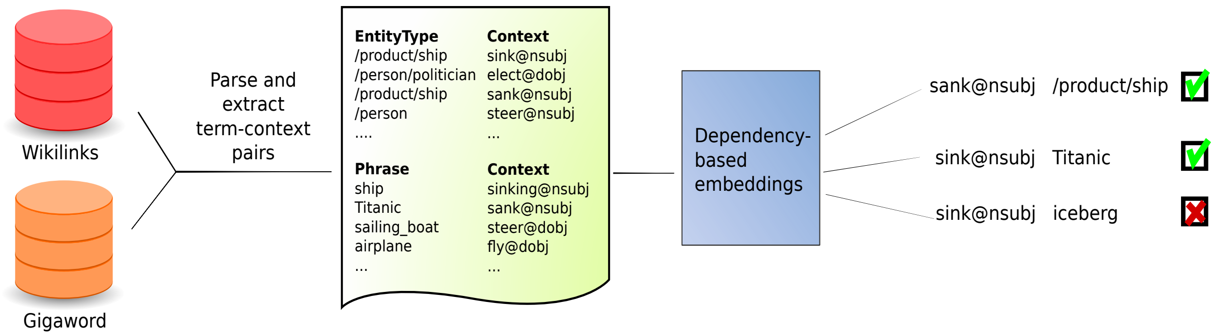
More than ten years ago, Kehler et al. (2004) declared the "non-utility of predicate-argument structures for pronoun resolution" and observed that minor improvements on a small dataset were due to fortuity rather than selectional preferences having captured meaningful world knowledge relations.

The claim by Kehler et al. (2004) is based on selectional preferences extracted from a, by current standards, small number of 2.8m predicate-argument pairs. Furthermore, they employ a simple (linear) maximum entropy classifier, which requires manual definition of feature combinations and is unlikely to fully capture the complex interaction between selectional preferences and other coreference features. Therefore, it is worth revisiting how a better selectional preference model affects the performance of a more complex coreference resolver.

In this work, we propose a fine-grained, high-coverage model of selectional preferences and study its impact on a state-of-the-art, non-linear coreference resolver. We show that the incorporation of our selectional preference model improves the performance. However, it is debatable whether such small improvements, that cost notable extra time or resources, are advantageous.

[*] These authors contributed equally to this work.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1332–1339
Copenhagen, Denmark, September 7–11, 2017.


Wikilinks
Gigaword
Parse and extract term-context pairs
Dependency-based embeddings
sank@nsubj /product/ship ✓
sink@nsubj Titanic ✓
sink@nsubj iceberg ☒

EntityType	Context
/product/ship	sink@nsubj
/person/politician	elect@dobj
/product/ship	sank@nsubj
/person	steer@nsubj
....	...

Phrase	Context
ship	sinking@nsubj
Titanic	sank@nsubj
sailing_boat	steer@dobj
airplane	fly@dobj
...	...

Figure 1: Dependency-based embedding model of selectional preferences.

2 Modeling Selectional Preferences

The main design choice when modeling selectional preferences is the selection of a relation inventory, i.e. the concepts and entities that can be relation arguments, and the semantic relationships that hold between them.

Prior work has studied many relation inventories. Predicate-argument statistics for word-word pairs (*eat, food*)[1] are easy to obtain but do not generalize to unseen pairs (Dagan and Itai, 1990). Class-based approaches generalize via word-class pairs (*eat, /nutrient/food*) (Resnik, 1993) or class-class pairs (*/ingest, /nutrient/food*) (Agirre and Martinez, 2001), but require disambiguation of words to classes and are limited by the coverage of the lexical resource providing such classes (e.g. WordNet).

Other possible relation inventories include semantic representations such as FrameNet frames and roles, event types and arguments, or abstract meaning representations. While these semantic representations are arguably well-suited to model meaningful world knowledge relationships, automatic annotation is limited in speed and accuracy, making it difficult to obtain a large number of such "more semantic" predicate-argument pairs. In comparison, syntactic parsing is both fast and accurate, making it trivial to obtain a large number of accurate, albeit "less semantic" predicate-argument pairs. The drawback of a syntactic model of selectional preferences is susceptibility to lexical and syntactic variation. For example, *The Titanic sank* and *The ship went under* differ lexically and syntactically, but would have the same or a very similar representation in a semantic framework such as FrameNet.

Our model of selectional preferences (Figure 1) overcomes this drawback via distributed representation of predicate-argument pairs, using (syntactic) dependencies that were specifically designed for semantic downstream tasks, and by resolving named entities to their fine-grained entity types.

Distributed representation. Inspired by Structured Vector Space (Erk and Padó, 2008), we embed predicates and arguments into a low-dimensional space in which (representations of) predicate slots are close to (representations of) their plausible arguments, as should be arguments that tend to fill the same slots of similar predicates, and predicate slots that have similar arguments. For example, *captain* should be close to *pilot*, *ship* to *airplane*, the subject of *steer* close to both *captain* and *pilot*, and also to, e.g., the subject of *drive*. Such a space allows judging the plausibility of unseen predicate-argument pairs.[2]

We construct this space via dependency-based word embeddings (Levy and Goldberg, 2014). To see why this choice is better-suited for modeling selectional preferences than alternatives such as word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014), consider the following example:

$$\begin{array}{ccccc} \text{captain} & \xleftarrow{\text{nsubj}} & \text{steers} & \xrightarrow{\text{dobj}} & \text{ship} \\ :: & & & & :: \\ \text{pilot} & \xleftarrow{\text{nsubj}} & \text{steers} & \xrightarrow{\text{dobj}} & \text{airplane} \end{array}$$

Here, *captain* and *ship*, have high syntagmatic similarity, i.e., these words are semantically related and tend to occur close to each other. This also holds for *pilot* and *airplane*. In contrast, *captain* and *pilot*, as well as *ship* and *airplane* have high paradigmatic similarity, i.e., they are seman-

[1] Examples due to Agirre and Martinez (2001).

[2] Prior work generalizes to unseen predicate-argument pairs via WordNet synsets (Resnik, 1993), a generalization corpus (Erk, 2007), or tensor factorization (Van de Cruys, 2010). Closest to our approach is neural model by Van de Cruys (2014), which, however, has much lower coverage since it is limited to 7k verbs and 30k arguments.

tically similar and occur in similar contexts. A model of selectional preferences requires paradigmatic similarity: The representations of *captain* and *pilot* in such a model should be similar, since they both can plausibly fill the subject slot of the predicate *steer*. Due to their use of linear context windows, word2vec and GloVe capture syntagmatic similarity, while dependency-based embeddings capture paradigmatic similarity (cf. Levy and Goldberg, 2014).

Enhanced++ dependencies. Due to distributed representation, our model generalizes over syntactic variation such as active/passive alternations: For example, *steer@dobj*[3] is highly similar to *steer@nsubjpass* (see Appendix for more examples). To further mitigate the effect of employing syntax as a proxy for semantics, we use Enhanced++ dependencies (Schuster and Manning, 2016). Enhanced++ dependencies aim to support semantic applications by modifying syntactic parse trees to better reflect relations between content words. For example, the plain syntactic parse of the sentence *Both of the girls laughed* identifies *Both* as subject of *laughed*. The Enhanced++ representation introduces a subject relation between *girls* and *laughed*, which allows learning more meaningful selectional preferences: Our model should learn that girls (and other humans) laugh, while learning that an unspecified *both* laughs is not helpful.

Fine-grained entity types. A good model of selectional preferences needs to generalize over named entities. For example, having encountered sentences like *The Titanic sank*, our model should be able to judge the plausibility of an unseen sentence like *The RMS Lusitania sank*. For popular named entities, we can expect the learned representations of *Titanic* and *RMS Lusitania* to be similar, allowing our model to generalize, i.e., it can judge the plausibility of *The RMS Lusitania sank* by virtue of the similarity between *Titanic* and *RMS Lusitania*. However, this will not work for rare or emerging named entities, for which no, or only low-quality, distributed representations have been learned. To address this issue, we incorporate fine-grained entity typing (Ling and Weld). For each named entity encountered during training, we generate an additional training instance by replacing the named entity with its entity type, e.g. *(Titanic, sank@nsubj)* yields *(/product/ship, sank@nsubj)*.

3 Implementation

We train our model by combining term-context pairs from two sources. Noun phrases and their dependency context are extracted from GigaWord (Parker et al., 2011) and entity types in context from Wikilinks (Singh et al., 2012). Term-context pairs are obtained by parsing each corpus with the Stanford CoreNLP dependency parser (Manning et al., 2014). After filtering, this yields ca. 1.4 billion phrase-context pairs such as *(Titanic, sank@nsubj)* from GigaWord and ca. 12.9 million entity type-context pairs such as *(/product/ship, sank@nsubj)* from Wikilinks. Finally, we train dependency-based embeddings using the generalized word2vec version by Levy and Goldberg (2014), obtaining distributed representations of selectional preferences. To identify fine-grained types of named entities at test time, we first perform entity linking using the system by Heinzerling et al. (2016), then query Freebase (Bollacker et al., 2008) for entity types and apply the mapping to fine-grained types by Ling and Weld.

The plausibility of an argument filling a particular predicate slot can now be computed via the cosine similarity of their associated embeddings. For example, in our trained model, the similarity of *(Titanic, sank@nsubj)* is 0.11 while the similarity of *(iceberg, sank@nsubj)* is -0.005, indicating that an iceberg sinking is less plausible.

4 Do Selectional Preferences Benefit Coreference Resolution?

We now investigate the effect of incorporating selectional preferences, implicitly and explicitly, in coreference resolution.

Figure 2 shows the selectional preference similarity of 10.000 coreferent and 10.000 non-coreferent mention pairs sampled randomly from the CoNLL 2012 training set. As we can see, while coreferent mention pairs are more similar than non-coreferent mention pairs according to the selectional preference similarity, there is not a direct relation between the similarity values and the coreferent relation. This indicates that coreference does not have a linear relation to the selectional preference similarities. However, it is worth investigating how these similarity values affect the overall performance when they are combined with

[3] In this work, a predicate's argument slots are denoted *predicate@slot*.

	MUC			B^3			CEAF$_e$			CoNLL	LEA		
	R	P	F$_1$	R	P	F$_1$	R	P	F$_1$	Avg. F$_1$	R	P	F$_1$
baseline	70.09	80.01	74.72	57.64	70.09	63.26	54.47	63.92	58.82	65.60	54.02	66.45	59.59
−gov	70.10	79.96	74.71	57.51	70.31	63.27	54.41	64.08	58.85	65.61	53.93	66.76	59.66
+SP	70.85	79.31	74.85	58.93	69.16	63.64	55.25	63.78	59.21	65.90	55.29	65.53	59.98
Reinforce	70.98	78.81	74.69	58.97	69.05	63.61	55.66	63.28	59.23	65.84	55.31	65.32	59.90

Table 1: Results on the CoNLL 2012 test set.

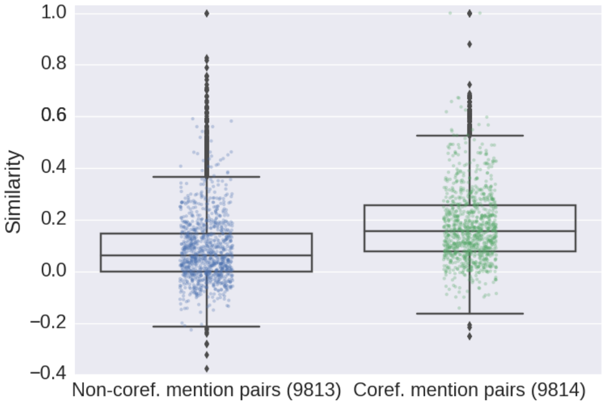


Figure 2: Selectional preference similarities of 10k coreferent and 10k non-coreferent mention pairs. Lines and boxes represent quartiles, diamonds outliers, points subsamples with jitter. Coreferent mention pairs are more similar than non-coreferent mention pairs with a Matthews correlation coefficient of 0.30, indicating weak to moderate correlation.

other knowledge sources in a non-linear way.

We select the ranking model of deep-coref (Clark and Manning, 2016b) as our baseline. deep-coref is a neural model that combines the input features through several hidden layers. *Baseline* in Table 1 reports our baseline results on the CoNLL 2012 test set. The results are reported using *MUC* (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), *CEAF$_e$* (Luo, 2005), the average F_1 score of these three metrics, i.e. CoNLL score, and *LEA* (Moosavi and Strube, 2016b). deep-coref includes the embeddings of the dependency governor of mentions. Combined with the relative position of a mention to its governor, deep-coref may be able to implicitly capture selectional preferences to some extent. *−gov* in Table 1 represents deep-coref performance when governors are not incorporated. As we can see, the exclusion of the governor information does not affect the performance. This result shows that the implicit modeling of selectional preferences does not provide any additional information to the coreference resolver.

	MUC	B^3	CEAF$_e$	CoNLL	LEA
	development				
baseline	74.10	63.95	59.73	65.93	60.16
+embedding	74.38	64.42	60.45	66.42	60.65
+binned sim.	74.36	64.54	60.21	66.37	60.77
	test				
baseline	74.72	63.26	58.82	65.60	59.59
+embedding	74.53	63.41	59.03	65.66	59.69
+binned sim.	74.85	63.64	59.21	65.90	59.98

Table 2: Incorporating the selectional preference model as new embeddings (+embedding) vs. as new pairwise features (+binned sim.).

For each mention, we consider (1) the whole mention string, (2) the whole mention string without articles, (3) mention head, (4) context representation, i.e. governor@dependency-relation, and (5) entity types if the mention is a named entity. We obtain an embedding for each of the above properties if they exist in the selectional preference model, otherwise we set them to unknown.

For each (antecedent, anaphor) pair, we consider all the acquired embeddings of anaphor and antecedent. We try two different ways of incorporating this knowledge into deep-coref including: (1) incorporating the computed embeddings directly as a new set of inputs, i.e. +*embedding* in Table 2. We add a new hidden layer on top of the new embeddings and combine its output with outputs of the hidden layers associated with other sets of inputs; and (2) computing a similarity value between all possible combinations of the antecedent-anaphor acquired embeddings and then binarizing all similarity values, i.e. +*binned sim.* in Table 2.

Providing selectional preference embeddings directly to deep-coref adds more complexity to the baseline coreference resolver. Yet, it performs on-par with +*binned sim.* on the development set and generalizes worse on the test set. +*SP* in Table 1 is the performance of +*binned sim.* on the test set. As we can see from the results, adding selectional

does [**that**]$_{ante}$ really impact the case ... [it]$_{ana}$ just shows	(impact@nsubj,shows@nsubj)
[it]$_{ante}$ will ask a U.S. bankruptcy court to allow [it]$_{ana}$	(ask@nsubj,allow@dobj)
[a **strain** that has n't even presented [itself]$_{ana}$]$_{ante}$	(presented@nsubj,presented@dobj)

Table 3: Examples of +SP correct links on the development set that do not exist in the baseline output.

Error type	Mention type		
	Proper	Common	Pronoun
Recall	-28	-29	-53
Precision	+18	+74	+61

Table 4: Differences in the number of recall and precision errors on the CoNLL'12 test set in comparison to the baseline.

preferences as binary features improves over the baseline.

Reinforce in Table 1 presents the results of the reward-rescaling model of Clark and Manning (2016a) that are so far the highest reported results on the official test set. The reward rescaling model of Clark and Manning (2016a) casts the ranking model of Clark and Manning (2016b) in the reinforcement learning framework which considerably increases the training time, from two days to six days in our experiments.

We analyze how our selectional preference model affects the resolution of various types of mentions. We use Martschat and Strube (2014)'s toolkit [4] to perform recall and error analyses. The differences in the number of recall and precision errors in *+SP* compared to *baseline* on the test set are reported in Table 4.

By using our selectional preference features, the number of recall errors decreases for all types of mentions. The recall error reduction is more prominent for pronouns. On the other hand, the number of precision errors increases for all types of mentions. The increase in the precision error is the highest for common nouns. Overall, *+SP* creates about 260 more links than *baseline*.

Table 3 lists a few examples from the development set in which *+SP* creates a link that *baseline* does not. It also includes the similarity that has a high value for the linked mentions and probably is the reason for creating the link. For instance, in the first example, based on our model, similarity(*impact@nsubj,shows@nsubj*) is known and it is also higher than similarity(*impact@dobj,shows@nsubj*).

In order to estimate a higher bound on the expected performance boost, we run the *baseline* and *+SP* models only on anaphoric mentions. By using anaphoric mentions, the performance improves by one percent, based on both the CoNLL score and *LEA*. This result indicates that the incorporation of selectional preferences creates many links for non-anaphoric mentions, which in turn decreases precision. Therefore, the overall performance does not improve substantially when system mentions are used. deep-coref incorporates anaphoricity scores at resolution time. One possible way to further improve the results of *+SP* is to incorporate anaphoricity scores at the input level. In this way, the coreference resolver could learn to use selectional preferences mainly for mentions that are more likely to be anaphoric. However, given that the F_1 score of current anaphoricity determiners or singleton detectors is only around 85 percent (Moosavi and Strube, 2016a, 2017), the effect of using system anaphoricity scores might be small.

5 Conclusions

We introduce a new model of selectional preferences, which combines dependency-based word embeddings and fine-grained entity types. In order to be effective, a selectional preference model should (1) have a high coverage so it can be used for large datasets like CoNLL, and (2) be combined with other knowledge sources in a non-linear way. Our selectional preference model slightly improves coreference resolution performance, but considering the extra resources that are required to train the model, it is debatable whether such small improvements are advantageous for solving coreference.

Acknowledgments

This work has been supported by the German Research Foundation (DFG) as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1 and the Klaus Tschira Foundation, Heidelberg, Germany.

[4] https://github.com/smartschat/cort

References

Eneko Agirre and David Martinez. 2001. Learning class-to-class selectional preferences. In *Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning (ConLL)*, pages 15–22.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the 1st International Conference on Language Resources and Evaluation,* Granada, Spain, 28–30 May 1998, pages 563–566.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data,* Vancouver, B.C., Canada, 10–12 June 2008, pages 1247–1250.

Kevin Clark and Christopher D. Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* Austin, Tex., 1–5 November 2016, pages 2256–2262.

Kevin Clark and Christopher D. Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Berlin, Germany, 7–12 August 2016.

Tim Van de Cruys. 2010. A non-negative tensor factorization model for selectional preference induction. *Natural Language Engineering*, 16(4):417–437.

Tim Van de Cruys. 2014. A neural network approach to selectional preference acquisition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 26–35, Doha, Qatar.

Ido Dagan and Alon Itai. 1990. Automatic processing of large corpora for the resolution of anaphora references. In *Proceedings of the 13th International Conference on Computational Linguistics,* Helsinki, Finland, 20–25 August 1990, volume 3, pages 330–332.

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics,* Prague, Czech Republic, 23–30 June 2007, pages 216–223.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 897–906.

Benjamin Heinzerling, Alex Judea, and Michael Strube. 2016. HITS at TAC KBP 2015: Entity discovery and linking, and event nugget detection. In *Proceedings of the Text Analysis Conference,* National Institute of Standards and Technology, Gaithersburg, Maryland, USA, 16–17 November 2015.

Jerry R. Hobbs. 1978. Resolving pronominal references. *Lingua*, 44:311–338.

Andrew Kehler, Douglas Appelt, Lara Taylor, and Aleksandr Simma. 2004. The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics,* Boston, Mass., 2–7 May 2004, pages 289–296.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308.

Xiao Ling and Daniel S. Weld. Fine-grained entity recognition. In *Proceedings of the 26th Conference on the Advancement of Artificial Intelligence,* Toronto, Ontario, Canada, 22–26 July 2012, pages 94–100.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Human Language Technology Conference and the 2005 Conference on Empirical Methods in Natural Language Processing,* Vancouver, B.C., Canada, 6–8 October 2005, pages 25–32.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Sebastian Martschat and Michael Strube. 2014. Recall error analysis for coreference resolution. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 2070–2081.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:405–418.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the ICLR 2013 Workshop Track.*

Nafise Sadat Moosavi and Michael Strube. 2016a. Search space pruning: A simple solution for better coreference resolvers. In *Proceedings of the*

2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, Cal., 12–17 June 2016, pages 1005–1011.

Nafise Sadat Moosavi and Michael Strube. 2016b. Which coreference evaluation metric do you trust? A proposal for a link-based entity aware metric. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Berlin, Germany, 7–12 August 2016, pages 632–642.

Nafise Sadat Moosavi and Michael Strube. 2017. Use generalized representations, but do not forget surface features. In *Proceedings of the 2nd Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2017)*, pages 1–7, Valencia, Spain.

Patrick Pantel, Rahul Bhagat, Bonaventura Coppola, Timothy Chklovski, and Eduard Hovy. 2007. ISP: Learning inferential selectional preferences. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 564–571.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition. LDC2011T07.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 1532–1543.

Philip Resnik. 1993. *Selection and Information: A Class-based Approach to Lexical Relationships*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Penn.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016).*

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015, University of Massachusetts, Amherst.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Message Understanding Conference (MUC-6)*, pages 45–52, San Mateo, Cal. Morgan Kaufmann.

Sam Wiseman, Alexander M. Rush, and Stuart Shieber. 2016. Learning global features for coreference resolution. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* San Diego, Cal., 12–17 June 2016. To appear.

Appendix

Query	Most sim. predicate slots	Most sim. entity types	Most sim. phrases
sink@nsubj	sink@nsubj:xsubj	/product/ship	Sea_Diamond
	sink@nsubjpass	/event/natural_disaster	Prestige_oil_tanker
	sinking@nmod:of	/finance/stock_exchange	Samina
	slide@nsubj	/astral_body	Estonia_ferry
	capsizing@nmod:of	/person/religious_leader	k-159
	plunge@nsubj	/finance/currency	Navy_gunboat
	sink@nmod:along_with	/military	Dona_Paz
	sinking@nsubj	/geography/glacier	ferry_Estonia
	tumble@nsubj	/product/airplane	add-fisk-independent-nytsf
	slip@nsubj	/transit	Al-Salam_Boccaccio
ship	capsize@nmod:of	/product/ship	vessel
	some@nmod:aboard	/train	cargo_ship
	experience@nmod:aboard	/product/airplane	cruise_ship
	afternoon@nmod:aboard	/transit	boat
	pier@nmod:for	/product/spacecraft	freighter
	escort@nmod:including	/location/bridge	container_ship
	lift-off@nmod:of	/broadcast/tv_channel	cargo_vessel
	disassemble@nsubjpass:xsubj	/location	Navy_ship
	near-collision@nmod:with	/living_thing	warship
	Conger@compound	/chemistry	tanker
steer@dobj	guide@dobj	/broadcast/tv_channel	business_way
	steer@nsubjpass	/product/car	newr_nbkg_nwer_ndjn
	shepherd@dobj	/organization/sports_team	BahrainDinar
	steering@nmod:of	/product/ship	reynard-honda
	nudge@dobj	/product/spacecraft	zigzag_course
	pilot@dobj	/event/election	team_home
	propel@dobj	/medicine/medical_treatment	U.S._energy_policy
	maneuver@dobj	/building/theater	williams-bmw
	divert@dobj	/education/department	interest-rate_policy
	lurch@nsubj	/product/airplane	trimaran
/product/ship	Repulse@conj:and	/product/airplane	battleship_Bismarck
	destroyer@amod	/train	pt_boat
	capsize@nmod:of	/product/car	battleship
	experience@nmod:aboard	/park	USS_Nashville
	near-collision@nmod:with	/military	USS_Indianapolis
	line@cc	/event/natural_disaster	k-159
	brig@conj:and	/award	frigate
	-lrb-@nmod:on	/geography/island	warship
	Umberto@conj:and	/person/soldier	Oriskany
	rumour@xcomp	/location/body_of_water	sister_ship

Figure 3: Most similar terms for the queries *sink@nsubj*, *ship*, *steer*, and */product/ship*.

Learning to Rank Semantic Coherence for Topic Segmentation

Liang Wang[1] **Sujian Li**[1,2] **Yajuan Lyu**[3] **Houfeng Wang**[1,2]
[1]Key Laboratory of Computational Linguistics, Peking University, MOE, China
[2]Collaborative Innovation Center for Language Ability, Xuzhou, Jiangsu, China
[3]Baidu Inc., Beijing, China
{intfloat,lisujian,wanghf}@pku.edu.cn lvyajuan@baidu.com

Abstract

Topic segmentation plays an important role for discourse parsing and information retrieval. Due to the absence of training data, previous work mainly adopts unsupervised methods to rank semantic coherence between paragraphs for topic segmentation. In this paper, we present an intuitive and simple idea to automatically create a "quasi" training dataset, which includes a large amount of text pairs from the same or different documents with different semantic coherence. With the training corpus, we design a symmetric CNN neural network to model text pairs and rank the semantic coherence within the learning to rank framework. Experiments show that our algorithm is able to achieve competitive performance over strong baselines on several real-world datasets.

1 Introduction

The goal of topic segmentation is to segment a document into several topically coherent parts, with different parts corresponding to different topics. Topic segmentation enables better understanding of document structure, and makes long document much easier to navigate. It also provides helpful information for tasks such as information retrieval, topic tracking etc (Purver, 2011).

Due to the lack of large scale annotated topic segmentation dataset, previous work mainly focus on unsupervised models to measure the coherence between two textual segments. The intuition behind unsupervised models is that two adjacent segments from the same topic are more coherent than those from different topics. Under this intuition, one direction of research attempts to measure coherence by computing text similarity. The typical methods include *TextTiling* (Hearst, 1997) and its variants, such as *C99* (Choi, 2000), *TopicTiling* (Riedl and Biemann, 2012b) etc. The other direction of research develops topic modeling techniques to explore topic representation of text and topic change between textual segments (Yamron et al., 1998; Eisenstein and Barzilay, 2008; Riedl and Biemann, 2012a; Du et al., 2013; Jameel and Lam, 2013). With carefully designed generative process and efficient inference algorithm, topic models are able to model coherence as latent variables and outperform lexical similarity based models.

Though unsupervised models make progress in modeling text coherence, they mostly suffer from one of the following two limitations. First, it is not precise to measure coherence with text similarity, since text similarity is just one aspect to influence coherence. Second, many assumptions and manually set parameters are usually involved in the complex modeling techniques, due to the absence of supervised information. To overcome aforementioned limitations, we prefer to directly model the text coherence by exploring possible supervised information. Then, we can learn a function $f(s1, s2)$ which takes two textual segments $s1$ and $s2$ as input, and directly measure their semantic coherence.

As we know, it is hard to directly compile and collect a large number of samples with coherence scores labeling. Here we propose an intuitive and simple strategy to automatically create a "quasi" training corpus for supervision. It is a common sense that the original documents written by human are generally more coherent than a patchwork of sentences or paragraphs randomly extracted from different documents. In such cases, two textual segments from the same document are more coherent than those from different documents, and two segments from the same paragraph

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1340–1344
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

are more coherent than those from different paragraphs. Then, we can get a large set of text pairs with partial ordering relations, which denote some text pairs are more coherent than other text pairs. With these ordering information, we propose to apply the learning to rank framework to model the semantic coherence function $f(s1, s2)$, based on which topic boundaries are identified.

The next key problem is how to model and represent text pairs. It is fortunate that neural networks have emerged as a powerful tool for modeling text pairs (Lu and Li, 2013; Severyn and Moschitti, 2015; Yin et al., 2015; Hu et al., 2014), freeing us from feature engineering. In this paper, we develop a symmetric convolutional neural network (CNN) framework, whose main idea is to jointly model text representation and interaction between texts. With our acquired large amount of training data, our CNN-based method is capable of reasonably rank semantic coherence and further conduct topic segmentation.

2 Model

2.1 Coherence Ordering between Text Pairs

In our work, we define $f(s1, s2)$ as a function, which returns a real number as semantic coherence score of the text pair <s1,s2>. To model $f(s1, s2)$ of any text pair, we aim to explore the partial ordering relations of coherence between different text pairs, since it is hard to get a corpus with labeled coherence scores.

Next, we exploit the two types of ordering relations stated in Section 1. To formalize, we notate a collection of documents as D. Each document $d_i \in D$ consists of several paragraphs, and each paragraph $p_j \in d_i$ consists of several sentences. We use $T_{ij}^{s:(s+k)}$ to represent a text segment covering k sentences starting from the s-th sentence in document d_i's j-th paragraph. To make symbols less cluttered, we omit k and simply use T_{ij}^{s} for the same meaning. A text pair $< T_{ij}^{s}, T_{i'j'}^{s'} >$ is a tuple of two text segments.

The first one ordering relation is: coherence score of a text pair from different documents is lower than that from the same document. Formally, its mathematical expression is shown below:

$$f(< T_{i\cdot}^{\cdot}, T_{i'\cdot}^{\cdot} >) < f(< T_{jm}^{\cdot}, T_{jm'}^{\cdot} >), \quad i \neq i', m \neq m' \tag{1}$$

where dot $\cdot$ means arbitrary value.

The second one is: coherence score of text pair from different paragraphs is lower than those from the same paragraph, as represented below.

$$f(< T_{ip}^{\cdot}, T_{ip'}^{\cdot} >) < f(< T_{jq}^{n}, T_{jq}^{n+k} >), p \neq p' \tag{2}$$

As our defined relations are partially ordering, they have the properties of reflexivity, transitivity, and antisymmetry, Then we can easily infer that coherence score of a text pair from different documents is also lower than that from the same paragraph.

2.2 Learning to Rank Semantic Coherence

Learning to rank is a widely used learning framework in the field of information retrieval (Liu et al., 2009). There are generally three formulations (Li, 2011): *pointwise ranking*, *pairwise ranking*, and *listwise ranking*. The goal is to learn a ranking function $f(\mathbf{w}, tp_i) \to y_i$ where tp_i denotes a text pair <s1,s2>. f maps tp_i to a real value y_i which is semantic coherence score in this paper, $\mathbf{w}$ is weight vector. We examine both *pointwise ranking* and *pairwise ranking* methods, *listwise ranking* is not naturally fit for our task, so it is not discussed here.

2.2.1 Pointwise Ranking

For pointwise formulation, $y_i = f(\mathbf{w}, tp_i) \in [0, 1]$ computes inner product between weight vector $\mathbf{w}$ and text pair tp_i's representation vector $\mathbf{h_i}$. Here we apply a $sigmoid$ non-linearity function.

$$y_i = \sigma(\mathbf{w} \cdot \mathbf{h_i}) \tag{3}$$

Representation vectors $\mathbf{h_i}$ of the text pair can be jointly learned through a neural network, which will be introduced in next subsection.

To conform to the partial ordering relations, we score each training instance tp_i as follows.

$$y_i^* = \begin{cases} 0, & \text{If } tp_i \text{ comes from different documents.} \\ 1, & \text{If } tp_i \text{ comes from same paragraph.} \\ \alpha, & \text{If } tp_i \text{ comes from different paragraphs.} \end{cases}$$

where $0 < \alpha < 1$ and α is a hyper-parameter chosen to maximize performance on validation dataset.

With N training instances, we formulate the coherence scoring as a regression problem and use cross entropy as loss function:

$$\min -\frac{1}{N}\sum_{i=1}^{N}(y_i \log y_i^* + (1-y_i)\log(1-y_i^*)) \tag{4}$$

Generally speaking, *pointwise ranking* is simple, scalable and efficient to train.

2.2.2 Pairwise Ranking with Sampling

Pairwise formulation explicitly compares each pair of training instance and requires a minimal margin ϵ between their ranking score.

$$f(\mathbf{w}, tp_i) > f(\mathbf{w}, tp_j) + \epsilon \quad (5)$$

Here, the text pair tp_i has a higher ranking score than tp_j, and $y_i = f(\mathbf{w}, tp_i) \in (-\infty, +\infty)$. Without loss of generality, we set $\epsilon = 1$ and use squared hinge loss as optimization function.

$$\min \ -\frac{1}{M}\sum_{i,j} max(0, 1 + y_j - y_i)^2 \quad (6)$$

where M is the number of pairs we need to compare. As we can see, in our problem setting, $M \approx N^2$, which makes M an extremely large number when $N \approx 10^5$.

To make training feasible, we adopt a straightforward sampling mechanism, which randomly samples pairs from different groups to construct a mini-batch on the fly during training.

Pairwise ranking is reported to have better performance than *pointwise ranking*, but it is less efficient to train.

2.3 Semantic Coherence Neural Network

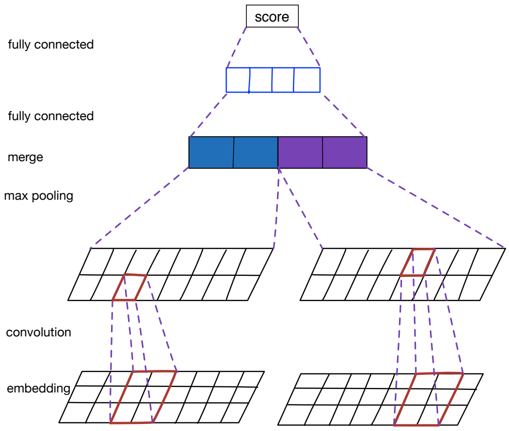


Figure 1: Semantic Coherence Neural Network

To model the text pair instances, we develop a symmetric convolutional neural network (CNN) architecture, as shown in Figure 1. Our model consists of two symmetric CNN models, and the two CNNs share their network configuration and parameters. Each CNN converts one text into a low-dimensional representation, and two generated text representation vectors are finally concatenated and fed into the scoring layer to get a real value as the coherence score.

2.4 Inference

At test time, coherence scores between any two adjacent paragraphs are computed. $T - 1$ paragraph boundaries with lowest semantic coherence score are chosen as topic boundaries, where T is ground-truth number of topics.

This inference procedure is computationally efficient. Unlike *TextTiling*, it doesn't need to calculate a so-called "depth score".

3 Experiments

3.1 Experimental Setup

Data In order to train our ranking neural network, we use full *English Wikipedia dump*, which consists of more than 5 million documents, to automatically construct text pairs.

For performance evaluation, we use topic segmentation dataset from (Jeong and Titov, 2010)[1]. This dataset consists of 864 manually labeled documents from four different areas, as shown in Table 1.

	News	*Lecture*	*Report*	*Biography*
#documents	184	120	160	400

Table 1: Overview of four datasets.

Baselines To compare with our method, *TextTiling* (Hearst, 1997), *TopicTiling* (Riedl and Biemann, 2012b) and *BayesSeg* (Eisenstein and Barzilay, 2008) are adopted as three baselines. We use open source implementations of *TextTiling*[2] and *TopicTiling*[3], and results of *BayesSeg* are from (Jeong and Titov, 2010).

Hyperparameters Our neural network implementation is based on *Tensorflow* (Abadi et al., 2015). We use pre-trained 50 dimensional *Glove* vectors (Pennington et al., 2014)[4] for word embeddings initialization. Each text pair consists of 2 text segments, and each text segment consists of

[1] We do not compare with *MultiSeg* model proposed by (Jeong and Titov, 2010), since our model is for single-document topic segmentation while *MultiSeg* is for multi-document topic segmentation.

[2] https://github.com/nltk/nltk/tree/develop/nltk/tokenize

[3] https://github.com/ldulcic/text-segmentation

[4] http://nlp.stanford.edu/projects/glove/

	News			*Lecture*			*Report*			*Biography*		
	P_k	WD	$F1$	P_k	WD	$F1$	P_k	WD	$F1$	P_k	WD	$F1$
TextTiling	0.340	0.344	0.447	0.204	0.206	0.231	0.466	0.469	0.365	0.335	0.403	0.361
TopicTiling	0.415	0.436	0.338	0.359	0.379	0.571	0.288	0.296	0.383	0.381	0.423	0.390
BayesSeg	0.318	0.326	0.537	**0.173**	0.190	0.526	0.254	0.255	0.526	**0.186**	**0.208**	0.470
Ours-pair-finetune	0.180	0.181	0.570	0.200	0.202	0.560	0.263	0.263	0.492	0.223	0.228	0.448
Ours-point-finetune	0.182	0.183	0.572	0.197	0.200	0.569	0.245	0.247	0.511	0.229	0.232	0.442
Ours-pair-static	0.176	0.178	0.580	0.177	0.180	0.600	0.252	0.253	0.518	0.220	0.224	0.472
Ours-point-static	**0.173**	**0.175**	**0.587**	0.176	**0.179**	**0.608**	**0.240**	**0.241**	**0.529**	0.216	0.219	**0.479**

Table 2: Experimental results. (a) *Ours-pair-finetune* is pairwise ranking model with word embedding fine-tuning. (b) *Ours-point-static* is pointwise ranking model without word embedding fine-tuning, etc.

no more than 3 sentences. Stop words and digits are removed from input text, and all words are converted to lowercase. We pad input sequence to 40 tokens. In order to capture information of different granularity, convolution window size of both 2 and 3 are used, with 64 filters for each window size. L2 regularization coefficient is set to 0.001. *Adam* algorithm (Kingma and Ba, 2014) is used for loss function minimization. We set α to 0.7 for *pointwise ranking*.

Evaluation System performance is evaluated according to three metrics: P_k (Beeferman et al., 1999), WindowDiff(WD) (Pevzner and Hearst, 2002) and $F1$ score. P_k and WD are calculated based on sliding windows, and can assign partial score to incorrect segmentation. Note that P_k and WD are penalty metrics, smaller value means better performance.

3.2 Results and Analysis

Experimental results are shown in Table 2. Our proposed model is examined in 4 different settings, including whether to use *pointwise ranking* or *pairwise ranking* algorithm, and whether to fine-tune word embeddings or not. The best model *Ours-pointwise-static* is able to achieve better or competitive performance compared to *BayesSeg* and *TopicTiling* according to all three metrics, especially on *News* dataset. *TopicTiling* is reported to perform well on heuristically constructed dataset (Riedl and Biemann, 2012b), but behave mediocre on manually labeled dataset in our experiments.

One interesting phenomenon is that fine-tuned word embeddings has negative impact on overall performance, which is generally not the case in many NLP tasks. The reason may be that our task involves domain adaptation, and word embeddings should generalize well across different domains rather than adapt to *Wikipedia* text. Though our proposed sampling mechanism enables easier training of *pairwise ranking* model, it inevitably loses some ordering information, which makes *pairwise ranking* model perform slightly worse than *pointwise ranking* model.

Text Pair	Score
A: A variety of techniques have been directed toward the study of blood group antibodies. **B:** If I'd work on my place-kicking he thought he could use me.	0.022
A: A second miracle is required for her to proceed to canonization. **B:** Mother Teresa inspired a variety of commemorations.	0.587
A: Plants have an amazing ability to respond to stimuli from their environment. **B:** These responses to environmental factors are known as tropisms.	0.861

Table 3: Coherence Score between Text Pairs.

To illustrate what the model has learned, we show some typical examples of coherence score for text pair **<A**,**B>** in Table 3. There is almost no lexical overlap for all the three text pairs, cosine similarity between one-hot vectors would surely fail to rank them, even though *"canonization"* and *"commemorations"*, *"respond"* and *"responses"*, *"environment"* and *"environmental"* are closely related semantically. As we expect, our proposed model is able to capture such semantic relatedness and assign reasonable score to each text pair, which is a key to topic boundary detection.

4 Conclusion

This paper proposes a novel approach for topic segmentation by learning to rank semantic coherence. Symmetric convolutional neural network is used for text pair modeling. Training data can be automatically constructed from unlabeled documents, and no labeled data is needed. Experiments show promising performance on dataset from various domains.

Acknowledgments

We thank the anonymous reviewers for their insightful comments on this paper. This work was partially supported by National Natural Science Foundation of China (61572049 and 61333018) and Baidu-Peking University Joint Project. The correspondence author of this paper is Sujian Li.

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org* 1.

Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical models for text segmentation. *Machine learning* 34(1-3):177–210.

Freddy YY Choi. 2000. Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*. Association for Computational Linguistics, pages 26–33.

Lan Du, Wray L Buntine, and Mark Johnson. 2013. Topic segmentation with a structured topic model. In *HLT-NAACL*. pages 190–200.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 334–343.

Marti A Hearst. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics* 23(1):33–64.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*. pages 2042–2050.

Shoaib Jameel and Wai Lam. 2013. An unsupervised topic segmentation model incorporating word order. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, pages 203–212.

Minwoo Jeong and Ivan Titov. 2010. Multi-document topic segmentation. In *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, pages 1119–1128.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Hang Li. 2011. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems* 94(10):1854–1862.

Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3(3):225–331.

Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*. pages 1367–1375.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543.

Lev Pevzner and Marti A Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics* 28(1):19–36.

Matthew Purver. 2011. Topic segmentation. *Spoken language understanding: systems for extracting semantic information from speech* pages 291–317.

Martin Riedl and Chris Biemann. 2012a. Text segmentation with topic models. *Journal for Language Technology and Computational Linguistics* 27(1):47–69.

Martin Riedl and Chris Biemann. 2012b. Topictiling: a text segmentation algorithm based on lda. In *Proceedings of ACL 2012 Student Research Workshop*. Association for Computational Linguistics, pages 37–42.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pages 373–382.

Jonathan P Yamron, Ira Carp, Larry Gillick, Steve Lowe, and Paul van Mulbregt. 1998. A hidden markov model approach to text segmentation and event tracking. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. IEEE, volume 1, pages 333–336.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193* .

GrASP: Rich Patterns for Argumentation Mining

Eyal Shnarch **Ran Levy*** **Vikas Raykar**‡ **Noam Slonim**
IBM Research - Haifa, Israel
‡IBM Research - Bangalore, India
{eyals, ranl, noams}@il.ibm.com viraykar@in.ibm.com

Abstract

We present the GrASP algorithm for automatically extracting patterns that characterize subtle linguistic phenomena. To that end, GrASP augments each term of input text with multiple layers of linguistic information. These different facets of the text terms are systematically combined to reveal rich patterns. We report highly promising experimental results in several challenging text analysis tasks within the field of Argumentation Mining. We believe that GrASP is general enough to be useful for other domains too.

1 Introduction

Many standard text analysis tasks can be addressed relatively well while exploiting simple textual features, e.g., Bag-Of-Words representation and Naive Bayes for document classification (McCallum and Nigam, 1998). However, the identification of more subtle linguistic phenomena, that are further reflected via relatively short texts – as opposed to whole documents – may require a wider spectrum of linguistic features.

The main contribution of this work is in outlining a simple method to automatically extract rich linguistic features in the form of patterns, and demonstrate their utility on tasks related to *Argumentation Mining* (Mochales Palau and Moens, 2009), although we believe that the proposed approach is not limited to this domain.

Argumentation mining involves automatically identifying argumentative structures within a corpus – e.g., claims or conclusions, and evidence instances or premises – as well as their interrelations. For instance, each of the following sentences includes a *claim* for a **[topic]**.

1. Opponents often argue that *the open primary is unconstitutional.* **[Open Primaries]**
2. Prof. Smith suggested that *affirmative action devalues the accomplishments of the chosen.* **[Affirmative Action]**
3. The majority stated that *the First Amendment does not guarantee the right to offend others.* **[Freedom of Speech]**

These sentences share almost no words in common, however, they are similar at a more abstract level. A human observer may notice the following underlying common structure, or *pattern*: `[someone][argue/suggest/state][that]` `[topic term][sentiment term]`

We present GrASP, standing for GReedy Augmented Sequential Patterns, an algorithm that aims to automatically capture such underlying structures of the given data. Table 1 shows the pattern that GrASP may find for the above examples, along with its matches in those texts. Such patterns can then be used to detect the existence of the phenomenon in new texts.

The algorithm starts with augmenting the terms of the input with various layers of *attributes*, such as hypernyms from WordNet (Fellbaum, 1998), named entity types, and domain knowledge (Section 3.1). This multi-layered representation enables GrASP to consider many facets of each term. Next, it finds the most indicative attributes (Section 3.2) and iteratively grows patterns by blending information from different attributes (Section 3.3). A greedy step is performed at the end of each iteration, when the algorithm only keeps the top *k* patterns, ranked by their predictive power. This results with a set of cross-layered patterns whose match in a given text instance suggests the appearance, or the non-appearance, of the target phenomenon.

Researchers can add layers of attributes of different kinds without being worried about which of

[*] First two authors contributed equally.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1345–1350
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

[noun]		[express]	[that]		[noun,topic]		[sentiment]	
Opponents	often	argue	that	the	open primary	is	unconstitutional	.
Prof. Smith		suggested	that		affirmative action		devalues	the ...
The majority		stated	that	the	First Amendment	does not guarantee ... to	offend	others.

Table 1: Claim sentences aligned by their common underlying pattern. [express] stands for all its (in)direct hyponyms, and [noun,topic] means a noun which is also related to the topic.

them are useful for the detection of the target phenomenon, and how to combine them. GrASP performs feature selection while generating complex patterns out of these attributes that best capture aspects of the target phenomenon.[1]

In experiments over different argumentation mining tasks, we show that GrASP outperforms classical techniques, and boosts full argumentation mining systems when added to them.

2 Background

While some aspects of GrASP were considered in the past, to the best of our knowledge, no prior work has presented a framework that allows users to: (i) easily add any type of attribute to the pattern alphabet, and (ii) consider **all** attributes when searching for patterns. GrASP provides a framework to integrate information from different layers, choosing the best combination to produce highly expressive patterns.

The alphabet of Hearst (1992) patterns is mainly stop words and noun-phrase tags, while Snow et al. (2004) add syntactic relations. Yangarber et al. (2000) consider a larger set of attributes (e.g., named entities, numeric expressions), however they commit to one generalization of each term. In contrast, we do not limit our alphabet and systematically consider all attributes of each term.

Riloff and Wiebe (2003) start with a small set of syntactic templates, composed of a single syntactic relation and a single POS tag, to learn a variety of lexicalized patterns that match these templates. RAPIER (Califf and Mooney, 2003) constraints are similar to our attributes, but are basic (surface form, POS tag, and hypernyms only), and expanding them will exponentially increase its complexity. In contrast, adding attributes to GrASP only increment runtime linearly (see Section 3.2).

To summarize, prior works usually have a basic alphabet and commit to one rule to generalize each term. Commonly, they do not allow gaps between their elements, nor assigning several attributes to a single element of the pattern.

Such characteristics are presented in *sequential patterns* (Agrawal and Srikant, 1995) which are mainly used for data mining and rarely for unstructured text (Jindal and Liu, 2006). GrASP also has these characteristics, and in addition it can learn *negative* patterns, indicating that the examined text *does not* contain the target phenomenon.

The phenomena we target are from the area of Argumentation Mining (see Lippi and Torroni (2016) for a survey). We focus on open-domain argument extraction. In this context, Levy et al. (2014) detect claims relevant to a debatable topic, Lippi and Torroni (2015) defined the context-*independent* claim detection task, and Rinott et al. (2015) introduced the context dependent evidence detection task (which is further split into different types of evidence, e.g., a *study* that supports a claim or a relevant *expert* testimony). These tasks aim to capture a subtle and rare linguistic phenomenon within large corpora, hence are suitable for demonstrating the potential of GrASP.

3 The GrASP Algorithm

The algorithm depicted in Algorithm 1. Its input is a set of positive and negative examples for the target phenomenon. The output is a ranked list of patterns, aiming to indicate the presence – or absence – of this phenomenon. In the following, a pattern is considered to be *matched* in a text iff all its elements are found in it, in the specified order, possibly with gaps between them, within a window of size w.

3.1 Multi-Layered Term Representations

Consider the verbs (argue/suggest/state) in the examples in Section 1. Using the POS tag *verb* to generalize them will end up with an overly general representation, while their hypernym, *express*, offers a better level of generalization.

Aiming to formalize this intuition, we start by augmenting each term in the input with a variety of linguistic *attributes* such as its POS tag, its syntac-

[1] Get GrASP cloud service at http://ibm.biz/graspULP

Algorithm 1: The GrASP algorithm.

```
Input: positive/negative text examples, $k_1$, $k_2$, maxLen
Output: a ranked list of patterns
1  $(pos, neg) \leftarrow augment(positives, negatives)$
2  $attributes \leftarrow extractAttributes(pos, neg)$
3  $alphabet \leftarrow chooseTopK(attributes, k_1)$
4  $patterns \leftarrow alphabet$
5  $last \leftarrow patterns$
6  for $length \leftarrow 2$ to maxLen do
7      $curr \leftarrow \theta$
8      for $p \in last$ do
9          for $a \in alphabet$ do
10             $curr \leftarrow curr \cup \{growRight(p, a)\}$
11             $curr \leftarrow curr \cup \{growInside(p, a)\}$
12     $last \leftarrow curr$
13     $patterns \leftarrow$
         $chooseTopK(patterns \cup current, k_2)$
14 return patterns
```

tic relation in a parse tree, and semantic attributes such as its hypernyms, WordNet superclasses, indications whether it is a named entity, and whether it bears a sentiment.[2] This attributes set can serve as a starting point for many text analysis tasks.

In addition, GrASP allows to add task-specific attributes. Thus, for context-dependent arguments detection we add boolean attributes indicating whether the term is related to the topic, whether it appears in a lexicon characterizing argumentative texts, or in a lexicon characterizing the topic.[3]

After augmentation, the representation of *argue*, from the first example, is: `[argue, VB, hypernyms={present, state, express}, in claim lexicon, root node, supeclass = communication]`.

3.2 Defining the Patterns Alphabet

The augmented representation, described above, is the first step (line 1 in Algorithm 1). Next, we define the alphabet of attributes that will be used to compose longer patterns (lines 2–3). To that end, we first discard non-frequent attributes that are matched in less than t_1 of all input examples.

Then, we sort all remaining attributes by their information gain (Mitchell, 1997) with the label, and select the top k_1 attributes. We discard redundant attributes whose correlation to some previously selected attribute is above t_2, measured by the normalized mutual information (Cover and Thomas, 2006). The selected k_1 attributes constitute the *alphabet* of the algorithm, or "patterns" of length 1. Note that considering additional attributes only affects this first iteration, and only increases it linearly.

3.3 Growing Patterns

Learning longer patterns is done by iteratively growing patterns selected in previous iterations, keeping only the most indicative ones (lines 6–13). We apply two methods for growing a pattern, p (e.g., `[noun]`) w.r.t. an attribute a (e.g., *obj*): (i) grow right – add a as another term in the pattern (i.e., `[noun][obj]`); (ii) grow inside – add a as another attribute to the last term of p, making it more specific (i.e., `[noun, obj]`). After each iteration (line 13), the top k_2 patterns are kept (after sorting by information gain and discarding redundant ones). Iterations continue till reaching *maxLen*.

Since GrASP relies on information gain for sorting, it can identify indicative *negative* patterns, implying that the target phenomenon is *less likely* to be presented in the examined example if such patterns were matched in it.

GrASP can be seen as a simple formal interface, allowing the user to examine a wide range of information sources letting the algorithm to select and combine them all and come up with the most useful patterns.

4 Evaluation and Results

In the following experiments we used a logistic regression classifier on top of the extracted patterns. Each pattern is used as a binary feature, which receives value of 1 iff it is matched in the candidate.

To demonstrate the robustness of the this approach, in all experiments we report the results of a single configuration of GrASP parameters, selected based on a quick analysis over a small portion of the claim-sentence detection data (task (a) below).[4] Specifically, we used minimal frequency threshold $t_1 = 0.005$, correlation threshold $t_2 = 0.5$, size of the alphabet $k_1 = 100$, number of patterns in the output $k_2 = 100$, maximal pattern length *maxLen* $= 5$, and window size $w = 10$.

This configuration is by no means the optimal one, and we saw that by carefully tuning the parameters per task, results were improved.

[2] We used OpneNLP POS tagger, Stanford NER, McCord and Bernth (2010) parser, WordNet superclasses, and the lexicon in Hu and Liu (2004) for sentiment words.

[3] We utilize existing lexicons, learning them is out of the scope of this work.

[4] We randomly chose 10 topics. The performance over them was somewhat inferior to that over all 58 topics.

	(a) Claim sentence			(b) Expert evidence			(c) Study evidence		
system	P@5	P@10	P@20	P@5	P@10	P@20	P@5	P@10	P@20
Naive Bayes	13.8	10.2	8.1	8.4	9.6	8.0	13.1	11.6	9.3
Basic patterns	21.4	15.5	12.7	16.5	15.2	12.2	18.3	16.1	12.8
CNN	25.5	21.2	16.9	18.2	16.3	14.6	26.5	22.2	18.4
GrASP alphabet	30.0	25.7	22.8	25.8	22.5	18.7	30.5	25.6	**21.1**
GrASP	**41.7****	**34.5****	**27.0****	**29.0***	**25.2***	**21.9***	**35.4***	**25.7**	20.0

Table 2: Macro-averaged precision results for GrASP over three argumentation mining tasks. Significant results in comparison to GrASP alphabet/CNN are marked with **/* respectively (paired t-test with $p < 0.01/0.02$ respectively).

4.1 Direct Evaluation

We consider three context-dependent argumentation mining tasks: (a) Claim sentence detection (Levy et al., 2014), (b) Expert evidence detection, and (c) Study evidence detection. The latter two tasks are described in Rinott et al. (2015), where the goal is to detect sentences that can be used as an evidence to support/contest the topic.[5]

The benchmark data for these tasks was extracted from the data released by Rinott et al. (2015), consisting of 547 Wikipedia articles in which claims and evidence instances were manually annotated, in the context of 58 debatable topics. In all tasks the data is highly skewed towards negative examples (only 2.5% of 80.5K instances are positives in task (a), 4% of 55.6K in task (b), and 3.7% of 31.8K in task (c)), making these tasks especially challenging.

As (Levy et al., 2014; Rinott et al., 2015) we use a leave-one-topic-out schema; training over 57 topics, testing over the left out topic.

Our group develops debate supportive technologies which can assist humans to reason, make decisions, or persuade others.[6] Since in this scenario humans mainly consider top results (similar to information retrieval), precision is more relevant than recall. Thus, we report the macro-averaged Precision@K, where $K \in \{5, 10, 20\}$.

We considered the following baselines:

Naive Bayes: over BOW representation, discarding unigrams which appear less than 10 times.

Basic Patterns: a baseline that reflects common practices in the literature where a pattern is a consecutive ordered list of stop words or POS tags. We add a symbol for topic match (for the context-dependent tasks). A brute force process generates all possible patterns up to size *maxLen* and selects top k by the same procedure as GrASP. For each task we report the best results obtained with $k \in \{50, 100, .., 400\}$.

Convolutional Neural Network (CNN): following (Kim, 2014; Vinyals and Le, 2015) we used CNN whose input is a concatenation of the topic and the candidate.[7] The final state vector is fed to a LR soft-max layer. Cross-entropy loss function was used for training. The embedding layer was initialized using word2vec vectors (Mikolov et al., 2013). Hyper-parameters were tuned on the same portion of the dataset as used by GrASP for tuning.

For these baselines, we are not aware of available methods to incorporate GrASP multi-layered representation.

GrASP alphabet: a simplified version of GrASP which uses the chosen alphabet, or "patterns" of length 1. This baseline does utilize all the information available for GrASP.

Table 2 shows that *Naive Bayes* performance is the lowest, demonstrating that a simple representation is not sufficient for such complex tasks. Using *Basic patterns* yields better performance, and *CNN* performs even better. *GrASP alphabet* outperforms CNN, indicating the potential of explicitly incorporating linguistic information. Finally, using the patterns extracted by *GrASP* outperforms all other methods, emphasizing the added value of constructing patterns over the initial contribution of the multi-layered representation.

GrASP provides an easy way to analyze the importance of each attribute by inspecting its score at the end of the first iteration, the one which determines the alphabet. For example, *PERCENT* score was very high in the alphabet for Study evidence patterns (task b), and *Person* and *Organization* were ranked high in the alphabet of the Expert evidence (task c). Still, these three named enti-

[5] We did not examine the Anecdotal type due to the small size of the available benchmark data.

[6] for more details see IBM Debating Technologies.

[7] RNN, LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014) were also considered but resulted with inferior performance.

system	P@5	P@10	P@20	P@50	R@50
Levy14	–	–	–	18	40
Levy14-rep	30.9	27.3	23.5	17.6	38.4
GrASP	32.7	30	23.9	17.5	36.2
Combined	**40***	**32.4***	**28***	**20.2***	**43.2***

Table 3: Adding GrASP to a full claim detection system. Significant results in comparison to Levy14-rep are marked by * (paired t-test with $p < 0.02$).

ties were not selected as part of the alphabet for Claim sentences (task a) – reflecting the importance of *PERCENT* in sentences describing studies and their numeric results, and the importance of authoritative source (either *Person* or *Organization*) in evidence based on expert testimonies.

For task (a) we performed two ablation tests, each of them yielded a decrease in performance: (i) not limiting the match of a pattern in a window (a decrease of 10.3 for P@5 and 2.8 for P@20), and (ii) not enforcing the order defined by the pattern (a decrease of 7.6 for P@5 and 2.8 for P@20).

4.2 Indirect Evaluation

In this evaluation we add GrASP patterns as additional features to the full claim detection system of Levy et al. (2014) to inspect their contribution. This evaluation is performed on a second claim detection benchmark (on which they have reported results), released by Aharoni et al. (2014) (1,387 annotated claims associated with 33 topics).

The system of Levy et al. (2014) is comprised of a cascade of three components; (i) detecting sentences which contain claims, (ii) identifying the exact boundaries of the claim part within the sentence, and (iii) ranking the claim candidates. Each of these components applies a classifier over dedicated features. Results were reported for the full cascade and for the first component, which is our task (a). For an idea on how to adapt GrASP for the claim boundaries detection task, see Section 5.

Table 3 presents measures reported in Levy et al. (2014) (right hand side) as well as additional measures which reflect the focus of this work on the precision of the top ranked candidates (performance of all systems on P@200 and R@200 were comparable and were omitted due to space limitations). The system of Levy et al. (2014), denoted *Levy14*, and our reproduction of it, denoted *Levy14-rep*, obtained comparable results.[8]

Evidently, utilizing GrASP patterns alone achieve similar performance as Levy14-rep. Considering the fact that Levy14-rep is a full system, tailored for claim detection via a lengthy feature engineering process, these results, obtained using only GrASP patterns, are promising. Adding GrASP features to Levy14-rep, denoted *Combined*, we observe a significant improvement, demonstrating their complementary value.

5 Discussion

GrASP extracts rich patterns that characterize subtle linguistic phenomena. It exploits a wide variety of information layers in a unified manner, identifying the most discriminative attributes for the given task, and greedily composes them into patterns. We demonstrated GrASP significant impact on several argumentation mining tasks.

As this was not the focus of this work, we chose standard statistical criteria to sort the candidate patterns and to filter redundant ones. Considering other criteria, and also more sophisticated search strategies to explore the huge space of possible patterns, is left for future work.

In addition to their value in classification tasks, the patterns revealed by GrASP are easy to interpret, in contrast to alternative techniques, like Deep Learning. Thus, these patterns can provide researchers with additional insights regarding the target phenomenon. These insights can be integrated back to by considering additional attributes to be explored in subsequent runs. Thus, GrASP can significantly expedite the research process, especially when addressing novel tasks.

Finally, we would like to hint on a sequel work that demonstrates how GrASP can be easily modified to address another important task – detecting the claim boundaries within its surrounding sentence (see italic text in the examples in Section 1). To cope with this unique task, we enhance the term representation (Section 3.1), by tripling each attribute a to distinguish between its appearance before (`PRE-a`), within (`IN-a`), or after (`POST-a`) the candidate claim boundaries. With this change only, GrASP was able to identify patterns for this new task, that were used to indicate the boundaries of a claim with promising preliminary results.

Acknowledgments

The authors thank Matan Ninio for giving the algorithm its catchy name.

[8] We reproduced their work to perform significant test and report the additional measures.

References

Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining sequential patterns. In *International Conference on Data Engineering*, pages 3–14.

Ehud Aharoni, Anatoly Polnarov, Tamar Lavee, Daniel Hershcovich, Ran Levy, Ruty Rinott, Dan Gutfreund, and Noam Slonim. 2014. A benchmark dataset for automatic detection of claims and evidence in the context of controversial topics. In *First Workshop on Argumentation Mining*, pages 64–68.

Mary Elaine Califf and Raymond J. Mooney. 2003. Bottom-up relational learning of pattern matching rules for information extraction. *Journal of Machine Learning Research*, pages 177–210.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Massachusetts.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING*, pages 539–545.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177.

Nitin Jindal and Bing Liu. 2006. Mining comparative sentences and relations. In *AAAI*, pages 1331–1336.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *COLING*, pages 1489–1500, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Marco Lippi and Paolo Torroni. 2015. Context-independent claim detection for argument mining. In *IJCAI*, pages 185–191.

Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology (TOIT)*, 16(2):10.

Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for naive bayes text classification. In *AAAI workshop on learning for text categorization*, pages 41–48. AAAI Press.

Michael C McCord and Arendse Bernth. 2010. Using slot grammar. *IBM TJ Watson Res. Center, Yorktown Heights, NY, IBM Res. Rep. RC23978*.

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

Thomas M. Mitchell. 1997. *Machine Learning*. McGraw-Hill, Inc.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: the detection, classification and structure of arguments in text. In *Artificial Intelligence and Law*, pages 98–109.

Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *EMNLP*, pages 105–112.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence - an automatic method for context dependent evidence detection. In *EMNLP*, pages 440–450, Lisbon, Portugal. Association for Computational Linguistics.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.

Roman Yangarber, Ralph Grishman, and Pasi Tapanainen. 2000. Unsupervised discovery of scenario-level patterns for information extraction. In *ANLP-NAACL*.

Patterns of Argumentation Strategies across Topics

Khalid Al-Khatib Henning Wachsmuth Matthias Hagen Benno Stein
Faculty of Media, Bauhaus-Universität Weimar, Germany
<firstname>.<lastname>@uni-weimar.de

Abstract

This paper presents an analysis of argumentation strategies in news editorials within and across topics. Given nearly 29,000 argumentative editorials from the New York Times, we develop two machine learning models, one for determining an editorial's topic, and one for identifying evidence types in the editorial. Based on the distribution and structure of the identified types, we analyze the usage patterns of argumentation strategies among 12 different topics. We detect several common patterns that provide insights into the manifestation of argumentation strategies. Also, our experiments reveal clear correlations between the topics and the detected patterns.

1 Introduction

Most current research in computational argumentation addresses argument mining, i.e., the identification of pro and con arguments in a text. Computational approaches that study how to deliver the arguments *persuasively* are still scarce — despite the importance of such studies for envisaged applications that deal with the synthesis of effective argumentation, such as debating systems.

Many studies have indicated that it is important to follow a specific *strategy* of how to deliver arguments in order to achieve persuasion in argumentative texts, and they proposed models for possible strategies. A recent work in this direction models the argumentation strategy of a text as an author's decision on what types of *evidence* to include in the text as well as on how to order them (Al-Khatib et al., 2016). This is in line with studies in communication theory, where many experiments have been conducted on the persuasiveness of different evidence types (Hornikx, 2005) and their combinations (Allen and Preiss, 1997).

Based on the model of Al-Khatib et al. (2016), the paper at hand investigates the usage patterns of argumentation strategies within and across topics. The study is rooted in our hypotheses that (1) effective strategies for synthesizing an argumentative text can be derived from the analysis of existing strategies that humans use in high-quality texts, and (2) the decision for preferring one strategy over another is affected by several text characteristics such as genre, provenance, and *topic*.

We approach our study within three steps. Starting from a collection of argumentative news editorials, we (1) categorize the editorials into n topics, (2) identify the evidence types (*statistics, testimony, anecdote*) in each editorial, and (3) analyze the selection and ordering of evidence types within editorials across topics. The output of these steps will be beneficial for synthesizing an effective argumentative text for a given topic (see Figure 1). The first two steps are carried out with supervised learning based on selected linguistic features, whereas the third step quantifies the distribution of evidence types and their *flows* (Wachsmuth et al., 2015).

To evaluate our approach, experiments are conducted on 28,986 editorials extracted from the *New York Times (NYT) Annotated Corpus* (Sandhaus, 2008). We automatically categorize these editorials into 12 coarse-grained topics (such as economics, arts, health, etc.). Our results expose significant differences in the distribution of evidence types across the 12 topics. Furthermore, they discriminate a number of flows of evidence types which are common in editorials. Both results provide insights into what patterns of argumentation strategies exist in editorials across different topics.

To foster future research on evidence identification and argumentation strategies, the topic categorization of all editorials as well as the developed evidence classifier are publicly available at http://www.webis.de.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1351–1357
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

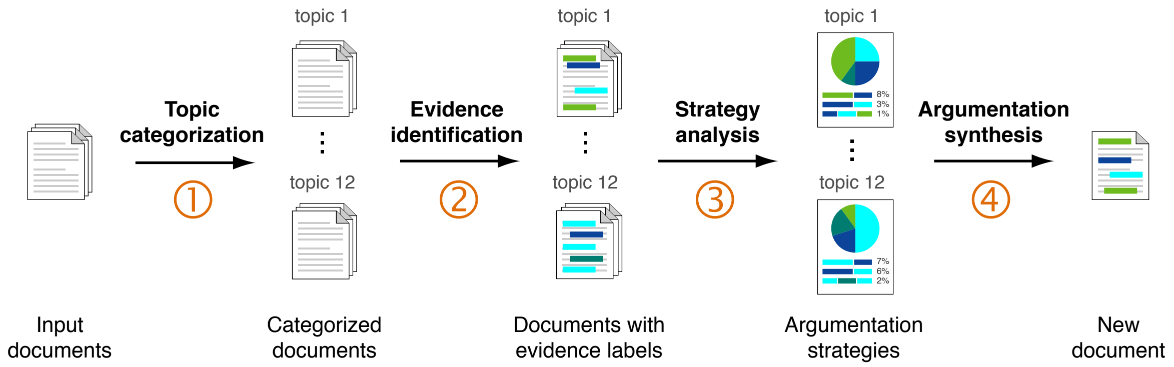


Figure 1: Four major steps of an envisioned system for synthesizing argumentative text with a particular strategy. This paper present approaches to the first three steps, whereas the fourth is left to future work.

2 Topic Categorization

The NYT Annotated Corpus comprises about 1.8 million articles published by the New York Times between 1987 and 2007. The corpus covers several types of articles that mainly categorized into 12 topics (the topics are given in Table 3) according to which section or sub-section the article is placed into in the news portal's hierarchy. Each article comes with 48 metadata tags that were assigned manually or semi-automatically by employees of the NYT. The tags cover several types of information such as *types of material* (e.g., review, editorial, etc.) and *taxonomic classifiers* (the hierarchy of articles section), among others.

All 28,986 articles tagged as "editorial" are used in our analysis. However, identifying an editorial's topic is not straightforward: While the NYT classifies the topic of most non-editorial articles, only 6% of all editorials are provided with topic information. The remaining 94% are labeled as "opinion". Analyzing the corpus, we observed that several tags include terms that describe the content of an article, such as "global warming". Some terms even include the topic itself, such as "Politics and Government". Thus, we exploited these tags to develop a standard supervised classifier for the topic categorization of editorials. In particular, we trained the classifier on all 1.29 million non-editorial articles that are assigned a topic, and then used it to classify editorials with unknown topic.

We used the default configuration of the Weka Naïve Bayes multinomial model with unigram features (Hall et al., 2009), as related studies suggest that this classifier performs particularly well in topic categorization (Husby and Barbosa, 2012). Since articles may have more than one topic, we label each article with all topics given a probability of at least 0.3 by the classifier. This threshold has been selected based on the training data.

The 6% of editorials, which are provided with "topic" labels in the corpus, were used for testing the effectiveness of our topic classifier. The classifier obtained an accuracy of 0.82 on these articles.

3 Evidence Identification

This section describes and evaluates our approach for identifying evidence types in an editorial.

All experiments are based on the corpus of Al-Khatib et al. (2016), which contains 300 editorials from three news portals: The Guardian, Al Jazeera, and Fox News. Each of these editorials is separated into argumentative segments, and every segment is labeled with one of six types. Three types refer to evidence: (1) *statistics*, where the segment states or quotes the results or conclusions of quantitative research, studies, empirical data analyses, or similar, (2) *testimony*, where the segment states or quotes that a proposition was made by some expert, authority, witness, group, organization, or similar, and (3) *anecdote*, where the segment states personal experience of the author, a concrete example, an instance, a specific event, or similar. We use the labels of all three evidence types, whereas we consider all remaining types in the corpus (e.g., *assumption*) as belonging to the type *other*.

Each segment in the corpus spans one sentence or less. Accordingly, it is possible that a sentence includes multiple types (e.g., *testimony* and *statistics*), although the proportion of such sentences is very low (less than 5%). We hence decided to simplify the task by identifying only one type for each sentence; in case a sentence has more than one type, we favor evidence types over *other*, and less frequent evidence types over more frequent ones.

Thereby, we avoid dealing with argumentative text segmentation and multi-type classification.

For identifying evidence types, we rely on supervised learning. The task is similar to tasks concerned with the pragmatic level of text, such as language function analysis (Wachsmuth and Bujna, 2011) or speech act classification (Ferschke et al., 2012). We employ several features that capture the content, syntax, style, and semantics of a sentence. Some of them have been used for the mentioned tasks, others are tailored to our task—based on our inspection of the training set of the corpus.

Lexical Features Previous work on speech acts classification showed a strong positive impact of lexical features, e.g., (Jeong et al., 2009). In case of evidence types, words such as "study" and "find" are indicators for *statistics*,"according" and "states" for *testimony*, and "example" and "year" for *anecdote*, for instance. We represent this feature type as the frequency of word unigrams, bigrams, and trigrams. We also consider punctuation and digits in our features; quotes play an important role for *testimony*, numbers for *statistics*.

Style Features We hypothesize that texts with different evidence types show specific style characteristics. To test this, we use character 1–3-grams, chunk 1–3-grams, function word 1–3-grams, and the first 1–3 tokens in a sentence. Similarly, we expect *anecdote* and *testimony* sentences to be longer than *statistics*, which we capture by the number of characters, syllables, tokens, and phrases in a sentence. Moreover, we assess whether a sentence is the first, second, or last within a paragraph.

Syntactic Features Syntax plays a role in different linguistic tasks. For evidence type identification, narrative tenses may be indicators of anecdotes, for instance. We model syntax simply via the frequencies of part of speech tag 1–3-grams.

Semantic Features We use the frequency of person, location, organization, and misc entities, as well as the proportion of each of these entity types. In many cases, a sentence with evidence refers to specific entities (e.g., a scientific lab in *statistics*). Also, we use the mean SentiWordNet score of the words in a sentence, once for the word's first sense and once for its average sense (Baccianella et al., 2010). Moreover, we compute the frequency of each word class of the *General Inquirer* (`http://www.wjh.harvard.edu/~inquirer`).

In our experiments, the sequential minimal optimization (SMO) implementation of support vector machines from Weka performed best among several models on the validation set of the given corpus. There, SMO achieved the highest results for a cost hyperparameter value of 5, which we then used to evaluate SMO on the test set.

#	Feature Type	Accuracy	F_1-Score
1	Lexical features	0.76	0.73
2	Style features	0.74	0.70
3	Syntactic features	0.74	0.71
4	Semantics features	0.71	0.67
1 – 4	**Complete feature set**	**0.78**	**0.77**
	Majority baseline	0.69	0.56

Table 1: Effectiveness of each feature type and the complete feature set in identifying evidence types.

Type	Precision	Recall	F_1-Score
Statistics	0.69	0.40	0.50
Testimony	0.63	0.55	0.59
Anecdote	0.55	0.47	0.51
Other	0.84	0.90	0.87

Table 2: Precision, recall, and F_1-Score for all four classes in the identification of evidence types.

Results Table 1 shows the effectiveness of our classifier in terms of accuracy and weighted average F_1-score for each single feature type as well as for the complete feature set. In general, lexical features are the most discriminative, closely followed by the syntax features. All feature types contribute to the effectiveness of the complete feature set. Table 2 shows the precision, recall, and F_1-score values for classifying each of the three evidence types as well as the class *other*. The classifier achieved the highest F_1-score for *other*, followed by *testimony*, *anecdote*, and *statistics* respectively.

Error Analysis The classifier has a small tendency towards labeling sentences with the majority class *other*. However, sampling the training set yielded worse results for all classes. Overall, the task is challenging, and the results we obtained are in line with those that have been reported in speech act classification. Also, the decision to classify each sentence with one of the evidence classes (to avoid segmentation) may render the type identification itself harder. For example, some features such as quotation marks can be helpful to identify *testimony*. However, if some *testimony* evidence covers several sentences, the ones which are between the first and the last sentences might be difficult to be identified as part of the *testimony*.

Evidence Type		All	Arts	Econ.	Edu.	Envir.	Health	Law	Polit.	Relig.	Science	Sports	Style	Tech.
AN	Anecdote	24.9	**31.6**	22.1	24.1	25.7	21.9	27.5	24.4	31.1	24.9	31.1	29.7	23.7
TE	Testimony	7.7	**11.3**	6.2	9.6	5.1	5.7	7.4	8.4	10.8	6.3	6.5	7.1	6.3
ST	Statistics	3.0	1.5	**5.0**	4.4	3.4	4.9	2.7	2.1	1.8	3.0	2.8	2.3	2.3
OT	Other	64.4	55.6	66.7	62.0	65.8	67.5	62.4	65.1	56.3	65.8	59.6	60.9	**67.7**
Editorials		28986	1274	3158	1977	1687	2524	2327	12912	243	455	953	960	516

Table 3: Distribution of the four evidence types in all editorials and in those of each topic, given in percent. The bottom line shows the number of editorials of each topic.Values discussed in Section 4 are in bold.

4 Argumentation Strategy Analysis

In this section, we analyze strategy patterns across editorials of 12 topics, exploring the selection and ordering based on the distribution and sequential flows of evidence types respectively.

To this end, we applied our topic and evidence type classifiers to all given 28,986 NYT editorials. As the analysis of argumentation strategies depends strongly on the effectiveness of evidence type identification, we consider the impact of classification errors in the analysis results as follows. For each evidence type t in dataset d, we compute a confidence interval $[lower\ bound, upper\ bound]$ for the n sentences that the classifier labels with t. The interval is derived from the precision and recall of our classifier for type t (determined on the ground truth): We compute the lower bound as $n \cdot precision(t)$ and the upper bound as $n/recall(t)$.

Based on the mean of $lower\ bound$ and $upper\ bound$, we perform a significance test among the evidence type distribution across topics. In particular, we use the chi-square statistical method with a significance level of 0.001. For the sequential flows, however, a consideration of the impact of misclassified sentences seems unreliable: As each editorial is represented by only one flow, the 60 editorials in the test set of Al-Khatib et al. (2016) are not enough for computing precision and recall. In contrast, we again use chi-square with a significance level of 0.001 for specifying significant differences among the flows.

Distribution of Evidence Types Altogether, the given 28,986 editorials contain 669,092 sentences whose type we classified. As Table 3 shows, the most frequent type is *other* (64.4%) according to our classifier, followed by *anecdote* (24.9%), *testimony* (7.7%), and *statistics* (3.0%).

In terms of the performed chi-squared tests, all pairs of topic-specific type distributions in Table 3 are significantly different from each other with only one exception: *arts* and *religion*. This results strongly support the hypothesis that topic influences the usage of evidence types. For anecdotes, the values of both *science* and *technology* differ not significantly from *all*. For testimony, *law* does not differ significantly from *all*, and for statistics, the analog holds for *science* and *sports*.

The highest relative frequency of anecdotes is observed for *arts* (31.6%) and *religion* (31.1%), followed by *sports* (31.1%). Matching intuition, authors of *arts* and *religion* editorials add much testimony evidence (11.3% and 10.8% respectively). In contrast, anecdotes and testimony are clearly below the average for *health*, while statistics play a more important role there with 4.9%, the second highest percentage after *economy* (5.0%).

Sequential Flows of Evidence Types Following related research (Wachsmuth et al., 2015), we designate the *flow* here as a sequential representation of all evidence types in an editorial. Following one the flow generalizations proposed by Wachsmuth et al. (2015), we abstract flows considering only changes of evidence types. For example, the flow (AN, AN, TE) for an editorial will be abstracted into (AN, TE). Such an abstraction produces more frequent and thus reliable patterns. Table 4 lists the resulting *evidence change flows* that are most common among all editorials.

The most frequent flow is (AN), representing 16.6% of all editorials across topics. This means that about one sixth of all editorials contain only this evidence type. The frequency of (AN) ranges from 9.3% (*education*) to 26.7% (*style*), revealing the varying importance of anecdotes in editorials of different topics. The frequency of (AN, TE, AN) is more stable across topics; only *health* and *technology* show notably lower values there (8.8% and 9.5% respectively). For *technology*, the percentage is much above the average for some other flows based on AN and TE, such as (AN, TE) (10.7% vs. 6.9%) and (TE, AN) (4.3% vs. 2.6%). Hence, the ordering of evidence seems to make a difference.

#	Evidence Change Flow	All	Arts	Econ.	Edu.	Envir.	Health	Law	Polit.	Relig.	Science	Sports	Style	Tech.
1	(AN)	**16.6**	16.0	13.5	**9.3**	21.3	17.4	17.4	16.2	11.9	20.4	21.8	**26.7**	20.9
2	(AN, TE, AN)	13.2	13.5	10.3	10.2	11.6	**8.8**	14.7	15.1	14.0	13.2	14.1	15.5	**9.5**
3	(AN, TE)	**6.9**	7.9	4.6	7.5	5.9	6.7	8.1	7.0	7.8	7.7	7.2	7.0	**10.7**
4	(AN, ST, AN)	**5.3**	3.6	6.7	4.1	**8.6**	7.2	6.2	4.2	4.9	6.8	7.3	5.8	4.7
5	(AN, TE, AN, TE, AN)	5.3	8.4	3.4	4.3	3.9	2.4	6.4	6.3	7.0	3.1	4.6	3.5	6.6
6	(AN, TE, AN, TE)	4.9	6.2	3.3	4.9	3.5	3.2	5.3	5.7	8.2	4.0	4.8	4.0	4.3
7	(TE, AN)	**2.6**	2.4	2.2	2.3	1.7	2.5	1.8	3.0	<0.5	2.2	2.4	2.3	**4.3**
8	(AN, ST)	2.2	0.7	3.8	1.9	3.1	4.3	2.4	1.5	1.2	3.1	1.9	2.0	1.6
9	(AN, TE, AN, TE, AN, TE)	2.2	2.9	1.3	1.8	1.2	1.1	2.8	2.8	2.9	0.7	1.3	1.7	1.4
10	(AN, TE, AN, TE, AN, TE, AN)	2.0	4.3	1.5	1.8	0.8	1.0	1.9	2.3	5.8	1.3	1.6	1.7	1.4
11	(TE, AN, TE, AN)	1.8	2.2	0.9	2.5	0.7	1.0	1.5	2.3	2.1	0.7	1.8	1.4	1.0
12	(AN, ST, AN, TE, AN)	1.4	0.9	1.8	1.6	2.4	1.2	0.9	1.3	0.8	2.2	1.8	1.3	0.6
13	(ST, AN)	1.3	<0.5	2.2	1.0	1.3	2.8	1.2	0.9	<0.5	2.0	0.7	1.4	2.3
14	(TE, AN, TE)	1.3	1.4	0.8	1.4	0.7	0.9	1.1	1.6	2.1	1.5	<0.5	0.6	1.9
15	(AN, ST, AN, TE)	1.2	0.7	1.6	1.5	2.0	1.5	1.4	1.0	1.2	0.9	1.3	0.9	1.4

Table 4: Relative frequency of the top 15 evidence change flows in all editorials and in those of each topic, given in percent. In the flows, the type *Other* is ignored. Values discussed in Section 4 are in bold.

In accordance with literature on argumentation in editorials (van Dijk, 1995), many common flows start with an anecdote and end with one. While testimony occurs most often between the anecdotes, the fourth most frequent flow is (AN, ST, AN) (5.3%). This flow occurs particularly often in editorials about *environment* (8.6%), even though statistics are not that frequent in these editorials (see Table 4) — and similar holds for (AN, ST). Such observations emphasize the role of topic on ordering decisions in argumentation strategies.

5 Related Work

In addition to the work on argumentation strategies in editorials (Al-Khatib et al., 2016) that we have discussed in Section 3, several approaches have been proposed for modeling and identifying the types or roles of argumentative units. For instance, Stab and Gurevych (2014) distinguish premises from claims and major claims, and Park and Cardie (2014) unverifiable from verifiable statements.

In this line of research, Rinott et al. (2015) have proposed a supervised learning model for identifying context-dependent evidence in Wikipedia articles. While the authors target the same evidence types that we consider in our work, they approach a different task. In particular they classify only evidence that is *related to given claims*. Hence, a comparison of their effectiveness results with ours would be meaningless. Moreover, some of their features rely on resources that are not publicly available (e.g., lexicons), which is why could not resort to their approach or compare it to ours.

The NYT Annotated Corpus has been analyzed in several papers. Among others, Li et al. (2016) and Hong and Nenkova (2014) used the metadata tag *abstract*, which contains a manually created article summary. Other tags, such as those for people, locations, and organizations mentioned in an article, have been used by Dunietz and Gillick (2014).

6 Conclusion

This paper has studied argumentation strategies in news editorials of different topics. We have observed varying distributions of evidence types across the topics as well as varying sequential flows of these types. Overall, our analysis has revealed several patterns of how authors argue in news editorials, and how the topic influences such patterns. We believe that the obtained results provide valuable insights for research on the synthesis of effective argumentative texts.

Besides text synthesis, we consider this study as beneficial for argument mining as well as for the topic categorization of argumentative texts. It provides insights and empirical results on prior knowledge regarding distributional and structural probabilities for evidence usage among topics. Our findings can be incorporated into unsupervised classification models (Hu et al., 2015).

In future work, we plan to investigate argumentation strategies across different genres and provenances. Also, we will further explore whether there are important types of evidence in editorials and similar texts that we have not considered in this paper so far, such as analogies.

References

Khalid Al-Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, and Benno Stein. 2016. A News Editorial Corpus for Mining Argumentation Strategies. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, pages 3433–3443. http://aclweb.org/anthology/C16-1324.

Mike Allen and Raymond W. Preiss. 1997. Comparing the Persuasiveness of Narrative and Statistical Evidence using Meta-Analysis. *Communication Research Reports* 14(2):125–131.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Languages Resources Association (ELRA). http://aclweb.org/anthology/L10-1531.

Jesse Dunietz and Daniel Gillick. 2014. A New Entity Salience Task with Millions of Training Examples. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*. Association for Computational Linguistics, pages 205–209. https://doi.org/10.3115/v1/E14-4040.

Oliver Ferschke, Iryna Gurevych, and Yevgen Chebotar. 2012. Behind the Article: Recognizing Dialog Acts in Wikipedia Talk Pages. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 777–786. http://aclweb.org/anthology/E12-1079.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations* 11(1):10–18.

Kai Hong and Ani Nenkova. 2014. Improving the Estimation of Word Importance for News Multi-Document Summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 712–721. https://doi.org/10.3115/v1/E14-1075.

Jos Hornikx. 2005. A Review of Experimental Research on the Relative Persuasiveness of Anecdotal, Statistical, Causal, and Expert Evidence. *Studies in Communication Sciences* 5(1):205–216.

Linmei Hu, Juanzi Li, Xiaoli Li, Chao Shao, and Xuzhong Wang. 2015. TSDPMM: Incorporating Prior Topic Knowledge into Dirichlet Process Mixture Models for Text Clustering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 787–792. https://doi.org/10.18653/v1/D15-1091.

Stephanie Husby and Denilson Barbosa. 2012. Topic Classification of Blog Posts Using Distant Supervision. In *Proceedings of the Workshop on Semantic Analysis in Social Media*. Association for Computational Linguistics, pages 28–36. http://aclweb.org/anthology/W12-0604.

Minwoo Jeong, Chin-Yew Lin, and Geunbae Gary Lee. 2009. Semi-supervised Speech Act Recognition in Emails and Forums. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1250–1259. http://aclweb.org/anthology/D09-1130.

Jessy Junyi Li, Kapil Thadani, and Amanda Stent. 2016. The Role of Discourse Units in Near-Extractive Summarization. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, pages 137–147. http://aclweb.org/anthology/W16-3617.

Joonsuk Park and Claire Cardie. 2014. Identifying Appropriate Support for Propositions in Online User Comments. In *Proceedings of the First Workshop on Argumentation Mining*. Association for Computational Linguistics, pages 29–38. https://doi.org/10.3115/v1/W14-2105.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, M. Mitesh Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show Me Your Evidence - An Automatic Method for Context Dependent Evidence Detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 440–450. https://doi.org/10.18653/v1/D15-1050.

E. Sandhaus. 2008. The New Xork Times Annotated corpus. Corpus number LDC2008T19. In *Linguistic Data Consortium, Philadelphia*.

Christian Stab and Iryna Gurevych. 2014. Identifying Argumentative Discourse Structures in Persuasive Essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 46–56. https://doi.org/10.3115/v1/D14-1006.

Teun A. van Dijk. 1995. Opinions and Ideologies in Editorials. In *Proceedings of the 4th International Symposium of Critical Discourse Analysis, Language, Social Life and Critical Thought*. Athens.

Henning Wachsmuth and Kathrin Bujna. 2011. Back to the Roots of Genres: Text Classification by Language Function. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, pages 632–640. http://aclweb.org/anthology/I11-1071.

Henning Wachsmuth, Johannes Kiesel, and Benno Stein. 2015. Sentiment Flow — A General Model of Web Review Argumentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 601–611. https://doi.org/10.18653/v1/D15-1072.

Using Argument-based Features to Predict and Analyse Review Helpfulness

Haijing Liu[1,2]**, Yang Gao**[1]**, Pin Lv**[1]**, Mengxue Li**[1,2]**, Shiqiang Geng**[3]**,**
Minglan Li[1,2] **and Hao Wang**[1]

[1]Institute of Software, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]Beijing Information Science and Technology University
{*haijing2015, gaoyang, lvpin, mengxue2015*}*@iscas.ac.cn*
gsqwxh@163.com,{*minglan2015, wanghao*}*@iscas.ac.cn*

Abstract

We study the helpful product reviews identification problem in this paper. We observe that the evidence-conclusion discourse relations, also known as *arguments*, often appear in product reviews, and we hypothesise that some *argument*-based features, e.g. the percentage of argumentative sentences, the evidences-conclusions ratios, are good indicators of helpful reviews. To validate this hypothesis, we manually annotate arguments in 110 hotel reviews, and investigate the effectiveness of several combinations of argument-based features. Experiments suggest that, when being used together with the argument-based features, the state-of-the-art baseline features can enjoy a performance boost (in terms of F1) of 11.01% in average.

1 Introduction

Product reviews have significant influences on potential customers' opinions and their purchase decisions (Chatterjee, 2001; Chen et al., 2004; Dellarocas et al., 2004). Instead of reading a long list of reviews, customers usually are only willing to view a handful of *helpful reviews* to make their purchase decisions. In other words, helpful reviews have even greater influences on the potential customers' decision-making processes and thus on the sales; as a result, the automatic identification of helpful reviews has received considerable research attentions in recent years (Kim et al., 2006; Liu et al., 2008; Mudambi, 2010; Xiong and Litman, 2014; Martin and Pu, 2014; Yang et al., 2015, 2016).

Existing works on helpful reviews identification mostly focus on designing efficient features. Widely used features include *external features*, (e.g. date (Liu et al., 2008), product rating (Kim et al., 2006) and product type (Mudambi, 2010)) and *intrinsic features* (e.g. semantic dictionaries (Yang et al., 2015) and emotional dictionaries (Martin and Pu, 2014)). Compared to external features, intrinsic features can provide some insights and explanations for the prediction results, and support better cross-domain generalisation. In this work, we investigate a new form of intrinsic features: the *argument* features.

An argument is a basic unit people use to persuade their audiences to accept a particular state of affairs (Eckle-Kohler et al., 2015). An argument usually consists of a *claim* (also known as *conclusion*) and some *premises* (also known as *evidences*) offered in support of the claim. For example, consider the following review excerpt: "*The staff were amazing, they went out of their way to help us*"; the texts before the comma constitute a claim, and the texts after the comma give a premise supporting the claim. Argumentation mining (Moens, 2013; Lippi and Torroni, 2016) receives growing research interests in various domains (Palau and Moens, 2009; Contractor et al., 2012; Park and Cardie, 2014; Madnani et al., 2012; Kirschner et al., 2015; Wachsmuth et al., 2014, 2015). Recent advances in automatic arguments identification (Stab and Gurevych, 2014), has stimulated the usage of argument features in multiple domains, e.g. essay scoring (Wachsmuth et al., 2016) and online forum comments ranking (Wei12 et al., 2016).

The motivation of this work is a hypothesis that, the helpfulness of a review is closely related to some argument-related features, e.g. the percentage of argumentative sentences, the average number of premises in each argument, etc. To validate our hypothesis, we manually annotate arguments in 110 hotel reviews so as to use

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1358–1363
Copenhagen, Denmark, September 7–11, 2017.

these “ground truth” arguments to testify the effectiveness of argument-based features for detecting helpful hotel reviews. Empirical results suggest that, for four baseline feature sets we test, their performances can be improved, in average, by 11.01% in terms of F1-score and 10.40% in terms of AUC when they are used together with some argument-based features. Furthermore, we use the effective argument-based features to give some insights into which product reviews are more helpful.

2 Corpus

We use the Tripadvisor hotel reviews corpus built by (O’Mahony and Smyth, 2010) to test the performance of our helpful reviews classifier. Each entry in this corpus includes the review texts, the number of people that have viewed this review (denoted by Y) and the number of people that think this review is helpful (denoted by X).

We randomly sample 110 hotel reviews from this corpus to annotate the “ground truth” argument structures [1]. In line with (Wachsmuth et al., 2015), we view each sub-sentence in the review as a *clause* and ask three annotators independently to annotate each clause as one of the following seven argument components:

Major Claim: a summary of the main opinion of a review. For instance, *“I have enjoyed the stay in the hotel”, “I am sad to say that i am very disappointed with this hotel”*;

Claim: a subjective opinion on a certain aspect of a hotel. For example, *“The staff was amazing”, “The room is spacious”*;

Premise: an objective reason/evidence supporting a claim. For instance, *“The staff went out of their way to help us”*, it supports the first example claim above; *“We had a sitting room as well as a balcony”*, it supports the second example claim above;

Premise Supporting an Implicit Claim (PSIC): an objective reason/evidence that supporting an *implicit claim*, which does appear in review. For instance, *“just five minutes’ walk to the down town”* supports some implicit claims like *“the location of the hotel is good”*, although this implicit claims has never appeared in the review;

Background: an objective description that does not give direct opinions but provides some background information. For example, *“We checked into this hotel at midnight”, “I stayed five nights at this hotel because i was attending a conference at the hotel”*;

Component Type	Number	Kappa
Major claim	143	0.86
Claim	581	0.77
Premise	206	0.65
PSIC	121	0.94
Background	80	0.89
Recommendation	70	1.00
Non-argumentative	145	0.78

Table 1: The number and Fleiss’ kappa for each argument component type we annotate.

Recommendation: a positive or negative recommendation for the hotel. For instance, *“I would definitely come to this hotel again the next time I visit London”, “Do not come to this hotel if you look for some clean places to live”*;

Non-argumentative: for all the other clauses.

We use the Fleiss’ kappa metric (Fleiss, 1971) to evaluate the quality of the obtained annotations, and the results are presented in Table 1. We can see that the lowest Kappa scores (for Premise) is still above 0.6, suggesting that the quality of the annotations are *substantial* (Landis and Koch, 1977); in other words, there exist little noises in the ground truth argument structures. We aggregate the annotations using majority voting.

3 Features

In line with (Yang et al., 2015), we consider the helpfulness as an intrinsic characteristic of product reviews, and thus only consider the following four intrinsic features as our baseline features.

Structural features (**STR**) (Kim et al., 2006; Xiong and Litman, 2014): we use the following structural features: total number of tokens, total number of sentences, average length of sentences, number of exclamation marks, and the percentage of question sentences.

Unigram features (**UGR**) (Kim et al., 2006; Xiong and Litman, 2014): we remove all stopwords and non-frequent words ($tf < 3$) to build the unigram vocabulary. Each review is represented by the vocabulary with *tf-idf* weighting for each appeared term.

Emotional features (**GALC**) (Martin and Pu, 2014): the Geneva Affect Label Coder (GALC) dictionary proposed by (Scherer, 2005) defines 36 emotion states distinguished by words. We build a

[1] The annotated corpus can be obtained by contacting the first author

real feature vector with the number of occurrences of each emotional word plus one additional dimension for the number of non-emotional words.

Semantic features (**INQUIRER**) (Yang et al., 2015): the General Inquirer (INQUIRER) dictionary proposed by (Stone et al., 1962) maps each word to some semantic tags, e.g. word *absurd* is mapped to tags NEG and VICE; similar to the GALC features, the semantic features include the number of occurrences of each semantic tag.

3.1 Argument-based Features

The argument-based features can have different granularity: for example, the number of argument components can be used as features, and the number of tokens (words) in the argument components can also be used as features. We consider four granularity of argument features, detailed as follows.

Component-level argument features. A natural feature that we believe to be useful is the ratio of different argument component numbers. For example, we may be interested in the ratio between the number of premises and that of claims; a high ratio suggests that there are more premises supporting each claim, indicating that the review gives many evidences. To generalise this component ratio feature, we propose *component-combination ratio* features: we compute the ratios between any two argument components combinations. For example, we may be interested in the ratio between the number of MajorClaim+Claim+Premise and that of Background+Non-argumentative. As there are 7 types of labels, the number of possible combinations is $2^7 - 1 = 127$, and thus the possible number of combination ratio pairs is $127 \times 126 = 16002$. In other words, the component-level feature is a 16002-dimensional real vector.

Token-level argument features. In a finer-granularity, we consider the number of tokens in argument components to build features: for example, suppose a review has only two claims, one has 10 words and the other has 5 words; we may want to know the average number of words contained in each claim, the total number of words in claims, etc. In total, for each argument component type, we consider 5 types of token-level statistics: the total number of words in the given component type, the length (in terms of word) of the shortest/longest component of the given type, and the mean/variance of the number of words in each component of the given type. Thus, there are in total $7 \times 5 = 35$ features to represent the token-level statistics.

In addition, the ratio of some token-level statistics may also be of interests: for example, given a review, we may want to know the ratio between the number of words in Claims+MajorClaims and that in Premises. Thus, the combination ratio can also be applied here. We consider only the combination ratio for two statistics: the total number of words and the average number of words in each component-combination; hence, there are $16002 \times 2 = 32004$ dimensions for the combination ratio for the statistics. In total, there are $32004 + 35 = 32039$ dimensions for the token-level argument features.

Letter-level argument features. In the finest-granularity, we consider the letter-level features, which may give some information the token-level features do not contain: for example, if a review has a big number of letters and a small number of words, it may suggests that many long and complex words are used in this review, which, in turn, may suggests that the linguistic complexity of the review is relative high and the review may gives some very professional opinions. Similar to the token-level features above, we design 5 types of statistics and their combination ratios. Thus, the dimension for the letter-level features is the same to that of the token-level features.

Position-level argument features. Another dimension to consider argument features is the positions of argument components: for example, if the major claims of a review are all at the very beginning, we may think that readers can more easily grasp the main idea of the review and, thus, the review is more likely to be helpful. For each component, we use a real number to represent its position: for example, if a review has 10 sub-sentences (i.e. clauses) in total and the first sub-sentence the component overlaps is the second sub-sentence, then the position for this component is $2/10 = 0.2$. For each type of argument component, we may be interested in some statistics for its positions: for example, if a review has several premises, we may want to know the location of the earliest/latest appearance of premises, the average position of all premises and its variance, etc. Similar to the token- and letter-level features, we design the same number of features for position-

	Accuracy	Precision	Recall	F1-score	AUC
AF	0.617	0.625	0.617	0.620	0.611
STR	0.600	0.360	0.600	0.450	0.500
STR+AF	0.604	0.614	0.604	0.607	0.599
UGR	0.697	**0.760**	0.697	0.646	0.627
UGR+AF	**0.718**	0.718	**0.719**	**0.717**	**0.706**
GALC	0.621	0.605	0.621	0.579	0.560
GALC+AF	0.647	0.654	0.647	0.649	0.640
INQUIRER	0.533	0.512	0.533	0.517	0.493
INQUIRER+AF	0.657	0.664	0.657	0.659	0.651

Table 2: Helpful reviews identification performances using argument-based features and/or baseline features. AF stands for argument-based features.

level features.

4 Experiments

Following (O'Mahony and Smyth, 2010; Martin and Pu, 2014), we model the helpfulness prediction task as a classification problem; thus, we use accuracy, precision, recall, macro F1 and area under the curve (AUC) to as evaluation metrics. Similar to (O'Mahony and Smyth, 2010), we consider a review as helpful if and only if at least 75% opinions for the review are positive, i.e. $X/Y \geq 0.75$ (see X and Y in Sect. 2). For the features whose number of dimensions is more than 10k (i.e. the UGR features and argument-based features), to reduce their dimensions and to improve the performance, we only use the positive-information-gain features in these feature sets. In line with most existing works on helpfulness prediction (Martin and Pu, 2014; Yang et al., 2015), we use the LibSVM (Chang and Lin, 2011) as our classifier.

The performances of different features are presented in Table 2. Each number in the table is the average performance in 10-fold cross-validation tests. From the table we can see that, when being used together with the argument-based features, either of the four baseline features enjoys a performance boost in terms of all metrics we consider. To be more specific, in terms of accuracy, precision, recall, F1 and AUC, the average improvement for the baseline features are 4.33%, 10.30%, 4.32%, 11.01% and 10.40%, respectively. However, we observe that the precision of UGR+AF, although gives the second highest score among all feature combinations, is lower than that of UGR; we leave it for future work. Also, we notice that when using the argument-based features alone, its performance (in terms of Precision, F1 and AUC) is superior to those of STR, GALC and INQUIRER, and is only inferior to UGR. However, a major drawback of the UGR feature is its huge and document-dependent dimensionality, while the dimensionality of argument-based features is fixed, regardless of the size of the input documents. Moreover, the UGR features are sparse and problematic in online learning. To summarise, compared with the other state-of-the-art features, argument-based features are effective in identifying helpful reviews, and can represent some complementary information that cannot be represented in other features.

5 What Makes a Review Helpful ?

Argument-based features can not only improve the performance of review helpfulness identification, but also can be used to interpret what makes a review helpful. We analyse the information gain ranking of the argument-based features and find that, among all the positive-information-gain argument features, 36% are from the token-level argument feature set, and 29% are from the letter-level argument feature set, suggesting that these two feature sets are most effective in identifying helpful reviews. Among all the token-level argument features with positive information gain, 69% are ratios of sum of token number between component-combinations, and the remaining are ratios of the mean token numbers between component-combinations. We interpret this observation as follows: given a review, the larger number of tokens it contains, and the more likely the review is helpful. In fact, helpful reviews are tend to occur in those long reviews, which generally provide with more experiences and comments about the product being reviewed. Among all the letter-level argument features, around three-quarters are ratios of the sum of the number of letters between component-combinations. This observation, again, suggests that the length of reviews plays an important role in the review helpfulness identification.

Moreover, among all the argument-based features with positive information gain values, a quarter of features are the position-level argument feature. This is because the position of each argument component influences the logic flow of reviews, which, in turn, influences the readability, convincingness and helpfulness of the reviews. This information can hardly be represented by all the baseline features we considered, and we believe this explains why the performances of the baseline features are improved when being used together with the argument-based features. However, among all the argument-based features with positive information gain values, only 10% are the component-level argument feature. This indicates that compared to three finer-granularity argument features above, the component-level argument feature provides less useful information in review helpfulness identification.

6 Conclusion and Future Work

In this work, we propose a novel set of intrinsic features of identifying helpful reviews, namely the *argument*-based features. We manually annotate 110 hotel reviews, and compare the performances of argument-based features with those of some state-of-the-art features. Empirical results suggest that, argument-based features include some complementary information that the other feature sets do not include; as a result, for each baseline feature, the performance (in terms of various metrics) of jointly using this feature and argument-based features is higher than using this baseline feature alone. In addition, by analysing the effectiveness of different argument-based features, we give some insights into which reviews are more likely to be helpful, from an argumentation perspective.

For future work, an immediate next step is to explore the usage of automatically extracted arguments in helpful reviews identification: in this work, all argument-based features are based on manually annotated arguments; deep-learning based argument mining (Li et al., 2017; Eger et al., 2017) has produced some promising results recently, and we plan to investigate whether the automatically extracted arguments can be used to identify helpful reviews, and how the errors made in the argument extraction stage will influence the performance of helpful reviews identification. We also plan to investigate the effectiveness of argument-based features in other domains.

Acknowledgements

Yang Gao is supported by National Natural Science Foundation of China (NSFC) grant 61602453, and Hao Wang is supported by NSFC grant 61672501, 61402447 and 61502466.

References

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27.

Patrali Chatterjee. 2001. Online reviews: Do consumers use them? *ACR North American Advances*.

Pei-Yu Chen, Shin-yi Wu, and Jungsun Yoon. 2004. The impact of online recommendations and consumer feedback on sales. *ICIS 2004 Proceedings*, page 58.

Danish Contractor, Yufan Guo, and Anna Korhonen. 2012. Using argumentative zones for extractive summarization of scientific articles. In *coling*, volume 12, pages 663–678.

Chrysanthos Dellarocas, Neveen Awad, and Xiaoquan Zhang. 2004. Exploring the value of online reviews to organizations: Implications for revenue forecasting and planning. *ICIS 2004 Proceedings*, page 30.

Judith Eckle-Kohler, Roland Kluge, and Iryna Gurevych. 2015. On the role of discourse markers for discriminating claims and premises in argumentative discourse. In *EMNLP*, pages 2236–2242.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. *arXiv preprint arXiv:1704.06104*.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. 2006. Automatically assessing review helpfulness. In *Proceedings of the 2006 Conference on empirical methods in natural language processing*, pages 423–430. Association for Computational Linguistics.

Christian Kirschner, Judith Eckle-Kohler, and Iryna Gurevych. 2015. Linking the thoughts: Analysis of argumentation structures in scientific publications. In *ArgMining@ HLT-NAACL*, pages 1–11.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Minglan Li, Yang Gao, Hui Wen, Yang Du, Haijing Liu, and Hao Wang. 2017. Joint rnn model for argument component boundary detection. *arXiv preprint arXiv:1705.02131*.

Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology (TOIT)*, 16(2):10.

Yang Liu, Xiangji Huang, Aijun An, and Xiaohui Yu. 2008. Modeling and predicting the helpfulness of online reviews. In *Data mining, 2008. ICDM'08. Eighth IEEE international conference on*, pages 443–452. IEEE.

Nitin Madnani, Michael Heilman, Joel Tetreault, and Martin Chodorow. 2012. Identifying high-level organizational elements in argumentative discourse. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 20–28. Association for Computational Linguistics.

Lionel Martin and Pearl Pu. 2014. Prediction of helpful reviews using emotions extraction. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14)*, EPFL-CONF-210749.

Marie-Francine Moens. 2013. Argumentation mining: Where are we now, where do we want to be and how do we get there? In *Post-Proceedings of the 4th and 5th Workshops of the Forum for Information Retrieval Evaluation*, page 2. ACM.

Susan M Mudambi. 2010. What makes a helpful online review? astudy of customer reviews on amazon. com. *MIS Quarterly*, 34(1):185–200.

Michael P O'Mahony and Barry Smyth. 2010. A classification-based review recommender. *Knowledge-Based Systems*, 23(4):323–329.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th international conference on artificial intelligence and law*, pages 98–107. ACM.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *ArgMining@ ACL*, pages 29–38.

Klaus R Scherer. 2005. What are emotions? and how can they be measured? *Social science information*, 44(4):695–729.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *EMNLP*, pages 46–56.

Philip J Stone, Robert F Bales, J Zvi Namenwirth, and Daniel M Ogilvie. 1962. The general inquirer: A computer system for content analysis and retrieval based on the sentence as a unit of information. *Behavioral Science*, 7(4):484–498.

Henning Wachsmuth, Khalid Al Khatib, and Benno Stein. 2016. Using argument mining to assess the argumentation quality of essays. In *COLING*, pages 1680–1691.

Henning Wachsmuth, Johannes Kiesel, and Benno Stein. 2015. Sentiment flow-a general model of web review argumentation. In *EMNLP*, pages 601–611.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, and Gregor Engels. 2014. Modeling review argumentation for robust sentiment analysis. In *COLING*, pages 553–564.

Zhongyu Wei12, Yang Liu, and Yi Li. 2016. Is this post persuasive? ranking argumentative comments in the online forum. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 195.

Wenting Xiong and Diane J Litman. 2014. Empirical analysis of exploiting review helpfulness for extractive summarization of online reviews. In *COLING*, pages 1985–1995.

Yinfei Yang, Cen Chen, and Forrest Sheng Bao. 2016. Aspect-based helpfulness prediction for online product reviews. In *Tools with Artificial Intelligence (ICTAI), 2016 IEEE 28th International Conference on*, pages 836–843. IEEE.

Yinfei Yang, Yaowei Yan, Minghui Qiu, and Forrest Sheng Bao. 2015. Semantic analysis and helpfulness prediction of text for online product reviews. In *ACL (2)*, pages 38–44.

Here's My Point: Joint Pointer Architecture for Argument Mining

Peter Potash, Alexey Romanov, Anna Rumshisky
Department of Computer Science
University of Massachusetts Lowell
{ppotash,aromanov,arum}@cs.uml.edu

Abstract

In order to determine argument structure in text, one must understand how individual components of the overall argument are linked. This work presents the first neural network-based approach to link extraction in argument mining. Specifically, we propose a novel architecture that applies Pointer Network sequence-to-sequence attention modeling to structural prediction in discourse parsing tasks. We then develop a joint model that extends this architecture to simultaneously address the link extraction task and the classification of argument components. The proposed joint model achieves state-of-the-art results on two separate evaluation corpora, showing far superior performance than the previously proposed corpus-specific and heavily feature-engineered models. Furthermore, our results demonstrate that jointly optimizing for both tasks is crucial for high performance.

1 Introduction

An important goal in argument mining is to understand the structure in argumentative text (Persing and Ng, 2016; Peldszus and Stede, 2015; Stab and Gurevych, 2016; Nguyen and Litman, 2016). One fundamental assumption when working with argumentative text is the presence of Arguments Components (ACs). The types of ACs are generally characterized as a *claim* or a *premise* (Govier, 2013), with premises acting as support (or possibly attack) units for claims (though some corpora have further AC types, such as *major claim* (Stab and Gurevych, 2016, 2014b)).

The task of processing argument structure encapsulates four distinct subtasks (our work focuses on subtasks 2 and 3): (1) Given a sequence of tokens that represents an entire argumentative text, determine the token subsequences that constitute non-intersecting ACs; (2) Given an AC, determine the type of AC (*claim*, *premise*, etc.); (3) Given a set/list of ACs, determine which ACs have directed links that encapsulate overall argument structure; (4) Given two linked ACs, determine whether the link is a supporting or attacking relation. This can be labeled as a 'micro' approach to argument mining (Stab and Gurevych, 2016). In contrast, there have been a number of efforts to identify argument structure at a higher level (Boltuzic and Šnajder, 2014; Ghosh et al., 2014; Cabrio and Villata, 2012), as well as slightly re-ordering the pipeline with respect to AC types (Rinott et al., 2015)).

There are two key assumptions our work makes going forward. First, we assume subtask 1 has been completed, i.e. ACs have already been identified. Second, we follow previous work that assumes a tree structure for the linking of ACs (Palau and Moens, 2009; Cohen, 1987; Peldszus and Stede, 2015; Stab and Gurevych, 2016). Specifically, a given AC can only have a single outgoing link, but can have numerous incoming links. Furthermore, there is a 'head' component that has no outgoing link (the top of the tree). Depending on the corpus (see Section 4), an argument structure can be either a single tree or a forest, consisting of multiple trees. Figure 1 shows an example that we will use throughout the paper to concretely explain how our approach works. First, the left side of the figure presents the raw text of a paragraph in a persuasive essay (Stab and Gurevych, 2016), with the ACs contained in square brackets. Squiggly vs straight underlining differentiates between claims and premises, respectively. The ACs have been annotated as to how they are linked, and the right side of the figure reflects this structure. The argument

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1364–1373
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

First, [cloning will be beneficial for many people who are in need of organ transplants]$_{AC1}$. In addition, [it shortens the healing process]$_{AC2}$. Usually, [it is very rare to find an appropriate organ donor]$_{AC3}$ and [by using cloning in order to raise required organs the waiting time can be shortened tremendously]$_{AC4}$.

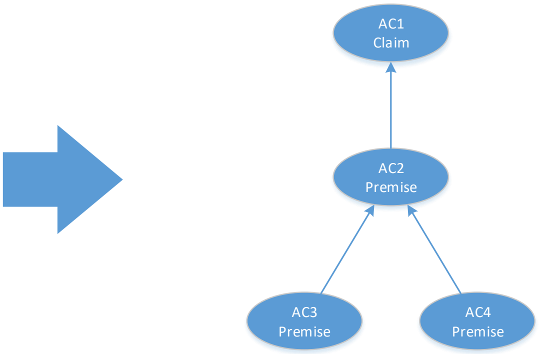


Figure 1: An example of argument structure with four ACs. The left side shows raw text that has been annotated for the presence of ACs. Squiggly or straight underlining means an AC is a *claim* or *premise*, respectively. The ACs in the text have also been annotated for links to other ACs, which is shown in the right figure. ACs 3 and 4 are *premises* that link to another *premise*, AC2. Finally, AC2 links to a *claim*, AC1. AC1 therefore acts as the central argumentative component.

structure with four ACs forms a tree, where AC2 has two incoming links, and AC1 acts as the head, with no outgoing links. We also specify the *type* of AC, with the head AC marked as a *claim* and the remaining ACs marked as *premises*. Lastly, we note that the order of argument components can be a strong indicator of how components should relate. Linking to the first argument component can provide a competitive baseline heuristic (Peldszus and Stede, 2015; Stab and Gurevych, 2016).

Given the above considerations, we propose that sequence-to-sequence attention modeling, in the spirit of a Pointer Network (PN) (Vinyals et al., 2015b), can be effective for predicting argument structure. To the best of our knowledge, a clean, elegant implementation of a PN-based model has yet to be introduced for discourse parsing tasks. A PN is a sequence-to-sequence model (Sutskever et al., 2014) that outputs a distribution over the encoding indices at each decoding timestep. More generally, it is a recurrent model with attention (Bahdanau et al., 2014), and we claim that as such, it is promising for link extraction because it inherently possesses three important characteristics: (1) it is able to model the sequential nature of ACs, (2) it constrains ACs to have a single outgoing link, thus partly enforcing the tree structure, and (3) the hidden representations learned by the model can be used for jointly predicting multiple subtasks. Furthermore, we believe the sequence-to-sequence aspect of the model provides two distinct benefits: (1) it allows for two separate representations of a single AC (one for the source and one for the target of the link), and (2) the decoder network-could learn to predict correct sequences of linked indices, which is a second recurrence over ACs. Note that we also test the sequence-to-sequence architecture against a simplified model that only uses hidden states from an encoding network to make predictions (see Section 5).

The main technical contribution of our work is a joint model that simultaneously predicts links between ACs and determines their *type*. Our joint model uses the hidden representation of ACs produced during the encoding step (see Section 3.4). While PNs were originally proposed to allow a variable length decoding sequence, our model differs in that it decodes for the same number of timesteps as there are inputs. This is a key insight that allows for a sequence-to-sequence model to be used for structural prediction. Aside from the partial assumption of tree structure in the argumentative text, our models do not make any additional assumptions about the AC types or connectivity, unlike the work of Peldszus (2014). Lastly, in respect to the broad task of parsing, our model is flexible because it can easily handle non-projective, multi-root dependencies. We evaluate our models on the corpora of Stab and Gurevych (2016) and Peldszus (2014), and compare our results with the results of the aformentioned authors. Our results show that (1) joint modeling is imperative for competitive performance on the link extraction task, (2) the presence of the second recurrence improves performance over a non-sequence-to-sequence model, and (3) the joint model can outperform models with heavy feature-engineering and corpus-specific constraints.

2 Related Work

Palau and Moens (2009) is an early work in argument mining, using a hand-crafted Context-Free Grammar to determine the structure of ACs in a corpus of legal texts. Lawrence et al. (2014) leverage a topic modeling-based AC similarity to uncover tree-structured arguments in philosophical texts. Recent work offers data-driven approaches to the task of predicting links between ACs. Stab and Gurevych (2014b) approach the task as a binary classification problem. The authors train an SVM with various semantic and structural features. Peldszus and Stede have also used classification models for predicting the presence of links (2015). The first neural network-based model for argumentation mining was proposed by Laha and Raykar (2016), who use two recurrent networks in end-to-end fashion to classify AC types.

Various authors have also proposed to jointly model link extraction with other subtasks from the argument mining pipeline, using either an Integer Linear Programming (ILP) framework (Persing and Ng, 2016; Stab and Gurevych, 2016) or directly feeding previous subtask predictions into a tree-based parser. The former joint approaches are evaluated on an annotated corpus of persuasive essays (Stab and Gurevych, 2014a, 2016), and the latter on a corpus of microtexts (Peldszus, 2014). The ILP framework is effective in enforcing a tree structure between ACs when predictions are made from otherwise naive base classifiers.

Recurrent neural networks have previously been proposed to model tree/graph structures in a linear manner. Vinyals et al. (2015c) use a sequence-to-sequence model for the task of syntactic parsing. Bowman et al. (2015) experiment on an artificial entailment dataset that is specifically engineered to capture recursive logic (Bowman et al., 2014). Standard recurrent neural networks can take in complete sentence sequences and perform competitively with a recursive neural network. Multi-task learning for sequence-to-sequence has also been proposed (Luong et al., 2015), though none of the models used a PN for prediction.

In the field of discourse parsing, the work of Li et al. (2016) is the only work, to our knowledge, that incorporates attention into the network architecture. However, the attention is only used in the process of creating representations of the text itself. Attention is *not* used to predict the overall discourse structure. In fact, the model still relies on a binary classifier to determine if textual components should have a link. Arguably the most similar approach to ours is in the field of dependency parsing (Cheng et al., 2016). The authors propose a model that performs 'queries' between word representations in order to determine a distribution over potential headwords.

3 Proposed Approach

In this section, we describe our approach to using a sequence-to-sequence model with attention for argument mining, specifically, identifying AC types and extracting the links between them. We begin by giving a brief overview of these models.

3.1 Pointer Network

A PN is a sequence-to-sequence model (Sutskever et al., 2014) with attention (Bahdanau et al., 2014) that was proposed to handle decoding sequences over the encoding inputs, and can be extended to arbitrary sets (Vinyals et al., 2015a). The original motivation for a pointer network was to allow networks to learn solutions to algorithmic problems, such as the traveling salesperson and convex hull problems, where the solution is a sequence over input points. The PN model is trained on input/output sequence pairs (E, D), where E is the source and D is the target (our choice of E,D is meant to represent the encoding, decoding steps of the sequence-to-sequence model). Given model parameters Θ, we apply the chain rule to determine the probability of a single training example:

$$p(D|E;\Theta) = \prod_{i=1}^{m(E)} p(D_i|D_1, ..., D_{i-1}, E; \Theta) \tag{1}$$

where the function m signifies that the number of decoding timesteps is a function of each individual training example. We will discuss shortly why we need to modify the original definition of m for our application. By taking the log-likelihood of Equation 1, we arrive at the optimization objective:

$$\Theta^* = \arg\max_{\Theta} \sum_{E,D} \log p(D|E;\Theta) \tag{2}$$

which is the sum over all training example pairs.

The PN uses Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) for sequential modeling, which produces a hidden layer h at each encoding/decoding timestep. In practice, the PN has two separate LSTMs, one for

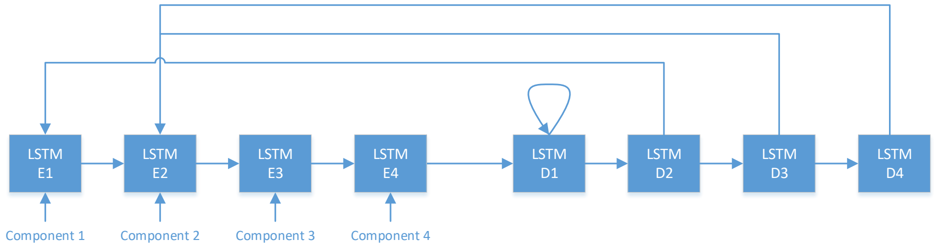


Figure 2: Applying a Pointer Network to the example paragraph in Figure 1 with LSTMs unrolled over time. Note that D1 points to itself to denote that it has not outgoing link and is therefore the head of a tree.

encoding and one for decoding. Thus, we refer to encoding hidden layers as e, and decoding hidden layers as d.

The PN uses a form of content-based attention (Bahdanau et al., 2014) to allow the model to produce a distribution over input elements. This can also be thought of as a distribution over input indices, wherein a decoding step 'points' to the input. Formally, given encoding hidden states $(e_1, ..., e_n)$, the model calculates $p(D_i|D_1, ..., D_{i-1}, E)$ as follows:

$$u^i_j = v^T \tanh(W_1 e_j + W_2 d_i) \quad (3)$$

$$p(D_i|D_1, ..., D_{j-1}, E) = softmax(u^i) \quad (4)$$

where matrices W_1, W_2 and vector v are parameters of the model (along with the LSTM parameters used for encoding and decoding). In Equation 3, prior to taking the dot product with v, the resulting transformation can be thought of as creating a joint hidden representation of inputs i and j. Vector u^i in equation 4 is of length n, and index j corresponds to input element j. Therefore, by taking the softmax of u^i, we are able to create a distribution over the input.

3.2 Link Extraction as Sequence Modeling

A given piece of text has a set of ACs, which occur in a specific order in the text: $(C_1, ..., C_n)$. Therefore, at encoding timestep i, the model is fed a representation of C_i. Since the representation is large and sparse (see Section 3.3 for details on how we represent ACs), we add a fully-connected layer before the LSTM input. Given a representation R_i for AC C_i, the LSTM input A_i is calculated as:

$$A_i = \sigma(W_{rep} R_i + b_{rep}) \quad (5)$$

where W_{rep}, b_{rep} in turn become model parameters, and σ is the sigmoid function[1]. Similarly, the decoding network applies a fully-connected layer with sigmoid activation to its inputs, see Figure 3. At encoding step i, the encoding LSTM produces hidden layer e_i, which can be thought of as a hidden representation of AC C_i.

In order to make sequence-to-sequence modeling applicable to the problem of link extraction, we explicitly set the number of decoding timesteps to be equal to the number of input components. Using notation from Equation 1, the decoding sequence length for an encoding sequence E is simply $m(E) = |\{C_1, ..., C_n\}|$, which is trivially equal to n. By constructing the decoding sequence in this manner, we can associate decoding timestep i with AC C_i.

From Equation 4, decoding timestep i will output a distribution over input indices. The result of this distribution will indicate to which AC component C_i links. Recall there is a possibility that an AC has no outgoing link, such as if it's the root of the tree. In this case, we state that if AC C_i does not have an outgoing link, decoding step D_i will output index i. Conversely, if D_i outputs index j, such that j is not equal to i, this implies that C_i has an outgoing link to C_j. For the argument structure in Figure 1, the corresponding decoding sequence is $(1, 1, 2, 2)$. The topology of this decoding sequence is illustrated in Figure 2. Observe how C_1 points to itself since it has no outgoing link.

Finally, we note that we have a Bidirectional LSTM (Graves and Schmidhuber, 2005) as the encoder, unlike the model proposed by Vinyals et al. (2015b). Thus, e_i is the concatenation of forward and backward hidden states $\overrightarrow{e}_i$ and $\overleftarrow{e}_{n-i+1}$, produced by two separate LSTMs. The decoder remains a standard forward LSTM.

3.3 Representing Argument Components

At each timestep of the encoder, the network takes in a representation of an AC. Each AC is itself

[1] We also experimented with relu and elu activations, but found sigmoid to yield the best performance.

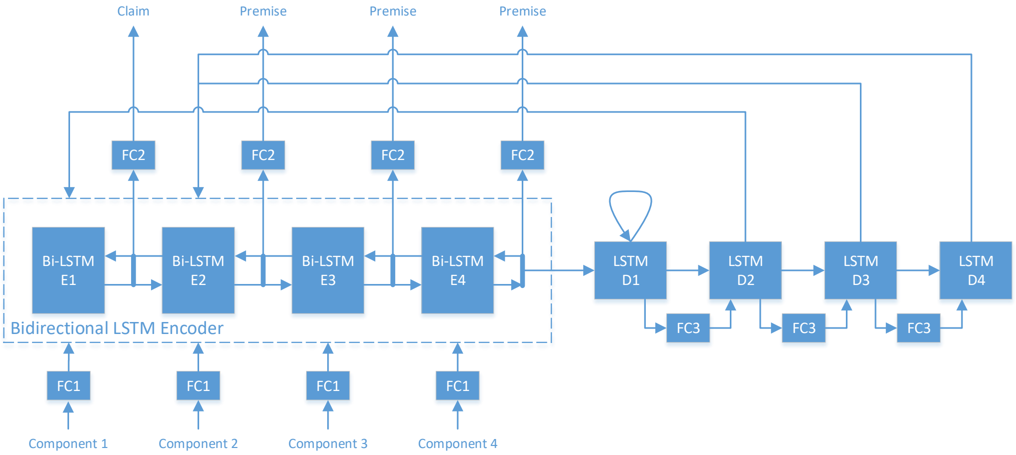


Figure 3: Architecture of the joint model applied to the example in Figure 1. Note that D1 points to itself to denote that it has not outgoing link and is therefore the head of a tree.

a sequence of tokens, similar to the Question-Answering dataset from Weston et al. (2015). We follow the work of Stab and Gurevych (2016) and focus on three different types of features to represent our ACs: (1) Bag-of-Words of the AC; (2) Embedding representation based on GloVe embeddings (Pennington et al., 2014), which uses average, max, and min pooling across the token embeddings; (3) Structural features: Whether or not the AC is the first AC in a paragraph, and whether the AC is in an opening, body, or closing paragraph. See Section 6 for an ablation study of the proposed features.

3.4 Joint Neural Model

Up to this point, we focused on the task of extracting links between ACs. However, recent work has shown that joint models that simultaneously try to complete multiple aspects of the subtask pipeline outperform models that focus on a single subtask (Persing and Ng, 2016; Stab and Gurevych, 2014b; Peldszus and Stede, 2015). Therefore, we will modify the single-task architecture so that it would allow us to perform AC classification (Kwon et al., 2007; Rooney et al., 2012) together with link prediction. Knowledge of an individual subtask's predictions can aid in other subtasks. For example, *claims* do not have an outgoing link, so knowing the type of AC can aid in the link prediction task. This can be seen as a way of regularizing the hidden representations from the encoding component (Che et al., 2015).

At each timestep, predicting AC type is a straightforward classification task: given AC C_i, we need to predict whether it is a *claim*, *premise*, or possibly *major claim*. More generally, this is another sequence modeling problem: given input sequence E, we want to predict a sequence of argument types T. For encoding timestep i, the model creates hidden representation e_i. This can be thought of as a representation of AC C_i. Therefore, our joint model will simply pass this representation through a fully-connected layer as follows:

$$z_i = W_{cls}e_i + b_{cls} \tag{6}$$

where W_{cls}, b_{cls} become elements of the model parameters, Θ. The dimensionality of W_{cls}, b_{cls} is determined by the number of classes. Lastly, we use softmax to form a distribution over the possible classes.

Consequently, the probability of predicting the component type at timestep i is defined as:

$$p(T_i|E_i;\Theta) = softmax(z_i) \tag{7}$$

Finally, combining this new prediction task with Equation 2, we arrive at the new training objective:

$$\begin{aligned}\Theta^* = \arg\max_{\Theta} \alpha \sum_{E,D} \log p(D|E;\Theta) \\ +(1-\alpha)\sum_{E} \log p(T|E;\Theta)\end{aligned} \tag{8}$$

which simply sums the costs of the individual prediction tasks, and the second summation is the cost for the new task of predicting AC type. $\alpha \in [0,1]$ is a hyperparameter that specifies how we weight the two prediction tasks in our cost function. The architecture of the joint model, applied to our ongoing example, is illustrated in Figure 3.

4 Experimental Design

As we have mentioned, our work assumes that ACs have already been identified. The order of ACs corresponds directly to the order in which the ACs appear in the text. We test the effectiveness of our proposed model on a dataset of persuasive essays (PEC) (Stab and Gurevych, 2016), as well as a dataset of microtexts (MTC) (Peldszus, 2014). The feature space for the PEC has roughly 3,000 dimensions, and the MTC feature space has between 2,500 and 3,000 dimensions, depending on the data split. The PEC contains a total of 402 essays, with a frozen set of 80 essays held out for testing. There are three AC types in this corpus: *major claim*, *claim*, and *premise*. In this corpus, individual structures can be either trees or forests. Also, in this corpus, each essay has multiple paragraphs, and argument structure is only uncovered within a given paragraph. The MTC contains 112 short texts. Unlike the PEC, each text in this corpus is itself a complete example, as well as a single tree. Since the dataset is small, the authors have created 10 sets of 5-fold cross-validation, reporting the the average across all splits for final model evaluation. This corpus contains only two types of ACs: *claim* and *premise*. Note that link prediction is directed, i.e., predicting a link between the pair $C_i, C_j (i \neq j)$ is different than C_j, C_i.

We implement our models in TensorFlow (Abadi et al., 2015). We use the following parameters: hidden input dimension size 512, hidden layer size 256 for the bidirectional LSTMs, hidden layer size 512 for the LSTM decoder, α equal to 0.5, and dropout (Srivastava et al., 2014) of 0.9. We believe the need for such high dropout is due to the small amounts of training data (Zarrella and Marsh, 2016), particularly in the MTC. All models are trained with Adam optimizer (Kingma and Ba, 2014) with a batch size of 16. For a given training set, we randomly select 10% to become the validation set. Training occurs for 4,000 epochs. Once training is completed, we select the model with the highest validation accuracy (on the link prediction task) and evaluate it on the held-out test set. At test time, we take a greedy approach and select the index of the probability distribution (whether link or type prediction) with the highest value.

5 Results

The results of our experiments are presented in Tables 1 and 2. For each corpus, we present f1 scores for the AC type classification experiment, with a macro-averaged score of the individual class f1 scores. We also present the f1 scores for predicting the presence/absence of links between ACs, as well as the associated macro-average between these two values.

We implement and compare four types of neural models: 1) The previously described joint model from Section 3.4 (called Joint Model in the tables); 2) The same as 1), but without the fully-connected input layers (called Joint Model No FC Input in the table); 3) The same as 1), but the model only predicts the link task, and is therefore not optimized for type prediction (called Single-Task Model in the table); 4) A non-sequence-to-sequence model that uses the hidden layers produced by the BLSTM encoder with the same type of attention as the joint model (called Joint Model No Seq2Seq in the table). That is, d_i in Equation 3 is replaced by e_i.

In both corpora we compare against the following previously proposed models: Base Classifier (Stab and Gurevych, 2016) is a feature-rich, task-specific (AC type or link extraction) SVM classifier. Neither of these classifiers enforce structural or global constraints. Conversely, the ILP Joint Model (Stab and Gurevych, 2016) provides constraints by sharing prediction information between the base classifiers. For example, the model attempts to enforce a tree structure among ACs within a given paragraph, as well as using incoming link predictions to better predict the type class *claim*. For the MTC only, we also have the following comparative models: Simple (Peldszus and Stede, 2015) is a feature-rich logistic regression classifier. Best EG (Peldszus and Stede, 2015) creates an Evidence Graph (EG) from the predictions of a set of base classifiers. The EG models the potential argument structure, and offers a global optimization objective that the base classifiers attempt to optimize by adjusting their individual weights. Lastly, MP+p (Peldszus and Stede, 2015) combines predictions from base classifiers with a Minimum Spanning Tree Parser (MSTParser).

6 Discussion

First, we point out that the joint model achieves state-of-the-art on 10 of the 13 metrics in Tables 1 and 2, including the highest results in all metrics on the PEC, as well as link prediction on the MTC. The performance on the MTC is very en-

	Type prediction				Link prediction		
Model	Macro f1	MC f1	Cl f1	Pr f1	Macro f1	Link f1	No Link f1
Base Classifier	.794	.891	.611	.879	.717	.508	.917
ILP Joint Model	.826	.891	.682	.903	.751	.585	.918
Single-Task Model	-	-	-	-	.709	.511	.906
Joint Model No Seq2Seq	.810	.830	.688	.912	.754	.589	.919
Joint Model No FC Input	.791	.826	.642	.906	.708	.514	.901
Joint Model	**.849**	**.894**	**.732**	**.921**	**.767**	**.608**	**.925**

Table 1: Results on the Persuasive Essay corpus. All models we tested are joint models, except for the Single-Task Model model, which only predicts links. All model have a fully-connected input layer, except for the row titled 'Joint Model No FC Input'. See Section 5 for a full description of the models.

	Type prediction			Link prediction		
Model	Macro f1	Cl f1	Pr f1	Macro f1	Link f1	No Link f1
Simple	.817	-	-	.663	.478	.848
Best EG	**.869**	-	-	.693	.502	.884
MP+p	.831	-	-	.720	.546	.894
Base Classifier	.830	.712	.937	.650	.446	.841
ILP Joint Model	.857	.770	.943	.683	.486	.881
Joint Model	.813	.692	.934	**.740**	**.577**	**.903**

Table 2: Results on the Microtext corpus.

couraging for several reasons. First, the fact that the model can perform so well with only a hundred training examples is rather remarkable. Second, although we motivate the use of an attention model due to the fact that it partially enforces a tree structure, other models we compare against explicitly contain further constraints (for example, only premises can have outgoing links). Moreover, the MP+p model directly enforces the single tree constraint unique to the microtext corpus (the PEC allows forests). Even though the joint model does not have the tree constraint directly encoded, it able to learn the structure effectively from the training examples so that it can outperform the Mp+p model for link prediction. As for the other neural models, the joint model with no seq2seq performs competitively with the ILP joint model on the PEC, but trails the performance of the joint model. We believe this is because the joint model is able to create two different representations for each AC, one each in the encoding/decoding state, which benefits performance in the two tasks. We also believe that the joint model benefits from a second recurrence over the ACs, modeling the tree/forest structure in a linear manner. Conversely, the joint model with no seq2seq must encode information relating to type as well as link prediction in a single hidden representation. On one hand, the joint model no seq2seq outperforms the ILP model on link prediction, yet it is not able to match the ILP joint model's performance on type prediction, primarily due to the poor performance on predicting the *major claim* class. Another interesting outcome is the importance of the fully-connected layer before the LSTM input. This extra layer seems to be crucial for improving performance on this task. The results dictate that even a simple fully-connected layer with sigmoid activation can provide a useful dimensionality reduction step. Finally, and arguably most importantly, the single-task model, only optimized for link prediction, suffers a large drop in performance, conveying that the dual optimization of the joint model is vital for high performance in the link prediction task. We believe this is because the joint optimization creates more expressive representations of the ACs, which capture the natural relation between AC type and AC linking.

Table 3 shows the results of an ablation study for AC feature representation. Regarding link prediction, BOW features are clearly the most important, as their absence results in the highest drop in performance. Conversely, the presence of structural features provides the smallest boost in performance, as the model is still able to record state-

	Type prediction				Link prediction		
Model	Macro f1	MC f1	Cl f1	Pr f1	Macro f1	Link f1	No Link f1
No structural	.808	.824	.694	.907	.760	.598	.922
No BOW	.796	.833	.652	.902	.728	.543	.912
No Embeddings	.827	.874	.695	.911	.750	.581	.918
Only Avg Emb*	.832	.873	.717	.917	.751	.583	.918
Only Max Emb*	.843	.874	**.732**	**.923**	.766	**.608**	.924
Only Min Emb*	.838	.878	.719	.918	.763	.602	.924
All features	**.849**	**.894**	**.732**	.921	**.767**	**.608**	**.925**

Table 3: Feature ablation study. * indicates that both BOW and Structural are present, as well as the stated embedding type.

	Type prediction				Link prediction		
Bin	Macro f1	MC f1	Cl f1	Pr f1	Macro f1	Link f1	No Link f1
$1 \leq len < 4$	.863	.902	.798	.889	.918	.866	.969
$4 \leq len < 8$	.680	.444	.675	.920	.749	.586	.912
$8 \leq len < 12$	.862*	.000*	.762	.961	.742	.542	.941

Table 4: Results of binning test data by length of AC sequence. * indicates that this bin does not contain any *major claim* labels, and this average only applies to *claim* and *premise* classes. However, we do not disable the model from predicting this class: the model was able to avoid predicting this class on its own.

of-the-art results compared to the ILP Joint Model. This shows that the Joint Model is able to capture structural cues through sequence modeling and semantics. When considering type prediction, both BOW and structural features are important, and it is the embedding features that provide the least benefit. The ablation results also provide an interesting insight into the effectiveness of different pooling strategies for using individual token embeddings to create a multi-word embedding. The popular method of averaging embeddings (which is used by Stab and Gurevych (2016) in their system) is in fact the worst method, although its performance is still competitive with the previous state-of-the-art. Conversely, max pooling results are on par with the joint model results in Table 1.

Table 4 shows results on the PEC test set with the test examples binned by sequence length. First, it is not surprising to see that the model performs best when the sequences are the shortest (for link prediction; type prediction actually sees the worst performance in the middle bin). As the sequence length increases, the accuracy on link prediction drops. This is possibly due to the fact that as the length increases, a given AC has more possibilities as to which other AC it can link to, making the task more difficult. Conversely, there is actually a rise in no link prediction accuracy from the second to third row. This is likely due to the fact that since the model predicts at most one outgoing link, it indirectly predicts no link for the remaining ACs in the sequence. Since the chance probability is low for having a link between a given AC in a long sequence, the no link performance is actually better in longer sequences. The results of the length-based binning could also potentially give insight into the poor performance on the type prediction task in the MTC. Since the arguments in the MTC average 5 ACs, they would be in the second bin (row 2) of Table 4. The *claim* and *premise* f1 scores for this bin are similar to those from the same system's performance on the MTC.

7 Conclusion

In this paper we have proposed how to use a joint sequence-to-sequence model with attention (Vinyals et al., 2015b) to both extract links between ACs and classify AC type. We evaluate our models on two corpora: a corpus of persuasive essays (Stab and Gurevych, 2016), and a corpus of microtexts (Peldszus, 2014). The Joint Model records state-of-the-art results on the persuasive essay corpus, as well as achieving state-of-the-art results for link prediction on the microtext corpus. The results show that jointly modeling the two prediction tasks is critical for high performance. Future work can attempt to learn the AC representations themselves, such as in Kumar et al. (2015).

Lastly, future work can integrate subtasks 1 and 4 into the model. The representations produced by Equation 3 could potentially be used to predict link type, i.e. supporting or attacking (the fourth subtask in the pipeline). In addition, a segmenting technique, such as the one proposed by Weston et al. (2014), can accomplish subtask 1.

Acknowledgments

This work was supported in part by the U.S. Army Research Office under Grant No. W911NF-16-1-0174.

References

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

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

Filip Boltuzic and Jan Šnajder. 2014. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58. Citeseer.

Samuel R Bowman, Christopher D Manning, and Christopher Potts. 2015. Tree-structured composition in neural networks without tree-structured architectures. *arXiv preprint arXiv:1506.04834*.

Samuel R Bowman, Christopher Potts, and Christopher D Manning. 2014. Recursive neural networks can learn logical semantics. *arXiv preprint arXiv:1406.1827*.

Elena Cabrio and Serena Villata. 2012. Natural language arguments: A combined approach. In *ECAI*, volume 242, pages 205–210.

Zhengping Che, David Kale, Wenzhe Li, Mohammad Taha Bahadori, and Yan Liu. 2015. Deep computational phenotyping. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 507–516. ACM.

Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional attention with agreement for dependency parsing. *arXiv preprint arXiv:1608.02076*.

Robin Cohen. 1987. Analyzing the structure of argumentative discourse. *Computational linguistics*, 13(1-2):11–24.

Debanjan Ghosh, Smaranda Muresan, Nina Wacholder, Mark Aakhus, and Matthew Mitsui. 2014. Analyzing argumentative discourse units in online interactions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 39–48.

Trudy Govier. 2013. *A practical study of argument*. Cengage Learning.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

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

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Namhee Kwon, Liang Zhou, Eduard Hovy, and Stuart W Shulman. 2007. Identifying and classifying subjective claims. In *Proceedings of the 8th annual international conference on Digital government research: bridging disciplines & domains*, pages 76–81. Digital Government Society of North America.

Anirban Laha and Vikas Raykar. 2016. An empirical evaluation of various deep learning architectures for bi-sequence classification tasks. In *COLING*.

John Lawrence, Chris Reed, Colin Allen, Simon McAlister, Andrew Ravenscroft, and David Bourget. 2014. Mining arguments from 19th century philosophical texts using topic based modelling. In *Proceedings of the First Workshop on Argumentation Mining*, pages 79–87. Citeseer.

Qi Li, Tianshi Li, and Baobao Chang. 2016. Discourse parsing with attention-based hierarchical neural networks. In *EMNLP*.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Huy V Nguyen and Diane J Litman. 2016. Context-aware argumentative relation mining.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th international conference on artificial intelligence and law*, pages 98–107. ACM.

Andreas Peldszus. 2014. Towards segment-based recognition of argumentation structure in short texts. *ACL 2014*, page 88.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 938–948.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

Isaac Persing and Vincent Ng. 2016. End-to-end argumentation mining in student essays. In *Proceedings of NAACL-HLT*, pages 1384–1394.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence-an automatic method for context dependent evidence detection. In *EMNLP*, pages 440–450.

Niall Rooney, Hui Wang, and Fiona Browne. 2012. Applying kernel methods to argumentation mining. In *FLAIRS Conference*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Christian Stab and Iryna Gurevych. 2014a. Annotating argument components and relations in persuasive essays. In *COLING*, pages 1501–1510.

Christian Stab and Iryna Gurevych. 2014b. Identifying argumentative discourse structures in persuasive essays. In *EMNLP*, pages 46–56.

Christian Stab and Iryna Gurevych. 2016. Parsing argumentation structures in persuasive essays. *arXiv preprint arXiv:1604.07370*.

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

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2015a. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015b. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015c. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Guido Zarrella and Amy Marsh. 2016. Mitre at semeval-2016 task 6: Transfer learning for stance detection. *arXiv preprint arXiv:1606.03784*.

Identifying *attack* and *support* argumentative relations using deep learning

Oana Cocarascu and **Francesca Toni**
Department of Computing
Imperial College London

Abstract

We propose a deep learning architecture to capture argumentative relations of *attack* and *support* from one piece of text to another, of the kind that naturally occur in a debate. The architecture uses two (unidirectional or bidirectional) Long Short-Term Memory networks and (trained or non-trained) word embeddings, and allows to considerably improve upon existing techniques that use syntactic features and supervised classifiers for the same form of (relation-based) argument mining.

1 Introduction

Argument Mining (AM) is a relatively new research area which involves, amongst others, the automatic detection in text of arguments, argument components, and relations between arguments (see (Lippi and Torroni, 2016) for an overview). We focus on a specific type of AM, referred to as *Relation-based AM* (Carstens and Toni, 2015), which has recently received attention by several researchers (e.g. see (Bosc et al., 2016; Carstens and Toni, 2017)). This type of AM aims at identifying argumentative relations of *attack* and *support* between natural language arguments in text, by classifying pairs of pieces of text as belonging to *attack*, *support* or *neither attack nor support* relations. For example, consider the three texts taken from Carstens and Toni (2015):

t_1: *'We should grant politicians immunity from prosecution'*

t_2: *'Giving politicians immunity allows them to focus on performing their duties'*

t_3: *'The ability to prosecute politicians is the ultimate protection against abuse of power'*

Here t_2 *supports* t_1, t_3 *attacks* t_1, and t_2 and t_3 *neither attack nor support* one another.

Relation-based AM is useful, for example, to pave the way towards identifying accepted opinions (Bosc et al., 2016) or divisive issues (Konat et al., 2016) within debates.

We propose a deep learning architecture for Relation-based AM based on Long-Short Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997). Within the architecture, each input text is fed, as a (trained or non-trained) 100-dimensional GloVe embedding (Pennington et al., 2014), into a (unidirectional or bidirectional) LSTM which produces a vector representation of the text independently of the other text being analysed. The two vectors are then merged (using element-wise sum or concatenation) and the resulting vector is fed to a softmax classifier which predicts whether the pair of input texts belongs to the *attack*, *support* or *neither* relations. The input texts may be at most 50 words long, but are not restricted to single sentences.

We experimented with several instances of the architecture and achieved 89.53% accuracy and 89.07% F_1 using unidirectional LSTMs and concatenation as the merge layer, considerably outperforming feature-based supervised classifiers used in the studies which presented the corpus we also use (Carstens and Toni, 2015, 2017).

The remainder of the paper is organised as follows. In Section 2 we discuss related work and the corpus we use. In Section 3 we describe our deep learning architecture and report experiments and results in Section 4. We conclude the paper and propose directions for future work in Section 5.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1374–1379
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

2 Background

2.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) (Elman, 1990; Mikolov et al., 2010) are a type of neural networks in which the hidden layer is connected to itself so that the previous hidden state is used along with the input at the current step. RNNs tend to suffer from the vanishing gradients problem (Bengio et al., 1994) while trying to capture long-term dependencies.

LSTM models (Hochreiter and Schmidhuber, 1997), a type of RNNs, address this problem by introducing memory cells and gates into networks. LSTMs use memory cells to store contextual information and three types of gates (input, forget, and output gates) that determine which information needs to be added or removed to learn long-term dependencies within a sequence.

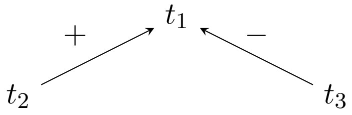
One problem with RNNs/LSTMs in natural language processing is that they do not make use of the information of future words. Bidirectional RNNs/LSTMs (BiRNNs/BiLSTMs) (Schuster and Paliwal, 1997) solve this problem by using both previous and future words while processing the input sequence with two RNNs: one in the forward and one in the backward direction, resulting in two vectors for each input.

2.2 Related work

Identifying relations between texts has recently received a great deal of attention, e.g. in Argument Mining (AM) (see (Lippi and Torroni, 2016) for a recent overview). In particular, *Relation-based AM* (Carstens and Toni, 2015) aims to automatically identify argumentative relations to create Bipolar Argumentation Frameworks (BAFs) (Cayrol and Lagasquie-Schiex, 2005).

BAFs are triples $\langle AR, attacks, supports\rangle$ consisting of a set of arguments AR and two binary relations $attacks$ and $supports$ between arguments.

The example texts introduced in Section 1 form a BAF with $AR = \{t_1, t_2, t_3\}$ and $attacks$, $supports$ given graphically (as -, + respectively) as follows:



Carstens and Toni (2017) obtained 61.8% accuracy and 62.1% F_1 on a news articles using Support Vector Machines (SVMs) and features such as distance measures, word overlap, sentence metrics and occurences of sentiment words.

Bosc et al. (2016) used a corpus consisting of tweets to determine *attack* and *support* relations between tweets. Using an encoder-decoder architecture and two LSTMs (the second LSTM initialised with the last hidden state of the first LSTM), they obtained negative results (0.2 F_1 for *support* and 0.16 F_1 for *attack*).

Other works in AM use deep learning models to determine relations between arguments, but of a different kind than in our work. Notably, Habernal and Gurevych (2016) experimented with LSTM models extended with an attention mechanism and a convolution layer over the input pairs to determine whether an input argument is more convincing than the other input argument. Thus, their focus is on determining a “more convincing than” relation, rather than *attack* and *support* argumentative relations, between arguments.

Several authors used neural network models for tasks related to the form of AM we consider. Yin et al. (2016) proposed three attention mechanisms for Convolutional Neural Networks to model pairs of sentences in tasks such as textual entailment and answer selection, whereas dos Santos et al. (2016) proposed a two-way attention mechanism to jointly learn the representation of two inputs in an answer selection setting.

Bowman et al. (2015) used stacked LSTMs to determine entailment, neutral and contradiction relations amongst sentence pairs using the SNLI (Stanford Natural Language Inference) corpus, with the bottom layer taking as input the concatenation of the input sentences. Recognising textual entailment between two sentences was also addressed in (Rocktäschel et al., 2015) which used LSTMs and a word-by-word neural attention mechanism on the SNLI corpus.

Liu et al. (2016) proposed two models capturing the interdependencies between the two parallel LSTMs encoding two input sentences for the tasks of recognising textual entailment and matching questions and answers. Further, Koreeda et al. (2016) used a BiRNN with a word-embedding-based attention model to determine whether a piece of an evidence supports a claim that a phrase promotes or suppresses a value, using a dataset of 1000 pairs.

2.3 Dataset

Determining relations between any texts can be seen as a three-class problem, with labels $L = \{attack, support, neither\}$. We used a dataset covering various topics such as movies, technology and politics[1], where *attack* relations represent 31% of the dataset, *support* relations represent 32% of the dataset and *neither* relations represent 37% of the dataset.

We have also explored the use of other corpora (e.g. the SNLI corpus (Bowman et al., 2015) and Araucaria in AIFdb[2]) that we ultimately decided not to include due to their structure not being directly amenable to our analysis.

3 Architecture

Figure 1 summarises the deep learning architecture that we use for predicting which relation from $L = \{attack, support, neither\}$ holds between the first and the second texts in any input pair.

We do not limit input texts to be single sentences, but limit them to 50 words (as this is the average text length in our corpus): inputs whose size is smaller than this threshold are padded with zeros at the end to give sequences of exactly 50 words. The input texts are (separately) embedded as 100-dimensional GloVe vectors (Pennington et al., 2014), with the words that do not appear in the vectors being treated as unknown. As we will see in Section 4, we experimented with the pre-trained word representations (freezing the weights during learning) as well as learning the weights.

The architecture relies upon two parallel LSTMs to model the two texts separately. We experimented with both unidirectional and bidirectional LSTMs (see Section 4). In both cases, we set the LSTM dimension to 32, as this proved to be the best, amongst alternatives (64, 100, 128), for mitigating overfitting. In addition, our LSTMs use a Rectified Linear Unit (ReLU) activation, each returning a vector of dimension 32.

Each LSTM network produces a vector representation of the input text, independently of the other text being analysed. The two vectors are then merged and the resulting vector fed to a softmax classifier which predicts the label for the relation between the first and the second input texts. As we will see in Section 4, we experimented with two types of merge layer: *sum*, which performs element-wise sum, and *concat*, which performs tensor concatenation.

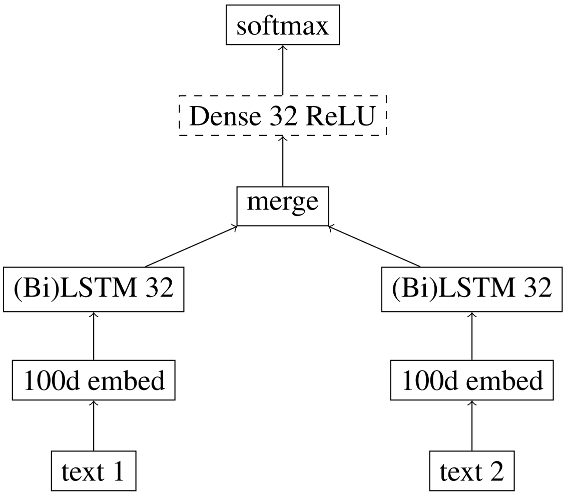


Figure 1: Our architecture: two (unidirectional or bidirectional) LSTMs are run with one text each. The dashed layer (Dense 32 ReLU) is optional.

Hyper-parameter	Value	Hyper-parameter	Value
Dropout	0.2	LSTM size	32
Embedding size	100	Dense size	32
Sequence length	50	Batch size	128

Table 1: Hyper-parameters for our (Bi)LSTMs.

After the merge layer, our architecture incorporates an optional dense feedforward layer. Our experiments (see Section 4) included testing whether the inclusion of this layer has an impact on the results. Again, we chose the dimension (32) as it proved better for mitigating overfitting than alternatives that we tried (64).

The values for the hyper-parameters used in our experiments (see Section 4) are summarised in Table 1. We used a mini-batch size of 128 and cross-entropy loss. To avoid overfitting, we applied dropout before the merge layer with probability 0.2, but not on the recurrent units. The hyper-parameters were optimised using the Adam method (Kingma and Ba, 2014) with learning rate 0.001, which turned out to give better performances than alternative optimisers we tried (Adagrad, Adadelta and RMSprop).

[1] https://www.doc.ic.ac.uk/~lc1310/

[2] https://corpora.aifdb.org

4 Results

We trained for 50 epochs or until the performance on the development set stopped improving, in order to avoid overfitting. The development set was 20% of the training dataset in the 10-fold cross-validation setup. In more detail, we run 10 stratified fold cross-validation for 5 times (so that each fold is a good representative of the whole). We report the average results of the 5x10 fold cross-validation in Table 2. As baseline, we used Logistic Regression (LR) and unigrams obtained from concatenating the two input texts.

We experimented with using BiLSTMs and unidirectional LSTMs with the two types of merge layers and using non-trained embeddings, namely pre-trained word representations (freezing the weights during learning), or trained embeddings, learning the weights during training.

We achieved 89.53% accuracy and 89.07% F_1 by concatenating the output of the two separate LSTMs. Unexpectedly, BiLSTMs performed worse than LSTMs (Table 2 only includes the best performing BiLSTM instance of the architecture, using concatenation and the feedforward layer). We believe this is because of the size of the dataset and that this effect could be diminished by acquiring more data. For the LSTM model with trained embeddings, the accuracy varied between 84.84% and 90.02%. Concatenating the LSTMs' output vectors yields better performance than performing element-wise sum of the vectors. We believe this is because this allows the system to encode more features, allowing the network to use more information.

Using the default, pre-trained word embeddings yields worse results compared to the baseline. We believe this is because the quality of word embeddings is dependent on the training corpora.[3] Training the word embeddings results in better performance compared to the baseline with improvements of up to 12% in accuracy and up to 11.5% in F_1.

In all cases, training the word embeddings results in dramatic improvements compared to freezing the embedding weights during learning, varying from 9.9% to 21.3% increase in accuracy and up to 25% in F_1. We also report the standard deviation of our models with trained embeddings. This shows that our best models (LSTMs with a concatenation layer) are stable and perform consistently on the task considered. Using One-Way ANOVA, the result is significant at $p < 0.05$ (the f-ratio value is 145.45159, the p-value is $<$ 0.00001).

5 Conclusion

We proposed a deep learning architecture based on Long Short-Term Memory (LSTM) networks to capture the argumentative relation of *attack* and *support* between any two texts. Our architecture uses two (unidirectional or bidirectional) LSTMs to analyse separately two 100-dimensional (non-trained or trained) GloVe vectors representing the two input texts. The outputs of the two LSTMs are then concatenated and fed to a softmax classifier to predict the relation between the input texts.

Our unidirectional LSTM model with trained embeddings and a concatenation layer achieved 89.53% accuracy and 89.07% F_1. The results indicate that LSTMs may be better suited for Relation-based Argument Mining at least for non-micro texts (Bosc et al., 2016) than standard classifiers as used in e.g. Carstens and Toni (2017), as LSTMs are better at capturing long-term dependencies between words and they operate over sequences, as found in text.

In future work, we plan to test our model on corpora such as the Language of Opposition from AIFdb[4] (by converting the finer-grained relation types used in this corpus to argumentative relations of the kind we considered), on datasets proposed for different tasks (e.g. identifying textual entailment could be seen as identifying support) and thus possibly use the corpus proposed by Bowman et al. (2015), as well as the twitter dataset of Bosc et al. (2016) once it becomes publicly available. Also, attack and support relations of the kind we have considered in this paper may be seen as special types of discourse relations (Teufel et al., 1999; Lin et al., 2009). It would be interesting to see whether any corpora for identifying discourse relations could be useful for furthering our experimentation. Finally, we plan to incorporate an attention-based mechanism as well as additional features (e.g. extracted through sentiment analysis) to determine which parts of the texts are most relevant in identifying attack and support.

[3] Pennington et al. (2014) computed the 100-dimensional GloVe embeddings on a a dump of English Wikipedia pages from 2014 consisting of 400k words.

[4] `https://corpora.aifdb.org`

Baseline	**A%**	**P%**	**R%**	**F$_1$%**						
LR (unigrams)	77.87	78.02	77.87	77.89						
Model/	**Non-trained embeddings**				**Trained embeddings**					
Merge/Dense	**A%**	**P%**	**R%**	**F$_1$%**	**A%**	**P%**	**R%**	**F$_1$%**	**A std**	**F$_1$ std**
BiLSTM/c/T	60.72	64.36	52.64	57.36	70.66	73.18	62.96	66.93	2.06	4.60
LSTM/c/F	68.25	72.39	59.07	64.38	**89.53**	90.80	87.67	**89.07**	**0.47**	**0.73**
LSTM/c/T	68.68	72.77	58.21	63.49	90.02	90.89	88.26	89.41	2.09	2.92
LSTM/s/T	64.21	69.18	51.07	57.09	84.84	86.75	79.98	82.35	5.02	9.26

Table 2: 5x10 fold cross-validation results, using *c(oncat)* or *s(um)* for merging the output of the two (Bi)LSTMs, with (non-)trained embeddings; T (*True*)/F (*False*) represent inclusion/omission, respectively, of the Dense 32 ReLU layer. *std* represents standard deviation of 5x10 fold cross-validation.

References

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Transactions on Neural Networks*, 5(2):157–166.

Tom Bosc, Elena Cabrio, and Serena Villata. 2016. Tweeties squabbling: Positive and negative results in applying argument mining on social media. In *Computational Models of Argument COMMA*, pages 21–32.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Lucas Carstens and Francesca Toni. 2015. Towards relation based argumentation mining. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 29–34.

Lucas Carstens and Francesca Toni. 2017. Using argumentation to improve classification in natural language problems. *ACM Transactions on Embedded Computing Systems*. Forthcoming.

Claudette Cayrol and Marie-Christine Lagasquie-Schiex. 2005. On the acceptability of arguments in bipolar argumentation frameworks. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty: 8th European Conference*, pages 378–389.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Ivan Habernal and Iryna Gurevych. 2016. What makes a convincing argument? Empirical analysis and detecting attributes of convincingness in web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1214–1223.

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

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Barbara Konat, John Lawrence, Joonsuk Park, Katarzyna Budzynska, and Chris Reed. 2016. A corpus of argument networks: Using graph properties to analyse divisive issues. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC)*.

Yuta Koreeda, Toshihiko Yanase, Kohsuke Yanai, Misa Sato, and Yoshiki Niwa. 2016. Neural attention model for classification of sentences that support promoting/suppressing relationship. In *Proceedings of the Third Workshop on Argument Mining*.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 343–351.

Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology*, 16(2):10.

Pengfei Liu, Xipeng Qiu, Yaqian Zhou, Jifan Chen, and Xuanjing Huang. 2016. Modelling interaction of sentence pair with coupled-lstms. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1703–1712.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664.

Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Mike Schuster and K. Kuldip Paliwal. 1997. Bidirectional recurrent neural networks. *Transactions on Signal Processing*, 45(11):2673–2681.

Simone Teufel, Jean Carletta, and Marc Moens. 1999. An annotation scheme for discourse-level argumentation in research articles. In *EACL 9th Conference of the European Chapter of the Association for Computational Linguistics*, pages 110–117.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: attention-based convolutional neural network for modeling sentence pairs. *TACL*, 4:259–272.

Neural Lattice-to-Sequence Models for Uncertain Inputs

Matthias Sperber[1], **Graham Neubig**[2], **Jan Niehues**[1], **Alex Waibel**[1]
[1]Karlsruhe Institute of Technology, Germany
[2]Carnegie Mellon University, USA
matthias.sperber@kit.edu

Abstract

The input to a neural sequence-to-sequence model is often determined by an up-stream system, e.g. a word segmenter, part of speech tagger, or speech recognizer. These up-stream models are potentially error-prone. Representing inputs through word lattices allows making this uncertainty explicit by capturing alternative sequences and their posterior probabilities in a compact form. In this work, we extend the TreeLSTM (Tai et al., 2015) into a LatticeLSTM that is able to consume word lattices, and can be used as encoder in an attentional encoder-decoder model. We integrate lattice posterior scores into this architecture by extending the TreeLSTM's child-sum and forget gates and introducing a bias term into the attention mechanism. We experiment with speech translation lattices and report consistent improvements over baselines that translate either the 1-best hypothesis or the lattice without posterior scores.

1 Introduction

In many natural language processing tasks, we will require a down-stream system to consume the input of an up-stream system, such as word segmenters, part of speech taggers, or automatic speech recognizers. Among these, one of the most prototypical and widely used examples is speech translation, where a down-stream translation system must consume the output of an up-stream automatic speech recognition (ASR) system.

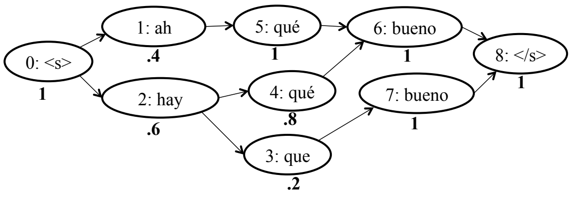


Figure 1: A lattice with 3 possible paths and posterior scores. Translating the whole lattice potentially allows for recovering from errors in its 1-best hypothesis.

Previous research on traditional phrase-based or tree-based statistical machine translation have used word lattices (e.g. Figure 1) as an effective tool to pass on uncertainties from a previous step (Ney, 1999; Casacuberta et al., 2004). Several works have shown quality improvements by translating lattices, compared to translating only the single best upstream output. Examples include translating lattice representations of ASR output (Saleem et al., 2004; Zhang et al., 2005; Matusov et al., 2008), multiple word segmentations, and morphological alternatives (Dyer et al., 2008).

Recently, neural sequence-to-sequence (seq2seq) models (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015) have often been preferred over the traditional methods for their strong empirical results and appealing end-to-end modeling. These models force us to rethink approaches to handling lattices, because their recurrent design no longer allows for efficient lattice decoding using dynamic programming as was used in the earlier works.

As a remedy, Su et al. (2017) proposed replacing the sequential encoder by a lattice encoder to obtain a lattice-to-sequence (lat2seq) model. This is achieved by extending the encoder's Gated Recurrent Units (GRUs) (Cho et al., 2014) to be conditioned on multiple predecessor paths. The authors demonstrate improvements in Chinese-to-English translation by translating lattices that combine the output of multiple word segmenters, rather than a single segmented sequence.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1380–1389
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

However, this model does not address one aspect of lattices that we argue is critical to obtaining good translation results: their ability to encode the certainty or uncertainty of the paths through the use of posterior scores. Specifically, we postulate that these scores are essential for tasks that require handling lattices with a considerable amount of erroneous content, such as those produced by ASR systems. In this paper, we propose a lattice-to-sequence model that accounts for this uncertainty. Specifically, our contributions are as follows:

- We employ the popular child-sum TreeLSTM (Tai et al., 2015) to derive a lattice encoder that replaces the sequential encoder in an attentional encoder-decoder model. We show empirically that this approach yields only minor improvements compared to a baseline fine-tuned on sequential ASR outputs. This finding stands in contrast to the positive results by Su et al. (2017), and by Ladhak et al. (2016) on a lattice classification task, and suggests higher learning complexity of our speech translation task.
- We hypothesize that lattice scores are crucial in aiding training and inference, and propose several techniques for integrating lattice scores into the model: (1) We compute *weighted child-sums*,[1] where hidden units in the lattice encoder are conditioned on their predecessor hidden units such that predecessors with low probability are less influential on the current hidden state. (2) We bias the TreeLSTM's *forget gates* for each incoming connection toward being more forgetful for predecessors with low probability, such that their cell states become relatively less influential. (3) We *bias the attention mechanism* to put more focus on source embeddings belonging to nodes with high lattice scores. We demonstrate empirically that the third proposed technique is particularly effective and produces strong gains over the baseline. According to our knowledge, this is the first attempt of integrating lattice scores already at the *training stage* of a machine translation model.
- We exploit the fact that our lattice encoder is a strict generalization of a sequential encoder by *pre-training* on sequential data, obtaining faster and better training convergence on large corpora of parallel sequential data.

We conduct experiments on the Fisher and Callhome Spanish–English Speech Translation Corpus (Post et al., 2013) and report improvements of 1.4 BLEU points on Fisher and 0.8 BLEU points on Callhome, compared to a strong baseline optimized for translating 1-best ASR outputs. We find that the proposed integration of lattice scores is crucial for achieving these improvements.

2 Background

Our work extends the seminal work on attentional encoder-decoder models (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015) which we survey in this section.

Given an input sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$, the goal is to generate an appropriate output sequence $\boldsymbol{y} = (y_1, y_2, \ldots, y_M)$. The conditional probability $p(\boldsymbol{y} \mid \boldsymbol{x})$ is estimated using parameters trained on a parallel corpus, e.g. of sentences in the source and target language in a translation task. This probability is factorized as the product of conditional probabilities of each token to be generated: $p(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{M} p(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$. The training objective is to estimate parameters θ that maximize the log-likelihood of the sentence pairs in a given parallel training set D: $J(\theta) = \sum_{(\boldsymbol{x},\boldsymbol{y})\in D} \log p(\boldsymbol{y} \mid \boldsymbol{x}; \theta)$.

2.1 Encoder

In our baseline model, the encoder is a bi-directional recurrent neural network (RNN), following (Bahdanau et al., 2015). Here, the source sentence is processed in both the forward and backward directions with two separate RNNs. For every input x_i, two hidden states are generated as

$$\overrightarrow{\mathbf{h}}_i = \mathrm{LSTM}\left(E_{\mathit{fwd}}(x_i), \overrightarrow{\mathbf{h}}_{i-1}\right) \qquad (1)$$

$$\overleftarrow{\mathbf{h}}_i = \mathrm{LSTM}\left(E_{\mathit{bwd}}(x_i), \overleftarrow{\mathbf{h}}_{i+1}\right), \qquad (2)$$

where E_{fwd} and E_{bwd} are source embedding lookup tables. We opt for long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) recurrent units because of their high performance and in order to later take advantage of the TreeLSTM extension (Tai et al., 2015). We stack multiple LSTM layers and concatenate the final layer into the final source hidden state $\mathbf{h}_i = \overrightarrow{\mathbf{h}}_i \mid \overleftarrow{\mathbf{h}}_i$, where layer indices are omitted for simplicity.

2.2 Attention

We use an attention mechanism (Luong et al., 2015) to summarize the encoder outputs into a

[1] This is reminiscent of the weighted pooling strategy by Ladhak et al. (2016) for spoken utterance classification.

fixed-size representation. At each decoding time step j, a context vector $\mathbf{c}_j$ is computed as a weighted average of the source hidden states: $\mathbf{c}_j = \sum_{i=1}^{N} \alpha_{ij}\mathbf{h}_i$. The normalized attentional weights α_{ij} measure the relative importance of the source words for the current decoding step and are computed as a softmax with normalization factor Z summing over i:

$$\alpha_{ij} = \frac{1}{Z}\exp\left(\mathrm{s}\left(\mathbf{s}_{j-1}, \mathbf{h}_i\right)\right) \tag{3}$$

$\mathrm{s}(\cdot)$ is a feed-forward neural network with a single layer that estimates the importance of source hidden state $\mathbf{h}_i$ for producing the next target symbol y_j, conditioned on the previous decoder state $\mathbf{s}_{j-1}$.

2.3 Decoder

The decoder creates output symbols one by one, conditioned on the encoder states via the attention mechanism. It contains another LSTM, initialized using the final encoder hidden state: $\mathbf{s}_0 = \mathbf{h}_N$. The decoding at step j assumes a special start-of-sequence symbol y_0 and is computed as $\mathbf{s}_j = \mathrm{LSTM}\left(E_{trg}(y_{j-1}), \mathbf{s}_{j-1}\right)$, and then $\tilde{\mathbf{s}}_t = \tanh(W_{hs}[\mathbf{s}_j; \mathbf{c}_j] + \mathbf{b}_{hs})$ The conditional probability that the j-th target word is generated is: $p(y_j \mid \boldsymbol{y}_{<j}, \boldsymbol{x}) = \mathrm{softmax}(W_{so}\tilde{\mathbf{s}}_t + \mathbf{b}_{so})$. Here, E_{trg} is the target embedding lookup table, W_{hs} and W_{so} are weight matrices, and $\mathbf{b}_{hs}$ and $\mathbf{b}_{so}$ are bias vectors.

During decoding beam search is used to find an output sequence with high conditional probability.

3 Attentional Lattice-to-Sequence Model

The seq2seq model described above assumes sequential inputs and is therefore limited to taking a single output of an up-stream model as input. Instead, we wish to consume lattices to carry over uncertainties from an up-stream model.

3.1 Lattices

Lattices (e.g. Figure 1) represent multiple ambiguous or competing sequences in a compact form. They are a more efficient alternative to enumerating all hypotheses as an n-best list, as they allow for avoiding redundant computation over subsequences shared between multiple hypotheses. Lattices can either be produced directly, e.g. by an ASR dumping its pruned search space (Post et al., 2013), or can be obtained by merging several n-best sequences (Dyer et al., 2008; Su et al., 2017).

A word lattice $\mathcal{G} = \langle V, E\rangle$ is a directed, connected, and acyclic graph with nodes V and edges E. $V{\subset}\mathbb{N}$ is a node set, and $(k,i){\in}E$ denotes an edge connecting node k to node i. $C(i)$ denotes the set of predecessor nodes for node i. We assume that all nodes follow a topological ordering, such that $k{<}i \; \forall \; k{\in}C(i)$. Each node i is assigned a word label $w(i)$. [2] Every lattice contains exactly one start-of-sequence node with only outgoing edges, and exactly one end-of-sequence node with only incoming edges.

3.2 Lattices and the TreeLSTM

One thing to notice here is that lattice nodes can have multiple predecessor states. In contrast, hidden states in LSTMs and other sequential RNNs are conditioned on only one predecessor state ($\tilde{h}_j$ in left column of Table 1), rendering standard RNNs unsuitable for the modeling of lattices. Luckily Tai et al. (2015)'s TreeLSTM, which was designed to compose encodings in trees, is also straightforward to apply to lattices; the TreeLSTM composes multiple child states into a parent state, which can also be applied to lattices to compose multiple predecessor states into a successor state. Table 1, middle column, shows the TreeLSTM in its child-sum variant that supports an arbitrary number of predecessors. Conditioning on multiple predecessor hidden states is achieved by simply taking their sum as $\tilde{\mathbf{h}}_i$. Cell states from multiple predecessor are each passed through their own forget gates $\mathbf{f}_{jk}$ and then summed.

Encoding a lattice results in one hidden state for each lattice node. Our lat2seq framework uses this network as encoder, computing the attention over all lattice nodes.[3] In other words we replace (1) by the following:

$$\overrightarrow{\mathbf{h}}_i = \mathrm{LatticeLSTM}\left(\mathbf{x}_i, \{\overrightarrow{\mathbf{h}}_k \mid k{\in}C(i)\}\right) \tag{4}$$

Similarly, we encode the lattice in backward direction and replace (2) accordingly. Figure 2 illustrates the result. The computational complexity of the encoder is $\mathcal{O}(|V| + |E|)$, i.e. linear in the number of nodes plus number of edges in the graph. The complexity of the attention mechanism is $\mathcal{O}(|V|M)$, where M is the output sequence

[2] It is perhaps more common to think of each edge representing a word, but we will motivate why we instead assign word labels to nodes in §3.3.

[3] This is similar in spirit to Eriguchi et al. (2016) who used the TreeLSTM in an attentional tree-to-sequence model.

	Sequential LSTM	TreeLSTM	Proposed LatticeLSTM
recurrence	$\tilde{\mathbf{h}}_i = \mathbf{h}_{i-1}$	$\tilde{\mathbf{h}}_i = \sum_{k \in C(i)} \mathbf{h}_k$	$\tilde{\mathbf{h}}_i = \sum_{k \in C(i)} \frac{w_{b/f,k}^{\mathbf{S}_h}}{\mathbf{Z}_{h,k}} \mathbf{h}_k$ (5)
forget gt.	$\mathbf{f}_i = \sigma\left(W_f \mathbf{x}_i + U_f \tilde{\mathbf{h}}_i + \mathbf{b}_f\right)$	$\mathbf{f}_{ik} = \sigma(W_f \mathbf{x}_i +$ $U_f \mathbf{h}_k + \mathbf{b}_f)$	$\mathbf{f}_{ik} = \sigma(W_f \mathbf{x}_i + U_f \mathbf{h}_k +$ $\left[\ln w_{b/f,k} \mathbf{S}_f - \mathbf{Z}_{f,k}\right] + \mathbf{b}_f)$ (6)
input gt.	$\mathbf{i}_i = \sigma\left(W_{in} \mathbf{x}_i + U_{in} \tilde{\mathbf{h}}_i + \mathbf{b}_{in}\right)$	as sequential	as sequential
output gt.	$\mathbf{o}_i = \sigma\left(W_o \mathbf{x}_i + U_o \tilde{\mathbf{h}}_i + \mathbf{b}_o\right)$	as sequential	as sequential
update	$\mathbf{u}_i = \tanh\left(W_u \mathbf{x}_i + U_u \tilde{\mathbf{h}}_i + \mathbf{b}_u\right)$	as sequential	as sequential
cell	$\mathbf{c}_i = \mathbf{i}_i \odot \mathbf{u}_i + \mathbf{f}_i \odot \mathbf{c}_{i-1}$	$\mathbf{c}_i = \mathbf{i}_i \odot \mathbf{u}_i +$ $\sum_{k \in C(i)} \mathbf{f}_{ik} \odot \mathbf{c}_k$	as TreeLSTM
hidden	$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i)$	as sequential	as sequential
attention	$\alpha_{ij} \propto \exp\left(s\left(\cdot\right)\right)$		$\alpha_{ij} \propto \exp\left[s\left(\cdot\right) + S_a \ln w_{m,i}\right]$ (7)

Table 1: Formulas for sequential and TreeLSTM encoders according to Tai et al. (2015), the proposed LatticeLSTM encoder, and conventional vs. proposed integration into the attention mechanism (bottom row). Inputs $\mathbf{x}_j$ are word embeddings or hidden states of a lower layer. $W_\cdot$ and $U_\cdot$ denote parameter matrices, $\mathbf{b}_\cdot$ bias terms, other terms are described in the text.

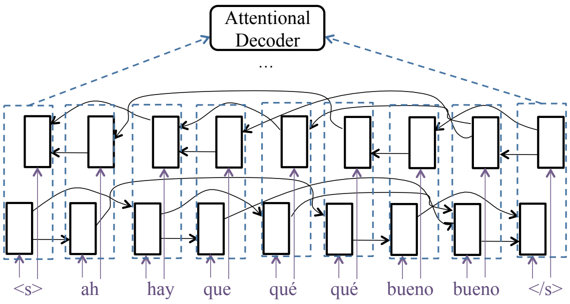


Figure 2: Network structure of a bidirectional lattice encoder with one layer.

length. $|V|$ depends on both the expected input sentence length and the lattice density.

3.3 Node-labeled Lattices

At this point we take a step back to motivate our choice of assigning word labels to lattice nodes, which is in contrast to the prior work by Ladhak et al. (2016) and Su et al. (2017) who assign word labels to edges. Recurrent states in edge-labeled lattice encoders are conditioned not only on multiple predecessor states, but must also aggregate words from multiple incoming edges. This implies that hidden units may represent more than one word in the lattice. Moreover, in the edge-labeled case hidden units that are in the same position in forward and backward encoders represent different words, but are nevertheless concatenated and attended to jointly. For these reasons we find our approach of encoding word-labeled lattices more intuitively appealing when used as input to an attentional decoder, although empirical justification is beyond the scope of this paper. We also note that it is easy to convert an edge-labeled lattice into a node-labeled lattice using the line-graph algorithm (Hemminger and Beineke, 1978), which we utilize in this work.

4 Integration of Lattice Scores

This section describes the key technical contribution of our work: integration of lattice scores encoding input uncertainty into the lat2seq framework. These lattice scores assign different probabilities to competing paths, and are often provided by up-stream statistical models. For example, an ASR may attach posterior probabilities that capture acoustic evidence and linguistic plausibility of words in the lattice. In this section, we describe our method, first explaining how we normalize scores to a format that is easily usable in our method, then presenting our methods for incorporating these scores into our encoder calculations.

4.1 Lattice Score Normalization

Lattice scores that are obtained from upstream systems (such as ASR) are typically given in forward-normalized fashion, interpreted as the probability

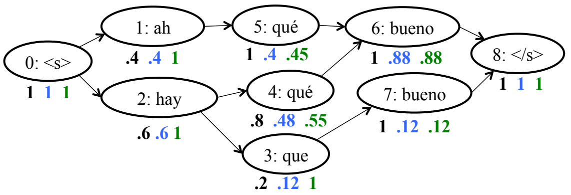


Figure 3: Lattice with forward-normalized, marginal, and backward-normalized scores.

of a node given its predecessor. Here, outgoing edges sum up to one, as illustrated in Figure 1. However, in some of our methods it will be necessary that scores be normalized in the backward direction, so that the weights from incoming connections sum up to one, or globally normalized, so that the probability of the node is the marginal probability of all the paths containing that node.

Let $w_{f,i}$, $w_{m,i}$, $w_{b,i}$ denote forward-normalized, marginal, and backward-normalized scores for node i respectively, illustrated in Figure 3. Given $w_{f,i}$, we can compute marginal probabilities recursively as $w_{m,i} = \sum_{k \in C(i)} w_{m,k} \cdot w_{f,i}$ by using the forward algorithm (Rabiner, 1989). Then, we can normalize backward using $w_{b,i} = \frac{w_{m,i}}{\sum_{k \in C'(i)} w_{m,k}}$, where $C'(i)$ denotes the successors of node i. All 3 forms are employed in the sections below.

Furthermore, when integrating these scores into the lat2seq framework, it is desirable to maintain flexibility over how strongly they should impact the model. For this purpose, we introduce a peakiness coefficient S. Given a lattice score $w_{b,i}$ in backward direction, we compute $w_{b,i}^{S}/Z_i$. $Z_i = \sum_{k \in C(i)} w_{b,k}$ is a re-normalization term to ensure that incoming connections still sum up to one. In the forward direction, we compute $w_{f,i}^{S}/Z_i$ and normalize analogously over outgoing connections. Setting $S=0$ amounts to ignoring the scores by flattening their distribution, while letting $S \to \infty$ puts emphasis solely on the strongest nodes. S can be optimized jointly with the other model parameters via back-propagation during model training.

4.2 Integration Approaches

We suggest three methods to integrate these scores into our lat2seq model, with equations shown in the right column of Table 1. These methods can optionally be combined, and we conduct an ablation study to assess the effectivity of each method in isolation (§5.3).

The first method consists of computing a weighted child-sum (WCS), using lattice scores as weights when composing the hidden state $\tilde{\mathbf{h}}_i$. This is based on the intuition that predecessor hidden states with high lattice weights should have a higher influence on their successor than states with low weights. The precise formulas for WCS are shown in (5).

The second method biases the forget gate $\mathbf{f}_{ik}$ for each predecessor cell state such that predecessors with high lattice score are more likely to pass through the forget gate (BFG). The intuition for this is similar to WCS; the composed cell state is more highly influenced by cell states from predecessors with high lattice score. BFG is implemented by introducing a bias term inside the sigmoid as in (6).

In the cases of both WCS and BFG, all hidden units have their own independent peakiness. Thus $\mathbf{S}_h$ and $\mathbf{S}_f$ are vectors, applied element-wise after broadcasting the lattice score. The renormalization terms $\mathbf{Z}_{h,k}$ and $\mathbf{Z}_{f,k}$ are also vectors and are applied element-wise. We use backward-normalized scores $w_{b,i}$ for the forward-directed encoder, and forward-normalized scores $w_{f,i}$ for the backward-directed encoder.

In the third and final method, we bias the attentional weights (BATT) to put more focus on lattice nodes with high lattice scores. This can potentially mitigate the problem of having multiple contradicting lattice nodes that may confuse the attentional decoder. BATT is computed by introducing a bias term to the attention as in (7). Attentional weights are scalars, so here the peakiness S_a is also a scalar. We drop the normalization term, relying instead on the softmax normalization. Both BFG and BATT use the logarithm of lattice scores so that values will still be in the probability domain after the softmax or sigmoid is computed.

4.3 Pre-training

Finally, note that our lattice encoder is a strict generalization of a sequential encoder. To reduce the computational burden, we exploit this fact and perform a two-step training process where the model is first *pre-trained* on sequential data, then *fine-tuned* on lattice data.[4] The pre-training, like standard training for neural machine translation (NMT), allows for efficient training using mini-batches, and also allows for training on standard text corpora for which we might not have lat-

[4] For the sequential data, we set all confidence scores to 1.

tices available. The fine-tuning is then performed on parallel data with lattices on the source side. This is much slower[5] than the pre-training because the network structure changes from sentence to sentence, preventing us from using efficient minibatched calculations. However, fine-tuning for only a small number of iterations is generally sufficient, as the model is already relatively accurate in the first place. In practice we found it important to use minibatches when fine-tuning, accumulating gradients over several examples before performing parameter updates. This provided negligible speedups but greatly improved optimization stability.

At test time, the model is able to translate both sequential and lattice inputs and can therefore be used even in cases where no lattices are available, at potentially diminished accuracy.

5 Experiments

5.1 Setting

We conduct experiments on the Fisher and Callhome Spanish–English Speech Translation Corpus (Post et al., 2013), a corpus of Spanish telephone conversations that includes automatic transcripts and lattices. The Fisher portion consists of telephone conversations between strangers, while the Callhome portion contains telephone conversations between relatives or friends. The training data size is 138,819 sentences (Fisher/Train), and 15,000 sentences (Callhome/Train). Held-out testing data is shown in Table 2. ASR word error rates (WER) are relatively high, due to the spontaneous speaking style and challenging acoustics. Lattices contain on average 3.4 (Fisher/Train) or 4.5 (Callhome/Train) times more words than the corresponding reference transcripts.

For preprocessing, we tokenized and lowercased source and target sides. We removed punctuation from the reference transcripts on the source side for consistency with the automatic transcripts and lattices. All models are pre-trained and fine-tuned on Fisher/Train unless otherwise noted. Our source-side vocabulary contains all words from the automatic transcripts for Fisher/Train, replacing singletons by an unknown word token, totaling 14,648 words. Similarly, on the target side we used all words from the reference translations of Fisher/Train, replacing singletons by the unknown word, yielding 10,800 words in total.

	1-best WER	oracle WER	# sent.
Fisher/Dev	41.3	19.3	3,979
Fisher/Dev2	40.0	19.4	3,961
Fisher/Test	36.5	16.1	3,641
Callhome/Devtest	64.7	36.4	3,966
Callhome/Evltest	65.3	37.9	1,829

Table 2: Development data statistics. Average sentence length is between 11.8 and 13.1.

Our implementation is based on lamtram (Neubig, 2015) and the DyNet (Neubig et al., 2017a) toolkit. We use the implemented attentional model with default parameters: a layer size of 256 per encoder direction and 512 for the decoder. Word embedding size was also set to 512. We used two encoder layers and two decoder layers for better baseline performance. For the sequential baselines, the LSTM variant in the left column of Table 1 was employed. We initialized the forget gate biases to 1 as recommended by Jozefowicz et al. (2015).

We used Adam (Kingma and Ba, 2014) for training, with an empirically determined initial learning rate of 0.001 for pre-training and 0.0001 for fine-tuning. We halve the learning rate when the dev perplexity (on Fisher/Dev) gets worse. Pre-training and fine-tuning on 1-best sequences is performed until convergence, and training on lattices is performed for 2 epochs to keep experimental effort manageable. On Fisher/Train, this took 3-4 days on a fast CPU.[6] Minibatch size was 1000 target words for pre-training, and 20 sentences for lattice training. Unless otherwise noted, we employed all three proposed lattice score integration approaches, and optimized peakiness coefficients jointly during training. We repeat training 3 times with different random seeds for parameter initialization and data shuffling, and report averaged results. We set the decoding beam size to 5.

[5] Our implementation processed sequential inputs about 75 times faster than lattice inputs during training, and overall fine-tuning convergence was 15 times faster. Decoding was only 1.2 times slower when using lattice inputs. Note that recently proposed approaches for autobatching (Neubig et al., 2017b) may considerably speed up lattice training.

[6] For comparison, we tried training on lattices from scratch, which took 9 days (6 epochs) to converge at a dev perplexity that was 10% worse than with the pre-training plus fine-tuning strategy. We also confirmed BLEU scores to be much inferior without pretraining.

5.2 Main Results

We compare 4 systems: Performing pre-training on the sequential reference transcripts only (R), fine-tuning on 1-best transcripts (R+1), fine-tuning on lattices without scores (R+L), and fine-tuning on lattices including lattice scores (R+L+S). At test time, we try references, lattice oracles,[7] 1-best transcripts, and lattices as inputs to all 4 systems. The former 2 experiments give upper bounds on achievable translation accuracy, while the latter 2 correspond to a realistic setting. Table 3 shows the results on Fisher/Dev2 and Fisher/Test.

Before even considering lattices, we can see that 1-best fine-tuning boosted BLEU scores quite impressively (1-best/R vs. 1-best/R+1), with gains of 1.3 and 0.7 BLEU points. This stands in contrast to Post et al. (2013) who find the 1-best transcripts not to be helpful for training a hierarchical machine translation system. Possible explanations are learning from repeating error patterns, and improved robustness to erroneous inputs. On top of these gains, our proposed set-up (lattice/R+L+S) improve BLEU scores by another 1.4. Removing the lattice scores (lattice/R+L) diminishes the results and performs worse than the 1-best baseline (1-best/R+1), indicating that the proposed lattice score integration is crucial for good performance. This demonstrates a clear advantage of our proposed method over that of Su et al. (2017).

As can be seen in the table, models fine-tuned on lattices show reasonable performance for both lattice and sequential inputs (1-best/R+L, lattice/R+L, 1-best/R+L+S, lattice/R+L+S). This is not surprising, given that the lattice training data includes lattices of varying density, including lattices with very few paths or even only one path. On the other hand, without fine-tuning on lattices, using lattices as input performs poorly (lattice/R and lattice/R+1). A closer look revealed that translations were often too long, potentially because implicitly learned mechanisms for length control were not ready to handle lattice inputs.

Table 3 reports perplexities for Fisher/Dev2. Unlike the corresponding BLEU scores, the lattice encoder appears stronger than the 1-best baseline in terms of perplexity even without lattice scores (lattice/R+L vs. 1-best/R+1). To understand this better, we computed the average entropy of the decoder softmax, a measure of how much confusion there is in the decoder predictions, independent of whether it selects the correct answer or not. Over the first 100 sentences, this value was 2.24 for 1-best/R+1, 2.39 for lattice/R+L, and 2.15 for lattice/R+L+S. This indicates that the decoder is more confused for lattices without scores, while integrating lattice scores removes this problem. These numbers also suggest that it may be possible to obtain further gains using methods that stabilize the decoder.

5.3 Ablation Experiments

Next, we conduct an ablation study to assess the impact of the three proposed extensions for integrating lattice scores (§4.2). We train models with different peakiness coefficients S, either ignoring lattices scores by fixing S=0, using lattice scores as-is by fixing S=1, or optimizing S during training. Table 4 shows the results. Overall, joint training of S gives similar results as fixing S=1, but both clearly outperform fixing S=0. Removing confidences (setting S=0) in one place at a time reveals that the attention mechanism is clearly the most important point of integration, while gains from the integration into child-sum and forget gate are smaller and not always consistent.

We also analyzed what peakiness values were actually learned. We found that all 3 models that we trained for the averaging purposes converged to S_a=0.62. $\mathbf{S}_h$ and $\mathbf{S}_f$ had per-vector means between 0.92 and 1.0, at standard deviations between 0.02 and 0.04. We conclude that while the peakiness coefficients were not particularly helpful in our experiments, stable convergence behavior makes them safe to use, and they might be helpful on other data sets that may contain lattice scores of higher or lower reliability.

5.4 Callhome Experiments

In this experiment, we test a situation in which we have a reasonable amount of sequential data available for pre-training, but only a limited amount of lattice training data for the fine-tuning step. This may be a more realistic situation, because speech translation corpora are scarce. To investigate in this scenario, we again pre-train our models on Fisher/Train, but then fine-tune them on the 9 times smaller Callhome/Train portion of the corpus. We fine-tune for 10 epochs, all other settings are as before. We use Callhome/Evltest for testing. Table 5 shows the results. The trends are consistent to §5.2: The proposed model (lattice/R+L+S) outperforms the 1-best baseline (1-best/R+1) by

[7] The path through the lattice with the best WER.

test-time	Trained on				Trained on			
inputs	R	R+1	R+L	R+L+S	R	R+1	R+L	R+L+S
reference	53.9 *(7.1)*	53.8 *(6.5)*	53.7 *(6.8)*	54.0 *(6.7)*	52.2	51.8	52.2	52.7
oracle	44.9 *(13.4)*	45.6 *(9.5)*	45.2 *(10.6)*	45.2 *(10.6)*	44.4	44.6	44.6	44.8
1-best	35.8 *(24.7)*	37.1 *(13.7)*	36.2 *(16.4)*	36.2 *(16.3)*	35.9	36.6	36.2	36.4
lattice	25.9 *(23.4)*	25.8 *(15.7)*	36.9 *(13.0)*	**38.5** *(12.6)*	26.2	25.8	36.1	**38.0**
	Fisher/Dev2				Fisher/Test			

Table 3: BLEU scores (4 references) and perplexities *(in brackets)*. Models are pre-trained only (R), fine-tuned on either 1-best outputs (R+1), lattices without scores (R+L), or lattices with scores (R+L+S). Statistically significant improvement (paired bootstrap resampling, $p < 0.05$) over 1-best/R+1 is in bold.

BATT	WCS	BFG	Fisher	Fisher
S_a	$\mathbf{S}_h$	$\mathbf{S}_f$	/Dev2	/Test
0	**0**	**0**	36.9	36.1
1	**1**	**1**	**38.2**	**37.4**
*	*	*	**38.5**	**38.0**
0	**1**	**1**	37.2	36.2
1	**0**	**1**	**37.9**	**37.5**
1	**1**	**0**	**38.2**	**37.8**
0	*	*	37.0	36.3
*	**0**	*	**38.3**	**37.9**
*	*	**0**	**38.1**	**37.5**
1-best/R+1			37.2	36.6

Table 4: BLEU scores (4 references) for differently configured peakiness coefficients $S_a, \mathbf{S}_h, \mathbf{S}_f$. 0/1 means fixing to that value, * indicates optimization during training. Statistically significant improvement over 1-best/R+1 is in bold.

test-time	Trained on			
inputs	R	R+1	R+L	R+L+S
reference	24.7	24.3	24.8	24.4
oracle	15.8	16.8	16.3	15.9
1-best	11.8	13.3	12.4	12.0
lattice	9.3	7.1	**13.7**	**14.1**

Table 5: BLEU scores on Callhome/Evltest (1 reference). All models are pre-trained on Fisher/Train references (R), and potentially fine-tuned on Callhome/Train. The best result using 1-best or lattice inputs is in bold. Statistically significant improvement over 1-best/R+1 is in bold.

0.8 BLEU points, which in turn beats the pre-trained system (1-best/R) by 1.5 BLEU points. Including the lattice scores is clearly beneficial, although lattices without scores also improve over 1-best inputs in this experiment.

5.5 Impact of Lattice Quality

Next, we analyze the impact of using lattices and lattice scores as the ASR WER changes. We concatenate all test data from Table 2 and divide the result into bins according to the 1-best WER. We sample 1000 sentences from each bin, and compare BLEU scores between several models.

The results are shown in Figure 4. For very good WERs, lattices do not improve over 1-best inputs, which is unsurprising. In all other cases, lattices are helpful. Lattice scores are most beneficial for moderate WERs, and not beneficial for very high WERs. We speculate that for high WERs, the lattice scores tend to be less reliable than for lower WERs.

6 Conclusion

We investigated translating uncertain inputs from an error-prone up-stream component using a neu-

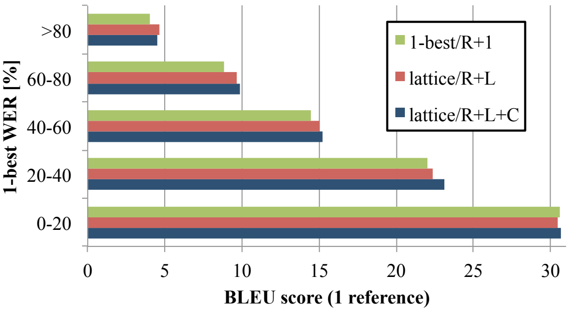


Figure 4: BLEU score over varying 1-best WERs.

ral lattice-to-sequence model. Our proposed model takes word lattices as input and is able to take advantage of lattice scores. In our experiments in a speech translation task we find consistent improvements over translating 1-best transcriptions and that consideration of lattice scores, especially in the attention mechanism, is crucial for obtaining these improvements.

Promising avenues for future work are investigating consensus networks (Mangu et al., 2000) for potential gains in terms of speed or quality as compared to lattice inputs, explicitly dealing with rare or unknown words in the lattice, and facilitating GPU training via autobatching (Neubig et al., 2017b).

Acknowledgments

We thank Paul Michel and Austin Matthews for their helpful comments on earlier drafts of this paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Representation Learning (ICLR)*, San Diego, USA.

F. Casacuberta, H. Ney, F. J. Och, E. Vidal, J. M. Vilar, S. Barrachina, I. Garcia-Varea, D. Llorens, C. Martinez, S. Molau, F. Nevado, M. Pastor, D. Picco, A. Sanchis, and C. Tillmann. 2004. Some approaches to statistical and finite-state speech-to-speech translation. *Computer Speech and Language*, 18(1):25–47.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078*.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing Word Lattice Translation. Technical Report LAMP-TR-149, University of Maryland, Institute For Advanced Computer Studies.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-Sequence Attentional Neural Machine Translation. In *Association for Computational Linguistic (ACL)*, pages 823–833, Berlin, Germany.

R.L. Hemminger and L.W. Beineke. 1978. Line graphs and line digraphs. In *Selected Topics in Graph Theory*, pages 271–305. Academic Press Inc.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An Empirical Exploration of Recurrent Network Architectures. In *International Conference on Machine Learning (ICML)*, Lille, France.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1700–1709, Seattle, Washington, USA.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Faisal Ladhak, Ankur Gandhe, Markus Dreyer, Lambert Mathias, Ariya Rastrow, and Björn Hoffmeister. 2016. LatticeRnn: Recurrent Neural Networks over Lattices. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 695–699, San Francisco, USA.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal.

Lidia Mangu, Eric Brill, and Andreas Stolcke. 2000. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech & Language*, 14(4):373–400.

Evgeny Matusov, Björn Hoffmeister, and Hermann Ney. 2008. ASR word lattice translation with exhaustive reordering is possible. In *Annual Conference of the International Speech Communication Association (InterSpeech)*, pages 2342–2345, Brisbane, Australia.

Graham Neubig. 2015. lamtram: A Toolkit for Language and Translation Modeling using Neural Networks. http://www.github.com/neubig/lamtram.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017a. DyNet: The Dynamic Neural Network Toolkit. *arXiv preprint arXiv:1701.03980*.

Graham Neubig, Yoav Goldberg, and Chris Dyer. 2017b. On-the-fly Operation Batching in Dynamic Computation Graphs. *arXiv preprint arXiv:1705.07860*.

Hermann Ney. 1999. Speech Translation: Coupling of Recognition and Translation. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 517–520, Phoenix, USA.

Matt Post, Gaurav Kumar, Adam Lopez, Damianos Karakos, Chris Callison-Burch, and Sanjeev Khudanpur. 2013. Improved Speech-to-Text Translation with the Fisher and Callhome Spanish–English Speech Translation Corpus. In *International Workshop on Spoken Language Translation (IWSLT)*, Heidelberg, Germany.

Lawrence R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*.

Shirin Saleem, Szu-Chen (Stan) Jou, Stephan Vogel, and Tanja Schultz. 2004. Using Word Lattice Information for a Tighter Coupling in Speech Translation Systems. In *International Conference of Spoken Language Processing (ICSLP)*, pages 41–44, Jeju Island, Korea.

Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. 2017. Lattice-Based Recurrent Neural Network Encoders for Neural Machine Translation. In *Conference on Artificial Intelligence (AAAI)*, pages 3302–3308.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112, Montréal, Canada.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Association for Computational Linguistic (ACL)*, pages 1556–1566, Beijing, China.

Ruiqiang Zhang, Genichiro Kikui, Hirofumi Yamamoto, and Wai-Kit Lo. 2005. A Decoding Algorithm for Word Lattice Translation in Speech Translation. In *International Workshop on Spoken Language Translation (IWSLT)*, pages 23–29, Pittsburgh, USA.

Memory-augmented Neural Machine Translation

Yang Feng[2,1,4], **Shiyue Zhang**[1,3], **Andi Zhang**[1,3], **Dong Wang**[1*] **and Andrew Abel**[5]
[1]Center for Speech and Language Technologies, RIIT, Tsinghua University
[2]Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences
[3]Beijing University of Posts and Telecommunications, China
[4]Huilan Corporation, Beijing, China
[5]Xi'An Jiaotong Liverpool-University, Suzhou, China
fengyang@ict.ac.cn, {byryuer, andizhang912}@gmail.com
wangdong99@mails.tsinghua.edu.cn, andrew.abel@xjtlu.edu.cn

Abstract

Neural machine translation (NMT) has achieved notable success in recent times, however it is also widely recognized that this approach has limitations with handling infrequent words and word pairs. This paper presents a novel memory-augmented NMT (M-NMT) architecture, which stores knowledge about how words (usually infrequently encountered ones) should be translated in a memory and then utilizes them to assist the neural model. We use this memory mechanism to combine the knowledge learned from a conventional statistical machine translation system and the rules learned by an NMT system, and also propose a solution for out-of-vocabulary (OOV) words based on this framework. Our experiments on two Chinese-English translation tasks demonstrated that the M-NMT architecture outperformed the NMT baseline by 9.0 and 2.7 BLEU points on the two tasks, respectively. Additionally, we found this architecture resulted in a much more effective OOV treatment compared to competitive methods.

1 Introduction

Neural Machine Translation (NMT) has been shown to have highly promising performance, particularly when a large amount of training data is available (Wu et al., 2016; Johnson et al., 2016; Mi et al., 2016). Although there are different model architectures (Sutskever et al., 2014; Bahdanau et al., 2015), the common principle behind the NMT approach is the same: encoding the meaning of the input into a concept space and performing translation based on this encoding. This 'meaning encoding' principle leads to a deeper understanding and learning of the translation rules, and hence a better translation than conventional statistic machine translation (SMT) that considers only surface forms, i.e., words and phrases (Koehn et al., 2003).

src.	人类共有二十三对染色体。
ref.	Humans have 23 pairs of chromosomes.
NMT	There are 23-year history of human history.

Table 1: An example of Chinese-to-English 'meaning drift' with NMT.

Despite positive results obtained so far, a particular problem of the NMT approach is that it has a tendency towards overfitting to frequent observations (words, word co-occurrences, translation pairs, etc.), but overlooking special cases that are not frequently observed. For example, NMT is good at learning translation pairs that are frequently observed, and can make use of them well at run-time, but for low-frequency pairs in the training data, the system may 'forget' to use them when they should be. Unfortunately, rare words are inevitable for all translation tasks due to Zipf's law, and indeed they are often the most important parts of a sentence, e.g., domain-specific entity names. Table 1 shows an example, where the word '染色体(chromosomes)' is an infrequent word. As the system does not know (or has effectively 'forgotten') this keyword, it does not translate correctly, and an irrelevant translation is produced, leading to the phenomenon of 'meaning drift'. This weakness with regard to infrequent words/pairs with NMT has been noticed by a number of researchers, and some studies have been conducted to address this problem, e.g., Luong et al. (2014); Cho et al. (2014); Li et al. (2016); Arthur et al. (2016); Bentivogli et al. (2016); Zhang et al. (2017).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1390–1399
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Superficially, this problem appears to be caused by the imperfect embeddings of infrequent words or the limited vocabulary size of NMT systems, but we argue that the deeper reason should be attributed to the nature of neural models: the translation function, represented by various neural networks, is shared amongst all of the translation pairs, so high-frequency and low-frequency pairs impact each other by adapting their shared parameters. Due to the overwhelming proportion of high-frequency pairs in the training data, the resulting trained model will naturally be much more focused on these frequently observed pairs. More seriously, because the translation function is smooth, infrequent pairs tend to be wrongly seen as noise in the training process and so are largely ignored by the model.

In contrast to this, the conventional SMT approach is based on statistics of words and/or phrases, which, in principle, is a symbolic method that uses a discrete model and involves little parameter sharing. The discrete model means that no matter how infrequently a pair occurs, its probability cannot be smoothed out, and the lack of shared parameters means that the frequent words or pairs have much less impact on infrequent words or pairs. Essentially, SMT memorizes as many of the observed patterns as possible, usually using a phrase table.

The respective advantages of SMT and NMT suggest that neither the pure neural approach nor the pure symbolic approach can provide a complete solution for machine translation, and a combined system that exploits the advantages of both approaches would be ideal. This idea has been adopted in early research into neural-based MT methods, where neural models were utilized to improve SMT performance (Zhang et al., 2015). However, this seems to be counterintuitive, as intuitively learning general rules should be the first step, rather than first memorizing special cases and then learning general rules. This suggests that the combined system should be primarily based on the neural architecture, with symbolic knowledge as a complementary support.

This paper presents such a neural-symbolic architecture, which involves a neural model component to deal with frequently seen patterns, and a memory component to provide knowledge for infrequently used words and pairs. More specifically, each memory element stores a source-target pair, specifying that a word defined by the source part should be translated to the word defined by the target part. This knowledge is then used to improve the neural model. This is analogy to an experienced translator, who can work well in most cases using their own knowledge (i.e. the neural model aspect), but for unfamiliar and uncommon words that they have little experience of, they will still need to refer to a dictionary (i.e., the memory). This proposed memory-augmented NMT, or M-NMT, is therefore arguably much more similar to human translators than either NMT or SMT.

2 Attention-based NMT

Before introducing our M-NMT architecture, we will give a brief review of our implementation of the attention-based RNN model first presented by Bahdanau et al. (2015). This model is regarded as the state-of-the-art model and will be used as the baseline system in this study. Additionally, the neural model component of the M-NMT architecture uses the same attention-based RNN model, as being presented in the following.

The attention-based RNN model is based on an encoder-decoder frame, where the input word sequence $[x_1, x_2, ...]$ in the source language is embedded as a sequence of hidden states $[h_1, h_2, ...]$ by a bi-directional RNN with GRU as the hidden units, and another RNN is used to produce the target sentence $[y_1, y_2, ...]$. To force the generation to focus on a particular segment of the input at each generation step, Bahdanau et al. (2015) proposed an attention mechanism. Specifically, when generating the i-th target word, the attention factor of the j-th source word (and its neighbors, precisely) is measured by the relevance between the current hidden state of the decoder, denoted by s_{t-1}, and the hidden state of the encoder at the j-th word h_j. This can be calculated by any similarity function, but a multiple layer perceptron (MLP) is often used, given by:

$$\alpha_{ij} = \frac{e_{ij}}{\sum e_{ik}}; \quad e_{ij} = a(s_{i-1}, h_j)$$

where $a(\cdot, \cdot)$ is the MLP-based relevance function, and α_{ij} is the attention factor of x_j at decoding step i. The semantic content that the decoder focuses on, i.e. attended content, is then derived by:

$$c_i = \sum \alpha_{ij} h_j.$$

The decoder updates the hidden state with a recurrent function f_d, formulated by:

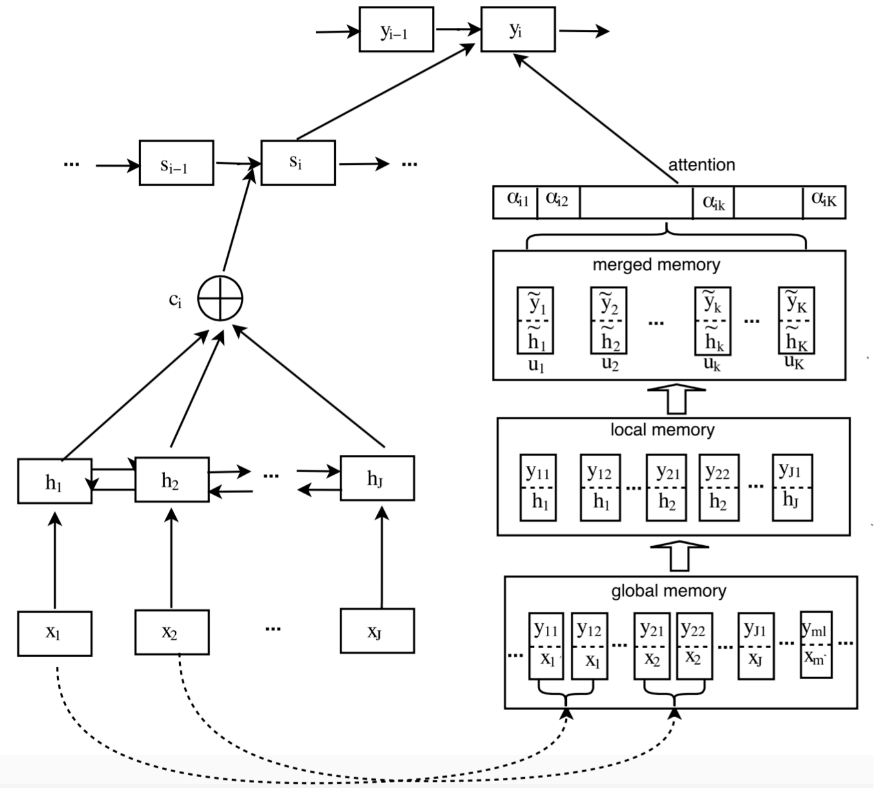


Figure 1: The structure of the M-NMT architecture.

$$s_i = f_d(y_{i-1}, s_{i-1}, c_i), \qquad (1)$$

and the next word y_i is generated according to the following posterior:

$$p(y_i) = \sigma(y_i^T W z_i) \qquad (2)$$

where $\sigma(\cdot)$ is the softmax function, W is a parameter matrix for word vector projection. The intermediate variable z_i is computed by a neural net with a single maxout hidden layer g, given by:

$$z_i = g(y_{i-1}, s_{i-1}, c_i).$$

We used Tensorflow to implement this model, and the training recipe largely followed the seminal paper of Bahdanau et al. (2015).

3 Memory-Augmented NMT

This section presents the M-NMT architecture. We first introduce the model, and then describe how the memory is constructed.

3.1 The Architecture

The M-NMT architecture is illustrated in Figure 1. It involves two components: the model and the memory components. The model component is a typical attention-based RNN model as presented in Section 2, which is regarded as being good at dealing with frequent words and pairs, and the memory component provides knowledge for infrequent words and pairs that are not easy for the neural model component to learn. The outputs of the two components are combined to produce a final consolidated translation.

3.2 Memory Elements

We define each item of memory as a mapping from a word in the source language to its translation in the target language. If there are multiple translations for a word, then several of the best will be added to the memory according to the probability of the translation, until the maximum number of target words is reached. A memory element can be formally written by:

$$u_{jl} = \begin{bmatrix} y_{jl} \\ x_j \end{bmatrix}$$

where y_{jl} is the l-th translation of word x_j. This mapping will be saved as a memory element and will be used during translation. We refer to this memory as the *global memory*, which is static during all the running time. The global memory is shown on the bottom-right of Figure 1.

To translate an input sentence, the memory elements the source words of which are in the input sentence are selected to form a *local memory*. This is shown in the right-middle of Figure 1. In order to include the context information in the local memory, the source part x_j is replaced by its annotation h_j:

$$u_{jl} = \begin{bmatrix} y_{jl} \\ h_j \end{bmatrix}$$

A consequence of the source encoding is that if a source word occurs multiple times in the sentence, all the occurrences should be put into the local memory, with different h_j to distinguish the context of each. Finally, the local memory is compressed as follows. For each distinct target word $\tilde{y}_k$ in the local memory, all the elements with $\tilde{y}_k$ as the target are merged into a single element u_k, for which the source part is the average of the source part of all the elements to be merged, given by:

$$u_k = \begin{bmatrix} \tilde{y}_k \\ \tilde{h}_k \end{bmatrix} = \begin{bmatrix} \tilde{y}_k \\ \sum_j p(x_j|\tilde{y}_k)h_j \end{bmatrix}; \ \forall \tilde{y}_k \in \{y_{jl}\} \quad (3)$$

where $p(x_j|\tilde{y}_k)$ means the probability that x_j is translated into $\tilde{y}_k$ and can be obtained from either a human-defined dictionary or the dictionary of an SMT system.

3.3 Memory Attention

In order to use the information stored in the memory to improve NMT, we need to pick up appropriate elements from the local memory at each translation step. A similar attention mechanism as in the neural model is designed. Denote the attention factor of each memory element u_k at each translation step i by α^m_{ik}, and assume it is derived from a relevance function e^m_{ik}:

$$\alpha^m_{ik} = \frac{e^m_{ik}}{\sum_{k=1}^{K} e^m_{ik}},$$

where K is the number of target words in the merged memory. The relevance function can be changed, but in this study, we use a simple design:

$$e^m_{ik} = (v^m)^\top tanh(W^m_s s_{i-1} + W^m_u u_k + W^m_y y_{i-1}) \quad (4)$$

where $tanh(\cdot)$ is the hyperbolic function, s_{i-1} is the current state of the decoder of the neural model, and y_{i-1} is the generated word in the previous step. The parameters of the memory attention mechanism include $\theta^m = \{v^m, W^m_s, W^m_u, W^m_y\}$, as defined in Eq. 4.

The attention factor α^m_{ik} can be used in different ways, here they are simply treated as the posterior to predict the next word to generate. Since the normalization is over all the target words in the local memory rather than the full vocabulary, treating α^m_{ik} as the posterior of all words is only an approximation, but was found in our experiments to be a good solution. This memory-based posterior is combined with the posterior of the neural model, resulting in a consolidated posterior, given by:

$$\tilde{p}(y_i) = \beta\alpha^m_{ik} + (1-\beta)p(y_i)$$

where $p(y_i)$ is the posterior produced by the neural model, as shown in Eq. 2, and β is a predefined interpolation factor. Here α^m_{ik} corresponds to the attention to the same word in the merged memory as the predicted word y_i. This simple posterior combination indicates the flexibility of the M-NMT architecture. Existing knowledge can be compiled into the local memory to improve model-based prediction, or if no knowledge is available, the system will rely on conventional NMT.

An advantage of this simple combination is that the memory component can be trained independently of the neural model. We set the objective of the training is to let the memory attention as accurate as possible. Given the n-th training sequence, at each translation step i, the target attention should be 1 on the current word y^n_i and 0 elsewhere. The objective function therefore can be written as the cross entropy between the target attention and the output of the attention function, given as follows:

$$L(\theta) = \sum_n \sum_i log(\alpha^m_{ik^n_i})$$

where k^n_i is the position of y^n_i in the merged memory. The optimization is conducted with respect to the parameters θ^m. The optimization algorithm is the stochastic gradient descent (SGD) with AdaDelta to adjust the learning rate (Zeiler, 2012).

It should be noted that joint training of the memory and the model is possible, but it requires a large amount of GPU memory and risks overfitting. Therefore, we only train the memory, with the model component unchanged. Efficient model-memory joint training is beyond the scope of this paper and can be investigated in future work. Particularly, the parameter β could be optimized to balance the contribution from the model part and the memory part, but constrains have to be carefully settled to avoid overfitting to the training data.

3.4 Memory for SMT Integration

The M-NMT architecture is a flexible framework that provides extra knowledge to the conventional model-based NMT. If the knowledge is generated by a conventional SMT system, it is essentially an elegant combination of SMT and NMT. In this work, we use the translation dictionary produced by an SMT system as the knowledge to create the memory, which involves first aligning the training sentence pairs using the GIZA++ toolkit (Och and Ney, 2003) in both directions, and applying the "intersection" refinement rules (Koehn et al., 2003) to get a single one-to-one alignment for each sentence pair, and then extracting the translation dictionary based on these alignments. We can see the dictionary as the phrase pairs of the length 1 and leave the phrase pairs longer than 1 as future work.

The key information provided by the dictionary is the conditional probability that a source and a target word are translated to each other. This information is used twice during local memory construction. Firstly, the conditional $p(y_{jl}|x_j)$ is used to select the most possible target words y_{jl} to participate the local memory, and secondly, the conditional $p(x_j|\tilde{y}_k)$ is used to merge the elements whose target words are $\tilde{y}_k$, as shown in Eq. 3.

3.5 Memory for OOV Treatment

The memory also provides a flexible way to address OOV words. OOV words can be defined in multiple ways, but here we focus on true OOVs that are totally new in both bilingual and monolingual data (i.e. rare words that are not present in any training data). One example is when a model is migrated to a specific domain. To address these OOVs, we firstly need a manually defined dictionary to specify how an OOV word should be translated, where the target word could be either an in-vocabulary word or an OOV.

This dictionary will be used as the knowledge to construct the local memory at run-time. Specifically, if an OOV word is encountered on either the source or target side during local memory construction, the vector of a similar word is borrowed to represent the OOV word. Since the words are totally new, the similar word has to be defined manually. To avoid any confusion with other words, the selected similar word should not appear in the existing input sequence if the OOV is in the source side, and should not match any target words in the existing local memory. To achieve this, several candidates have to be pre-defined for each OOV, so that alternative choices are available at run-time. A problem of this approach is that the vocabulary of the neural model is fixed, so cannot output probabilities for OOV words. To solve this, we let the selected similar word entirely overwritten by the OOV word, and any prediction for the similar word will be 're-directed' to the OOV word.

4 Related Work

The idea of memory augmentation was inspired by recent advances in the neural Turing machine (Graves et al., 2014, 2016) and memory network (Weston et al., 2014). These new models equip neural networks with an external memory that can be accessed and manipulated via some trainable operations. The memory idea has been utilized in NMT. For example, Wang et al. (2016) used a memory to extend the state of the decoder RNN in the attention-based NMT. In this case, the contribution of the memory is to provide temporary variables to assist RNN decoding. In contrast, our work uses memory to store knowledge. The memory in Wang et al.'s work could be considered to be note paper, while the memory in our work is more like a dictionary.

The idea of combining SMT and NMT was adopted by early NMT research, but these combinations were mostly based on the SMT framework, as discussed in depth in the review paper from Zhang et al. (2015). Cohn et al. (2016) proposed to enhance the attention-based NMT by using some structural knowledge from a word-based alignment model. The focus of their work was to use the extra knowledge to produce a better attention. In contrast, our work promotes the target words directly using the word mapping stored in the memory. Arthur et al. (2016) proposed to involve lexical knowledge to assist with translation, particularly for low-frequency words. This is similar to our proposed idea, with the key difference that their work uses the attention information to select the target words, while ours trains a separate attention, based on both the source and target words.

Regarding handling OOV words, Jean et al. (2015) presented an efficient training method to support a larger vocabulary, which helps alleviate the OOV problem significantly. Stahlberg et al. (2016) used SMT to produce candidate results in the form of lattice and NMT to re-score the results. As SMT uses a larger vocabulary than NMT, some OOV words can be retained. Sennrich et al. (2016) proposed a subword approach, where OOV words are expected to be spelled out by subword units. Luong et al. (2014) proposed a post-processing approach that learns the position of the source word when an UNK symbol is produced during decoding. By this position information, the UNK symbol (unknown words) can be replaced by the correct translation using a lexical table. Li et al. (2016) proposed a replace-and-restore approach that replaces infrequent words with similar words before the training and decoding, and restores rare words and their target words, obtained from a lexical table. Compared to the work of (Luong et al., 2014) and (Li et al., 2016), which relies on post-processing, our M-NMT approach is more like pre-processing. This means that the required information for OOV words is prepared before decoding. This seems more flexible than the post-processing methods, as we can easily deal with multiple targets for OOVs, by letting the decoder select which target is the most appropriate. Nevertheless, we do share the same idea of using similar words as in (Li et al., 2016), which we think is inevitable if the OOV words are totally new.

5 Experiments

5.1 Data

The experiments were conducted for Chinese-English translation using two datasets, the relatively small IWSLT dataset, and the much larger NIST dataset. As we will see, the NMT and SMT approaches exhibit different behaviours on these two datasets, and the memory-augmentation approach offers different contributions to them.

The IWSLT corpus The training data consists of 44K sentences from the tourism and travel domain. The development set was composed of the ASR devset 1 and devset 2 from IWSLT 2005, and testing used the IWSLT 2005 test set.

The NIST corpus The training data consists of 1M sentence pairs with 19M source tokens and 24M target tokens from the LDC corpora of LDC2002E18, LDC2003E07, LDC2003E14, and Hansard's portion of LDC2004T07, LDC2004T08 and LDC2005T06. We use the NIST 2002 test set as the development set and the NIST 2003 test set as the test set.

Memory data To construct the memory, we used the GIZA++ toolkit (Koehn et al., 2003) to align the training data in both directions, and kept the word pairs that appeared in the phrase tables of both directions. The global memory size is $80K$ for the IWSLT task, and $500K$ for the NIST task. These word pairs were then filtered according to the conditional probability $p(w_t|w_s)$ where w_s and w_t are source and target language words, respectively. For each w_s, at most two candidates of w_t were retained.

5.2 Systems

We used a conventional SMT system and an attention-based RNN NMT system as the baselines, and investigated a variety of M-NMT architectures.

SMT baseline: For the SMT system (denoted by *Moses*), Moses (Koehn et al., 2007), a state-of-the-art open-source toolkit, was used. The default configuration was used where the phrase length was 7 and the following features were employed: relative translation frequencies and lexical translation probabilities on both directions, distortion distance, language model and word penalty. For the language model, the KenLM toolkit (Heafield, 2011) was used to build a 5-gram language model (with the Keneser-Ney smoothing) on the target side of the training data.

NMT baseline: For NMT, we reproduced the attention-based RNN model proposed by Bahdanau et al. (2015), which is denoted by *NMT*. The implementation was based on Tensorflow[1]. We compared our implementations with a public implementation using Theano[2], and achieved comparable (even slightly better) performances on the same data sets with the same parameter settings.

M-NMT system: The M-NMT system was implemented by combining the memory structure and the NMT system. The model part is the same as the NMT baseline, while the attention function of

[1] https://www.tensorflow.org/

[2] https://github.com/lisa-groundhog/GroundHog

System	Attending	Attended
M-NMT(s, u^y)	s_{i-1}	$u_k(y)$
M-NMT(s, u^{xy})	s_{i-1}	$u_k(x), u_k(y)$
M-NMT(sy, u^y)	s_{i-1}, y_{i-1}	$u_k(y)$
M-NMT(sy, u^{xy})	s_{i-1}, y_{i-1}	$u_k(x), u_k(y)$

Table 2: M-NMT systems with different configurations.

the memory part was trained. During the training, if the target word is an UNK symbol, or the target word is not in the memory (due to the limited word pairs in the memory), this word is simply skipped from back-propagation. This skipping is important as it avoids bias caused by the large amount of UNK symbols. The trained M-NMT system can be readily used to deal with OOV words, without any re-training.

For M-NMT, the complete form of the relevance function for memory attention is $e^m_{ik}(s_{i-1}, y_{i-1}, u_k)$. Of these, s_{i-1} and y_{i-1} are 'attending factors' that represent the information used 'to attend', while u_k, which consists of a source part $u_k(x)$ and a target part $u_k(y)$, involves 'attended factors' that represent the content 'to be attended'. To investigate the contribution of different attending and attended factors, these factors are combined in different ways, leading to different M-NMT variants, as shown in 2. Note that *M-NMT($s; u^y$)* is the simplest configuration and the attention essentially learns a target-side language model. *M-NMT($s; u^{xy}$)* involves the source part of the memory, which implicitly learns a bilingual language model. Involving the decoding history y_{i-1} makes this learning more explicit.

Settings For a fair comparison, the models configurations in the NMT system and the M-NMT system were intentionally set to be identical. The number of hidden units, the word embedding dimensionality and the vocabulary size were empirically set to 500, 310 and 30000, respectively. In the training process, the batch size of the SGD algorithm was set to 80, and the parameters for AdaDelta were set to be $\rho = 0.95$ and $\epsilon = 10^{-6}$. The decoding is implemented as a beam search, where the beam size was set to be 5.

Evaluation metrics The translation performance was evaluated using the BLEU score with case-insensitive n $\leq$ 4-grams (Papineni et al., 2002).

System	IWSLT05	NIST03
Moses	52.5	30.6
NMT	43.9	31.3
NMT-L Arthur et al.	45.9	31.7
M-NMT(s, u^y)	49.8	32.3
M-NMT(sy, u^y)	50.7	32.5
M-NMT(s, u^{xy})	51.4	32.8
M-NMT(sy, u^{xy})	**52.9**	**34.0**

Table 3: BLEU scores with different translation systems on the two Chinese-English translation datasets.

5.3 SMT-NMT Integration Experiments

In the first experiment, the M-NMT architecture combined SMT and NMT by using SMT to construct the memory to assist with NMT. For comparison purposes, the lexical prediction approach proposed by (Arthur et al., 2016) was also implemented. This uses the phrase table produced by SMT to improve NMT. Our implementation is a linear combination, and for a fair comparison, the neural model part was kept unchanged. At each step i, the auxiliary probability provided by the lexical part is $P(y_i) = \sum_j \alpha_{ij} P(y_i|x_j)$, where α_{ij} is the attention weight from the neural model, and $P(y_i|x_j)$ is obtained from the phrase table. This can be regarded as a simple memory approach, with memory attention borrowed from the neural model, rather than being learned separately.

Table 3 shows the BLEU results with different systems. Firstly, it can be observed that with the small IWSLT05 dataset, the SMT outperforms the baseline NMT, but with the large NIST dataset, NMT outperforms SMT. This is unsurprising as neural models often need more training data. Secondly, the results show that with both datasets, the lexical approach (NMT-L) can improve NMT performance, showing that using SMT knowledge helps NMT. However, the improvement seems less significant than reported in (Arthur et al., 2016). This is likely to be because our implementation focuses on creating a simple, extensible, and generalizable system, and therefore does not allow re-training the neural model.

The M-NMT system provides significant performance improvement, even with the simplest setting (M-NMT(s, u^y)). More information factors tend to offer better performance, and the best M-NMT system, M-NMT(sy, u^{xy}), outperforms the baseline NMT by 9.0 and 2.7 BLEU points

	T-INV		T-OOV	
System	Recall	BLEU	Recall	BLEU
NMT	0.06	15.1	0	13.7
M-NMT	0.05	16.0	0	14.6
NMT-PL Luong et al.	0.09	15.4	0.08	14.3
M-NMT+OOV	**0.28**	**17.0**	**0.40**	**15.9**

Table 4: The OOV recall rates and BLEU scores on sentences with OOV words. 'T-INV' refers to the case where the target words of the OOV input are in-vocabulary, and 'T-OOV' means the case where the target words are also OOV.

on the two datasets respectively. Notably, the improvement with the IWSLT05 dataset is impressive, the best M-NMT system outperforms even the very strong SMT baseline, which strongly supports our conjecture that NMT must be equipped with a symbolic structure to deal with infrequent words. It also suggests that the M-NMT architecture is a promising way to apply neural methods to low-resource tasks.

5.4 OOV Treatment Experiments

Here the M-NMT architecture was used to handle OOV words. The experiments were conducted on the NIST dataset, for which we collected 312 test sentences containing OOV words. This test set was divided into two subsets: the T-INV set, containing sentences with source OOV words whose translations are NOT OOV in the target language; and the T-OOV set, containing sentences with OOV words that are OOV in both source and translation. There were 491 source-side OOV words in total, among which 276 words have in-vocabulary translations and 215 words only have OOV translation. We constructed a translation table with three items for each OOV word: (1) its translation; (2) its similar word; (3) the similar word of its translation, if the translation is also an OOV word. All the above was designed by hand, and for each OOV word, there was only a single translation. Although it is not difficult to collect most of this information automatically (e.g., by using an SMT phrase table), we are simulating the scenario where OOV words are newly coined, or where the system is migrated to a new domain, meaning that some words are totally new to the system. Handling OOV words of this type is certainly challenging, but it is also practically valuable.

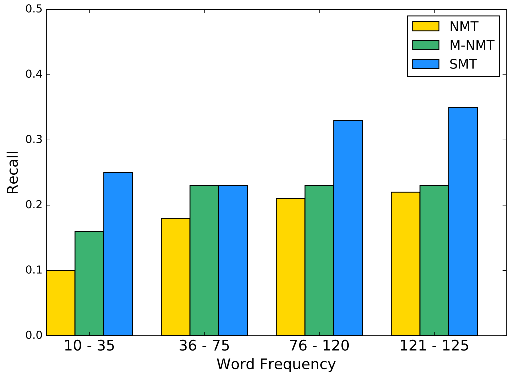


Figure 2: The recall rates of words in different frequency bins.

src.	人类共有二十三对染色体。
ref.	Humans have 23 pairs of chromosomes.
Moses	A total of 23 human chromosome.
NMT	There are 23-year history of human history .
M-NMT	There have a total of 23 species of chromosomes .

Table 5: The translations from different systems for the Chinese-to-English 'meaning drift' example.

For comparison, the place-holder approach proposed by Luong et al. (2014) was also implemented. Here, OOV words in the target language are substituted by position-aware UNKs, and a post-processing step is used to replace UNKs with the correct translation. We denote this system as 'NMT-PL'. For M-NMT, only the best configuration M-NMT(sy, u^{xy}) was tested in this experiment. Two scenarios were considered: the original M-NMT system, and the M-NMT system with OOV words involved in the memory (denoted by M-NMT+OOV).

The results with the NIST dataset are shown in Table 4. In addition to the BLEU scores, we also report the OOV recall rates, defined as the proportion of OOV words that are correctly translated. It can be seen that both the basic NMT and M-NMT systems work badly with OOV words: they can only process OOV words whose translations are not OOV in the target language, and the recall rate is very low (0.05 approximately). The placeholder approach (NMT-PL) can address both types of OOV words, but the recall rate is still low. The M-NMT system with OOV memory, in contrast, is much more effective in OOV word translation, as shown in Table 4. We also implemented the replace-and-restore approach reported by Li et al. (2016), but found performance to be poor (the BLEU scores are 13.9 on T-INV and 13.3 on T-

OOV). This may be due to no re-training of the neural model (again, in order to keep the extensibility and generalizability of the M-NMT system).

5.5 Frequency Analysis

To form a better understanding of the memory mechanism, we distribute the test sentences into four bins according to the lowest word frequency among the sentence and compute the recall rates for words in these bins. Once a generated translation word is also in one of the references, we treat it as one hit. The experiment was conducted with the NIST dataset. The results in Figure 2 show that M-NMT offers more improvement with infrequent words, in accordance with our argument that the memory mechanism helps NMT in dealing with infrequent words.

Finally, we demonstrate the translation with M-NMT for the example in Table 1, as shown in Table 5. It can be clearly seen that the memory has remembered the infrequent word 'Chromosome', which resulted in an improved translation.

6 Conclusions

This paper presented a memory-augmented NMT approach, which introduces a memory mechanism for conventional NMT to assist with translating words not well learned by the neural model. Our experiments demonstrated that the new architecture is highly effective, providing performance improvement by 9.0 and 2.7 BLEU scores on two Chinese-English translation datasets, respectively. Additionally, it offers a very flexible and effective OOV treatment. In our experiments, The OOV recalls are 28% and 40% for the OOV words whose target words are INV and OOV, respectively, a significant improvement on competing methods. Future work will investigate better model-memory integration, e.g., by joint training.

Acknowledgement

This paper was supported by the National Natural Science Foundation of China (NSFC) under the project NO.61371136, NO.61633013, NO.61472428, NO.61472428.

References

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proc. of EMNLP*, pages 1557–1567.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR'2015*.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. *arXiv preprint arXiv:1608.04631*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proc. of NAACL*, pages 876–885.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing machines. *arXiv preprint arXiv:1410.5401*.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proc. of the Sixth Workshop on Statistical Machine Translation*, pages 187–197.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *ACL'15*.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch Mayne, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL, Demonstration Session*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL*, pages 48–54. Association for Computational Linguistics.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. Towards zero unknown word in neural machine translation. In *Proc. of IJCAI*, pages 2852–2858.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. *arXiv preprint arXiv:1605.03148*.

Frans J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29:19–51.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL 16*.

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016. Syntactically guided neural machine translation. In *Proc. of ACL*, pages 299–305.

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

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Memory-enhanced decoder for neural machine translation. In *Proc. of EMNLP*, pages 278–286.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

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

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Jiajun Zhang, Chengqing Zong, et al. 2015. Deep neural networks in machine translation: An overview. *IEEE Intelligent Systems*, 30(5):16–25.

Jiyuan Zhang, Yang Feng, Dong Wang, Yang Wang, Andrew Abel, Shiyue Zhang, and Andi Zhang. 2017. Flexible and creative chinese poetry generation using neural memory. *arXiv preprint arXiv:1705.03773*.

Dynamic Data Selection for Neural Machine Translation

Marlies van der Wees
Informatics Institute
University of Amsterdam

Arianna Bisazza*
LIACS
Leiden University

Christof Monz
Informatics Institute
University of Amsterdam

Abstract

Intelligent selection of training data has proven a successful technique to simultaneously increase training efficiency and translation performance for phrase-based machine translation (PBMT). With the recent increase in popularity of neural machine translation (NMT), we explore in this paper *to what extent* and *how* NMT can also benefit from data selection. While state-of-the-art data selection (Axelrod et al., 2011) consistently performs well for PBMT, we show that gains are substantially lower for NMT. Next, we introduce *dynamic data selection* for NMT, a method in which we vary the selected subset of training data between different training epochs. Our experiments show that the best results are achieved when applying a technique we call *gradual fine-tuning*, with improvements up to +2.6 BLEU over the original data selection approach and up to +3.1 BLEU over a general baseline.

1 Introduction

Recent years have shown a rapid shift from phrase-based (PBMT) to neural machine translation (NMT) (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014) as the most common machine translation paradigm. With large quantities of parallel data, NMT outperforms PBMT for an increasing number of language pairs (Bojar et al., 2016). Unfortunately, training an NMT model is often a time-consuming task, with training times of several weeks not being unusual.

Despite its training inefficiency, most work in NMT greedily uses all available training data for a given language pair. However, it is unlikely that all data is equally helpful to create the best-performing system. In PBMT, this issue has been addressed by applying intelligent data selection, and it has consistently been shown that using more data does not always improve translation quality (Moore and Lewis, 2010; Axelrod et al., 2011; Gascó et al., 2012). Instead, for a given translation task, the training bitext likely contains sentences that are irrelevant or even harmful, making it beneficial to keep only the most relevant subset of the data while discarding the rest, with the additional benefit of smaller models and faster training.

Motivated by the success of data selection in PBMT, we investigate in this paper *to what extent* and *how* NMT can benefit from data selection as well. While data selection has been applied to NMT to reduce the size of the data (Cho et al., 2014; Luong et al., 2015b), the effects on translation quality have not been investigated. Intuitively, and confirmed by our exploratory experiments in Section 5.1, this is a challenging task; NMT systems are known to under-perform when trained on limited parallel data (Zoph et al., 2016; Fadaee et al., 2017), and do not have a separate large-scale target-side language model to compensate for smaller parallel training data.

To alleviate the negative effect of small training data on NMT, we introduce *dynamic data selection*. Following conventional data selection, we still dramatically reduce the training data size, favoring parts of the data which are most relevant to the translation task at hand. However, we exploit the fact that the NMT training process iterates over the training corpus in multiple epochs, and we alter the quantity or the composition of the training data *between epochs*. The proposed method requires no modifications to the NMT architecture or parameters, and substantially speeds up training times while simultaneously improving translation quality with respect to a complete-bitext baseline.

[*] Work done while at University of Amsterdam

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1400–1410
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

In summary, our contributions are as follows:

(i) We compare the effects of a commonly used data selection approach (Axelrod et al., 2011) on PBMT and NMT using four different test sets. We find that this method is much less effective for NMT than for PBMT, while using the exact same training data subsets.

(ii) We introduce dynamic data selection as a way to make data selection profitable for NMT. We explore two techniques to alter the selected data subsets, and find that our method called *gradual fine-tuning* improves over conventional static data selection (up to +2.6 BLEU) and over a high-resource general baseline (up to +3.1 BLEU). Moreover, gradual fine-tuning approximates in-domain fine-tuning in ~20% of the training time, even when no parallel in-domain data is available.

2 Static data selection

As a first step towards dynamic data selection for NMT, we compare the effects of a commonly used, state-of-the-art data selection method (Axelrod et al., 2011) on both neural and phrase-based MT. Briefly, this approach ranks sentence pairs in a large training bitext according to their difference in cross-entropy with respect to an in-domain corpus (i.e., a corpus representing the test data) and a general corpus. Next, the top n sentence pairs with the highest rank—thus lowest cross-entropy—are selected and used for training an MT system.

Formally, given an in-domain corpus I, we first create language models from the source side f of I ($LM_{I,f}$) and the target side e of I ($LM_{I,e}$). We then draw a random sample (similar in size to I) of the large general corpus G and create language models from the source and target sides of G: $LM_{G,f}$ and $LM_{G,e}$, respectively. Note that the data for creating these LMs need not be parallel but can be independent corpora in both languages.

Next, we compute for each sentence pair s in G four cross-entropy scores, defined as:

$$H_{C,s_b} = -\sum p\left(s_b\right) \log\left(LM_{C,b}\left(s_b\right)\right), \qquad (1)$$

where $C \in \{I, G\}$ is the corpus, $b \in \{f, e\}$ refers to the bitext side, and s_b is the bitext side b of sentence pair s in the parallel training corpus.

To find sentences that are similar to the in-domain corpus, i.e., have low H_I, and at the same time dissimilar to the general corpus, i.e., have high H_G, we compute for each sentence pair s the bilingual cross-entropy difference CED_s following Axelrod et al. (2011):

$$CED_s = (H_{I,s_f} - H_{G,s_f}) + (H_{I,s_e} - H_{G,s_e}). \qquad (2)$$

Finally, we rank all sentence pairs $s \in G$ according to their CED_s, and then select only the top n sentence pairs with the lowest CED_s.

Following related work by Moore and Lewis (2010), we restrict the vocabulary of the LMs to the words occurring at least twice in the in-domain corpus. To analyze the quality of the selected data subsets, we also run experiments on random selections, all performed in threefold. Finally, we always use the exact same selection of sentence pairs in equivalent PBMT and NMT experiments.

LSTM versus n-gram The described data selection method uses n-gram LMs to determine the domain-relevance of sentence pairs. We adhere to this setting for our comparative experiments on PBMT and NMT (Section 5.1). However, when applying data selection to NMT, we examine the potential benefit of replacing the conventional n-gram LMs with LSTMs[1]. These have the advantage to remember longer histories, and do not have to back off to shorter histories when encountering out-of-vocabulary words. In this neural variant to rank sentences, the score for each sentence pair in G is still computed as the bilingual cross-entropy difference in Equation (2). In addition, we use the same in-domain and general corpora as with the n-gram method, and we again restrict the vocabulary to the most frequent words.

3 Dynamic data selection

While data selection aims to discard irrelevant data, it can also exacerbate the problem of low vocabulary coverage and unreliable statistics for rarer words in the 'long tail', which are major issues in NMT (Luong et al., 2015b; Sennrich et al., 2016b). In addition, it has been shown that NMT performance drops tremendously in low-resource scenarios (Zoph et al., 2016; Fadaee et al., 2017; Koehn and Knowles, 2017).

To overcome this problem, we introduce *dynamic data selection*, in which we vary the selected data subsets *during* training. Unlike other MT paradigms, which require training data to be fixed during the entire training process, NMT iterates over the training corpus in several epochs,

[1] We use four-layer LSTMs with embedding and hidden sizes of 1,024, which we train for 30 epochs.

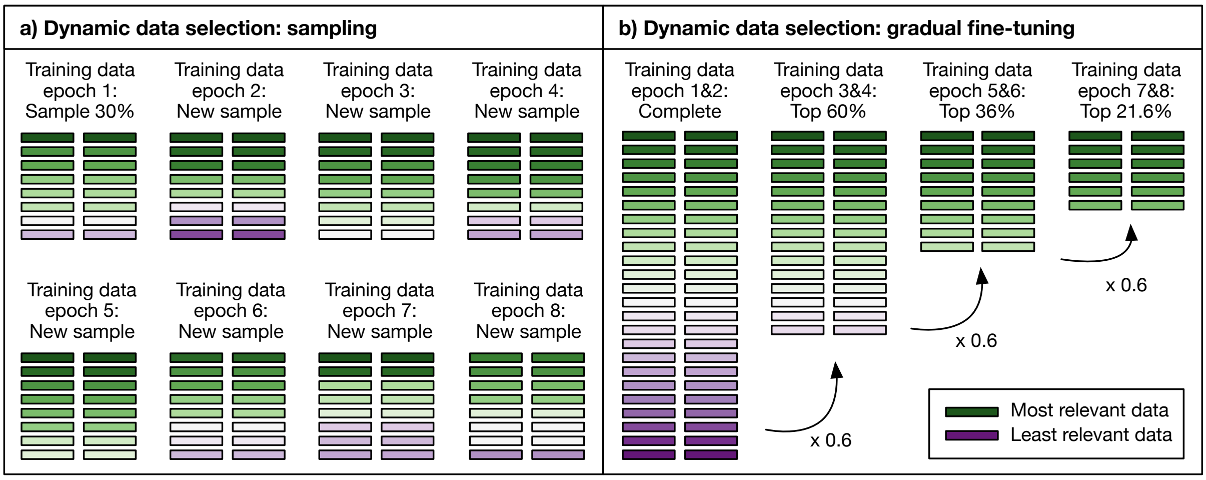

a) Dynamic data selection: sampling
Training data epoch 1: Sample 30%
Training data epoch 2: New sample
Training data epoch 3: New sample
Training data epoch 4: New sample
Training data epoch 5: New sample
Training data epoch 6: New sample
Training data epoch 7: New sample
Training data epoch 8: New sample
b) Dynamic data selection: gradual fine-tuning
Training data epoch 1&2: Complete
Training data epoch 3&4: Top 60%
Training data epoch 5&6: Top 36%
Training data epoch 7&8: Top 21.6%
x 0.6
x 0.6
x 0.6
Most relevant data
Least relevant data

Figure 1: Illustration of two dynamic bitext selection techniques for NMT: *sampling* (left) and *gradual fine-tuning* (right). Measured over 16 training epochs (which is used in this work), the total training time of both examples would be ∼30% of the training time needed when using the complete bitext.

allowing to use a different subset of the training data in every epoch.

Dynamic data selection starts from a relevance-ranked bitext, which we create using CED scores as computed in Equation (2). Given this ranking, we investigate two dynamic data selection techniques[2] that vary per epoch the composition or the size of the selected training data. Both techniques aim to favor highly relevant sentences over less relevant sentences while not completely discarding the latter. In all experiments, we use a fixed vocabulary created from the complete bitext.

While we use in this work a domain-relevance ranking of the bitext following Axelrod et al. (2011), dynamic data selection can also be applied using other ranking criteria, for example limiting redundancy in the training data (Lewis and Eetemadi, 2013) or complementing similarity with diversity (Ruder and Plank, 2017).

Sampling sentence pairs In the first technique, illustrated in Figure 1a, we sample for every epoch n sentence pairs from G, using a distribution computed from the domain-specific CED_s scores. Concretely, this is done as follows:

First, since higher ranked sentence pairs have lower CED_s scores, and they can be either negative or positive, we scale and invert CED_s scores such that $0 \leq CED'_s \leq 1$ for each sentence pair $s \in G$:

$$CED'_s = 1 - \frac{CED_s - \min(CED_G)}{\max(CED_G) - \min(CED_G)}, \qquad (3)$$

where CED_G refers to the set of CED_s scores for bitext G.

Next, we convert CED'_s scores to relative weights, such that $\sum_{s \in G} w(s) = 1$:

$$w(s) = \frac{CED'_s}{\sum_{s_i \in G} CED'_{s_i}}. \qquad (4)$$

We then use $\{w(s) : s \in G\}$ to perform weighted sampling, drawing for each epoch n sentence pairs without replacement. While all selection weights are very close to zero, higher ranked sentences have a noticeably higher probability of being selected than lower-ranked sentences; in practice we find that top-ranked sentences get selected in nearly each epoch, while bottom-ranked sentence pairs get selected at most once. Note that the sampled selection for any epoch is independent of selections for all other epochs.

Gradual fine-tuning The second dynamic data selection technique, see Figure 1b, is inspired by the success of domain-specific fine-tuning (Luong and Manning, 2015; Zoph et al., 2016; Sennrich et al., 2016a; Freitag and Al-Onaizan, 2016), in which a model trained on a large general-domain bitext is trained for a few additional epochs only on small in-domain data. However, rather than training a full model on the complete bitext G, we gradually decrease the training data size, starting from G and keeping only the top n sentence pairs for the duration of η epochs, where the top n pairs are defined by their CED_s scores. Given its resemblance to fine-tuning, we refer to this variant as *gradual fine-tuning*.

[2] Code for bitext ranking and both selection techniques: github.com/marliesvanderwees/dds-nmt.

During gradual fine-tuning, the selection size n is a function of epoch i:

$$n(i) = \alpha \cdot |G| \cdot \beta^{\lfloor (i-1)/\eta \rfloor}. \qquad (5)$$

Here $0 \leq \alpha \leq 1$ is the *relative start size*, i.e., the fraction of general bitext G used for the first selection, $0 \leq \beta \leq 1$ is the *retention rate*, i.e., the fraction of data to be kept in each new selection, and $\eta \geq 1$ is the number of consecutive epochs each selected subset is used. Note that $\lfloor i/\eta + 1 \rfloor$ indicates rounding down $i/\eta + 1$ to the nearest integer. For example, if we start with the complete bitext ($\alpha = 1$), select the top 60% ($\beta = 0.6$) every second epoch ($\eta = 2$), then we run epochs 1 and 2 with a subset of size $|G|$, epochs 3 and 4 with a subset of size $0.6 \cdot |G|$, epochs 5 and 6 with a subset of size $0.36 \cdot |G|$, and so on. For every size n, the actual selection contains the top n sentences pairs of G.

4 Experimental settings

We evaluate static and dynamic data selection on a German→English translation task comprising four test sets. Below we describe the MT systems and data specifications.

4.1 Machine translation systems

While the main aim of this paper is to improve data selection for NMT, we also perform comparative experiments using PBMT. Our PBMT system is an in-house system similar to Moses (Koehn et al., 2007). To create optimal PBMT systems given the available resources, we apply test-set-specific parameter tuning using PRO (Hopkins and May, 2011). In addition, we use a linearly interpolated target-side language model trained with Kneser-Ney smoothing on 480M tokens of data in various domains. LM interpolation weights are also optimized per test set. Consistent with Axelrod et al. (2011), we do not vary the target-side LM between different experiments on the same test set. All n-gram models in our work are 5-gram.

For our NMT experiments we use an in-house encoder-decoder[3] model with global attention as described in Luong et al. (2015a). This choice comes at the cost of optimal translation quality but allows for a relatively fast realization of large-scale experiments given our available resources. Both the encoder and decoder are four-layer unidirectional LSTMs, with embedding and layer sizes of 1,000. We uniformly initialize all parameters, and use SGD with a mini-batch size of 64 and an initial learning rate of 1, which is decayed by a factor two every epoch after the fifth epoch. We use dropout with probability 0.3, and a beam size of 12. We train for 16 epochs and test on the model from the last epoch. All NMT experiments are run on a single NVIDIA Titan X GPU.

	Train		Dev/valid		Test	
Corpus	Lines	Tokens	Lines	Tokens	Lines	Tokens
EMEA	206K	3.3M	3.9K	59K	5.8K	93K
Movies	101K	1.2M	4.5K	54K	7.1K	87K
TED	189K	3.3M	2.5K	50K	5.4K	99K
WMT	3.8M	84M	3.0K	64K	3.0K	65K
Mix	4.3M	92M	3.5K	61K	–	–

Table 1: Data specifications with tokens counted on the German side. The WMT training corpus contains Commoncrawl, Europarl, and News Commentary but no in-domain news data.

4.2 Training and evaluation data

We evaluate all experiments on four domains: (i) EMEA medical guidelines (Tiedemann, 2009), (ii) movie dialogues (van der Wees et al., 2016) constructed from OpenSubtitles (Lison and Tiedemann, 2016), (iii) TED talks (Cettolo et al., 2012), and (iv) WMT news. For TED, we use IWSLT2010 as development set and IWSLT2011-2014 as test set, and for WMT we use newstest2013 as development set and newstest2016 as test set. We train our systems on a mixture of domains, comprising Commoncrawl, Europarl, News Commentary, EMEA, Movies, and TED. Corpus specifications are listed in Table 1.

The in-domain LMs used to rank training sentences for data selection are trained on small portions of in-domain parallel data whenever available (3.3M, 1.2M and 3.3M German tokens for EMEA, Movies and TED, respectively). Since no sizeable in-domain parallel text is available for WMT, we independently sample 200K sentences from the WMT monolingual News Crawl corpora (3.3M German tokens or 3.5M English tokens). This demonstrates the applicability of data selection techniques even in cases where one lacks parallel in-domain data.

Before running data selection, we preprocess our data by tokenizing, lowercasing and remov-

[3] github.com/ketranm/tardis

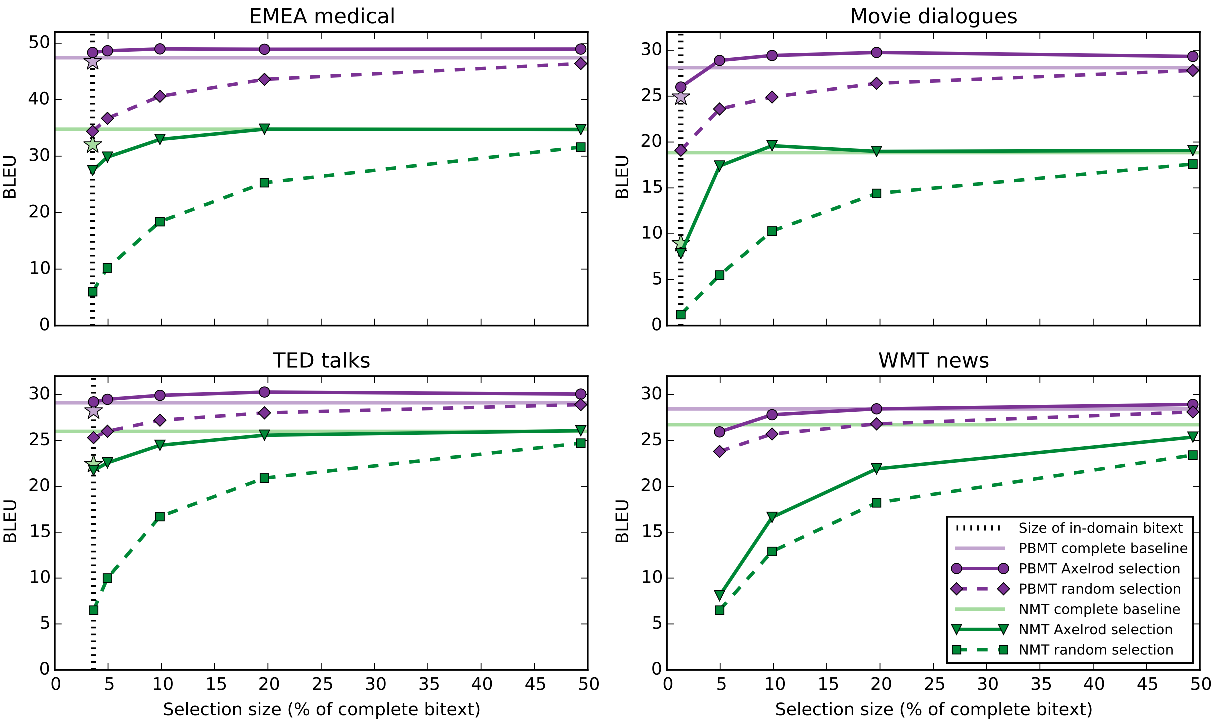


Figure 2: PBMT (purple) and NMT (green) German→English results of Axelrod data selection and random data selection (average of three runs) for four domains. Purple and green stars indicate BLEU scores when only the available in-domain data is used. We use selections of the in-domain size $|I|$, and 5%, 10%, 20%, and 50% of the complete bitext, which are exactly the same for PBMT and NMT.

ing sentences that are longer than 50 tokens or that are identified as a different language. After selection, we apply Byte-pair encoding (BPE, Sennrich et al. (2016b)) with 40K merge operations on either side of the complete mix-of-domains training bitext. For our NMT experiments we use BPE-processed corpora on both bitext sides, while for PBMT we only apply BPE to the German side. Our NMT systems use a vocabulary size of 40K on both the source and target side.

5 Results

Below we discuss the results of our translation experiments using static and dynamic data selection, measuring translation quality with case-insensitive untokenized BLEU (Papineni et al., 2002).

5.1 Static data selection for PBMT and NMT

We first compare the effects of static data selection with n-gram LMs on both NMT and PBMT using various selection sizes. Concretely, we select the top n sentence pairs such that the number of selected tokens $t \in \{5\%, 10\%, 20\%, 50\%\}$ of G, or $t = |I|$ (the in-domain corpus size). Figure 2 shows German→English translation performance in BLEU for our four test sets. The benefits of n-gram-based data selection for PBMT (purple circles) are confirmed: In all test sets, the selection of size $|I|$ (dotted vertical line) yields better performance than using only the in-domain data of the exact same size (purple star), and at least one of the selected subsets—often using only 5% of the complete bitext—outperforms using the complete bitext (light purple line). We also show that the informed selections are superior to random selections of the same size (purple diamonds).

In NMT, results of n-gram-based data selection (green triangles) vary: While for Movies a selection of only 10% outperforms the complete bitext (light green line), none of the selected subsets for other test sets is noticeably better than the full bitext.[4] Interestingly, the same selections of size $|I|$ that proved useful in PBMT, never beat the system that uses exactly the available in-domain data (green star), indicating that the current selections can be further improved for NMT. In all scenarios we see that NMT suffers much more from small-data settings than PBMT. Finally, the random se-

[4] Validation cross-entropy converges after 10–12 epochs, never reaching the scores of the complete bitext.

lections (green squares) show that NMT not only needs large quantities of data, but it is also affected when the selected data is of low quality. In PBMT, both low-quantity and low-quality scenarios appear to be compensated for by the large monolingual LM on the target side.

When comparing the different test sets, we observe that the impact of domain mismatch in NMT with respect to PBMT is largest for the two domains that are most distinct from the general bitext, EMEA and Movies. For WMT, both MT systems achieve very similar baseline results, but translation quality deteriorates considerably in data selection experiments, which is likely caused by the lack of in-domain data in the general bitext.

LSTM versus n-gram Before proceeding with dynamic data selection for NMT, we test whether bitext ranking for NMT can be improved using LSTMs rather than conventional n-gram LMs. Table 2 shows NMT BLEU scores of a few different sizes of selected subsets created using n-gram LMs or LSTMs. While results vary among test sets and selection sizes, we observe an average improvement of 0.4 BLEU when using LSTMs instead of n-gram LMs. For PBMT, similar results have been reported when replacing n-gram LMs with recurrent neural LMs (Duh et al., 2013). In all subsequent experiments we use relevance rankings computed with LSTMs instead of n-gram LMs.

Selection	LM type	EMEA	Movies	TED	WMT
5%	n-gram	29.8	17.4	22.6	8.1
	LSTM	**30.0**	**17.8**	22.6	**9.6**
10%	n-gram	33.0	19.6	24.5	16.6
	LSTM	33.0	**19.7**	**24.7**	**17.4**
20%	n-gram	**34.8**	19.0	25.6	21.9
	LSTM	34.5	**19.6**	**26.6**	21.9

Table 2: NMT BLEU comparison between using n-gram LMs and LSTMs for bitext ranking. Selection sizes concern the selected bitext subsets; LMs are created from the exact same in-domain data.

5.2 Dynamic data selection for NMT

Equipped with a relevance ranking of sentence pairs in bitext G, we now examine two variants of dynamic data selection as described in Section 3.

We are interested in reducing training time while limiting the negative effect on BLEU for various domains. Therefore we report BLEU as well as the *relative training time* of each experiment. Since wall-clock times depend on other factors such as the NMT architecture and memory speed, we define training time as the total number of tokens observed while training the NMT system, i.e., the sum of tokens in the selected subsets of all epochs. We report all training times relative to the training time of our complete-bitext baseline (i.e., 4.3M tokens × 16 epochs). Note that this measure of training time corresponds closely but not exactly to the number of model updates, as the latter relies on the number of sentences, which vary in length, rather than the number of tokens in the training data. For completeness: Training the 100% baseline takes 106 hours, while our fastest dynamic selection variant takes 19–21 hours. Computing CED scores takes ∼15 minutes when using n-gram LMs and 5–6 hours when using LSTMs.

Figure 3 shows BLEU scores of some selected experiments as a function of relative training time. Compared to static data selection (blue lines), our weighted sampling technique (orange triangles) yields variable results. When sampling a subset of 20% of $|G|$ from the top 50% of the ranked bitext, we obtain small improvements for TED and WMT, but small drops for EMEA and Movies. Other selection sizes (30% and 40%, not shown) give similar results lacking a consistent pattern.

By contrast, our gradual fine-tuning method performs consistently better than static selection, and even beats the general baseline in three out of four test sets. The displayed version uses settings ($\alpha = 0.5, \beta = 0.7, \eta = 2$) and is at least as fast as static selection using 20% of the bitext, yielding up to +2.6 BLEU improvement (for WMT news) over this static version. Compared to the complete baseline, this gradual fine-tuning method improves up to +3.1 BLEU (for TED talks).

Table 3 provides detailed information on additional experiments using other settings. For all three test domains which are covered in the parallel data—EMEA, Movies and TED—improvements are highest when starting gradual fine-tuning with only the top 50% of the ranked bitext, which are also the fastest approaches. For WMT, which is not covered in the general bitext, adding more data clearly benefits translation quality. These findings are consistent with the static data selection patterns; Using low-ranked sentences on top of the most relevant selection

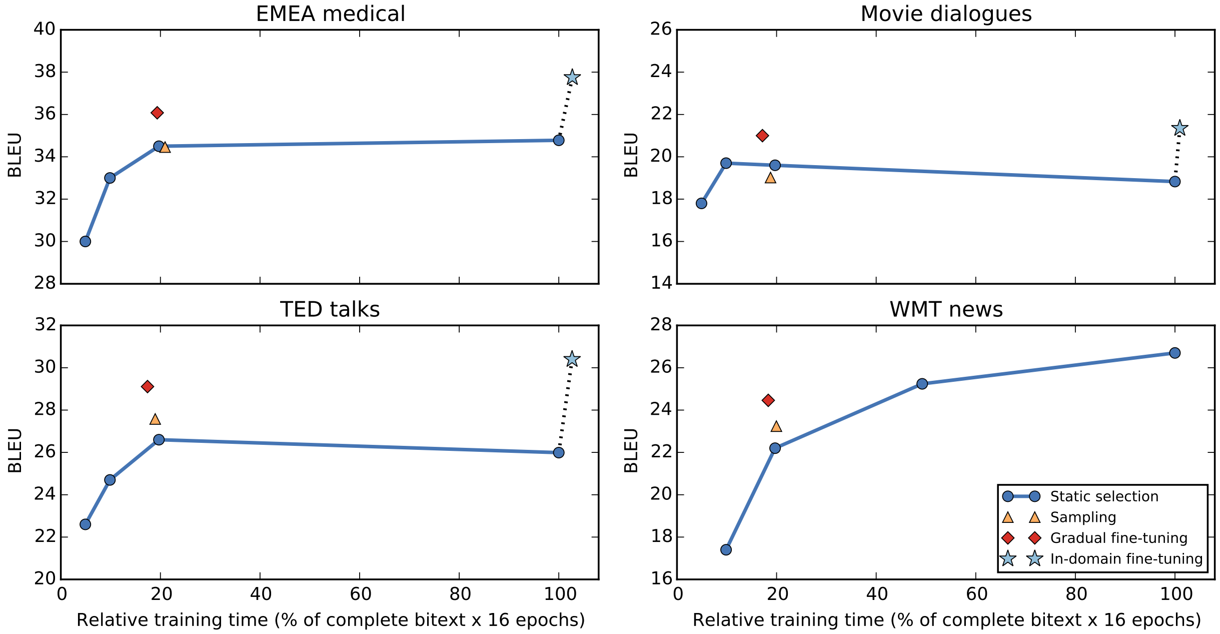


Figure 3: Selected German→English translation results of dynamic data selection methods (orange and red markers) compared to conventional static data selection (blue circles). *Relative training time* equals the total number of training tokens relative to the complete baseline, which takes 106 hours to train and is represented by the rightmost blue circle. Note that no parallel in-domain data is available for WMT news. All y-axes are scaled equally for easy comparison of BLEU differences across domains.

Experiment			Relative training time	BLEU			
Start size	Retention rate β	Decrease every		EMEA	Movies	TED	WMT
Static selection top 20%			20%	34.5	19.6	26.6	21.9
50% ($\alpha = 0.5$)	0.7	$\eta = 2$ epochs	18–20%	**36.1 (+1.6)**	21.0 (+1.4)	**29.1 (+2.5)**	24.5 (+2.6)
50% ($\alpha = 0.5$)	0.5	$\eta = 4$ epochs	21–23%	36.0 (+1.5)	**21.2 (+1.6)**	29.0 (+2.4)	25.0 (+3.1)
50% ($\alpha = 0.5$)	0.6	$\eta = 4$ epochs	25–27%	35.6 (+1.1)	21.0 (+1.4)	28.5 (+1.9)	25.1 (+3.2)
100% ($\alpha = 1$)	0.6	$\eta = 2$ epochs	29–31%	35.5 (+1.0)	21.1 (+1.5)	29.0 (+2.4)	25.6 (+3.7)
100% ($\alpha = 1$)	0.7	$\eta = 2$ epochs	37–39%	35.9 (+1.4)	20.4 (+0.8)	28.2 (+1.6)	25.8 (+3.9)
100% ($\alpha = 1$)	0.9	$\eta = 1$ epoch	50–52%	35.4 (+0.9)	19.6 (±0.0)	27.4 (+0.8)	**26.1 (+4.2)**
Complete bitext baseline			100%	34.8	18.8	26.0	26.7
Gold: fine-tuning on in-domain data			101–103%	37.7	21.3	30.4	–

Table 3: German→English BLEU results of various gradual fine-tuning experiments sorted by relative training time. Indicated improvements are with respect to static selection using 20% of the bitext, and highest scores per test set are bold-faced. Results from the first experiment are also shown in Figure 3.

does not improve translation performance for any domain except WMT news.

Finally, we compare our data selection experiments to domain-specific fine-tuning (light blue stars in Figure 3), which is the current state-of-the-art for domain adaptation in NMT. To this end, we first train a model on the complete bitext, and then train for twelve additional epochs on available in-domain data, using an initial learning rate of 1 which halves every epoch. Depending on the test set, this approach yields +2.5–4.4 BLEU improvements over our baselines, however it does not speed up training and requires a parallel in-domain text which may not be available (e.g., for WMT). While none of our data selection experiments outperforms domain-specific fine-tuning, we obtain competitive translation quality in only 20% of the training time. In additional experiments we found that in-domain fine-tuning on top of our selection approaches does not yield improvements.

6 Further analysis

In this section we conduct a few additional experiments and analyses. We restrict to one parameter setting per selection approach: Static selection and sampling with 20% of the data, and gradual fine-tuning using ($\alpha = 0.5, \beta = 0.7, \eta = 2$). All have very similar training times.

First, we hypothesize that dynamic data selection works well because more different sentence pairs are observed during training, and it therefore increases coverage with respect to static data selection. To verify this, we measure for each test set the number of unseen source word types in the training data of different selection methods. Figure 4 shows indeed that the average number of unseen word types is reduced noticeably in both of our dynamic selection techniques, being much closer to the complete bitext baseline than to static selection. Note that all methods use the same vocabulary during training.

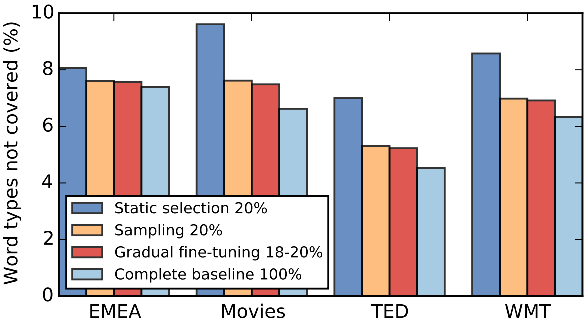


Figure 4: Test set source words not covered in the training data of different data selection methods.

Next, following the static data selection experiments in Section 5.1, we examine how well dynamic data selection performs using random selections. To this end, we repeat all techniques using a bitext which is ranked randomly rather than by its relevance to the test sets. The results in Table 4 show that the bitext ranking plays a crucial role in the success of data selection. However, the results also show that *even* in the absence of an appropriate bitext ranking, dynamic data selection—and in particular gradual fine-tuning—is still superior to static data selection. We explain this result as follows: Compared to static selection, both sampling and gradual fine-tuning have better coverage due to their improved exploration of the data. However, sampling also suffers from a surprise effect of observing new data in every epoch. Gradual fine-tuning on the other hand gradually improves learning on a subset of the selected data, suggesting that repetition across epochs has a positive effect on translation quality.

Ranking	Method	EMEA	Movies	TED	WMT
Relevance	Gradual FT	**36.1**	**21.0**	**29.1**	**24.5**
	Sampling 20%	34.5	19.0	27.6	23.2
	Static 20%	34.5	19.6	26.6	21.9
Random	Gradual FT	**29.2**	**16.1**	**23.2**	**21.3**
	Sampling 20%	26.7	14.4	22.0	19.8
	Static 20%	25.3	14.4	20.9	18.2

Table 4: BLEU scores of data selection using relevance versus random ranking of the bitext. Gradual fine-tuning uses ($\alpha = 0.5, \beta = 0.7, \eta = 2$), with relative training times of 18–20%.

One could expect that changing the data during training results in volatile training behavior. To test this, we inspect cross-entropy of our development sets after every training epoch. Figure 5 shows these results for TED. Clearly, static data selection converges most steadily. However, both dynamic selection techniques eventually converge to a lower cross-entropy value which is reflected by higher translation quality of the test set. We observe very similar behavior for the other test sets.

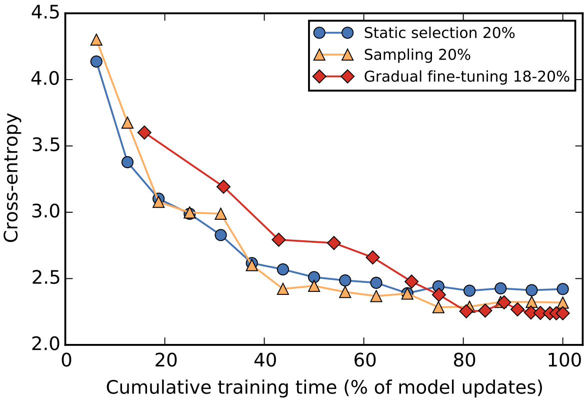


Figure 5: German→English cross-entropy of the TED dev set as a function of training time. Each data point represents a completed training epoch.

By its nature, our gradual fine-tuning technique uses training epochs of different sizes, and therefore also implicitly differs from other methods in its parameter optimization behavior. Since we decrease both the training data size and the SGD learning rate after finishing complete training epochs, we automatically decay the learning rate at decreasing time intervals. We therefore study how this approach is affected when we (i) decay the learning rate after a fixed number of updates (i.e., the same as in static data selection) rather than per epoch, or (ii) keep the learning rate fixed. In the first scenario, we observe that translation performance drops with –1.1–2.0 BLEU. When keeping a fixed learning rate, BLEU scores hardly change or even improve, indicating that the implicit change in search behavior may contribute to the success of gradual fine-tuning.

7 Related work

A few research topics are related to our work. Regarding data selection for SMT, previous work has targeted two goals; to reduce model sizes and training times, or to adapt to new domains. Data selection methods for domain adaptation mostly employ information theory metrics to rank training sentences by their relevance to the domain at hand. This has been applied monolingually (Gao et al., 2002) as well as bilingually (Yasuda et al., 2008). In more recent work, training sentences are typically ranked according to their cross-entropy *difference* between in-domain and general-domain data (Moore and Lewis, 2010; Axelrod et al., 2011, 2015), favoring sentences that are similar to the test domain and at the same time dissimilar from the general domain. Duh et al. (2013) and Chen and Huang (2016) present similar methods in which n-gram LMs are replaced by neural LMs or neural classifiers, respectively.

Data selection with the aim of model size and training time reduction has the objective to use the minimum amount of data while still maintaining high vocabulary coverage (Eck et al., 2005; Gascó et al., 2012; Lewis and Eetemadi, 2013). In a comparative study, Mirkin and Besacier (2014) find that similarity-objected methods perform best if the test domain and general corpus are very different, while a coverage-objected method is superior if test and general corpus are relatively similar. A comprehensive survey on data selection for SMT is provided by Eetemadi et al. (2015). While in this work we have used a similarity objective to rank our bitext, one could also apply dynamic data selection using a coverage objective.

In NMT, data selection can serve similar goals as in PBMT; increasing training efficiency or domain adaptation. Domain adaptation in NMT typically involves training a model on the complete bitext, followed by fine-tuning the parameters on a smaller in-domain corpus (Luong and Manning, 2015; Zoph et al., 2016). Other work combines fine-tuning with model ensembles (Freitag and Al-Onaizan, 2016) or with domain-specific tags in the training corpus (Chu et al., 2017). Finally, Sennrich et al. (2016a) adapt their systems by back-translating in-domain data, which is then added to the training data and used for fine-tuning.

Some other previous work has addressed training efficiency for NMT, for example by parallelizing models or data (Wu et al., 2016), modifying the NMT network structure (Kalchbrenner et al., 2016), decreasing the number of parameters through knowledge distillation (Crego et al., 2016; Kim and Rush, 2016), or by boosting parts of the data that are 'challenging' to the NMT system (Zhang et al., 2016). The latter is most related to our work since training data is also adjusted during training, however we reduce the training data size much more aggressively and study different techniques of data selection.

Finally, recent work comparing various aspects for PBMT and NMT includes (Bentivogli et al., 2016; Farajian et al., 2017; Toral and Sánchez-Cartagena, 2017; Koehn and Knowles, 2017).

8 Conclusions

With the recent increase in popularity of neural machine translation (NMT), we explored in this paper *to what extent* and *how* NMT can benefit from data selection. We first showed that a state-of-the-art data selection method yields unreliable results for NMT while consistently performing well for PBMT. Next, we have introduced *dynamic data selection* for NMT, which entails varying the selected subset of training data between different training epochs. We explored two techniques of dynamic data selection and found that our *gradual fine-tuning* technique, in which we gradually reduce training size, improves consistently over conventional static data selection (up to +2.6 BLEU) and over a high-resource general baseline (up to +3.1 BLEU). Moreover, gradual fine-tuning approximates in-domain fine-tuning using only $\sim$20% of the training time, even when no parallel in-domain data is available.

Acknowledgments

This research was funded in part by NWO under project numbers 639.022.213 and 639.021.646. We thank Ke Tran for providing the NMT system, and the reviewers for their valuable comments.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362.

Amittai Axelrod, Philip Resnik, Xiaodong He, and Mari Ostendorf. 2015. Data selection with fewer words. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 58–65.

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

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 257–267.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 conference on machine translation (WMT16). In *Proceedings of the first conference on machine translation (WMT16)*.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268.

Boxing Chen and Fei Huang. 2016. Semi-supervised convolutional networks for translation adaptation with tiny amount of in-domain data. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 314–323.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of simple domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Josep Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, et al. 2016. SYSTRAN's pure neural machine translation systems. *arXiv preprint arXiv:1610.05540*.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 678–683.

Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Low cost portability for statistical machine translation based on n-gram frequency and TF-IDF. In *Proceedings of the 2005 International Workshop on Spoken Language Translation*, pages 61–67.

Sauleh Eetemadi, William Lewis, Kristina Toutanova, and Hayder Radha. 2015. Survey of data-selection methods in statistical machine translation. *Machine Translation*, 29(3-4):189–223.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

M. Amin Farajian, Marco Turchi, Matteo Negri, Nicola Bertoldi, and Marcello Federico. 2017. Neural vs. phrase-based machine translation in a multi-domain scenario. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 280–284.

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. *arXiv preprint arXiv:1612.06897*.

Jianfeng Gao, Joshua Goodman, Mingjing Li, and Kai-Fu Lee. 2002. Toward a unified approach to statistical language modeling for chinese. *ACM Transactions on Asian Language Information Processing (TALIP)*, 1(1):3–33.

Guillem Gascó, Martha-Alicia Rocha, Germán Sanchis-Trilles, Jesús Andrés-Ferrer, and Francisco Casacuberta. 2012. Does more data always yield better translations? In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 152–161.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Demo and Poster Sessions*, pages 177–180.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*.

William D. Lewis and Sauleh Eetemadi. 2013. Dramatically reducing training data size through vocabulary saturation. In *Proceedings of the 8th Workshop on Statistical Machine Translation*, pages 281–291.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the 12th International Workshop on Spoken Language Translation*, pages 76–79.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Minh-Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19.

Shachar Mirkin and Laurent Besacier. 2014. Data selection for compact adapted SMT models. In *Proceedings of the 11th Conference of the Association for Machine Translation in the Americas*, pages 301–314.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with bayesian optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

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

Jörg Tiedemann. 2009. News from OPUS-a collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*, volume 5, pages 237–248.

Antonio Toral and Víctor M. Sánchez-Cartagena. 2017. A multifaceted evaluation of neural versus phrase-based machine translation for 9 language directions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1063–1073.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2016. Measuring the effect of conversational aspects on machine translation quality. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*, pages 2571–2581.

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

Keiji Yasuda, Ruiqiang Zhang, Hirofumi Yamamoto, and Eiichiro Sumit. 2008. Method of selecting training data to build a compact and efficient translation model. In *International Joint Conference on Natural Language Processing (IJCNLP)*, pages 655–660.

Dakun Zhang, Jungi Kim, Joseph Crego, and Jean Senellart. 2016. Boosting neural machine translation. *arXiv preprint arXiv:1612.06138*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. *arXiv preprint arXiv:1604.02201*.

Neural Machine Translation
Leveraging Phrase-based Models in a Hybrid Search

Leonard Dahlmann and **Evgeny Matusov** and **Pavel Petrushkov** and **Shahram Khadivi**
eBay Inc.
Kasernenstr. 25
52064 Aachen, Germany
{fdahlmann, ematusov, ppetrushkov, skhadivi}@ebay.com

Abstract

In this paper, we introduce a hybrid search for attention-based neural machine translation (NMT). A target phrase learned with statistical MT models extends a hypothesis in the NMT beam search when the attention of the NMT model focuses on the source words translated by this phrase. Phrases added in this way are scored with the NMT model, but also with SMT features including phrase-level translation probabilities and a target language model. Experimental results on German→English news domain and English→Russian e-commerce domain translation tasks show that using phrase-based models in NMT search improves MT quality by up to 2.3% BLEU absolute as compared to a strong NMT baseline.

1 Introduction

Neural machine translation has become state-of-the-art in recent years, reaching higher translation quality than statistical phrase-based machine translation (PBMT) on many tasks. Human analysis (Bentivogli et al., 2016) showed that NMT makes significantly fewer reordering errors, and also is able to select correct word forms more often than PBMT in the case of morphologically rich target languages. Overall, the fluency of the MT output improves when NMT is used, and the number of lexical choice errors is also reduced. However, state-of-the-art NMT approaches based on an encoder-decoder architecture with an attention mechanism as introduced by (Bahdanau et al., 2014) exhibit weaknesses that sometimes lead to MT errors which a phrase-based MT system does not make. In particular, PBMT usually can better translate rare words (e.g. singletons), as well as memorize and use phrasal translations. NMT has problems translating rare words because of limitations on the vocabulary size, as well as the fact that word embeddings are used to represent both source and target words. A rare word's embedding can not be trained reliably.

Another handicap of NMT is a general difficulty of fixing errors made by a neural MT system. Since NMT does not explicitly use or save word-to-word or phrase-to-phrase mappings, and its search is a target word beam search with almost no constraints, it is difficult to fix errors by an NMT system. It is important to quickly fix certain errors in real-life applications of MT systems to avoid negative user feedback or other (e.g. legal) consequences. An error identified in the output of a PBMT system can be fixed by tracing which phrase pair was used that resulted in the error, and down-weighting or even removing the phrase pair. Also, in PBMT it is easy to add an "override" translation.

In this work, we combine the strengths of NMT and PBMT approaches by introducing a novel hybrid search algorithm. In this algorithm, the standard NMT beam search is extended with phrase translation hypotheses from a statistical phrase table. The decision on when to use what phrasal translations is taken based on the attention mechanism of the NMT model, which provides a soft coverage of the source sentence words. All partial phrasal translations are scored with the NMT decoder and can be continued with a word-based NMT translation candidate or another phrasal translation candidate.

The proposed search algorithm uses a log-linear model in which the NMT translation score is combined with standard phrase translation scores, including a target n-gram language model (LM) score. Thus, a LM trained on additional monolingual data can be used. The decisions on the word

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1411–1420
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

order in the produced target translation are taken based only on the states of the NMT decoder.

This paper is structured as follows. We review related work in Section 1.1. The baseline NMT model we use is described in Section 2, where we also recap the log-linear model combination used in PBMT. Section 3 presents the details of the proposed hybrid search. Experimental results are presented in Section 4, followed by conclusions and outlook in Section 5.

1.1 Related Work

In the line of research closely related to our approach, neural models are used as additional features in vanilla phrase-based systems. Examples include the work of (Devlin et al., 2014), (Junczys-Dowmunt et al., 2016), etc. Such approaches have certain limitations: first, the search space of the model is still restricted by what can be produced using a phrase table extracted from parallel data based on word alignments. Second, the organization of the search, in which only a limited target word history (e.g. 4 last target words) is available for each partial hypothesis, makes it difficult to integrate recurrent neural network LMs and translation models which take all previously generated target words into account. That is why, for instance, the attention-based NMT models were usually applied only in rescoring (Peter et al., 2016).

In (Stahlberg et al., 2017), a two-step translation process is used, where in the first step a SMT translation lattice is generated, and in the second step the NMT decoder combines NMT scores with the Bayes-risk of the translations according to the lattice. In contrast, we explicitly use phrasal translations and language model scores in an integrated search.

In (Arthur et al., 2016), a statistical word lexicon is used to influence NMT hypotheses, also based on the attention mechanism. (Gülçehre et al., 2015) combine target n-gram LM scores with NMT scores to find the best translation. (He et al., 2016) also use a target LM, but add further SMT features such as word penalty and word lexica to the NMT beam search. To the best of our knowledge, no previous work extends the beam search with phrasal translation hypotheses of PBMT, like we propose in this paper.

In (Tang et al., 2016), the NMT decoder is modified to switch between using externally defined phrases and standard NMT word hypotheses. However, only one target phrase per source phrase is considered, and the reported improvements are significant only when manually selected phrase pairs (mostly for rare named entities) are used.

Somewhat related to our work is the concept of coverage-based NMT (Tu et al., 2016), where the model architecture is changed to explicitly account for source coverage. In our work, we use a standard NMT architecture, but track coverage with accumulated attention weights.

2 Background

2.1 Neural MT

Neural MT proposed by (Bahdanau et al., 2014) maximizes the conditional log-likelihood of the target sentence $E : e_1, \ldots, e_I$ given the source sentence $F : f_1, \ldots, f_J$:

$$H_D = -\frac{1}{N}\sum_{n=1}^{N} \log p_\theta(E_n|F_n)$$

where (E_n, F_n) refers to the n-th training sentence pair in a dataset D, and N denotes the total number of sentence pairs in the training corpus. When using the encoder-decoder architecture by (Cho et al., 2014), the conditional probability can be written as:

$$p(e_1 \cdots e_I|f_1 \cdots f_J) = \prod_{i=1}^{I} p(e_i|e_{i-1} \cdots e_1, c)$$

with $p(e_i|e_{i-1} \cdots e_1, c) = g(s_i, e_{i-1}, c)$, where I is the length of the target sentence and J is the length of source sentence, c is a fixed-length vector to encode the source sentence, s_i is a hidden state of RNN at time step i, and $g(\cdot)$ is a non-linear function to approximate the word probability. When the attention mechanism is used, the vector c in each sentence is replaced by a time-variant representation c_i that is a weighted summary over a sequence of annotations $(h_1, \ldots, h_J)$, and h_j contains information about the whole input sentence, but with a strong focus on the parts surrounding the j-th word (Bahdanau et al., 2014). Then, the context vector can be defined as:

$$c_i = \sum_{j}^{J} \alpha_{ij} h_j \quad \text{where} \quad \alpha_{ij} = \frac{exp(r_{ij})}{\sum_{j=1}^{J} exp(r_{ij})}.$$

Therefore, α_{ij} is normalized over all source positions j. Also, $r_{ij} = a(s_{i-1}, h_j)$ is the attention model used to calculate the log-likelihood of aligning the i-th target word to the j-th source word.

2.2 Phrase-based MT

The log-linear model, as introduced in (Och and Ney, 2002), allows decomposing the translation probability $Pr(e_1^I|f_1^J)$ by using an arbitrary number of features $h_m(f_1^J, e_1^I)$. Each feature is multiplied by a corresponding scaling factor λ_m:

$$Pr(e_1^I|f_1^J) = \frac{\exp\left(\sum_{m=1}^M \lambda_m h_m(f_1^J, e_1^I)\right)}{\sum_{\tilde{e}_1^{\tilde{I}}} \exp\left(\sum_{m=1}^M \lambda_m h_m(f_1^J, \tilde{e}_1^{\tilde{I}})\right)}.$$

The standard PBMT approach uses a log-linear model in which bidirectional phrasal and lexical scores, language model scores, distortion scores, word penalties and phrase penalties are combined as features.

3 Hybrid Approach

In this section we describe our proposed hybrid NMT approach. The algorithm allows translations to be generated partially by phrases[1] and partially by words. Section 3.1 describes the models we use to score hypotheses. The search algorithm is presented in Section 3.2.

3.1 Log-linear Combination

We use a log-linear model combination to introduce SMT models into the NMT search. Since translations can be partially generated by phrases, we introduce the phrase segmentation s_1^K as a hidden variable into the models similarly to (Zens and Ney, 2008), where K is the number of phrases used in the translation. Note that, unlike standard PBMT, s_1^K does not need to cover the whole source sentence, as parts of the translation can be generated by words. Using the maximum approximation, the search criterion then is

$$\hat{e}_1^{\hat{I}} = \arg\max_{I, e_1^I} \left\{ \max_{s_1^K} \sum_{m=1}^M \lambda_m h_m(f_1^J, e_1^I, s_1^K) \right\}. \quad (1)$$

Let $\tilde{f}_k, \tilde{e}_k$ be the chosen phrase pairs in the segmentation s_1^K for $k = 1, \ldots, K$. In our experiments with the proposed hybrid search, we use the following features:

1. The NMT feature h_{NMT}.
2. The word penalty feature h_{WP} counts the number of target words. This feature can help control the length of translations.
3. The source word coverage feature h_{SWC} counts the number of source words translated by phrases:
$$h_{\text{SWC}}(f_1^J, e_1^I, s_1^K) = \sum_{k=1}^K |\tilde{f}_k|.$$
The purpose of this feature is to control the usage of phrases.
4. The phrase penalty feature h_{PP} counts the number of phrases used. Together with the word penalty and the source word coverage feature, the phrase penalty can control the length of chosen phrases.
5. The n-gram language model feature h_{LM}.
6. The bidirectional phrase features h_{Phr} and h_{iPhr}. Note that these features are only applied for those parts of the translation that are generated by phrases. The other parts get a phrase score of zero.

The scaling factors λ_m are tuned with minimum error rate training (MERT) (Och, 2003) on n-best lists of the development set.

3.2 Search

The algorithm is based on the beam search for NMT, which generates translations one word per time step in a left-to-right fashion. We modify this search to allow hypothesizing phrases in addition to normal word hypotheses. The phrases are suggested based on the neural attention, starting from the source position with the maximal current attention. We only suggest phrases if a source position is focused. We check that suggested phrases do not overlap with already translated source words by keeping track of the sum of attention in previous time steps for each source position. Thus, the problem of global reordering is left entirely to the NMT model and we follow the attention when hypothesizing phrases.

Hypotheses are scored by NMT and SMT models. The beam is divided into two parts of fixed size: the word beam and the phrase beam. The phrase beam is used to score target phrases which were hypothesized from an entry in a previous word beam. In order to score a target phrase consisting of k words with the NMT model, we use k time steps, allowing us to keep the efficiency of batched NMT scoring. Once a target phrase has been fully scored (and if the hypothesis has

[1] As in SMT, phrases can consist of only a single token.

not been pruned), the hypothesis is returned to the word beam. Both beams are generated and pruned independently in each time step.

The algorithm has some hyper-parameters that need to be set manually. First, we have the beam size N_p for phrase hypotheses and the beam size N_w for word hypotheses. Second, τ_{focus} is the minimum attention that needs to be on a source position to consider it for extending with a phrase translation candidate whose source phrase starts on that position. Third, τ_{cov} is the minimum sum of attention of a source position over previous time steps at which it is considered to be covered. We do not hypothesize phrases that overlap with covered positions.

In the following, we describe the search in detail. Let f_1^J be the source sentence. Before search, we run the standard phrase matching algorithm on the source sentence to retrieve the translation options $E(j, j')$ for source positions $1 \leq j < j' \leq J$ from a given SMT phrase table. With each hypothesis h, we associate the following items:

- $C(h, j)$ is the sum of the NMT attention to source position j involved in generating the target words of h. This can be considered as a soft coverage vector for h.
- $Q(h)$ is the partial log-linear score of h according to Equation 1.
- $E(h)$ is the n-gram target word history of h.
- If h is a phrase hypothesis with target phrase $\tilde{e}$, of which k words already have been scored by NMT, then $P(h) := (\tilde{e}, k)$ is the phrase state.

Also, each hypothesis is associated with its corresponding NMT hidden state. We initialize the beam to consist of an empty word hypothesis. Each step of the beam search proceeds as follows:

1. Let $B = [B_w, B_p]$ be the previous beam with word/phrase hypotheses, respectively. First, we generate the attention vector $\alpha_{h,j}$ and the distribution over target words $\hat{p}_h(e)$ for each hypothesis $h \in B$ and word e in the NMT target vocabulary V_T using the NMT model in batched scoring [2].
2. Initialize new beam $[B'_w, B'_p] = [\emptyset, \emptyset]$.
3. Generate new word hypotheses: find the maximal N_w pairs (h, e) with $h \in B_w$ and $e \in V_T$ according to the score $Q(h) + \lambda_{\text{NMT}} \cdot \log \hat{p}_h(e)$. For the top pairs $h' = (h, e)$, set
$$Q(h') = Q(h) + \lambda_{NMT} \cdot \log \hat{p}_h(e) \\ + \lambda_{LM} \cdot \log p_{\text{LM}}(e|E(h)) + \lambda_{WP}$$
and insert h' into B'_w. Update the soft coverage $C(h', j) = C(h, j) + \alpha_{h,j}$ for $1 \leq j \leq J$.
4. Generate new phrase hypotheses: for each previous word hypothesis $h \in B_w$, convert the soft attention $C(h, \cdot)$ into a binary coverage set C, such that $j \in C$ iff. $C(h, j) > \tau_{\text{cov}}$. Identify the current NMT focus as
$$\hat{j} = \underset{1 \leq j \leq J,\ \alpha_{h,j} > \tau_{\text{focus}}}{\arg\max} \alpha_{h,j}.$$
If there is no such j with $\alpha_{h,j} > \tau_{\text{focus}}$, no phrase hypotheses are generated from h in this step. Otherwise, for each source phrase length l with $C \cap \{\hat{j}, \hat{j}+1, \ldots, \hat{j}+l-1\} = \emptyset$ and each target phrase $\tilde{e} \in E(\hat{j}, \hat{j}+l)$, create a new hypothesis $h' = (h, \tilde{e}_1)$ with the score
$$\begin{aligned} Q(h') = {} & Q(h) + \lambda_{\text{NMT}} \cdot \log \hat{p}_h(e_1) \\ & + \lambda_{\text{LM}} \cdot \log p_{\text{LM}}(\tilde{e}|E(h)) + |\tilde{e}| \cdot \lambda_{\text{WP}} \\ & + \lambda_{\text{PP}} + l \cdot \lambda_{\text{SWC}}. \end{aligned} \tag{2}$$
Note that, in this step, the full target phrase is scored using the language model, while only the first target word is scored using NMT. Initialize the phrase state of h': $P(h') = (\tilde{e}, 1)$. As in step 3, update the soft coverage. If $|\tilde{e}| = 1$, insert h' into B'_w, otherwise insert into B'_p.
5. Advance previous phrase hypotheses: for each $h \in B_p$, with phrase state $P(h) = (\tilde{e}, k)$, score the $(k+1)$-th target word of $\tilde{e}$ using NMT, setting $h' = (h, \tilde{e}_{k+1})$ and
$$Q(h') = Q(h) + \lambda_{\text{NMT}} \cdot \log \hat{p}_h(\tilde{e}_{k+1}).$$
As in step 3, update the soft coverage. Set the new phrase state as $P(h') = (\tilde{e}, k+1)$. If $k + 1 = |\tilde{e}|$, we are finished scoring the phrase and h' is inserted into B'_w. Otherwise, h' is inserted in B'_p.
6. Prune B'_w to N_w entries and B'_p to N_p entries according to $Q(\cdot)$.
7. Insert all hypotheses from the pruned B'_w and B'_p where the last word is the sentence end token into the set of finished hypotheses B_f.
8. $B := [B'_w, B'_p]$.

[2] If a target word e is not in V_T, set $\hat{p}_h(e) = \hat{p}_h(\text{UNK})$ where UNK is a special token denoting unknowns. Note that this almost never happens when using a word segmentation like BPE (Sennrich et al., 2016b).

Data set		WMT		E-commerce	
Language		German	English	English	Russian
Training	Sentences	5,597,491		2,919,406	
	Running words	129,083,315	134,469,297	46,715,319	45,305,268
	Full vocabulary	1,961,186	884,075	326,015	774,435
Dev	Sentences	2169 (WMT 15)		950	
	Running words	56,593	51,324	24,487	24,087
Test	Sentences	6002 (WMT 14 + 16)		1051 (item/product descriptions)	
	Running words	160,469	144,387	29,165	26,476

Table 1: Corpus statistics for the WMT German→English and e-commerce English→Russian MT tasks.

If phrase scores from a phrase table are to be included in the search, Equation 2 needs to be modified by adding $\lambda_{\text{Phr}} \log p(\tilde{f}|\tilde{e})$ and $\lambda_{\text{iPhr}} \log p(\tilde{e}|\tilde{f})$.

As in the pure NMT beam search, this procedure is repeated until either the last word of all hypotheses in a step is the sentence end token, or $2 \cdot J$ many beam steps have been performed. Finally, the best translation is chosen as the one in B_f with the highest score.

Note that the same target sequence can be generated with different phrasal segmentations. During search, if two hypotheses have the same full target history in a beam, we recombine them and discard the hypothesis with the lower score.

4 Experiments

We perform experiments comparing the translation quality of our hybrid approach to phrase-based and pure end-to-end NMT baselines. We present results on two tasks: an in-house English→Russian e-commerce task (translation of real product/item descriptions from an e-commerce site), and the WMT 2016 German→English task (news domain). The corpus statistics are shown in Table 1.

For the English→Russian task, the parallel training data consists of an in-domain part (ca. 5.5M running words) of product/item titles and descriptions and other e-commerce content. The rest is out-of-domain data (UN, subtitles, TAUS data collections, etc.) sampled to have significant n-gram overlap with the in-domain description data. Item descriptions are provided by private sellers and, like any user-generated content, may contain ungrammatical sentences, spelling errors, and other noise. Product descriptions usually originate from product catalogs and are more "clean", but on the other hand, are difficult to translate because of rare domain-specific terminology. Both types of text contain itemizations, measurement units, and other structures which are usually not found in normal sentences. We tune the system on a development set that is a mix of product and item descriptions, and evaluate on separate product/item description test sets. For development and test sets, two reference translations are used.

The German→English system is trained on parallel corpora provided for the constrained WMT 2017 evaluation (Europarl, Common Crawl, and others). We use the WMT 2015 evaluation data as development set, and the evaluation is performed on two sets from the WMT evaluations in 2014 and 2016. Only a single human reference translation is provided.

For the phrase-based baselines, we use an in-house phrase-decoder (Matusov and Köprü, 2010) which is similar to the Moses decoder (Koehn et al., 2007). We use standard SMT features, including word-level and phrase-level translation probabilities, the distortion model, 5-gram LMs, and a 7-gram joint translation and reordering model reimplemented based on the work of (Guta et al., 2015). The language model for the e-commerce task is trained on additional monolingual Russian item description data containing 28.2M words. For the WMT task, we use the English News Crawl data containing 3.8B words for additional language model data. The tuning is performed using MERT (Och, 2003) to increase the BLEU score on the development set. To stabilize the optimization on the English→Russian task, we detach Russian morphological suffixes from the word stems both in hypotheses and references using a context-independent "poor man's" morphological analysis. We prefix each suffix with a special symbol and treat them as separate tokens.

We have implemented our NMT model in

		Item descriptions		Product descriptions	
System description	Beam size	BLEU [%]	TER [%]	BLEU [%]	TER [%]
Phrase-based	-	21.3	61.6	22.7	56.6
+ 1000-best rescoring with NMT	-	23.1	60.1	25.8	54.7
NMT	12	26.4	56.4	28.4	52.0
NMT	128	26.3	56.6	28.5	51.9
Full hybrid approach	128	26.7	56.1	29.9	51.2
+ extra LM data	128	27.4	55.4	30.8	50.5
NMT + WP + LM (with extra data)	128	26.2	57.3	29.0	51.8

Table 2: Overview of translation results on the e-commerce English→Russian task.

Python using the TensorFlow[3] deep learning library. We use the embedding size of 620, RNN size of 1000 and GRU cells. The model is trained with maximum likelihood loss for 15 epochs using Adam optimizer (Kingma and Ba, 2014) on complete data in batches of 100 sentences. The learning rate is initialized to 0.0002, decaying by 0.9 each epoch. For regularization we use L2 loss with weight 10^{-7} and dropout following Gal and Ghahramani (2016). We set the dropout probability for input and recurrent connections of the RNN to 0.2 and word embedding dropout probability to 0.1. On the English→Russian task, the model is then fine-tuned on in-domain data for 10 epochs. The vocabulary is limited using byte pair encoding (BPE) (Sennrich et al., 2016b) with 40K splits separately for each language. To speed up training we use approximate loss as described in (Jean et al., 2015). For pure NMT experiments, we employ length normalization (Wu et al., 2016), as otherwise short translations would be favored.

For the hybrid approach, we use the same trained end-to-end model as in the NMT baseline. We use all the phrase-based model features plus the NMT score and run MERT as described in Section 3.1. Language models are trained on the level of BPE tokens. We consider at most 100 translation options for each source phrase. If not specified otherwise, we use a beam size of 96 for phrase hypotheses and a beam size of 32 for word hypotheses, resulting in a combined beam size of 128. Furthermore, we set the focus threshold $\tau_{\text{focus}} = 0.3$ and the coverage threshold $\tau_{\text{cov}} = 0.7$ by default. We also perform experiments where these hyper-parameters are varied.

4.1 E-commerce English→Russian

The results on the e-commerce English→Russian task are summarized in Table 2.

NMT vs. phrase-based SMT

The pure NMT system exhibits large improvements over the phrase-based baseline[4]. These improvements are also significantly larger than when we use the NMT model to rescore PBMT 1000-best lists. NMT results are not improved when the beam size is increased from 12 to 128.

Hybrid search vs. pure NMT search

For the hybrid approach, we train a phrase-table on the in-domain data and split the source and target phrases with BPE afterwards for compatibility with the NMT vocabulary. With the hybrid approach, when using a LM trained only on the target side of bilingual data, we get an improvement of 0.3% BLEU on item descriptions and 1.4% BLEU on product descriptions over the pure NMT system. When we use the LM trained on extra monolingual data, we get total improvements of 1.0% BLEU and 2.3% BLEU with the hybrid approach. In contrast, when we add this language model and a word penalty on top of the pure NMT system and tune scaling factors with MERT, we get small improvements (last row of Table 2) only on product descriptions. This shows that the hybrid approach can exploit the LM better than a purely word-based NMT approach. We have also performed experiments utilizing the additional monolingual data for synthetic training data for NMT as in (Sennrich et al., 2016a), but did not get improvements.

To analyze the improvements of the hybrid system, we perform experiments in which we either

[3] `http://tensorflow.org`

[4] The significance of these improvements was also confirmed by an in-house human evaluation with 3 judges.

	Item descriptions		Product descriptions	
System description	BLEU [%]	TER [%]	BLEU [%]	TER [%]
Full hybrid approach	27.4	55.4	30.8	50.5
Without LM	26.5	55.9	29.2	51.0
Without source word coverage feature	26.7	56.1	29.4	51.2
Without phrase scores	27.2	55.9	30.6	50.6
Maximal source phrase length 1	26.7	56.4	29.1	51.6
Minimal source phrase length 2	27.0	55.9	30.0	51.1

Table 3: Translation results of the hybrid approach on the e-commerce English→Russian task with different SMT model combinations. The first row shows results with all models enabled. In the following rows, we either remove or limit exactly one model compared to the full system.

disable or limit some of the SMT models. The results are shown in Table 3. Without the language model, the hybrid approach has almost no improvements over the NMT baseline. This indicates that the language model is crucial in selecting appropriate phrase candidates. Similarly, when we disable the source word coverage feature, the translation quality is degraded, suggesting that this feature helps choose between phrase hypotheses and word hypotheses during the search. Next, we do not use phrase-level scores. Here, we observe only a small degradation of translation quality. Finally, we limit the source length of phrases used in the search, allowing only one-word source phrases in one experiment and only source phrases with two or more words in another experiment. In both cases, the translation quality decreases. Thus, both one-word phrases and longer phrases are necessary to obtain the best results.

Tuning the beam size

Next, we study the effect of different beam sizes on translation quality. The results are shown in Table 4. Note that we retune the system for each choice. With a total beam size of 128, we get the best results by using a phrase beam size of 96 and a word beam size of 32. When we use a phrase beam size of 116 or 64 instead, the translation quality worsens. In another experiment, we decrease the total beam size to 64. The translation quality degrades only slightly, which means that we can still expect MT quality improvements with hybrid search even if we optimize the system for speed. To further test this, we reduce the beam sizes to $N_w = 12$ and $N_p = 4$ after tuning with $N_w = 32$ and $N_p = 96$. We get BLEU scores of 27.1% on item descriptions and 30.1% on product descriptions, losing 0.3% and 0.7% BLEU respectively compared to the full beam size.

Beam size		Item descr.		Product descr.	
N_p	N_w	BLEU [%]	TER [%]	BLEU [%]	TER [%]
116	12	26.7	55.9	29.8	51.1
96	32	27.4	55.4	30.8	50.5
64	64	26.8	55.6	30.1	50.7
32	32	27.1	55.8	30.7	50.5

Table 4: Effect of the beam size (word beam size N_w + phrase beam size N_p) for the hybrid approach on the e-commerce English→Russian task.

Tuning the attention focus/coverage thresholds

Table 5 shows results with different values for the coverage threshold τ_{cov}. Again, we retune the system for each choice. Setting the coverage threshold to 1.0 or even disabling the coverage check (by setting $\tau_{\text{cov}} = \infty$) has little effect on the translation scores on this task. This can be explained by the fact that translation from English to Russian is mostly monotonic. We also tried varying the focus threshold τ_{focus} between 0.0 and 0.3 but did not notice any significant effect on this task.

		Item descr.		Product descr.	
τ_{focus}	τ_{cov}	BLEU [%]	TER [%]	BLEU [%]	TER [%]
0.3	0.7	27.4	55.4	30.8	50.5
0.3	1.0	27.2	55.4	30.3	50.3
0.3	∞	27.5	55.4	30.4	50.9

Table 5: Effect of the threshold parameters on the hybrid approach on the e-commerce English→Russian task.

Analysis

To understand the behavior of the hybrid search, we count the number of source words that are

		newstest2014		newstest2016	
System description	Beam size	BLEU [%]	TER [%]	BLEU [%]	TER [%]
Phrase-based	-	22.9	59.4	26.9	54.1
+ News Crawl LM data	-	25.4	59.0	29.2	53.8
NMT	12	26.9	53.0	32.3	47.6
NMT	64	27.0	53.0	32.2	47.6
Hybrid approach	64	27.8	53.2	32.4	48.2
+ tuning τ_{focus}, τ_{cov}	64	28.0	53.0	33.3	47.4
+ News Crawl LM data	64	29.7	52.2	35.3	46.7

Table 6: Overview of translation results on the WMT German→English task.

translated by phrases in the product descriptions test set. Of the 9320 source words, 7109 (76.3%) are covered by phrase hypotheses. 78.3% of the source phrases are unigrams, 19.5% are bigrams and 2.2% are trigrams or longer. Among the many one-word phrases used, almost all (99.2%) are also within the top 3 predictions of word-based NMT, and 90.3% are equal to the top NMT prediction.

Further human analysis by a native Russian speaker of the pure NMT vs. hybrid search translations shows that hybrid search is often able to correct the following known NMT handicaps:

- incorrect translation of rare words (among other reasons, due to incorrect sub-word unit translation in which rare words are aggressively segmented).
- repetition of same or similar words as a result of multiple attention to the same source word, as well as untranslated words that received no attention.
- incorrect or partially correct word-by-word translation when a phrasal (non-literal) translation should be used instead.

In all of these cases, the usage of phrasal translations is able to better enforce the coverage, and this, in turn, leads to improved lexical choice. The fact that not many long phrase pairs are selected indicates, in our opinion, that the search and modeling problem in NMT is far from being solved: with the right, diverse model scores, the proposed hybrid search is able to select and extend better hypotheses with words, most of which already had a high NMT probability. Yet they are not always selected in the pure NMT beam search, among other reasons, due to competition from words erroneously placed near them in the embedding space.

4.2 WMT 2016 German→English

The results on the WMT German→English task are shown in Table 6. The initial phrase-based baseline uses the 5-gram language model estimated on the target side of bilingual data. By adding the News Crawl LM data, we gain 2.5% and 2.3% BLEU on the test sets, but PBMT still is behind NMT.

For the hybrid approach, we use a beam size of 64 and a maximal number of beam steps of $1.5 \cdot J$ (instead of $2 \cdot J$) to speed up experiments. We use separate word penalty features, one for word-based hypotheses and one for phrase-based hypotheses to allow for more control of translation lengths. With the hybrid approach, using the 5-gram language model estimated on the target side of bilingual data, and phrase scores, we get small improvements in BLEU over the NMT baseline. However, the TER increases. We experiment with different thresholds, setting $\tau_{\text{focus}} = 0.1$ and $\tau_{\text{cov}} = 1.0$. With this hybrid system, we get improvements of 1.0% and 1.1% BLEU over pure NMT. Finally, we add the News Crawl LM data on top. This significantly improves the results by 1.7% and 2.0% BLEU. In total, we gain 2.7% and 3.1% BLEU over pure NMT. These results reinforce the fact that, similar to PBMT, language model quality is important for the proposed hybrid search. In contrast, we have also tried applying only the LM (including News Crawl data) with a word penalty on top of NMT, but did not get consistent improvements.

Figure 1 shows an example for the phrase pairs chosen by the hybrid system on top of the NMT attention. The hybrid approach correctly translates the German idiom “nach und nach” as “gradually”, while the pure NMT system incorrectly translates it word-by-word as “after and after”.

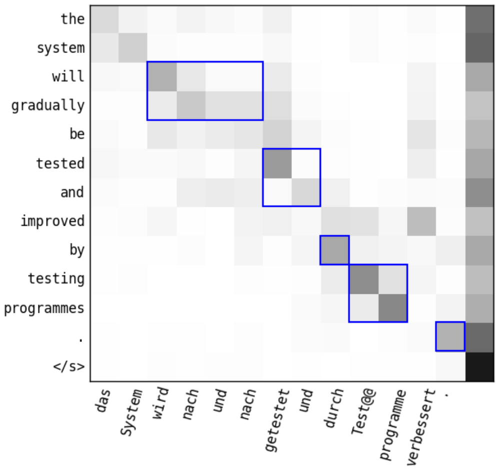


Figure 1: Example alignment from the hybrid search, with the source sentence on the bottom and the translation on the left. The blue rectangles signify phrase pairs on top of the NMT attention. The pure NMT translation is "the system is tested after and after testing and improved by testing programs."

5 Conclusion

In this work, we proposed a novel hybrid search that extends NMT with phrase-based models. The NMT beam search was modified to insert phrasal translations based on the current and accumulated attention weights of the NMT decoder RNN. The NMT model score was used in a log-linear model with standard phrase-based scores as well as an n-gram language model. We described the algorithm in detail, in which we keep separate beams for NMT word hypotheses and hypotheses with an incomplete phrasal translation, as well as introduce parameters which control the source sentence coverage. Numerous experiments on two large vocabulary translation tasks showed that the hybrid search improves BLEU scores significantly as compared to a strong NMT baseline that already outperforms phrase-based SMT by a large margin.

In the future, we plan to focus on integration of phrasal components into NMT training, including better coverage constraints, as well as methods for context-dependent translation override within our hybrid search algorithm.

Acknowledgments

We would like to thank Tamer Alkhouli and Jan-Thorsten Peter for helpful discussions. We thank the anonymous reviewers for their suggestions.

References

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 1557–1567.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 257–267.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pages 1724–1734.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M. Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL*. pages 1370–1380.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29*. pages 1019–1027.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *CoRR* abs/1503.03535.

Andreas Guta, Tamer Alkhouli, Jan-Thorsten Peter, Joern Wuebker, and Hermann Ney. 2015. A comparison between count and neural network models based on joint translation and reordering sequences. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 1401–1411.

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. 2016. Improved neural machine translation with SMT features. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. pages 151–157.

Sébastien Jean, KyungHyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the*

7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing. pages 1–10.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Rico Sennrich. 2016. The AMU-UEDIN submission to the WMT16 news translation task: Attention-based NMT models as feature functions in phrase-based SMT. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL*. pages 319–325.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics, pages 177–180.

Evgeny Matusov and Selçuk Köprü. 2010. AppTek's APT Machine Translation System for IWSLT 2010. In Marcello Federico, Ian Lane, Michael Paul, and François Yvon, editors, *International Workshop on Spoken Language Translation, IWSLT*. pages 29–36.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*. pages 160–167.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. pages 295–302.

Jan-Thorsten Peter, Andreas Guta, Nick Rossenbach, Miguel Graa, and Hermann Ney. 2016. The rwth aachen machine translation system for iwslt 2016. In *International Workshop on Spoken Language Translation, IWSLT*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL*.

Felix Stahlberg, Adrià de Gispert, Eva Hasler, and Bill Byrne. 2017. Neural machine translation by minimising the Bayes-risk with respect to syntactic translation lattices. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Valencia, Spain.

Yaohua Tang, Fandong Meng, Zhengdong Lu, Hang Li, and Philip L. H. Yu. 2016. Neural machine translation with external phrase memory. *CoRR* abs/1606.01792.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144.

Richard Zens and Hermann Ney. 2008. Improvements in dynamic programming beam search for phrase-based statistical machine translation. In *International Workshop on Spoken Language Translation, IWSLT*. pages 198–205.

Translating Phrases in Neural Machine Translation

Xing Wang[†] **Zhaopeng Tu**[‡] **Deyi Xiong**[†*] **Min Zhang**[†]
[†]Soochow University, Suzhou, China
xingwsuda@gmail.com, {dyxiong, minzhang}@suda.edu.cn
[‡] Tencent AI Lab, Shenzhen, China
tuzhaopeng@gmail.com

Abstract

Phrases play an important role in natural language understanding and machine translation (Sag et al., 2002; Villavicencio et al., 2005). However, it is difficult to integrate them into current neural machine translation (NMT) which reads and generates sentences word by word. In this work, we propose a method to translate phrases in NMT by integrating a phrase memory storing target phrases from a phrase-based statistical machine translation (SMT) system into the encoder-decoder architecture of NMT. At each decoding step, the phrase memory is first re-written by the SMT model, which dynamically generates relevant target phrases with contextual information provided by the NMT model. Then the proposed model reads the phrase memory to make probability estimations for all phrases in the phrase memory. If phrase generation is carried on, the NMT decoder selects an appropriate phrase from the memory to perform phrase translation and updates its decoding state by consuming the words in the selected phrase. Otherwise, the NMT decoder generates a word from the vocabulary as the general NMT decoder does. Experiment results on the Chinese→English translation show that the proposed model achieves significant improvements over the baseline on various test sets.

1 Introduction

Neural machine translation (NMT) has been receiving increasing attention due to its impressive translation performance (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015; Wu et al., 2016). Significantly different from conventional statistical machine translation (SMT) (Brown et al., 1993; Koehn et al., 2003; Chiang, 2005), NMT adopts a big neural network to perform the entire translation process in one shot, for which an encoder-decoder architecture is widely used. Specifically, the encoder encodes a source sentence into a continuous vector representation, then the decoder uses the continuous vector representation to generate the corresponding target translation word by word.

The word-by-word generation philosophy in NMT makes it difficult to translate multi-word phrases. Phrases, especially multi-word expressions, are crucial for natural language understanding and machine translation (Sag et al., 2002; Villavicencio et al., 2005) as the meaning of a phrase cannot be always deducible from the meanings of its individual words or parts. Unfortunately current NMT is essentially a word-based or character-based (Chung et al., 2016; Costa-jussà and Fonollosa, 2016; Luong and Manning, 2016) translation system where phrases are not considered as translation units. In contrast, phrases are much better than words as translation units in SMT and have made a significant advance in translation quality. Therefore, a natural question arises: Can we translate phrases in NMT?

Recently, there have been some attempts on multi-word phrase generation in NMT (Stahlberg et al., 2016b; Zhang and Zong, 2016). However these efforts constrain NMT to generate either syntactic phrases or domain phrases in the word-by-word generation framework. To explore the phrase generation in NMT beyond the word-by-word generation framework, we propose a novel architecture that integrates a phrase-based SMT

[*] Corresponding author

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1421–1431
Copenhagen, Denmark, September 7–11, 2017.

model into NMT. Specifically, we add an auxiliary phrase memory to store target phrases in symbolic form. At each decoding step, guided by the decoding information from the NMT decoder, the SMT model dynamically generates relevant target phrase translations and writes them to the memory. Then the NMT decoder scores phrases in the phrase memory and selects a proper phrase or word with the highest probability. If the phrase generation is carried out, the NMT decoder generates a multi-word phrase and updates its decoding state by consuming the words in the selected phrase.

Furthermore, in order to enhance the ability of the NMT decoder to effectively select appropriate target phrases, we modify the encoder of NMT to make it fit for exploring structural information of source sentences. Particularly, we integrate syntactic chunk information into the NMT encoder, to enrich the source-side representation. We validate our proposed model on the Chinese→English translation task. Experiment results show that the proposed model significantly outperforms the conventional attention-based NMT by 1.07 BLEU points on multiple NIST test sets.

The rest of this paper is organized as follows. Section 2 briefly introduces the attention-based NMT as background knowledge. Section 3 presents our proposed model which incorporates the phrase memory into the NMT encoder-decoder architecture, as well as the reading and writing procedures of the phrase memory. Section 4 presents our experiments on the Chinese→English translation task and reports the experiment results. Finally we discuss related work in Section 5 and conclude the paper in Section 6.

2 Background

Neural machine translation often adopts the encoder-decoder architecture with recurrent neural networks (RNN) to model the translation process. The bidirectional RNN encoder which consists of a forward RNN and a backward RNN reads a source sentence $\mathbf{x} = x_1, x_2, ..., x_{T_x}$ and transforms it into word annotations of the entire source sentence $\mathbf{h} = h_1, h_2, ..., h_{T_x}$. The decoder uses the annotations to emit a target sentence $\mathbf{y} = y_1, y_2, ..., y_{T_y}$ in a word-by-word manner.

In the training phase, given a parallel sentence $(\mathbf{x}, \mathbf{y})$, NMT models the conditional probability as follows,

$$P(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{T_y} P(y_i|\mathbf{y_{<i}}, \mathbf{x}) \tag{1}$$

where y_i is the target word emitted by the decoder at step i and $\mathbf{y_{<i}} = y_1, y_2, ..., y_{i-1}$. The conditional probability $P(y_i|\mathbf{y_{<i}}, \mathbf{x})$ is computed as

$$P(y_i|\mathbf{y_{<i}}, \mathbf{x}) = softmax(f(s_i, y_{i-1}, c_i)) \tag{2}$$

where $f(\cdot)$ is a non-linear function and s_i is the hidden state of the decoder at step i:

$$s_i = g(s_{i-1}, y_{i-1}, c_i) \tag{3}$$

where $g(\cdot)$ is a non-linear function. Here we adopt Gated Recurrent Unit (Cho et al., 2014) as the recurrent unit for the encoder and decoder. c_i is the context vector, computed as a weighted sum of the annotations $\mathbf{h}$:

$$c_i = \sum_{j=1}^{T_x} \alpha_{t,j} h_j \tag{4}$$

where h_j is the annotation of source word x_j and its weight $\alpha_{t,j}$ is computed by the attention model.

We train the attention-based NMT model by maximizing the log-likelihood:

$$C(\theta) = \sum_{n=1}^{N} \sum_{i=1}^{T_y} \log P(y_i^n|\mathbf{y}_{<i}^n, \mathbf{x}^n) \tag{5}$$

given the training data with N bilingual sentences (Cho, 2015).

In the testing phase, given a source sentence $\mathbf{x}$, we use beam search strategy to search a target sentence $\hat{\mathbf{y}}$ that approximately maximizes the conditional probability $P(\mathbf{y}|\mathbf{x})$

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \tag{6}$$

3 Approach

In this section, we introduce the proposed model which incorporates a phrase memory into the encoder-decoder architecture of NMT. Inspired by the recent work on attaching an external structure to the encoder-decoder architecture (Gulcehre et al., 2016; Gu et al., 2016; Tang et al., 2016; Wang et al., 2017), we adopt a similar approach to incorporate the phrase memory into NMT.

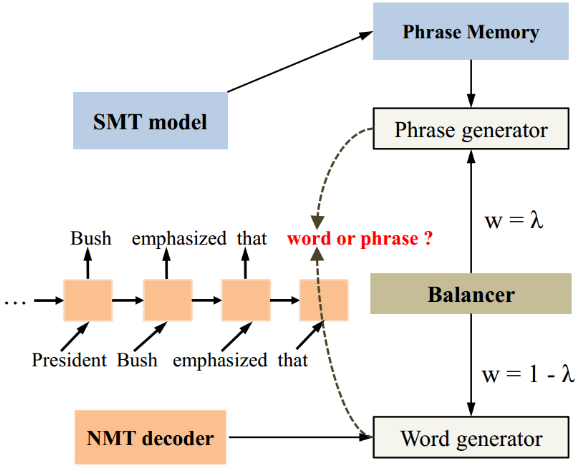


Figure 1: Architecture of the NMT decoder with the phrase memory. The NMT decoder performs phrase generation using the balancer and the phrase memory.

3.1 Framework

Figure 1 shows an example. Given the generated words "*President Bush emphasized that*", the model generates the next fragment either from a word generation mode or a phrase generation mode. If the model selects the word generation mode, it generates a word by the NMT decoder as in the standard NMT framework. Otherwise, it generates a multi-word phrase by enquiring a phrase memory, which is written by an SMT decoder based on the dynamic decoding information from the NMT model for each step. The trade-off between word generation mode and phrase generation mode is balanced by a weight λ, which is produced by a neural network based *balancer*.

Formally, a generated translation $\mathbf{y} = \{y_1, y_2, \ldots, y_{T_y}\}$ consists of two sets of fragments: words generated by NMT decoder $\mathbf{w} = \{w_1, w_2, \ldots, w_K\}$ and phrases generated from the phrase memory $\mathbf{p} = \{p_1, p_2, \ldots, p_L\}$. The probability of generating $\mathbf{y}$ is calculated by

$$P(\mathbf{y}|\mathbf{x}) = \prod_{w_k \in \mathbf{w}} (1 - \lambda_{t(w_k)}) P_{word}(w_k) \times \prod_{p_l \in \mathbf{p}} \lambda_{t(p_l)} P_{phrase}(p_l) \quad (7)$$

where $P_{word}(w_k)$ is the probability of generating the word w_k (see Equation 2), $P_{phrase}(p_l)$ is that of generating the phrase p_l which will be described in Section 3.2, and $t(\cdot)$ is the decoding step to generate the corresponding fragment.

The balancing weight λ is produced by the *balancer* – a multi-layer network. The balancer network takes as input the decoding information, including the context vector c_i, the previous decoding state s_{i-1} and the previous generated word y_{i-1}:

$$\lambda_i = \sigma(f_b(s_i, y_{i-1}, c_i)) \quad (8)$$

where $\sigma(\cdot)$ is a sigmoid function and $f_b(\cdot)$ is the activation function. Intuitively, the weight λ can be treated as the estimated importance of the phrase to be generated. We expect λ to be high if the phrase is appropriate at the current decoding step.

Well-Formed Phrases We employ a source-side chunker to chunk the source sentence, and only phrases that corresponds to a source chunk are used in our model. We restrict ourselves to the well-formed chunk phrases based on the following considerations: (1) In order to take advantage of dynamic programming, we restrict ourselves to non-overlap phrases.[1] (2) We explicitly utilize the boundary information of the source-side chunk phrases, to better guide the proposed model to adopt a target phrase at an appropriate decoding step. (3) We enable the model to exploit the syntactic categories of chunk phrases to enhance the proposed model with its selection preference for special target phrases. With these information, we enrich the context vector c_i to enable the proposed model to make better decisions, as described below.

Following the commonly-used strategy in sequence tagging tasks (Xue and Shen, 2003), we allow the words in a phrase to share the same chunk tag and introduce a special tag for the beginning word. For example, the phrase " 信息 安全 (information security)" is tagged as a noun phrase "NP", and the tag sequence should be "NP_B NP". Partially motivated by the work on integrating linguistic features into NMT (Sennrich and Haddow, 2016), we represent the encoder input as the combination of word embeddings and chunking tag embeddings, instead of word embeddings alone in the conventional NMT. The new input is formulated as follows:

$$[E^w x_i, E^t t_i] \quad (9)$$

[1] Overlapped phrases may result in a high dimensionality in translation hypothesis representation and make it hard to employ shared fragments for efficient dynamic programming.

where $E^w \in \mathbb{R}^{dw \times |V^{NMT}|}$ is a word embedding matrix and dw is the word embedding dimensionality, $E^t \in \mathbb{R}^{dt \times |V^{TAG}|}$ is a tag embedding matrix and dt is the tag embedding dimensionality. $[\cdot]$ is the vector concatenation operation.

3.2 Phrase Memory

The phrase memory stores relevant target phrases provided by an SMT model, which is trained on the same bilingual corpora. At each decoding step, the memory is firstly erased and re-written by the SMT model, the decoding of which is based on the translation information provided by the NMT model. Then, the proposed model enquires phrases along with their probabilities P_{phrase} from the memory.

Writing to Phrase Memory Given a partial translation $\mathbf{y}_{<i} = \{y_1, y_2, \dots, y_{t-1}\}$ generated from NMT, the SMT model picks potential phrases extracted from the translation table. The phrases are scored with multiple SMT features, including the language model score, the translation probabilities, the reordering score, and so on. Specially, the reordering score depends on alignment information between source and target words, which is derived from attention distribution produced by the NMT model (Wang et al., 2017). SMT coverage vector in (Wang et al., 2017) is also introduced to avoid repeat phrasal recommendations. In our work, the potential phrase is phrase with high SMT score which is defined as following:

$$SMT_{score}(p_l|\mathbf{y}_{<\mathbf{t}}, \mathbf{x}) = \sum_{m=1}^{M} w_m h_m(p_l, x(p_l)) \tag{10}$$

where p_l is a target phrase and $x(p_l)$ is its corresponding source span. $h_m(p_l, x(p_l))$ is a SMT feature function and w_m is its weight. The feature weights can be tuned by the minimum error rate training (MERT) algorithm (Och, 2003).

This leads to a better interaction between SMT and NMT models. It should be emphasized that our memory is dynamically updated at each decoding step based on the decoding history from both SMT and NMT models.

The proposed model is very flexible, where the phrase memory can be either fully dynamically generated by an SMT model or directly extracted from a bilingual dictionary, or any other bilingual resources storing idiomatic translations or bilingual multi-word expressions, which may lead to a further improvement. [2]

Reading Phrase Memory When phrases are read from the memory, they are rescored by a neural network based score function. The score function takes as input the phrase itself and decoding information from NMT ($i = t(p_l)$ denotes the current decoding step):

$$score_{phrase}(p_l) = g_s\big(e(p_l), s_i, y_{i-1}, c_i\big) \tag{11}$$

where $g_s(\cdot)$ is either an identity or a non-linear function. $e(p_l)$ is the representation of phrase p_l, which is modeled by a recurrent neural networks. Again, s_i is the decoder state, y_{i-1} is the lastly generated word, and c_i is the context vector. The scores are normalized for all phrases in the phrase memory, and the probability for phrase p_l is calculated as

$$P_{phrase}(p_l) = softmax(score_{phrase}(p_l)) \tag{12}$$

The probability calculation is controlled with parameters, which are trained together with the parameters from the NMT model.

3.3 Training

Formally, we train both the default parameters of standard NMT and the new parameters associated with phrase generation on a set of training examples $\{[\mathbf{x}^n, \mathbf{y}^n]\}_{n=1}^{N}$:

$$C(\theta) = \sum_{n=1}^{N} \log P(\mathbf{y}^n|\mathbf{x}^n) \tag{13}$$

where $P(\mathbf{y}^n|\mathbf{x}^n)$ is defined in Equation 7. Ideally, the trained model is expected to produce a higher balance weight λ and phrase probability P_{phrase} when a phrase is selected from the memory, and lower scores in other cases.

3.4 Decoding

During testing, the NMT decoder generates a target sentence which consists of a mixture of words and phrases. Due to the different granularities of words and phrases, we design a variant of beam search strategy: At decoding step i, we first compute P_{phrase} for all phrases in the phrase memory

[2] Bilingual resources can be utilized in two ways: First, we can store the bilingual resources in a static memory and keep all items available to NMT in the whole decoding period. Second, we can integrate the bilingual resources into SMT and then dynamically feed them into the phrase memory.

and P_{word} for all words in NMT vocabulary. Then the balancer outputs a balancing weight λ_i, which is used to scale the phrase and word probabilities : $\lambda_i \times P_{phrase}$ and $(1 - \lambda_i) \times P_{word}$. Now outputs are normalized probabilities on the concatenation of phrase memory and the general NMT vocabulary. At last, the NMT decoder generates a proper phrase or word of the highest probability.

If a target phrase in the phrase memory has the highest probability, the decoder generates the target phrase to complete the multi-word phrase generation process, and updates its decoding state by consuming the words in the selected phrase as described in Equation 3. All translation hypotheses are placed in the corresponding beams according to the number of generated target words.

4 Experiments

In this section, we evaluated the effectiveness of our model on the Chinese→English machine translation task. The training corpora consisted of about 1.25 million sentence pairs[3] with 27.9 million Chinese words and 34.5 million English words respectively. We used NIST 2006 (NIST06) dataset as development set, and NIST 2004 (NIST04), 2005 (NIST05) and 2008 (NIST08) datasets as test sets. We report experiment results with case-insensitive BLEU score[4].

We compared our proposed model with two state-of-the-art systems:

* **Moses**: a state-of-the-art phrase-based SMT system (Koehn et al., 2007) with its default settings, where feature function weights are tuned by the minimum error rate training (MERT) algorithm (Och, 2003).

* **RNNSearch**: an in-house implementation of the attention-based NMT system (Bahdanau et al., 2015) with its default settings.

For Moses, we used the full bilingual training data to train the phrase-based SMT model and the target portion of the bilingual training data to train a 4-gram language model using KenLM[5]. We ran Giza++ on the training data in both Chinese-to-English and English-to-Chinese directions and applied the "grow-diag-final" refinement rule (Koehn et al., 2003) to obtain word alignments. The maximum phrase length is set to 7.

For RNNSearch, we generally followed settings in the previous work (Bahdanau et al., 2015; Tu et al., 2017a,b). We only kept a shortlist of the most frequent 30,000 words in Chinese and English, covering approximately 97.7% and 99.3% of the data in the two languages respectively. We constrained our source and target sequences to have a maximum length of 50 words in the training data. The size of embedding layer of both sides was set to 620 and the size of hidden layer was set to 1000. We used a minibatch stochastic gradient descent (SGD) algorithm of size 80 together with Adadelta (Zeiler, 2012) to train the NMT models. The decay rates ρ and ϵ were set as 0.95 and 10^{-6}. We clipped the gradient norm to 1.0 (Pascanu et al., 2013). We also adopted the dropout technique. Dropout was applied only on the output layer and the dropout rate was set to 0.5. We used a simple beam search decoder with beam size 10 to find the most likely translation.

For the proposed model, we used a Chinese chunker[6] (Zhu et al., 2015) to chunk the source-side Chinese sentences. 13 chunking tags appeared in our chunked sentences and the size of chunking tag embedding was set to 10. We used the trained phrase-based SMT to translate the source-side chunks. The top 5 translations according to their translation scores (Equation 10) were kept and among them multi-word phrases were used as phrasal recommendations for each source chunk phrase. For a source-side chunk phrase, if there exists phrasal recommendations from SMT, the output chunk tag was used as its chunking tag feature as described in Section 3.1. Otherwise, the words in the chunk were treated as general words by being tagged with the default tag. In the phrase memory, we only keep the top 7 target translations with highest SMT scores at each decoding step. We used a forward neural network with two hidden layers for both the balancer (Equation 8) and the scoring function (Equation 11). The numbers of units in the hidden layers were set to 2000 and 500 respectively. We used a backward RNN encoder to learn the phrase representations of target phrases in the phrase memory.

[3] The corpus includes LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

[4] ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl

[5] https://kheafield.com/code/kenlm/

[6] http://www.niuparser.com/

SYSTEM	NIST04	NIST05	NIST08	Avg
Moses	34.74	31.99	23.69	30.14
RNNSearch	37.80	34.70	24.93	32.48
+memory	38.21	35.15	25.48†	32.95
+memory +chunking tag	38.83‡	35.72‡	26.09‡	33.55

Table 1: Main experiment results on the NIST Chinese-English translation task. BLEU scores in the table are case insensitive. Moses and RNNSearch are SMT and NMT baseline system respectively. "†": significantly better than RNNSearch ($p < 0.05$); "‡": significantly better than RNNSearch ($p < 0.01$).

	NIST04	NIST05	NIST08
+memory	34.3%	29.4%	22.2%
+chunking tag	66.4%	63.1%	58.4%

Table 2: Percentages of sentences that contain phrases generated by the proposed model.

Type	All		New	
	Total	Correct	Total	Correct
NP	81.0%	38.7%	46.0%	11.5%
VP	8.0%	1.7%	6.5%	0.8%
QP	10.8%	4.1%	6.2%	0.9%
Others	0.2%	0%	0.2%	0%
Sum	100%	44.5%	58.9%	13.2%

Table 3: Percentages of phrase categories to the total number of generated ones. "All" denotes all generated phrases, and "New" means new phrases that cannot be found in translations generated by the baseline system. "Total" is the total number of generated phrases and "Correct" denotes the fully correct ones.

4.1 Main Results

Table 1 reports main results of different models measured in terms of BLEU score. We observe that our implementation of RNNSearch outperforms Moses by 2.34 BLEU points. (*+memory*) which is the proposed model with the phrase memory obtains an improvement of 0.47 BLEU points over the baseline RNNSearch. With the source-side chunking tag feature, (*+memory+chunking tag*) outperforms the baseline RNNSearch by 1.07 BLEU points, showing the effectiveness of chunking syntactic categories on the selection of appropriate target phrases. From here on, we use "*+memory+chunking tag*" as the default setting in the following experiments if not otherwise stated.

Number of Sentences Affected by Generated Phrases We also check the number of translations that contain phrases generated by the proposed model, as shown in Table 2. As seen, a large portion of translations take the recommended phrases, and the number increases when the chunking tag feature is used.[7] Considering BLEU scores reported in Table 1, we believe that the chunking tag feature benefits the proposed model on its phrase generation.

4.2 Analysis on Generated Phrases

Syntactic Categories of Generated Phrases We first investigate which category of phrases is more likely to be selected by the proposed approach. There are some phrases, such as noun phrases (NPs, e.g., "national laboratory" and "vietnam airlines") and quantifier phrases (QPs, e.g., "15 seconds" and "two weeks") , that we expect to be favored by our approach. Statistics shown in Table 3 confirm our hypothesis. Let's first concern all generated phrases (i.e., column "All"): most selected phrases are noun phrases (81.0%) and quantifier phrases (10.8%). Among them, 44.5% percent of them are fully correct[8]. Specifically, NPs have relative higher generation accuracy (i.e., 47.8% = 38.7%/81.0%) while VPs have lower accuracy (i.e., 21.2% = 1.7%/8.0%). By looking into the wrong cases, we found most errors are related to verb tense, which is the drawback of SMT models.

Concerning the newly introduced phrases that cannot be found in baseline translations (i.e., column "New"), 13.2% of generated phrases are both new and fully correct, which contribute most to the performance improvement. We can also find that most newly introduced verb phrases and quantifier phrases are not correct, the patterns of which can be well learned by word-based NMT models.

[7] The numbers on NIST08 are relatively lower since part of the test set contains sentences from Web forums, which contain less multi-word expressions.

[8] Fully correct means that the generated phrases can be retrieved in corresponding references as a whole unit.

Words	All		New	
	Total	Correct	Total	Correct
2	66.2%	33.6%	34.9%	9.1%
3	20.7%	8.4%	13.4%	3.2%
4	7.4%	1.9%	5.4%	0.6%
$\geqslant$5	5.7%	0.6%	5.2%	0.3%

Table 4: Percentages of phrases with different word counts to the total number of generated ones.

Number of Words in Generated Phrases Table 4 lists the distribution of generated phrases based on the number of inside words. As seen, most generated phrases are short phrases (e.g., 2-gram and 3-gram phrases), which also contribute most to the new and fully correct phrases (i.e., $12.3\% = 9.1\% + 3.2\%$). Focusing on long phrases (e.g., order$\geqslant$ 4), most of them are newly introduced (10.6% out of 13.1%). Unfortunately, only a few portion of these phrases are fully correct, since long phrases have higher chance to contain one or two unmatched words.

SYSTEM	Test
+memory	32.95
+memory +NULL	31.63
+memory +chunking tag	33.55
+memory +chunking tag +NULL	30.81

Table 5: Additional experiment results on the translation task to directly measure the improvement obtained by the phrase generation. "+NULL" denotes that we replace the generated target phrases with a special symbol "NULL" in test sets. BLEU scores in the table are case insensitive.

Effect of Generated Phrases on Translation Performance Note that the proposed model benefits not only from fully matched phrases, but also from partially matched phrases. For example, the baseline system translates "国家 航空 暨 太空 总署" in a word-by-word manner and outputs "state aviation and space department". The generated phrase provided by SMT is "national aviation and space administration", but the only correct reference is "national aeronautics and space administration". The generated phrase is not fully correct but still useful.

To directly measure the improvement obtained by the phrase generation, we replace the generated target phrases with a special symbol "NULL" in test sets. As shown in Table 5, when deleting the generated target phrases, ("*+memory+chunking tag*") and ("*+memory*") translation performances decrease by 2.74 BLEU points and 1.32 BLEU points respectively. Moreover, translation performances on NIST08 decrease less than those on NIST04 and NIST05 in both settings. The reason is that NIST08 which contains sentences from web data has little influence on generating target phrases which are provided from a different domain [9]. The overall results demonstrate that neural machine translation benefits from phrase translation.

4.3 Effect of Balancer

Weight	Test
Dynamic	33.55
Constant ($\lambda = 0.1$)	31.35

Table 6: Translation performance with a variety of balancing weight strategies. "Dynamic" is the proposed approach and "Constant ($\lambda = 0.1$)" denotes fixing the balancing weight to 0.1. BLEU scores in the table are case insensitive.

The balancer which is used to coordinate the phrase generation and word generation is very crucial for the proposed model. We conducted an additional experiment to validate the effectiveness of the neural network based balancer. We use the setting "+memory +chunking tag" as baseline system to conduct the experiments. In this experiment, we fixed the balancing weight λ (Equation 8) to 0.1 during training and testing and report the results. As shown in Table 6, we find that using the fixed value for the balancing weight (*Constant ($\lambda = 0.1$)*) decreases the translation performance sharply. This demonstrates that the neural network based balancer is an essential component for the proposed model.

4.4 Comparison to Word-Level Recommendations and Discussions

Our approach is related to our previous work (Wang et al., 2017) which integrates the SMT word-level knowledge into NMT. To make a comparison, we conducted experiments followed settings in (Wang et al., 2017). The comparison results are reported in Table 7. We find that our approach is marginally better than the word-level

[9] The parallel training data are mainly from news domain.

SYSTEM	Test
+word level recommendation	33.27
+memory +chunking tag	33.55

Table 7: Experiment results on the translation task. "+word level recommendation" is the proposed model in (Wang et al., 2017). BLEU scores in the table are case insensitive.

model proposed in (Wang et al., 2017) by 0.28 BLEU points.

In our approach, the SMT model translates source-side chunk phrases using the NMT decoding information. Although we use high-quality target phrases as phrasal recommendations, our approach still suffers from the errors in segmentation and chunking. For example, the target phrase "laptop computers" cannot be recommended by the SMT model if the Chinese phrase "手提电脑" is not chunked as a phrase unit. This is the reason why some sentences do not have corresponding phrasal recommendations (Table 2). Therefore, our approach can be further enhanced if we can reduce the error propagations from the segmenter or chunker, for example, by using n-best chunk sequences instead of the single best chunk sequence.

Additionally, we also observe that some target phrasal recommendations have been also generated by the baseline system in a word-by-word manner. These phrases, even taken as parts of final translations by the proposed model, do not lead to improvements in terms of BLEU as they have already occurred in translations from the baseline system. For example, the proposed model successfully carries out the phrase generation mode to generate a target phrase "guangdong province" (the translation of Chinese phrase "广东省") which has appeared in the baseline system.

As external resources, e.g., bilingual dictionary, which are complementary to the SMT phrasal recommendations, are compatible with the proposed model, we believe that the proposed model will get further improvement by using external resources.

5 Related work

Our work is related to the following research topics on NMT:

Generating phrases for NMT In these studies, the generated NMT multi-word phrases are either from an SMT model or a bilingual dictionary. In syntactically guided neural machine translation (SGNMT), the NMT decoder uses phrase translations produced by the hierarchical phrase-based SMT system Hiero, as hard decoding constraints. In this way, syntactic phrases are generated by the NMT decoder (Stahlberg et al., 2016b). Zhang and Zong (2016) use an SMT translation system, which is integrated an additional bilingual dictionary, to synthesize pseudo-parallel sentences and feed the sentences into the training of NMT in order to translate low-frequency words or phrases. Tang et al. (2016) propose an external phrase memory that stores phrase pairs in symbolic forms for NMT. During decoding, the NMT decoder enquires the phrase memory and properly generates phrase translations. The significant differences between these efforts and ours are 1) that we dynamically generate phrase translations via an SMT model, and 2) that at the same time we modify the encoder to incorporate structural information to enhance the capability of NMT in phrase translation.

Incorporating linguistic information into NMT NMT is essentially a sequence to sequence mapping network that treats the input/output units, eg., words, subwords (Sennrich et al., 2016), characters (Chung et al., 2016; Costa-jussà and Fonollosa, 2016), as non-linguistic symbols. However, linguistic information can be viewed as the task-specific knowledge, which may be a useful supplementary to the sequence to sequence mapping network. To this end, various kinds of linguistic annotations have been introduced into NMT to improve its translation performance. Sennrich and Haddow (2016) enrich the input units of NMT with various linguistic features, including lemmas, part-of-speech tags, syntactic dependency labels and morphological features. García-Martínez et al. (2016) propose factored NMT using the morphological and grammatical decomposition of the words (factors) in output units. Eriguchi et al. (2016) explore the phrase structures of input sentences and propose a tree-to-sequence attention model for the vanilla NMT model. Li et al. (2017) propose to linearize source-side parse trees to obtain structural label sequences and explicitly incorporated the structural sequences into NMT, while Aharoni and Goldberg (2017) propose to incorporate target-side syntactic information into NMT by serializing the target sequences into linearized, lexicalized constituency trees. Zhang et al. (2016) integrate topic knowledge into NMT for domain/topic adaptation.

Combining NMT and SMT A variety of approaches have been explored for leveraging the advantages of both NMT and conventional SMT. He et al. (2016) integrate SMT features with the NMT model under the log-linear framework in order to help NMT alleviate the limited vocabulary problem (Luong et al., 2015; Jean et al., 2015) and coverage problem (Tu et al., 2016). Arthur et al. (2016) observe that NMT is prone to making mistakes in translating low-frequency content words and therefore attempt at incorporating discrete translation lexicons into the NMT model, to alliterate the imprecise translation problem (Wang et al., 2017). Motivated by the complementary strengths of syntactical SMT and NMT, different combination schemes of Hiero and NMT have been exploited to form SGNMT (Stahlberg et al., 2016a,b). Wang et al. (2017) propose an approach to incorporate the SMT model into attention-based NMT. They combine NMT posteriors with SMT word recommendations through linear interpolation implemented by a gating function which dynamically assigns the weights. Niehues et al. (2016) propose to use SMT to pre-translate the inputs into target translations and employ the target pre-translations as input sequences in NMT. Zhou et al. (2017) propose a neural system combination framework to directly combine NMT and SMT outputs. The combination of NMT and SMT has been also introduced in interactive machine translation to improve the system's suggestion quality (Wuebker et al., 2016). In addition, word alignments from the traditional SMT pipeline are also used to improve the attention mechanism in NMT (Cohn et al., 2016; Mi et al., 2016; Liu et al., 2016).

6 Conclusion

In this paper, we have presented a novel model to translate source phrases and generate target phrase translations in NMT by integrating the phrase memory into the encoder-decoder architecture. At decoding, the SMT model dynamically generates relevant target phrases with contextual information provided by the NMT model and writes them to the phrase memory. Then the proposed model reads the phrase memory and uses the balancer to make probability estimations for the phrases in the phrase memory. Finally the NMT decoder selects a phrase from the phrase memory or a word from the vocabulary of the highest probability to generate. Experiment results on Chinese→English translation have demonstrated that the proposed model can significantly improve the translation performance.

Acknowledgments

We would like to thank three anonymous reviewers for their insightful comments, and also acknowledge Zhengdong Lu, Lili Mou for useful discussions. This work was supported by the National Natural Science Foundation of China (Grants No.61525205, 61373095 and 61622209).

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. *arXiv preprint arXiv:1704.04743*.

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on EMNLP*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd ACL*.

Kyunghyun Cho. 2015. Natural language understanding with distributed representation. *arXiv preprint*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on EMNLP*.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th ACL*, pages 1693–1703, Berlin, Germany.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of NAACL 2016*, San Diego, California.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th ACL*, pages 357–361, Berlin, Germany.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th ACL*, pages 823–833, Berlin, Germany.

Mercedes García-Martínez, Loïc Barrault, and Fethi Bougares. 2016. Factored neural machine translation. *arXiv preprint arXiv:1609.04621*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th ACL*, Berlin, Germany.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of ACL 2016*, Berlin, Germany.

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. 2016. Improved neural machine translation with smt features. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd ACL and the 7th IJCNLP*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on EMNLP*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th ACL Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 NAACL*.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. *arXiv preprint arXiv:1705.01020*.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *Proceedings of COLING 2016*, pages 3093–3102, Osaka, Japan.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th ACL*.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd ACL and the 7th IJCNLP*.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Supervised attentions for neural machine translation. In *Proceedings of EMNLP 2016*, pages 2283–2288, Austin, Texas.

Jan Niehues, Eunah Cho, Thanh-Le Ha, and Alex Waibel. 2016. Pre-translation for neural machine translation. In *Proceedings of COLING 2016*, Osaka, Japan.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st ACL*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, pages 1310 - –1318.

Ivan A Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 1–15. Springer.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th ACL*, pages 1715–1725, Berlin, Germany.

Felix Stahlberg, Adrià de Gispert, Eva Hasler, and Bill Byrne. 2016a. Neural machine translation by minimising the bayes-risk with respect to syntactic translation lattices. *arXiv preprint arXiv:1612.03791*.

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016b. Syntactically guided neural machine translation. In *Proceedings of the 54th ACL (Volume 2: Short Papers)*, Berlin, Germany.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*.

Yaohua Tang, Fandong Meng, Zhengdong Lu, Hang Li, and Philip LH Yu. 2016. Neural machine translation with external phrase memory. *arXiv preprint arXiv:1606.01792*.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017a. Context gates for neural machine translation. *Transactions of the Association of Computational Linguistics*.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017b. Neural machine translation with reconstruction. In *Proceedings of the 31th AAAI Conference on Artificial Intelligence*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th ACL*.

Aline Villavicencio, Francis Bond, Anna Korhonen, and Diana McCarthy. 2005. Introduction to the special issue on multiword expressions: Having a crack at a hard nut. *Computer Speech & Language*, 19(4):365–377.

Xing Wang, Zhengdong Lu, Zhaopeng Tu, Hang Li, Deyi Xiong, and Min Zhang. 2017. Neural machine translation advised by statistical machine translation. In *Proceedings of the 31th AAAI Conference on Artificial Intelligence*.

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

Joern Wuebker, Spence Green, John DeNero, Sasa Hasan, and Minh-Thang Luong. 2016. Models and inference for prefix-constrained machine translation. In *Proceedings of the 54th ACL*, pages 66–75, Berlin, Germany.

Nianwen Xue and Libin Shen. 2003. Chinese word segmentation as lmr tagging. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pages 176–179, Sapporo, Japan.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Jiajun Zhang and Chengqing Zong. 2016. Bridging neural machine translation and bilingual dictionaries. *arXiv preprint arXiv:1610.07272*.

Jian Zhang, Liangyou Li, Andy Way, and Qun Liu. 2016. Topic-informed neural machine translation. In *Proceedings of COLING 2016*, pages 1807–1817, Osaka, Japan.

Long Zhou, Wenpeng Hu, Jiajun Zhang, and Chengqing Zong. 2017. Neural system combination for machine translation. *arXiv preprint arXiv:1704.06393*.

Jingbo Zhu, Muhua Zhu, Qiang Wang, and Tong Xiao. 2015. Niuparser: A chinese syntactic and semantic parsing toolkit. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, Beijing, China.

Towards Bidirectional Hierarchical Representations for Attention-Based Neural Machine Translation

Baosong Yang[†] **Derek F. Wong**[†*] **Tong Xiao**[‡] **Lidia S. Chao**[†] **Jingbo Zhu**[‡]
[†]NLP^{2}CT Lab, Department of Computer and Information Science,
University of Macau, Macau, China
[‡]NiuTrans Lab, Northeastern University, Shenyang, China
nlp2ct.baosong@gmail.com, {derekfw,lidiasc}@umac.mo,
{xiaotong,zhujingbo}@mail.neu.edu.cn

Abstract

This paper proposes a hierarchical attentional neural translation model which focuses on enhancing source-side hierarchical representations by covering both local and global semantic information using a bidirectional tree-based encoder. To maximize the predictive likelihood of target words, a weighted variant of an attention mechanism is used to balance the attentive information between lexical and phrase vectors. Using a tree-based rare word encoding, the proposed model is extended to sub-word level to alleviate the out-of-vocabulary (OOV) problem. Empirical results reveal that the proposed model significantly outperforms sequence-to-sequence attention-based and tree-based neural translation models in English-Chinese translation tasks.

1 Introduction

Neural machine translation (NMT) automatically learns the abstract features of and semantic relationship between the source and target sentences, and has recently given state-of-the-art results for various translation tasks (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015). The most widely used model is the encoder-decoder framework (Sutskever et al., 2014), in which the source sentence is encoded into a dense representation, followed by a decoding process which generates the target translation. By exploiting the attention mechanism (Bahdanau et al., 2015), the generation of target words is conditional on the source hidden states, rather than on the context vector alone. From a model architecture perspective, prior studies of the attentive encoder-decoder translation model are mainly divided into two types.

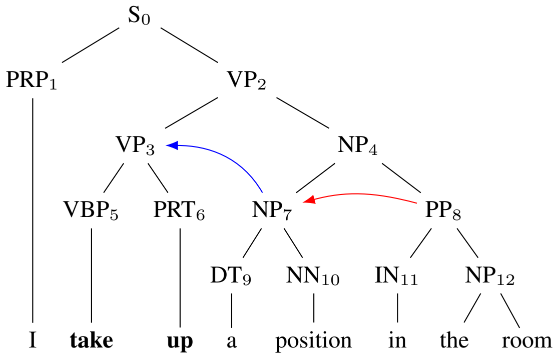


Figure 1: Induction of phrase and sentence representations over the syntactic structure of a sentence.

The **sequence-to-sequence model** treats a sentence as a sequence of tokens. The most fundamental approaches transform the source sentence sequentially into a fixed-length context vector, and the annotation vector of each word summarizes the preceding words (Sutskever et al., 2014; Cho et al., 2014b). Although Bahdanau et al. (2015) used a bidirectional recurrent neural network (RNN) (Schuster and Paliwal, 1997) to consider preceding and following words jointly, these sequential representations are insufficient to fully capture the semantics of a sentence, due to the fact that they do not account for the syntactic interpretations of sentence structure (Eriguchi et al., 2016; Tai et al., 2015). By incorporating additional features into a sequential model, Sennrich and Haddow (2016) and Stahlberg et al. (2016) suggest that a greater amount of linguistic information can improve the translation performance.

The **tree-to-sequence model** encodes a source sentence according to a given syntactic tree

[*] Corresponding author

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1432–1441
Copenhagen, Denmark, September 7–11, 2017.

over the sentence. The existing tree-based encoders (Tai et al., 2015; Eriguchi et al., 2016; Zhou et al., 2016) recursively generate phrase (sentence) representations in a bottom-up fashion, whereby the annotation vector of each phrase is derived from its constituent sub-phrases. As a result, the learned representations are limited to local information, while failing to capture the global meaning of a sentence. As illustrated in Figure 1, the phrases "take up"[1] and "a position"[2] have different meanings in different contexts. However, in composing the representations h_{VP_3} and h_{NP_7} for phrases VP$_3$ and NP$_7$, the current approaches do not account for the differences in meaning which arise as a result of ignoring the neighboring context as well as the remote context, i.e. $h_{\mathrm{NP}_7} \leftarrow h_{\mathrm{PP}_8}$ (sibling) and $h_{\mathrm{VP}_3} \leftarrow h_{\mathrm{NP}_7}$ (child of sibling). More specifically, at the encoding step t, the generated phrase is based on the results at the previous time steps h_{t-1} and h_{t-2}, but has no information about the parent phrases $h_{t'}$ for $t' > t$.

To address the above problems, we propose a novel architecture, a bidirectional hierarchical encoder, which extends the existing attentive tree-structured models (Eriguchi et al., 2016). In contrast to the model of Eriguchi et al. (2016), we first use a bidirectional RNN (Schuster and Paliwal, 1997) at lexical level to concatenate the forward and backward states as the hidden states of source words, to capture the preceding and following contexts (described in Section 3.1). Secondly, we propose a bidirectional tree-based encoder (described in Section 3.2), in which the original bottom-up encoding model is extended using an additional top-down encoding process. In the bidirectional hierarchical model, the vector representations of the sentence, phrases as well as words, are therefore based on the global context rather than local information.

To effectively leverage hierarchical representations in generating the target words, we adopt a variant weighted tree-based attention mechanism (described in Section 3.4) in which a time-dependent gating scalar is used to control the proportion of conditional information between the word and phrase vectors. To alleviate the out-of-vocabulary (OOV) problem, we further extend the proposed tree-based model to the sub-word level by integrating byte-pair encoding (BPE) (Sennrich et al., 2016) into the tree-based model (as described in Section 3.3). Experimental results for the NIST English-to-Chinese translation task reveal that the proposed model significantly outperforms the vanilla tree-based (Eriguchi et al., 2016) and sequential NMT models (Bahdanau et al., 2015) (Section 4.1).

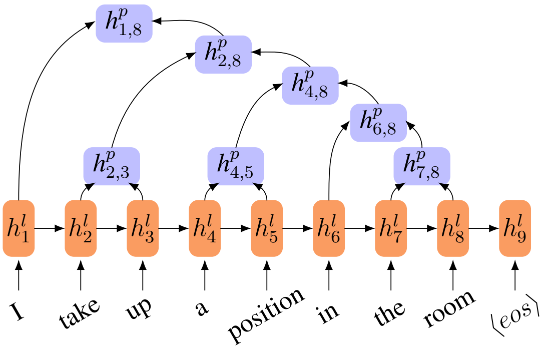


Figure 2: The tree-based model of Eriguchi et al. (2016) comprising a structured and sequential encoder.

2 Tree-Based Neural Machine Translation

A neural machine translation system (NMT) aims to use a single neural network to build a translation model, which is trained to maximize the conditional distribution of sentence pairs using a parallel training corpus (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014b,a). By incorporating syntactic information, the tree-based NMT exploits an additional syntactic structure of the source sentence to improve the translation. Since most existing NMTs generate one target word at a time, given a source sentence $\mathbf{x} = (x_1, ..., x_N)$ and its corresponding syntactic tree $\mathbf{tr}$, the conditional probability of a target sentence $\mathbf{y} = (y_1, ..., y_M)$ is formally expressed as:

$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{tr}) = \prod_{1}^{M} p(y_j \mid y_1, ..., y_{j-1}, \mathbf{x}, \mathbf{tr}; \theta),$$

where θ represents the model parameters. A tree-based NMT consists of a tree-based encoder and a decoder.

2.1 Tree-Based Encoder

In a tree-based encoder, the source language $\mathbf{x}$ is encoded according to a given syntactic structure

[1] **Take up** has the meanings of *start doing something new*, *use space/time*, *accept an offer*, etc.

[2] **Position** has the meanings of *location*, *job offer*, *rank/status*, etc.

tr of the sentence. As shown in Figure 2, Eriguchi et al. (2016) employed a forward Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) recurrent neural network (RNN) to encode the lexical nodes and a tree-LSTM (Tai et al., 2015) to generate the phrase representations in a bottom-up fashion. In the present study, we utilize the gated recurrent unit (GRU) (Cho et al., 2014b) instead of an LSTM, in view of its comparable performance (Chung et al., 2014) and since it yields even better results for certain tasks (Józefowicz et al., 2015). The lexical annotation vectors $(h^l_1, ..., h^l_N)$ are sequentially generated by using a GRU. The i-th leaf node vector is calculated as:

$$h^l_i = f^l_{GRU}(x_i, h^l_{i-1}), \qquad (1)$$

where x_i is the i-th source word embedding and h^l_{i-1} denotes the previous hidden state. The parent hidden state $h^{\uparrow}_{i,j}$ summarizes its left child $h^{\uparrow}_{i,k}$ and right child $h^{\uparrow}_{k+1,j}$ $(i < k < j)$ by applying the tree-GRU (Zhou et al., 2016) as follows:

$$\begin{aligned}
z^{\uparrow}_{i,j} &= \sigma(U^L_{(z)} h^{\uparrow}_{i,k} + U^R_{(z)} h^{\uparrow}_{k+1,j} + b^{\uparrow}_{(z)}) \\
r^{\uparrow}_{i,k} &= \sigma(U^L_{(rL)} h^{\uparrow}_{i,k} + U^R_{(rL)} h^{\uparrow}_{k+1,j} + b^{\uparrow}_{(rL)}) \\
r^{\uparrow}_{k+1,j} &= \sigma(U^L_{(rR)} h^{\uparrow}_{i,k} + U^R_{(rR)} h^{\uparrow}_{k+1,j} + b^{\uparrow}_{(rR)}) \\
\widetilde{h}^{\uparrow}_{i,j} &= \tanh(U^L_{(h)}(r^{\uparrow}_{i,k} \odot h^{\uparrow}_{i,k}) \\
&\quad + U^R_{(h)}(r^{\uparrow}_{k+1,j} \odot h^{\uparrow}_{k+1,j}) + b^{\uparrow}_{(h)}) \\
h^{\uparrow}_{i,j} &= z^{\uparrow}_{i,j} \widetilde{h}^{\uparrow}_{i,j} + (1 - z^{\uparrow}_{i,j})(h^{\uparrow}_{i,k} + h^{\uparrow}_{k+1,j}),
\end{aligned}$$

where $z^{\uparrow}_{i,j}$ is the update gate; $r^{\uparrow}_{i,k}$, $r^{\uparrow}_{k+1,j}$ are the reset gates for the left and right children; $\widetilde{h}^{\uparrow}_{i,j}$ denotes the candidate activation; $U^L_{(\cdot)}$ and $U^R_{(\cdot)}$ represent weight matrices; $b^{\uparrow}_{(\cdot)}$ denote bias vectors; σ is the logistic sigmoid function; and the operator $\odot$ denotes element-wise multiplication between vectors. The phrase representations are recursively built in an upward direction.

2.2 Decoding with a Tree-Based Attention Mechanism

In generating the target words, we employ a sequential decoder with an input-feeding method (Luong et al., 2015) and attention mechanism (Bahdanau et al., 2015). The conditional probability of the j-th target word y_j is calculated using a non-linear function $f_{softmax}$:

$$p(y_j \mid y_1, ..., y_{j-1}, \mathbf{x}, \mathbf{tr}; \theta) = f_{softmax}(c_j),$$

where c_j is the composite hidden state, which consists of a target hidden state s_j and a context vector d_j:

$$c_j = f_{\tanh}([s_j, d_j]).$$

Given the previous target word y_{j-1}, the concatenation of the previous hidden state s_{j-1} and the previous context vector c_{j-1} (input-feeding) (Luong et al., 2015), s_j, is calculated using a standard sequential GRU network:

$$s_j = f^{dec}_{gru}(y_{j-1}, [s_{j-1}, c_{j-1}]).$$

The context vector d_j is computed using an attention model which is used to softly summarize the attended part of the source-side representations. Eriguchi et al. (2016) adopted a tree-based attention mechanism to consider both the word and phrase vectors:

$$d_j = \sum_{i=1}^{N} \alpha_j(i) h^l_i + \sum_{k=1}^{N-1} \alpha_j(k) h^p_k, \qquad (2)$$

where h^l_i is the i-th hidden state of the source word at leaf level, and h^p_k is the k-th hidden state of the source phrase. The weight $\alpha_j(t)$ of node t is computed by:

$$\begin{aligned}
\alpha_j(t) &= \frac{\exp(e_t)}{\sum_{i=1}^{N} \exp(e^l_i) + \sum_{k=1}^{N-1} \exp(e^p_k)} \\
e_t &= (V_a)^T \tanh(U_a s_j + W_a h_t + b_a),
\end{aligned}$$

where h_t is the hidden state of the node. V_a, U_a, W_a and b_a are the model parameters.

3 The Bidirectional Hierarchical Model

Although the tree-based encoder of Eriguchi et al. (2016) has shown certain advantages in translation tasks involving distant language pairs, e.g. English-Japanese, the representation of a phrase relies solely on its child nodes, and the word representation at leaf level only takes into account the sequential information. We argue that the incorporation of more hierarchical information into the representations may contribute to an improvement in the translation. In particular, the use of global information can help in distinguishing the differences between word meanings. Based on this hypothesis, we propose an alternative architecture, the bidirectional hierarchical model, to enhance the source-side representations.

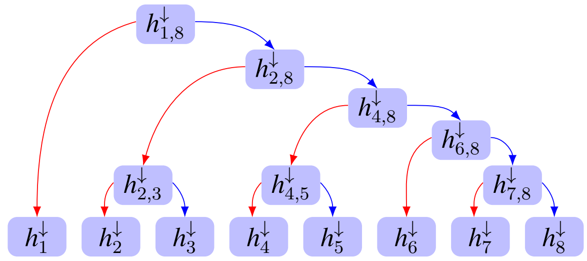


Figure 3: A top-down encoding process updates the hidden states recursively from root to leaf nodes. The red and blue lines denote the use of different learning parameters.

3.1 Bidirectional Leaf-Node Encoding

As discussed in Section 1, the unidirectional recurrent neural network reads an input sequence in order, from the first symbol to the last. In order to generate leaf node annotation vectors which jointly take into account both preceding and following annotations, we exploit a bidirectional RNN encoder (Bahdanau et al., 2015). The hidden state of the i-th leaf node h^l_i is the concatenation of the forward and backward vectors:

$$h^l_i = [\overrightarrow{h}^l_i, \overleftarrow{h}^l_i],$$

where $\overrightarrow{h}^l_i$ is obtained by a rightward GRU, as shown in Equation 1, and a leftward GRU calculates $\overleftarrow{h}^l_i$, as follows:

$$\overleftarrow{h}^l_i = f_{\overleftarrow{GRU}}(x_i, \overleftarrow{h}^l_{i-1}),$$

where $\overleftarrow{h}^l_{i-1}$ is the previous hidden state.

3.2 Bidirectional Tree-Node Encoding

Since the hidden states of leaf nodes are derived in a sequential, context-sensitive way, by generating phrase annotations in a bottom-up fashion, the sequential context can be propagated to tree nodes. However, the learned annotation vectors still fail to capture global information from the upper nodes. To enhance the representations with global semantic information, we propose to use a standard GRU recurrent network to update representations in a **top-down fashion**, as shown in Figure 3. The annotation vectors, which are learned by the previous encoding steps, are fed to the updating process.

First, we treat the bottom-up hidden state of root $h^{\uparrow}_{root}$, which covers the global meaning as well as the syntactic information of the source sentence, as the initial state of the top-down GRU network:

$$h^{\downarrow}_{root} = h^{\uparrow}_{root}.$$

Given an updated hidden state of the parent node $h^{\downarrow}_{i,j}$, the hidden states of left and right children $h^{\downarrow}_{i,k}$ and $h^{\downarrow}_{k+1,j}$ are calculated as:

$$\begin{aligned} h^{\downarrow}_{i,k} &= f^{ld}_{GRU}(h^{\uparrow}_{i,k}, h^{\downarrow}_{i,j}) \\ h^{\downarrow}_{k+1,j} &= f^{rd}_{GRU}(h^{\uparrow}_{k+1,j}, h^{\downarrow}_{i,j}), \end{aligned}$$

where $h^{\uparrow}_{i,k}$ and $h^{\uparrow}_{k+1,j}$ are the left and right child annotation vectors generated via the bottom-up tree-GRU network. Contrary to the similar top-down encoding for sentiment classification (Kokkinos and Potamianos, 2017), which uses same weighting parameters to handle both left and right child nodes, f^{ld}_{GRU} and f^{rd}_{GRU} with different parameters are applied in the proposed model to distinguish the left and right structural information. According to the definition of a GRU (Cho et al., 2014b), f^{ld}_{GRU} uses an update gate $z^{\downarrow}_{i,k}$, a reset gate $r^{\downarrow}_{i,k}$ and a candidate activation $\widetilde{h}^{\downarrow}_{i,k}$ to generate $h^{\downarrow}_{i,k}$, as follows:

$$\begin{aligned} z^{\downarrow}_{i,k} &= \sigma(W^{ld}_{(z)} h^{\uparrow}_{i,k} + U^{ld}_{(z)} h^{\downarrow}_{i,j} + b^{ld}_{(z)}) \\ r^{\downarrow}_{i,k} &= \sigma(W^{ld}_{(r)} h^{\uparrow}_{i,k} + U^{ld}_{(r)} h^{\downarrow}_{i,j} + b^{ld}_{(r)}) \\ \widetilde{h}^{\downarrow}_{i,k} &= \tanh(W^{ld}_{(h)} h^{\uparrow}_{i,k} + U^{ld}_{(h)} (r^{\downarrow}_{i,k} \odot h^{\downarrow}_{i,j}) + b^{ld}_{(h)}) \\ h^{\downarrow}_{i,k} &= (1 - z^{\downarrow}_{i,k}) h^{\downarrow}_{i,j} + z^{\downarrow}_{i,k} \widetilde{h}^{\downarrow}_{i,k}, \end{aligned} \quad (3)$$

where $W^{ld}_{(\cdot)}$ and $U^{ld}_{(\cdot)}$ represent weight matrices, and $b^{ld}_{(\cdot)}$ denote bias vectors. f^{rd}_{GRU} is defined in a similar way.

From a linguistic point of view, in the top-down GRU network, the reset gate is able to retain the useful global information and drop irrelevant information from the parent state $h^{\downarrow}_{i,j}$, while the proportions of the global context from the top-down state $h^{\downarrow}_{i,j}$, and the local context from the bottom-up state $h^{\uparrow}_{i,k}$ are controlled by the update gate. As it covers both the partial meaning of the phrase and the whole meaning of the sentence, $h^{\downarrow}_{i,k}$ is regarded as the final representation of $node_{i,k}$:

$$h^p_{i,k} = h^{\downarrow}_{i,k}.$$

With the propagation of information from root to leaf nodes, the i-th leaf node representation is updated as:

$$h^l_i = h^{\downarrow}_i.$$

As each source-side hidden state of the leaf nodes and tree nodes carries the hierarchical information of the sentence, we interpret such an encoded state as a **hierarchical representation**.

3.3 Handling Out-of-Vocabulary: Tree-Based Rare Word Encoding

In NMT, the translation of rare words and unknown words is an open problem, since the computational cost increases with the size of the vocabulary. Sennrich et al. (2016) proposed a simple and effective approach to handling out-of-vocabulary by representing rare words as a sequence of sub-word units, which are segmented using byte-pair encoding (BPE) (Gage, 1994).

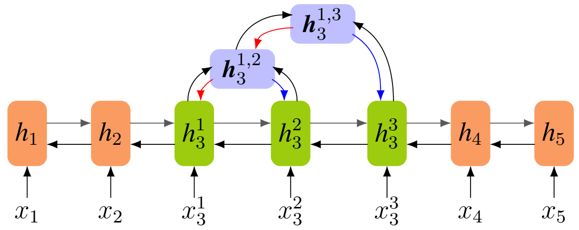


Figure 4: Encoding sub-word units with an additional binary lexical tree, where x_3^1, x_3^2, x_3^3 are the sub-units of word x_3.

We propose a variant tree-based rare word encoding approach which extends the tree-based model to the sub-word level. Sub-word units are encoded following an additional binary lexical tree. For a sentence $\mathbf{x} = (x_1, ..., x_i, ..., x_N)$, BPE segments the word x_i into a sequence of sub-word units $(x_i^1, ..., x_i^n)$. The binary lexical tree is simply built by composing two nodes in a rightwards fashion, $(((x_i^1, x_i^2), x_i^3)...), x_i^n)$, as shown in Figure 4. From the i-th leaf node, the original syntactic tree is extended downwards using the binary lexical tree, and the set of leaf nodes are replenished as $\mathbf{x} = (x_1, ..., x_i^1, x_i^2, ..., x_i^n, ..., x_N)$. Sub-word units can therefore be regarded as leaf nodes, and can be encoded using the proposed encoder, as illustrated in Figure 5. The experimental results in Section 4.1 demonstrate the effectiveness of this simple approach.

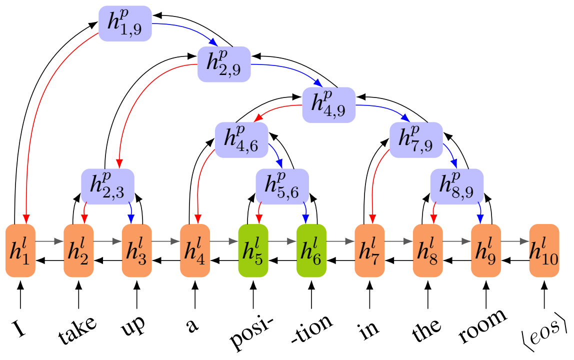


Figure 5: Illustration of the bidirectional hierarchical encoder: representations are enhanced by a bidirectional leaf-node encoding and a bidirectional tree-node encoding. The green nodes indicate the sub-word representations.

3.4 Decoder with Weighted Variant of Attention Mechanism

Since each representation carries both local and global information, in this case, attending fairly to the lexical and phrase representations in each decoding step may cause the problem of over-translation (repeatedly attending and translating the same constituent of a sentence). An alternative approach is to balance the attentive information between the lexical and phrase vectors in the context vector. To effectively leverage these hierarchical representations, we propose a weighted variant of the tree-based attention mechanism (the original is defined in Equation 2). Formally, the calculation of the context vector d_j at step j is modified as:

$$d_j = (1 - \beta_j) \sum_{i=1}^{n} \alpha_j(i) h_i^l + \beta_j \sum_{k=1}^{n-1} \alpha_j(k) h_k^p \quad (4)$$

where $\beta_j \in [0, 1]$ is used to weight the expected importance of the representations. Inspired by work on a multi-modal NMT (Calixto et al., 2017) which exploits a gating scalar (Xu et al., 2015) to weight the image context vector, we use such a scalar in our model in order to dynamically adapt the weighting scalar. The gating scalar β_j at step j is calculated by :

$$\beta_j = \sigma(W_\beta c_{j-1} + b_\beta),$$

where W_β and b_β represent the model parameters. In contrast with α, which denotes the correspondence between each source annotation and the current target hidden state, β is dominated by the target composite hidden state alone. In other words, β is a time-dependent scalar in relation to the current target word, and therefore enables the attention model to explicitly quantify how far the leaf and no-leaf states contribute to the word prediction at each time step. In the proposed model, the phrase and lexical context vectors are learned by a single attention model, meaning that they are dependent, and the gating scalar weights the phrase and lexical context vectors in complementary fashion, as shown in Equation 4. This distinguishes the model from that introduced by Calixto et al. (2017), in which the context vectors of the source sentence and image (bi-modal) are measured using two independent attention models and the gating scalar is merely used to weight the image context vector.

4 Experiments

4.1 Data

Training	Dev	Test		
LDC En-Zh	mt08	mt04	mt05	mt06
1,435,575	1,357	1,788	1,082	1,664

Table 1: Data used in the experiments.

We evaluate the proposed model on an English-to-Chinese translation task. For reasons of computational efficiency, we extracted 1.4M sentence pairs, in which the maximum length of the sentence was 40, from the LDC parallel corpus[3] as our training data. The models were developed using NIST mt08 data and were examined using NIST mt04, mt05, and mt06 data. The number of sentences in each dataset is shown in Table 1. On the English side, we used the *constituent parser* (Zeng et al., 2014, 2015) to produce a binary syntactic tree for each sentence, in constrast to the use of the *HPSG parser* by Eriguchi et al. (2016). On the Chinese side, the sentences are segmented using the Chinese word segmentation toolkit of *NiuTrans* (Xiao et al., 2012).

To avoid data sparsity, words referring to time, date and number, which are low in frequency, are generalized as '$time', '$date' and '$number'. In addition, as described in Section 3.3, the vocabularies are further compressed by segmenting the rare words into sub-word units using BPE.

4.2 Experimental Settings

Training Set	Original	Generalization	BPE
$\|V\|$ in En	159k	120k	40k
$\|V\|$ in Zh	198k	125k	40k

Table 2: The vocabulary size of the training set before and after applying the generalization and BPE segmentation.

As shown in Table 2, which gives the statistics of the token types, we limit the source and target vocabulary size to 40,000, in order to cover all the English and Chinese tokens. The dimensions of word embedding and hidden layer are respectively set as 620 and 1,000. Due to the concatenation in the bidirectional leaf-node encoding, the dimensions of the forward and backward vectors, which are half of those of the other hidden states, are set to 500. In order to prevent over-fitting, the training data is shuffled following each epoch. Moreover, the model parameters are optimized using AdaDelta (Zeiler, 2012), due to its capability for dynamically adapting the learning rate. We set the mini-batch size to 16 and the beam search size to 5. The accuracy of the translation relative to a reference is assessed using the BLEU metric (Papineni et al., 2002). In order to give an equitable comparison, all the NMT models used for comparison are implemented or re-implemented using GRU in our code, based on *dl4mt*[4].

4.3 Enhanced Hierarchical Representations

Firstly, the effectiveness of the enhanced hierarchical representations is evaluated through a set of experiments, the results of which are summarized in Table 3.

Compared with the original **tree-based encoder** (Eriguchi et al., 2016), the model with bidirectional leaf-node encoding (described in Section 3.1) shows better performance. This also reveals that the future context at leaf level can contribute to word prediction. Secondly, although the representations of leaf nodes are learned in a sequential, context-sensitive way, the translation quality is further improved by considering the global semantic information in the top-down encoding (Section 3.2).

By incorporating the above enhancements into the model, the proposed **hierarchical encoder** yields significant improvements over both the sequential and the tree-based models. The problem of OOV is alleviated by further extending the tree-

[3] Our training data was selected from *LDC2000T46*, *LDC2000T50*, *LDC2003E14*, *LDC2004T08*, *LDC2004T08* and *LDC2005T10*.

[4] `https://github.com/nyu-dl/dl4mt-tutorial`

Model	BPE	# of params	MT04	MT05	MT06	Dev.
sequential encoder	no	86.8M	31.26	23.98	24.02	17.20
+ sequential rare word encoding	yes	86.8M	32.54	25.09	25.07	18.19
+ tree-based rare word encoding	yes	104.1M	32.56	25.30	24.96	18.33
tree-based encoder	no	95.0M	31.90	24.68	24.40	17.63
+ bidirectional leaf-node encoding	no	92.0M	32.13	24.94	25.02	18.12
+ top-down encoding	no	101.1M	32.85	25.37	25.30	18.26
+ tree-based rare word encoding	yes	95.0M	33.02	25.62	25.24	18.59
hierarchical encoder ($\beta = 0.5$)	no	104.1M	32.91↑	25.55↑	25.52↑	18.46↑
hierarchical encoder ($\beta = 0.5$)	yes	104.1M	33.81⇑	26.47⇑	26.31⇑	19.41⇑
+ gating scalar	yes	105.1M	**34.33**⇑	**26.72**⇑	**26.58**⇑	**20.10**⇑

Table 3: Translation results for the various models. The first column shows the models; the second column indicates whether the corresponding experiment uses BPE data. The number of parameters (M = millions) in each model is given in the third column. The remaining columns are the translation accuracies for the test sets and development set, evaluated using BLEU scores (%). "↑ / ⇑": indicates that the hierarchical encoder is significantly better than the vanilla tree-based encoder ($p < 0.05/p < 0.01$).

based model to sub-word level (Section 3.3). In addition, we evaluate our tree-based rare word encoding method against the conventional rare word encoding (Sennrich et al., 2016) using the **sequential encoder** (Bahdanau et al., 2015). The empirical results confirm that our proposed tree-based BPE method achieves performance comparable to that of the standard BPE in the sequential model, but is applicable to the tree-based NMT model.

Overall, the proposed **hierarchical encoder** has demonstrated the ability to effectively model source-side representations from both the sequential and structural context. The NMT systems based on the proposed model significantly outperform those of conventional models using the **sequential encoder** and the **tree-based encoder**.

4.4 Weighted Attention Model

As discussed in Section 3.4, in order to effectively leverage hierarchical representations in generating the target word, we adopt a variant weighted tree-based attention mechanism which incorporates a scalar to control the proportion of conditional information between the word and phrase vectors. By manually or automatically varying the weight β, the utilization of the weighted attention model is assessed for four cases:

- $\beta = \mathbf{0.0}$: We manually set the weight of phrase vectors to 0.0; in other words, the decoder is forced to ignore the phrase vectors. The final translation is therefore generated by merely summarizing the leaf vectors.
- $\beta = \mathbf{0.5}$: The representations of non-leaf nodes and leaf nodes participate equally in the translation process. The decoder of this case therefore employs the same attention mechanism as that of the original model (Section 2.2).
- $\beta = \mathbf{1.0}$: In the reverse of the first case, the weight of the leaf nodes is manually set to 0.0. Thus, only the phrase vectors are used to predict the target words.
- Gating scalar (GS): A gating scalar is used for dynamically learning to control the proportion in which the lexical and phrase contexts contribute to the generation of the target words (Section 3.4).

Model	BLEU	Perplexity	Avg. Length
$\beta = 1.0$	17.16	98.65	21.13
$\beta = 0.5$	19.41	94.73	23.08
$\beta = 0.0$	19.83	94.68	23.33
GS	**20.10**	**94.18**	23.24

Table 4: Translation results for the development set. The last column indicates the average length of translation sentences, and the average length of reference sentences is 23.19.

The experimental results are shown in Table 4. The model which attends only to lexical annotation vectors ($\beta = 0.0$) gives slightly better performance than that which uses equal weights for

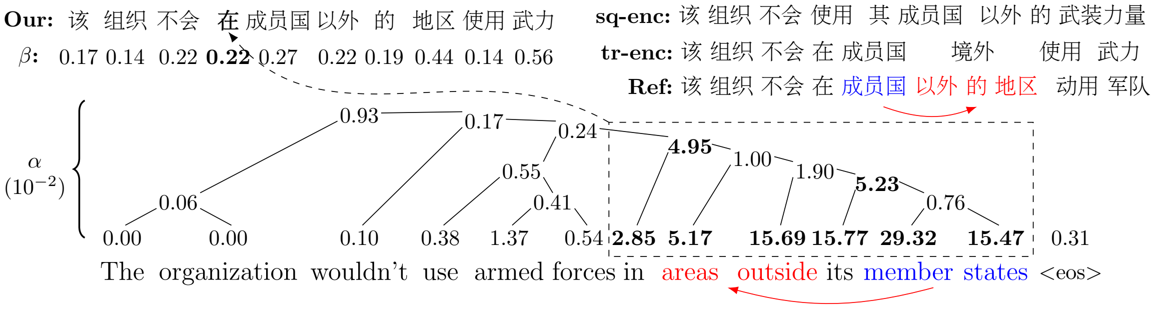

Our: 该 组织 不会 在 成员国 以外 的 地区 使用 武力
β: 0.17 0.14 0.22 0.22 0.27 0.22 0.19 0.44 0.14 0.56
sq-enc: 该 组织 不会 使用 其 成员国 以外 的 武装力量
tr-enc: 该 组织 不会 在 成员国 境外 使用 武力
Ref: 该 组织 不会 在 成员国 以外 的 地区 动用 军队
α
(10^−2)
0.93
0.17
0.24
0.06
0.55
0.41
4.95
1.00
1.90
5.23
0.76
0.00 0.00 0.10 0.38 1.37 0.54 2.85 5.17 15.69 15.77 29.32 15.47 0.31
The organization wouldn't use armed forces in areas outside its member states <eos>

Figure 6: Translations of an English sentence output using the NMT models with bidirectional hierarchical model (**our**), sequential encoder (**seq-enc**) and original tree-based encoder (**tr-enc**). **Ref** indicates the reference Chinese sentence. The attention scores (α), which are noted over the source-side syntactic tree, are output by the bidirectional hierarchical model at the step where the fourth target word "在" is translated. The sequence of scores β denote the value of the gating scalar at each translation step.

lexical and phrase vectors ($\beta = 0.5$). The use of global information contributes to distinguishing the differences between word meanings, although the similar semantic information in the lexical and phrase representations aggravates the over-translation problem observed in the translation results. However, we found that the model which attends only to phrase representations tends to generate shorter translation of an average of 21.13 words in length, as shown in the last column of the first row of Table 4. Furthermore, the model that neglects the leaf representations ($\beta = 1.0$) is likely to underperform the others that are also conditioned on the leaf nodes. Even though the phrase representations are derived from the lexical level via a bottom-up encoding, we believe it is unable to fully capture the lexical information of the source sentence. Through the use of the gating scalar, the hierarchical model achieves progressive improvements, as shown in Tables 3 and 4, the problem of over-translation is also alleviated. The representations of non-leaf nodes can be regarded as supplements in the translation process.

5 Qualitative Analysis

Figure 6 shows an English sentence and its binary tree representation, together with the corresponding Chinese translations produced by the different NMT models. All the models successfully give the correct Chinese translation "该 组织 不会" for the first three words of the English sentence "the organization wouldn't". Differences appear in the translation of the fourth word, and these lead to markedly different meanings. The translation "使用 其 成员国 以外 的 武装力量" output by the sequential model, means "use the armed forces other than its member states" where "other than its member states" is incorrectly interpreted as a complement to "armed forces". This is caused by the intrinsic limitations of the sequential model, whereby it is unable to properly interpret the syntactic relationship of words. By explicitly incorporating the syntactic information, both the proposed hierarchical model and the tree-based model can accurately attend to the dashed section of Figure 6, and the translations can be correctly generated to reflect the meaning of the source sentence. The distinction between the translations produced by the original tree-based model and our hierarchical model is the interpretation of the words "areas outside". The tree-based model interprets it into "境外 (outside)", while our model correctly translates it into "以外 的 地区 (areas outside)". We believe that, with the help of global and local contextual information, our model is able to capture the short as well as long range dependencies.

We conducted an in-depth analysis of the BPE segmented units of rare words. It was observed that the sub-word units could be categorized into three groups. The first group of units involve the phonetic Romanization (Pinyin) of Chinese. In translation, these are simply transliterated into the corresponding Chinese characters. As shown in the second row of Table 5, "Liu/jing/min" is a person's name. The segmented units are the phonetic representations. Both models can successfully transliterate this into the Chinese equivalent, "刘/敬/民". The second group of sub-

Source	Reference	Hierarchical	Sequential
liu/jing/min	刘/敬/民 Liú/jìng/mín	刘/敬/民 Liú/jìng/mín	刘/敬/民 Liú/jìng/mín
adventur/er	探险家 Tàn xiǎn jiā	探险家 Tàn xiǎn jiā	探险者 Tàn xiǎn zhě
hi/k/ed	上调 Shàng tiáo	上升 Shàng shēng	发生 Fā shēng

Table 5: Translation examples of sub-words, where '/' indicates a separation between sub-word units. The first two columns show the segmented words and their Chinese references. The last two columns report the translations given by the hierarchical and sequential models respectively.

word units are likely to represent the word morphemes. The words are segmented into sub-word units, which are to some extent close to the linguistic word stems and suffixes. For example, the word "adventurer" is segmented into "adventur/er", which is correctly translated into the Chinese translations "探险/家" and "探险/者" respectively by the hierarchical and sequential models, while the third group of sub-word units offer no linguistic interpretation. It is easy to see, using the BPE algorithm, that the identification of sub-word units is merely based on their frequency in the training data, with the result that not all units are well-formed linguistic morphemes. However, an interesting finding arises regarding the translation of these segmented units. In the sequential model, the word is incorrectly translated; however, it can be correctly translated by the hierarchical model. Taking "hi/k/ed" as an example, the sequential model gives an incorrect translation "发生(happened)", while the hierarchical model translates it into "上升(rise)" which is a synonym of "hiked". This result indicates that in our hierarchical model, the parent node of hierarchical representation for sub-word units "hi/k/ed" is better able to capture the meaning of the word as a whole; this cannot be captured independently by the sequential model.

6 Conclusion

In this paper, we propose an improved NMT system with a novel bidirectional hierarchical encoder, which enhances the source-side representations of a sentence, that is, both phrases and words, with local and global context information. By introducing a tree-based rare word encoding, the hierarchical model is extended to sub-word level in order to alleviate the problem of OOVs. To effectively leverage the enhanced hierarchical representations, we also propose a weighted variant of the attention model which dynamically adjusts the proportion of conditional information between the lexical and phrase annotation vectors. Experimental results for NIST English-Chinese translation tasks demonstrate that the proposed model significantly outperforms the vanilla tree-based and sequential NMT models.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant No. 61672555), a Multiyear Research Grant from the University of Macau (Grant Nos. MYRG2017-00087-FST, MYRG2015-00175-FST and MYRG2015-00188-FST) and the Science and Technology Development Fund of Macau (Grant No. 057/2014/A). The work of Tong Xiao and Jingbo Zhu was supported in part by the National Natural Science Foundation of China (Grant Nos. 61672138 and 61432013), the Fundamental Research Funds for the Central Universities, and the Opening Project of Beijing Key Laboratory of Internet Culture and Digital Dissemination Research.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017. Doubly-Attentive Decoder for Multi-modal Neural Machine Translation. *CoRR*, abs/1702.01287.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *Advances in neural information processing systems (NIPS 2014) Deep Learning and Representation Learning Workshop*.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-Sequence Attentional Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 823–833.

Philip Gage. 1994. A New Algorithm for Data Compression. *The C Users Journal*, 12(2):23–38.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12(10):2451–2471.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An Empirical Exploration of Recurrent Network Architectures. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2342–2350.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Filippos Kokkinos and Alexandros Potamianos. 2017. Structural Attention Neural Networks for Improved Sentiment Analysis. *CoRR*, abs/1701.01811.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Rico Sennrich and Barry Haddow. 2016. Linguistic Input Features Improve Neural Machine Translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–90.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016. Syntactically Guided Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 299–305.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in neural information processing systems (NIPS 2014)*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 1556–1566.

Tong Xiao, Jingbo Zhu, Hao Zhang, and Qiang Li. 2012. NiuTrans: An Open Source Toolkit for Phrase-based and Syntax-based Machine Translation. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics, System Demonstrations*, pages 19–24.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 77–81.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701.

Xiaodong Zeng, Derek F. Wong, Lidia S. Chao, and Isabel Trancoso. 2015. Graph-Based Lexicon Regularization for PCFG With Latent Annotations. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):441–450.

Xiaodong Zeng, Derek F. Wong, Lidia S. Chao, Isabel Trancoso, Liangye He, and Qiuping Huang. 2014. Lexicon Expansion for Latent Variable Grammars. *Pattern Recognition Letters*, (42):47–55.

Yao Zhou, Cong Liu, and Yan Pan. 2016. Modelling Sentence Pairs with Tree-structured Attentive Encoder. In *Proceedings of 26th International Conference on Computational Linguistics*, pages 2912–2922.

Massive Exploration of Neural Machine Translation Architectures

Denny Britz[*][†]**, Anna Goldie**[*]**, Minh-Thang Luong, Quoc V. Le**
{dennybritz,agoldie,thangluong,qvl}@google.com
Google Brain

Abstract

Neural Machine Translation (NMT) has shown remarkable progress over the past few years, with production systems now being deployed to end-users. As the field is moving rapidly, it has become unclear which elements of NMT architectures have a significant impact on translation quality. In this work, we present a large-scale analysis of the sensitivity of NMT architectures to common hyperparameters. We report empirical results and variance numbers for several hundred experimental runs, corresponding to over 250,000 GPU hours on a WMT English to German translation task. Our experiments provide practical insights into the relative importance of factors such as embedding size, network depth, RNN cell type, residual connections, attention mechanism, and decoding heuristics. As part of this contribution, we also release an open-source NMT framework in TensorFlow to make it easy for others to reproduce our results and perform their own experiments.

1 Introduction

Neural Machine Translation (NMT) (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014) is an end-to-end approach to machine translation. NMT has shown impressive results (Jean et al., 2015; Luong et al., 2015b; Sennrich et al., 2016a; Wu et al., 2016) surpassing those of phrase-based systems while addressing shortcomings, such as the need for hand-engineered features. The most popular approaches to NMT are based on sequence-to-sequence models, an encoder-decoder architecture consisting of two recurrent neural networks (RNNs) and an attention mechanism that aligns target with source tokens (Bahdanau et al., 2015; Luong et al., 2015a).

One drawback of current NMT architectures is the huge amount of compute required to train them. Training on real-world datasets of several million examples typically requires dozens of GPUs and convergence time is on the order of days to weeks (Wu et al., 2016). While sweeping across large hyperparameter spaces is common in Computer Vision (Huang et al., 2016b), such exploration would be prohibitively expensive for NMT models, limiting researchers to well-established architecture and hyperparameter choices. Furthermore, there have been no large-scale studies of how these hyperparameters affect the performance of NMT systems. As a result, it remains unclear why these models perform as well as they do or how we might improve them.

In this work, we present an extensive analysis of architectural hyperparameters for NMT systems. Using a total of more than 250,000 GPU hours, we explore common variations of NMT architectures and provide insights into which architectural choices matter most. We report BLEU scores, perplexities, model sizes, and convergence time for all experiments, including variance numbers calculated across several runs of each experiment. In addition, we release the software framework that we wrote to facilitate this exploration.

In summary, the main contributions of this work are as follows:

- We provide immediately applicable insights into the optimization of NMT models, as well as promising directions for future research. For example, we found that deep encoders are

[*] Both authors contributed equally to this work.

[†] Work done as a member of the Google Brain Residency program (g.co/brainresidency).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1442–1451
Copenhagen, Denmark, September 7–11, 2017.

more difficult to optimize than decoders, that dense residual connections yield better performance than regular residual connections, and that a well-tuned beam search is surprisingly critical to obtaining state-of-the-art results. By presenting practical advice for choosing baseline architectures, we help researchers avoid wasting time on unpromising model variations.

- We also establish the extent to which metrics such as BLEU are influenced by random initialization and slight hyperparameter variation, allowing researchers to better distinguish statistically significant results from noise.

- Finally, we release an open-source TensorFlow package, specifically designed to implement reproducible state-of-the-art sequence-to-sequence models. All experiments were run using this framework and we include all configuration files and processing scripts needed to reproduce the experiments in this paper. We hope to accelerate future research by releasing this framework to the public.

2 Background and Preliminaries

2.1 Neural Machine Translation

Our models are based on an encoder-decoder architecture with attention mechanism (Bahdanau et al., 2015; Luong et al., 2015a), as shown in figure 1. An encoder function f_{enc} takes as input a sequence of source tokens $\mathbf{x} = (x_1, ..., x_m)$ and produces a sequence of states $\mathbf{h} = (h_1, ..., h_m)$. In our base model, f_{enc} is a bi-directional RNN and the state h_i corresponds to the concatenation of the states produced by the backward and forward RNNs, $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$. The decoder f_{dec} is an RNN that predicts the probability of a target sequence $\mathbf{y} = (y_1, ..., y_k)$ based on $\mathbf{h}$. The probability of each target token $y_i \in 1, ...V$ is predicted based on the recurrent state in the decoder RNN s_i, the previous words, $y_{<i}$, and a context vector c_i. The context vector c_i is also called the attention vector and is calculated as a weighted average of the source states.

$$c_i = \sum_j a_{ij} h_j \tag{1}$$

$$a_{ij} = \frac{\hat{a}_{ij}}{\sum_j \hat{a}_{ij}} \tag{2}$$

$$\hat{a}_{ij} = att(s_i, h_j) \tag{3}$$

Here, $att(s_i, h_j)$ is an attention function that calculates an unnormalized alignment score between the encoder state h_j and the decoder state s_i. In our base model, we use a function of the form $att(s_i, h_j) = \langle W_h h_j, W_s s_i \rangle$, where the matrices W are used to transform the source and target states into a representation of the same size.

The decoder outputs a distribution over a vocabulary of fixed-size V:

$$\begin{aligned} &P(y_i | y_1, ..., y_{i-1}, \mathbf{x}) \\ &= \text{softmax}(W[s_i; c_i] + b) \end{aligned}$$

The whole model is trained end-to-end by minimizing the negative log likelihood of the target words using stochastic gradient descent.

3 Experimental Protocols

3.1 Datasets and Preprocessing

We run all experiments on the WMT'15 English→German task consisting of 4.5M sentence pairs, obtained by combining the Europarl v7, News Commentary v10, and Common Crawl corpora. We use newstest2013 as our validation set and newstest2014 and newstest2015 as our test sets. We focus on WMT English→German because it is a morphologically rich language therefore has been a standard benchmark in previous important work in Neural Machine Translation (Jean et al., 2015; Luong et al., 2015a; Sennrich et al., 2016b; Zhou et al., 2016; Wu et al., 2016)

To test for generality, we also ran a small number of experiments on English→French translation, and we found that the performance was highly correlated with that of English→German but that it took much longer to train models on the larger English→French dataset. Given that translation from the morphologically richer German is also considered a more challenging task, we felt justified in using the English→German translation task for this hyperparameter sweep.

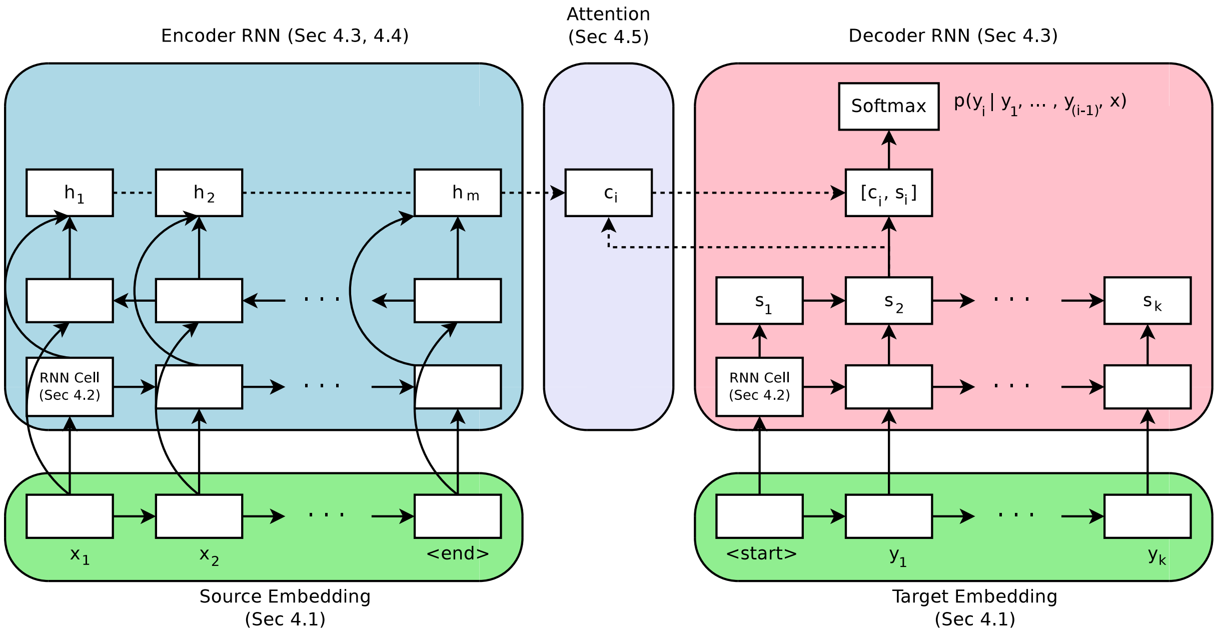


Figure 1: Encoder-Decoder architecture with attention module. Section numbers reference experiments corresponding to the components.

We tokenize and clean all datasets with the scripts in Moses[1] and learn shared subword units using Byte Pair Encoding (BPE) (Sennrich et al., 2016b) using 32,000 merge operations for a final vocabulary size of approximately 37k. We discovered that data preprocessing can have a large impact on final numbers, and since we wish to enable reproducibility, we release our data preprocessing scripts together with the NMT framework to the public. For more details on data preprocessing parameters, we refer the reader to the code release.

3.2 Training Setup and Software

All of the following experiments are carried out using our own implementation based on TensorFlow (Abadi et al., 2016). We built this framework to enable reproducible state-of-the-art implementations of Neural Machine Translation architectures. As part of our contribution, we are releasing the framework and all configuration files needed to reproduce our results. Training is performed on Nvidia Tesla K40m and Tesla K80 GPUs, distributed over 8 parallel workers and 6 parameter servers per experiment. We use a batch size of 128 and decode using beam search with a beam width of 10 and the length normalization penalty of 0.6 described in (Wu et al., 2016). BLEU scores are calculated on tokenized data using the *multi-bleu.perl* script in Moses.[2] Each experiment is run for a maximum of 2.5M steps and replicated 4 times with different initializations. We save model checkpoints every 30 minutes and choose the best checkpoint based on the validation set BLEU score. We report *mean and standard deviation* as well as *highest scores* (as per cross validation) for each experiment.

3.3 Baseline Model

Based on a review of recent literature, we chose a baseline model that we knew would perform reasonably well. Our goal was to keep the baseline model simple and standard, not to advance the start of the art. The model (described in 2.1) consists of a 2-layer bidirectional encoder (1 layer in each direction), and a 2 layer decoder with a multiplicative (Luong et al., 2015a) attention mechanism. We use 512-unit GRU (Cho et al., 2014) cells for both the encoder and decoder and apply Dropout of 0.2 at the input of each cell. We train using the Adam optimizer and a fixed learning rate of 0.0001 without decay. The embedding dimensionality is set to 512. A more detailed description of all model hyperparameters can be found in the supplementary material.

In each of the following experiments, the hy-

[1] https://github.com/moses-smt/mosesdecoder/

[2] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

perparameters of the baseline model are held constant, except for the one hyperparameter being studied. We hope that this allows us to isolate the effect of various hyperparameter changes. We recognize that this procedure does not account for interactions between hyperparameters, and we perform additional experiments when we believe such interactions are likely to occur (e.g., skip connections and number of layers).

4 Experiments and Results

For the sake of brevity, we only report mean BLEU, standard deviation, highest BLEU in parentheses, and model size in the following tables. Log perplexity, tokens/sec and convergence times can be found in the supplementary material tables. All reported p-values were calculated with a two-sample t-test that assumed equal variances.

4.1 Embedding Dimensionality

With a large vocabulary, the embedding layer may account for a significant fraction of the model parameters. Historically, researchers have used 620-dimensional (Bahdanau et al., 2015) or 1024-dimensional (Luong et al., 2015a) embeddings. We expected larger embeddings to result in better BLEU scores, or at least lower perplexities, but this wasn't always the case. While table 1 shows that 2048-dimensional embeddings yielded the overall best result, they only outperformed the smallest 128-dimensional embeddings by a narrow yet statistically significant margin (p = 0.01), but took nearly twice as long to converge. Gradient updates to both small and large embeddings did not differ significantly from each other and the norm of gradient updates to the embedding matrix stayed approximately constant throughout training, regardless of size. We did not observe overfitting with large embeddings and training log perplexity was almost equal across experiments, suggesting that the model does not make efficient use of the extra parameters and that there may be a need for better optimization techniques. Alternatively, it could be the case that models with large embeddings simply need far more than 2.5M steps to converge to the best solution.

Dim	newstest2013	Params
128	21.50 ± 0.16 (21.66)	36.13M
256	21.73 ± 0.09 (21.85)	46.20M
512	21.78 ± 0.05 (21.83)	66.32M
1024	21.36 ± 0.27 (21.67)	106.58M
2048	**21.86** ± 0.17 (22.08)	187.09M

Table 1: BLEU scores on newstest2013, varying the embedding dimensionality.

4.2 RNN Cell Variant

Both LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014) cells are commonly used in NMT architectures. While there exist studies (Greff et al., 2016) that explore cell variants on small sequence tasks of a few thousand examples, we are not aware of any such studies in large-scale NMT settings.

The vanishing gradient problem is a motivation for gated cells, such as the GRU and LSTM. Using vanilla RNN cells, deep networks cannot efficiently propagate information and gradients through multiple layers and time steps. We initialize the decoder state to zero instead of passing the encoder state, and we experiment with using a vanilla RNN cell in the decoder only (Vanilla-Dec below). For the LSTM and GRU variants, we vary cell types in both the encoder and decoder. We use LSTM cells without peephole connections and initialize the forget bias of both LSTM and GRU cells to 1.

Cell	newstest2013	Params
LSTM	**22.22** ± 0.08 (22.33)	68.95M
GRU	21.78 ± 0.05 (21.83)	66.32M
Vanilla-Dec	15.38 ± 0.28 (15.73)	63.18M

Table 2: BLEU scores on newstest2013, varying the type of encoder and decoder cell.

In our experiments, *LSTM cells consistently outperformed GRU cells*, a result which was statistically significant ($p < 0.00001$). Since the computational bottleneck in our architecture is the softmax operation, we did not observe large differences in training speed between LSTM and GRU cells. Somewhat to our surprise, we found that the vanilla decoder is unable to learn nearly as well as the gated variant. This suggests that the decoder indeed passes information in its own state through multiple time steps instead of relying solely on the attention mechanism and current input (which includes the previous attention context). It could also be the case that the gating mechanism is necessary to mask out irrelevant parts of the input.

4.3 Encoder and Decoder Depth

We generally expect deeper networks to converge to better solutions than shallower ones (He et al., 2016). While some work (Luong et al., 2015b; Zhou et al., 2016; Luong and Manning, 2016; Wu et al., 2016) has achieved state-of-the-art results using deep networks, others (Jean et al., 2015; Chung et al., 2016; Sennrich et al., 2016b) have produced similar results with far shallower ones. Therefore, it is unclear how important depth is, and whether shallow networks are capable of producing results competitive with those of deep networks. Here, we explore the effect of both encoder and decoder depth up to 8 layers. For the bidirectional encoder, we separately stack the RNNs in both directions. For example, the Enc-8 model corresponds to one forward and one backward 4-layer RNN. For deeper networks, we also experiment with two variants of residual connections (He et al., 2016; Srivastava et al., 2015) to encourage gradient flow. In the standard variant, shown in equation (4), we insert residual connections between consecutive layers. If $h_t^{(l)}(x_t^{(l)}, h_{t-1}^{(l)})$ is the RNN output of layer l at time step t, then:

$$x_t^{(l+1)} = h_t^{(l)}(x_t^{(l)}, h_{t-1}^{(l)}) + x_t^{(l)} \tag{4}$$

where $x_t^{(0)}$ are the embedded input tokens. We also explore a dense (”ResD” below) variant of residual connections similar to those used by (Huang et al., 2016a) in Image Recognition. In this variant, we add skip connections from each layer to all other layers:

$$x_t^{(l+1)} = h_t^{(l)}(x_t^{(l)}, h_{t-1}^{(l)}) + \sum_{j=0}^{l} x_t^{(j)} \tag{5}$$

Our implementation differs from (Huang et al., 2016a) in that we use addition instead of concatenation in order to keep the state size constant.

Table 3 shows results of varying encoder and decoder depth with and without residual connection. We found no benefit to increasing encoder depth beyond two layers, as we observed no statistically significant improvement from going to four layers and even deeper models generally diverged during training. The best deep residual models achieved good results, but only one of four runs converged, as suggested by the large standard deviation.

Depth	newstest2013	Params
Enc-2	21.78 ± 0.05 (21.83)	66.32M
Enc-4	**21.85** ± 0.32 (22.23)	69.47M
Enc-8	21.32 ± 0.14 (21.51)	75.77M
Enc-8-Res	19.23 ± 1.96 (21.97)	75.77M
Enc-8-ResD	17.30 ± 2.64 (21.03)	75.77M
Dec-1	21.76 ± 0.12 (21.93)	64.75M
Dec-2	21.78 ± 0.05 (21.83)	66.32M
Dec-4	**22.37** ± 0.10 (22.51)	69.47M
Dec-4-Res	17.48 ± 0.25 (17.82)	68.69M
Dec-4-ResD	21.10 ± 0.24 (21.43)	68.69M
Dec-8	01.42 ± 0.23 (1.66)	75.77M
Dec-8-Res	16.99 ± 0.42 (17.47)	75.77M
Dec-8-ResD	20.97 ± 0.34 (21.42)	75.77M

Table 3: BLEU scores on newstest2013, varying the encoder and decoder depth and type of residual connections.

On the decoder side, deeper models outperformed shallower ones by a small but statistically significant margin ($p < 0.00001$), but without residual connections, we were unable to train decoders with 8 or more layers. Across the deep decoder experiments, dense residual connections consistently outperformed regular residual connections ($p < 0.00001$) and converged much faster in terms of step count, as shown in figure 2. We expected deep models to perform better (Sutskever et al., 2014; Zhou et al., 2016; Wu et al., 2016) across the board, and we believe that our experiments demonstrate the need for more robust techniques for optimizing deep sequential models. For example, we may need a better-tuned SGD optimizer or some form of batch normalization, in order to robustly train deep networks with residual connections.

4.4 Unidirectional vs. Bidirectional Encoder

In the literature, we see bidirectional encoders (Bahdanau et al., 2015), unidirectional encoders (Luong et al., 2015a), and a mix of both (Wu et al., 2016) being used. Bidirectional encoders are able to create representations that take into account both past and future inputs, while unidirectional encoders can only take past inputs into account. The benefit of unidirectional encoders is that their computation can be easily parallelized on GPUs, allowing them to run faster than their bidirectional counterparts. We are not aware of any studies that explore the necessity of bidirectional-

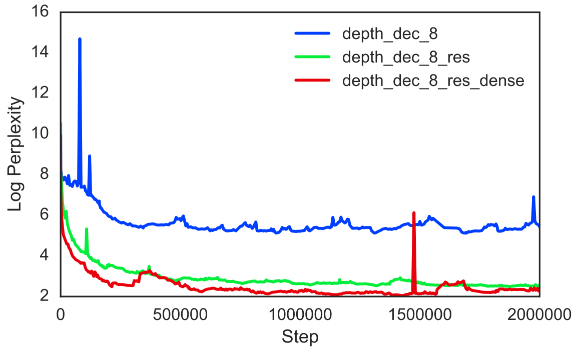


Figure 2: Training plots for deep decoder with and without residual connections, showing log perplexity on the eval set.

ity. In this set of experiments, we explore unidirectional encoders of varying depth with and without reversed source inputs, as this is a commonly used trick that allows the encoder to create richer representations for earlier words. Given that errors on the decoder side can easily cascade, the correctness of early words has disproportionate impact.

Cell	newstest2013	Params
Bidi-2	**21.78** ± 0.05 (21.83)	66.32M
Uni-1	20.54 ± 0.16 (20.73)	63.44M
Uni-1R	21.16 ± 0.35 (21.64)	63.44M
Uni-2	20.98 ± 0.10 (21.07)	65.01M
Uni-2R	21.76 ± 0.21 (21.93)	65.01M
Uni-4	21.47 ± 0.22 (21.70)	68.16M
Uni-4R	21.32 ± 0.42 (21.89)	68.16M

Table 4: BLEU scores on newstest2013, varying the type of encoder. The ”R” suffix indicates a reversed source sequence.

Table 4 shows that bidirectional encoders generally outperform unidirectional encoders, but not by a statistically significant margin. The encoders with reversed source consistently outperform their non-reversed counterparts (p = 0.009 for one layer models, p = 0.0003 for two layers, p = 0.2751 for four layers), but do not beat shallower bidirectional encoders.

4.5 Attention Mechanism

The two most commonly used attention mechanisms are the additive (Bahdanau et al., 2015) variant, equation (6) below, and the computationally less expensive multiplicative variant (Luong et al., 2015a), equation (7) below. Given an attention key h_j (an encoder state) and attention query s_i (a decoder state), the attention score for each pair is calculated as follows:

$$\text{score}(h_j, s_i) = \langle v, tanh(W_1 h_j + W_2 s_i)\rangle \quad (6)$$

$$\text{score}(h_j, s_i) = \langle W_1 h_j, W_2 s_i\rangle \quad (7)$$

We call the dimensionality of $W_1 h_j$ and $W_2 s_i$ the ”attention dimensionality” and vary it from 128 to 1024 by changing the layer size. We also experiment with using no attention mechanism by initializing the decoder state with the last encoder state (None-State), or concatenating the last encoder state to each decoder input (None-Input). The results are shown in table 5.

Attention	newstest2013	Params
Mul-128	22.03 ± 0.08 (22.14)	65.73M
Mul-256	22.33 ± 0.28 (22.64)	65.93M
Mul-512	21.78 ± 0.05 (21.83)	66.32M
Mul-1024	18.22 ± 0.03 (18.26)	67.11M
Add-128	22.23 ± 0.11 (22.38)	65.73M
Add-256	22.33 ± 0.04 (22.39)	65.93M
Add-512	**22.47** ± 0.27 (22.79)	66.33M
Add-1024	22.10 ± 0.18 (22.36)	67.11M
None-State	9.98 ± 0.28 (10.25)	64.23M
None-Input	11.57 ± 0.30 (11.85)	64.49M

Table 5: BLEU scores on newstest2013, varying the type of attention mechanism.

We found that the parameterized additive attention mechanism slightly but consistently outperformed the multiplicative one (p = 0.013 for 128 units, p = 0.5 for 256 units, p = 0.0012 for 512 units, and p < 0.00001 for 1024/8 units), with the attention dimensionality having little effect.

While we did expect the attention-based models to significantly outperform those without an attention mechanism, we were surprised by just how poorly the ”Non-Input” models fared, given that they had access to encoder information at each time step. Furthermore, we found that the attention-based models exhibited significantly larger gradient updates to decoder states throughout training. This suggests that the attention mechanism acts more like a ”weighted skip connection” that optimizes gradient flow than like a ”memory” that allows the encoder to access source states, as is commonly stated in the literature. We believe that further research in this direction is necessary to shed light on the role of the attention mechanism and whether it may be purely a vehicle for easier optimization.

4.6 Beam Search Strategies

Beam Search is a commonly used technique to find target sequences that maximize some scoring function $s(\mathbf{y}, \mathbf{x})$ through tree search. In the simplest case, the score to be maximized is the log probability of the target sequence given the source. Recently, extensions such as coverage penalties (Tu et al., 2016) and length normalizations (Wu et al., 2016) have been shown to improve decoding results. It has also been observed (Tu et al., 2017) that very large beam sizes, even with length penalty, perform worse than smaller ones. Thus, choosing the correct beam width can be crucial to achieving the best results.

Beam	**newstest2013**	**Params**
B1	20.66 ± 0.31 (21.08)	66.32M
B3	21.55 ± 0.26 (21.94)	66.32M
B5	21.60 ± 0.28 (22.03)	66.32M
B10	21.57 ± 0.26 (21.91)	66.32M
B25	21.47 ± 0.30 (21.77)	66.32M
B100	21.10 ± 0.31 (21.39)	66.32M
B10-LP-0.5	21.71 ± 0.25 (22.04)	66.32M
B10-LP-1.0	**21.80** ± 0.25 (22.16)	66.32M

Table 6: BLEU scores on newstest2013, varying the beam width and adding length penalties (LP).

Table 6 shows the effect of varying beam widths and adding length normalization penalties. A beam width of 1 corresponds to greedy search. We found that a well-tuned beam search is crucial to achieving good results, and that it leads to consistent gains of more than one BLEU point. Similar to (Tu et al., 2017) we found that very large beams yield worse results and that there is a ”sweet spot” of optimal beam width. We believe that further research into the robustness of hyperparameters in beam search is crucial to progress in NMT. We also experimented with a coverage penalty, but found no additional gain over a sufficiently large length penalty.

4.7 Final System Comparison

Finally, we compare our best performing model across all experiments, as chosen on the newstest2013 validation set, to historical results found in the literature in Table 8. Interestingly, the best performing model turned out to be nearly equivalent to the base model (described in Section 3.3), differing only in that it used 512-dimensional additive attention. While not the focus on this work, we were able to achieve further improvements by combining all of our insights into a single model described in Table 7.

Hyperparameter	**Value**
embedding dim	512
rnn cell variant	LSTMCell
encoder depth	4
decoder depth	4
attention dim	512
attention type	Bahdanau
encoder	bidirectional
beam size	10
length penalty	1.0

Table 7: Hyperparameter settings for our final combined model, consisting of all of the individually optimized values.

Model	**newstest14**	**newstest15**
Ours (best performing)	22.03	24.75
Ours (combined)	22.19	25.23
OpenNMT	19.34	-
Luong	20.9	-
BPE-Char	21.5	23.9
BPE	-	20.5
RNNSearch-LV	19.4	-
RNNSearch	-	16.5
Deep-Att*	20.6	-
GNMT*	24.61	-
Deep-Conv*	-	24.3

Table 8: Comparison to RNNSearch (Jean et al., 2015), RNNSearch-LV (Jean et al., 2015), BPE (Sennrich et al., 2016b), BPE-Char (Chung et al., 2016), Deep-Att (Zhou et al., 2016), Luong (Luong et al., 2015a), Deep-Conv (Gehring et al., 2016), GNMT (Wu et al., 2016), and OpenNMT (Klein et al., 2017). Systems with an * do not have a public implementation.

Although we do not offer architectural innovations, we do show that through careful hyperparameter tuning and good initialization, it is possible to achieve state-of-the-art performance on standard WMT benchmarks. Our model is outperformed only by (Wu et al., 2016), a model which is significantly more complex and lacks a public implementation.

To test whether our findings generalize to other languages, we also trained a model with the same hyperparameter configurations on the ASPEC Japanese to English translation task and achieved a BLEU score of 38.87, which is state-of-the-art.

5 Open-Source Release

We demonstrated empirically how small changes to hyperparameter values and different initialization can affect results, and how factors such as a well-tuned beam search are critical to high quality translation results. To move towards reproducible research, we believe it is important that researchers start building upon common frameworks and data processing pipelines. With this goal in mind, we built a modular software framework that allows researchers to explore novel architectures with minimal code changes, and define experimental parameters in a reproducible manner. While our initial experiments are in Machine Translation, our framework can easily be adapted to problems in Summarization (e.g., Nallapati et al. (2016)), Conversational Modeling (e.g., Vinyals and Le (2015); Shang et al. (2015); Sordoni et al. (2015); Li et al. (2015)) or Image-To-Text (e.g., Vinyals et al. (2015); Karpathy and Fei-Fei (2015); Xu et al. (2015)).

Although there exist open-source libraries such as OpenNMT (Klein et al., 2017) that share similar goals, they have not yet achieved state-of-the-art results (see table 8) and lack some important features, such as support for distributed training. We hope that by open sourcing our experimental toolkit, we can help to accelerate research in neural machine translation and sequence-to-sequence modeling.

6 Conclusion

We conducted a large-scale empirical analysis of architecture variations for Neural Machine Translation, teasing apart the key factors to achieving state-of-the-art results. We demonstrated a number of surprising insights, including the fact that beam search tuning is just as crucial as most architectural variations, and that with current optimization techniques deep models do not always outperform shallow ones. Here, we summarize our practical findings:

- Large embeddings with 2048 dimensions achieved the best results, but only by a small margin. Even small embeddings with 128 dimensions seem to have sufficient capacity to capture most of the necessary semantic information.

- LSTM Cells consistently outperformed GRU Cells.

- Bidirectional encoders with 2 to 4 layers performed best. Deeper encoders were significantly more likely to diverge, but show potential if they can be optimized well.

- Deep 4-layer decoders slightly outperformed shallower decoders. Residual connections were necessary to train decoders with 8 layers and dense residual connections offer additional robustness.

- Parameterized additive attention yielded the overall best results.

- A well-tuned beam search with length penalty is crucial. Beam widths of 5 to 10 along with a length penalty of 1.0 seemed to work well.

We highlighted several important research questions, including the efficient use of embedding parameters (4.1), the role of attention mechanisms as weighted skip connections (4.5) as opposed to memory units, the need for better optimization methods for deep recurrent networks (4.3), and the need for a better beam search (4.6) robust to hyperparameter variations.

Finally, given the recent surge in new applications for sequence-to-sequence models, we believe our new findings and state-of-the-art open-source package can significantly accelerate the pace of research in this domain.

References

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A system for large-scale machine learning. In *OSDI*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *ACL*.

Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. 2016. A convolutional encoder model for neural machine translation. *CoRR* abs/1611.02344.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2016. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems* PP(99):1–11.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

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

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. 2016a. Densely connected convolutional networks. *CoRR* abs/1608.06993.

Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. 2016b. Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR* abs/1611.10012.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *ACL*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. *CoRR* abs/1701.02810.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* .

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *ACL*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *ACL*.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for wmt 16. In *ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *ACL*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364* .

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714* .

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387* .

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *AAAI*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *ACL*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* .

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus

Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

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

Learning Translations via Matrix Completion

Derry Wijaya[1], **Brendan Callahan**[1], **John Hewitt**[1], **Jie Gao**[1], **Xiao Ling**[1],
Marianna Apidianaki[12] and **Chris Callison-Burch**[1]
[1]Computer and Information Science Department, University of Pennsylvania
[2]LIMSI, CNRS, Université Paris-Saclay, 91403 Orsay
derry@seas.upenn.edu

Abstract

Bilingual Lexicon Induction is the task of learning word translations without bilingual parallel corpora. We model this task as a matrix completion problem, and present an effective and extendable framework for completing the matrix. This method harnesses diverse bilingual and monolingual signals, each of which may be incomplete or noisy. Our model achieves state-of-the-art performance for both high and low resource languages.

1 Introduction

Machine translation (MT) models typically require large, sentence-aligned bilingual texts to learn good translation models (Wu et al., 2016; Sennrich et al., 2016a; Koehn et al., 2003). However, for many language pairs, such parallel texts may only be available in limited quantities, which is problematic. Alignments at the word- or subword- levels (Sennrich et al., 2016b) can be inaccurate in the limited parallel texts, which can in turn lead to inaccurate translations. Due to the low quantity and thus coverage of the texts, there may still be "out-of-vocabulary" words encountered at run-time. The Bilingual Lexicon Induction (BLI) task (Rapp, 1995), which learns word translations from monolingual or comparable corpora, is an attempt to alleviate this problem. The goal is to use plentiful, more easily obtainable, monolingual or comparable data to infer word translations and reduce the need for parallel data to learn good translation models. The word translations obtained by BLI can, for example, be used to augment MT systems and improve alignment accuracy, coverage, and translation quality (Gulcehre et al., 2016; Callison-Burch et al., 2006; Daumé and Jagarlamudi, 2011).

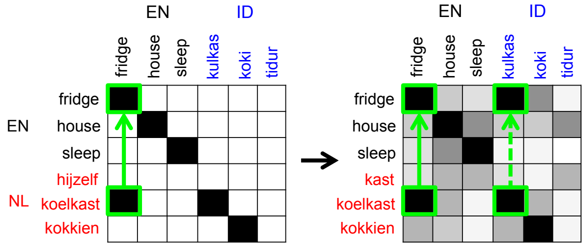


Figure 1: Our framework allows us to use a diverse range of signals to learn translations, including incomplete bilingual dictionaries, information from related languages (like Indonesian loan words from Dutch shown here), word embeddings, and even visual similarity cues.

Previous research has explored different sources for estimating translation equivalence from monolingual corpora (Schafer and Yarowsky, 2002; Klementiev and Roth, 2006; Irvine and Callison-Burch, 2013, 2017). These monolingual signals, when combined in a supervised model, can enhance end-to-end MT for low resource languages (Klementiev et al., 2012a; Irvine and Callison-Burch, 2016). More recently, similarities between words in different languages have been approximated by constructing a shared bilingual word embedding space with different forms of bilingual supervision (Upadhyay et al., 2016).

We present a framework for learning translations by combining diverse signals of translation that are each potentially sparse or noisy. We use matrix factorization (MF), which has been shown to be effective for harnessing incomplete or noisy distant supervision from multiple sources of information (Fan et al., 2014; Rocktäschel et al., 2015). MF is also shown to result in good cross-lingual representations for tasks such as alignment (Goutte et al., 2004), QA (Zhou et al., 2013), and cross-lingual word embeddings (Shi et al., 2015).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1452–1463
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Specifically, we represent translation as a matrix with source words in the columns and target words in the rows, and model the task of learning translations as a matrix completion problem. Starting from some observed translations (e.g., from existing bilingual dictionaries,) we infer missing translations in the matrix using MF with a Bayesian Personalized Ranking (BPR) objective (Rendle et al., 2009). We select BPR for a number of reasons: (1) BPR has been shown to outperform traditional supervised methods in the presence of positive-only data (Riedel et al., 2013), which is true in our case since we only observe positive translations. (2) BPR is easily extendable to incorporate additional signals for inferring missing values in the matrix (He and McAuley, 2016). Since observed translations may be sparse, i.e. the "cold start" problem in the matrix completion task, incorporating additional signals of translation equivalence estimated on monolingual corpora is useful. (3) BPR is also shown to be effective for multilingual transfer learning (Verga et al., 2016). For low resource source languages, there may be related, higher resource languages from which we can project available translations (e.g., translations of loan words) to the target language (Figure 1).

We conduct large scale experiments to learn translations from both low and high resource languages to English and achieve state-of-the-art performance on these languages. Our main contributions are as follows:

- We introduce a MF framework that learns translations by integrating diverse bilingual and monolingual signals of translation, each potentially noisy/incomplete.
- The framework is easily extendable to incorporate additional signals of translation equivalence. Since ours is a framework for integration, each signal can be improved separately to improve the overall system.
- Large scale experiments on both low and high resource languages show the effectiveness of our model, outperforming the current state-of-the-art.
- We make our code, datasets, and output translations publicly available.[1]

2 Related Work

Bilingual Lexicon Induction Previous research has used different sources for estimating translations from monolingual corpora. Signals such as contextual, temporal, topical, and ortographic similarities between words are used to measure their translation equivalence (Schafer and Yarowsky, 2002; Klementiev and Roth, 2006; Irvine and Callison-Burch, 2013, 2017).

With the increasing popularity of word embeddings, many recent works approximate similarities between words in different languages by constructing a shared bilingual embedding space (Klementiev et al., 2012b; Zou et al., 2013; Vulić and Moens, 2013; Mikolov et al., 2013a; Faruqui and Dyer, 2014; Chandar A P et al., 2014; Gouws et al., 2015; Luong et al., 2015; Lu et al., 2015; Upadhyay et al., 2016). In the shared space, words from different languages are represented in a language-independent manner such that similar words, regardless of language, have similar representations. Similarities between words can then be measured in the shared space. One approach to induce this shared space is to learn a mapping function between the languages' monolingual semantic spaces (Mikolov et al., 2013a; Dinu et al., 2014). The mapping relies on seed translations which can be from existing dictionaries or be reliably chosen from pseudo-bilingual corpora of comparable texts e.g., Wikipedia with interlanguage links. Vulić and Moens (2015) show that by learning a linear function with a reliably chosen seed lexicon, they outperform other models with more expensive bilingual signals for training on benchmark data.

Most prior work on BLI however, either makes use of only one monolingual signal or uses unsupervised methods (e.g., rank combination) to aggregate the signals. Irvine and Callison-Burch (2016) show that combining monolingual signals in a supervised logistic regression model produces higher accuracy word translations than unsupervised models. More recently, Vulić et al. (2016) show that their multi-modal model that employs a simple weighted-sum of word embeddings and visual similarities can improve translation accuracy. These works show that there is a need for combining diverse, multi-modal monolingual signals of translations. In this paper, we take this step further by combining the monolingual signals with bilingual signals of translations from existing bilingual dictionaries of related, "third" languages.

Bayesian Personalized Ranking (BPR) Our approach is based on extensions to the probabilis-

[1] http://www.cis.upenn.edu/%7Ederry/translations.html

tic model of MF in collaborative filtering (Koren et al., 2009; Rendle et al., 2009). We represent our translation task as a matrix with source words in the columns and target words in the rows (Figure 1). Based on some observed translations in the matrix found in a seed dictionary, our model learns low-dimensional feature vectors that encode the latent properties of the words in the row and the words in the column. The dot product of these vectors, which indicate how "aligned" the source and the target word properties are, captures how likely they are to be translations.

Since we do not observe false translations in the seed dictionary, the training data in the matrix consists only of positive translations. The absence of values in the matrix does not imply that the corresponding words are not translations. In fact, we seek to predict which of these missing values are true. The BPR approach to MF (Rendle et al., 2009) formulates the task of predicting missing values as a ranking task. With the assumption that observed true translations should be given higher values than unobserved translations, BPR learns to optimize the difference between values assigned to the observed translations and values assigned to the unobserved translations.

However, due to the sparsity of existing bilingual dictionaries (for some language pairs such dictionaries may not exist), the traditional formulation of MF with BPR suffers from the "cold start" issue (Gantner et al., 2010; He and McAuley, 2016; Verga et al., 2016). In our case, these are situations in which some source words have no translations to any word in the target or related languages. For these words, additional information, e.g., monolingual signals of translation equivalence or language-independent representations such as visual representations, must be used.

We use bilingual translations from the source to the target language, English, obtained from Wikipedia page titles with interlanguage links. Since Wikipedia pages in the source language may be linked to pages in languages other than English, we also use high accuracy, crowdsourced translations (Pavlick et al., 2014) from these *third* languages to English as additional bilingual translations. To alleviate the cold start issue, when a source word has no existing known translation to English or other third languages, our model backs-off to additional signals of translation equivalence estimated based on its word embedding and visual representations.

3 Method

In this section, we describe our framework for integrating bilingual and monolingual signals for learning translations. First we formulate the task of Bilingual Lexicon Induction, and introduce our model for learning translations given observed translations and additional monolingual/language-independent signals. Then we derive our learning procedure using the BPR objective function.

Problem Formulation Given a set of source words F, a set of target words E, the pair $\langle e, f\rangle$ where $e \in E$ and $f \in F$ is a candidate translation with an associated score $x_{e,f} \in [0,1]$ indicating the confidence of the translation. The input to our model is a set of observed translations $T := \{\langle e, f\rangle \mid x_{e,f} = 1\}$. These could come from an incomplete bilingual dictionary. We also add word identities to the matrix i.e., we define $T^{identity} := \{\langle e, e\rangle\}$, where $T^{identity} \subset T$. The task of Bilingual Lexicon Induction is then to generate *missing* translations: for a given source word f and a set of target words $\{e \mid \langle e, f\rangle \notin T\}$, predict the score $x_{e,f}$ of how likely it is for e to be a translation of f.

Bilingual Signals for Translation One way to predict $x_{e,f}$ is by using matrix factorization. The problem of predicting $x_{e,f}$ can be seen as a task of estimating a matrix $X : E \times F$. X is approximated by a matrix product of two low-rank matrices $P : |E| \times k$ and $Q : |F| \times k$:

$$\hat{X} := PQ^T$$

where k is the rank of the approximation. Each row p_e in P can be seen as a feature vector describing the latent properties of the target word e, and each row q_f of Q describes the latent properties of the source word f. Their dot product encodes how aligned the latent properties are and, since these vectors are trained on observed translations, it encodes how likely they are to be translation of each other. Thus, we can write this formulation of predicted scores $\hat{x}_{e,f}$ with MF as:

$$\hat{x}^{\mathrm{MF}}_{e,f} = p_e^T q_f = \sum_{i=1}^{k} p_{ei} \cdot q_{fi} \qquad (1)$$

Auxiliary Signals for Translation Because the observed bilingual translations may be sparse, the

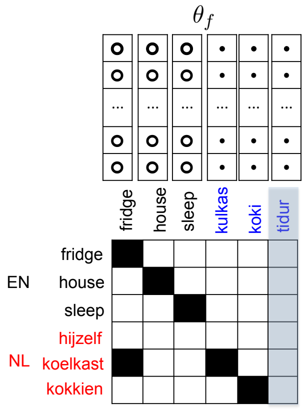


Figure 2: The word *tidur* (id) is a cold word with no associated translation in the matrix. Auxiliary features θ_f about the words can be used to predict translations for cold words.

MF approach can suffer from the existence of *cold* items: words that have none or too few associated observed translations to estimate their latent dimensions accurately (Figure 2). Additional signals for measuring translation equivalence can alleviate this problem. Hence, in the case of cold words, we use a formulation of $\hat{x}_{e,f}$ that involves auxiliary features about the words in the predicted $\hat{x}_{e,f}$:

$$\hat{x}^{\text{AUX}}_{e,f} = \theta_e^T \theta_f + \beta^T \theta_f \qquad (2)$$

θ_f represents an auxiliary information about the cold word f e.g., its word embedding or visual features. θ_e is a feature vector to be trained, whose dot product with θ_f models the extent to which the word e matches the auxiliary features of word f. In practice, learning θ_e amounts to learning a classifier, one for each target word e that learns weights θ_e given the feature vectors θ_f of its translations. β models the targets' overall bias toward a given word f.

Since each word can have multiple additional feature vectors, we can formulate $\hat{x}^{\text{AUX}}_{e,f}$ as a weighted sum of available auxiliary features[2]:

$$\hat{x}^{\text{AUX}}_{e,f} = \alpha_1\, \theta_e^T \theta_f + \alpha_2\, \gamma_e^T \gamma_f + ... + \alpha_n\, \delta_e^T \delta_f$$

where α_m are parameters assigned to control the contribution of each auxiliary feature.

In practice, we can combine the MF and auxiliary formulations by defining:

$$\hat{x}_{e,f} = \hat{x}^{\text{MF}}_{e,f} + \hat{x}^{\text{AUX}}_{e,f}$$

However, since bilingual signals that are input to $\hat{x}^{\text{MF}}_{e,f}$ are often precise but sparse, while monolingual signals that are input to $\hat{x}^{\text{AUX}}_{e,f}$ are often noisy and not sparse, in our model we only back-off to the less precise $\hat{x}^{\text{AUX}}_{e,f}$ for cold source words that have none or too few associated translations (more details are given in the experiments, Section 4). For other source words, we use $\hat{x}^{\text{MF}}_{e,f}$ to predict.

Learning with Bayesian Personalized Ranking Unlike traditional supervised models that try to maximize the scores assigned to positive instances (in our case, observed translations), the objective of Bayesian Personalized Ranking (BPR) is to maximize the *difference* in scores assigned to the observed translations compared to those assigned to the unobserved translations. Given a training set D consisting of triples of the form $\langle e, f, g\rangle$, where $\langle e, f\rangle \in T$ and $\langle e, g\rangle \notin T$, BPR wants to maximize $\hat{x}_{e,f,g}$, defined as:

$$\hat{x}_{e,f,g} = \hat{x}_{e,f} - \hat{x}_{e,g}$$

where $\hat{x}_{e,f}$ and $\hat{x}_{e,g}$ can be defined either by eq. 1 or eq. 2 (for cold words). Specifically, BPR optimizes (Rendle et al., 2009):

$$\sum_{\langle e,f,g\rangle \in D} \ln \sigma(\hat{x}_{e,f,g}) - \lambda_\Theta ||\Theta||^2$$

where σ is the logistic sigmoid function, Θ is the parameter vector of $\hat{x}_{e,f,g}$ to be trained, and λ_Θ is its hyperparameter vector. BPR can be trained using stochastic gradient ascent where a triple $\langle e, f, g\rangle$ is sampled from D and parameter updates are performed:

$$\Theta \leftarrow \Theta + \eta.(\sigma(-\hat{x}_{e,f,g})\frac{\partial \hat{x}_{e,f,g}}{\partial \Theta} - \lambda_\Theta \Theta)$$

η is the learning rate. Hence, for the MF formulation of $\hat{x}_{e,f,g}$, we can sample a triple $\langle e, f, g\rangle$ from D and update its parameters as:

$$p_e \leftarrow p_e + \eta.(\sigma(-\hat{x}^{\text{MF}}_{e,f,g})(q_f - q_g) - \lambda_P\, p_e)$$
$$q_f \leftarrow q_f + \eta.(\sigma(-\hat{x}^{\text{MF}}_{e,f,g})(p_e) - \lambda_{Q^+}\, q_f)$$
$$q_g \leftarrow q_g + \eta.(\sigma(-\hat{x}^{\text{MF}}_{e,f,g})(-p_e) - \lambda_{Q^-}\, q_g)$$

while for the auxiliary formulation of $\hat{x}_{e,f,g}$, we can sample a triple $\langle e, f, g\rangle$ from D and update its parameters as:

$$\theta_e \leftarrow \theta_e + \eta.(\sigma(-\hat{x}^{\text{AUX}}_{e,f,g})(\theta_f - \theta_g) - \lambda_\Theta\, \theta_e)$$
$$\beta \leftarrow \beta + \eta.(\sigma(-\hat{x}^{\text{AUX}}_{e,f,g})(\theta_f - \theta_g) - \lambda_\beta\, \beta)$$

[2] We omit bias terms for brevity.

4 Experiments

To implement our approach, we extend the implementation of BPR in LibRec[3] which is a publicly available Java library for recommender systems.

We evaluate our model for the task of Bilingual Lexicon Induction (BLI). Given a source word f, the task is to rank all candidate target words e by their predicted translation scores $\hat{x}_{e,f}$. We conduct large-scale experiments on 27 low- and high-resource source languages and evaluate their translations to English. We use the 100K most frequent words from English Wikipedia as candidate English target words (E).

At test time, for each source language, we evaluate the top-10 accuracy (Acc_{10}): the percent of source language words in the test set for which a correct English translation appears in the top-10 ranked English candidates.

4.1 Data

4.1.1 Test sets

We use benchmark test sets for the task of bilingual lexicon induction to evaluate the performance of our model. The **VULIC1000** dataset (Vulic and Moens, 2016) comprises 1000 nouns in Spanish, Italian, and Dutch, along with their one-to-one ground-truth word translations in English.

We construct a new test set (CrowdTest) for a larger set of 27 languages from crowdsourced dictionaries (Pavlick et al., 2014). For each language, we randomly pick up to 1000 words that have only one English word translation in the crowdsourced dictionary to be the test set for that language. On average, there are 967 test source words with a variety of POS per language. Since different language treats grammatical categories such as tense and number differently (for example, unlike English, tenses are not expressed by specific forms of words in Indonesian (`id`); rather, they are expressed through context), we make our evaluation on all languages in CrowdTest generic by treating a predicted English translation of a foreign word as correct as long as it has the same lemma as the gold English translation. To facilitate further research, we make CrowdTest publicly available in our website.

4.1.2 Bilingual Signals for Translation

We use Wikipedia to incorporate information from a *third* language into the matrix, with observed translations to both the source language and the target language, English. We first collect all interlingual links from English Wikipedia pages to pages in other languages. Using these links, we obtain translations of Wikipedia page titles in many languages to English e.g., `id.wikipedia.org/wiki/Kulkas` → *fridge* (en). The observed translations are projected to fill the missing translations in the matrix (Figure 3). We call these bilingual translations **`WIKI`**.

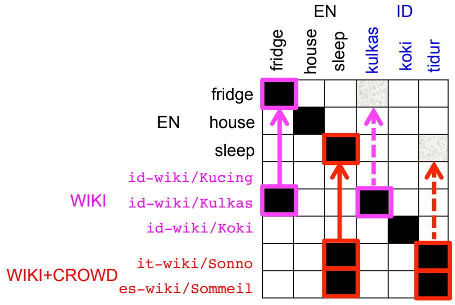


Figure 3: Wikipedia pages with observed translations to the source (id) and the target (en) languages act as a *third* language in the matrix.

From the links that we have collected, we can also infer links from Wikipedia pages in the source language to other pages in non-target languages e.g., `id.wikipedia.org/wiki/Kulkas` → `it.wikipedia.org/wiki/Frigorifero`. The titles of these pages can be translated to English if they exist as entries in the dictionaries. These non-source, non-target language pages can act as yet another third language whose observed translations can be projected to fill the missing translations in the matrix (Figure 3). We call these bilingual translations **`WIKI+CROWD`**.

4.1.3 Monolingual Signals for Translation

We define *cold* source words in our experiments as source words that have no associated **`WIKI`** translations and fewer than 2 associated **`WIKI+CROWD`** translations. For each cold source word f, we predict the score of its translation to each candidate English word e using the auxiliary formulation of $\hat{x}_{e,f}$ (Equation 2). There are two auxiliary signals about the words that we use in our experiments: (1) bilingually informed word embeddings and (2) visual representations.

Bilingually Informed Word Embeddings For each language, we learn monolingual embeddings for its words by training a standard monolingual `word2vec` skipgram model (Mikolov

[3] https://www.librec.net/index.html

et al., 2013b) on tokenized Wikipedia pages of that language using Gensim (Řehůřek and Sojka, 2010). We obtain 100-dimensional word embeddings with 15 epochs, 15 negatives, window size of 5, and cutoff value of 5.

Given two monolingual embedding spaces $\mathbb{R}^{d_F}$ and $\mathbb{R}^{d_E}$ of the source and target languages F and E, where d_f and d_e denote the dimensionality of the monolingual embedding spaces, we use the set of crowdsourced translations that are not in the test set as our seed bilingual translations[4] and learn a mapping function $\mathbf{W} \in \mathbb{R}^{d_E \times d_F}$ that maps the target language vectors in the seed translations to their corresponding source language vectors.[5]

We learn two types of mapping: linear and non-linear, and compare their performances. The linear mapping (Mikolov et al., 2013a; Dinu et al., 2014) minimizes: $||\mathbf{X_E W} - \mathbf{X_F}||_F^2$ where, following the notation in (Vulić and Korhonen, 2016), $\mathbf{X_E}$ and $\mathbf{X_F}$ are matrices obtained by respective concatenation of target language and source language vectors that are in the seed bilingual translations. We solve this optimization problem using stochastic gradient descent (SGD).

We also consider a non-linear mapping (Socher et al., 2013) using a simple four-layer neural network, $\mathbf{W}$ = ($\phi^{(1)}$, $\phi^{(2)}$, $\phi^{(3)}$, $\phi^{(4)}$) that is trained to minimize:

$$\sum_{\mathbf{x}_f \in \mathbf{X_F}} \sum_{\mathbf{x}_e \in \mathbf{X_E}} ||\mathbf{x}_f - \phi^{(4)} s(\phi^{(3)} s(\phi^{(2)} s(\phi^{(1)} \mathbf{x}_e)))||^2$$

where $\phi^{(1)} \in \mathbb{R}^{h_1 \times d_E}$, $\phi^{(2)} \in \mathbb{R}^{h_2 \times h_1}$, $\phi^{(3)} \in \mathbb{R}^{h_3 \times h_2}$, $\phi^{(4)} \in \mathbb{R}^{d_F \times h_3}$, h_n is the size of the hidden layer, and $s = \tanh$ is the chosen non-linear function.

Once the map $\mathbf{W}$ is learned, all candidate target word vectors $\mathbf{x}_e$ can be mapped into the source language embedding space $\mathbb{R}^{d_F}$ by computing $\mathbf{x}_e^T \mathbf{W}$. Instead of the raw monolingual word embeddings $\mathbf{x}_e$, we use these bilingually-informed mapped word vectors $\mathbf{x}_e^T \mathbf{W}$ as the auxiliary word features **WORD-AUX** to estimate $\hat{x}_{e,f}^{\mathrm{AUX}}$.

Visual Representations Pilot Study Recent work (Vulić et al., 2016) has shown that combining word embeddings and visual representations of words can help achieve more accurate bilingual translations. Since the visual representation of a word seems to be language-independent (e.g. the concept of *water* has similar images whether expressed in English or French (Figure 4), the visual representations of a word may be useful for inferring its translation and for complementing the information learned in text.

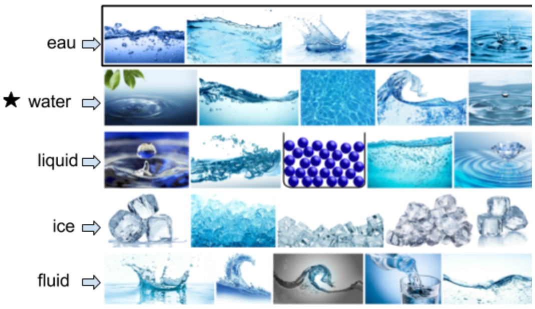


Figure 4: Five images for the French word *eau* and its top 4 translations ranked using visual similarities of images associated with English words (Bergsma and Van Durme, 2011)

We performed a pilot study to include visual features as auxiliary features in our framework. We use a large multilingual corpus of labeled images (Callahan, 2017) to obtain the visual representation of the words in our source and target languages. The corpus contains 100 images for up to 10k words in each of 100 foreign languages, plus images of each of their translations into English. For each of the images, a convolutional neural network (CNN) feature vector is also provided following the method of Kiela et al. (2015). For each word, we use 10 images provided by this corpus and use their CNN features as auxiliary visual features **VISUAL-AUX** to estimate $\hat{x}_{e,f}^{\mathrm{AUX}}$.

4.1.4 Combining Signals

During training, we trained the parameters of $\hat{x}_{e,f}^{\mathrm{MF}}$ and $\hat{x}_{e,f}^{\mathrm{AUX}}$ using a variety of signals:

- $\hat{x}_{e,f}^{\mathrm{MF-W}}$ is trained using **WIKI** translations as the set of observed translations T
- $\hat{x}_{e,f}^{\mathrm{MF-W+C}}$ is trained using **WIKI+CROWD** translations as the set of observed T
- $\hat{x}_{e,f}^{\mathrm{AUX-WE}}$ is trained using the set of word identities $T^{identity}$ and **WORD-AUX** as θ_f
- $\hat{x}_{e,f}^{\mathrm{AUX-VIS}}$ is trained using the set of word identities $T^{identity}$ and **VISUAL-AUX** as θ_f

During testing, we use the following back-off scheme to predict translation scores given a source word f and a candidate target word e:

[4] On average, there are 9846 crowdsourced translations per language that we can use as seed translations.

[5] We find that mapping from target to source vectors gives better performances across models in our experiments.

$$\hat{x}_{e,f} = \begin{cases} \hat{x}^{\mathrm{MF-W}}_{e,f} & \text{if } f \text{ has } \geq 1 \text{ associated } \texttt{WIKI}, \\ \hat{x}^{\mathrm{MF-W+C}}_{e,f} & \text{else if } f \text{ has } \geq 2 \\ & \quad \text{associated } \texttt{WIKI+CROWD}, \\ \hat{x}^{\mathrm{AUX}}_{e,f} & \text{otherwise} \end{cases}$$

where $\hat{x}^{\mathrm{AUX}}_{e,f} = \alpha_{\mathrm{we}}\, \hat{x}^{\mathrm{AUX-WE}}_{e,f} + \alpha_{\mathrm{vis}}\, \hat{x}^{\mathrm{AUX-VIS}}_{e,f}$

4.2 Results

We conduct experiments using the following variants of our model, each of which progressively incorporates more signals to rank candidate English target words. When a variant uses more than one formulation of $\hat{x}_{e,f}$, it applies them using the backoff scheme that we have described before.

- **BPR_W** uses *only* $\hat{x}^{\mathrm{MF-W}}_{e,f}$
- **BPR_W+C** uses $\hat{x}^{\mathrm{MF-W}}_{e,f}$ and $\hat{x}^{\mathrm{MF-W+C}}_{e,f}$
- **BPR_LN** uses *only* $\hat{x}^{\mathrm{AUX-WE}}_{e,f}$ with linear mapping
- **BPR_NN** uses *only* $\hat{x}^{\mathrm{AUX-WE}}_{e,f}$ with neural network (NN) mapping
- **BPR_WE** uses $\hat{x}^{\mathrm{MF-W}}_{e,f}$, $\hat{x}^{\mathrm{MF-W+C}}_{e,f}$, and $\hat{x}^{\mathrm{AUX-WE}}_{e,f}$ with NN mapping
- **BPR_VIS** adds $\hat{x}^{\mathrm{AUX-VIS}}_{e,f}$ to **BPR_WE**

Table 1: Acc_{10} performance on **VULIC1000**

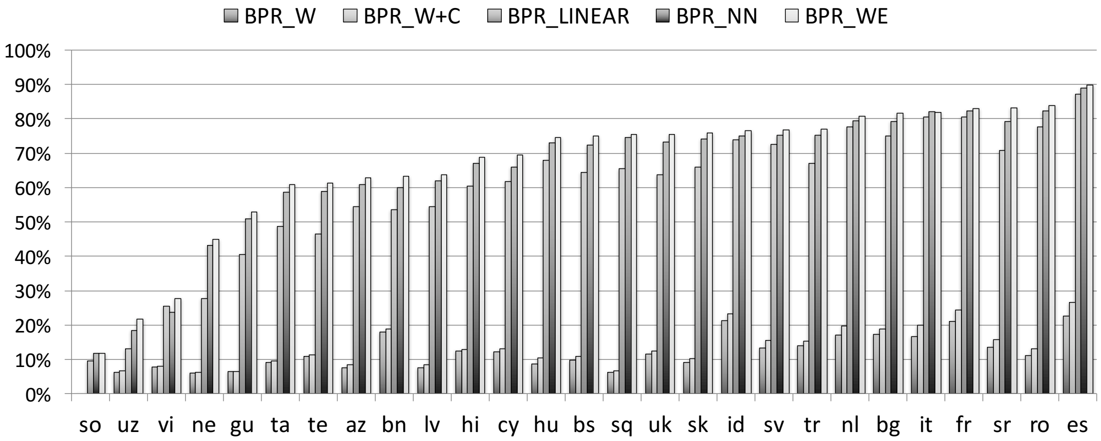
	Baseline (MNN)	**BPR+MNN**	**BPR_LN**	**BPR_WE**
IT-EN	78.8%	79.4%	81.3%	86.0%
ES-EN	81.8%	82.1%	83.4%	87.1%
NL-EN	80.8%	81.6%	83.2%	87.2%

We evaluate the performance of **BPR_WE** against a baseline that is the state-of-the-art model of Vulić and Korhonen (2016), on benchmark **VULIC1000** (Table 1). The baseline (**MNN**) learns a linear mapping between monolingual embedding spaces and finds translations in an *unsupervised* manner: it ranks candidate target words based on their cosine similarities to the source word in the mapped space. As seed translation pairs, **MNN** uses mutual nearest neighbor pairs (MNN) obtained from *pseudo-bilingual* corpora constructed from unannotated monolingual data of the source and target languages (Vulic and Moens, 2016). We train **MNN** and our models using the same 100-dimensional `word2vec` monolingual word embeddings.

As seen in Table 1, we see the benefit of learning translations in a supervised manner. **BPR+MNN** uses the same MNN seed translations as **MNN**, obtained from unannotated monolingual data of English and the foreign language, to learn the linear mapping between their embedding spaces. However, unlike **MNN**, **BPR+MNN** uses the mapped word vectors to predict ranking in a *supervised* manner with BPR objective. This results in higher accuracies than **MNN**. Using seed translations from crowdsourced dictionaries to learn the linear mapping (**BPR_LN**) improves accuracies even further compared to using MNN seed translations obtained from unannotated data. Finally, **BPR_WE** that learns translations in a supervised manner and uses third language translations and non-linear mapping (trained with crowdsourced translations not in the test set) performs consistently and very significantly better than the state-of-the-art on all benchmark test sets. This shows that incorporating more and better signals of translation can improve performance significantly.

Evaluating on **CROWDTEST**, we observe a similar trend over all 27 languages (Figure 5). Particularly, we see that **BPR_W** and **BPR_W+C** suffer from the *cold* start issue where there are too few or no observed translations in the matrix to make accurate predictions. Incorporating auxiliary information in the form of bilingually-informed word embeddings improves the accuracy of the predictions dramatically. For many languages, learning these bilingually-informed word embeddings with non-linear mapping improves accuracy even more. The top accuracy scores achieved by the model vary across languages and seem to be influenced by the amount of data i.e., Wikipedia tokens and seed lexicons entries available for training. Somali (`so`) for example, has only 0.9 million tokens available in its Wikipedia for training the `word2vec` embeddings and only 3 thousand seed translations for learning the mapping between the word embedding spaces. In comparison, Spanish (`es`) has over 500 million tokens available in its Wikipedia and 11 thousand seed translations. We also believe that our choice of tokenization may not be suitable for some languages – we use a simple regular-expression based tokenizer for many languages that do not have a trained NLTK[6] tokenization model. This may influence performance on languages such as Vietnamese (`vi`) on which we have a low performance despite its large Wikipedia corpus.

[6] http://www.nltk.org/



language	so	uz	vi	ne	gu	ta	te	az	bn	lv	hi	cy	hu	bs	sq	uk	sk	id	sv	tr	nl	bg	it	fr	sr	ro	es
BPR_WE	11.7%	21.7%	27.6%	45.0%	52.9%	60.8%	61.2%	62.8%	63.2%	63.8%	68.8%	69.4%	74.6%	75.1%	75.4%	75.5%	75.9%	76.5%	76.8%	77.1%	80.8%	81.6%	81.8%	82.9%	83.3%	83.8%	89.7%
tokens	0.9M	12M	118M	3M	6M	16M	16M	21M	11M	15M	25M	9M	105M	15M	16M	156M	32M	64M	349M	58M	249M	55M	418M	605M	60M	65M	528M
seeds	3K	4K	6K	10K	12K	12K	11K	7K	12K	10K	12K	6K	11K	12K	10K	9K	11K	9K	10K	13K	10K	12K	9K	11K	13K	11K	11K

Figure 5: Acc_{10} on **CROWDTEST** across all 27 languages show that adding more and better signals for translation improves translation accuracies. The top accuracies achieved by our model: **BPR_WE** vary across languages and appear to be influenced by the amount of data (Wikipedia tokens and seed translations) and tokenization available for the language.

Table 2: Top-5 translations of the Indonesian word *kesadaran* (*awareness*) using different model variants

BPR_W	**BPR_LN**	**BPR_NN**	**BPR_WE**
kesadaran	*kesadaran*	*kesadaran*	*kesadaran*
consciousness	consciousness	consciousness	conscience
goddess	*awareness*	empathy	*awareness*
friendship	empathy	*awareness*	understanding
night	perception	perceptions	consciousness
nation	mindedness	perception	acquaintance

Some example translations of an Indonesian word produced by different variants of our model are shown in Table 2. Adding third language translation signals on top of the bilingually-informed auxiliary signals improves accuracies even further.[7] The accuracies achieved by **BPR_WE** on these languages are significantly better than previously reported accuracies (Irvine and Callison-Burch, 2017) on test sets constructed from the same crowdsourced dictionaries (Pavlick et al., 2014)[8].

The accuracies across languages appear to improve consistently with the amount of signals being input to the model. In the following experiments, we investigate how sensitive these improvements are with varying training size.

In Figure 6, we show accuracies obtained by **BPR_WE** with varying sizes of seed translation lexicons used to train its mapping. The results show that a seed lexicon size of 5K is enough across languages to achieve optimum performance. This finding is consistent with the finding of Vulić and Korhonen (2016) that accuracies peak at about 5K seed translations across all their models and languages. For future work, it will be interesting to investigate further why this is the case: e.g., how optimal seed size is related to the quality of the seed translations and the size of the test set, and how the optimum seed size should be chosen. Lastly, we experiment with incorporating auxiliary visual signals for learning translations on the multilingual image corpus (Callahan, 2017). The corpus contains 100 images for up to 10K words in each of 100 foreign languages, plus images of each of their translations into English. We train and test our **BPR_VIS** model to learn translations of 5 low- and high-resource languages in this corpus. We use the translations of up to 10K words in each of these languages as test set and use up to

[7] Actual improvement per language depends on the coverage of the Wikipedia interlanguage links for that language

[8] The comparison however, cannot be made apples-to-apples since the way Irvine and Callison-Burch (2017) select test sets from the crowdsourced dictionaries maybe different and they do not release the test sets

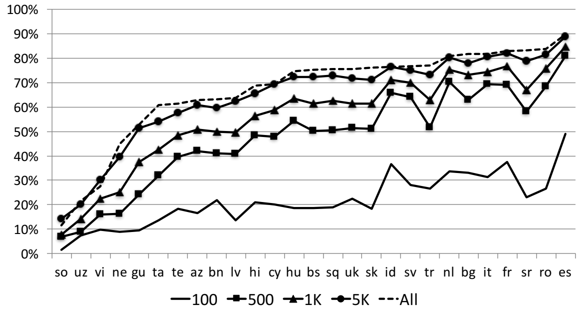


Figure 6: Acc_{10} across different seed lexicon sizes

Table 3: Acc_{10} performance on the multilingual image corpus test set (Callahan, 2017)

	Baseline (CNN-AvgMax)	BPR_VIS	# Seeds
IT-EN	31.4%	55.8%	581
ES-EN	33.0%	58.3%	488
NL-EN	35.5%	69.2%	1857
FR-EN	37.1%	65.9%	1697
ID-EN	36.9%	45.3%	462

10 images (CNN features) of the words in this set as auxiliary visual signals to predict their translations. In this experiment, we weigh auxiliary word embedding and visual features equally. To train the mapping of our word embedding features, we use as seeds crowdsourced translations not in test set.

We compare the quality of our translations with the baseline CNN-AvgMax (Bergsma and Van Durme, 2011), which considers cosine similarities between individual images from the source and target word languages and takes average of their maximum similarities as the final similarity between a source and a target word. For each source word, the candidate target words are ranked according to these final similarities. This baseline has been shown to be effective for inducing translations from images, both in the uni-modal (Bergsma and Van Durme, 2011; Kiela et al., 2015) and multi-modal models (Vulić et al., 2016).

As seen in Table 3, incorporating additional bilingual and textual signals to the visual signals improves translations. Accuracies on these image corpus' test sets are lower overall as they contain a lot of translations from our crowdsourced dictionaries; thus we have much less seeds to train our word embedding mapping. Furthermore, these test sets contain 10 times as many translations as our previous test sets. Using more images instead of just 10 per word may also improve performance.

5 Conclusion

In this paper, we propose a novel framework for combining diverse, sparse and potentially noisy multi-modal signals for translations. We view the problem of learning translations as a matrix completion task and use an effective and extendable matrix factorization approach with BPR to learn translations.

We show the effectiveness of our approach in large scale experiments. Starting from minimally-trained monolingual word embeddings, we consistently and very significantly outperform state-of-the-art approaches by combining these features with other features in a supervised manner using BPR. Since our framework is modular, each input to our prediction can be improved separately to improve the whole system e.g., by learning better word embeddings or a better mapping function to input into the auxiliary component. Our framework is also easily extendable to incorporate more bilingual and auxiliary signals of translation equivalence.

Acknowledgments

This material is based in part on research sponsored by DARPA under grant number HR0011-15-C-0115 (the LORELEI program). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA and the U.S. Government.

This work was also supported by the French National Research Agency under project ANR-16-CE33-0013, and by Amazon through the Amazon Academic Research Awards (AARA) program.

References

Shane Bergsma and Benjamin Van Durme. 2011. Learning Bilingual Lexicons Using the Visual Similarity of Labeled Web Images. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, pages 1764–1769, Barcelona, Spain.

Brendan Callahan. 2017. Image-based bilingual lexicon induction for low resource languages. Master's thesis, University of Pennsylvania.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved Statistical Machine Translation Using Paraphrases. In *Proceedings of the*

Human Language Technology Conference of the NAACL, Main Conference, pages 17–24, New York City.

Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An Autoencoder Approach to Learning Bilingual Word Representations. In *Advances in Neural Information Processing Systems 27*, pages 1853–1861.

Hal Daumé and Jagadeesh Jagarlamudi. 2011. Domain Adaptation for Machine Translation by Mining Unseen Words. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 407–412, Portland, Oregon.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2014. Improving zero-shot learning by mitigating the hubness problem. In *Proceedings of ICLR Workshop*, San Diego, California.

Miao Fan, Deli Zhao, Qiang Zhou, Zhiyuan Liu, Thomas Fang Zheng, and Edward Y. Chang. 2014. Distant Supervision for Relation Extraction with Matrix Completion. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849, Baltimore, Maryland.

Manaal Faruqui and Chris Dyer. 2014. Improving Vector Space Word Representations Using Multilingual Correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden.

Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. 2010. Learning attribute-to-feature mappings for cold-start recommendations. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 176–185. IEEE.

Cyril Goutte, Kenji Yamada, and Eric Gaussier. 2004. Aligning words using matrix factorisation. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 502. Association for Computational Linguistics.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast Bilingual Distributed Representations without Word Alignments. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, pages 748–756, Lille, France.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the Unknown Words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany.

Ruining He and Julian McAuley. 2016. Vbpr: Visual bayesian personalized ranking from implicit feedback. In *AAAI Conference on Artificial Intelligence*, pages 144–150. AAAI Press.

Ann Irvine and Chris Callison-Burch. 2013. Supervised Bilingual Lexicon Induction with Multiple Monolingual Signals. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2013)*, pages 518–523, Atlanta, Georgia.

Ann Irvine and Chris Callison-Burch. 2016. End-to-end statistical machine translation with zero or small parallel texts. *Natural Language Engineering*, 22(04):517–548.

Ann Irvine and Chris Callison-Burch. 2017. A Comprehensive Analysis of Bilingual Lexicon Induction. *Computational Linguistics*, 43(2):273–310.

Douwe Kiela, Ivan Vulić, and Stephen Clark. 2015. Visual Bilingual Lexicon Induction with Transferred ConvNet Features. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 148–158, Lisbon, Portugal.

Alexandre Klementiev, Ann Irvine, Chris Callison-Burch, and David Yarowsky. 2012a. Toward Statistical Machine Translation without Parallel Corpora. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 130–140, Avignon, France.

Alexandre Klementiev and Dan Roth. 2006. Weakly Supervised Named Entity Transliteration and Discovery from Multilingual Comparable Corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 817–824, Sydney, Australia.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012b. Inducing Crosslingual Distributed Representations of Words. In *Proceedings of COLING 2012*, pages 1459–1474, Mumbai, India.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep Multilingual Correlation for Improved Word Embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 250–256, Denver, Colorado.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual Word Representations with Monolingual Quality in Mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, Denver, Colorado.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Ellie Pavlick, Matt Post, Ann Irvine, Dmitry Kachaev, and Chris Callison-Burch. 2014. The language demographics of Amazon Mechanical Turk. *Transactions of the Association for Computational Linguistics*, 2:79–92.

Reinhard Rapp. 1995. Identifying Word Translations in Non-Parallel Texts. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 320–322, Cambridge, Massachusetts.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation Extraction with Matrix Factorization and Universal Schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting Logical Background Knowledge into Embeddings for Relation Extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1119–1129, Denver, Colorado.

Charles Schafer and David Yarowsky. 2002. Inducing Translation Lexicons via Diverse Similarity Measures and Bridge Languages. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–7, Taipei, Taiwan.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany.

Tianze Shi, Zhiyuan Liu, Yang Liu, and Maosong Sun. 2015. Learning Cross-lingual Word Embeddings via Matrix Co-factorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 567–572, Beijing, China.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-Shot Learning Through Cross-Modal Transfer. In *Advances in Neural Information Processing Systems 26*, pages 935–943.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1661–1670, Berlin, Germany.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 886–896, San Diego, California.

Ivan Vulić, Douwe Kiela, Stephen Clark, and Marie-Francine Moens. 2016. Multi-Modal Representations for Improved Bilingual Lexicon Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 188–194, Berlin, Germany.

Ivan Vulić and Anna Korhonen. 2016. On the Role of Seed Lexicons in Learning Bilingual Word Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 247–257, Berlin, Germany.

Ivan Vulić and Marie-Francine Moens. 2013. A study on bootstrapping bilingual vector spaces from non-parallel data (and nothing else). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1613–1624, Seattle, Washington.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual Word Embeddings from Non-Parallel Document-Aligned Data Applied to Bilingual Lexicon Induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 719–725, Beijing, China.

Ivan Vulic and Marie-Francine Moens. 2016. Bilingual Distributed Word Representations from Document-Aligned Comparable Data. *Journal of Artificial Intelligence Research*, 55:953–994.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*.

Guangyou Zhou, Fang Liu, Yang Liu, Shizhu He, and Jun Zhao. 2013. Statistical machine translation improves question retrieval in community question answering via matrix factorization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 852–861, Sofia, Bulgaria.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398, Seattle, Washington.

Reinforcement Learning for Bandit Neural Machine Translation with Simulated Human Feedback

Khanh Nguyen[⊙♢] and **Hal Daumé III**[⊙♠♢♡] and **Jordan Boyd-Graber**[⊙♠♣♢]
University of Maryland: Computer Science,[⊙]Language Science,[♠]iSchool,[♣]UMIACS[♢]
Microsoft Research, New York[♡]
{kxnguyen,hal,jbg}@umiacs.umd.edu

Abstract

Machine translation is a natural candidate problem for reinforcement learning from human feedback: users provide quick, dirty ratings on candidate translations to guide a system to improve. Yet, current neural machine translation training focuses on expensive human-generated reference translations. We describe a reinforcement learning algorithm that improves neural machine translation systems from simulated human feedback. Our algorithm combines the advantage actor-critic algorithm (Mnih et al., 2016) with the attention-based neural encoder-decoder architecture (Luong et al., 2015). This algorithm (a) is well-designed for problems with a large action space and delayed rewards, (b) effectively optimizes traditional corpus-level machine translation metrics, and (c) is robust to skewed, high-variance, granular feedback modeled after actual human behaviors.

1 Introduction

Bandit structured prediction is the task of learning to solve complex joint prediction problems (like parsing or machine translation) under a very limited feedback model: a system must produce a *single* structured output (e.g., translation) and then the world reveals a *score* that measures how good or bad that output is, but provides neither a "correct" output nor feedback on any other possible output (Chang et al., 2015; Sokolov et al., 2015). Because of the extreme sparsity of this feedback, a common experimental setup is that one pre-trains a good-but-not-great "reference" system based on whatever labeled data is available, and then seeks to improve it over time using this bandit feedback. A common motivation for this problem setting is cost. In the case of translation, bilingual "experts" can read a source sentence and a possible translation, and can much more quickly provide a rating of that translation than they can produce a full translation on their own. Furthermore, one can often collect even less expensive ratings from "non-experts" who may or may not be bilingual (Hu et al., 2014). Breaking this reliance on expensive data could unlock previously ignored languages and speed development of broad-coverage machine translation systems.

All work on bandit structured prediction we know makes an important simplifying assumption: the *score* provided by the world is *exactly* the score the system must optimize (§2). In the case of parsing, the score is attachment score; in the case of machine translation, the score is (sentence-level) BLEU. While this simplifying assumption has been incredibly useful in building algorithms, it is highly unrealistic. Any time we want to optimize a system by collecting user feedback, we must take into account:

1. The metric we care about (e.g., expert ratings) may not correlate perfectly with the measure that the reference system was trained on (e.g., BLEU or log likelihood);
2. Human judgments might be more granular than traditional continuous metrics (e.g., thumbs up vs. thumbs down);
3. Human feedback have high *variance* (e.g., different raters might give different responses given the same system output);
4. Human feedback might be substantially *skewed* (e.g., a rater may think all system outputs are poor).

Our first contribution is a strategy to simulate expert and non-expert ratings to evaluate the robustness of bandit structured prediction algorithms in general, in a more realistic environment (§4). We

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1464–1474
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

construct a family of perturbations to capture three attributes: *granularity*, *variance*, and *skew*. We apply these perturbations on automatically generated scores to simulate noisy human ratings. To make our simulated ratings as realistic as possible, we study recent human evaluation data (Graham et al., 2017) and fit models to match the noise profiles in actual human ratings (§4.2).

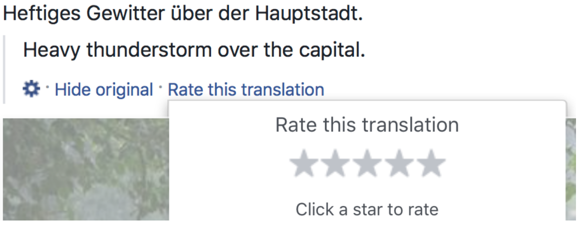
Our second contribution is a reinforcement learning solution to bandit structured prediction and a study of its robustness to these simulated human ratings (§ 3).[1] We combine an encoder-decoder architecture of machine translation (Luong et al., 2015) with the advantage actor-critic algorithm (Mnih et al., 2016), yielding an approach that is simple to implement but works on low-resource bandit machine translation. Even with substantially restricted granularity, with high variance feedback, or with skewed rewards, this combination improves pre-trained models (§6). In particular, under realistic settings of our noise parameters, the algorithm's online reward and final held-out accuracies do not significantly degrade from a noise-free setting.

2 Bandit Machine Translation

The bandit structured prediction problem (Chang et al., 2015; Sokolov et al., 2015) is an extension of the contextual bandits problem (Kakade et al., 2008; Langford and Zhang, 2008) to structured prediction. Bandit structured prediction operates over time $i = 1 \dots K$ as:

1. World reveals context $\boldsymbol{x}^{(i)}$
2. Algorithm predicts structured output $\hat{\boldsymbol{y}}^{(i)}$
3. World reveals reward $R\left(\hat{\boldsymbol{y}}^{(i)}, \boldsymbol{x}^{(i)}\right)$

We consider the problem of *learning to translate from human ratings* in a bandit structured prediction framework. In each round, a translation model receives a source sentence $\boldsymbol{x}^{(i)}$, produces a translation $\hat{\boldsymbol{y}}^{(i)}$, and receives a rating $R\left(\hat{\boldsymbol{y}}^{(i)}, \boldsymbol{x}^{(i)}\right)$ from a human that reflects the quality of the translation. We seek an algorithm that achieves high reward over K rounds (high cumulative reward). The challenge is that even though the model knows how good the translation is, it knows neither *where* its mistakes are nor *what* the "correct" translation looks like. It must balance exploration (finding new good predictions) with exploitation (producing predictions it already knows are good). This is especially difficult in a task like machine translation, where, for a twenty token sentence with a vocabulary size of $50k$, there are approximately 10^{94} possible outputs, of which the algorithm gets to test exactly one.


Heftiges Gewitter über der Hauptstadt.

Heavy thunderstorm over the capital.

Figure 1: A translation rating interface provided by Facebook. Users see a sentence followed by its machined-generated translation and can give ratings from one to five stars.

Despite these challenges, learning from non-expert ratings is desirable. In real-world scenarios, non-expert ratings are easy to collect but other stronger forms of feedback are prohibitively expensive. Platforms that offer translations can get quick feedback "for free" from their users to improve their systems (Figure 1). Even in a setting in which annotators are paid, it is much less expensive to ask a bilingual speaker to provide a rating of a proposed translation than it is to pay a professional translator to produce one from scratch.

3 Effective Algorithm for Bandit MT

This section describes the neural machine translation architecture of our system (§ 3.1). We formulate bandit neural machine translation as a reinforcement learning problem (§ 3.2) and discuss why standard actor-critic algorithms struggle with this problem (§ 3.3). Finally, we describe a more effective training approach based on the advantage actor-critic algorithm (§3.4).

3.1 Neural machine translation

Our neural machine translation (NMT) model is a neural encoder-decoder that directly computes the probability of translating a target sentence $\boldsymbol{y} = (y_1, \cdots, y_m)$ from source sentence $\boldsymbol{x}$:

$$P_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{m} P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) \qquad (1)$$

where $P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$ is the probability of outputting the next word y_t at time step t given a translation prefix $\boldsymbol{y}_{<t}$ and a source sentence $\boldsymbol{x}$.

[1] Our code is at https://github.com/khanhptnk/bandit-nmt (in PyTorch).

We use an encoder-decoder NMT architecture with global attention (Luong et al., 2015), where both the encoder and decoder are recurrent neural networks (RNN) (see Appendix A for a more detailed description). These models are normally trained by supervised learning, but as reference translations are not available in our setting, we use reinforcement learning methods, which only require numerical feedback to function.

3.2 Bandit NMT as Reinforcement Learning

NMT generating process can be viewed as a Markov decision process on a continuous state space. The states are the hidden vectors $\boldsymbol{h}_t^{dec}$ generated by the decoder. The action space is the target language's vocabulary.

To generate a translation from a source sentence $\boldsymbol{x}$, an NMT model starts at an initial state $\boldsymbol{h}_0^{dec}$: a representation of $\boldsymbol{x}$ computed by the encoder. At time step t, the model decides the next action to take by defining a stochastic policy $P_{\boldsymbol{\theta}}(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$, which is directly parametrized by the parameters $\boldsymbol{\theta}$ of the model. This policy takes the current state $\boldsymbol{h}_{t-1}^{dec}$ as input and produces a probability distribution over all actions (target vocabulary words). The next action $\hat{y}_t$ is chosen by taking $\arg\max$ or sampling from this distribution. The model computes the next state $\boldsymbol{h}_t^{dec}$ by updating the current state $\boldsymbol{h}_{t-1}^{dec}$ by the action taken $\hat{y}_t$.

The objective of bandit NMT is to find a policy that maximizes the expected reward of translations sampled from the model's policy:

$$\max_{\boldsymbol{\theta}} \mathcal{L}_{pg}(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \mathbb{E}_{\substack{\boldsymbol{x}\sim D_{\text{tr}} \\ \hat{\boldsymbol{y}}\sim P_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})}} \left[R(\hat{\boldsymbol{y}}, \boldsymbol{x}) \right] \tag{2}$$

where D_{tr} is the training set and R is the reward function (the rater).[2] We optimize this objective function with policy gradient methods. For a fixed $\boldsymbol{x}$, the gradient of the objective in Eq 2 is:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{pg}(\boldsymbol{\theta}) = \mathbb{E}_{\hat{\boldsymbol{y}}\sim P_{\boldsymbol{\theta}}(\cdot)} \left[R(\hat{\boldsymbol{y}}) \nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\hat{\boldsymbol{y}}) \right] \tag{3}$$

$$= \sum_{t=1}^{m} \mathbb{E}_{\substack{\hat{y}_t\sim \\ P_{\boldsymbol{\theta}}(\cdot|\hat{\boldsymbol{y}}_{<t})}} \left[Q(\hat{\boldsymbol{y}}_{<t}, \hat{y}_t) \nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\hat{y}_t \mid \hat{\boldsymbol{y}}_{<t}) \right]$$

where $Q(\hat{\boldsymbol{y}}_{<t}, \hat{y}_t)$ is the expected future reward of $\hat{y}_t$ given the current prefix $\hat{\boldsymbol{y}}_{<t}$, then continuing sampling from $P_{\boldsymbol{\theta}}$ to complete the translation:

$$Q(\hat{\boldsymbol{y}}_{<t}, \hat{y}_t) = \mathbb{E}_{\hat{\boldsymbol{y}}'\sim P_{\boldsymbol{\theta}}(\cdot|\boldsymbol{x})} \left[\tilde{R}(\hat{\boldsymbol{y}}', \boldsymbol{x}) \right] \tag{4}$$

with $\tilde{R}(\hat{\boldsymbol{y}}', \boldsymbol{x}) \equiv R(\hat{\boldsymbol{y}}', \boldsymbol{x}) \mathbb{1}\left\{ \hat{\boldsymbol{y}}'_{<t} = \hat{\boldsymbol{y}}_{<t}, \hat{y}'_t = \hat{y}_t \right\}$ $\mathbb{1}\{\cdot\}$ is the indicator function, which returns 1 if the logic inside the bracket is true and returns 0 otherwise.

The gradient in Eq 3 requires rating all possible translations, which is not feasible in bandit NMT. Naïve Monte Carlo reinforcement learning methods such as REINFORCE (Williams, 1992) estimates Q values by sample means but yields very high variance when the action space is large, leading to training instability.

3.3 Why are actor-critic algorithms not effective for bandit NMT?

Reinforcement learning methods that rely on function approximation are preferred when tackling bandit structured prediction with a large action space because they can capture similarities between structures and generalize to unseen regions of the structure space. The actor-critic algorithm (Konda and Tsitsiklis) uses function approximation to directly model the Q function, called the *critic* model. In our early attempts on bandit NMT, we adapted the actor-critic algorithm for NMT in Bahdanau et al. (2017), which employs the algorithm in a supervised learning setting. Specifically, while an encoder-decoder critic model $Q_{\boldsymbol{\omega}}$ as a substitute for the true Q function in Eq 3 enables taking the full expectation (because the critic model can be queried with any state-action pair), we are unable to obtain reasonable results with this approach.

Nevertheless, insights into why this approach fails on our problem explains the effectiveness of the approach discussed in the next section. There are two properties in Bahdanau et al. (2017) that our problem lacks but are key elements for a successful actor-critic. The first is access to reference translations: while the critic model is able to observe reference translations during training in their setting, bandit NMT assumes those are never available. The second is per-step rewards: while the reward function in their setting is known and can be exploited to compute immediate rewards after taking each action, in bandit NMT, the actor-critic algorithm struggles with credit assignment because it only receives reward when a translation is completed. Bahdanau et al. (2017) report that the algorithm degrades if rewards are delayed until the end, consistent with our observations.

With an enormous action space of bandit NMT, approximating gradients with the Q critic model

[2] Our raters are *stochastic*, but for simplicity we denote the reward as a function; it should be expected reward.

induces biases and potentially drives the model to wrong optima. Values of rarely taken actions are often overestimated without an explicit constraint between Q values of actions (e.g., a sum-to-one constraint). Bahdanau et al. (2017) add an ad-hoc regularization term to the loss function to mitigate this issue and further stablizes the algorithm with a delay update scheme, but at the same time introduces extra tuning hyper-parameters.

3.4 Advantage Actor-Critic for Bandit NMT

We follow the approach of advantage actor-critic (Mnih et al., 2016, A2C) and combine it with the neural encoder-decoder architecture. The resulting algorithm—which we call NED-A2C—approximates the gradient in Eq 3 by a single sample $\hat{\boldsymbol{y}} \sim P(\cdot \mid \hat{\boldsymbol{x}})$ and centers the reward $R(\hat{\boldsymbol{y}})$ using the state-specific expected future reward $V(\hat{\boldsymbol{y}}_{<t})$ to reduce variance:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{pg}(\boldsymbol{\theta}) \approx \sum_{t=1}^{m} \bar{R}_t(\hat{\boldsymbol{y}}) \nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\hat{y}_t \mid \hat{\boldsymbol{y}}_{<t}) \tag{5}$$

$$\begin{aligned} \text{with } \bar{R}_t(\hat{\boldsymbol{y}}) &\equiv R(\hat{\boldsymbol{y}}) - V(\hat{\boldsymbol{y}}_{<t}) \\ V(\hat{\boldsymbol{y}}_{<t}) &\equiv \mathbb{E}_{\hat{y}'_t \sim P(\cdot \mid \hat{\boldsymbol{y}}_{<t})} \left[Q(\hat{\boldsymbol{y}}_{<t}, \hat{y}'_t) \right] \end{aligned}$$

We train a separate attention-based encoder-decoder model $V_{\boldsymbol{\omega}}$ to estimate V values. This model encodes a source sentence $\boldsymbol{x}$ and decodes a sampled translation $\hat{\boldsymbol{y}}$. At time step t, it computes $V_{\boldsymbol{\omega}}(\hat{\boldsymbol{y}}_{<t}, \boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{h}_t^{crt}$, where $\boldsymbol{h}_t^{crt}$ is the current decoder's hidden vector and $\boldsymbol{w}$ is a learned weight vector. The critic model minimizes the MSE between its estimates and the true values:

$$\mathcal{L}_{crt}(\boldsymbol{\omega}) = \mathbb{E}_{\substack{\boldsymbol{x} \sim D_{\text{tr}} \\ \hat{\boldsymbol{y}} \sim P_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x})}} \left[\sum_{t=1}^{m} L_t(\hat{\boldsymbol{y}}, \boldsymbol{x}) \right] \tag{6}$$

$$\text{with } L_t(\hat{\boldsymbol{y}}, \boldsymbol{x}) \equiv \left[V_{\boldsymbol{\omega}}(\hat{\boldsymbol{y}}_{<t}, \boldsymbol{x}) - R(\hat{\boldsymbol{y}}, \boldsymbol{x}) \right]^2 .$$

We use a gradient approximation to update $\boldsymbol{\omega}$ for a fixed $\boldsymbol{x}$ and $\hat{\boldsymbol{y}} \sim P(\cdot \mid \hat{\boldsymbol{x}})$:

$$\nabla_{\boldsymbol{\omega}} \mathcal{L}_{crt}(\boldsymbol{\omega}) \approx \sum_{t=1}^{m} \left[V_{\boldsymbol{\omega}}(\hat{\boldsymbol{y}}_{<t}) - R(\hat{\boldsymbol{y}}) \right] \nabla_{\boldsymbol{\omega}} V_{\boldsymbol{\omega}}(\hat{\boldsymbol{y}}_{<t}) \tag{7}$$

NED-A2C is better suited for problems with a large action space and has other advantages over actor-critic. For large action spaces, approximating gradients using the V critic model induces lower biases than using the Q critic model. As implied by its definition, the V model is robust to biases incurred by rarely taken actions since rewards of those actions are weighted by very small probabilities in the expectation. In addition, the V model has a much smaller number of parameters and thus is more sample-efficient and more stable to train than the Q model. These attractive properties were not studied in A2C's original paper (Mnih et al., 2016).

Algorithm 1 The NED-A2C algorithm for bandit NMT.

```
for $i = 1 \cdots K$ do
    receive a source sentence $\boldsymbol{x}^{(i)}$
    sample a translation: $\hat{\boldsymbol{y}}^{(i)} \sim P_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x}^{(i)})$
    receive reward $R(\hat{\boldsymbol{y}}^{(i)}, \boldsymbol{x}^{(i)})$
    update the NMT model using Eq 5.
    update the critic model using Eq 7.
end for
```

Algorithm 1 summarizes NED-A2C for bandit NMT. For each $\boldsymbol{x}$, we draw a single sample $\hat{\boldsymbol{y}}$ from the NMT model, which is used for both estimating gradients of the NMT model and the critic model. We run this algorithm with mini-batches of $\boldsymbol{x}$ and aggregate gradients over all $\boldsymbol{x}$ in a mini-batch for each update. Although our focus is on bandit NMT, this algorithm naturally works with any bandit structured prediction problem.

4 Modeling Imperfect Ratings

Our goal is to establish the feasibility of using *real* human feedback to optimize a machine translation system, in a setting where one can collect *expert* feedback as well as a setting in which one only collects *non-expert* feedback. In all cases, we consider the expert feedback to be the "gold standard" that we wish to optimize. To establish the feasibility of driving learning from human feedback *without* doing a full, costly user study, we begin with a simulation study. The key aspects (Figure 2) of human feedback we capture are: (a) mismatch between training objective and feedback-maximizing objective, (b) human ratings typically are binned (§ 4.1), (c) individual human ratings have high variance (§4.2), and (d) non-expert ratings can be skewed with respect to expert ratings (§4.3).

In our simulated study, we begin by modeling gold standard human ratings using add-one-smoothed sentence-level BLEU (Chen and Cherry, 2014).[3] Our evaluation criteria, therefore, is average sentence-BLEU over the run of our algo-

[3] "Smoothing 2" in Chen and Cherry (2014).

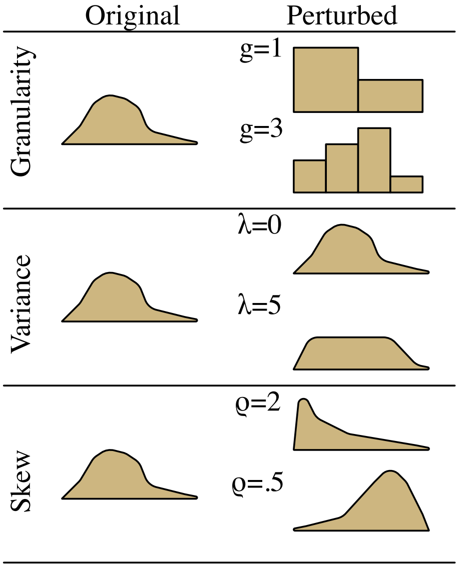


Figure 2: Examples of how our perturbation functions change the "true" feedback distribution (left) to ones that better capture features found in human feedback (right).

rithm. However, in any realistic scenario, human feedback will vary from its average, and so the reward that our algorithm receives will be a *perturbed* variant of sentence-BLEU. In particular, if the sentence-BLEU score is $s \in [0,1]$, the algorithm will only observe $s' \sim \text{pert}(s)$, where pert is a perturbation distribution. Because our reference machine translation system is pre-trained using log-likelihood, there is already an (a) mismatch between training objective and feedback, so we focus on (b-d) below.

4.1 Humans Provide Granular Feedback

When collecting human feedback, it is often more effective to collect discrete *binned* scores. A classic example is the Likert scale for human agreement (Likert, 1932) or star ratings for product reviews. Insisting that human judges provide continuous values (or feedback at too fine a granularity) can demotivate raters without improving rating quality (Preston and Colman, 2000).

To model granular feedback, we use a simple rounding procedure. Given an integer parameter g for degree of granularity, we define:

$$\text{pert}^{\text{gran}}(s; g) = \frac{1}{g}\text{round}(gs) \qquad (8)$$

This perturbation function divides the range of possible outputs into $g + 1$ bins. For example, for $g = 5$, we obtain bins $[0, 0.1)$, $[0.1, 0.3)$, $[0.3, 0.5)$, $[0.5, 0.7)$, $[0.7, 0.9)$ and $[0.9, 1.0]$. Since most sentence-BLEU scores are much closer to zero than to one, many of the larger bins are frequently vacant.

4.2 Experts Have High Variance

Human feedback has high variance around its expected value. A natural goal for a variance model of human annotators is to simulate—as closely as possible—how human raters actually perform. We use human evaluation data recently collected as part of the WMT shared task (Graham et al., 2017). The data consist of 7200 sentences multiply annotated by giving non-expert annotators on Amazon Mechanical Turk a reference sentence and a *single* system translation, and asking the raters to judge the adequacy of the translation.[4]

From these data, we treat the *average* human rating as the ground truth and consider how individual human ratings vary around that mean. To visualize these results with kernel density estimates (standard normal kernels) of the *standard deviation*. Figure 3 shows the mean rating (x-axis) and the deviation of the human ratings (y-axis) at each mean.[5] As expected, the standard deviation is small at the extremes and large in the middle (this is a bounded interval), with a fairly large range in the middle: a translation whose average score is 50 can get human evaluation scores anywhere between 20 and 80 with high probability. We use a linear approximation to define our variance-based perturbation function as a Gaussian distribution, which is parameterized by a scale λ that grows or shrinks the variances (when $\lambda = 1$ this exactly matches the variance in the plot).

$$\text{pert}^{\text{var}}(s; \lambda) = \text{Nor}\left(s, \lambda\sigma(s)^2\right) \qquad (9)$$

$$\sigma(s) = \begin{cases} 0.64s - 0.02 & \text{if } s < 50 \\ -0.67s + 67.0 & \text{otherwise} \end{cases}$$

4.3 Non-Experts are Skewed from Experts

The preceding two noise models assume that the reward closely models the value we want to optimize (has the same mean). This may not be the case with non-expert ratings. Non-expert

[4] Typical machine translation evaluations evaluate pairs and ask annotators to choose which is better.

[5] A current limitation of this model is that the simulated noise is i.i.d. conditioned on the rating (homoscedastic noise). While this is a stronger and more realistic model than assuming no noise, real noise is likely heteroscedastic: dependent on the input.

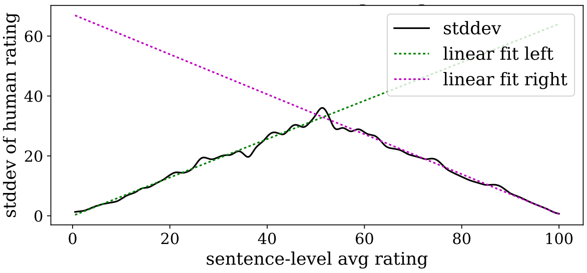


Figure 3: Average rating (x-axis) versus a kernel density estimate of the variance of human ratings around that mean, with linear fits. Human scores vary more around middling judgments than extreme judgments.

	De-En	Zh-En
Supervised training	186K	190K
Bandit training	167K	165K
Development	7.7K	7.9K
Test	9.1K	7.4K

Table 1: Sentence counts in data sets.

raters are skewed both for reinforcement learning (Thomaz et al., 2006; Thomaz and Breazeal, 2008; Loftin et al., 2014) and recommender systems (Herlocker et al., 2000; Adomavicius and Zhang, 2012), but are typically bimodal: some are harsh (typically provide very low scores, even for "okay" outputs) and some are motivational (providing high scores for "okay" outputs).

We can model both harsh and motivations raters with a simple deterministic skew perturbation function, parametrized by a scalar $\rho \in [0, \infty)$:

$$\text{pert}^{\text{skew}}(s; \rho) = s^{\rho} \tag{10}$$

For $\rho > 1$, the rater is harsh; for $\rho < 1$, the rater is motivational.

5 Experimental Setup

We choose two language pairs from different language families with different typological properties: German-to-English and (De-En) and Chinese-to-English (Zh-En). We use parallel transcriptions of TED talks for these pairs of languages from the machine translation track of the IWSLT 2014 and 2015 (Cettolo et al., 2014, 2015, 2012). For each language pair, we split its data into four sets for supervised training, bandit training, development and testing (Table 1). For English and German, we tokenize and clean sentences using Moses (Koehn et al., 2007). For Chinese, we use the Stanford Chinese word segmenter (Chang et al., 2008) to segment sentences and tokenize. We remove all sentences with length greater than 50, resulting in an average sentence length of 18. We use IWSLT 2015 data for supervised training and development, IWSLT 2014 data for bandit training and previous years' development and evaluation data for testing.

5.1 Evaluation Framework

For each task, we first use the supervised training set to pre-train a reference NMT model using supervised learning. On the same training set, we also pre-train the critic model with translations sampled from the pre-trained NMT model. Next, we enter a bandit learning mode where our models only observe the source sentences of the bandit training set. Unless specified differently, we train the NMT models with NED-A2C for one pass over the bandit training set. If a perturbation function is applied to Per-Sentence BLEU scores, it is only applied in this stage, not in the pre-training stage.

We measure the *improvement* ΔS of an evaluation metric S due to bandit training: $\Delta S = S_{A2C} - S_{ref}$, where S_{ref} is the metric computed on the reference models and S_{A2C} is the metric computed on models trained with NED-A2C. Our primary interest is *Per-Sentence* BLEU: average sentence-level BLEU of translations that are sampled and scored during the bandit learning pass. This metric represents average expert ratings, which we want to optimize for in real-world scenarios. We also measure *Heldout* BLEU: corpus-level BLEU on an unseen test set, where translations are greedily decoded by the NMT models. This shows how much our method improves translation quality, since corpus-level BLEU correlates better with human judgments than sentence-level BLEU.

Because of randomness due to both the random sampling in the model for "exploration" as well as the randomness in the reward function, we repeat each experiment five times and report the mean results with 95% confidence intervals.

5.2 Model configuration

Both the NMT model and the critic model are encoder-decoder models with global attention (Luong et al., 2015). The encoder and the decoder are unidirectional single-layer LSTMs. They have the same word embedding size and

LSTM hidden size of 500. The source and target vocabulary sizes are both 50K. We do not use dropout in our experiments. We train our models by the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9, \beta_2 = 0.999$ and a batch size of 64. For Adam's α hyperparameter, we use 10^{-3} during pre-training and 10^{-4} during bandit learning (for both the NMT model and the critic model). During pre-training, starting from the fifth pass, we decay α by a factor of 0.5 when perplexity on the development set increases. The NMT model reaches its highest corpus-level BLEU on the development set after ten passes through the supervised training data, while the critic model's training error stabilizes after five passes. The training speed is 18s/batch for supervised pre-training and 41s/batch for training with the NED-A2C algorithm.

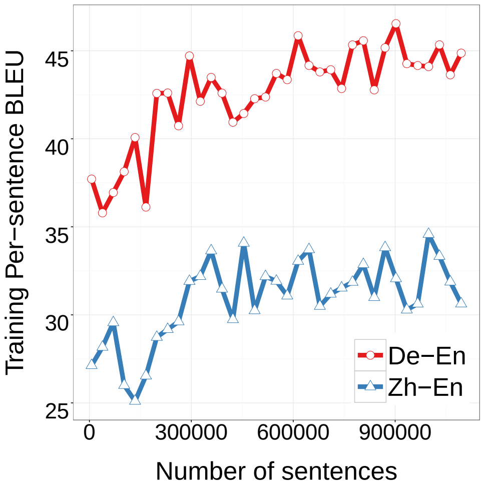


Figure 4: Learning curves of models trained with NED-A2C for five epochs.

6 Results and Analysis

In this section, we describe the results of our experiments, broken into the following questions: how NED-A2C improves reference models (§6.1); the effect the three perturbation functions have on the algorithm (§ 6.2); and whether the algorithm improves a corpus-level metric that corresponds well with human judgments (§6.3).

6.1 Effectiveness of NED-A2C under Un-perturbed Bandit Feedback

We evaluate our method in an ideal setting where *un-perturbed* Per-Sentence BLEU simulates ratings during both training and evaluation (Table 2).

Single round of feedback. In this setting, our models only observe each source sentence once and before producing its translation. On both De-En and Zh-En, NED-A2C improves Per-Sentence BLEU of reference models after only a single pass (+2.82 and +1.08 respectively).

Poor initialization. Policy gradient algorithms have difficulty improving from poor initializations, especially on problems with a large action space, because they use model-based exploration, which is ineffective when most actions have equal probabilities (Bahdanau et al., 2017; Ranzato et al., 2016). To see whether NED-A2C has this problem, we repeat the experiment with the same setup but with reference models pre-trained for only a single pass. Surprisingly, NED-A2C is highly effective at improving these poorly trained models (+7.07 on De-En and +3.60 on Zh-En in Per-Sentence BLEU).

Comparisons with supervised learning. To further demonstrate the effectiveness of NED-A2C, we compare it with training the reference models with supervised learning for a single pass on the bandit training set. Surprisingly, observing ground-truth translations barely improves the models in Per-Sentence BLEU when they are fully trained (less than +0.4 on both tasks). A possible explanation is that the models have already reached full capacity and do not benefit from more examples.[6] NED-A2C further enhances the models because it eliminates the mismatch between the supervised training objective and the evaluation objective. On weakly trained reference models, NED-A2C also significantly outperforms supervised learning (ΔPer-Sentence BLEU of NED-A2C is over three times as large as those of supervised learning).

Multiple rounds of feedback. We examine if NED-A2C can improve the models even further with multiple rounds of feedback.[7] With supervised learning, the models can memorize the reference translations but, in this case, the models have to be able to exploit and explore effectively. We train the models with NED-A2C for five

[6] This result may vary if the domains of the supervised learning set and the bandit training set are dissimilar. Our training data are all TED talks.

[7] The ability to receive feedback on the same example multiple times might not fit all use cases though.

	De-En			Zh-En		
	Reference	Δ_{sup}	Δ_{A2C}	Reference	Δ_{sup}	Δ_{A2C}
Fully pre-trained reference model						
Per-Sentence BLEU	38.26 ± 0.02	0.07 ± 0.05	**2.82 ± 0.03**	32.79 ± 0.01	0.36 ± 0.05	**1.08 ± 0.03**
Heldout BLEU	24.94 ± 0.00	1.48 ± 0.00	**1.82 ± 0.08**	13.73 ± 0.00	**1.18 ± 0.00**	0.86 ± 0.11
Weakly pre-trained reference model						
Per-Sentence BLEU	19.15 ± 0.01	2.94 ± 0.02	**7.07 ± 0.06**	14.77 ± 0.01	1.11 ± 0.02	**3.60 ± 0.04**
Heldout BLEU	19.63 ± 0.00	**3.94 ± 0.00**	1.61 ± 0.17	9.34 ± 0.00	**2.31 ± 0.00**	0.92 ± 0.13

Table 2: Translation scores and improvements based on a single round of un-perturbed bandit feedback. Per-Sentence BLEU and Heldout BLEU are not comparable: the former is sentence-BLEU, the latter is corpus-BLEU.

passes and observe a much more significant ΔPer-Sentence BLEU than training for a single pass in both pairs of language (+6.73 on De-En and +4.56 on Zh-En) (Figure 4).

6.2 Effect of Perturbed Bandit Feedback

We apply perturbation functions defined in § 4.1 to Per-Sentence BLEU scores and use the perturbed scores as rewards during bandit training (Figure 5).

Granular Rewards. We discretize raw Per-Sentence BLEU scores using $\text{pert}^{gran}(s; g)$ (§4.1). We vary g from one to ten (number of bins varies from two to eleven). Compared to continuous rewards, for both pairs of languages, ΔPer-Sentence BLEU is not affected with g at least five (at least six bins). As granularity decreases, ΔPer-Sentence BLEU monotonically degrades. However, even when $g = 1$ (scores are either 0 or 1), the models still improve by at least a point.

High-variance Rewards. We simulate noisy rewards using the model of human rating variance $\text{pert}^{var}(s; \lambda)$ (§ 4.2) with $\lambda \in \{0.1, 0.2, 0.5, 1, 2, 5\}$. Our models can withstand an amount of about 20% the variance in our human eval data without dropping in ΔPer-Sentence BLEU. When the amount of variance attains 100%, matching the amount of variance in the human data, ΔPer-Sentence BLEU go down by about 30% for both pairs of languages. As more variance is injected, the models degrade quickly but still improve from the pre-trained models. Variance is the most detrimental type of perturbation to NED-A2C among the three aspects of human ratings we model.

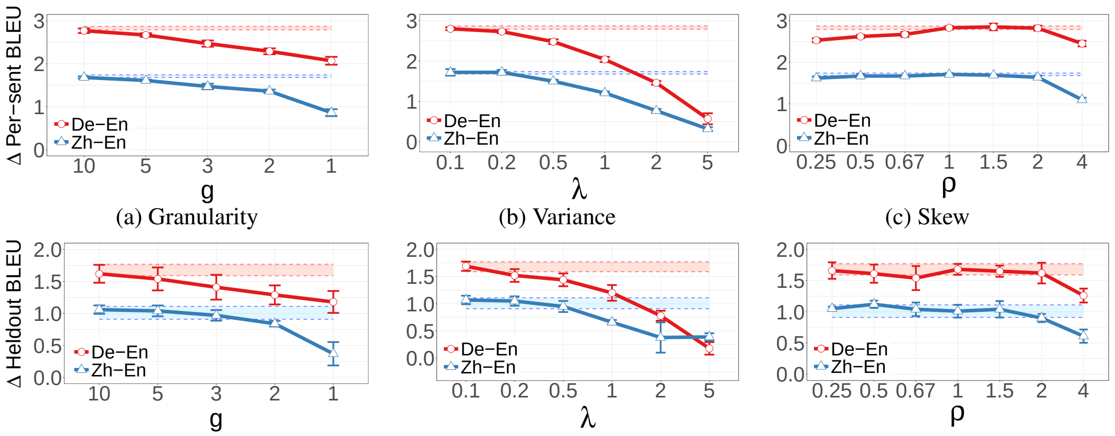
Skewed Rewards. We model skewed raters using $\text{pert}^{skew}(s; \rho)$ (§ 4.3) with $\rho \in \{0.25, 0.5, 0.67, 1, 1.5, 2, 4\}$. NED-A2C is robust to skewed scores. ΔPer-Sentence BLEU is at least 90% of unskewed scores for most skew values. Only when the scores are extremely harsh ($\rho = 4$) does ΔPer-Sentence BLEU degrade significantly (most dramatically by 35% on Zh-En). At that degree of skew, a score of 0.3 is suppressed to be less than 0.08, giving little signal for the models to learn from. On the other spectrum, the models are less sensitive to motivating scores as Per-Sentence BLEU is unaffected on Zh-En and only decreases by 7% on De-En.

6.3 Held-out Translation Quality

Our method also improves pre-trained models in Heldout BLEU, a metric that correlates with translation quality better than Per-Sentence BLEU (Table 2). When scores are perturbed by our rating model, we observe similar patterns as with Per-Sentence BLEU: the models are robust to most perturbations except when scores are very coarse, or very harsh, or have very high variance (Figure 5, second row). Supervised learning improves Heldout BLEU better, possibly because maximizing log-likelihood of reference translations correlates more strongly with maximizing Heldout BLEU of predicted translations than maximizing Per-Sentence BLEU of predicted translations.

7 Related Work and Discussion

Ratings provided by humans can be used as effective learning signals for machines. Reinforcement learning has become the *de facto* standard for incorporating this feedback across diverse tasks such as robot voice control (Tenorio-Gonzalez et al.,



(a) Granularity (b) Variance (c) Skew

(d) Granularity (e) Variance (f) Skew

Figure 5: Performance gains of NMT models trained with NED-A2C in Per-Sentence BLEU (top row) and in Heldout BLEU (bottom row) under various degrees of granularity, variance, and skew of scores. Performance gains of models trained with un-perturbed scores are within the shaded regions.

2010), myoelectric control (Pilarski et al., 2011), and virtual assistants (Isbell et al., 2001). Recently, this learning framework has been combined with recurrent neural networks to solve machine translation (Bahdanau et al., 2017), dialogue generation (Li et al., 2016), neural architecture search (Zoph and Le, 2017), and device placement (Mirhoseini et al., 2017). Other approaches to more general structured prediction under bandit feedback (Chang et al., 2015; Sokolov et al., 2016a,b) show the broader efficacy of this framework. Ranzato et al. (2016) describe MIXER for training neural encoder-decoder models, which is a reinforcement learning approach closely related to ours but requires a policy-mixing strategy and only uses a linear critic model. Among work on bandit MT, ours is closest to Kreutzer et al. (2017), which also tackle this problem using neural encoder-decoder models, but we (a) take advantage of a state-of-the-art reinforcement learning method; (b) devise a strategy to simulate noisy rewards; and (c) demonstrate the robustness of our method on noisy simulated rewards.

Our results show that bandit feedback can be an effective feedback mechanism for neural machine translation systems. This is *despite* that errors in human annotations hurt machine learning models in many NLP tasks (Snow et al., 2008). An obvious question is whether we could extend our framework to model individual annotator preferences (Passonneau and Carpenter, 2014) or learn personalized models (Mirkin et al., 2015; Rabinovich et al., 2017), and handle heteroscedastic noise (Park, 1966; Kersting et al., 2007; Antos et al., 2010). Another direction is to apply active learning techniques to reduce the sample complexity required to improve the systems or to extend to richer action spaces for problems like simultaneous translation, which requires prediction (Grissom II et al., 2014) and reordering (He et al., 2015) among other strategies to both minimize delay and effectively translate a sentence (He et al., 2016).

Acknowledgements

Many thanks to Yvette Graham for her help with the WMT human evaluations data. We thank CU CLEAR and UMD CLIP lab members for useful discussions that led to the ideas of this paper. We also thank the anonymous reviewers for their thorough and insightful comments. This work was supported by NSF grants IIS-1320538. Boyd-Graber is also partially supported by NSF grants IIS- 1409287, IIS-1564275, IIS-IIS-1652666, and NCSE-1422492. Daumé III is also supported by NSF grant IIS-1618193, as well as an Amazon Research Award. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor(s).

References

Gediminas Adomavicius and Jingjing Zhang. 2012. Impact of data characteristics on recommender systems performance. *ACM Transactions on Management Information Systems (TMIS)* 3(1):3.

András Antos, Varun Grover, and Csaba Szepesvári. 2010. Active learning in heteroscedastic noise. *Theoretical Computer Science* 411(29-30):2712–2728.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations (ICLR)*.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Conference of the European Association for Machine Translation (EAMT)*. Trento, Italy.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2015. The IWSLT 2015 evaluation campaign. In *International Workshop on Spoken Language Translation (IWSLT)*.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign, IWSLT 2014. In *International Workshop on Spoken Language Translation (IWSLT)*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Pi-Chuan Chang, Michel Galley, and Chris Manning. 2008. Optimizing chinese word segmentation for machine translation performance. In *Workshop on Machine Translation*.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. In *Association for Computational Linguistics (ACL)*.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2017. Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering* 23(1):3–30.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

He He, Jordan Boyd-Graber, and Hal Daumé III. 2016. Interpretese vs. translationese: The uniqueness of human strategies in simultaneous interpretation. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

He He, Alvin Grissom II, Jordan Boyd-Graber, and Hal Daumé III. 2015. Syntax-based rewriting for simultaneous machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Jonathan L Herlocker, Joseph A Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *ACM Conference on Computer Supported Cooperative Work*.

Chang Hu, Philip Resnik, and Benjamin B Bederson. 2014. Crowdsourced monolingual translation. *ACM Transactions on Computer-Human Interaction (TOCHI)* 21(4):22.

Charles Isbell, Christian R Shelton, Michael Kearns, Satinder Singh, and Peter Stone. 2001. A social reinforcement learning agent. In *International Conference on Autonomous Agents (AA)*.

Sham M Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. 2008. Efficient bandit algorithms for online multiclass prediction. In *International Conference on Machine learning (ICML)*.

Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard. 2007. Most likely heteroscedastic gaussian process regression. In *International Conference on Machine Learning (ICML)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Association for Computational Linguistics (ACL)*.

Vijay R Konda and John N Tsitsiklis. ???? Actor-critic algorithms. In *Advances in Neural Information Processing Systems (NIPS)*.

Julia Kreutzer, Artem Sokolov, and Stefan Riezler. 2017. Bandit structured prediction for neural sequence-to-sequence learning. In *Association of Computational Linguistics (ACL)*.

John Langford and Tong Zhang. 2008. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in Neural Information Processing Systems (NIPS)*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of Psychology* 22(140):1–55.

Robert Loftin, James MacGlashan, Michael L Littman, Matthew E Taylor, and David L Roberts. 2014. A strategy-aware technique for learning behaviors from discrete human feedback. Technical report, North Carolina State University. Dept. of Computer Science.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Azalia Mirhoseini, Hieu Pham, Quoc V Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. 2017. Device placement optimization with reinforcement learning. In *International Conference on Machine Learning (ICML)*.

Shachar Mirkin, Scott Nowson, Caroline Brun, and Julien Perez. 2015. Motivating personality-aware machine translation. In *The 2015 Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*.

Rolla E Park. 1966. Estimation with heteroscedastic error terms. *Econometrica* 34(4):888.

Rebecca J Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics (TACL)* 2:311–326.

Patrick M Pilarski, Michael R Dawson, Thomas Degris, Farbod Fahimi, Jason P Carey, and Richard S Sutton. 2011. Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning. In *IEEE International Conference on Rehabilitation Robotics (ICORR)*.

Carolyn C Preston and Andrew M Colman. 2000. Optimal number of response categories in rating scales: reliability, validity, discriminating power, and respondent preferences. *Acta Psychologica* 104(1):1–15.

Ella Rabinovich, Shachar Mirkin, Raj Nath Patel, Lucia Specia, and Shuly Wintner. 2017. Personalized machine translation: Preserving original author traits. *Association for Computational Linguistics (ACL)* .

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *International Conference on Learning Representations (ICLR)* .

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Artem Sokolov, Julia Kreutzer, Christopher Lo, and Stefan Riezler. 2016a. Learning structured predictors from bandit feedback for interactive NLP. In *Association for Computational Linguistics (ACL)*.

Artem Sokolov, Julia Kreutzer, and Stefan Riezler. 2016b. Stochastic structured prediction under bandit feedback. In *Advances In Neural Information Processing Systems (NIPS)*.

Artem Sokolov, Stefan Riezler, and Tanguy Urvoy. 2015. Bandit structured prediction for learning from partial feedback in statistical machine translation. In *In Proceedings of MT Summit XV. Miami, FL*.

Ana C Tenorio-Gonzalez, Eduardo F Morales, and Luis Villaseñor-Pineda. 2010. Dynamic reward shaping: training a robot by voice. In *Ibero-American Conference on Artificial Intelligence*. Springer, pages 483–492.

Andrea L Thomaz and Cynthia Breazeal. 2008. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence* 172(6-7):716–737.

Andrea Lockerd Thomaz, Cynthia Breazeal, et al. 2006. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Barret Zoph and Quoc V. Le. 2017. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

Towards Compact and Fast Neural Machine Translation Using a Combined Method

Xiaowei Zhang[1,2], **Wei Chen**[1*], **Feng Wang**[1,2], **Shuang Xu**[1] and **Bo Xu**[1]
[1]Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, China
{zhangxiaowei2015,w.chen,feng.wang,shuang.xu,boxu}@ia.ac.cn

Abstract

Neural Machine Translation (NMT) lays intensive burden on computation and memory cost. It is a challenge to deploy NMT models on the devices with limited computation and memory budgets. This paper presents a four stage pipeline to compress model and speed up the decoding for NMT. Our method first introduces a compact architecture based on convolutional encoder and weight shared embeddings. Then weight pruning is applied to obtain a sparse model. Next, we propose a fast sequence interpolation approach which enables the greedy decoding to achieve performance on par with the beam search. Hence, the time-consuming beam search can be replaced by simple greedy decoding. Finally, vocabulary selection is used to reduce the computation of softmax layer. Our final model achieves 10× speedup, 17× parameters reduction, <35MB storage size and comparable performance compared to the baseline model.

1 Introduction

Neural Machine Translation (NMT) (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015) has recently gained popularity in solving the machine translation problem. Although NMT has achieved state-of-the-art performance for several language pairs (Jean et al., 2015; Wu et al., 2016), like many other deep learning domains, it is both computationally intensive and memory intensive. This leads to a challenge of deploying NMT models on the devices with limited computation and memory budgets.

Numerous approaches have been proposed for compression and inference speedup of neural networks, including but not limited to low-rank approximation (Denton et al., 2014), hash function (Chen et al., 2015), knowledge distillation (Hinton et al., 2015), quantization (Courbariaux et al., 2015; Han et al., 2016; Zhou et al., 2017) and sparsification (Han et al., 2015; Wen et al., 2016).

Weight pruning and knowledge distillation have been proved to be able to compress NMT models (See et al., 2016; Kim and Rush, 2016; Freitag et al., 2017). The above methods reduce the parameters from a global perspective. However, embeddings dominate the parameters in a relatively compact NMT model even if subword (Sennrich et al., 2016) (typical about 30K) is used. Character-aware methods (Ling et al., 2015; Lee et al., 2016) have fewer embeddings while suffer from slower decoding speed (Wu et al., 2016). Recent work by Li et al. (2016) has shown that weight sharing can be adopted to compress embeddings in language model. We are interested in applying embeddings weight sharing to NMT.

As for decoding speedup, Gehring et al. (2016); Kalchbrenner et al. (2016) tried to improve the parallelism in NMT by substituting CNNs for RNNs . Kim and Rush (2016) proposed sequence-level knowledge distillation which allows us to replace beam search with greedy decoding. Gu et al. (2017) exploited trainable greedy decoding by the actor-critic algorithm (Konda and Tsitsiklis, 2002). Wu et al. (2016) evaluated the quantized inference of NMT. Vocabulary selection (Jean et al., 2015; Mi et al., 2016; L'Hostis et al., 2016) was commonly used to speed up the softmax layer. Search pruning was also applied to speed up beam search (Hu et al., 2015; Wu et al., 2016; Freitag and Al-Onaizan, 2017). Compared to search pruning, the speedup of greedy decoding is more attractive. Knowledge distillation improves the per-

[*]Corresponding author

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1475–1481
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

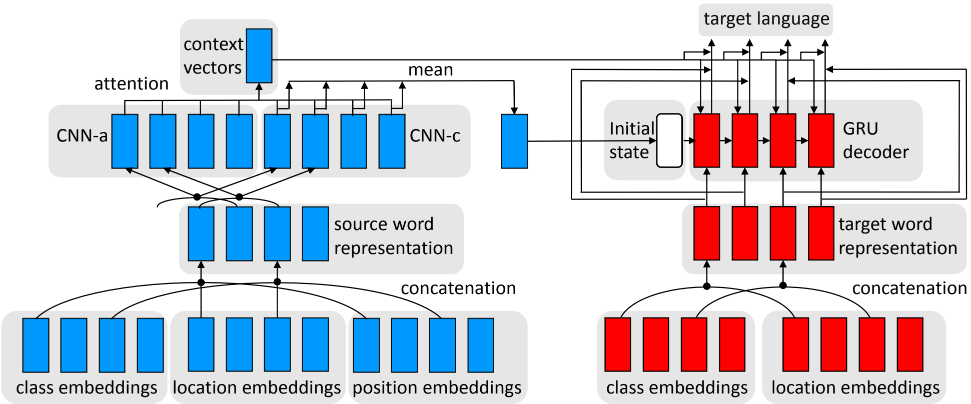


Figure 1: Our network architecture. The 2-component word representation contains class embeddings and location embeddings. Position embeddings are concatenated to convey the absolute positional information of each source word.

formance of greedy decoding while the method needs to run beam search over the training set, and therefore results in inefficiency for tens of millions of corpus. The trainable greedy decoding using a relatively sophisticated training procedure. We prefer a simple and fast approach that allows us to replace beam search with the greedy search.

In this work, a novel approach is proposed to improve the performance of greedy decoding directly and the embeddings weight sharing is introduced into NMT. We investigate the model compression and decoding speedup for NMT from the views of network architecture, sparsification, computation and search strategy, and test the performance of their combination. Specifically, we present a four stage pipeline for model compression and decoding speedup. Firstly, we train a compact NMT model based on convolutional encoder and weight sharing. The convolutional encoder works well with smaller model size and is robust for pruning. Weight sharing further reduces the number of embeddings by several folds. Then weight pruning is applied to get a sparse model. Next, we propose fast sequence interpolation to improve the performance of greedy decoding directly. This approach uses batched greedy decoding to obtain samples and therefore is more efficient than Kim and Rush (2016). Finally, we use vocabulary selection to reduce the computation of the softmax layer. Our final model achieves 10× speedup, 17× parameters reduction, <35MB storage size and comparable performance compared to the baseline model.

2 Method

2.1 Compact Network Architecture

Our network architecture is illustrated in Figure 1. This architecture works well with fewer parameters, which allows us to match the performance of the baseline model at lower capacity. The convolutional encoder is similar to Gehring et al. (2016), which consists of two convolutional neural networks: *CNN-a* for attention score computation and *CNN-c* for the conditional input to be fed to the decoder. The CNNs are constructed by blocks with residual connections (He et al., 2015). We use the *relu6*[1] non-linear activation function instead of *tanh* in Gehring et al. (2016) and achieve better training stability.

To compress the embeddings, the cluster based 2-component word representation is introduced: we cluster the words into C classes by *word2vector*[2] (Mikolov et al., 2013), and each class contains up to L words. Then the conventional embedding lookup table is replaced by $C + L$ unique vectors. For each word, we first do a lookup from C class embeddings according to which cluster the word belongs, next we do another lookup from L location embeddings according to the location of the word. We concatenate the results of the two embedding lookup as the 2-component word representation. As a result, the number of embeddings is reduced from about $C \times L$ to $C + L$. Referring to Gehring et al. (2016), position embeddings are concatenated to convey the absolute positional information of each source word within a sentence.

[1] Computes *relu6*: min(max(features, 0), 6).

[2] https://code.google.com/archive/p/word2vec

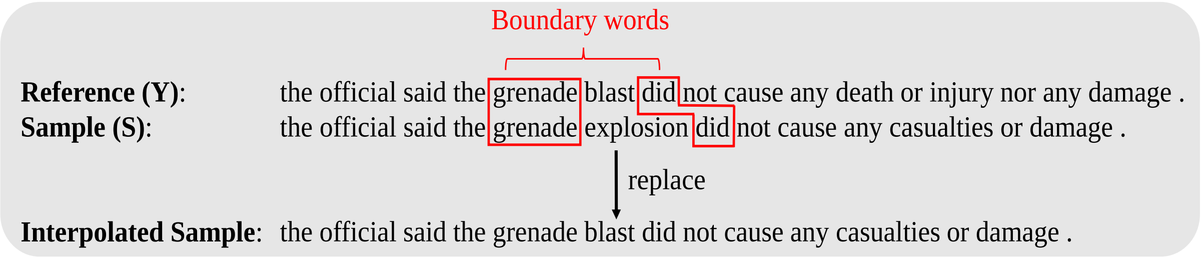

Boundary words
Reference (Y): the official said the grenade blast did not cause any death or injury nor any damage .
Sample (S): the official said the grenade explosion did not cause any casualties or damage .
replace
Interpolated Sample: the official said the grenade blast did not cause any casualties or damage .

Figure 2: Editing operation. We search for subsequences with the same boundary words between S and Y. The words within the boundary words can be different. Then we replace the subsequence in S by the subsequence in Y.

2.2 Weight Pruning

Then the iterative pruning (Han et al., 2015) is applied to obtain a sparse network, which allows us to use sparse matrix storage. In order to further reduce the storage size, most sparse matrix index of our pruned model is stored using *uint8* and *uint16* depend on the matrix dimension.

2.3 Fast Sequence Interpolation

Let (X, Y) be a source-target sequence pair. Given X as input, S is the corresponding greedy decoding result using a well trained model. Then we make two assumptions:
(1) Let $\tilde{S}$ be a sequence close to S. If training with $(X, \tilde{S})$, $\tilde{S}$ will replace S to become the result of greedy decoding with a probability $P(\tilde{S}, S)$.
(2) The following relationship holds:

$$\begin{cases} P(\tilde{S}, S) \propto sim(\tilde{S}, S) \\ sim(\tilde{S}, S) > sim(Y, S) \end{cases}$$

where sim is a function measuring closeness such as edit-distance. If $\tilde{S}$ has higher evaluation metric[3] (we write as E) than S, according to (2) we have:

$$\begin{cases} P(\tilde{S}, S) > P(Y, S) \\ E(\tilde{S}, Y) > E(S, Y) \end{cases}$$

Algorithm 1 Editing algorithm of fast sequence interpolation.

Input: (X, Y, S, k): (X, Y) is a sequence pair in training data. S is the result of the greedy decoding using source sequence X. k is the maximum number of tokens in replaced subsequence of S or Y.
Output: $\tilde{S}$: the edited sample.

```
for $s_i$ in $S$, $y_j$ in $Y$ do
    if $(s_i == y_j)$ then
        for $1 \le p \le k+1, 1 \le q \le k+1$ do
            if $(s_{i+p} == y_{j+q})$ then
                $Ss = (s_i, ..., s_p)$
                $Ys = (y_j, ..., y_q)$
                Break
            end if
        end for
    end if
    Replace subsequence of $S$: $S_s \leftarrow Y_s$
    Break
end for
$\tilde{S} \leftarrow S$
```

We note that using $\tilde{S}$ as a label is more attractive than Y for improving the performance of greedy decoding. The reason is that S and Y are often quite different (Kim and Rush, 2016), resulting in a relatively low $P(Y, S)$. We bridge the gap between S and Y by interpolating inner sequence between them. Specifically, we *edit* S toward Y, which can be seen as interpolation. *Editing* is a heuristic operation as illustrated in Figure 2. Concretely, let S_s be a subsequence of S and let Y_s be a subsequence of Y. Given that:

$$\begin{cases} S = (s_0, ..., s_p, S_s, s_q, ..., s_n) \\ Y = (y_0, ..., s_p, Y_s, s_q, ..., y_m) \\ \text{length}(S_s) \leqslant k \\ \text{length}(Y_s) \leqslant k \end{cases}$$

where k is the length limit of S_s and Y_s. The interpolated sample $\tilde{S}$ has the following form:

$$\tilde{S} = (s_0, ..., s_p, Y_s, s_q, ..., s_n)$$

To obtain the target sequence $\tilde{Y}$ for training, we substitute $\tilde{S}$ for Y according to the following rule:

$$\tilde{Y} = \begin{cases} \tilde{S} & E(\tilde{S}, Y) - E(S, Y) > \varepsilon \\ Y & \text{otherwise} \end{cases}$$

where ε aims to ensure the quality of $\tilde{S}$. We define *substitution rate* as the ratio of $\tilde{S}$ substituting Y

[3] We use smoothed sentence-level BLEU (Chen and Cherry, 2014).

over the training set. In summary, the following procedure is done iteratively: (1) get a new batch of (X, Y), (2) run batched greedy decoding on X, (3) edit S to obtain $\tilde{S}$, (4) get $\tilde{Y}$ according to the substitution rule, (5) train on the batched $(X, \tilde{Y})$.

2.4 Vocabulary Selection

We use word alignment[4] (Dyer et al., 2013) to build a candidate dictionary. For each source word, we build a list of candidate target words. When decoding, top *n* candidates of each word are merged to form a short-list for softmax layer. We do not apply vocabulary selection in training.

3 Experiments

3.1 Setup

Datasets and Evaluation Metrics: We evaluate these approaches on two pairs of languages: English-German and Chinese-English. Our English-German data comes from WMT'14[5]. The training set consists of 4.5M sentence pairs with 116M English words and 110M German words. We choose *newstest2013* as the development set and *newstest2014* as the test set. The Chinese-English training data consists of 1.6M pairs with 34M Chinese words and 38M English words. We choose *NIST 2002* as the development set and *NIST 2005* as the test set.

For the two translation task, top 50K and 30k most frequent words are kept respectively. The rest words are replaced with UNK. We only use sentences of length up to 50 symbols. We do not use any UNK handling methods for fair comparison. The evaluation metric is case-insensitive BLEU (Papineni et al., 2002) as calculated by the *multi-bleu.perl* script.

Hyper-parameters: For the baseline model, we use a 2-layer bidirectional GRU encoder (1 layer in each direction) and a 1-layer GRU decoder. In *Baseline*$_L$, the embedding size is 512 and the hidden size is 1024. In *Baseline*$_S$, the embedding size is 256 and the hidden size is 512. Our baseline models are similar to the architecture in DL4MT[6]. For the convolutional encoder model, 512 hidden units are used for the 6-layer CNN-a, and 256 hidden units are used for the 8-layer CNN-c. The embedding size is 256. The hidden size of the decoder is 512. The kernel width in CNNs is 3. The number of clusters for both source and target vocabulary is 6. The editing rule for fast sequence interpolation is detailed in Algorithm 1. We use the top 50 candidates for each source word in vocabulary selection. The initial dropout rate is 0.3, and gradually decreases to 0 as the pruning rate increases. We use AdaDelta optimizer and clip the norm of the gradient to be no more than 1. Our methods are implemented with TensorFlow[7] (Abadi et al., 2015). We run one sentence decoding for all models under the same computing environment[8].

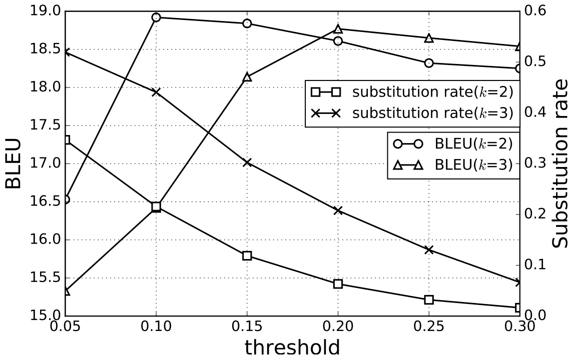


Figure 3: The performance and the *substitution rate* of *FSI* on English-German (newstest2013) development set with varying threshold ε and subsequence length limit k.

3.2 Results and Discussions

Our experimental results are summarized in Table 1. The convolutional encoder model matches the performance of the GRU encoder model with about 2× fewer parameters. Combining with embeddings weight sharing results in a compact model that has about 3.5× fewer parameters than the baseline model. After pruning 80% of the weights, we reduce the parameters by about 17× with only a decrease of 0.2 BLEU. The storage size of the final models is about 30MB, which is easily fit into the memory of a mobile device. We find that the pruning rate of embeddings is highest even if weight sharing is used. Furthermore, the pruning rate of CNN layers is higher than GRU layers. This reveals that the CNNs are more robust for pruning than RNNs. The pruning rate of each

[4] https://github.com/clab/fast_align

[5] http://statmt.org/wmt14

[6] https://github.com/nyu-dl/dl4mt-tutorial

[7] https://github.com/zxw866/CFNMT

[8] We also test batched greedy decoding with a batch size 128. We find that batched greedy decoding is nearly ten times faster than one sentence greedy decoding. We conjecture that our current one sentence decoding implementation does not fully make use of available hardware optimized for parallel computations. We can obtain a higher speedup with well optimized one sentence decoding implementation.

	English→German				*Chinese→English*			
Approach	Params	Storage	BLEU$_{k:10/1}$	T$_{\text{dec}}$	Params	Storage	BLEU$_{k:10/1}$	T$_{\text{dec}}$
Baseline$_{\text{L}}$	110m	423MB	17.84/16.23	5145	80m	305MB	32.47/28.89	1914
Baseline$_{\text{S}}$	47m	179MB	15.81/14.02	4056	31m	121MB	30.95/26.68	1412
Conv-Enc	50m	193MB	18.17/16.35	4159	35m	135MB	32.72/29.13	1437
+EWS	31m	119MB	17.85/15.89	4096	23m	90MB	32.44/28.62	1413
+Prune 80%	6m	33MB	17.63/16.02	4112	5m	25MB	32.78/28.95	1484
+FSI	6m	33MB	17.63/17.21	776	5m	25MB	32.69/31.74	297
+VS	6m	33MB	17.61/17.18	512	5m	25MB	32.63/31.67	198

Table 1: Results on English-German (newstest2014) and Chinese-English (nist05) test sets. **EWS**: embeddings weights sharing. **VS**: vocabulary selection. **FSI**: fast sequence interpolation. ***k***: beam size. **T**$_{\text{dec}}$: decoding time on the test set in seconds. Pruned models are saved as compressed sparse row (CSR) format with low bit index. Decoding runs on CPU in a preliminary implementation with TensorFlow, sparse matrix multiplication is unused for pruned models. After applying *FSI*, beam search with a beam size 10 is replaced by **greedy decoding** when recording T$_{\text{dec}}$.

Prune %	EN→DE	CN→EN
0%	17.85	32.44
50%	17.83	32.47
60%	17.91	32.55
70%	17.81	32.65
75%	17.78	32.81
80%	17.63	32.78
85%	17.36	32.84

Table 2: BLEU on test sets with varying pruning rate. Model config: *Conv Encoder+EWS*.

class number	EN→DE	CN→EN
225	15.85	30.44
10	17.83	32.78
6	18.85	34.44
4	18.96	34.60
2	19.11	34.71

Table 3: BLEU on development sets with varying class number. Model config: *Conv Encoder*.

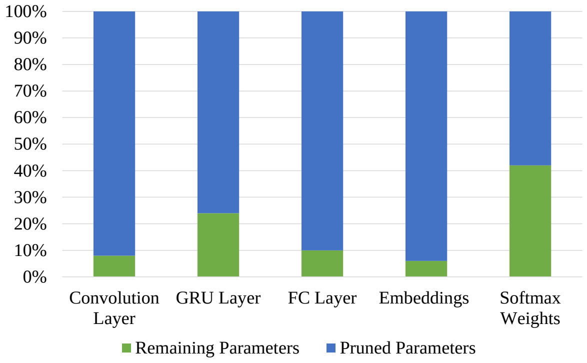


Figure 4: Pruning rate of different layers.

layer and the performance with increasing pruning rate are detailed in Figure 4. The compact architecture reduces the decoding time by only 20%. The reason is that the decoding time is dominated by the softmax layer. After applying fast sequence interpolation, we replace beam search with greedy decoding, which results in a speedup of over 5× with little loss in performance. We find that the details of the editing rules have little effect on *FSI*. Because we only accept $\tilde{S}$ that BLEU improved by more than the threshold ε, otherwise we choose the gold target sequence. Figure 3 shows that appropriate *substitution rate* is important for fast sequence interpolation. We conjecture that edited samples are still worse than gold target sequences, and therefore relatively high substitution rate may lead to instability in training. The speedup of vocabulary selection is only about 30%. It shows that the softmax layer no longer dominates the decoding time when using greedy search.

4 Conclusion and Future Work

We investigate the model compression and decoding speedup for NMT from the views of network architecture, sparsification, computation and search strategy, and verify the performance on their combination. A novel approach is proposed to improve the performance of greedy decoding and the embeddings weight sharing is introduced into NMT. In the future, we plan to integrate weight quantization into our method.

Acknowledgments

This work is supported by the National Key Research & Development Plan of China (No. 2016YFB1001404).

References

Martın Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, and et al Sanjay Ghemawat. 2015. Tensorflow: A system for large-scale machine learning. *arXiv preprint arXiv:1605.08695*.

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural machine translation by jointly learning to align and translate. *In ICLR*.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level BLEU. *In Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Wenlin Chen, James T Wilson, Stephen Tyree, Kilian Q Weinberger, and Yixin Chen. 2015. Compressing neural networks with the hashing trick. *In ICML*.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. Low precision arithmetic for deep learning. *In ICLR*.

Emily L Denton, Wojciech Zaremba, and Yann LeCun Joan Bruna, and Rob Fergus. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. *In NIPS*.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. *In NAACL*.

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. *arXiv preprint arXiv:1702.01806*.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. 2017. Ensemble distillation for neural machine translation. *arXiv preprint arXiv:1702.01802*.

Jonas Gehring, Michael Auli, David Grangierm, and Yann N. Dauphin. 2016. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611 02344*.

Jiatao Gu, Kyunghyun Cho, and Victor O.K.Li. 2017. Trainable greedy decoding for neural machine translation. *arXiv preprint arXiv:1702.02429*.

Song Han, Huizi Mao, and William J Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *In ICLR*.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural networks. *In NIPS*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *In CVPR*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *In NIPS Workshop*.

Xiaoguang Hu, Wei Li, Xiang Lan, Hua Wu, and Haifeng Wang. 2015. Improved beam search with constrained softmax for NMT. *In MT Summit*.

S. Jean, K. Cho, R. Memisevic, and Y. Bengio. 2015. On using very large target vocabulary for neural machine translation. *In ACL*.

Kalchbrenner and Blunsom. 2013. Recurrent continuous translation models. *In EMNLP*.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. *In EMNLP*.

VR Konda and JN Tsitsiklis. 2002. On actor-critic algorithms. *Siam Journal on Control & Optimization*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*.

Gurvan L'Hostis, David Grangier, and Michael Auli. 2016. Vocabulary selection strategies for neural machine translation. *arXiv preprint arXiv:1610.00072*.

Xiang Li, Tao Qin, Jian Yang, and Tie-Yan Liu. 2016. LightRNN: Memory and computation-efficient recurrent neural networks. *In NIPS*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Vocabulary manipulation for neural machine translation. *In ACL*.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. *In ACL*.

Abigail See, Minh-Thang Luong, and Christopher D. Manning. 2016. Compression of neural machine translation models via pruning. *In CoNLL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *In ACL*.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. *In NIPS*.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. *In NIPS*.

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

Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. 2017. Incremental network quantization: towards lossless cnns with low-precision weights. *In ICLR*.

Instance Weighting for Neural Machine Translation Domain Adaptation

Rui Wang[1]**, Masao Utiyama**[1]**, Lemao Liu**[2]**, Kehai Chen**[1,3] **and Eiichiro Sumita**[1]
[1]National Institute of Information and Communications Technology (NICT)
[2]Tencent AI Lab
[3]Harbin Institute of Technology
{wangrui, mutiyama and eiichiro.sumita}@nict.go.jp
lemaoliu@gmail.com and khchen@hit.edu.cn

Abstract

Instance weighting has been widely applied to phrase-based machine translation domain adaptation. However, it is challenging to be applied to Neural Machine Translation (NMT) directly, because NMT is not a linear model. In this paper, two instance weighting technologies, i.e., sentence weighting and domain weighting with a dynamic weight learning strategy, are proposed for NMT domain adaptation. Empirical results on the IWSLT English-German/French tasks show that the proposed methods can substantially improve NMT performance by up to 2.7-6.7 BLEU points, outperforming the existing baselines by up to 1.6-3.6 BLEU points.

1 Introduction

In Statistical Machine Translation (SMT), unrelated additional corpora, known as out-of-domain corpora, have been shown not to benefit some domains and tasks, such as TED-talks and IWSLT tasks (Axelrod et al., 2011; Luong and Manning, 2015). Several Phrase-based SMT (PBSMT) domain adaptation methods have been proposed to overcome this problem of the lack of substantial data in some specific domains and languages:

i) Data selection. The main idea is to score the out-of-domain data using models trained from the in-domain and out-of-domain data, respectively. Then select training data by using these ranked scores (Moore and Lewis, 2010; Axelrod et al., 2011; Duh et al., 2013; Hoang and Sima'an, 2014a,b; Durrani et al., 2015; Chen et al., 2016).

ii) Model Linear Interpolation. Several PBSMT models, such as language models, translation models, and reordering models, individually corresponding to each corpus, are trained. These models are then combined to achieve the best performance (Sennrich, 2012; Sennrich et al., 2013; Durrani et al., 2015, 2016; Imamura and Sumita, 2016).

iii) Instance Weighting. Instance Weighting has been applied to several NLP domain adaptation tasks (Jiang and Zhai, 2007), such as POS tagging, entity type classification and especially PBSMT (Matsoukas et al., 2009; Shah et al., 2010; Foster et al., 2010; Rousseau et al., 2011; Zhou et al., 2015; Wang et al., 2016; Imamura and Sumita, 2016). They firstly score each instance/domain by using rules or statistical methods as a weight, and then train PBSMT models by giving each instance/domain the weight.

For Neural Machine Translation (NMT) domain adaptation, the sentence selection can also be used (Chen et al., 2016; Wang et al., 2017). Meanwhile, the model linear interpolation is not easily applied to NMT directly, because NMT is not a linear model. There are two methods for model combination of NMT: i) the in-domain model and out-of-domain model can be *ensembled* (Jean et al., 2015). ii) an NMT further training (fine-tuning) method (Luong and Manning, 2015). The training is performed in two steps: first, the NMT system is trained using out-of-domain data, and then further trained using in-domain data. Recently, Chu et al. (2017) make an empirical comparison of NMT further training (Luong and Manning, 2015) and domain control (Kobus et al., 2016), which applied word-level domain features to word embedding layer. This approach provides natural baselines for comparison.

To the best of our knowledge, there is no existing work concerning instance weighting in

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1482–1488
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

NMT. The main challenge is that NMT is not a liner model or combination of linear models, where the instance weight can be integrated into directly. To overcome this difficulty, we try to integrate the instance weight into NMT objective function. Two technologies, i.e., sentence weighting and domain weighting, are proposed to apply instance weighting to NMT. In addition, we also propose a dynamic weight learning strategy to tune the proposed domain weights.

2 NMT Background

An attention based NMT is a neural network that directly models the conditional probability $p(\mathbf{y}|\mathbf{x})$ of translating a source sentence, $\mathbf{x} = \{x_1, ..., x_n\}$, to a target sentence, $\mathbf{y} = \{y_1, ..., y_m\}$ (Luong et al., 2015):

$$p(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^{m} softmax(g(y_j|y_{j-1}, s_j, c_j)), \quad (1)$$

with g being the transformation function that outputs a vocabulary-sized vector, s_j being the RNN hidden unit and c_j being the weighted sum of source annotations $\mathbf{H}_{\mathbf{x}}$. The NMT training objective (maximize) is formulated as,

$$J = \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}} \log p(\mathbf{y}|\mathbf{x}), \quad (2)$$

where $\mathcal{D}$ is the parallel training corpus.

3 Instance weighting for NMT

In this paper, we integrate the instance weight into the NMT objective function. Our main hypothesis is that the in-domain data should have a higher weight in the NMT objective function than the out-of-domain ones.

The training corpus $\mathcal{D}$ can be divided into in-domain one $\mathcal{D}_{in}$ and the out-of-domain one $\mathcal{D}_{out}$. So, the Eq. (2) can be rewritten as,

$$J = (\sum_{\langle \mathbf{x},\mathbf{y} \rangle \in \mathcal{D}_{in}} \log p(\mathbf{y}|\mathbf{x}) + \sum_{\langle \mathbf{x}',\mathbf{y}' \rangle \in \mathcal{D}_{out}} \log p(\mathbf{y}'|\mathbf{x}')), \quad (3)$$

where $\langle \mathbf{x}, \mathbf{y} \rangle$ is a parallel sentence pair.

3.1 Sentence Weighting

A general method is to give each sentence a weight. As Axelrod et al. (2011) mentioned, there are some pseudo in-domain data in out-of-domain data, which are close to in-domain data. We can apply their bilingual cross-entropy method to score each $\langle \mathbf{x}_i, \mathbf{y}_i \rangle$ as a weight λ_i, the higher the better,[1]

$$\begin{aligned} \lambda_i = \delta(&H_{out}(\mathbf{x}_i) - H_{in}(\mathbf{x}_i) \\ &+ H_{out}(\mathbf{y}_i) - H_{in}(\mathbf{y}_i)). \end{aligned} \quad (4)$$

Take $H_{in}(\mathbf{x}_i)$ as example, it indicates the cross-entropy between sentence $\mathbf{x}_i$ and in-domain language model (Axelrod et al., 2011). Min-max normalization δ (Priddy and Keller, 2005) is used to normalize each λ_i into range $[0, 1]$,

$$\delta(\lambda_i) = \frac{\lambda_i - \lambda_{min}}{\lambda_{max} - \lambda_{min}}. \quad (5)$$

The λ for in-domain data will set as one directly. The updated objective function by sentence weighting (J_{sw}) can be rewritten as,

$$J_{sw} = \sum_{\langle \mathbf{x}_i,\mathbf{y}_i \rangle \in \mathcal{D}} \lambda_i \log p(\mathbf{y}_i|\mathbf{x}_i). \quad (6)$$

3.2 Domain Weighting

An alternative way is to modify the weight of each domain in objective function. For we design a weight parameter λ_{in} for in-domain data. The updated objective function by domain weighting (J_{dw}) can be estimated as,

$$J_{dw} = \lambda_{in} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}_{in}} logp(\mathbf{y}|\mathbf{x}) + \sum_{(\mathbf{x}',\mathbf{y}') \in \mathcal{D}_{out}} logp(\mathbf{y}'|\mathbf{x}'). \quad (7)$$

3.2.1 Batch weighting

A straightforward domain weighting implementation is to modify the ratio between in-domain and out-of-domain data in each NMT mini-batch. That is, we can increase the in-domain weight by increasing the number of in-domain sentences included in a mini-batch. The updated in-domain data ratio $\mathcal{R}_{in}$ in each NMT mini-batch can be calculated as,

$$\mathcal{R}_{in} = \frac{|\hat{\mathcal{D}}_{in}|}{|\hat{\mathcal{D}'}_{in}| + |\hat{\mathcal{D}'}_{out}|} = \frac{\lambda_{in}}{\lambda_{in} + 1}, \quad (8)$$

where $|\hat{\mathcal{D}}_{in}|$ and $|\hat{\mathcal{D}}_{out}|$ are the sentence number from in and out-of-domain data in each mini-batch, respectively.

[1] The original cross-entropy is the lower the better, and we swap the in and out order.

Take the IWSLT EN-DE corpus in Table 1 as example, the original ratio $\mathcal{R}_{in}$ between in-domain data and all of the data is around 1:20. That is, for a 80-sized mini-batch, it would include around four sentence from in-domain data and 76 from out-of-domain data on average. For batch weighting, we can set the ratio $\mathcal{R}_{in}$ as 1:2 manually. That is, we load 40 in-domain and 40 out-of-domain sentences into each mini-batch.

In practice, we create two data iterators, one for in-domain and one for out-of-domain. Both of the in and out-of-domain data will be randomly shuffled and then loaded into corresponding data iterators before each epoch. For each mini-batch, the data from these two data iterators are determined by the ratio $\mathcal{R}_{in}$. Because the size of out-of-domain data is much larger than the in-domain one, the in-domain data will be loaded and trained for several epochs, while the out-of-domain data is only trained for one epoch at the same time.

3.2.2 Dynamic Weight Tuning

For the batch weight tuning, one way is to fix the weights for several systems and select the best-performed system on the development data. Besides this, we also tried to learn the batch weighting dynamically. That is, the initial in-domain data ration in mini-batch is set as 0%. We increased 10% ratio of in-domain data in the mini-batch if the training cost does not decrease for ten-time evaluations (the training cost is evaluated on development data set every 1K batches training).

4 Experiments

4.1 Data Sets

The proposed methods were evaluated by adapting WMT corpora to IWSLT (mainly contains TED talks) corpora.[2] Statistics on data sets were shown in Table 1.

- IWSLT 2015 English (EN) to German (DE) training corpus (Cettolo et al., 2015) was used as in-domain training data. Out-of-domain corpora contained WMT 2014 English-German corpora. This adaptation corpora settings were the same as those used in (Luong and Manning, 2015).
- IWSLT 2014 English (EN) to French (FR) training corpus (Cettolo et al., 2014) was used as in-domain training data. Out-of-domain corpora contained WMT 2015 English-French corpora. This adaptation corpora settings were nearly the same as those used in (Wang et al., 2016).

IWSLT EN-DE	Sentences	Tokens
TED training (in-domain)	207.1K	3.2M
WMT training (out-of-domain)	4.5M	119.9M
TED tst2012 (development)	1.7K	29.2K
TED tst2013 (test)	0.9K	19.6K
TED tst2014 (test)	1.3K	23.8K
IWSLT EN-FR	**Sentences**	**Tokens**
TED training (in-domain)	178.1K	3.5M
WMT training (out-of-domain)	17.8M	450.0M
TED dev2010 (development)	0.9K	20.1K
TED tst2010 (test)	1.6K	31.9K
TED tst2011 (test)	0.8K	21.4K

Table 1: Statistics on data sets.

4.2 NMT Systems

We implemented the proposed method in Nematus[3] (Sennrich et al., 2017) and online available[4], which is one of the state-of-the-art NMT frameworks. The default settings of Nematus were applied to all NMT systems (both baselines and the proposed methods): the word embedding dimension was 620 and the size of a hidden layer was 1000, the batch size was 80, the maximum sequence length were 50, and the beam size for decoding was 10. The 30K-sized vocabulary, which was created by using both in and out-of-domain data, were applied to all of the systems. Default dropout was applied. Each NMT model was trained for 500K batches by using ADADELTA optimizer (Zeiler, 2012). Training was conducted on a single Tesla P100 GPU, taking 7-10 days. We observed that all of the systems converged before 500K batches training.

For the coding cost of duplicating data, we only add two data iterators as mentioned in 3.2.1. For the training cost, using batch weighting can a accelerate the model converge on development data in our experiments, because the development data are also in-domain data. Overall, the overhead cost is not too much.

[2] In practice, we also also evaluated on the Chinese-to-English NIST task. Due to limited time and space, we only showed the IWSLT task.

[3] https://github.com/EdinburghNLP/nematus

[4] https://github.com/wangruinlp/nmt_instance_weighting The batch weighting part was partially motivated by Nematus.

4.3 Results and Analysis

In Tables 2 and 3, SMT indicates standard PBSMT (Koehn et al., 2007) models were trained by corresponding corpora (*in*, *out*, and *in+out*). The *in*, *out* and *in + out* indicate that the in-domain, out-of-domain and their mixture were used as the NMT training corpora.

For related NMT domain adaptation baselines, “ensemble” indicates *in* and *out* models were ensembled in decoding and “sampler” indicates that we sampled duplicated in-domain data into training data, to make the ratio between *in*/*out* be 1:1 manually. Actually, if the mini-batch size was as large as the whole corpus, the sampling method, and batch weighting method would be the same. Batch weighting method makes the data more balanced in each single mini-batch. However, the mini-batch size is limited, so these two methods are different.

We also compared Axelrod et al. (2011)’s sentence selection and Kobus et al. (2016)’s domain control method, which added a word feature (in or out) to each word in the training corpora. For all of the baselines, we tried our best to re-implemented their methods. The translation performance was measured by the case-insensitive BLEU (Papineni et al., 2002), with the paired bootstrap re-sampling test (Koehn, 2004)[5].

IWSLT EN-DE	tst2012	tst2013	tst2014
SMT (in)	20.70	21.01	18.50
SMT (out)	18.82	18.12	16.85
SMT (in + out)	20.04	20.23	17.08
in	23.07	25.40	21.45
out	18.87	21.23	17.07
in + out	21.31	23.54	19.41
ensemble (in + out)	**24.34**	**25.83**	**22.50**
sampler	23.37	25.22	21.91
Kobus et al. (2016)	23.23	25.70	22.03
Axelrod et al. (2011)	23.87	25.52	22.41
sentence weighting	23.46	26.26+	22.51
domain weighting	23.55	25.47	21.45
batch weighting (bw)	25.33++	27.45++	23.68++
bw + dynamic tuning	**26.03++**	**28.58++**	**24.12++**

Table 2: IWSLT EN-DE results. The marks (the same in Tables 3) indicate whether the proposed methods were significantly better than the best performed baselines in bold (“++”: better at significance level $\alpha = 0.01$, “+”: $\alpha = 0.05$).

IWSLT EN-FR	dev2010	tst2010	tst2011
SMT (in)	27.35	31.06	32.50
SMT (out)	26.26	30.04	29.29
SMT (in + out)	27.16	30.00	30.26
in	27.66	32.11	35.22
out	24.93	29.60	32.27
in + out	25.14	29.94	33.50
ensemble (in + out)	28.48	33.63	37.67
sampler	**28.67**	**34.12**	38.08
Kobus et al. (2016)	27.87	33.81	37.44
Axelrod et al. (2011)	27.85	34.03	**38.30**
sentence weighting	29.14+	34.80+	38.73
domain weighting	29.05	34.72+	39.06+
batch weighting(bw)	29.81++	35.54++	39.48++
bw + dynamic tuning	**30.40++**	**36.50++**	**41.90++**

Table 3: IWSLT EN-FR results.

In Tables 2 and 3, we reached the following observations:

- Adding out-of-domain to in-domain data, or directly using out-of-domain data, degraded NMT performance.
- The proposed instance weighting methods substantially improved NMT performance (in) up to 2.7-6.7 BLEU points, and outperformed the best existing baselines up to 1.6-3.6 BLEU points.
- Among the proposed methods, batch weighting performed the best, although it was the simplest one. The reason may be: a) the batch weighting method directly balanced the in-domain data ratio in each mini-batch, to overcome the in-domain data sparse problem. b) The batch weight can be tuned on development data, in comparison with sentence weighting method, whose weights were learned and fixed before NMT training.
- The dynamic weight tuning strategy outperformed the fixed weight tuning strategy by 0.6-2.4 BLEU points.

5 Discussions

5.1 Weights Tuning

Figure 1 showed the batch weight tuning experiments on development data of IWSLT EN-DE, where the horizontal axis indicates the in-domain ratio $\mathcal{R}_{in}$ in Eq. (8). “Fix” indicates several systems were trained with fixed weights and the best-performed system would be selected. “Dynamic” indicates that only one system was trained and the domain weight was learned dynamically as mentioned in Section 3.2.2.

[5] http://www.ark.cs.cmu.edu/MT

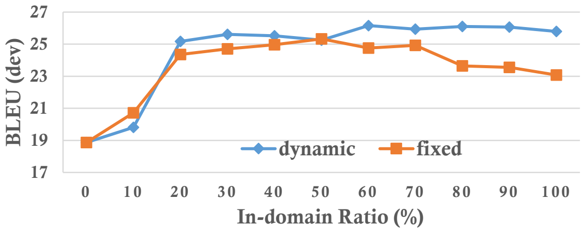


Figure 1: Batch weight tuning on IWSLT EN-DE.

As shown in Figure 1, the fixed weight learning reached the highest BLEU on dev at around 50% and dynamic learning at around 60%. If we keep training the dynamic learning after 100% in-domain data were used, the performance would trend to become similar to only using in-domain data from the beginning.

5.2 Further Training

Further training (Luong and Manning, 2015) can be viewed as a special case of the proposed batch weighting method. That is, it trained NMT model by using 0% in-domain data at first and then using 100% in-domain data. In comparison, our batch weighting kept some ratio of out-of-domain data during the whole training process. In addition, further training can work together with batch weighting. That is, NMT was trained with 0% in-domain data at first and then with batch weighing method for further training (Luong + bw in Table 4). . $\mathcal{R}_{in}$ was tuned on development data. As mentioned in Section 3.2.2, “bw + dynamic tuning” indicates that this batch weighting was learned dynamically.

IWSLT EN-DE	tst2012	tst2013	tst2014
Luong	25.68	28.14	24.31
Luong + bw	25.87	28.54+	**24.53**
bw + dynamic tuning	**26.03**	**28.58+**	24.12
IWSLT EN-FR	dev2010	tst2010	tst2011
Luong	29.33	35.36	40.62
Luong + bw	29.65	35.65	41.20+
bw + dynamic tuning	**30.40++**	**36.50++**	**41.90++**

Table 4: Further training (Luong and Manning, 2015) is the baseline for significance test.

Table 4 shows that batch weighting worked synergistically with Luong’s further training method, and slightly improved NMT performance. The “bw + dynamic tuning” method outperformed both of them. We observed that the original further training overfitted quickly after around one epoch training. Keeping some out-of-domain data would prevent further training from overfitting.

6 Conclusion and Future Work

In this paper, we proposed two straightforward instance weighting methods with a dynamic weight learning strategy for NMT domain adaptation. Empirical results on IWSLT EN-DE/FR tasks showed that the proposed methods can substantially improve NMT performances and outperform state-of-the-art NMT adaptation methods.

The current sentence weighting method is a simple implementation of the existing PBSMT adaptation methods. In the future, we will try to study a specific sentence weighting method for NMT domain adaptation.

Acknowledgments

We are grateful to the Dr. Andrew Finch, Dr. Atsushi Fujita and three anonymous reviewers for their insightful comments and suggestions. This work is partially supported by the program “Promotion of Global Communications Plan: Research, Development, and Social Demonstration of Multilingual Speech Translation Technology” of MIC, Japan.

References

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 355–362, Edinburgh, Scotland, U.K.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2015. The IWSLT 2015 evaluation campaign. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 2–14, Da Nang, Vietnam.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 2–17, Lake Tahoe, CA, USA.

Boxing Chen, Roland Kuhn, George Foster, Colin Cherry, and Fei Huang. 2016. Bilingual methods for adaptive training data selection for machine translation. In *The Twelfth Conference of The Association for Machine Translation in the Americas*, pages 93–106, Austin, Texas.

Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of simple domain adaptation methods for neural machine translation. *arXiv preprint arXiv:1701.03214*.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 678–683, Sofia, Bulgaria.

Nadir Durrani, Hassan Sajjad, Shafiq Joty, and Ahmed Abdelali. 2016. A deep fusion model for domain adaptation in phrase-based MT. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3177–3187, Osaka, Japan.

Nadir Durrani, Hassan Sajjad, Shafiq Joty, Ahmed Abdelali, and Stephan Vogel. 2015. Using joint models for domain adaptation in statistical machine translation. In *Proceedings of MT Summit XV*, pages 117–130, Miami, FL, USA.

George Foster, Cyril Goutte, and Roland Kuhn. 2010. Discriminative instance weighting for domain adaptation in statistical machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 451–459, Cambridge, MA.

Cuong Hoang and Khalil Sima'an. 2014a. Latent domain phrase-based models for adaptation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 566–576, Doha, Qatar.

Cuong Hoang and Khalil Sima'an. 2014b. Latent domain translation models in mix-of-domains haystack. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1928–1939, Dublin, Ireland.

Kenji Imamura and Eiichiro Sumita. 2016. Multi-domain adaptation for statistical machine translation based on feature augmentation. In *Proceedings of the 12th Conference of the Association for Machine Translation in the Americas*, Austin, Texas, USA.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, Prague, Czech Republic.

Catherine Kobus, Josep Crego, and Jean Senellart. 2016. Domain control for neural machine translation. *arXiv preprint arXiv:1612.06140*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*, pages 76–79, Da Nang, Vietnam.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal.

Spyros Matsoukas, Antti-Veikko I. Rosti, and Bing Zhang. 2009. Discriminative corpus weight estimation for machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 708–717, Singapore.

Robert C Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 220–224, Uppsala, Sweden.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania.

Kevin L. Priddy and Paul E. Keller. 2005. *Artificial Neural Networks: An Introduction (SPIE Tutorial Texts in Optical Engineering, Vol. TT68)*. SPIE-International Society for Optical Engineering.

Anthony Rousseau, Fethi Bougares, Paul Deléglise, Holger Schwenk, and Yannick Estève. 2011. Liums systems for the iwslt 2011 speech translation tasks. In *International Workshop on Spoken Language Translation*, San Francisco, USA.

Rico Sennrich. 2012. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 539–549, Avignon, France.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain.

Rico Sennrich, Holger Schwenk, and Walid Aransa. 2013. A multi-domain translation model framework for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 832–840, Sofia, Bulgaria.

Kashif Shah, Loïc Barrault, and Holger Schwenk. 2010. Translation model adaptation by resampling. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 392–399.

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichro Sumita. 2017. Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada.

Rui Wang, Hai Zhao, Bao-Liang Lu, Masao Utiyama, and Eiichiro Sumita. 2016. Connecting phrase based statistical machine translation adaptation. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3135–3145, Osaka, Japan.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Xinpeng Zhou, Hailong Cao, and Tiejun Zhao. 2015. Domain adaptation for SMT using sentence weight. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 153–163, Guangzhou, China.

Regularization techniques for fine-tuning in neural machine translation

Antonio Valerio Miceli Barone   **Barry Haddow**
Ulrich Germann   **Rico Sennrich**
School of Informatics, The University of Edinburgh
{amiceli, bhaddow, ugermann}@inf.ed.ac.uk
rico.sennrich@ed.ac.uk

Abstract

We investigate techniques for supervised domain adaptation for neural machine translation where an existing model trained on a large out-of-domain dataset is adapted to a small in-domain dataset.

In this scenario, overfitting is a major challenge. We investigate a number of techniques to reduce overfitting and improve transfer learning, including regularization techniques such as dropout and L2-regularization towards an out-of-domain prior. In addition, we introduce *tuneout*, a novel regularization technique inspired by dropout. We apply these techniques, alone and in combination, to neural machine translation, obtaining improvements on IWSLT datasets for English→German and English→Russian. We also investigate the amounts of in-domain training data needed for domain adaptation in NMT, and find a logarithmic relationship between the amount of training data and gain in BLEU score.

1 Introduction

Neural machine translation (Bahdanau et al., 2015; Sutskever et al., 2014) has established itself as the new state of the art at recent shared translation tasks (Bojar et al., 2016; Cettolo et al., 2016). In order to achieve good generalization accuracy, neural machine translation, like most other large machine learning systems, requires large amounts of training examples sampled from a distribution as close as possible to the distribution of the inputs seen during execution. However, in many applications, only a small amount of parallel text is available for the specific application domain, and it is therefore desirable to leverage larger out-domain datasets.

Owing to the incremental nature of stochastic gradient-based training algorithms, a simple yet effective approach to transfer learning for neural networks is *fine-tuning* (Hinton and Salakhutdinov, 2006; Mesnil et al., 2012; Yosinski et al., 2014): to continue training an existing model which was trained on out-of-domain data with in-domain training data. This strategy was also found to be very effective for neural machine translation (Luong and Manning, 2015; Sennrich et al., 2016b).

Since the amount of in-domain data is typically small, overfitting is a concern. A common solution is early stopping on a small held-out in-domain validation dataset, but this reduces the amount of in-domain data available for training.

In this paper, we show that we can make fine-tuning strategies for neural machine translation more robust by using several regularization techniques. We consider fine-tuning with varying amounts of in-domain training data, showing that improvements are logarithmic in the amount of in-domain data.

We investigate techniques where domain adaptation starts from a pre-trained out-domain model, and only needs to process the in-domain corpus. Since we do not need to process the large out-domain corpus during adaptation, this is suitable for scenarios where adaptation must be performed quickly or where the original out-domain corpus is not available. Other works consider techniques that jointly train on the out-domain and in-domain corpora, distinguishing them using specific input features (Daume III, 2007; Finkel and Manning, 2009; Wuebker et al., 2015). These techniques are largely orthogonal to

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1489–1494
Copenhagen, Denmark, September 7–11, 2017.

ours[1] and can be used in combination. In fact, Chu et al. (2017) successfully apply fine-tuning in combination with joint training.

2 Regularization Techniques for Transfer Learning

Overfitting to the small amount of in-domain training data that may be available is a major challenge in transfer learning for domain adaptation. We investigate the effect of different regularization techniques to reduce overfitting, and improve the quality of transfer learning.

2.1 Dropout

The first variant that we consider is fine-tuning with dropout. Dropout (Srivastava et al., 2014) is a stochastic regularization technique for neural networks. In particular, we consider "Bayesian" dropout for recurrent neural networks (Gal and Ghahramani, 2016).

In this technique, during training, the columns of the weight matrices of the neural network are randomly set to zero, independently for each example and each epoch, but with the caveat that when the same weight matrix appears multiple times in the unrolled computational graph of a given example, the same columns are zeroed.

For an arbitrary layer that takes an input vector h and computes the pre-activation vector v (ignoring the bias parameter),

$$v_{i,j} = W \cdot M_{W,i,j} \cdot h_{i,j} \tag{1}$$

where $M_{W,i,j} = \frac{1}{p}\mathrm{diag}(\mathrm{Bernoulli}^{\otimes n}(p))$ is the dropout mask for matrix W and training example i seen in epoch j. This mask is a diagonal matrix whose entries are drawn from independent Bernoulli random variables with probability p and then scaled by $1/p$. Gal and Ghahramani (2016) have shown that this corresponds to approximate variational Bayesian inference over the weight matrices considered as model-wise random variables, where the individual weights have a Gaussian prior with zero mean and small diagonal covariance. During execution we simply set the dropout masks to identity matrices, as in the standard approximation scheme.

Since dropout is not a specific transfer learning technique per se, we can apply it during fine-tuning, irrespective of whether or not the original out-of-domain model was also trained with dropout.

2.2 MAP-L2

L2-norm regularization is widely used for machine learning and statistical models. For linear models, it corresponds to imposing a diagonal Gaussian prior with zero mean on the weights. Chelba and Acero (2006) extended this technique to transfer learning by penalizing the weights of the in-domain model by their L2-distance from the weights of the previously trained out-of-domain model.

For each parameter matrix W, the penalty term is

$$L_W = \lambda \cdot \left\| W - \hat{W} \right\|_2^2 \tag{2}$$

where W is the in-domain parameter matrix to be learned and $\hat{W}$ is the corresponding fixed out-of-domain parameter matrix. Bias parameters may be regularized as well. For linear models, this corresponds to maximum a posteriori inference w.r.t. a diagonal Gaussian prior with mean equal to the out-of-domain parameters and $1/\lambda$ variance.

To our knowledge this method has not been applied to neural networks, except for a recent work by Kirkpatrick et al. (2017) which investigates a variant of it for *continual learning* (learning a new task while preserving performance on previously learned task) rather than domain adaptation. In this work we investigate L2-distance from out-of-domain penalization (MAP-L2) as a domain adaptation technique for neural machine translation.

2.3 Tuneout

We also propose a novel transfer learning technique which we call *tuneout*. Like Bayesian dropout, we randomly drop columns of the weight matrices during training, but instead of setting them to zero, we set them to the corresponding columns of the out-of-domain parameter matrices.

This can be alternatively seen as learning matrices of parameter differences between in-domain and out-of-domain models with standard dropout, starting from a zero initialization at the beginning of fine-tuning. Therefore, equation 2 becomes

$$v_{i,j} = (\hat{W} + \Delta W \cdot M_{\Delta W,i,j}) \cdot h_{i,j} \tag{3}$$

where $\hat{W}$ is the fixed out-of-domain parameter matrix and ΔW is the parameter difference matrix to be learned and $M_{\Delta W,i,j}$ is a Bayesian dropout mask.

[1] although in the special case of linear models, they are related to MAP-L2 fine-tuning.

3 Evaluation

We evaluate transfer learning on test sets from the IWSLT shared translation task (Cettolo et al., 2012).

3.1 Data and Methods

Test sets consist of transcripts of TED talks and their translations; small amounts of in-domain training data are also provided. For English-to-German we use IWSLT 2015 training data, while for English-to-Russian we use IWSLT 2014 training data. For the out-of-domain systems, we use training data from the WMT shared translation task,[2] which is considered permissible for IWSLT tasks, including back-translations of monolingual training data (Sennrich et al., 2016b), i.e., automatic translations of data available only in target language "back" into the source language[3].

We train out-of-domain systems following tools and hyperparameters reported by Sennrich et al. (2016a), using Nematus (Sennrich et al., 2017) as the neural machine translation toolkit. We differ from their setup only in that we use Adam (Kingma and Ba, 2015) for optimization. Our baseline fine-tuning models use the same hyperparameters, except that the learning rate is 4 times smaller and the validation frequency for early stopping 4 times higher. Early stopping serves an important function as the only form of regularization in the baseline fine-tuning model. We also use this configuration for the in-domain only baselines.

After some exploratory experiments for English-to-German, we set dropout retention probabilities to 0.9 for word-dropout and 0.8 for all the other parameter matrices. Tuneout retention probabilities are set to 0.6 (word-dropout) and 0.2 (other parameters). For MAP-L2 regularization, we found that a penalty of 10^{-3} per mini-batch performs best. For English-to-Russian, retention probabilities of 0.95 (word-dropout) 0.89 (other parameters) for both dropout and tuneout performed best.

The out-of-domain training data consists of about $7.92M$ sentence pairs for English-to-German and $4.06M$ sentence pairs for English-to-Russian. In-domain training data is about $206k$ sentence pairs for English-to-German and $181k$ sentence pairs for English-to-Russian. Training data is tokenized, truecased and segmented into subword units using byte-pair encoding (BPE) (Sennrich et al., 2016c).

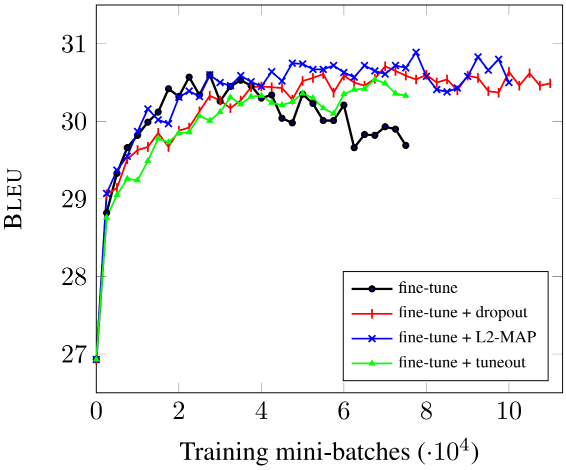


Figure 1: English→German validation BLEU over training mini-batches.

For replicability and ease of adoption, we include our implementation of dropout and MAP-L2 in the master branch of Nematus. Tuneout regularization is available in a separate code branch of Nematus.[4]

3.2 Results

We report the translation quality in terms of NIST-BLEU scores of our models in Table 1 for English-to-German and Table 2 for English-to-Russian. Statistical significance on the concatenated test sets scores is determined via bootstrap resampling (Koehn, 2004).

Dropout and MAP-L2 improve translation quality when fine-tuning both separately and in combination. When the two methods are used in combination, the improvements are significant at 5% for both language pairs, while in isolation dropout is non-significant and MAP-L2 is only significant for English-to-Russian. Tuneout does not yield improvements for English-to-German, in fact it is significantly worse, but yields a small, non-significant improvement for English-to-Russian.

In order to obtain a better picture of the training dynamics, we plot training curves[5] for several of our English-to-German models in Figure 1.

[2] http://www.statmt.org/wmt16/

[3] http://data.statmt.org/rsennrich/wmt16_backtranslations/

[4] https://github.com/EdinburghNLP/nematus/tree/tuneout-branch

[5] These BLEU scores are computed using Moses `multi-bleu.perl` which gives slightly different results than NIST `mteval-v13a.pl` that is used for Table 1.

Table 1: English-to-German translation BLEU scores

	valid	test			
System	tst2010	tst2011	tst2012	tst2013	avg
Out-of-domain only	27.19	29.65	25.78	27.85	27.76
In-domain only	25.95	27.84	23.68	25.83	25.78
Fine-tuning	30.53	32.62	28.86	32.11	31.20
Fine-tuning + dropout	30.63	33.06	28.90	32.02	31.33
Fine-tuning + MAP-L2	30.81	32.87	28.99	31.88	31.25
Fine-tuning + tuneout	30.49	32.07	28.66	31.60	30.78†
Fine-tuning + dropout + MAP-L2	30.80	**33.19**	**29.13**	**32.13**	**31.48†**

†: different from the fine-tuning baseline at 5% significance.

Table 2: English-to-Russian translation BLEU scores

	valid	test			
System	dev2010	tst2011	tst2012	tst2013	avg
Out-of-domain only	15.74	17.48	15.15	17.81	16.81
Fine-tuning	17.47	19.67	17.17	19.18	18.67
Fine-tuning + dropout	17.68	**19.96**	17.11	19.32	18.80
Fine-tuning + MAP-L2	**17.77**	19.91	17.34	19.49	18.91†
Fine-tuning + tuneout	17.51	19.72	17.27	19.35	18.78
Fine-tuning + dropout + MAP-L2	17.74	19.68	**17.83**	**19.78**	**19.10†**

†: different from the fine-tuning baseline at 5% significance.

Baseline fine-tuning starts to noticeably overfit between the second and third epoch (1 epoch $\approx 10^4$ mini-batches), while dropout, MAP-L2 and tuneout seem to converge without displaying noticeable overfitting.

In our experiments, all forms of regularization, including early stopping, have shown to be successful at mitigating the effect of overfitting. Still, our results suggest that there is value in not relying only on early stopping:

- our results suggest that multiple regularizers outperform a single one.
- if the amount of in-domain data is very small, we may want to use all of it for fine-tuning, and not hold out any for early stopping.

To evaluate different fine-tuning streategies on varying amounts of in-domain data, we tested fine-tuning with random samples of in-domain data, ranging from 10 sentence pairs to the full data set of $206k$ sentence pairs. Fine-tuning with low amounts of training data is of special interest for online adaptation scenarios where a system is fed back post-edited translation.[6] Results are shown

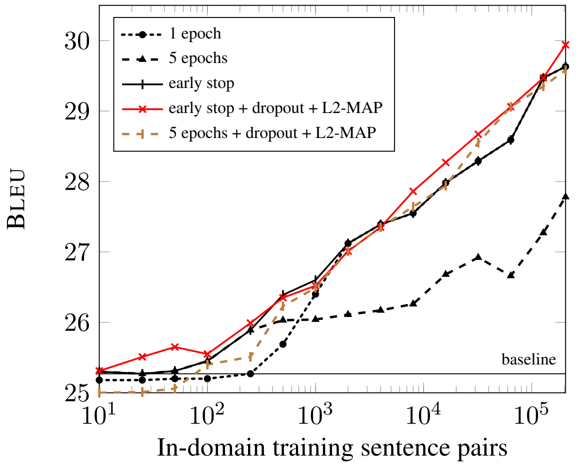


Figure 2: English→German test BLEU with fine-tuning on different in-domain data set size. Baseline trained on WMT data.

[6] We expect even bigger gains in that scenario because we would not train on a random sample, but on translations that are conceivably from the same document.

in Figure 2.

The results show an approximately logarithmic relation between the size of the in-domain training set and BLEU. We consider three baseline approaches: fine-tuning for a fixed number of epochs (1 or 5), or early stopping. All three baseline approaches have their disadvantages. Fine-tuning for 1 epoch shows underfitting on small amounts of data (less than 1,000 sentence pairs); fine-tuning for 5 epochs overfits on 500-200,000 sentence pairs. Early stopping is generally a good strategy, but it requires an in-domain held-out dataset.

On the same amount of data, regularization (dropout+MAP-L2) leads to performance that is better (or no worse) than the baseline with only early stopping. Fine-tuning with regularization is also more stable, and if we have no access to a in-domain valdiation set for early stopping, can be run for a fixed number of epochs with little or no accuracy loss.

4 Conclusion

We investigated fine-tuning for domain adaptation in neural machine translation with different amounts of in-domain training data, and strategies to avoid overfitting. We found that our baseline that relies only on early stopping has a strong performance, but fine-tuning with recurrent dropout and with MAP-L2 regularization yield additional small improvements of the order of 0.3 BLEU points for both English-to-German and English-to-Russian, while the improvements in terms of final translation accuracy of tuneout appear to be less consistent.

Furthermore, we found that regularization techniques that we considered make training more robust to overfitting, which is particularly helpful in scenarios where only small amounts of in-domain data is available, making early-stopping impractical as it relies on a sufficiently large in-domain validation set. Given the results of our experiments, we recommend using both dropout and MAP-L2 regularization for fine-tuning tasks, since they are easy to implement, efficient, and yield improvements while stabilizing training. We also present a learning curve that shows a logarithmic relationship between the amount of in-domain training data and the quality of the adapted system.

Our techniques are not specific to neural machine translation, and we propose that they could be also tried for other neural network architectures and other tasks.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements 644333 (TraMOOC) and 645487 (ModernMT). We also thank Booking.com for their support.

References

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2015. "Neural Machine Translation by Jointly Learning to Align and Translate." *Proceedings of the International Conference on Learning Representations (ICLR).*

Bojar, Ondřej, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. "Findings of the 2016 Conference on Machine Translation (WMT16)." *Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers*, 131–198. Berlin, Germany.

Cettolo, Mauro, Christian Girardi, and Marcello Federico. 2012. "WIT^3: Web Inventory of Transcribed and Translated Talks." *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, 261–268. Trento, Italy.

Cettolo, Mauro, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2016. "Report on the 13th IWSLT Evaluation Campaign." *IWSLT 2016*. Seattle, USA.

Chelba, Ciprian and Alex Acero. 2006. "Adaptation of maximum entropy capitalizer: Little data can help a lot." *Computer Speech & Language*, 20(4):382–399.

Chu, Chenhui, Raj Dabre, and Sadao Kurohashi. 2017. "An Empirical Comparison of Simple Domain Adaptation Methods for Neural Machine Translation." *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Vancouver, Canada.

Daume III, Hal. 2007. "Frustratingly Easy Domain Adaptation." *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 256–263. Prague, Czech Republic.

Finkel, Jenny Rose and Christopher D. Manning. 2009. "Hierarchical Bayesian Domain Adaptation." *Proceedings of Human Language Technologies: The*

2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, NAACL '09, 602–610. Stroudsburg, PA, USA.

Gal, Yarin and Zoubin Ghahramani. 2016. "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks." *Advances in Neural Information Processing Systems 29 (NIPS)*.

Hinton, Geoffrey E and Ruslan R Salakhutdinov. 2006. "Reducing the dimensionality of data with neural networks." *Science*, 313(5786):504–507.

Kingma, Diederik P. and Jimmy Ba. 2015. "Adam: A Method for Stochastic Optimization." *The International Conference on Learning Representations*. San Diego, California, USA.

Kirkpatrick, James, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. "Overcoming catastrophic forgetting in neural networks." *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Koehn, Philipp. 2004. "Statistical Significance Tests for Machine Translation Evaluation." *Proceedings of EMNLP 2004*, 388–395. Barcelona, Spain.

Luong, Minh-Thang and Christopher D. Manning. 2015. "Stanford Neural Machine Translation Systems for Spoken Language Domains." *Proceedings of the International Workshop on Spoken Language Translation 2015*. Da Nang, Vietnam.

Mesnil, Grégoire, Yann Dauphin, Xavier Glorot, Salah Rifai, Yoshua Bengio, Ian J Goodfellow, Erick Lavoie, Xavier Muller, Guillaume Desjardins, David Warde-Farley, et al. 2012. "Unsupervised and Transfer Learning Challenge: a Deep Learning Approach." *ICML Unsupervised and Transfer Learning*, 27:97–110.

Sennrich, Rico, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. "Nematus: a Toolkit for Neural Machine Translation." *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, 65–68. Valencia, Spain.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016a. "Edinburgh Neural Machine Translation Systems for WMT 16." *Proceedings of the First Conference on Machine Translation*, 371–376. Berlin, Germany.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016b. "Improving Neural Machine Translation Models with Monolingual Data." *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 86–96. Berlin, Germany.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016c. "Neural Machine Translation of Rare Words with Subword Units." *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1715–1725. Berlin, Germany.

Srivastava, Nitish, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, 15(1):1929–1958.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. 2014. "Sequence to Sequence Learning with Neural Networks." *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, 3104–3112. Montreal, Quebec, Canada.

Wuebker, Joern, Spence Green, and John DeNero. 2015. "Hierarchical Incremental Adaptation for Statistical Machine Translation." *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1059–1065. Lisbon, Portugal.

Yosinski, Jason, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. "How transferable are features in deep neural networks?" *Advances in neural information processing systems*, 3320–3328.

Source-Side Left-to-Right or Target-Side Left-to-Right? An Empirical Comparison of Two Phrase-Based Decoding Algorithms

Yin-Wen Chang
Google Research, New York
yinwen@google.com

Michael Collins[*]
Google Research, New York
mjcollins@google.com

Abstract

This paper describes an empirical study of the phrase-based decoding algorithm proposed by Chang and Collins (2017). The algorithm produces a translation by processing the source-language sentence in strictly left-to-right order, differing from commonly used approaches that build the target-language sentence in left-to-right order. Our results show that the new algorithm is competitive with Moses (Koehn et al., 2007) in terms of both speed and BLEU scores.

1 Introduction

Phrase-based models (Koehn et al., 2003; Och and Ney, 2004) have until recently been a state-of-the-art method for statistical machine translation, and Moses (Koehn et al., 2007) is one of the most used phrase-based translation systems. Moses uses a beam search decoder based on a dynamic programming algorithm that constructs the target-language sentence from left to right (Koehn et al., 2003). Neural machine translation systems (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014), which have given impressive improvements over phrase-based systems, also typically use models and decoders that construct the target-language string in strictly left-to-right order.

Recently, Chang and Collins (2017) proposed a phrase-based decoding algorithm that processes the *source-language* string in strictly left-to-right order. Reordering is implemented by maintaining multiple sub-strings in the target-language, with phrases being used to extend these sub-strings by various operations (see Section 2 for a full description). With a fixed distortion limit on reordering, the time complexity of the algorithm is linear in terms of sentence length, and is polynomial time in other factors.

Chang and Collins (2017) present the algorithm and give a proof of its time complexity, but do not describe experiments, leaving an open question of whether the algorithm is useful in practice. This paper complements the original paper by studying the algorithm empirically. In addition to an exact dynamic programming implementation, we study the use of beam search with the algorithm, and another pruning method that restricts the maximum number of target-language strings maintained at any point. The experiments show that the algorithm is competitive with Moses in terms of both speed and translation quality (BLEU score).

The new decoding algorithm is of interest for a few reasons. While the experiments in this paper are with phrase-based translation systems, the method could potentially be extended to neural translation, for example with an attention-based model that is in some sense monotonic (left-to-right). The decoder may be relevant to work on simultaneous translation (He et al., 2016). The ideas may be applicable to string-to-string transduction problems other than machine translation.

2 A Sketch of the Decoding Algorithm of Chang and Collins (2017)

This section gives a sketch of the decoding algorithm of Chang and Collins (2017). We first define the phrase-based decoding problem, and then describe the algorithm.

2.1 The Phrase-based Decoding Problem

Throughout this paper we will consider the following decoding problem. Given a source sentence $x_1 \dots x_n$ for $n \geq 1$, a phrase $p = (s, t, e)$ specifies a possible translation from $x_s \dots x_t$ to a string

[*] On leave from Columbia University.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1495–1499
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Sub-derivation	State
$\langle\langle(\mathbf{1},\mathbf{1},\text{<s>})\rangle\rangle$	$(1,\{(1,\text{<s>},\ 1,\text{<s>})\})$
$p=(2,3,\text{we must})$, operation = replace π_1 by CONCAT(π_1,p)	
$\langle\langle(1,1,\text{<s>})(\mathbf{2},\mathbf{3},\text{we must})\rangle\rangle$	$(3,\{(1,\text{<s>},\ 3,\text{must})\})$
$p=(4,4,\text{also})$, operation = replace π_1 by CONCAT(π_1,p)	
$\langle\langle(1,1,\text{<s>})(2,3,\text{we must})(\mathbf{4},\mathbf{4},\text{also})\rangle\rangle$	$(4,\{(1,\text{<s>},\ 4,\text{also})\})$
$p=(5,6,\text{these criticisms})$, operation = add a new sequence $\pi_2=\langle p\rangle$	
$\langle\langle(1,1,\text{<s>})(2,3,\text{we must})(4,4,\text{also})\rangle,\ \langle(\mathbf{5},\mathbf{6},\text{these criticisms})\rangle\rangle$	$(6,\{(1,\text{<s>},\ 4,\text{also}),$ $(5,\text{these},6,\text{criticisms})\})$
$p=(7,7,\text{seriously})$, operation = replace π_2 by CONCAT(π_2,p)	
$\langle\langle(1,1,\text{<s>})(2,3,\text{we must})(4,4,\text{also})\rangle,\ \langle(5,6,\text{these criticisms})(\mathbf{7},\mathbf{7},\text{seriously})\rangle\rangle$	$(7,\{(1,\text{<s>},\ 4,\text{also}),$ $(5,\text{these},7,\text{seriously})\})$
$p=(8,8,\text{take})$, operation = replace π_1,π_2 by CONCAT(π_1,p,π_2)	
$\langle\langle(1,1,\text{<s>})(2,3,\text{we must})(4,4,\text{also})(\mathbf{8},\mathbf{8},\text{take})(5,6,\text{these criticisms})(7,7,\text{seriously})\rangle\rangle$	$(8,\{(1,\text{<s>},\ 7,\text{seriously})\})$
$p=(9,9,\text{</s>})$, operation = replace π_1 by CONCAT(π_1,p)	
$\langle\langle(1,1,\text{<s>})(2,3,\text{we must})(4,4,\text{also})(8,8,\text{take})(5,6,\text{these criticisms})(7,7,\text{seriously})(\mathbf{9},\mathbf{9},\text{</s>})\rangle\rangle$	$(9,\{(1,\text{<s>},\ 9,\text{</s>})\})$

Figure 1: Illustrations of how the new algorithm produces the output translation. We note each phrase that is being added and the operation it takes to generate the next segments of phrase sequence.

of target-language words $e = e_1 \dots e_m$. We use $s(p)$, $t(p)$, and $e(p)$ to refer to the three elements of a phrase p. A *derivation* is a sequence of L phrases, $p_1 \dots p_L$. The derivation gives a translation by concatenating the target-language strings $e(p_1) \dots e(p_L)$.

We will always assume that $x_1 = \text{<s>}$, the start-of-sentence symbol, and $x_n = \text{</s>}$, the end-of-sentence symbol. The only phrases covering positions 1 and n are $(1, 1, \text{<s>})$ and $(n, n, \text{</s>})$.

A derivation $p_1 \dots p_L$ is *valid* if each word in the source sentence is translated exactly once, and if for $i = 2 \dots L$ we have $|t(p_{i-1})+1-s(p_i)| \leq d$, where d is the distortion limit.

The score for any derivation is

$$f(p_1 \dots p_L) = \lambda(e(p_1) \dots e(p_L)) + \sum_{i=1}^{L} \kappa(p_i) + \sum_{i=2}^{L} \eta \times |t(p_{i-1}) + 1 - s(p_i)|$$

where the parameter η is the distortion penalty, $\lambda(e)$ is a language model score for the word sequence e, and $\kappa(p)$ is the score for phrase p under the phrase-based model. For example under a bigram language model, we have $\lambda(e_1 \dots e_m) = \sum_{i=2}^{m} \lambda(e_i|e_{i-1})$. where $\lambda(v|u)$ is the score for bigram (u, v).

The phrase-based decoding problem is to find

$$\arg\max_{p_1 \dots p_L \in \mathcal{P}} f(p_1 \dots p_L)$$

where $\mathcal{P}$ is the set of all valid derivations for the input sentence.

2.2 The Decoding Algorithm

At a high level, the decoding algorithm of Chang and Collins (2017) differs from the commonly-used approach of Koehn et al. (2003) in two important respects:

1. The decoding algorithm proceeds in strictly left-to-right order in the *source* sentence.

2. Each sub-derivation (item) in the beam consists of *multiple* sequences of phrases, instead of a single sequence.

To be more precise, each sub-derivation in the decoding algorithm consists of:

1. An integer j specifying the length of the derivation (i.e., that words $x_1 \dots x_j$ have been translated).

2. A set of segments $\{\pi_1, \pi_2, \dots, \pi_r\}$ where $r \geq 1$. Each segment π is a sequence of phrases. The segment π_1 always has $(1, 1, \text{<s>})$ as its first element. Each word $x_1 \dots x_j$ is translated exactly once in these segments.

As one example, the sub-derivation $(1, \{\langle(1, 1, \text{<s>})\rangle\})$ is always the initial sub-derivation, with only the first word x_1 being translated, and with a single segment

$\pi_1 = \langle(1, 1, \texttt{<s>})\rangle$. A more complex sub-derivation is

$$(7, \{\langle(1, 1, \texttt{<s>})(2, 3, \text{we must})(4, 4, \text{also})\rangle,$$
$$\langle(5, 6, \text{these criticisms})(7, 7, \text{seriously})\rangle\}) \quad (1)$$

which translates words $x_1 \dots x_7$, and has two segments,

$$\pi_1 = \langle(1, 1, \texttt{<s>})(2, 3, \text{we must})(4, 4, \text{also})\rangle$$
$$\pi_2 = \langle(5, 6, \text{these criticisms})(7, 7, \text{seriously})\rangle$$

We now describe how sub-derivations can be built as the source sentence is processed in left-to-right order. A derivation $(j, \{\pi_1 \dots \pi_r\})$ can be extended as follows:

1. First select some phrase $p = (j + 1, t, e)$ where the phrase-based lexicon specifies that words $x_{j+1} \dots x_t$ can be translated as the English sequence $e = e_1 \dots e_m$.

2. Second, extend the derivation using one of the following operations (we use CONCAT to denote an operation that concatenates two or more phrase sequences):

   (a) Replace π_i for some $i \in 1 \dots r$ by $\text{CONCAT}(\pi_i, p)$.

   (b) Replace π_i for some $i \in 2 \dots r$ by $\text{CONCAT}(p, \pi_i)$.

   (c) Replace $\pi_i, \pi_{i'}$ for integers $i \neq i'$ by $\text{CONCAT}(\pi_i, p, \pi_{i'})$

   (d) Create a new segment $\pi_{r+1} = \langle p \rangle$.

Figure 1 shows the sequence of steps, and the resulting sequence of sub-derivations, in the translation of a German sentence.

A few remarks:

Remark 1. The score for each of the operations (a)-(d) described above is easily calculated using a combination of phrase, language model, and distortion scores.

Remark 2. The distortion limit can be used to rule out some of the operations (a)-(d) above, depending on the phrase p and the start/end points of each of the segments $\pi_1 \dots \pi_r$.

Remark 3. Dynamic programming can be used with this algorithm. Under a bigram language model, the dynamic programming state for a sub-derivation $(j, \{\pi_1 \dots \pi_r\})$ records the words and positions at the start and end of each segment $\pi_1 \dots \pi_r$. For example under a bigram language model the sub-derivation $(7, \dots)$ in Eq. 1 would be mapped to the dynamic-programming state $(7, \{(1, \texttt{<s>}, 4, \text{also}), (5, \text{these}, 7, \text{seriously})\})$. See Chang and Collins (2017) for more details.

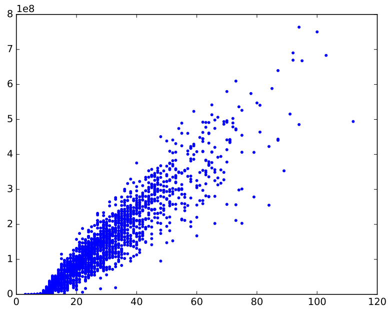


Figure 2: The total number of dynamic programming transitions and the sentence length.

Remark 4. It is simple to use beam search in conjunction with the algorithm. Different derivations of the same length j are compared in the beam. A heuristic—typically a lower-order language model—can be used to score the first $n - 1$ words in each segment $\pi_1 \cdots \pi_r$: this can be used as the "future score" for each item in the beam. This is arguably simpler than the future scores used in (Koehn et al., 2003), which have to take into account the fact that different items in the beam correspond to translations of different subsets of words in the source sentence. In our approach different derivations of the same length j have translated the same set of words $x_1 \cdots x_j$. For example in the sub-derivation $(7, \dots)$ given above (Eq. 1), and given a trigram language model, the initial bigram *these criticisms* in π_2 is scored as $p_u(\text{these}) \times p_b(\text{criticisms}|\text{these})$ where p_u and p_b are unigram and bigram language models.

3 Experiments

The original motivation for Chang and Collins (2017) was to develop a dynamic-programming algorithm for phrase-based decoding that for a fixed distortion limit d was polynomial time in other factors: the resulting dynamic programming algorithm is $O(nd!lh^{d+1})$ time, where d is the distortion limit, l is a bound on the number of phrases starting at any position, and h is related to the maximum number of different target translations for any source position. However an open question is whether the algorithm is useful in practice when used in conjunction with beam search. This

	d	SegmentD		Segment2		Moses	
		BLEU	time	BLEU	time	BLEU	time
cs-en	8	13.67[1]	5m31s	17.42	2m49s	**17.56**	3m32s
de-en	12	25.89	9m02s	**26.69**	5m25s	**26.69**	7m37s
es-en	4	32.02	4m27s	32.01	3m58s	**32.03**	4m01s
fi-en	8	23.02	5m03s	23.66	3m09s	**23.73**	3m37s
fr-en	4	31.42	4m58s	31.43	4m23s	**31.45**	4m20s
it-en	4	28.44	4m26s	**28.44**	3m57s	28.41	3m36s
nl-en	8	24.96	7m53s	25.13	5m20s	**25.16**	5m56s
pt-en	4	**31.06**	4m28s	31.05	4m00s	31.05	3m31s
sv-en	4	31.33	3m58s	**31.35**	3m30s	31.34	3m02s
vi-en	8	20.48[2]	3m39s	**20.96**	2m08s	20.95	2m40s

Figure 3: Comparison of beam search under the new decoding algorithm and the Moses decoder. We show the BLEU score and the decoding time of three beam search based decoding methods.

section describes experiments comparing beam search under the new algorithm to the method of Koehn et al. (2003). Throughout this section we refer to the algorithm of Chang and Collins (2017) as the "new" decoding algorithm.

Data. We use the Europarl parallel corpus (Version 7)[3] (Koehn, 2005) for all language pairs except for Vietnamese-English (vi-en). For Czech-English (cs-en), we use the Newstest2015 as the development set and Newstest2016 as the test set. For European languages other than Czech, we use the development and test set released for the Shared Task of WPT 2005[4]. For vi-en, we use the IWSLT'15 data.

3.1 Search Space with a Bigram Model

We first analyze the properties of the algorithm by running the exact decoding algorithm with a bigram language model and a fixed distortion limit of four, with no pruning. In Figure 2, we plot the number of transitions computed versus sentence length for translation of 2,000 German sentences to English. The figure confirms that the search space grows linearly with the number of words in the source sentence.

3.2 Beam Search under the New Algorithm

Even though the exact algorithm is linear time in the input sentence length, other factors (the dependence on d, l, and h, as described above) make the exact algorithm too costly to be useful in practice. We experiment with beam search under the new algorithm,[5] both with and without further pruning or restriction.

# segments	# sentences	percentage
1	636	34.93%
2	1,136	62.38%
3	49	2.69%

(a) The distribution of the number of segments required for the optimal solutions. Note that the distortion limit is four.

# segments	# sentences	percentage
1	119,428	15.97%
2	541,833	72.44%
3	82,869	11.08%
4	3,747	0.50%
5	128	0.02%
6	1	0.00%

(b) The distribution of the number of segments required for reordering the parsed German sentence.

Figure 4: The number of segments required for German-to-English translation.

We experimented with a *segment constraint* on the new algorithm: more specifically, we describe experiments with a hard limit $r \leq 2$ on the number of segments $\pi_1 \ldots \pi_r$ used in any translation.

Figure 3 shows results using a trigram language model for the new algorithm with beam search (SegmentD), the new algorithm with beam search and a hard limit $r \leq 2$ on the number of segments (Segment2), and Moses. A beam size of 100 is used with all the algorithms. For each language pair, we pick the distortion limit that maximizes the BLEU score for Moses. Moses was used to train all the translation models. It can be seen that the Segment2 algorithm gives very similar performance to Moses, while SegmentD has inferior performance for languges which require a larger distortion limit.

3.3 Experiments on the Number of Segments Required for German-to-English Translation

Finally, we investigate empirically how many segments (the maximum value of r) are required for translation from German to English. In a first experiment, we use the system of Chang and Collins (2011) to give exact search for German-to-English

[1] Unable to produce translations for 36 sentences.

[2] Unable to produce translations for one sentence.

[3] http://www.statmt.org/europarl/

[4] ACL 2005 Workshop on Building and Using Parallel Texts.

[5] See Section 5.1 in Chang and Collins (2017)

translation under a trigram language model with a distortion limit $d = 4$, and then look at the maximum value for r for each optimal translation. Out of 1,821 sentences, 34.9% have a maximum value of $r = 1$, 62.4% have $r = 2$, and 2.69% have $r = 3$ (Table 4a). No optimal translations require a value of r greater than 3. It can be seen that very few translations require more than 2 segments.

In a second experiment, we take the reordering system of Collins et al. (2005) and test the maximum value for r on each sentence to capture the reordering rules. Table 4b gives the results. It can be seen that over 99% of sentences require a value of $r = 3$ or less, again suggesting that for at least this language pair, a choice of $r = 3$ or $r = 4$ is large enough to capture the majority of reorderings (assuming that the rules of Collins et al. (2005) are comprehensive).

4 Conclusion

The goal of this paper was to understand the empirical performance of a newly proposed decoding algorithm that operates from left to right on the source side. We compare our implementation of the new algorithm with the Moses decoder. The experimental results demonstrate that the new algorithm combined with beam search and segment-based pruning is competitive with the Moses decoder. Future work should consider integration of the method with more recent models, in particular neural translation models.

References

Yin-Wen Chang and Michael Collins. 2011. Exact decoding of phrase-based translation models through Lagrangian relaxation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 26–37.

Yin-Wen Chang and Michael Collins. 2017. A polynomial-time dynamic programming algorithm for phrase-based decoding with a fixed distortion limit. *Transactions of the Association for Computational Linguistics* 5:59–71.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Empirical Methods in Natural Language Processing (EMNLP)*. pages 1724–1734.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 531–540.

He He, Jordan Boyd-Graber, and Hal Daumé III. 2016. Interpretese vs. translationese: The uniqueness of human strategies in simultaneous interpretation. In *North American Association for Computational Linguistics*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1700–1709.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*. volume 5, pages 79–86.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics, pages 177–180.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 48–54.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational linguistics* 30(4):417–449.

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

Using Target-side Monolingual Data for Neural Machine Translation through Multi-task Learning

Tobias Domhan and **Felix Hieber**
Amazon
Berlin, Germany
{domhant,fhieber}@amazon.com

Abstract

The performance of Neural Machine Translation (NMT) models relies heavily on the availability of sufficient amounts of parallel data, and an efficient and effective way of leveraging the vastly available amounts of monolingual data has yet to be found. We propose to modify the decoder in a neural sequence-to-sequence model to enable multi-task learning for two strongly related tasks: target-side language modeling and translation. The decoder predicts the next target word through two channels, a target-side language model on the lowest layer, and an attentional recurrent model which is conditioned on the source representation. This architecture allows joint training on both large amounts of monolingual and moderate amounts of bilingual data to improve NMT performance. Initial results in the news domain for three language pairs show moderate but consistent improvements over a baseline trained on bilingual data only.

1 Introduction

In recent years, neural encoder-decoder models (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2014) have significantly advanced the state of the art in NMT, and now consistently outperform Statistical Machine Translation (SMT) (Bojar et al., 2016). However, their success hinges on the availability of sufficient amounts of parallel data, and contrary to the long line of research in SMT, there has only been a limited amount of work on how to effectively *and efficiently* make use of monolingual data which is typically amply available. We propose a modified neural sequence-to-sequence model with attention (Bahdanau et al., 2014; Luong et al., 2015b) that uses multi-task learning on the decoder side to jointly learn two strongly related tasks: target-side language modeling and translation. Our approach does not require any pre-translation or pre-training to learn from monolingual data and thus provides a principled way to integrate monolingual data resources into NMT training.

2 Related Work

Gülçehre et al. (2015) investigate two ways of integrating a pre-trained neural Language Model (LM) into a pre-trained NMT system: shallow fusion, where the LM is used at test time to rescore beam search hypothesis, requiring no additional fine-tuning and deep fusion, where hidden states of NMT decoder and LM are concatenated before making a prediction for the next word. Both components are pre-trained separately and fine-tuned together.

More recently, Sennrich et al. (2016) have shown significant improvements by *back-translating* target-side monolingual data and using such synthetic data as additional parallel training data. One downside of this approach is the significantly increased training time, due to training of a model in the reverse direction and translation of monolingual data. In contrast, we propose to train NMT models from scratch on both bilingual and target-side monolingual data in a multi-task setting.

Our approach aims to exploit the signals from target-side monolingual data to learn a strong language model that supports the decoder in making translation decisions for the next word. Our approach further relates to Zhang and Zong (2016), who investigate multi-task learning for sequence-to-sequence models by strengthening the encoder using source-side monolingual data. A shared encoder architecture is used to predict both, transla-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1500–1505
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

tions of parallel source sentences and permutations of monolingual source sentences. In this paper we focus on target-side monolingual data and only update encoder parameters based on existing parallel data.

In a broader context, multi-task learning has shown to be effective in the context of sequence-to-sequence models (Luong et al., 2015a), where different parts of the network can be shared across multiple tasks.

3 Neural Machine Translation

We briefly recap the baseline NMT model (Bahdanau et al., 2014; Luong et al., 2015b) and highlight architectural differences of our implementation where necessary.

Given source sentence $\mathbf{x} = x_1, ..., x_n$ and target sentence $\mathbf{y} = y_1, ..., y_m$, NMT models $p(\mathbf{y}|\mathbf{x})$ as a target language sequence model, conditioning the probability of the target word y_t on the target history $\mathbf{y}_{1:t-1}$ and source sentence $\mathbf{x}$. Each x_i and y_t are integer ids given by source and target vocabulary mappings, $\mathbf{V}_{src}$, $\mathbf{V}_{trg}$, built from the training data tokens. The target sequence is factorized as:

$$p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = \prod_{t=1}^{m} p(y_t|\mathbf{y}_{1:t-1}, \mathbf{x}; \boldsymbol{\theta}). \quad (1)$$

The model, parameterized by $\boldsymbol{\theta}$, consists of an *encoder* and a *decoder* part (Sutskever et al., 2014).

For training set $\mathbb{P}$ consisting of parallel sentence pairs $(\mathbf{x}, \mathbf{y})$, we minimize the cross-entropy loss w.r.t $\boldsymbol{\theta}$:

$$\mathcal{L}_{\boldsymbol{\theta}} = \sum_{(\mathbf{x},\mathbf{y})\in\mathbb{P}} -\log p(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}). \quad (2)$$

Encoder Given source sentence $\mathbf{x} = x_1, ..., x_n$, the encoder produces a sequence of hidden states $\mathbf{h}_1 \ldots \mathbf{h}_n$ through an Recurrent Neural Network (RNN), such that:

$$\overrightarrow{\mathbf{h}}_i = f_{enc}(\mathbf{E}_S\mathbf{x}_i, \overrightarrow{\mathbf{h}}_{i-1}), \quad (3)$$

where $\mathbf{h}_0 = \mathbf{0}$, $\mathbf{x}_i \in \{0,1\}^{|\mathbf{V}_{src}|}$ is the one-hot encoding of x_i, $\mathbf{E}_S \in \mathbb{R}^{e\times|\mathbf{V}_{src}|}$ is a source embedding matrix with embedding size e, and f_{enc} some non-linear function, such as the Gated Rectified Unit (GRU) (Cho et al., 2014) or a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) network.

Attentional Decoder The decoder also consists of an RNN to predict one target word at a time through a *state vector* $\mathbf{s}$:

$$\mathbf{s}_t = f_{dec}([\mathbf{E}_T\mathbf{y}_{t-1}; \bar{\mathbf{s}}_{t-1}], \mathbf{s}_{t-1}), \quad (4)$$

where $\mathbf{y}_{t-1} \in \{0,1\}^{|\mathbf{V}_{trg}|}$ is the one-hot encoding of the previous target word, $\mathbf{E}_T \in \mathbb{R}^{e\times|\mathbf{V}_{trg}|}$ the target word embedding matrix, f_{dec} an RNN, $\mathbf{s}_{t-1}$ the previous state vector, and $\bar{\mathbf{s}}_{t-1}$ the source-dependent *attentional vector*. The initial decoder hidden state is a non-linear transformation of the last encoder hidden state: $\mathbf{s}_0 = \tanh(\mathbf{W}_{init}\mathbf{h}_n + \mathbf{b}_{init})$. The attentional vector $\bar{\mathbf{s}}_t$ combines the decoder state with a *context vector* $\mathbf{c}_t$:

$$\bar{\mathbf{s}}_t = \tanh(\mathbf{W}_{\bar{s}}[\mathbf{s}_t; \mathbf{c}_t]), \quad (5)$$

where $\mathbf{c}_t$ is a weighted sum of encoder hidden states: $\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{ti}\mathbf{h}_i$ and brackets denote vector concatenation.

The attention vector $\boldsymbol{\alpha}_t$ is computed by an attention network (Bahdanau et al., 2014; Luong et al., 2015b):

$$\alpha_{ti} = softmax(score(\mathbf{s}_t, \mathbf{h}_i))$$
$$score(\mathbf{s}, \mathbf{h}) = \mathbf{v}_a^\top \tanh(\mathbf{W}_u\mathbf{s} + \mathbf{W}_v\mathbf{h}). \quad (6)$$

The next target word y_t is predicted through a softmax layer over the attentional vector $\bar{\mathbf{s}}_t$:

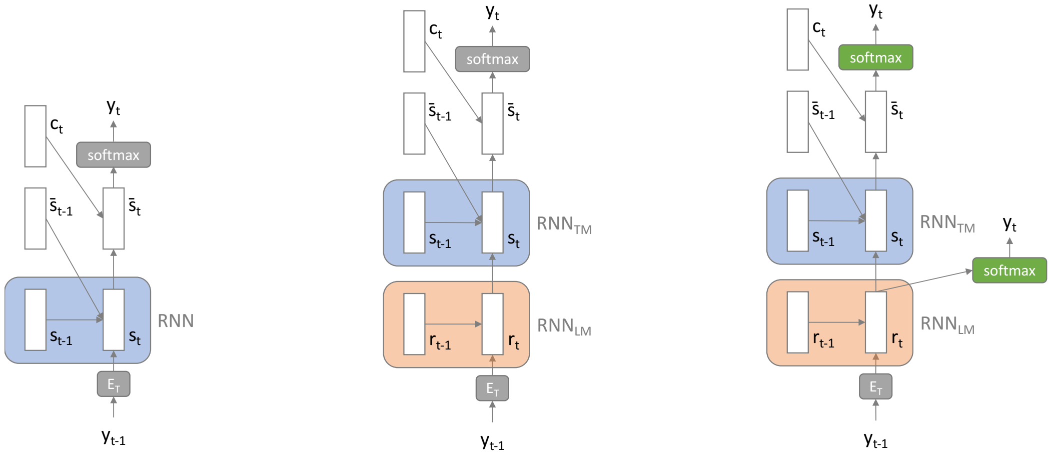
$$p(y_t|\mathbf{y}_{1:t-1}, \mathbf{x}; \boldsymbol{\theta}) = softmax(\mathbf{W}_o\bar{\mathbf{s}}_t + \mathbf{b}_o) \quad (7)$$

where $\mathbf{W}_o$ maps $\bar{\mathbf{s}}_t$ to the dimension of the target vocabulary. Figure 1a depicts this decoder architecture. Note that source information from $\mathbf{c}$ indirectly influences the states $\mathbf{s}$ of the decoder RNN as it takes $\bar{\mathbf{s}}$ as one of its inputs.

4 Incorporating Monolingual Data

4.1 Separate Decoder LM layer

The decoder RNN (Figure 1a) is essentially a target-side language model, additionally conditioned on source-side sequences. Such sequences are not available for monolingual corpora and previous work has tried to overcome this problem by either using synthetically generated source sequences or using a NULL token as the source sequence (Sennrich et al., 2016). As previously shown empirically, the model tends to "forget" source-side information if trained on much more monolingual than parallel data.



(a) baseline (b) +LML (c) +LML +MTL

Figure 1: Illustration of the proposed decoder architecture. (a) Baseline model with a single-layer decoder RNN and attention (b) Addition of a source-independent LM layer that feeds into the source-dependent decoder (c) Multi-task setting next-word prediction from both layers; green softmax layers are shared.

In our approach we explicitly define a source-independent network that only learns from target-side sequences (a language model), and a source-dependent network on top, that takes information from the source sequence into account (a translation model) through the attentional vector $\bar{\mathbf{s}}$. Formally, we modify the decoder RNN of Equation 4 to operate on the outputs an LM layer, which is independent of any source-side information:

$$\mathbf{s}_t = f_{dec}([\mathbf{r}_t; \bar{\mathbf{s}}_{t-1}], \mathbf{s}_{t-1}) \quad (8)$$

$$\mathbf{r}_t = f_{lm}(\mathbf{E}_T \mathbf{y}_{t-1}, \mathbf{r}_{t-1}) \quad (9)$$

Figure 1b illustrates this separation graphically.

4.2 Multi-task Learning

The separation from above allows us to train the target embeddings $\mathbf{E}_T$ and f_{lm} parameters from monolingual data, concurrent to training the rest of the network on bilingual data. Let us denote the source-independent parameters by $\boldsymbol{\sigma}$. We connect a second loss to f_{lm} to predict the next target word also conditioned only on target history information (Figure 1c). Parameters for softmax layers are shared such that predictions of the LM layer are given by:

$$p(y_t|\mathbf{y}_{1:t-1}, \boldsymbol{\sigma}) = softmax(\mathbf{W}_o \mathbf{r}_t + \mathbf{b}_o). \quad (10)$$

Formally, for a heterogeneous data set $\mathbb{Z} = \{\mathbb{P}, \mathbb{M}\}$, consisting of parallel and monolingual sentences $(\mathbf{x}, \mathbf{y}), (\mathbf{y})$, we optimize the following joint loss:

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\sigma}} = \frac{1}{|\mathbb{P}|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathbb{P}} -\log p(\mathbf{y}|\mathbf{x}; \boldsymbol{\theta}) + \gamma \frac{1}{|\mathbb{M}|} \sum_{\mathbf{y} \in \mathbb{M}} -\log p(\mathbf{y}; \boldsymbol{\sigma}), \quad (11)$$

where the source-independent parameters $\boldsymbol{\sigma} \subset \boldsymbol{\theta}$ are updated by gradients from both mono- and parallel data examples, and source-dependent parameters $\boldsymbol{\theta}$ are updated only through gradients from parallel data examples. $\gamma \geq 0$ is a scalar to influence the importance of the monolingual loss. In practice, we construct mini-batches of training examples, where 50% of the data is parallel, and 50% of the data is monolingual and set $\gamma = 1$.

Since parts of the decoder are shared among both tasks and we optimize both loss terms concurrently, we view this approach as an instance of multi-task learning rather than transfer learning, where optimization is typically carried out sequentially.

5 Experiments

We conduct experiments for three different language pairs in the news domain: FR→EN, EN→DE, and CS→EN.

5.1 Data

For EN→DE and CS→EN we use *news-commentary-v11* as bilingual training data, *NewsCrawl 2015* as monolingual data, and news development and test sets from

System	Data	EN→DE	FR→EN	CS→EN
baseline		20.3 39.9 63.0	21.7 27.5 59.1	17.0 24.4 65.2
+ LML		20.4 39.8 63.1	21.3 27.2 59.8	16.9 24.4 65.4
+ LML + MTL	+ mono	21.4 40.8 61.4	22.3 27.7 58.3	17.2 24.7 64.3
Sennrich et al. (2016)	+ synthetic	24.4 43.4 56.4	27.4 31.5 52.1	21.2 27.5 59.4
ensemble baseline		22.2 41.6 60.6	23.9 29.1 56.4	18.3 25.5 63.0
+ LML		22.4 41.8 60.9	23.5 28.7 57.2	18.3 25.6 63.4
+ LML + MTL	+ mono	23.6 42.8 58.9	24.2 29.2 55.9	18.8 25.9 62.2
ensemble Sennrich et al. (2016)	+ synthetic	25.7 44.6 55.0	29.1 32.6 50.3	22.5 28.4 57.8

Table 1: BLEU/METEOR/TER scores on test sets for different language pairs. For BLEU and METEOR higher is better. For TER lower is better.

WMT2016 (Bojar et al., 2016). For FR→EN we use *newscommentary-v9* as bilingual data, *NewsCrawl 2009-13* as monolingual data, and news development and test sets from WMT 2014 (Bojar et al., 2014). The number of sentences for these corpora is shown below:

Data Set	bilingual	monolingual
EN→DE	242,770	51,315,088
FR→EN	183,251	51,995,709
CS→EN	191,432	27,236,445

5.2 Experimental Setup

We tokenize all data and apply Byte Pair Encoding (BPE) (Sennrich et al., 2015) with 30k merge operations learned on the joined bilingual data. Models are evaluated in terms of BLEU (Papineni et al., 2002), METEOR (Lavie and Denkowski, 2009) and TER (Snover et al., 2006) on tokenized, cased test data. Decoding is performed using beam search with a beam of size 5. We implement all models using MXNet (Chen et al., 2015)[1].

Baselines Our baseline model consists of a 1-layer bi-directional LSTM encoder with an embedding size of 512 and a hidden size of 1024. The 1-layer LSTM decoder with 1024 hidden units uses an attention network with 256 hidden units. The model is optimized using Adam (Kingma and Ba, 2014) with a learning rate of 0.0003, no weight decay and gradient clipping if the norm exceeds 1.0. The batch size is set to 64 and the maximum sequence length to 100. Dropout (Srivastava et al., 2014) of 0.3 is applied to source word embeddings and outputs of RNN cells. We initialize all RNN parameters with orthogonal matrices (Saxe et al., 2013) and the remaining parameters with the Xavier (Glorot and Bengio, 2010) method. We use early stopping with respect to perplexity on the development set. We train each model configuration three times with different seeds and report average metrics across the three runs.

Further, we train models with synthetic parallel data generated through *back-translation* (Sennrich et al., 2016). For this, we first train a baseline model in the reverse direction and then translate a random sample of 200k sentences from the monolingual target data. On the combined parallel and synthetic training data we train a new model with the same training hyper-parameters as the baseline.

Language Model Layer The architecture with an additional source-independent LM layer (+LML) is trained with the same hyper-parameters and data as the baseline model. The LM RNN uses a hidden size of 1024. The multi-task system (+LML + MTL) is trained on both parallel and monolingual data. In practice, all +LML +MTL models converge before seeing the entire monolingual corpus and at about the same number of updates as the baseline.

6 Results

Table 1 shows results on the held-out test sets. We observe that a separate LM layer does not significantly impact performance across all metrics. Adding monolingual data in the described multi-task setting improves translation performance by a small but consistent margin across all metrics. Interestingly, the improvements from monolingual data are additive to the gains from ensembling of

[1] Baseline systems are equivalent to an earlier version of Sockeye: https://github.com/awslabs/sockeye

3 models with different random seeds. However, the use of synthetic parallel data still outperforms our approach both in single and ensemble systems.

While separating out a language model allowed us to carry out multi-task training on mixed data types, it constrains gradients from monolingual data examples to a subset of source-independent network parameters ($\boldsymbol{\sigma}$). In contrast, synthetic data always affects all network parameters ($\boldsymbol{\theta}$) and has a positive effect despite source sequences being noisy. We speculate that training from synthetic source data may also act as a model regularizer.

7 Conclusion

We proposed a way to directly integrate target-side monolingual data into NMT through multi-task learning. Our approach avoids costly pre-training processes and jointly trains on bilingual and monolingual data from scratch. While initial results show only moderate improvements over the baseline and fall short against using synthetic parallel data, we believe there is value in pursuing this line of research further to simplify training procedures.

References

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

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation (WMT'14)*, pages 12–58. Association for Computational Linguistics Baltimore, MD, USA.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 conference on machine translation (WMT16). In *Proceedings of the First Conference on Machine Translation (WMT'16)*, pages 131–198.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*, pages 1724–1734, Doha, Qatar.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AIStats*, volume 9, pages 249–256.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *CoRR*, abs/1503.03535.

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

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*, Seattle.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Alon Lavie and Michael J. Denkowski. 2009. The METEOR metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3):105–115.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task sequence to sequence learning. *CoRR*, abs/1511.06114.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015b. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL'02)*, Philadelphia, Pennsylvania.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *CoRR*, abs/1312.6120.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the*

54th Annual Meeting of the Association for Computational Linguistics, ACL'16, Berlin, Germany.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas,*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

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

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *Proceedings of EMNLP*.

Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling

Diego Marcheggiani[1] **Ivan Titov**[1,2]
[1]ILLC, University of Amsterdam
[2]ILCC, School of Informatics, University of Edinburgh
marcheggiani@uva.nl ititov@inf.ed.ac.uk

Abstract

Semantic role labeling (SRL) is the task of identifying the predicate-argument structure of a sentence. It is typically regarded as an important step in the standard NLP pipeline. As the semantic representations are closely related to syntactic ones, we exploit syntactic information in our model. We propose a version of graph convolutional networks (GCNs), a recent class of neural networks operating on graphs, suited to model syntactic dependency graphs. GCNs over syntactic dependency trees are used as sentence encoders, producing latent feature representations of words in a sentence. We observe that GCN layers are complementary to LSTM ones: when we stack both GCN and LSTM layers, we obtain a substantial improvement over an already state-of-the-art LSTM SRL model, resulting in the best reported scores on the standard benchmark (CoNLL-2009) both for Chinese and English.

1 Introduction

Semantic role labeling (SRL) (Gildea and Jurafsky, 2002) can be informally described as the task of discovering *who* did *what* to *whom*. For example, consider an SRL dependency graph shown above the sentence in Figure 1. Formally, the task includes (1) detection of predicates (e.g., *makes*); (2) labeling the predicates with a sense from a sense inventory (e.g., *make*.01); (3) identifying and assigning arguments to *semantic roles* (e.g., *Sequa* is A0, i.e., an agent / 'doer' for the corresponding predicate, and *engines* is A1, i.e., a patient / 'an affected entity'). SRL is often regarded as an important step in the standard NLP pipeline, providing information to downstream tasks such as information extraction and question answering.

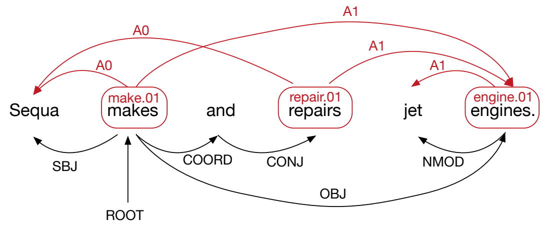


Figure 1: An example sentence annotated with semantic (top) and syntactic dependencies (bottom).

The semantic representations are closely related to syntactic ones, even though the syntax-semantics interface is far from trivial (Levin, 1993). For example, one can observe that many arcs in the syntactic dependency graph (shown in black below the sentence in Figure 1) are mirrored in the semantic dependency graph. Given these similarities and also because of availability of accurate syntactic parsers for many languages, it seems natural to exploit syntactic information when predicting semantics. Though historically most SRL approaches did rely on syntax (Thompson et al., 2003; Pradhan et al., 2005; Punyakanok et al., 2008; Johansson and Nugues, 2008), the last generation of SRL models put syntax aside in favor of neural sequence models, namely LSTMs (Zhou and Xu, 2015; Marcheggiani et al., 2017), and outperformed syntactically-driven methods on standard benchmarks. We believe that one of the reasons for this radical choice is the lack of simple and effective methods for incorporating syntactic information into sequential neural networks (namely, at the level of words). In this paper we

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1506–1515
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

propose one way how to address this limitation.

Specifically, we rely on graph convolutional networks (GCNs) (Duvenaud et al., 2015; Kipf and Welling, 2017; Kearnes et al., 2016), a recent class of multilayer neural networks operating on graphs. For every node in the graph (in our case a word in a sentence), GCN encodes relevant information about its neighborhood as a real-valued feature vector. GCNs have been studied largely in the context of undirected unlabeled graphs. We introduce a version of GCNs for modeling syntactic dependency structures and generally applicable to labeled directed graphs.

One layer GCN encodes only information about immediate neighbors and K layers are needed to encode K-order neighborhoods (i.e., information about nodes at most K hops aways). This contrasts with recurrent and recursive neural networks (Elman, 1990; Socher et al., 2013) which, at least in theory, can capture statistical dependencies across unbounded paths in a trees or in a sequence. However, as we will further discuss in Section 3.3, this is not a serious limitation when GCNs are used in combination with encoders based on recurrent networks (LSTMs). When we stack GCNs on top of LSTM layers, we obtain a substantial improvement over an already state-of-the-art LSTM SRL model, resulting in the best reported scores on the standard benchmark (CoNLL-2009), both for English and Chinese.[1]

Interestingly, again unlike recursive neural networks, GCNs do not constrain the graph to be a tree. We believe that there are many applications in NLP, where GCN-based encoders of sentences or even documents can be used to incorporate knowledge about linguistic structures (e.g., representations of syntax, semantics or discourse). For example, GCNs can take as input combined syntactic-semantic graphs (e.g., the entire graph from Figure 1) and be used within downstream tasks such as machine translation or question answering. However, we leave this for future work and here solely focus on SRL.

The contributions of this paper can be summarized as follows:

- we are the first to show that GCNs are effective for NLP;
- we propose a generalization of GCNs suited to encoding syntactic information at word level;
- we propose a GCN-based SRL model and obtain state-of-the-art results on English and Chinese portions of the CoNLL-2009 dataset;
- we show that bidirectional LSTMs and syntax-based GCNs have complementary modeling power.

2 Graph Convolutional Networks

In this section we describe GCNs of Kipf and Welling (2017). Please refer to Gilmer et al. (2017) for a comprehensive overview of GCN versions.

GCNs are neural networks operating on graphs and inducing features of nodes (i.e., real-valued vectors / embeddings) based on properties of their neighborhoods. In Kipf and Welling (2017), they were shown to be very effective for the node classification task: the classifier was estimated jointly with a GCN, so that the induced node features were informative for the node classification problem. Depending on how many layers of convolution are used, GCNs can capture information only about immediate neighbors (with one layer of convolution) or any nodes at most K hops aways (if K layers are stacked on top of each other).

More formally, consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ ($|V| = n$) and $\mathcal{E}$ are sets of nodes and edges, respectively. Kipf and Welling (2017) assume that edges contain all the self-loops, i.e., $(v, v) \in \mathcal{E}$ for any v. We can define a matrix $X \in \mathbb{R}^{m \times n}$ with each its column $x_v \in \mathbb{R}^m$ ($v \in \mathcal{V}$) encoding node features. The vectors can either encode genuine features (e.g., this vector can encode the title of a paper if citation graphs are considered) or be a one-hot vector. The node representation, encoding information about its immediate neighbors, is computed as

$$h_v = ReLU\left(\sum_{u \in \mathcal{N}(v)} (W x_u + b)\right), \qquad (1)$$

where $W \in \mathbb{R}^{m \times m}$ and $b \in \mathbb{R}^m$ are a weight matrix and a bias, respectively; $\mathcal{N}(v)$ are neighbors of v; $ReLU$ is the rectifier linear unit activation function.[2] Note that $v \in \mathcal{N}(v)$ (because of self-loops), so the input feature representation of v (i.e. x_v) affects its induced representation h_v.

[1] The code is available at `https://github.com/diegma/neural-dep-srl`.

[2] We dropped normalization factors used in Kipf and Welling (2017), as they are not used in our syntactic GCNs.

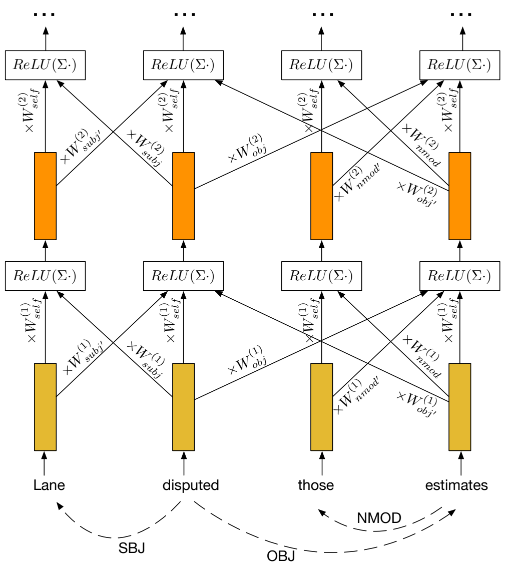


Figure 2: A simplified syntactic GCN (bias terms and gates are omitted); the syntactic graph of the sentence is shown with dashed lines at the bottom. Parameter matrices are sub-indexed with syntactic functions, and apostrophes (e.g., *subj'*) signify that information flows in the direction opposite of the dependency arcs (i.e., from dependents to heads).

As in standard convolutional networks (LeCun et al., 2001), by stacking GCN layers one can incorporate higher degree neighborhoods:

$$h_v^{(k+1)} = ReLU\left(\sum_{u\in\mathcal{N}(v)} W^{(k)} h_u^{(k)} + b^{(k)}\right)$$

where k denotes the layer number and $h_v^{(1)} = x_v$.

3 Syntactic GCNs

As syntactic dependency trees are directed and labeled (we refer to the dependency labels as *syntactic functions*), we first need to modify the computation in order to incorporate label information (Section 3.1). In the subsequent section, we incorporate gates in GCNs, so that the model can decide which edges are more relevant to the task in question. Having gates is also important as we rely on automatically predicted syntactic representations, and the gates can detect and downweight potentially erroneous edges.

3.1 Incorporating directions and labels

Now, we introduce a generalization of GCNs appropriate for syntactic dependency trees, and in general, for directed labeled graphs. First note that there is no reason to assume that information flows only along the syntactic dependency arcs (e.g., from *makes* to *Sequa*), so we allow it to flow in the opposite direction as well (i.e., from dependents to heads). We use a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the edge set contains all pairs of nodes (i.e., words) adjacent in the dependency tree. In our example, both (*Sequa*, *makes*) and (*makes*, *Sequa*) belong to the edge set. The graph is labeled, and the label $L(u, v)$ for $(u, v) \in \mathcal{E}$ contains both information about the syntactic function and indicates whether the edge is in the same or opposite direction as the syntactic dependency arc. For example, the label for (*makes*, *Sequa*) is $subj$, whereas the label for (*Sequa*, *makes*) is $subj'$, with the apostrophe indicating that the edge is in the direction opposite to the corresponding syntactic arc. Similarly, self-loops will have label $self$. Consequently, we can simply assume that the GCN parameters are label-specific, resulting in the following computation, also illustrated in Figure 2:

$$h_v^{(k+1)} = ReLU\left(\sum_{u\in\mathcal{N}(v)} W^{(k)}_{L(u,v)} h_u^{(k)} + b^{(k)}_{L(u,v)}\right).$$

This model is over-parameterized,[3] especially given that SRL datasets are moderately sized, by deep learning standards. So instead of learning the GCN parameters directly, we define them as

$$W^{(k)}_{L(u,v)} = V^{(k)}_{dir(u,v)}, \quad (2)$$

where $dir(u, v)$ indicates whether the edge (u, v) is directed (1) along, (2) in the opposite direction to the syntactic dependency arc, or (3) is a self-loop; $V^{(k)}_{dir(u,v)} \in \mathbb{R}^{m\times m}$. Our simplification captures the intuition that information should be propagated differently along edges depending whether this is a head-to-dependent or dependent-to-head edge (i.e., along or opposite the corresponding syntactic arc) and whether it is a self-loop. So we do not share any parameters between these three very different edge types. Syntactic functions are important, but perhaps less crucial, so they are encoded only in the feature vectors $b_{L(u,v)}$.

3.2 Edge-wise gating

Uniformly accepting information from all neighboring nodes may not be appropriate for the SRL

[3] Chinese and English CoNLL-2009 datasets used 41 and 48 different syntactic functions, which would result in having 83 and 97 different matrices in every layer, respectively.

setting. For example, we see in Figure 1 that many semantic arcs just mirror their syntactic counterparts, so they may need to be up-weighted. Moreover, we rely on automatically predicted syntactic structures, and, even for English, syntactic parsers are far from being perfect, especially when used out-of-domain. It is risky for a downstream application to rely on a potentially wrong syntactic edge, so the corresponding message in the neural network may need to be down-weighted.

In order to address the above issues, inspired by recent literature (van den Oord et al., 2016; Dauphin et al., 2016), we calculate for each edge node pair a scalar gate of the form

$$g_{u,v}^{(k)} = \sigma\left(h_u^{(k)} \cdot \hat{v}_{dir(u,v)}^{(k)} + \hat{b}_{L(u,v)}^{(k)}\right), \quad (3)$$

where σ is the logistic sigmoid function, $\hat{v}_{dir(u,v)}^{(k)} \in \mathbb{R}^m$ and $\hat{b}_{L(u,v)}^{(k)} \in \mathbb{R}$ are weights and a bias for the gate. With this additional gating mechanism, the final syntactic GCN computation is formulated as

$$h_v^{(k+1)} = ReLU(\sum_{u \in \mathcal{N}(v)} g_{v,u}^{(k)} (V_{dir(u,v)}^{(k)} h_u^{(k)} + b_{L(u,v)}^{(k)})). \quad (4)$$

3.3 Complementarity of GCNs and LSTMs

The inability of GCNs to capture dependencies between nodes far away from each other in the graph may seem like a serious problem, especially in the context of SRL: paths between predicates and arguments often include many dependency arcs (Roth and Lapata, 2016). However, when graph convolution is performed on top of LSTM states (i.e., LSTM states serve as input $x_v = h_v^{(1)}$ to GCN) rather than static word embeddings, GCN may not need to capture more than a couple of hops.

To elaborate on this, let us speculate what role GCNs would play when used in combinations with LSTMs, given that LSTMs have already been shown very effective for SRL (Zhou and Xu, 2015; Marcheggiani et al., 2017). Though LSTMs are capable of capturing at least some degree of syntax (Linzen et al., 2016) without explicit syntactic supervision, SRL datasets are moderately sized, so LSTM models may still struggle with harder cases. Typically, harder cases for SRL involve arguments far away from their predicates. In fact, 20% and 30% of arguments are more than 5 tokens away from their predicate, in our English and Chinese collections, respectively. However, if we imagine that we can ‘teleport’ even over a single (longest) syntactic dependency edge, the ’distance’ would shrink: only 9% and 13% arguments will now be more than 5 LSTM steps away (again for English and Chinese, respectively). GCNs provide this ‘teleportation’ capability. These observations suggest that LSTMs and GCNs may be complementary, and we will see that empirical results support this intuition.

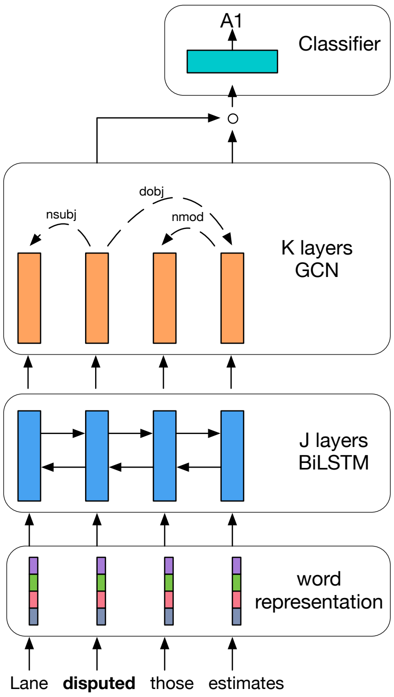


Figure 3: Predicting an argument and its label with an LSTM + GCN encoder.

4 Syntax-Aware Neural SRL Encoder

In this work, we build our semantic role labeler on top of the syntax-agnostic LSTM-based SRL model of Marcheggiani et al. (2017), which already achieves state-of-the-art results on the CoNLL-2009 English dataset. Following their approach we employ the same bidirectional (BiLSTM) encoder and enrich it with a syntactic GCN.

The CoNLL-2009 benchmark assumes that predicate positions are already marked in the test set (e.g., we would know that *makes*, *repairs* and *engines* in Figure 1 are predicates), so no predicate identification is needed. Also, as we focus here solely on identifying arguments and labeling them with semantic roles, for predicate disambiguation (i.e., marking *makes* as *make.01*) we use of an off-the-shelf disambiguation model (Roth and Lapata, 2016; Björkelund et al., 2009). As in Marcheggiani et al. (2017) and in most previous work, we process individual predicates in isolation, so for each predicate, our task reduces to a sequence labeling problem. That is, given a predicate (e.g., *disputed* in Figure 3) one needs to identify and label all its arguments (e.g., label *estimates* as A1 and label *those* as 'NULL', indicating that *those* is not an argument of *disputed*).

The semantic role labeler we propose is composed of four components (see Figure 3):

- look-ups of word embeddings;
- a BiLSTM encoder that takes as input the word representation of each word in a sentence;
- a syntax-based GCN encoder that re-encodes the BiLSTM representation based on the automatically predicted syntactic structure of the sentence;
- a role classifier that takes as input the GCN representation of the candidate argument and the representation of the predicate to predict the role associated with the candidate word.

4.1 Word representations

For each word w_i in the considered sentence, we create a sentence-specific word representation x_i. We represent each word w as the concatenation of four vectors:[4] a randomly initialized word embedding $x^{re} \in \mathbb{R}^{d_w}$, a pre-trained word embedding $x^{pe} \in \mathbb{R}^{d_w}$ estimated on an external text collection, a randomly initialized part-of-speech tag embedding $x^{pos} \in \mathbb{R}^{d_p}$ and a randomly initialized lemma embedding $x^{le} \in \mathbb{R}^{d_l}$ (active only if the word is a predicate). The randomly initialized embeddings x^{re}, x^{pos}, and x^{le} are fine-tuned during training, while the pre-trained ones are kept fixed. The final word representation is given by $x = x^{re} \circ x^{pe} \circ x^{pos} \circ x^{le}$, where $\circ$ represents the concatenation operator.

4.2 Bidirectional LSTM layer

One of the most popular and effective ways to represent sequences, such as sentences (Mikolov et al., 2010), is to use recurrent neural networks (RNN) (Elman, 1990). In particular their gated versions, Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRU) (Cho et al., 2014), have proven effective in modeling long sequences (Chiu and Nichols, 2016; Sutskever et al., 2014).

Formally, an LSTM can be defined as a function $LSTM_\theta(x_{1:i})$ that takes as input the sequence $x_{1:i}$ and returns a hidden state $h_i \in \mathbb{R}^{d_h}$. This state can be regarded as a representation of the sentence from the start to the position i, or, in other words, it encodes the word at position i along with its left context. However, the right context is also important, so Bidirectional LSTMs (Graves, 2008) use two LSTMs: one for the forward pass, and another for the backward pass, $LSTM_F$ and $LSTM_B$, respectively. By concatenating the states of both LSTMs, we create a complete context-aware representation of a word $BiLSTM(x_{1:n}, i) = LSTM_F(x_{1:i}) \circ LSTM_B(x_{n:i})$. We follow Marcheggiani et al. (2017) and stack J layers of bidirectional LSTMs, where each layer takes the lower layer as its input.

4.3 Graph convolutional layer

The representation calculated with the BiLSTM encoder is fed as input to a GCN of the form defined in Equation (4). The neighboring nodes of a node v, namely $\mathcal{N}(v)$, and their relations to v are predicted by an external syntactic parser.

4.4 Semantic role classifier

The classifier predicts semantic roles of words given the predicate while relying on word representations provided by GCN; we concatenate hidden states of the candidate argument word and the predicate word and use them as input to a classifier (Figure 3, top). The softmax classifier computes the probability of the role (including special 'NULL' role):

$$p(r|t_i, t_p, l) \propto \exp(W_{l,r}(t_i \circ t_p)), \quad (5)$$

where t_i and t_p are representations produced by the graph convolutional encoder, l is the lemma of predicate p, and the symbol $\propto$ signifies proportionality.[5] As FitzGerald et al. (2015) and Marcheggiani et al. (2017), instead of using a fixed matrix $W_{l,r}$ or simply assuming that $W_{l,r} = W_r$,

[4] We drop the index i from the notation for the sake of brevity.

[5] We abuse the notation and refer as p both to the predicate word and to its position in the sentence.

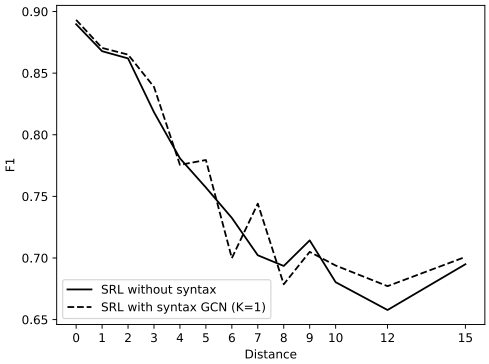


Figure 4: F_1 as function of word distance. The distance starts from zero, since nominal predicates can be arguments of themselves.

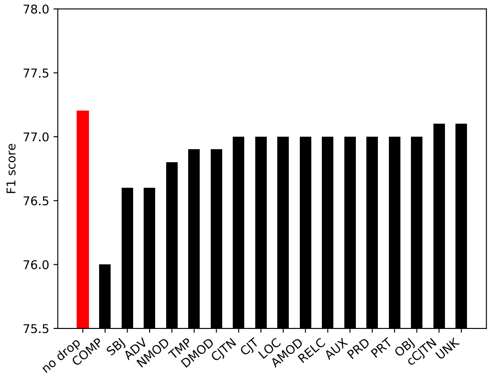


Figure 5: Performance with dependency arcs of given type dropped, on Chinese development set.

System (English)	P	R	F_1
LSTMs	84.3	81.1	82.7
LSTMs + GCNs (K=1)	85.2	81.6	83.3
LSTMs + GCNs (K=2)	84.1	81.4	82.7
LSTMs + GCNs (K=1), no gates	84.7	81.4	83.0
GCNs (no LSTMs), K=1	79.9	70.4	74.9
GCNs (no LSTMs), K=2	83.4	74.6	78.7
GCNs (no LSTMs), K=3	83.6	75.8	79.5
GCNs (no LSTMs), K=4	82.7	76.0	79.2

Table 1: SRL results without predicate disambiguation on the English development set.

we jointly embed the role r and predicate lemma l using a non-linear transformation:

$$W_{l,r} = ReLU(U(q_l \circ q_r)), \qquad (6)$$

where U is a parameter matrix, whereas $q_l \in \mathbb{R}^{d'_l}$ and $q_r \in \mathbb{R}^{d_r}$ are randomly initialized embeddings of predicate lemmas and roles. In this way each role prediction is predicate-specific, and, at the same time, we expect to learn a good representation for roles associated with infrequent predicates. As our training objective we use the categorical cross-entropy.

5 Experiments

5.1 Datasets and parameters

We tested the proposed SRL model on the English and Chinese CoNLL-2009 dataset with standard splits into training, test and development sets. The predicted POS tags for both languages were provided by the CoNLL-2009 shared-task organizers. For the predicate disambiguator we used the ones from Roth and Lapata (2016) for English and from Björkelund et al. (2009) for Chinese. We parsed English sentences with the BIST Parser (Kiperwasser and Goldberg, 2016), whereas for Chinese we used automatically predicted parses provided by the CoNLL-2009 shared-task organizers.

For English, we used external embeddings of Dyer et al. (2015), learned using the structured skip n-gram approach of Ling et al. (2015). For Chinese we used external embeddings produced with the neural language model of Bengio et al. (2003). We used *edge dropout* in GCN: when computing $h_v^{(k)}$, we ignore each node $v \in \mathcal{N}(v)$ with probability β. Adam (Kingma and Ba, 2015) was used as an optimizer. The hyperparameter tuning and all model selection were performed on the English development set; the chosen values are shown in Appendix.

5.2 Results and discussion

In order to show that GCN layers are effective, we first compare our model against its version which lacks GCN layers (i.e. essentially the model of Marcheggiani et al. (2017)). Importantly, to measure the genuine contribution of GCNs, we first tuned this syntax-agnostic model (e.g., the number of LSTM layers) to get best possible performance on the development set.[6]

We compare the syntax-agnostic model with 3 syntax-aware versions: one GCN layer over syntax ($K = 1$), one layer GCN without gates and two GCN layers ($K = 2$). As we rely on the same

[6] For example, if we would have used only one layer of LSTMs, gains from using GCNs would be even larger.

System (Chinese)	P	R	F_1
LSTMs	78.3	72.3	75.2
LSTMs + GCNs (K=1)	79.9	74.4	77.1
LSTMs + GCNs (K=2)	78.7	74.0	76.2
LSTMs + GCNs (K=1), no gates	78.2	74.8	76.5
GCNs (no LSTMs), K=1	78.7	58.5	67.1
GCNs (no LSTMs), K=2	79.7	62.7	70.1
GCNs (no LSTMs), K=3	76.8	66.8	71.4
GCNs (no LSTMs), K=4	79.1	63.5	70.4

Table 2: SRL results without predicate disambiguation on the Chinese development set.

System	P	R	F_1
Lei et al. (2015) (local)	-	-	86.6
FitzGerald et al. (2015) (local)	-	-	86.7
Roth and Lapata (2016) (local)	88.1	85.3	86.7
Marcheggiani et al. (2017) (local)	88.7	86.8	87.7
Ours (local)	**89.1**	**86.8**	**88.0**
Björkelund et al. (2010) (global)	88.6	85.2	86.9
FitzGerald et al. (2015) (global)	-	-	87.3
Foland and Martin (2015) (global)	-	-	86.0
Swayamdipta et al. (2016) (global)	-	-	85.0
Roth and Lapata (2016) (global)	90.0	85.5	87.7
FitzGerald et al. (2015) (ensemble)	-	-	87.7
Roth and Lapata (2016) (ensemble)	90.3	85.7	87.9
Ours (ensemble 3x)	**90.5**	**87.7**	**89.1**

Table 3: Results on the test set for English.

off-the-shelf disambiguator for all versions of the model, in Table 1 and 2 we report SRL-only scores (i.e., predicate disambiguation is not evaluated) on the English and Chinese development sets. For both datasets, the syntax-aware model with one GCN layers ($K = 1$) performs the best, outperforming the LSTM version by 1.9% and 0.6% for Chinese and English, respectively. The reasons why the improvements on Chinese are much larger are not entirely clear (e.g., both languages are relative fixed word order ones, and the syntactic parses for Chinese are considerably less accurate), this may be attributed to a higher proportion of long-distance dependencies between predicates and arguments in Chinese (see Section 3.3). Edge-wise gating (Section 3.2) also appears important: removing gates leads to a drop of 0.3% F_1 for English and 0.6% F_1 for Chinese.

Stacking two GCN layers does not give any benefit. When BiLSTM layers are dropped altogether, stacking two layers ($K = 2$) of GCNs greatly improves the performance, resulting in a 3.8% jump in F_1 for English and a 3.0% jump in F_1 for Chinese. Adding a 3rd layer of GCN ($K = 3$) further improves the performance.[7] This suggests that extra GCN layers are effective but largely redundant with respect to what LSTMs already capture.

System	P	R	F_1
Zhao et al. (2009) (global)	80.4	75.2	77.7
Björkelund et al. (2009) (global)	82.4	75.1	78.6
Roth and Lapata (2016) (global)	83.2	75.9	79.4
Ours (local)	**84.6**	**80.4**	**82.5**

Table 4: Results on the Chinese test set.

In Figure 4, we show the F_1 scores results on the English development set as a function of the distance, in terms of tokens, between a candidate argument and its predicate. As expected, GCNs appear to be more beneficial for long distance dependencies, as shorter ones are already accurately captured by the LSTM encoder.

We looked closer in contribution of specific dependency relations for Chinese. In order to assess this without retraining the model multiple times, we drop all dependencies of a given type at test time (one type at a time, only for types appearing over 300 times in the development set) and observe changes in performance. In Figure 5, we see that the most informative dependency is COMP (complement). Relative clauses in Chinese are very frequent and typically marked with particle 的 (de). The relative clause will syntactically depend on 的 as COMP, so COMP encodes important information about predicate-argument structure. These are often long-distance dependencies and may not be accurately captured by LSTMs. Although TMP (temporal) dependencies are not as frequent (∼2% of all dependencies), they are also important: temporal information is mirrored in semantic roles.

In order to compare to previous work, in Table 3 we report test results on the English in-domain (WSJ) evaluation data. Our model is *local*, as all the argument detection and labeling decisions are conditionally independent: their interaction is captured solely by the LSTM+GCN encoder. This makes our model fast and simple, though, as shown in previous work, *global* modeling of the structured output is beneficial.[8] We leave this extension for future work. Interestingly,

[7] Note that GCN layers are computationally cheaper than LSTM ones, even in our non-optimized implementation.

[8] As seen in Table 3, labelers of FitzGerald et al. (2015) and Roth and Lapata (2016) gained 0.6-1.0%.

System	P	R	F_1
Lei et al. (2015) (local)	-	-	75.6
FitzGerald et al. (2015) (local)	-	-	75.2
Roth and Lapata (2016) (local)	76.9	73.8	75.3
Marcheggiani et al. (2017) (local)	79.4	76.2	77.7
Ours (local)	**78.5**	**75.9**	**77.2**
Björkelund et al. (2010) (global)	77.9	73.6	75.7
FitzGerald et al. (2015) (global)	-	-	75.2
Foland and Martin (2015) (global)	-	-	75.9
Roth and Lapata (2016) (global)	78.6	73.8	76.1
FitzGerald et al. (2015) (ensemble)	-	-	75.5
Roth and Lapata (2016) (ensemble)	79.7	73.6	76.5
Ours (ensemble 3x)	**80.8**	**77.1**	**78.9**

Table 5: Results on the out-of-domain test set.

we outperform even the best global model and the best ensemble of global models, without using global modeling or ensembles. When we create an ensemble of 3 models with the product-of-expert combination rule, we improve by 1.2% over the best previous result, achieving 89.1% F_1.[9]

For Chinese (Table 4), our best model outperforms the state-of-the-art model of Roth and Lapata (2016) by even larger margin of 3.1%.

For English, in the CoNLL shared task, systems are also evaluated on the out-of-domain dataset. Statistical models are typically less accurate when they are applied to out-of-domain data. Consequently, the predicted syntax for the out-of-domain test set is of lower quality, which negatively affects the quality of GCN embeddings. However, our model works surprisingly well on out-of-domain data (Table 5), substantially outperforming all the previous syntax-aware models. This suggests that our model is fairly robust to mistakes in syntax. As expected though, our model does not outperform the syntax-agnostic model of Marcheggiani et al. (2017).

6 Related Work

Perhaps the earliest methods modeling syntax-semantics interface with RNNs are due to (Henderson et al., 2008; Titov et al., 2009; Gesmundo et al., 2009), they used shift-reduce parsers for joint SRL and syntactic parsing, and relied on RNNs to model statistical dependencies across syntactic and semantic parsing actions. A more modern (e.g., based on LSTMs) and effective reincarnation of this line of research has been proposed in Swayamdipta et al. (2016). Other recent work which considered incorporation of syntactic information in neural SRL models include: FitzGerald et al. (2015) who use standard syntactic features within an MLP calculating potentials of a CRF model; Roth and Lapata (2016) who enriched standard features for SRL with LSTM representations of syntactic paths between arguments and predicates; Lei et al. (2015) who relied on low-rank tensor factorizations for modeling syntax. Also Foland and Martin (2015) used (non-graph) convolutional networks and provided syntactic features as input. A very different line of research, but with similar goals to ours (i.e. integrating syntax with minimal feature engineering), used tree kernels (Moschitti et al., 2008).

Beyond SRL, there have been many proposals on how to incorporate syntactic information in RNN models, for example, in the context of neural machine translation (Eriguchi et al., 2017; Sennrich and Haddow, 2016). One of the most popular and attractive approaches is to use tree-structured recursive neural networks (Socher et al., 2013; Le and Zuidema, 2014; Dyer et al., 2015), including stacking them on top of a sequential BiLSTM (Miwa and Bansal, 2016). An approach of Mou et al. (2015) to sentiment analysis and question classification, introduced even before GCNs became popular in the machine learning community, is related to graph convolution. However, it is inherently single-layer and tree-specific, uses bottom-up computations, does not share parameters across syntactic functions and does not use gates. Gates have been previously used in GCNs (Li et al., 2016) but between GCN layers rather than for individual edges.

Previous approaches to integrating syntactic information in neural models are mainly designed to induce representations of sentences or syntactic constituents. In contrast, the approach we presented incorporates syntactic information at word level. This may be attractive from the engineering perspective, as it can be used, as we have shown, instead or along with RNN models.

7 Conclusions and Future Work

We demonstrated how GCNs can be used to incorporate syntactic information in neural models and specifically to construct a syntax-aware SRL

[9] To compare to previous work, we report combined scores which also include predicate disambiguation. As we use disambiguators from previous work (see Section 5.1), actual gains in argument identification and labeling are even larger.

model, resulting in state-of-the-art results for Chinese and English. There are relatively straightforward steps which can further improve the SRL results. For example, we relied on labeling arguments independently, whereas using a joint model is likely to significantly improve the performance. Also, in this paper we consider the dependency version of the SRL task, however the model can be generalized to the span-based version of the task (i.e. labeling argument spans with roles rather that syntactic heads of arguments) in a relatively straightforward fashion.

More generally, given simplicity of GCNs and their applicability to general graph structures (not necessarily trees), we believe that there are many NLP tasks where GCNs can be used to incorporate linguistic structures (e.g., syntactic and semantic representations of sentences and discourse parses or co-reference graphs for documents).

Acknowledgements

We would thank Anton Frolov, Michael Schlichtkrull, Thomas Kipf, Michael Roth, Max Welling, Yi Zhang, and Wilker Aziz for their suggestions and comments. The project was supported by the European Research Council (ERC StG BroadSem 678254), the Dutch National Science Foundation (NWO VIDI 639.022.518) and an Amazon Web Services (AWS) grant.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Proceedings of COLING: Demonstrations*.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of CoNLL*.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *TACL*, 4:357–370.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2016. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*.

David K. Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of NIPS*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL*.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. *arXiv preprint arXiv:1702.03525*.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of EMNLP*.

William Foland and James Martin. 2015. Dependency-based semantic role labeling using convolutional neural networks. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics (*SEM)*.

Andrea Gesmundo, James Henderson, Paola Merlo, and Ivan Titov. 2009. Latent variable model of synchronous syntactic-semantic parsing for multiple languages. In *Proceedings of CoNLL*.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*.

Alex Graves. 2008. *Supervised sequence labelling with recurrent neural networks*. Ph.D. thesis, München, Techn. Univ., Diss., 2008.

James Henderson, Paola Merlo, Gabriele Musillo, and Ivan Titov. 2008. A latent variable model of synchronous parsing for syntactic and semantic dependencies. In *Proceedings of CoNLL*.

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

Richard Johansson and Pierre Nugues. 2008. The effect of syntactic representation on semantic role labeling. In *Proceedings of COLING*.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. 2016. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of EMNLP*.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. 2001. Gradient-based learning applied to document recognition. In *Proceedings of Intelligent Signal Processing*.

Tao Lei, Yuan Zhang, Lluís Màrquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proceedings of NAACL*.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated graph sequence neural networks. In *Proceedings of ICLR*.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of NAACL*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *TACL*, 4:521–535.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of CoNLL*.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of ACL*.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of EMNLP*.

Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. 2016. Conditional image generation with PixelCNN decoders. In *Proceedings of NIPS*.

Sameer Pradhan, Kadri Hacioglu, Wayne H. Ward, James H. Martin, and Daniel Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *Proceedings of CoNLL*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of ACL*.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of WMT*.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*.

Swabha Swayamdipta, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Greedy, joint syntactic-semantic parsing with stack LSTMs. In *Proceedings of CoNLL*.

Cynthia A. Thompson, Roger Levy, and Christopher D. Manning. 2003. A generative model for semantic role labeling. In *Proceedings of ECML*.

Ivan Titov, James Henderson, Paola Merlo, and Gabriele Musillo. 2009. Online projectivisation for synchronous parsing of semantic and syntactic dependencies. In *Proceedings of IJCAI*.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of CoNLL*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of ACL*.

Neural Semantic Parsing with Type Constraints for Semi-Structured Tables

Jayant Krishnamurthy,[1] Pradeep Dasigi,[2] and Matt Gardner[1]
[1]Allen Institute for Artificial Intelligence
[2]Carnegie Mellon University
{jayantk, mattg}@allenai.org, pdasigi@cs.cmu.edu

Abstract

We present a new semantic parsing model for answering compositional questions on semi-structured Wikipedia tables. Our parser is an encoder-decoder neural network with two key technical innovations: (1) a grammar for the decoder that only generates well-typed logical forms; and (2) an entity embedding and linking module that identifies entity mentions while generalizing across tables. We also introduce a novel method for training our neural model with question-answer supervision. On the WIKITABLEQUESTIONS data set, our parser achieves a state-of-the-art accuracy of 43.3% for a single model and 45.9% for a 5-model ensemble, improving on the best prior score of 38.7% set by a 15-model ensemble. These results suggest that type constraints and entity linking are valuable components to incorporate in neural semantic parsers.

1 Introduction

Semantic parsing is the problem of translating human language into computer language, and therefore is at the heart of natural language understanding. A typical semantic parsing task is question answering against a database, which is accomplished by translating questions into executable logical forms (i.e., programs) that output their answers. Recent work has shown that recurrent neural networks can be used for semantic parsing by encoding the question then predicting each token of the logical form in sequence (Jia and Liang, 2016; Dong and Lapata, 2016). These approaches, while effective, have two major limitations. First, they treat the logical form as an unstructured sequence, thereby ignoring type constraints on well-formed programs. Second, they do not address entity linking, which is a critical subproblem of semantic parsing (Yih et al., 2015).

This paper introduces a novel neural semantic parsing model that addresses these limitations of prior work. Our parser uses an encoder-decoder architecture with two key innovations. First, the decoder generates from a grammar that guarantees that generated logical forms are well-typed. This grammar is automatically induced from typed logical forms, and does not require any manual engineering to produce. Second, the encoder incorporates an entity linking and embedding module that enables it to learn to identify which question spans should be linked to entities. Finally, we also introduce a new approach for training neural semantic parsers from question-answer supervision.

We evaluate our parser on WIKITABLEQUESTIONS, a challenging data set for question answering against semi-structured Wikipedia tables (Pasupat and Liang, 2015). This data set has a broad variety of entities and relations across different tables, along with complex questions that necessitate long logical forms. On this data set, our parser achieves a question answering accuracy of 43.3% and an ensemble of 5 parsers achieves 45.9%, both of which outperform the previous state-of-the-art of 38.7% set by an ensemble of 15 models (Haug et al., 2017). We further perform several ablation studies that demonstrate the importance of both type constraints and entity linking to achieving high accuracy on this task.

2 Related Work

Semantic parsers vary along a few important dimensions:

Formalism Early work on semantic parsing used lexicalized grammar formalisms such as Combinatory Categorial Grammar (Zettlemoyer

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1516–1526
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

and Collins, 2005, 2007; Kwiatkowski et al., 2011, 2013; Krishnamurthy and Mitchell, 2012; Artzi and Zettlemoyer, 2013) and others (Liang et al., 2011; Berant et al., 2013; Zhao and Huang, 2015; Wong and Mooney, 2006, 2007). These formalisms have the advantage of only generating well-typed logical forms, but the disadvantage of introducing latent syntactic variables that make learning difficult. Another approach is to treat semantic parsing as a machine translation problem, where the logical form is linearized then predicted as an unstructured sequence of tokens (Andreas et al., 2013). This approach is taken by recent neural semantic parsers (Jia and Liang, 2016; Dong and Lapata, 2016; Locascio et al., 2016; Ling et al., 2016). This approach has the advantage of predicting the logical form directly from the question without latent variables, which simplifies learning, but the disadvantage of ignoring type constraints on logical forms. Our type-constrained neural semantic parser inherits the advantages of both approaches: it only generates well-typed logical forms and has no syntactic latent variables as every logical form has a unique derivation. Recent work has explored similar ideas to ours in the context of Python code generation (Yin and Neubig, 2017; Rabinovich et al., 2017).

Entity Linking Identifying the entities mentioned in a question is a critical subproblem of semantic parsing in broad domains and proper entity linking can lead to large accuracy improvements (Yih et al., 2015). However, semantic parsers have typically ignored this problem by assuming that entity linking is done beforehand (as the neural parsers above do) or using a simple parameterization for the entity linking portion (as the lexicalized parsers do). Our parser explicitly includes an entity linking module that enables it to model the highly ambiguous and implicit entity mentions in WIKITABLEQUESTIONS.

Supervision Semantic parsers can be trained from labeled logical forms (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005) or question-answer pairs (Liang et al., 2011; Berant et al., 2013). Question-answer pairs were considered easier to obtain than labeled logical forms, though recent work has demonstrated that logical forms can be collected efficiently and are more effective (Yih et al., 2016). However, a key advantage of question-answer pairs is that they are agnostic to the domain representation and logical form language (e.g., lambda calculus or λ-DCS). This property is important for problems such as semi-structured tables where the proper domain representation is unclear.

Data Sets We use WIKITABLEQUESTIONS to evaluate our parser as this data set exhibits both a broad domain and complex questions. Early data sets, such as GEOQUERY (Zelle and Mooney, 1996) and ATIS (Dahl et al., 1994), have small domains with only a handful of different predicates. More recent data sets for question answering against Freebase have a much broader domain, but simple questions (Berant et al., 2013; Cai and Yates, 2013).

3 Model

This section describes our semantic parsing model and training procedure. For clarity, we describe the model on WIKITABLEQUESTIONS, though it is also applicable to other problems. The input to our model is a natural language question and a context in which it is to be answered, which in our task is a table. The model predicts the answer to the question by semantically parsing it to a logical form then executing it against the table.

Our model follows an encoder-decoder architecture using recurrent neural networks with Long Short Term Memory (LSTM) cells (Hochreiter and Schmidhuber, 1997). The input question and table entities are first encoded as vectors that are then decoded into a logical form (Figure 1). We make two significant additions to the encoder-decoder architecture. First, the encoder includes a special entity embedding and linking module that produces a *link embedding* for each question token that represents the table entities it links to (Section 3.2). Second, the action space of the decoder is defined by a *type-constrained grammar* which guarantees that generated logical forms satisfy type constraints (Section 3.3).

We train the parser using question-answer pairs as supervision using an approximation of marginal loglikelihood based on enumerating logical forms via dynamic programming on denotations (Pasupat and Liang, 2016) (Section 3.4). This approximation makes it possible to train neural models with question-answer supervision, which is otherwise difficult for efficiency and gradient variance reasons.

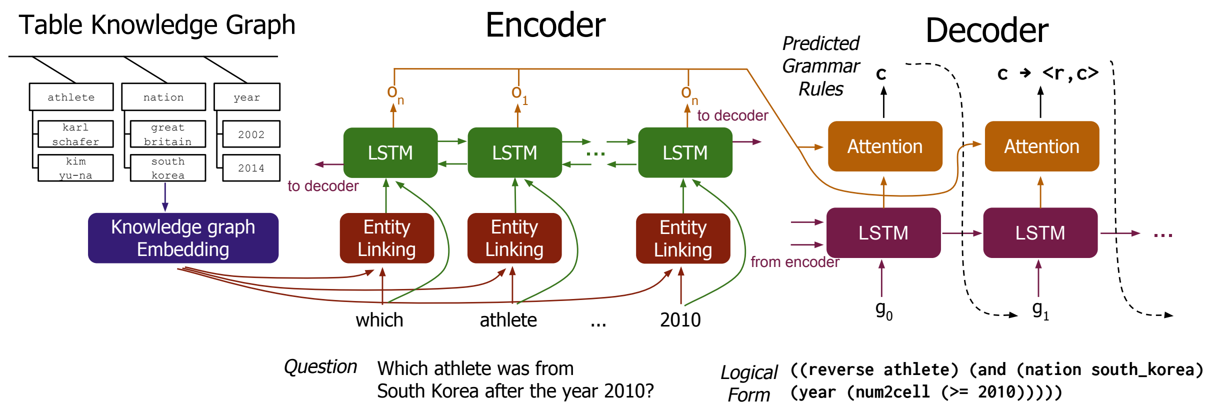


Figure 1: Overview of our semantic parsing model. The encoder performs entity embedding and linking before encoding the question with a bidirectional LSTM. The decoder predicts a sequence of grammar rules that generate a well-typed logical form.

3.1 Preliminaries

We follow (Pasupat and Liang, 2015) in using the same table structure representation and λ-DCS language for expressing logical forms. In this representation, tables are expressed as knowledge graphs over 6 types of entities: cells, cell parts, rows, columns, numbers and dates. Each entity also has a name, which is typically a string value in the table. Our parser uses both the entity names and the knowledge graph structure to construct embeddings for each entity.

The logical form language consists of a collection of named sets and entities, along with operators on them. The named sets are used to select table cells, e.g., `united_states` is the set of cells that contain the text “united states”. The operators include functions from sets to sets, e.g., the `next` operator maps a row to the next row. Columns are treated as functions from cells to their rows, e.g., `(country united_states)` generates the rows whose `country` column contains “united states”. Other operators include reversing relations (e.g., in order to map rows to cells in a certain column), relations that interpret cells as numbers and dates, and set and arithmetic operations. The language also includes aggregation and quantification operations such as `count` and `argmax`, along with λ abstractions that can be used to join binary relations.

Our parser also assigns a type to every λ-DCS expression, which is used to enforce type constraints on generated logical forms. The base types are cells `c`, parts `p`, rows `r`, numbers `i`, and dates `d`. Columns such as `country` have the functional type $\langle \mathtt{c}, \mathtt{r}\rangle$, representing functions from cells `c` to rows `r`. Other operations have more complex functional types, e.g., `reverse` has type $\langle\langle \mathtt{c}, \mathtt{r}\rangle, \langle \mathtt{r}, \mathtt{c}\rangle\rangle$, which enables us to write `(reverse country)`.[1] The parser assigns every λ-DCS constant a type, then applies standard programming language type inference algorithms (Pierce, 2002) to automatically assign types to larger expressions.

3.2 Encoder

The encoder is a bidirectional LSTM augmented with an entity embedding and linking module.

Notation. Throughout this section, we denote entities as e, and their corresponding types as $\tau(e)$. The set of all entities is denoted as E, and the entities with type τ as E_τ. E includes the cells, cell parts, and columns from the table in addition to numeric entities detected in the question by NER. The question is denoted as a sequence of tokens $[q_1, ..., q_n]$. We use v_w to denote a learned vector representation (embedding) of word w, e.g., v_{q_i} denotes the vector representation of the ith question token.

Entity Embedding. The encoder first constructs an embedding for each entity in the knowledge graph given its type and position in the graph. Let $W(e)$ denote the set of words in the name of en-

[1] Technically, `reverse` has the parametric polymorphic type $\langle\langle \alpha, \beta\rangle, \langle \beta, \alpha\rangle\rangle$, where α and β are *type variables* that can be any type. This type allows `reverse` to reverse any function. However, this is a detail that can largely be ignored. We only use parametric polymorphism when typing logical forms to generate the type-constrained grammar; the grammar itself does not have type variables, but rather a fixed number of concrete instances – such as $\langle\langle \mathtt{c}, \mathtt{r}\rangle, \langle \mathtt{r}, \mathtt{c}\rangle\rangle$ – of the above polymorphic type.

tity e and $N(e)$ the neighbors of entity e in the knowledge graph. Specifically, the neighbors of a column are the cells it contains, and the neighbors of a cell are the columns it belongs to. Each entity's embedding r_e is a nonlinear projection of a type vector $v_{\tau(e)}$ and a neighbor vector $v_{N(e)}$:

$$v_{N(e)} = \frac{1}{|N(e)|} \sum_{e' \in N(e)} \frac{1}{|W(e')|} \sum_{w \in W(e')} v_w$$

$$r_e = \tanh\left(P_\tau v_{\tau(e)} + P_N v_{N(e)}\right)$$

The type vector $v_{\tau(e)}$ is a one-hot vector for $\tau(e)$, with dimension equal to the number of entity types in the grammar. The neighbor vector $v_{N(e)}$ is simply an average of the word vectors in the names of e's neighbors. P_τ and P_N are learned parameter matrices for combining these two vectors.

Entity Linking. This module generates a *link embedding* l_i for each question token representing the entities it links to. The first part of this module generates an entity linking score $s(e, i)$ for each entity e and token index i:

$$s(e, i) = \max_{w \in W(e)} v_w^\intercal v_{q_i} + \psi^\intercal \phi(e, i)$$

This score has two terms. The first represents similarity in word embedding space between the token and entity name, computed as function of the embeddings of words in e's name, $W(e)$, and the word embedding of the ith token, v_{q_i}. The max-pooling architecture allows each question token to pick its best match in the entity's name. The second term represents a linear classifier with parameters ψ on features $\phi(e, i)$. The feature function ϕ includes only a few features: indicators for exact token and lemma match, edit distance, an NER tag indicator, and a bias feature. It also includes "related column" versions of the token and lemma features that are active when e is a column and the original feature matches a cell entity in column e. We found that features were an effective way to address sparsity in the entity name tokens, many of which appear too infrequently to learn embeddings for. We produce an independent score for each entity and token index even though we expect entities to link to multi-token spans in order to avoid the quadratic computational complexity of scoring each span.

Finally, the entity embeddings and linking scores are combined to produce a link embedding for each token. The scores $s(e, i)$ are then fed into a softmax layer over all entities e of the same type, and the link embedding l_i is an average of entity vectors r_e weighted by the resulting distribution, then summed over all types. We include a null entity, $\varnothing$, in each softmax layer to permit the model to identify tokens that do not refer to an entity. The null entity's embedding is the all-zero vector and its score $s(\varnothing, \cdot) = 0$. Note that the null entity may still be assigned high probability as the other entity scores of may be negative. The link embedding ℓ_i for the ith question token is computed as:

$$p(e|i, \tau) = \frac{\exp s(e, i)}{\sum_{e' \in E_\tau \cup \{\varnothing\}} \exp s(e', i)}$$

$$l_i = \sum_{\tau} \sum_{e \in E_\tau} r_e p(e|i, \tau)$$

For WikiTableQuestions, we ran the entity embedding and linking module over every entity. However, this approach may be prohibitively expensive in applications with a very large number of entities. In these cases, our method can be applied by adding a preliminary filtering step to identify a subset of entities that may be mentioned in the question. This filter need not have high precision, and therefore could rely on simple text overlap or similarity heuristics. This reduced set of entities can then be fed into the entity embedding and linking module, which will learn to further prune this set of candidates.

Bidirectional LSTM. We concatenate the link embedding l_i and the word embedding v_{q_i} of each token in the question, and feed them into a bidirectional LSTM:

$$x_i = \begin{bmatrix} l_i \\ v_{q_i} \end{bmatrix}$$

$$(o_i^f, f_i) = \text{LSTM}(f_{i-1}, x_i)$$

$$(o_i^b, b_i) = \text{LSTM}(b_{i+1}, x_i)$$

$$o_i = \begin{bmatrix} o_i^f \\ o_i^b \end{bmatrix}$$

This process produces an encoded vector representation of each token o_i. The final LSTM hidden states f_{n+1} and b_0 are concatenated and used to initialize the decoder.

3.3 Decoder

The decoder is an LSTM with attention that selects parsing actions from a grammar over well-typed logical forms.

Type-Constrained Grammar. The parser maintains a state at each step of decoding that consists of a logical form with nonterminals standing for portions that are yet to be generated. Each nonterminal is a tuple $[\tau, \Gamma]$ of a type τ and a *scope* Γ that contains typed variable bindings, $(x : \alpha) \in \Gamma$, where x is a variable name and α is a type. The scope is used to store and generate the arguments of lambda expressions. The grammar consists of a collection of four kinds of production rules on nonterminals:

1. **Application** $[\tau, \Gamma] \rightarrow ([\langle \beta, \tau \rangle, \Gamma] \quad [\beta, \Gamma])$ rewrites a nonterminal of type τ by applying a function from β to τ to an argument of type β. We also permit applications with more than one argument.

2. **Constant** $[\tau, \Gamma] \rightarrow$`const` where constant `const` has type τ. This rule generates both table-independent operations such as `argmax` and table-specific entities such as `united_states`.

3. **Lambda** $[\langle \alpha, \tau \rangle, \Gamma] \rightarrow \lambda x. \, [\tau, \Gamma \cup \{(x : \alpha)\}]$ generates a lambda expression where the argument has type α. x represents a fresh variable name. The right hand side of this rule extends the scope Γ with a binding for x then generates an expression of type τ.

4. **Variable** $[\tau, \Gamma] \rightarrow x$ where $(x : \tau) \in \Gamma$. This rule generates a variable bound in a previously-generated lambda expression that is currently in scope.

We instantiate each of the four rules above by replacing the type variables τ, α, β with concrete types, producing, e.g., $[\mathtt{c}, \Gamma] \rightarrow ([\langle \mathtt{r}, \mathtt{c} \rangle, \Gamma] \ [\mathtt{r}, \Gamma])$ from the application rule. The set of instantiated rules is automatically derived from a corpus of logical forms, which we in turn produce by running dynamic programming on denotations (see Section 3.4). Every logical form can be derived in exactly one way using the four kinds of rules above; this derivation is combined with the (automatically-assigned) type of each of the logical form's subexpressions to instantiate the type variables in each rule. Finally, as a postprocessing step, we filter out table-dependent rules, such as those that generate table cells, to produce a table-independent grammar. The table-dependent rules are handled specially by the decoder in order to guarantee that they are only generated when answering questions against the appropriate table. The table-independent grammar generates well-typed expressions that include functions such as `next` and quantifiers such as `argmax`; however, it cannot generate cells, columns, or other table entities.

The first action of the parser is to predict a root type for the logical form, and then decoding proceeds according to the production rules above. Each time step of decoding fills the leftmost nonterminal in the logical form, and decoding terminates when no nonterminals remain. Figure 2 shows the sequence of decoder actions used to generate an example logical form.

Network Architecture. The decoder is an LSTM that outputs a distribution over grammar actions using an attention mechanism over the encoded question tokens. The decoder also uses a copy-like mechanism on the entity linking scores to generate entities. Say that, during the jth time step, the current nonterminal has type τ. The decoder generates a score for each grammar action whose left-hand side is τ using the following equations:

$$(y_j, h_j) = \mathrm{LSTM}(h_{j-1}, \begin{bmatrix} g_{j-1} \\ o_{j-1} \end{bmatrix}) \tag{1}$$

$$a_j = \mathrm{softmax}(O W^a y_j) \tag{2}$$

$$o_j = (a_j)^T O \tag{3}$$

$$s_j = W_\tau^2 \, \mathrm{relu}(W^1 \begin{bmatrix} y_j \\ o_j \end{bmatrix} + b^1) + b_\tau^2 \tag{4}$$

$$s_j(e_k) = \sum_i s(e_k, i) a_{ji} \tag{5}$$

$$p_j = \mathrm{softmax}(\begin{bmatrix} s_j \\ s_j(e_1) \\ s_j(e_2) \\ \dots \end{bmatrix}) \tag{6}$$

The input to the LSTM g_{j-1} is a grammar action embedding for the action chosen in previous time step. g_0 is a learned parameter vector, and h_0 is the concatenated final hidden states of the encoder LSTMs. The matrix O contains the encoded token vectors $o_1, ..., , o_n$ from the encoder. The first three lines above perform a softmax attention over O using a learned parameter matrix W^a. The fourth line generates scores s_j for the table-independent grammar rules applicable to type τ using a multilayer perceptron with

Question: name the largest lake

Logical Form:

```
((reverse name) (argmax allrows (reverse (λ (x)
((reverse num2cell) ((reverse area_in_km) x))))))
```

Derivation of Logical Form:

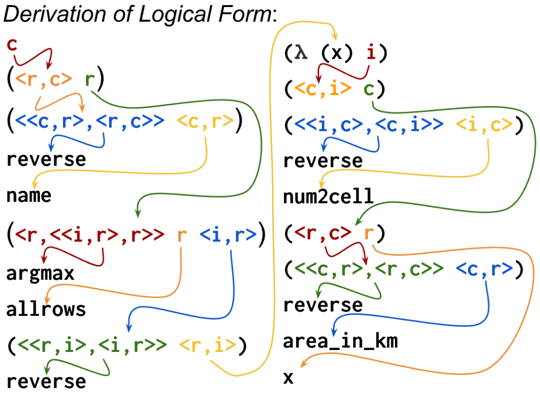


Figure 2: The derivation of a logical form using the type-constrained grammar. The nonterminals in the left column have empty scope, and those in the right column have scope $\Gamma = \{(x : \mathtt{r})\}$

weights W^1,b^1,W^2_τ,b^2_τ. The fifth line generates a score for each entity e with type τ by averaging the entity linking scores with the current attention a_j. Finally, the table-independent and -dependent scores are concatenated and softmaxed to produce a probability distribution p_j over grammar actions. If a table-independent action is chosen, g_j is a learned parameter vector for that action. Otherwise $g_j = g^\tau$, which is a learned parameter representing the selection of an entity with type τ.

3.4 DPD Training

Our parser is trained from question-answer pairs, treating logical forms as a latent variable. We use an approximate marginal loglikelihood objective function that first automatically enumerates a set of correct logical forms for each example, then trains on these logical forms. This objective simplifies the search problem during training and is well-suited to training our neural model.

The training data consists of a collection of n question-answer-table triples, $\{(q^i, a^i, T^i)\}_{i=1}^n$. We first run dynamic programming on denotations (Pasupat and Liang, 2016) on each table T^i and answer a^i to generate a set of logical forms $\ell \in \mathcal{L}^i$ that execute to the correct answer. Dynamic programming on denotations (DPD) is an automatic procedure for enumerating logical forms that execute to produce a particular value; it leverages the observation that there are fewer denotations than logical forms to enumerate this set relatively efficiently. However, many of these logical forms are *spurious*, in the sense that they do not represent the question's meaning. Therefore, the objective must marginalize over the many logical forms generated in this fashion:

$$\mathcal{O}(\theta) = \sum_{i=1}^{n} \log \sum_{\ell \in \mathcal{L}^i} P(\ell | q^i, T^i; \theta)$$

We optimize this objective function using stochastic gradient descent. If $|\mathcal{L}^i|$ is small, e.g., 5-10, the gradient of the ith example can be computed exactly by simply replicating the parser's network architecture $|\mathcal{L}^i|$ times, once per logical form. However, $|\mathcal{L}^i|$ often contains many thousands of logical forms, which makes the above computation infeasible. We address this problem by truncating $\mathcal{L}^i$ to the $m = 100$ shortest logical forms, then using a beam search with a beam of $k = 5$ to approximate the sum. Section 4.5 considers the effect of varying the number of logical forms m in this objective function.

We briefly contrast this approach with two other commonly-used approaches. The first is a similar marginal loglikelihood objective commonly used in prior semantic parsing work with loglinear models (Liang et al., 2011; Pasupat and Liang, 2015). However, this approach does not precompute correct logical forms. Therefore, computing its gradient requires running a wide beam search, generating, e.g., 300 logical forms, executing each one to identify which are correct, then backpropagating through a term for each. The wide beam is required to find correct logical forms; however, such a wide beam is prohibitively expensive with a neural model due to the cost of each backpropagation pass. Another approach is to train the network with REINFORCE (Williams, 1992), which essentially samples a logical form instead of using beam search. This approach is known to be difficult to apply when the space of outputs is large and the reward signal is sparse, and recent work has found that maximizing marginal loglikelihood is more effective in these circumstances (Guu et al., 2017). Our approach makes it tractable to maximize marginal loglikelihood with a neural model by using DPD to enumerate correct logical forms beforehand. This up-front enumeration, combined with the local normalization of the neural model, makes it possible to restrict the beam search to correct logical forms in the gradient computation, which enables training with a small beam size.

4 Evaluation

We evaluate our parser on the WIKITABLEQUESTIONS data set by comparing it to prior work and ablating several components to understand their contributions.

4.1 Experimental Setup

We used the standard train/test splits of WIKITABLEQUESTIONS. The training set consists of 14,152 examples and the test set consists of 4,344 examples. The training set comes divided into 5 cross-validation folds for development using an 80/20 split. All data sets are constructed so that the development and test tables are not present in the training set. We report question answering accuracy measured using the official evaluation script, which performs some simple normalization of numbers, dates, and strings before comparing predictions and answers. When generating answers from a model's predictions, we skip logical forms that do not execute (which may occur for some baseline models) or answer with the empty string (which is never correct). All reported accuracy numbers are an average of 5 parsers, each trained on one training fold, using the respective development set to perform early stopping.

We trained our parser with 20 epochs of stochastic gradient descent. We used 200-dimensional word embeddings for the question and entity tokens, mapping all tokens that occurred < 3 times in the training questions to UNK. (We tried using a larger vocabulary that included frequent tokens in tables, but this caused the parser to seriously overfit.) The hidden and output dimensions of the forward/backward encoder LSTMs were set to 100, such that the concatenated representations were also 200-dimensional. The decoder LSTM uses 100-dimensional action embeddings and has a 200-dimensional hidden state and output. The action selection MLP has a hidden layer dimension of 100. We used a dropout probability of 0.5 on the output of both the encoder and decoder LSTMs, as well as on the hidden layer of the action selection MLP. All parameters are initialized using Glorot initialization (Glorot and Bengio, 2010). The learning rate for SGD is initialized to 0.1 with a decay of 0.01. At test time, we decode with a beam size of 10.

Our model is implemented as a probabilistic neural program (Murray and Krishnamurthy, 2016). This Scala library combines ideas from dynamic neural network frameworks (Neubig et al., 2017) and probabilistic programming (Goodman and Stuhlmüller, 2014) to simplify the implementation of complex neural structured prediction models. This library enables a user to specify the structure of the model in terms of discrete nondeterministic choices – as in probabilistic programming – where a neural network is used to score each choice. We implement our parser by defining $P(\ell|q, T; \theta)$, from which the library automatically implements both inference and training. In particular, the beam search and the corresponding backpropagation bookkeeping to implement the objective in Section 3.4 are both automatically handled by the library. Code and supplementary material for this paper are available at:
`http://allenai.org/paper-appendix/emnlp2017-wt/`

4.2 Results

Table 1 compares the accuracy of our semantic parser to prior work on WIKITABLEQUESTIONS. We distinguish between single models and ensembles, as we expect ensembling to improve accuracy, but not all prior work has used it. Prior work on this data set includes a loglinear semantic parser (Pasupat and Liang, 2015), that same parser with a neural, paraphrase-based reranker (Haug et al., 2017), and a neural programmer that answers questions by predicting a sequence of table operations (Neelakantan et al., 2017). We find that our parser outperforms the best prior result on this data set by 4.6%, despite that prior result using a 15-model ensemble. An ensemble of 5 parsers improves accuracy by an additional 2.6% for a total improvement of 7.2%. This ensemble was constructed by averaging the logical form probabilities of parsers trained on each of the 5 cross-validation folds. Note that this ensemble is trained on the entire training set – the development data from one fold is training data for the others – so we therefore cannot report its development accuracy. We investigate the sources of this accuracy improvement in the remainder of this section via ablation experiments.

4.3 Type Constraints

Our second experiment measures the importance of type constraints on the decoder by comparing it to sequence-to-sequence (seq2seq) and sequence-to-tree (seq2tree) models. The seq2seq model generates the logical form a token at a time, e.g., [(, (,`reverse`, ...], and has been used in several

Model	Ensemble Size	Dev.	Test
Neelakantan et al. (2017)	1	34.1	34.2
Haug et al. (2017)	1	-	34.8
Pasupat and Liang (2015)	1	37.0	37.1
Neelakantan et al. (2017)	15	37.5	37.7
Haug et al. (2017)	15	-	38.7
Our Parser	1	42.7	43.3
Our Parser	5	-	**45.9**

Table 1: Development and test set accuracy of our semantic parser compared to prior work on WIKITABLEQUESTIONS.

recent neural semantic parsers (Jia and Liang, 2016; Dong and Lapata, 2016). The seq2tree model improves on the seq2seq model by including an action for generating matched parentheses, then recursively generating the subtree within (Dong and Lapata, 2016). These baseline models use the same network architecture (including entity embedding and linking) and training regime as our parser, but assign every constant the same type and have a different grammar in the decoder. These models were implemented by preprocessing logical forms and applying a different type system.

Table 2 compares the accuracy of our parser to both the seq2seq and seq2tree baselines. Both of these models perform considerably worse than our parser, demonstrating the importance of type constraints during decoding. Interestingly, we found that both baselines typically generate well-formed logical forms: only 7.4% of seq2seq and 6.6% of seq2tree's predicted logical forms failed to execute. Type constraints prevent these errors from occurring in our parser, though the relatively small number of such errors does not does not seem to fully explain the 9% accuracy improvement. We hypothesize that the additional improvement occurs because type constraints also increase the effective capacity of the model, as both the seq2seq and seq2tree models must use some of their capacity to learn the type constraints on logical forms.

4.4 Entity Embedding and Linking

Our next experiment measures the contribution of the entity embedding and linking module. We trained several ablated versions of our parser, removing both the embedding similarity and featurized classifier from the entity linking module. Table 3 shows the accuracy of the resulting models.

Model	Dev. Accuracy
seq2seq	31.3
seq2tree	31.6
Our Parser	42.7

Table 2: Development accuracy of our semantic parser compared to sequence-to-sequence and sequence-to-tree models.

Model	Dev. Accuracy
Full model	42.7
token features, no similarity	28.1
all features, no similarity	37.8
similarity only, no features	27.5

Table 3: Development accuracy of ablated parser variants trained without parts of the entity linking module.

The results demonstrate that the entity linking features are important, particularly the more complex features beyond simple token matching. In our experience, the "related column" features are especially important for this data set, as columns that appear in the logical form are often not mentioned in the text, but rather implied by a mention of a cell from the column. Embedding similarity alone is not very effective, but it does improve accuracy when combined with the featurized classifier. We found that word embeddings enabled the parser to overfit, which may be due to the relatively small size of the training set, or because we did not use pretrained embeddings. Incorporating pretrained embeddings is an area for future work.

We also examined the effect of the entity embeddings computed using each entity's knowledge graph context by replacing them with one-hot vectors for the entity's type. The accuracy of this parser dropped from 42.7% to 41.8%, demonstrating that the knowledge graph embeddings help.

4.5 DPD Training

Our final experiment examines the impact on accuracy of varying the number of logical forms m used when training with dynamic programming on denotations. Table 4 shows the development accuracy of several parsers trained with varying m. These results demonstrate that using more logical forms generally leads to higher accuracy.

# of logical forms	1	5	10	50	100
Dev. Accuracy	39.7	41.9	41.6	43.1	42.7

Table 4: Development accuracy of our semantic parser when trained with varying numbers of logical forms produced by dynamic programming on denotations.

4.6 Error Analysis

To better understand the mistakes made by our system, we analyzed a randomly selected set of 100 questions that were answered incorrectly. We identified three major classes of error:

Parser errors (41%): These are examples where a correct logical form is available, but the parser does not select it. A large number of these errors (15%) occur on questions that require selecting an answer from a given list of options, as in *Who had more silvers, Colombia or The Bahamas?* In such cases, the type of the predicted answer is often wrong. Another common subclass is entity linking errors due to missing background knowledge (13%), e.g., understanding that *largest* implicitly refers to the *Area* column.

Representation failures (25%): The knowledge graph representation makes certain assumptions about the table structure and cell values which are sometimes wrong. One common problem is that the graph lacks some cell parts necessary to answer the question (15%). For example, answering a question asking for a state may require splitting cell values in the *Location* column into city and state names. Another common problem is unusual table structures (10%), such as a table listing the number of Olympic medals won by each country that has a final row for the totals. These structures often cause quantifiers such as `argmax` to select the wrong row.

Unsupported operations (11%): These are examples where the logical form language lacks a necessary function. Examples of missing functions are finding consecutive sets of values, computing percentages and performing string operations on cell values.

5 Conclusion

We present a new semantic parsing model for answering compositional questions against semi-structured Wikipedia tables. Our semantic parser extends recent neural semantic parsers by enforcing type constraints during logical form generation, and by including an explicit entity embedding and linking module that enables it to identify entity mentions while generalizing across tables. An evaluation on WIKITABLEQUESTIONS demonstrates that our parser achieves state-of-the-art results, and furthermore that both type constraints and entity linking make significant contributions to accuracy. Analyzing the errors made by our parser suggests that improving entity linking and using the table structure are two directions for future work.

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics* 1(1):49–62.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *In Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the Workshop on Human Language Technology*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. pages 249–256.

Noah D Goodman and Andreas Stuhlmüller. 2014. The Design and Implementation of Probabilistic Programming Languages. `http://dippl.org`. Accessed: 2017-4-13.

Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Association for Computational Linguistics (ACL)*.

Till Haug, Octavian-Eugen Ganea, and Paulina Grnarova. 2017. Neural multi-step reasoning for question answering on semi-structured tables http://arxiv.org/abs/1702.06589.

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

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Jayant Krishnamurthy and Tom M. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Nicholas Locascio, Karthik Narasimhan, Eduardo De Leon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Kenton W. Murray and Jayant Krishnamurthy. 2016. Probabilistic neural programs http://arxiv.org/abs/1612.00712.

Arvind Neelakantan, Quoc V. Le, Martín Abadi, Andrew McCallum, and Dario Amodei. 2017. Learning a natural language interface with neural programmer. In *ICLR*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. DyNet: The dynamic neural network toolkit https://arxiv.org/abs/1701.03980.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Panupong Pasupat and Percy Liang. 2016. Inferring logical forms from denotations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Benjamin C. Pierce. 2002. *Types and Programming Languages*. The MIT Press, 1st edition.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *The 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8(3):229–256.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL*.

Yuk Wah Wong and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *The 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*.

Luke S Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *EMNLP-CoNLL*.

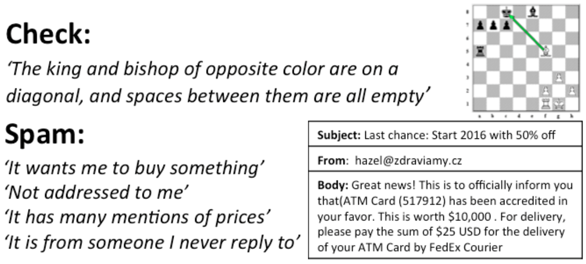
Kai Zhao and Liang Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *HLT-NAACL*.

Joint Concept Learning and Semantic Parsing from Natural Language Explanations

Shashank Srivastava **Igor Labutov** **Tom Mitchell**
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15217
ssrivastava@cmu.edu ilabutov@cs.cmu.edu tom.mitchell@cmu.edu

Abstract

Natural language constitutes a predominant medium for much of human learning and pedagogy. We consider the problem of concept learning from natural language explanations, and a small number of labeled examples of the concept. For example, in learning the concept of a phishing email, one might say 'this is a phishing email because it asks for your bank account number'. Solving this problem involves both learning to interpret open-ended natural language statements, as well as learning the concept itself. We present a joint model for (1) language interpretation (semantic parsing) and (2) concept learning (classification) that does not require labeling statements with logical forms. Instead, the model prefers discriminative interpretations of statements in context of observable features of the data as a weak signal for parsing. On a dataset of email-related concepts, this approach yields across-the-board improvements in classification performance, with a 30% relative improvement in F1 score over competitive classification methods in the low data regime.



Check:

'The king and bishop of opposite color are on a diagonal, and spaces between them are all empty'

Spam:

'It wants me to buy something'
'Not addressed to me'
'It has many mentions of prices'
'It is from someone I never reply to'

Subject: Last chance: Start 2016 with 50% off

From: hazel@zdraviamy.cz

Body: Great news! This is to officially inform you that(ATM Card (517912) has been accredited in your favor. This is worth $10,000 . For delivery, please pay the sum of $25 USD for the delivery of your ATM Card by FedEx Courier

Figure 1: Examples of concepts explained using natural language statements.

1 Introduction

The ability to automatically learn concepts[1] from examples is a core cognitive ability, with applications across diverse domains. Examples of such concepts include the concept of a 'negative review' in product reviews, the concept of 'check' over the domain of game states in chess, the concept of 'fraud' in credit history analysis, etc. Concept learning is generally approached using classification methods that can automatically leverage regularities in large amounts of labeled training data. However, there are two shortcomings of this paradigm. First, labeling large amounts of data is unnatural compared to how a person might teach another person (e.g., a human secretary) in a similar situation. For example, for identifying emails about postdoc positions, a university professor might say 'These inquiries usually seek a postdoc opportunity and include a CV', rather than label scores of examples of such emails. Second, acquiring large quantities of labeled data may be infeasible because of a long tail of concepts that are highly domain or user specific. For our example of a busy professor, it might be relevant to teach concepts such as 'postdoc seeking emails', 'course related questions from students', etc. to an email assistant in order to better manage her/his inbox. However, these concepts might be irrelevant to a general user.

On the other hand, humans can efficiently learn about new concepts and phenomena through language. In fact, verbal and written language form the basis for much of human learning and pedagogy, as reflected in text-books, lectures and student-teacher dialogues. Natural language explanations can be a potent mode of supervision, and can alleviate issues of data sparsity by directly encoding relevant knowledge about concepts. Figure 1 shows

[1] where a concept is any Boolean function on some domain of instances.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1527–1536
Copenhagen, Denmark, September 7–11, 2017.

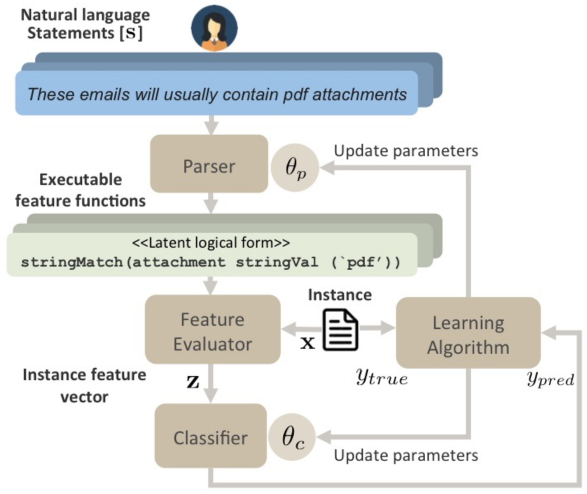


Figure 2: Schematic representation of approach

examples of concepts explained using natural language. In general, natural language can subsume several modes of supervision: instance labeling (e.g., 'This email is spam'), feature labeling (e.g., 'The word 'Viagra' indicates spam'), model expectations ('Spam emails *rarely* come from edu extensions'), etc. However, here we focus on the ability of natural language to express rich and compositional features for characterizing concepts.

In this paper, we address the task of learning concepts from natural language statements and a small number of labeled examples of the concept. Figure 2 summarizes the outline of our approach. We map statements to logical interpretations, which can be evaluated in context of new instances. In doing this, each statement s effectively acts as a binary feature function $\{z = f_s(x) \in \{0, 1\}\}$ that fires when the interpretation of a statement s is true for an instance x. The crux of our approach is that correct interpretations of natural language explanations are more likely to be useful in discriminating concepts, and this observation can be used to guide both semantic interpretation and concept learning[2].

In Section 3, we describe our probabilistic latent variable formulation that learns a semantic parser and a concept classifier from labeled examples of the concept. The latent variables correspond to evaluations of natural language statements for different instances, and training proceeds via a generalized EM procedure that iteratively (1) estimates evaluations of explanations (marginalizing over all interpretations), and (2) updates the classification and semantic parsing models. The inputs to the method consist of a small number of labeled examples and non-examples of a concept, natural language statements explaining the concept, and a domain specific lexicon. The method does not require labeling sentences with logical forms.

For our empirical evaluation, we focus on personal emails, a practical example of a domain where target concepts are often highly individualized and labeled data is scarce. The contributions of this work are:

- We introduce the problem of concept learning from natural language. We also collect a corpus of emails about common email concepts, along with statements from human users explaining these concepts.
- We provide a method for concept learning and language understanding that can be trained from a small number of labeled concept instances. Thus, we extend supervised semantic parsing by learning from a weaker form of supervision than has previously been explored.
- We demonstrate that for small labeled data, using natural language statements can achieve substantial gains in classification accuracy.

2 Related work

Concept learning from labeled examples has been a dominant focus of research in supervised learning (Caruana et al., 2008). Notable approaches such as Generalized Expectation (Mann and McCallum, 2010) and Posterior Regularization (Ganchev et al., 2010) have explored integration of manually provided 'side-information' (feature and label constraints) to guide machine learning models. Earlier work on Explanation-based learning (Mitchell et al., 1986; DeJong and Mooney, 1986) leverages structured knowledge to 'explain' why an example belongs to a concept. Recent work by Lake et al. (2015) explores visual concept learning from few examples, and presents encouraging results for one-shot learning by learning representations over Bayesian programs. However, none of these address the issue of learning from natural language.

Semantic interpretation of language has been explored in diverse domains. While semantic parsers have traditionally relied on labeled datasets of statements paired with labeled logical forms (Zettlemoyer and Collins, 2005), recent approaches have focused on training semantic parsers from

[2] e.g., a parser may associate multiple incorrect interpretations with the statement in Figure 2 (like `stringMatch(attachment stringVal ('usually'))`), which are unlikely to help in discriminating instances of the concept.

denotations of logical forms, rather than logical forms themselves (Krishnamurthy and Mitchell, 2012; Berant et al., 2013). Our work extends this paradigm by attempting to learn from still weaker signal, where denotations (evaluations) of logical forms too are not directly observed. Similar to our work, previous approaches have used different kinds of external-world signals to guide semantic interpretation (Liang et al., 2009; Branavan et al., 2009). Natural instructions have been studied in game playing frameworks (Branavan et al., 2012; Eisenstein et al., 2009). Our work is also closely related to work by Goldwasser and Roth (2014); Clarke et al. (2010), who also train semantic parsers in weakly supervised contexts, where language interpretation is integrated in real-world tasks. The general idea of learning through human interactions has previously been explored in settings such as behavioral programming (Harel et al., 2012), natural language programming (Biermann, 1983), learning by instruction (Azaria et al., 2016), etc. To the best of our knowledge, this work is the first to use semantic interpretation to guide concept learning.

3 Method

We consider concept learning problems in which the goal is to approximate an unknown classification function $f : X \to Y$ where $Y = \{0, 1\}$. The input to our learning algorithm consists of a set of labeled training examples $\mathcal{T} := \{(x_1, y_1), \ldots, (x_m, y_m)\}$, along with a set of natural language statements $\mathcal{S} := \{s_1 \ldots s_n\}$ about the concept. Our aim is to leverage statements in $\mathcal{S}$ to learn a better classifier for the concept. Our training data does not contain any other form of supervision (such as logical forms).

	s_1	s_2	s_j	s_m	Label
x_1	z_{11}	z_{12}	z_{1j}	z_{1m}	y_1
...					...
x_i	z_{i1}	z_{i2}	z_{ij}	z_{im}	y_i
...					...
x_n	z_{n1}	z_{n2}	z_{nj}	z_{nm}	y_n

Figure 3: Our data consist of instances x_i with binary labels y_i and statements $s_1 \ldots s_n$ about a concept. z_{ij} denotes whether the statement s_j applies to instance x_i, and is not observed in the data.

We assume that each statement s_j defines some Boolean property over the instances X; that is, statement s_j should be interpreted as defining a predicate $l_j : X \to \{0, 1\}$. We augment the representation of each instance, x_i, with a feature vector $\mathbf{z}_i$, that encodes the information contained in $\mathcal{S}$. The individual elements of this feature vector, $z_{ij} \in \{0, 1\}$, denote whether the statement s_j applies to instance x_i (see Figure 3). In the general case, the evaluation values $\mathbf{z}_i$'s are not directly observed. These are obtained by parsing each statement s_j into a logical expression $l_j : X \to \{0, 1\}$ which can be evaluated for an instance x_i to obtain $z_{ij} = [\![l_j]\!]_{x_i}$. Details of this evaluation are given in Section 3.4.

In this paper, we jointly learn a classifier and a semantic parser while treating z's as latent variables. For training, we maximize the conditional log likelihood of the observed data. Let us consider the log likelihood for a single data instance (ignoring the subscript i) for now. Since the evaluations $\mathbf{z}$ of natural statements for any context are latent, we marginalize over these. Using Jensen's inequality, any distribution q over the latent variables provides a lower-bound on the data log-likelihood:

$$\log p(y \mid x, \mathcal{S}) = \log \sum_{\mathbf{z}} p(y, \mathbf{z} \mid x, \mathcal{S})$$
$$\geq \sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{p(y, \mathbf{z} \mid x, \mathcal{S})}{q(\mathbf{z})}$$
$$= \sum_{\mathbf{z}} q(\mathbf{z}) \big(\underbrace{\log p_{\theta_c}(y \mid \mathbf{z}, x)}_{\text{classification}} + \underbrace{\log p_{\theta_p}(\mathbf{z} \mid x, \mathcal{S})}_{\text{parsing}} \big)$$
$$+ \mathcal{H}_q \qquad (1)$$

Here, $\mathcal{H}_q$ is the entropy term for the distribution q.

3.1 Coupling parsing and classification:

In Equation 1, we observe that the data likelihood decouples into the log probability of observing the concept labels $p_{\theta_c}(y_i \mid \mathbf{z}, x)$ conditioned on the statement evaluations and the log probability of the latent statement evaluations $p_{\theta_p}(\mathbf{z} \mid x, \mathcal{S})$. In particular, the first term can be naturally parametrized by a discriminative classifier such as a loglinear model (with associated parameters θ_c). We provide more details in Section 3.3.

On the other hand, the probability of the latent statement evaluation values $\mathbf{z}$ can be parametrized using a probabilistic semantic parsing model (with associated parameters θ_p). The second term decouples over evaluations of individual statements ($\log p_{\theta_p}(\mathbf{z} \mid x, \mathcal{S}) = \sum_j \log p_{\theta_p}(z_j \mid x, s_j)$). In turn, since we never observe the correct interpretation l for any statement, but only model its evaluation z_j, we marginalize over all interpretations whose evaluations in a context x matches z_j (similar to Liang et al. (2011)).

$$\log p_{\theta_p}(z_j \mid x, s_j) = \log \sum_{l:[\![l]\!]_x = z_j} p_{\theta_p}(l \mid s_j) \quad (2)$$

Following recent work in semantic parsing (Liang and Potts, 2015; Krishnamurthy and Mitchell, 2012), we use a log-linear model over logical forms:

$$p_{\theta_p}(l \mid s) \propto \exp({\theta_p}^T \phi(s, l)) \qquad (3)$$

where $\phi(s, l) \in \mathbb{R}^d$ is a feature vector over statements s and logical interpretations l.

3.2 Learning:

In Equation 1, $q(\mathbf{z})$ denotes a distribution over evaluation values of statements; whereas θ_c and θ_p denote the model parameters for the classifier and semantic parser. The learning algorithm consists of an iterative generalized EM procedure, which can be interpreted as a block-coordinate ascent in the estimates of statement evaluations $q(\mathbf{z})$ and the model parameters θ_c and θ_p.

E-step: In the E-step, we update our estimates of evaluation variables ($\mathbf{z}$). We make a mean-field approximation by assuming that the joint distribution over evaluations decouples as $q(\mathbf{z}) = \prod q_j(z_j)$. Then maximizing the lower bound in Equation 1 in terms of q_j leads to the following update:

$$q_j(z_j) \propto \exp\left(\mathop{\mathbb{E}}_{j' \neq j} [\log p_{\theta_c}(\mathbf{z}|x)] + \log p_{\theta_p}(z_j|x, s_j)\right) \qquad (4)$$

The first term in the update prefers values of an evaluation variable that are more discriminative on average (when values of other statements are marginalized out). The second term favours values of the evaluation variable that conforms with the most likely interpretations of the corresponding statement (s_j) by the semantic parser. Thus, in the E-step, we upweight evaluations of statements that are both discriminative, as well as supported by interpretations from the semantic parser.

M-step: In the M-step, we update the model parameters to maximize the lower bound in Equation 1. This corresponds to independently optimizing the log likelihoods for the classification model and the semantic parser, based on current estimates of $q_j(z_j)$'s of the statement evaluations. The entropy term H_q is constant from the perspective of model parameters, and is not relevant for the optimization. In particular, the semantic parser is updated to agree with evaluations of natural language statements that are discriminative. At the same time, the classification model is updated to fit evaluations that are supported by interpretations from the semantic parser.

We now describe the M-step updates for the log-linear semantic parser with parameters, θ_p. The updates for the classifier parameters, θ_c, depend on the form of the classification model, and are described in Section 3.3. For clarity, we focus on updates corresponding to a particular statement s_j from the training dataset. From Equations 1, 2 and 3, the objective for the semantic parser is given by:

$$\ell_j(\theta_p) = \sum_i \sum_{z \in \{0,1\}} q(z_{ij}) \log \frac{\sum_{l:[\![l]\!]_{x_i} = z} \exp({\theta_p}^T \phi(s_j, l))}{\sum_l \exp({\theta_p}^T \phi(s_j, l))} \qquad (5)$$

Semantic parsers are usually optimized using gradient updates. Here, the gradient is:

$$\nabla \ell_j(\theta_p) = \sum_{i,z,l} q(z_{ij}) p_{\theta_p}(l \mid s) \frac{p_{\theta_p}(z_{ij} \neq z | x_i, s)}{p_{\theta_p}(z_{ij} = z | x_i, s)} \phi(s_j, l) \qquad (6)$$

3.3 Classification models

The model and learning procedure described in Sections 3.1 and 3.2 is agnostic to the choice of the classification model (with parameters θ_c). For this work, we experimented with a logistic classifier (LR) and a Naive Bayes model (NB). We briefly describe these here:

Logistic Regression (LR): The form of the logistic function $\log p(y|\mathbf{z}) = -\log(1 + \exp(-\theta_c^T \mathbf{z}\, y))$ means that the likelihood does not decouple for individual components in $\mathbf{z}$. Hence, in the E-step, the expectation in Equation 4 cannot be computed analytically. Instead, we estimate this by drawing Bernoulli samples for individual z_j's using previous estimates of $q_j(z_j)$. In the M-step, we update classification parameters θ_c using stochastic gradient updates, while again sampling individual z_j's.

Naive Bayes (NB): The likelihood for this model is $p(y, \mathbf{z}) = \prod_j \theta_{cy}^{z_j} (1 - \theta_{cy})^{1 - z_j}$. In this case, the individual components of $\mathbf{z}$ decouple in the log likelihood, leading to simple updates in both the E and M steps. While this is not a conditional likelihood

Predicate	Description and evaluation
`stringVal`	Returns string value corresponding to a text span in the statement
`getPhraseMention`	Looks for matching tokens or phrases in a target text, and return true if an exact match is found. e.g., *The subject contains the word postdoc* → `getPhraseMention(subject,stringVal('postdoc'))`
`getPhrasesLike`	Uses an alignment based Textual Entailment (RTE) model to find the closest semantic match for a phrase in a text. Uses distributional semantics to identify semantically similar words. Returns true if a match is found. e.g., *The emails often want me to buy something* → `getPhrasesLike(email,stringVal('buy something'))`
`getSemanticCategory`	Looks for occurrences of pre-specified semantic categories in a target text (identified with Stanford CoreNLP's expanded NER tagger), and returns true if a match is found. e.g., *these emails often have contain prices and quotes* → `getSemanticCategory(body, MONEY)`
`stringMatch`	Returns true if one string value contains another. e.g., *Spam emails are rarely from a yahoo or gmail address* → `not( or(stringMatch(sender, stringVal('yahoo')), stringMatch(sender, stringVal('gmail'))))`
`stringEquals`	Returns true if two string values are equal
`or/ and/ not`	Boolean predicates with usual interpretations
`beginWith/endWith`	Return true if a target text contains a phrase, a similar phrase, or a semantic category at its beginning/end. e.g., *The emails often mention a phone number at the end* → `endWith(body, NUMBER)`
`merge`	Combines multiple elements into a list. e.g., *These emails often refer to problems like baldness and aging.* → `getPhrasesLike(email, merge(stringVal('baldness'), stringVal('aging')))`
`before`	Returns true if there is an instance of one type preceding an instance of another type in a text
`length`	Lengths of lists or text fields (in number of words)
$\geq$, `equals`	Usual arithmetical comparators
`unknown`	Return false by default. Used to deal with statements that cannot be reasonably expressed using predicates in the language. e.g., *These emails are from weird addresses.* → `unknown`

Table 1: Predicates in logical language used by our semantic parser for learning of email based concepts.

(as expected in Section 3.1), in our experiments we found that the NB objective to be empirically effective with our approach.

3.4 Semantic Parsing details

Semantic parsing refers to mapping a sentence s like '*The subject contains the word postdoc*' to a logical form l like `getPhraseMention(subject, stringVal('postdoc'))`. Logical forms can be evaluated in a context x (here, an email) to yield some meaningful output $[\![l]\!]_x$ (whether the statement is true for an email). The predicates (such as `stringVal`) and constants (such as `subject`) come from a pre-specified logical language. Since our focus in this work is concepts about emails, we specify a logical language that is expressive enough to be useful for concept learning in this domain. Table 1 lists the predicates in our logical language along with descriptions of their evaluation, and some illustrative examples showing how they can represent the meaning of natural statements[3]. Note that this logical language can express compositional meanings. e.g., '*These inquiries will usually seek a postdoc opportunity and include a CV*' can be expressed as `and (getPhrasesLike(email, stringVal('seek postdoc opportunity')), (stringMatch attachment (stringVal'CV')))`. The evaluations of some predicates uses NLP tools that go beyond exact keyword matching. In Section 5, we show that it is language understanding (semantic parsing), rather than these resources, which enables learning from natural explanations.

Semantic parsers involve grammars containing mappings from words to symbols in the logical language, as well as coarse syntactic rules. The grammar specifies the possible set of logical interpretations that can be associated with a natural language sentence. For this work, we use CCG based semantic parsing, a popular semantic parsing approach (Zettlemoyer and Collins, 2005; Artzi et al., 2015) that couples syntax with semantics. For the CCG grammar, we manually compile a domain lexicon containing a list of trigger words mapped to their syntactic categories and associated logical predicates. e.g. {'subject', NP, `subject`}. We then use the PAL lexicon induction algorithm (Krishnamurthy, 2016) to expand the lexicon by adding automatically generated entries. For training the parser, we follow the feature set from (Zettlemoyer and Collins, 2007), consisting of indicator features for lexicon entries and rule applications that fire for a given parse of a logical form. We also include

[3] We include a special predicate (`unknown`) to label statements whose meanings go beyond our logical language (last row in Table 1), essentially ignoring them. Such statements compose about 25% of our data. An agent should ideally be able to ask a user about unfamiliar concepts such as 'weird email addresses' that occur in explanations. See Section 6.

string based features denoting the number of words in a string span, and whether a string spans occur at the beginning or end of the utterance. For retrieving the best parses for a statement, we use beam search with a beam size of 500.

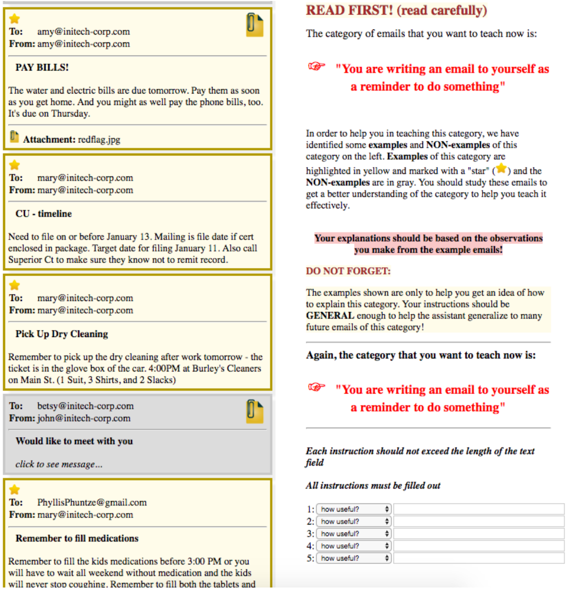
While we have chosen a particular instantiation of a semantic parsing formalism, our learning approach is independent of the semantic parsing framework in principle, and only assumes a log-linear parametrization over logical interpretations of sentences. Thus, while we present results for a particular parsing framework and lexicon, the method may conceptually extend to other parsing formalisms such as DCS (Liang et al., 2011).

4 Data

We created a dataset of 1,030 emails paired with 235 natural language statements made by human users in the process of teaching a set of seven concepts. The dataset was collected using the Amazon Mechanical Turk crowdsourcing platform. We deployed two tasks: (i) a *Generation* task requiring workers to create original emails, and (ii) a *Teaching* task requiring workers to write statements that characterize a concept. Below, we describe the data and the two tasks in more detail.

We create an email corpus, rather than use an existing corpus such as Enron, since we wanted diverse examples representative of everyday concepts that most people would be able to understand as well as teach to a computer. Much of the Enron corpus is highly specific and contextualized, making it difficult to teach for an outsider.

The *Generation* task consisted of a web-page resembling a traditional email composition form (with fields: *recipient*, *subject*, *body*, *attachment*), requiring workers to compose emails in a grounded setting. For this task, we recruited 146 workers residing in the United States. The workers were presented with each of the seven concepts in a sequence, where each concept was represented by a short prompt encouraging workers to imagine a scenario (e.g., a boss writing a request to an employee) and write a hypothetical email. See Table 2 for details of email concepts and corresponding prompts. Workers were instructed to be realistic (e.g., to include an attachment if an email is likely to have an attachment in reality), but also creative (to encourage diversity) in composing their emails.



★
To: amy@initech-corp.com
From: amy@initech-corp.com

PAY BILLS!

The water and electric bills are due tomorrow. Pay them as soon as you get home. And you might as well pay the phone bills, too. It's due on Thursday.

Attachment: redflag.jpg

★
To: mary@initech-corp.com
From: mary@initech-corp.com

CU - timeline

Need to file on or before January 13. Mailing is file date if cert enclosed in package. Target date for filing January 11. Also call Superior Ct to make sure they know not to remit record.

★
To: mary@initech-corp.com
From: mary@initech-corp.com

Pick Up Dry Cleaning

Remember to pick up the dry cleaning after work tomorrow - the ticket is in the glove box of the car. 4:00PM at Burley's Cleaners on Main St. (1 Suit, 3 Shirts, and 2 Slacks)

To: betsy@initech-corp.com
From: john@initech-corp.com

Would like to meet with you

click to see message...

★
To: PhyllisPhuntze@gmail.com
From: mary@initech-corp.com

Remember to fill medications

Remember to fill the kids medications before 3:00 PM or you will have to wait all weekend without medication and the kids will never stop coughing. Remember to fill both the tablets and

READ FIRST! (read carefully)

The category of emails that you want to teach now is:

☞ **"You are writing an email to yourself as a reminder to do something"**

In order to help you in teaching this category, we have identified some **examples** and **NON-examples** of this category on the left. **Examples** of this category are highlighted in yellow and marked with a "star" (★) and the **NON-examples** are in gray. You should study these emails to get a better understanding of the category to help you teach it effectively.

Your explanations should be based on the observations you make from the example emails!

DO NOT FORGET:

The examples shown are only to help you get an idea of how to explain this category. Your instructions should be **GENERAL** enough to help the assistant generalize to many future emails of this category!

Again, the category that you want to teach now is:

☞ **"You are writing an email to yourself as a reminder to do something"**

Each instruction should not exceed the length of the text field

All instructions must be filled out

1: how useful?
2: how useful?
3: how useful?
4: how useful?
5: how useful?

Figure 4: The *Teaching* task used to collect natural language statements characterizing a concept. Each worker is given a concept prompt, together with a set of emails. A turker can enter five statements characterizing the concept.

These emails usually closes with a name or title
Some reminders will have a date and time in the subject
The body of the email may say funny, picture, or internet
Messages to friends sometimes have jpg attachments
Emails from a public domain are not office requests

Table 3: Examples of natural language statements collected from the *Teaching* task

The *Teaching* task was then deployed to collect natural language statements that people would make to teach a particular concept to a machine. Workers were presented with five randomly selected concepts using the same prompts (Table 2) used in the *Generation* task. For each concept, a small sample of emails were shown in a style resembling a traditional email inbox (Figure 4) to illustrate the concept. Half of the emails were from the prompted concept (these emails were highlighted and "starred"), and half were sampled randomly from the other concepts. Workers were encouraged to peruse through the emails while creating up to five statements explaining the concept. A follow-up quiz assessed an understanding of the task, and contributions from workers with low scores were filtered. The final data contains between 30 and 35 statements describing each category.

5 Evaluation

In this section, we evaluate the performance of our approach from the perspectives of concept learn-

Concept	# of emails	Prompt
CONTACT	167	“You are writing an email to yourself to personally keep note of a person contact”
EMPLOYEE	149	“You are a boss writing an email to your employee requesting something to be done”
EVENT	138	“You are writing an email to a friend asking to meet up at some event”
HUMOR	134	“You are writing an email to a friend that includes something humorous from the Internet”
MEETING	142	“You are writing an email to a colleague trying to request a meeting about something”
POLICY	146	“You are writing an office email regarding announcement of some new policy”
REMINDER	154	“You are writing an email to yourself as a reminder to do something”

Table 2: Email concepts used in our experiment, together with the prompts used to describe the concept to workers. The same prompt was used in both the *Generation* and *Teaching* tasks.

	CONTACT	EMPLOYEE	EVENT	HUMOR	MEETING	POLICY	REMINDER	Average
BoW	0.510	0.354	0.381	0.484	0.455	0.588	0.415	0.455
BoW tf-Idf	0.431	0.379	0.402	0.513	0.392	0.576	0.399	0.441
Para2Vec	0.238	0.191	0.121	0.252	0.222	0.286	0.092	0.200
Bigrams	0.525	0.385	0.426	0.525	0.458	0.668	0.423	0.487
ESA	0.187	0.209	0.107	0.194	0.154	0.160	0.131	0.164
RTE	0.551	0.353	0.406	0.475	0.398	0.522	0.232	0.419
Keyword filtering	0.521	0.429	0.412	0.425	0.702	0.748	0.392	0.522
LNL-NB	0.628*	0.370	0.453*	0.590*	0.732*	0.878*	0.414	0.581*
LNL-LR	0.608*	0.351	0.568*	0.570*	0.757*	0.898*	0.437	0.598*
LNL-Gold	0.661	0.397	0.677	0.572	0.777	0.917	0.487	0.641
LNI-NB + BoW	0.644	0.409	0.520	0.709	0.723	0.878	0.543	0.632
LNI-LR + BoW	0.634	0.398	0.604	0.704	0.747	0.891	0.567	0.649
LNL-Gold+BoW	0.667	0.449	0.659	0.798	0.771	0.927	0.595	0.695

Table 4: Concept learning performance (F1 scores) using $n = 10$ labeled examples. Columns indicate different concept learning tasks defined over emails. * for the rows corresponding to LNL-NB and LNL-LR denotes statistical significance over the best performing non-LNL model

ing as well as semantic parsing. We first compare our methods against traditional supervised learning methods on the task of learning email-based concepts described in the previous section.

Our baselines include the following models:

Text-only models:

- *BoW:* A logistic regression (LR) classifier over bag-of-words representation of emails
- *BoW tf-idf:* LR classifier over bag-of-words representation, with tf-idf weighting
- *Para2Vec:* LR classifier over a distributed representation of documents, using deep neural network approach by Le and Mikolov (2014).
- *Bigram:* LR model also incorporating bigram features, known to be competitive on several text classification tasks (Wang and Manning, 2012).
- *ESA:* LR model over ESA (Explicit Semantic Analysis) representations of emails (Gabrilovich and Markovitch, 2007), which describe a text in terms of its Wikipedia topics.

Models incorporating Statements:

- *RTE:* This uses a Textual Entailment model (based on features from Sachan et al. (2015)) that computes a score for aligning of each statement to the text of each email. A logistic regression is trained over this representation of the data.
- *Keyword filtering:* Filters based on keywords are common in email systems. We add this as a baseline by manually filtering statements referring to occurrences of specific keywords. Such statements compose nearly 30% of the data. We train a logistic regression over this representation.

Table 4 shows classification performance of our approaches for Learning from Natural Language (*LNL*) against baselines described above for $n = 10$ labeled examples. The reported numbers are average F1 scores over 10 data draws. We observe that *Bigram* and bag-of-word methods are the most competitive among the baselines. On the other hand, *Para2Vec* doesn’t perform well, probably due to the relatively small scale of the available training data, while *ESA* fails due to the lack of topical associations in concepts. However, most significantly, we observe that both *LNL-NB* (Naive Bayes) and *LNL-LR* (Logistic Regression) dramatically outperform all baselines for most concepts (except EMPLOYEE), and show a 30% relative improvement in average F1 over other methods ($p < 0.05$, Paired Permutation test). Interestingly, we note that *LNL-NB* and *LNL-LR* show similar performance for most concepts. For evaluating semantic parsing of natural language statements, we manually annotated the statements in our dataset using the logical language described in Section 3.4. In Table 4, *LNL-Gold* denotes the classification performance with using these annotated gold parses. This corresponds to the hypothetical case where the classifier knows the correct semantic interpretation of each natural language sentence from an oracle. While this provides a further 10% relative improvement over our proposed models, the results suggest that our weakly supervised method is quite effective in interpreting natural language statements for concept learning, without explicit supervision. We also observe that *LNL* models perform significantly better than *Keyword filtering* ($p < 0.05$), indicating that the model leverages the expressiveness of our logical language.

Finally, the last three rows show performance when the *LNL* methods also utilize BoW representations of the data. The further gains over the base *LNL* models suggest that original feature representations and natural language explanations contain complementary information for many concepts.

A significant motivation for this work is the promise of natural language explanations in facilitating concept learning with a relatively small number of examples. Figure 5 shows the dependence of concept learning performance of *LNL(-LR)* on the number of labeled training examples (size of training set). We observe that while our approach consistently outperforms the bag-of-words model (*BoW*), *LNL* also requires fewer examples to reach near optimal performance, before it plateaus. In particular, the generalization performance for *LNL* is more robust than *BoW* for $n < 10$. The performance trajectory for *LNL(-NB)* is similar, and omitted in the figure for clarity.

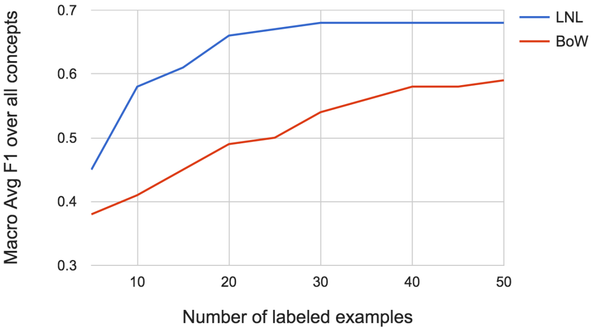


Figure 5: Figure showing Avg F1 accuracy over all concepts vs Number of labeled training examples

	Accuracy
Fully Supervised (ZC07)	0.63
LNL-LR	0.30
LNL-NB	0.28
No training	0.01

Table 5: Semantic parsing performance (exact match) for proposed weakly supervised methods vs full supervision (completely labeled logical forms)

Parsing performance: We next evaluate the parsing performance of our approach, which learns a semantic parser from only concept labels of examples. Table 5 evaluates parsing performance against the gold annotation logical forms for statements. For this task, we check for exact match of logical forms. In the table, full supervision refers to traditional training of a semantic parser using complete annotations of statements with their logical forms (Zettlemoyer and Collins, 2007). The results report average accuracy over 10-fold CV, and demonstrate that while not comparable to supervised parsing, our weakly supervised approach is relatively effective in learning semantic parsers.

Further, exact match to gold annotated logical forms is a restrictive measure. Qualitative analysis revealed that even when the predicted and gold annotation logical forms don't match, predicted logical forms are often strongly correlated in terms of evaluation to gold annotations. e.g., `getPhraseMention( email, stringVal(‘postdoc’))` vs `getPhraseMention( body, stringVal(‘postdoc’))`. In about 5% of cases, predicted and gold interpretations are different on the surface, but are semantically equivalent (e.g., `stringEquals( sender, recipient)` vs `stringEquals( recipient, sender)`).

Concept learning vs language interpretation: To delineate the relationship between parsing performance and concept learning more clearly, we plot concept classification performance for different levels of semantic parsing proficiency in Figure 6. For this, we choose the gold annotation logical form for a statement with a probability corresponding to the semantic parsing accuracy, or randomly select a candidate logical form with a uniform probability otherwise for all the statements in our data. The figure shows a (expectedly) strong association between parsing performance and concept learning, although gains from parsing taper after a certain level of proficiency. This is partially explained by the fact that natural statements in our

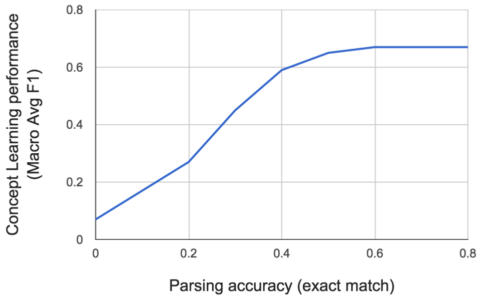


Figure 6: Figure showing concept classification performance vs parsing accuracy

data often contain overlapping information, and that the set of statements in our data set may not be sufficient to achieve perfect classification accuracy.

6 Conclusion and Future Work

We show that natural language explanations can be utilized by supervised learning methods to significantly improve generalization. This suggests a broader class of possible machine learning interfaces that use language to not only expedite learning, but make machine learning accessible to everyday users. Thus, we hope that the current work will inspire further explorations in learning from natural language explanations. In terms of scalability, learning from language would require specification of a logical language and a lexicon of trigger words for each new domain. However, this effort is one-time, and can find re-use across the long tail of concepts in a domain.

A consequence of the expressiveness of language is that in describing a concept, humans often invoke other concepts that may not correspond to existing predicates in the logical language. A natural solution could detect that a feature described in the statement is novel[4], and request the user to teach the unknown concept. The same principle can be applied recursively, resulting in a mixed-initiative dialog, much like between a student and a teacher. Future work can also incorporate other modes of supervision from language. For example, this work ignores modifiers such as ‘always’ and ‘usually’, which often carry valuable information that could be incorporated via model expectation constraints.

Acknowledgments

This research was supported by the Yahoo! InMind project. The authors thank Sujay Jauhar and anonymous reviewers for helpful comments and suggestions.

[4] currently the parser returns (unknown) for such statements

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal. Association for Computational Linguistics.

Amos Azaria, Jayant Krishnamurthy, and Tom M Mitchell. 2016. Instructable intelligent personal agent. In *AAAI*, pages 2681–2689.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, volume 2, page 6.

Alan W Biermann. 1983. Natural language programming. In *Computer Program Synthesis Methodologies*, pages 335–368. Springer.

SRK Branavan, Harr Chen, Luke S Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL: Volume 1-Volume 1*, pages 82–90. Association for Computational Linguistics.

SRK Branavan, David Silver, and Regina Barzilay. 2012. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704.

Rich Caruana, Nikos Karampatziakis, and Ainur Yessenalina. 2008. An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 96–103. ACM.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world’s response. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 18–27. Association for Computational Linguistics.

Gerald DeJong and Raymond Mooney. 1986. Explanation-based learning: An alternative view. *Machine learning*, 1(2):145–176.

Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. 2009. Reading to learn: Constructing features from semantic abstracts. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 958–967. Association for Computational Linguistics.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of* *the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1606–1611, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049.

Dan Goldwasser and Dan Roth. 2014. Learning from natural instructions. *Mach. Learn.*, 94(2):205–232.

David Harel, Assaf Marron, and Gera Weiss. 2012. Behavioral programming. *Commun. ACM*, 55(7):90–100.

Jayant Krishnamurthy. 2016. Probabilistic models for learning a semantic parser lexicon. In *Proceedings of NAACL-HLT*, pages 606–616.

Jayant Krishnamurthy and Tom M Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 754–765. Association for Computational Linguistics.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1188–1196.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 91–99. Association for Computational Linguistics.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 590–599. Association for Computational Linguistics.

Percy Liang and Christopher Potts. 2015. Bringing machine learning and compositional semantics together. *Annu. Rev. Linguist.*, 1(1):355–376.

Gideon S Mann and Andrew McCallum. 2010. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of machine learning research*, 11(Feb):955–984.

Tom M Mitchell, Richard M Keller, and Smadar T Kedar-Cabelli. 1986. Explanation-based generalization: A unifying view. *Machine learning*, 1(1):47–80.

Mrinmaya Sachan, Kumar Dubey, Eric P Xing, and Matthew Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *ACL (1)*, pages 239–249.

Sida Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 90–94, Stroudsburg, PA, USA. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 658–666.

Luke S Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*, pages 678–687.

Grasping the Finer Point: A Supervised Similarity Network for Metaphor Detection

Marek Rei[♣♠] **Luana Bulat**[♣] **Douwe Kiela**[◇] **Ekaterina Shutova**[♣]
[♣]Computer Laboratory, University of Cambridge, United Kingdom
[♠]The ALTA Institute, University of Cambridge, United Kingdom
[◇]Facebook AI Research, New York, USA
{marek.rei,luana.bulat,ekaterina.shutova}@cl.cam.ac.uk, dkiela@fb.com

Abstract

The ubiquity of metaphor in our everyday communication makes it an important problem for natural language understanding. Yet, the majority of metaphor processing systems to date rely on hand-engineered features and there is still no consensus in the field as to which features are optimal for this task. In this paper, we present the first deep learning architecture designed to capture metaphorical composition. Our results demonstrate that it outperforms the existing approaches in the metaphor identification task.

1 Introduction

Metaphor is pervasive in our everyday communication, enriching it with sophisticated imagery and helping us to reconcile our experience in the world with our conceptual system (Lakoff and Johnson, 1980). In the most influential account of metaphor to date, Lakoff and Johnson explain the phenomenon through the presence of systematic metaphorical associations between two distinct concepts or domains. For instance, when we talk about "*curing* juvenile delinquency" or "corruption *transmitting* through the government ranks", we view the general concept of *crime* (the target concept) in terms of the properties of a *disease* (the source concept). Such metaphorical associations are broad generalisations that allow us to project knowledge and inferences across domains; and our metaphorical use of language is a reflection of this process.

Given its ubiquity, metaphorical language poses an important problem for natural language understanding (Cameron, 2003; Shutova and Teufel, 2010). A number of approaches to metaphor processing have thus been proposed, focusing predominantly on classifying linguistic expressions as literal or metaphorical. They experimented with a range of features, including lexical and syntactic information (Hovy et al., 2013; Beigman Klebanov et al., 2016) and higher-level features such as semantic roles (Gedigian et al., 2006), domain types (Dunn, 2013), concreteness (Turney et al., 2011), imageability (Strzalkowski et al., 2013) and WordNet supersenses (Tsvetkov et al., 2014). While reporting promising results, all of these approaches used hand-engineered features and relied on manually-annotated resources to extract them. In order to reduce the reliance on manual annotation, other researchers experimented with sparse distributional features (Shutova et al., 2010; Shutova and Sun, 2013) and dense neural word embeddings (Bracewell et al., 2014; Shutova et al., 2016). Their experiments have demonstrated that corpus-driven lexical representations already encode information about semantic domains needed to learn the patterns of metaphor usage from linguistic data.

We take this intuition a step further and present the first deep learning architecture designed to capture metaphorical composition. Deep learning methods have already been shown successful in many other semantic tasks (e.g. Hermann et al., 2015; Kumar et al., 2015; Zhao et al., 2015), which suggests that designing a specialised neural network architecture for metaphor detection will lead to improved performance. In this paper, we present a novel architecture which (1) models the interaction between the source and target domains in the metaphor via a gating function; (2) specialises word representations for the metaphor identification task via supervised training; (3) quantifies metaphoricity via a weighted similarity function that automatically selects the relevant dimensions of similarity. We experimented with two types of word representations

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1537–1546
Copenhagen, Denmark, September 7–11, 2017.

as inputs to the network: the standard skip-gram word embeddings (Mikolov et al., 2013a) and the cognitively-driven attribute-based vectors (Bulat et al., 2017), as well as a combination thereof.

We evaluate our method in the metaphor identification task, focusing on adjective–noun, verb–subject and verb–direct object constructions where the verbs and adjectives can be used metaphorically. Our results show that our architecture outperforms both a metaphor agnostic deep learning baseline (a basic feed forward network) and the previous corpus-based approaches to metaphor identification. We also investigate the effects of training data on this task, and demonstrate that with a sufficiently large training set our method also outperforms the best existing systems based on hand-coded lexical knowledge.

2 Related Work

The majority of approaches to metaphor processing cast the problem as classification of linguistic expressions as metaphorical or literal. Gedigian et al. (2006) classified verbs related to MOTION and CURE within the domain of financial discourse. They used the maximum entropy classifier and the verbs' nominal arguments and their FrameNet roles (Fillmore et al., 2003) as features, reporting encouraging results. Dunn (2013) used a logistic regression classifier and high-level properties of concepts extracted from SUMO ontology, including domain types (ABSTRACT, PHYSICAL, SOCIAL, MENTAL) and event status (PROCESS, STATE, OBJECT). Tsvetkov et al. (2014) used random forest classifier and coarse semantic features, such as concreteness, animateness, named entity types and WordNet supersenses. They have shown that the model learned with such coarse semantic features is portable across languages. The work of Hovy et al. (2013) is notable as they focused on compositional rather than categorical features. They trained an SVM with dependency-tree kernels to capture compositional information, using lexical, part-of-speech tag and WordNet supersense representations of sentence trees. Mohler et al. (2013) aimed at modelling conceptual information. They derived semantic signatures of texts as sets of highly-related and interlinked WordNet synsets. The semantic signatures served as features to train a set of classifiers (maximum entropy, decision trees, SVM, random forest) that mapped new metaphors to the semantic signatures of the known ones.

With the aim of reducing the dependence on manually-annotated lexical resources, other research focused on modelling metaphor using corpus-driven information alone. Shutova et al. (2010) pointed out that the metaphorical uses of words constitute a large portion of the dependency features extracted for abstract concepts from corpora. For example, the feature vector for *politics* would contain GAME or MECHANISM terms among the frequent features. As a result, distributional clustering of abstract nouns with such features identifies groups of diverse concepts metaphorically associated with the same source domain. Shutova et al. (2010) exploit this property of co-occurrence vectors to identify new metaphorical mappings starting from a set of examples. Shutova and Sun (2013) used hierarchical clustering to derive a network of concepts in which metaphorical associations are learned in an unsupervised way. Do Dinh and Gurevych (2016) investigated metaphors through the task of sequence labelling, detecting metaphor related words in context. Gutiérrez et al. (2016) investigated metaphorical composition in the compositional distributional semantics framework. Their method learns metaphors as linear transformations in a vector space and they demonstrated that it produces superior phrase representations for both metaphorical and literal language, as compared to the traditional "single-sense" compositional distributional model. They then used these representations in the metaphor identification task, achieving promising results.

The more recent approaches of Shutova et al. (2016) and Bulat et al. (2017) used dense skip-gram word embeddings (Mikolov et al., 2013a) instead of the sparse distributional features. Shutova et al. (2016) investigated a set of metaphor identification methods using linguistic and visual features. They learned linguistic and visual representations for both words and phrases, using skip-gram and convolutional neural networks (Kiela and Bottou, 2014) respectively. They then measured the difference between the phrase representation and those of its component words in terms of their cosine similarity, which served as a predictor of metaphoricity. They found basic cosine similarity between the component words in the phrase to be a powerful measure – the neural embeddings of the words were compared with cosine similar-

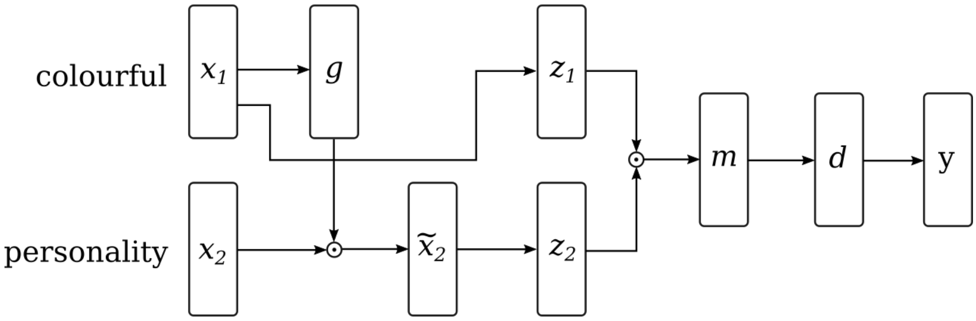


Figure 1: The network architecture for supervised metaphorical phrase classification. The $\odot$ symbol is used to indicate element-wise multiplication.

ity and a threshold was tuned on the development set to distinguish between literal and metaphorical phrases. This approach was their best performing linguistic model, outperformed only by a multi-modal system which included both linguistic and visual features.

Bulat et al. (2017) presented a metaphor identification method that uses representations constructed from human property norms (McRae et al., 2005). They first learn a mapping from the skip-gram embedding vector space to the property norm space using linear regression, which allows them to generate property norm representations for unseen words. The authors then train an SVM classifier to detect metaphors using these representations as input. Bulat et al. (2017) have shown that the cognitively-driven property norms outperform standard skip-gram representations in this task.

3 Supervised Similarity Network

Our method is inspired by the findings of Shutova et al. (2016), who showed that the cosine similarity between neural embeddings of the two words in a phrase is indicative of its metaphoricity. For example, the phrase *'colourful personality'* receives a score:

$$s = cos(x_c, x_p) \tag{1}$$

where x_c is the embedding for *colourful* and x_p is the embedding for *personality*. The combined phrase is classified as being metaphorical based on a threshold, which is optimised on a development dataset. In this paper, we propose several extensions to this general idea, creating a supervised version of the cosine similarity metric which can be optimised on training data to be more suitable for metaphor detection.

3.1 Word Representation Gating

Directly comparing the vector representations of both words treats each of the embeddings as an independent unit. In reality, however, word meanings vary and adapt based on the context. In case of metaphorical language (e.g. "*cure* crime"), the source domain properties of the verb (e.g. *cure*) are projected onto the target domain noun (e.g. *crime*), resulting in the interaction of the two domains in the interpretation of the metaphor.

In order to integrate this idea into the metaphor detection method, we can construct a gating function that modulates the representation of one word based on the other. Given embeddings x_1 and x_2, the gating values are predicted as a non-linear transformation of x_1 and applied to x_2 through element-wise multiplication:

$$g = \sigma(W_g x_1) \tag{2}$$

$$\widetilde{x}_2 = x_2 \odot g \tag{3}$$

where W_g is a weight matrix that is optimised during training, σ is the sigmoid activation function, and $\odot$ represents element-wise multiplication. In an adjective-noun phrase, this architecture allows the network to first look at the adjective, then use its meaning to change the representation of the noun. The sigmoid activation function makes it act as a filter, choosing which information from the original embedding gets through to the rest of the network. While learning a more complex gating function could be beneficial for very large training resources, the filtering approach is more suitable for the annotated metaphor datasets which are relatively small in size.

3.2 Vector Space Mapping

As the next step, we implement position-specific mappings for the word embeddings. The original method uses word embeddings that have been pre-trained using the distributional skip-gram objective (Mikolov et al., 2013a). While this tunes the vectors for predicting context words, there is no reason to believe that the same space is also optimal for the task of metaphor detection. In order to address this shortcoming, we allow the model to learn a mapping from the skip-gram vector space to a new metaphor-specific vector space:

$$z_1 = tanh(W_{z_1} x_1) \tag{4}$$

$$z_2 = tanh(W_{z_2} \widetilde{x}_2) \tag{5}$$

where W_{z_1} and W_{z_2} are weight matrices, z_1 and z_2 are the new position-specific word representations. While the original embeddings x_1 and x_2 are pre-trained on a large unannotated corpus, the transformation process is optimised using annotated metaphor examples, resulting in word representations that are more suitable for this task. Furthermore, the adjectives and nouns use separate mapping weights, which allows the model to better distinguish between the different functionalities of these words. In contrast, the original cosine similarity is not position-specific and would give the same result regardless of the word order.

3.3 Weighted Cosine

If the vectors x_1 and x_2 are normalised to unit length, the cosine similarity between them is equal to their dot product, which in turn is equal to their elementwise multiplication followed by a sum over all elements:

$$cos(x_1, x_2) \propto \sum_i x_{1,i} x_{2,i} \tag{6}$$

This calculation of cosine similarity can be formulated as a small neural network where the two unit-normalised input vectors are directly multiplied together. This is followed by a single output neuron, with all the intermediate weights set to value 1. Such a network would calculate the same sum over the element-wise multiplication, outputting the value of cosine similarity.

Since there is no reason to assume that all the embedding dimensions are equally important when detecting metaphors, we can explore other strategies for weighting the similarity calculation.

Metaphorical	Literal
absorb cost	accommodate guest
attack problem	attack village
attack cancer	blur vision
breathe life	breathe person
design excuse	deflate mattress
deflate economy	digest milk
leak news	land airplane
swallow anger	swim man

Table 1: Annotated verb-direct object and verb-subject pairs from MOH.

Rei and Briscoe (2014) used a fixed formula to calculate weights for different dimensions of cosine similarity and showed that it helped in recovering hyponym relations. We extend this even further and allow the network to use multiple different weighting strategies which are all optimised during training. This is done by first creating a vector m, which is an element-wise multiplication of the two word representations:

$$m_i = z_{1,i} z_{2,i} \tag{7}$$

where m_i is the i-th element of vector m and $z_{1,i}$ is the i-th element of vector z_1. After that, the resulting vector is used as input for a hidden neural layer:

$$d = \gamma(W_d m) \tag{8}$$

where W_d is a weight matrix and γ is an activation function. If the length of d is 1, all the weights in W_d have value 1, and γ is a linear activation, then this formula is equivalent to a regular cosine similarity. However, we use a larger length for d to capture more features, use $tanh$ as the activation function, and optimise the weights of W_d during training, giving the framework more flexibility to customise the model for the task of metaphor detection.

3.4 Prediction and Optimisation

Based on vector d we can output a prediction for the word pair, showing whether it is literal or metaphorical:

$$y = \sigma(W_y d) \tag{9}$$

where W_y is a weight matrix, σ is the logistic activation function, and y is a real-valued prediction with values between 0 and 1.

We optimise the model based on an annotated training dataset, while minimising the following hinge loss function:

$$E = \sum_{k} q_k \tag{10}$$

$$q_k = \begin{cases} (\widetilde{y} - y)^2 & \text{if } |\widetilde{y} - y| > 0.4 \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where y is the predicted value, $\widetilde{y}$ is the true label, and k iterates over all training examples. Equation 11 optimises the model to minimise the squared error between the predicted and true labels. However, this is only done for training examples where the predicted error is not already close enough to the desired result. The condition $|\widetilde{y} - y| > 0.4$ only updates training examples where the difference from the true label is greater than 0.4. The true labels $\widetilde{y}$ can only take values 0 (literal) or 1 (metaphorical), and the threshold 0.4 is chosen so that datapoints that are on the correct side of the decision boundary by more than 0.1 would be ignored, which helps reduce overfitting and allows the model to focus on the misclassified examples.

The diagram of the complete network can be seen in Figure 1.

4 Word Representations

Following Bulat et al. (2017) we experiment with two types of semantic vectors: skip-gram word embeddings and attribute-based representations.

The word embeddings are 100-dimensional and were trained using the standard log-linear skip-gram model with negative sampling of Mikolov et al. (2013b) on Wikipedia for 3 epochs, using a symmetric window of 5 and 10 negative samples per word-context pair.

We use the 2526-dimensional attribute-based vectors trained by Bulat et al. (2017), following Fagarasan et al. (2015). These representations were induced by using partial least squares regression to learn a cross-modal mapping function between the word embeddings described above and the McRae et al. (2005) property-norm semantic space.

5 Datasets

We evaluate our method using two datasets of phrases manually annotated for metaphoricity. Since these datasets include examples for different senses (both metaphorical and literal) of the same verbs or adjectives, they allow us to test the extent to which our model is able to discriminate between different word senses, as opposed to merely selecting the most frequent class for a given word.

Metaphorical	Literal
bloody stupidity	bloody nose
deep understanding	cold weather
empty promise	dry skin
green energy	empty can
healthy balance	frosty morning
hot topix	hot chocolate
muddy thinking	gold coin
ripe age	soft leather
sour mood	sour cherry
warm reception	steep hill

Table 2: Annotated adjective–noun pairs from TSV-TEST.

Mohammad et al. dataset (MOH) Mohammad et al. (2016) used WordNet to find verbs that had between three and ten senses and extracted the sentences exemplifying them in the corresponding glosses, yielding a total of 1639 verb uses in sentences. Each of these was annotated for metaphoricity by 10 annotators via the crowdsourcing platform CrowdFlower[1]. Mohammad et al. selected the verbs that were tagged by at least 70% of the annotators as metaphorical or literal to create their dataset. We extracted verb–direct object and verb–subject relations of the annotated verbs from this dataset, discarding the instances with pronominal or clausal subject or object. This resulted in a dataset of 647 verb–noun pairs (316 metaphorical and 331 literal). Some examples of annotated verb phrases from MOH are presented in Table 1.

Tsvetkov et al. dataset (TSV) Tsvetkov et al. (2014) construct a dataset of adjective–noun pairs annotated for metaphoricity. This is divided into a training set consisting of 884 literal and 884 metaphorical pairs (TSV-TRAIN) and a test set containing 100 literal and 100 metaphorical pairs (TSV-TEST). Table 2 shows a portion of annotated adjective-noun phrases from TSV-TEST. TSV-TRAIN was collected from publicly available metaphor collections on the web and manually

[1] www.crowdflower.com

curated by removing duplicates and metaphorical phrases that depend on wider context for their interpretation (e.g. *drowning students*). TSV-TEST was constructed by extracting nouns that co-occur with a list of 1000 frequent adjectives in the TenTen Web Corpus[2] using SketchEngine. The selected adjective-noun pairs were annotated for metaphoricity by 5 annotators with an inter-annotator agreement of $\kappa = 0.76$. Since TSV-TRAIN and TSV-TEST were constructed differently, we follow previous work (Tsvetkov et al., 2014; Shutova et al., 2016; Bulat et al., 2017) and report performance on TSV-TEST. We randomly separated 200 (out of the 1536) examples from the training set to use for development experiments.

6 Experiments and Results

The word representations in our model were initialised with either the 100-dimensional skip-gram embeddings or the 2,526-dimensional attribute vectors (Section 4). These were kept fixed and not updated, which reduces overfitting on the available training examples. For both word representations we use the same embeddings as Bulat et al. (2017), which makes the results directly comparable and shows that the improvements are coming from the novel architecture and are not due to a different embedding initialisation.

The network was optimised using AdaDelta (Zeiler, 2012) for controlling adaptive learning rates. The models were evaluated after each full pass over the training data and training was stopped if the F-score on the development set had not improved for 5 epochs. The transformed embeddings z_1 and z_2 were set to size 300, layer d was set to size 50. The values for these hyperparameters were chosen experimentally using the development dataset. In order to avoid drawing conclusions based on outlier results due to random initialisations, we ran each experiment 25 times with random seeds and present the averaged results in this paper. We implemented the framework using Theano (Al-Rfou et al., 2016) and are making the source code publicly available.[3]

Table 3 contains results of different system configurations on the TSV dataset. The original F-score by Tsvetkov et al. (2014) is still the highest, as they used a range of highly-engineered features that require manual annotation, such as the lexical abstractness, imageability scores and the relative number of supersenses for each word in the dataset. Our setup is more similar to the linguistic experiments by Shutova et al. (2016), where metaphor detection is performed using pretrained word embeddings. They also proposed combining the linguistic model with a system using visual word representations and achieved performance improvements. Recently, Bulat et al. (2017) compared different types of embeddings and showed that attribute-based representations can outperform regular skip-gram embeddings.

	Acc	P	R	F1
Tsvetkov et al. (2014)	-	-	-	**85**
Shutova et al. (2016)				
linguistic	-	73	80	76
multimodal	-	67	**96**	79
Bulat et al. (2017)	-	85	71	77
FFN skip-gram	77.6	86.6	65.4	74.4
FFN attribute	76.6	82.0	68.6	74.5
SSN skip-gram	82.2	**91.1**	71.6	80.1
SSN attribute	81.9	86.6	**75.7**	80.6
SSN fusion	**82.9**	90.3	73.8	**81.1**

Table 3: System performance on the Tsvetkov et al. dataset (TSV) in terms of accuracy (Acc), precision (P), recall (R) and F-score (F1).

As an additional baseline, we report the performance on metaphor detection using a basic feed-forward network (FFN). In this configuration, the word embeddings x_1 and x_2 are directly connected to the hidden layer d, skipping all the intermediate network structure. The FFN achieves 74.4% F-score on TSV-TEST, showing that even such a simple model can perform relatively well in a supervised setting. Using attribute vectors instead of skip-gram embeddings gives a slight improvement, especially on the recall metric, which is consistent with the findings by Bulat et al. (2017).

The architecture described in Section 3, which we refer to as a supervised similarity network (SSN), outperforms the baseline and achieves 80.1% F-score using skip-gram embeddings and 80.6% with attribute-based representations. We also created a fusion of these two models where the predictions from both are combined as a weighted average. In this setting, the two networks are trained in tandem and a real-valued weight, which is also optimised during training, is

[2] https://www.sketchengine.co.uk/ententen-corpus/

[3] http://www.marekrei.com/projects/ssn

	Acc	P	R	F1
Shutova et al. (2016)				
linguistic	-	67	76	71
multimodal	-	65	**87**	**75**
FFN skip-gram	71.2	70.4	71.8	70.5
FFN attribute	68.5	66.7	71.0	68.3
SSN skip-gram	**74.8**	**73.6**	**76.1**	**74.2**
SSN attribute	69.7	68.8	69.7	68.8
SSN fusion	70.8	70.1	70.9	69.9

Table 4: System performance on the Mohammad et al. dataset (MOH) in terms of accuracy (Acc), precision (P), recall (R) and F-score (F1).

used to combine them together. This configuration achieves 81.1% F-score, indicating that the the skip-gram embeddings and attribute vectors capture somewhat complementary information. Excluding the system by Tsvetkov et al. (2014) which requires hand-annotated features, the proposed similarity network outperforms all the previous systems, even improving over the multimodal system by Shutova et al. (2016) without requiring any visual information. The attribute-based SSN also improves over Bulat et al. (2017) by 5.6% absolute, using the same word representations as input.

Table 4 contains results of different system architectures on the MOH dataset. Shutova et al. (2016) reported 75% F-score on this dataset with a multimodal system, after randomly separating a subset for testing. Since this corpus contains only 647 annotated examples, we instead evaluated the systems using 10-fold cross-validation. The feedforward baseline with skip-gram embeddings returns an F-score that is close to the linguistic configuration of Shutova et al, whereas the best results are achieved by the similarity network with skip-gram embeddings. In this setting, the attribute-based representations did not improve performance – this is expected, as the attribute norms by McRae et al. (2005) are designed for nouns, whereas the MOH dataset is centered on verbs.

Input phrase	Gold	Predicted	Score
sunny country	0	0	0.152
sweet treat	0	0	0.358
lost wallet	0	0	0.439
meaningless discussion	0	0	0.150
gentle soldier	0	0	0.175
unforgiving heights	1	1	0.867
easy money	1	1	0.503
blind hope	1	1	0.813
rolling hills	1	1	0.677
educational gap	1	1	0.827
humane treatment	0	1	0.617
democratic candidate	0	1	0.510
rich programmer	0	1	0.514
fishy offer	1	0	0.290
backward area	1	0	0.161
sweet person	1	0	0.332

Table 5: Examples from the Tsvetkov development set, together with the gold label, predicted label, and the predicted score from the best model.

Table 5 contains examples from the TSV development set, together with gold annotations and predicted scores. The system confidently detects literal phrases such as *sunny country* and *meaningless discussion*, along with metaphorical phrases such as *unforgiving heights* and *blind hope*. The predicted output disagrees with the annotation on cases such as *humane treatment* and *rich programmer* – some of these examples could also be argued as being metaphorical, depending on the specific sense of the words. While the system was relatively unsure about the false positives (the scores were close to 0.5), it tended to assign more decisive scores to the false negatives.

7 The Effects of Training Data

Results in Section 6 show that performance on the TSV dataset is higher than the MOH dataset, likely due to the former having more examples available for training. Therefore, we ran an additional experiment to investigate the effect of dataset size on the performance of metaphor detection. Gutiérrez et al. (2016) annotated a dataset of adjective-noun phrases as being literal or metaphorical, and we are able to use this as an additional training resource. While it contains only 23 unique adjectives, the total number of phrases reaches 8,592. We remove any phrases that occur in the development or test data of TSV, then incrementally add the remaining examples to the TSV training data and evaluate on the TSV-TEST.

Figure 2 shows a graph of the system performance, when increasing the training data at intervals of 500. There is a very rapid increase in performance until around 2,000 training points, whereas the existing TSV-TRAIN is limited to 1,336 examples. Providing even more data to the system gives an additional increase that is more gradual. The final performance of the system us-

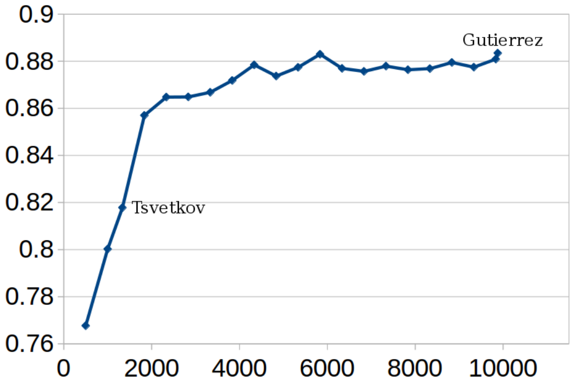


Figure 2: Performance as a function of training set size. The x-axis shows the number of training examples, the y-axis shows F-score on TSV-TEST.

Training data	Acc	P	R	F
Tsvetkov	83.0	88.3	76.3	81.8
Tsvetkov+Gutierrez	**88.7**	**91.6**	**85.4**	**88.3**

Table 6: System performance on the Tsvetkov et al. dataset (TSV), using additional training data.

ing both datasets is 88.3 F-score, which is the highest result reported on the TSV dataset and translates to 36% relative error reduction with respect to the same system trained only on the original dataset.

We report the exact values in Table 6 for the different training sets. The value on the Tsvetkov training data is different from the result in Table 3, which is due to the original attribute embeddings by Bulat et al. (2017) only containing representations for the vocabulary in the TSV dataset. In order to include the data from Gutiérrez et al. (2016), we recreated the attribute vectors for a larger vocabulary, which results in a slightly different baseline performance.

8 Qualitative analysis

The architecture in Section 3 also acts as a semantic composition model, extracting the meaning of the phrase by combining the meanings of its component words. Therefore, we performed a qualitative experiment to investigate: (1) how well do traditional compositional methods capture metaphors, without any fine-tuning; and (2) whether the supervised representations still retain their domain-specific semantic information. For this purpose, we construct three vector spaces and visualise some examples from the TSV training set, using t-SNE (Van Der Maaten and Hinton, 2008).

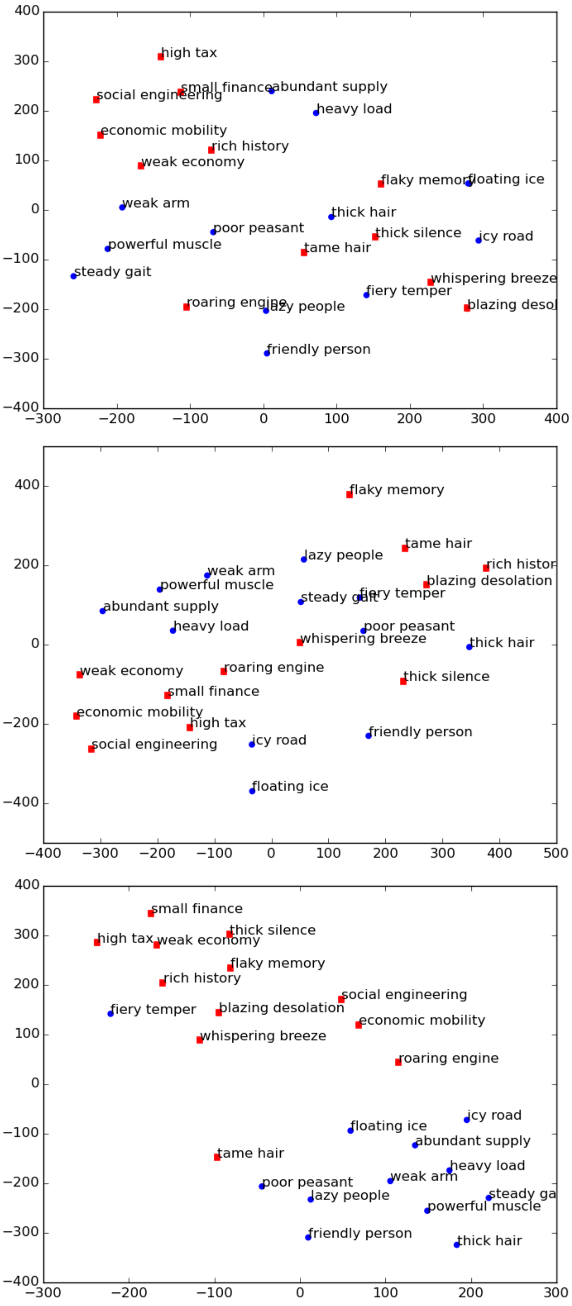


Figure 3: Comparison of metaphorical and literal phrases in different vector spaces. Blue circles indicate literal examples, red squares show metaphorical pairs. Top: additive vector space. Middle: multiplicative vector space. Bottom: vectors from layer m in the similarity network.

Figure 3 contains examples for three different composition methods: the additive method simply sums the skip-gram embeddings for both words (top); the multiplicative method multiplies the skip-gram embeddings (middle); the final system uses layer m from the SSN model to represent the phrases (bottom).

The visualisation shows that the additive and multiplicative models are both comparable when it comes to semantic clustering of the phrases, but metaphorical examples are mixed together with literal clusters. The SSN is optimised for metaphor classification and therefore it produces representations with a very clear boundary for metaphoricity. Interestingly, the graph also reveals a misannotated example in the dataset, since *'fiery temper'* should be labeled as a metaphor. At the same time, this space also retains the general semantic information, as similar phrases with the same label are still positioned close together. Future work could investigate models of multi-task training where metaphor detection is trained together with an unsupervised objective, allowing the system to take better advantage of unlabeled data while still learning to separate metaphors.

9 Conclusion

In this paper, we introduced the first deep learning architecture designed to capture metaphorical composition and evaluated it on a metaphor identification task.

Firstly, we demonstrated that the proposed framework outperforms both a metaphor-agnostic baseline (a feed-forward neural network) as well as previous corpus-driven approaches to metaphor identification. The results showed that it is beneficial to construct a specialised network architecture for metaphor detection, which includes a gating function for capturing the interaction between the source and target domains, word embeddings mapped to a metaphor-specific space, and optimisation using a hinge loss function.

Secondly, our qualitative analysis indicates that our supervised similarity network learns phrase representations with a very clear boundary for metaphoricity, in contrast to traditional compositional methods.

Finally, we show that with a sufficiently large training set our model can also outperform the state-of-the art metaphor identification systems based on hand-coded lexical knowledge.

Acknowledgments

Ekaterina Shutova's research is supported by the Leverhulme Trust Early Career Fellowship.

References

Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra, Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, and Others. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.0:19.

Beata Beigman Klebanov, Chee Wee Leong, E. Dario Gutierrez, Ekaterina Shutova, and Michael Flor. 2016. Semantic classifications for detection of verb metaphors. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 101–106, Berlin, Germany. Association for Computational Linguistics.

David Bracewell, Marc Tomlinson, Michael Mohler, and Bryan Rink. 2014. A tiered approach to the recognition of metaphor. *Computational Linguistics and Intelligent Text Processing*, 8403:403–414.

Luana Bulat, Stephen Clark, and Ekaterina Shutova. 2017. Modelling metaphor with attribute-based semantics. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*.

Lynne Cameron. 2003. *Metaphor in Educational Discourse*. Continuum, London.

Erik-Lân Do Dinh and Iryna Gurevych. 2016. Token-Level Metaphor Detection using Neural Networks. *Proceedings of the Fourth Workshop on Metaphor in NLP*.

Jonathan Dunn. 2013. Evaluating the premises and results of four metaphor identification systems. In *Proceedings of CICLing'13*, pages 471–486, Samos, Greece.

Luana Fagarasan, Eva Maria Vecchi, and Stephen Clark. 2015. From distributional semantics to feature norms: grounding semantic models in human perceptual data. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS'15)*, pages 52–57, London, UK. Association for Computational Linguistics.

Charles Fillmore, Christopher Johnson, and Miriam Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250.

Matt Gedigian, John Bryant, Srini Narayanan, and Branimir Ciric. 2006. Catching metaphors. In *In Proceedings of the 3rd Workshop on Scalable Natural Language Understanding*, pages 41–48, New York.

E. Darío Gutiérrez, Ekaterina Shutova, Tyler Marghetis, and Benjamin K. Bergen. 2016. Literal and Metaphorical Senses in Compositional

Distributional Semantic Models. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Dirk Hovy, Shashank Shrivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huying Li, Whitney Sanders, and Eduard Hovy. 2013. Identifying metaphorical word use with tree kernels. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 52–57, Atlanta, Georgia.

Douwe Kiela and Léon Bottou. 2014. Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR, abs/1506.07285*.

George Lakoff and Mark Johnson. 1980. *Metaphors We Live By*. University of Chicago Press, Chicago.

Ken McRae, George S Cree, Mark S Seidenberg, and Chris McNorgan. 2005. Semantic feature production norms for a large set of living and nonliving things. *Behavior Research Methods*, 37.

Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations (ICLR 2013)*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*, Scottsdale, AZ.

Saif M Mohammad, Ekaterina Shutova, and Peter D Turney. 2016. Metaphor as a medium for emotion: An empirical study. In *Proceedings of *SEM 2016*.

Michael Mohler, David Bracewell, Marc Tomlinson, and David Hinote. 2013. Semantic signatures for example-based linguistic metaphor detection. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 27–35, Atlanta, Georgia.

Marek Rei and Ted Briscoe. 2014. Looking for Hyponyms in Vector Space. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning (CoNLL 2014)*, pages 68–77.

Ekaterina Shutova, Douwe Kiela, and Jean Maillard. 2016. Black Holes and White Rabbits : Metaphor Identification with Visual Features. *Proceedings of NAACL-HLT 2016*.

Ekaterina Shutova and Lin Sun. 2013. Unsupervised metaphor identification using hierarchical graph factorization clustering. In *Proceedings of NAACL 2013*, Atlanta, GA, USA.

Ekaterina Shutova, Lin Sun, and Anna Korhonen. 2010. Metaphor identification using verb and noun clustering. In *Proceedings of Coling 2010*, pages 1002–1010, Beijing, China.

Ekaterina Shutova and Simone Teufel. 2010. Metaphor corpus annotated for source - target domain mappings. In *Proceedings of LREC 2010*, pages 3255–3261, Malta.

Tomek Strzalkowski, George Aaron Broadwell, Sarah Taylor, Laurie Feldman, Samira Shaikh, Ting Liu, Boris Yamrom, Kit Cho, Umit Boz, Ignacio Cases, and Kyle Elliot. 2013. Robust extraction of metaphor from novel data. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 67–76, Atlanta, Georgia.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor Detection with Cross-Lingual Model Transfer. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 248–258.

Peter D. Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 680–690, Stroudsburg, PA, USA. Association for Computational Linguistics.

Laurens Van Der Maaten and Geoffrey Hinton. 2008. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. *arXiv preprint arXiv:1504.05070*.

Identifying civilians killed by police with distantly supervised entity-event extraction

Katherine A. Keith, Abram Handler, Michael Pinkham,
Cara Magliozzi, Joshua McDuffie, and **Brendan O'Connor**
College of Information and Computer Sciences
University of Massachusetts Amherst
kkeith@cs.umass.edu, brenocon@cs.umass.edu
http://slanglab.cs.umass.edu/

Abstract

We propose a new, socially-impactful task for natural language processing: from a news corpus, extract names of persons who have been killed by police. We present a newly collected police fatality corpus, which we release publicly, and present a model to solve this problem that uses EM-based distant supervision with logistic regression and convolutional neural network classifiers. Our model outperforms two off-the-shelf event extractor systems, and it can suggest candidate victim names in some cases faster than one of the major manually-collected police fatality databases.

Appendix, software, and data are available online at: http://slanglab.cs.umass.edu/PoliceKillingsExtraction/

1 Introduction

The United States government does not keep systematic records of when police kill civilians, despite a clear need for this information to serve the public interest and support social scientific analysis. Federal records rely on incomplete cooperation from local police departments, and human rights statisticians assess that they fail to document thousands of fatalities (Lum and Ball, 2015).

News articles have emerged as a valuable alternative data source. Organizations including The Guardian, The Washington Post, Mapping Police Violence, and Fatal Encounters have started to build such databases of U.S. police killings by manually reading millions of news articles[1] and extracting victim names and event details. This approach was recently validated by a Bureau of Justice Statistics study (Banks et al., Dec. 2016) which augmented traditional police-maintained records with media reports, finding twice as many deaths compared to past government analyses. This suggests textual news data has enormous, real value, though *manual* news analysis remains extremely laborious.

Text	Person killed by police?
Alton Sterling was killed by police.	True
Officers shot and killed **Philando Castile**.	True
Officer **Andrew Hanson** was shot.	False
Police report **Megan Short** was fatally shot in apparent murder-suicide.	False

Table 1: Toy examples (with entities in bold) illustrating the problem of extracting from text names of persons who have been killed by police.

We propose to help automate this process by extracting the names of persons killed by police from event descriptions in news articles (Table 1). This can be formulated as either of two cross-document entity-event extraction tasks:

1. Populating an entity-event database: From a corpus of news articles $\mathcal{D}^{(test)}$ over timespan T, extract the names of persons killed by police during that same timespan ($\mathcal{E}^{(pred)}$).

2. Updating an entity-event database: In addition to $\mathcal{D}^{(test)}$, assume access to both a historical database of killings $\mathcal{E}^{(train)}$ and a historical news corpus $\mathcal{D}^{(train)}$ for events that occurred before T. This setting often occurs in practice, and is the focus of this paper; it allows for the use of distantly supervised learn-

[1] Fatal Encounters director D. Brian Burghart estimates he and colleagues have read 2 million news headlines and ledes to assemble its fatality records that date back to January, 2000 (pers. comm.); we find FE to be the most comprehensive publicly available database.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1547–1557
Copenhagen, Denmark, September 7–11, 2017.

ing methods.[2]

The task itself has important social value, but the NLP research community may be interested in a scientific justification as well. We propose that police fatalities are a useful test case for event extraction research. Fatalities are a well defined type of event with clear semantics for coreference, avoiding some of the more complex issues in this area (Hovy et al., 2013). The task also builds on a considerable information extraction literature on knowledge base population (e.g. Craven et al. (1998)). Finally, we posit that the field of natural language processing should, when possible, advance applications of important public interest. Previous work established the value of textual news for this problem, but computational methods could alleviate the scale of manual labor needed to use it.

To introduce this problem, we:

- Define the task of identifying persons killed by police, which is an instance of cross-document entity-event extraction (§3.1).
- Present a new dataset of web news articles collected throughout 2016 that describe possible fatal encounters with police officers (§3.2).
- Introduce, for the database update setting, a distant supervision model (§4) that incorporates feature-based logistic regression and convolutional neural network classifiers under a latent disjunction model.
- Demonstrate the approach's potential usefulness for practitioners: it outperforms two off-the-shelf event extractors (§5) and finds 39 persons not included in the Guardian's "The Counted" database of police fatalities as of January 1, 2017 (§6). This constitutes a promising first step, though performance needs to be improved for real-world usage.

2 Related Work

This task combines elements of information extraction, including: *event extraction* (a.k.a. *semantic parsing*), identifying descriptions of events and their arguments from text, and cross-document *relation extraction*, predicting semantic relations over entities. A fatality event indicates the killing of a particular person; we wish to specifically identify the names of fatality victims mentioned in text. Thus our task could be viewed as unary relation extraction: for a given person mentioned in a corpus, were they killed by a police officer?

Prior work in NLP has produced a number of event extraction systems, trained on text data hand-labeled with a pre-specified ontology, including ones that identify instances of killings (Li and Ji, 2014; Das et al., 2014). Unfortunately, they perform poorly on our task (§5), so we develop a new method.

Since we do not have access to text specifically annotated for police killing events, we instead turn to *distant supervision*—inducing labels by aligning relation-entity entries from a gold standard database to their mentions in a corpus (Craven and Kumlien, 1999; Mintz et al., 2009; Bunescu and Mooney, 2007; Riedel et al., 2010). Similar to this work, Reschke et al. (2014) apply distant supervision to multi-slot, template-based event extraction for airplane crashes; we focus on a simpler unary extraction setting with joint learning of a probabilistic model. Other related work in the cross-document setting has examined joint inference for relations, entities, and events (Yao et al., 2010; Lee et al., 2012; Yang et al., 2015).

Finally, other natural language processing efforts have sought to extract social behavioral event databases from news, such as instances of protests (Hanna, 2017), gun violence (Pavlick et al., 2016), and international relations (Schrodt and Gerner, 1994; Schrodt, 2012; Boschee et al., 2013; O'Connor et al., 2013; Gerrish, 2013). They can also be viewed as event database population tasks, with differing levels of semantic specificity in the definition of "event."

3 Task and Data

3.1 Cross-document entity-event extraction for police fatalties

From a corpus of documents $\mathcal{D}$, the task is to extract a list of candidate person names, $\mathcal{E}$, and for each $e \in \mathcal{E}$ find

$$P(y_e = 1 \mid x_{\mathcal{M}(e)}). \tag{1}$$

Here $y \in \{0, 1\}$ is the entity-level label where $y_e = 1$ means a person (entity) e was killed by police; $x_{\mathcal{M}(e)}$ are the sentences containing mentions $\mathcal{M}(e)$ of that person. A mention $i \in \mathcal{M}(e)$ is a token span in the corpus. Most entities have

[2] Konovalov et al. (2017) studies the database update task where edits to Wikipedia infoboxes constitute events.

Knowledge base	Historical	Test
FE incident dates	Jan 2000 – Aug 2016	Sep 2016 – Dec 2016
FE gold entities ($\mathcal{G}$)	17,219	452
News dataset	**Train**	**Test**
doc. dates	Jan 2016 – Aug 2016	Sep 2016 – Dec 2016
total docs. ($\mathcal{D}$)	866,199	347,160
total ments. ($\mathcal{M}$)	132,833	68,925
pos. ments. ($\mathcal{M}^+$)	11,274	6,132
total entities ($\mathcal{E}$)	49,203	24,550
pos. entities ($\mathcal{E}^+$)	916	258

Table 2: Data statistics for Fatal Encounters (FE) and scraped news documents. $\mathcal{M}$ and $\mathcal{E}$ result from NER processing, while $\mathcal{E}^+$ results from matching textual named entities against the gold-standard database ($\mathcal{G}$).

multiple mentions; a single sentence can contain multiple mentions of different entities.

3.2 News documents

We download a collection of web news articles by continually querying Google News[3] throughout 2016 with lists of police keywords (i.e police, officer, cop etc.) and fatality-related keywords (i.e. kill, shot, murder etc.). The keyword lists were constructed semi-automatically from cosine similarity lookups from the *word2vec* pretrained word embeddings[4] in order to select a high-recall, broad set of keywords. The search is restricted to what Google News defines as a "regional edition" of "United States (English)" which seems to roughly restrict to U.S. news though we anecdotally observed instances of news about events in the U.K. and other countries. We apply a pipeline of text extraction, cleaning, and sentence de-duplication described in the appendix.

3.3 Entity and mention extraction

We process all documents with the open source *spaCy* NLP package[5] to segment sentences, and extract entity mentions. Mentions are token spans that (1) were identified as "persons" by spaCy's named entity recognizer, and (2) have a (firstname, lastname) pair as analyzed by the HAPNIS rule-based name parser,[6] which extracts, for example, (*John*, *Doe*) from the string *Mr. John A. Doe Jr.*.[7]

	x	z	y
"Hard" training	observed	fixed (distantly labeled)	observed
"Soft" (EM) training	observed	latent	observed
Testing	observed	latent	latent

Table 3: Training and testing settings for mention sentences x, mention labels z, and entity labels y.

To prepare sentence text for modeling, our preprocessor collapses the candidate mention span to a special TARGET symbol. To prevent overfitting, other person names are mapped to a different PERSON symbol; e.g. "TARGET was killed in an encounter with police officer PERSON."

There were initially 18,966,757 and 6,061,717 extracted mentions for the train and test periods respectively. To improve precision and computational efficiency, we filtered to sentences that contained at least one police keyword and one fatality keyword. This filter reduced positive entity recall a moderate amount (from 0.68 to 0.57), but removed 99% of the mentions, resulting in the $|\mathcal{M}|$ counts in Table 2.[8]

Other preprocessing steps included heuristics for extraction and name cleanups and are detailed in the appendix.

4 Models

Our goal is to classify entities as to whether they have been killed by police (§4.1). Since we do not have gold-standard labels to train our model, we turn to *distant supervision* (Craven and Kumlien, 1999; Mintz et al., 2009), which heuristically aligns facts in a knowledge base to text in a corpus to impute positive mention-level labels for supervised learning. Previous work typically examines distant supervision in the context of binary relation extraction (Bunescu and Mooney, 2007; Riedel et al., 2010; Hoffmann et al., 2011), but we are concerned with the unary predicate "person was killed by police." As our gold standard knowledge

[3] https://news.google.com/

[4] https://code.google.com/archive/p/word2vec/

[5] Version 0.101.0, https://spacy.io/

[6] http://www.umiacs.umd.edu/~hal/HAPNIS/

[7] For both training and testing, we use a name matching assumption that a (firstname, lastname) match indicates coreference between mentions, and between a mention and a fatality database entity. This limitation does affect a small number of instances—the test set database contains the unique names of 453 persons but only 451 unique (firstname, lastname) tuples—but relaxing it raises complex issues for future work, such as how to evaluate whether a system correctly predicted two different fatality victims with the same name.

[8] In preliminary experiments, training and testing an n-gram classifier (§4.4) on the full mention dataset without keyword filtering resulted in a worse AUPRC than after the filter.

base ($\mathcal{G}$), we use Fatal Encounters' (FE) publicly available dataset: around 18,000 entries of victim's name, age, gender and race as well as location, cause and date of death. (We use a version of the FE database downloaded Feb. 27, 2017.) We compare two different distant supervision training paradigms (Table 3): "hard" label training (§4.2) and "soft" EM-based training (§4.3). This section also details mention-level models (§4.4,§4.5) and evaluation (§4.6).

4.1 Approach: Latent disjunction model

Our discriminative model is built on mention-level probabilistic classifiers. Recall a single entity will have one or more mentions (i.e. the same name occurs in multiple sentences in our corpus). For a given mention i in sentence x_i, our model predicts whether the person is described as having been killed by police, $z_i = 1$, with a binary logistic model,

$$P(z_i = 1 \mid x_i) = \sigma(\beta^{\mathsf{T}} f_\gamma(x_i)). \tag{2}$$

We experiment with both logistic regression (§4.4) and convolutional neural networks (§4.5) for this component, which use logistic regression weights β and feature extractor parameters γ. Then we must somehow aggregate mention-level decisions to determine entity labels y_e.[9] If a human reader were to observe at least one sentence that states a person was killed by police, they would infer that person was killed by police. Therefore we aggregate an entity's mention-level labels with a deterministic disjunction:

$$P(y_e = 1 \mid z_{\mathcal{M}(e)}) = 1\left\{\vee_{i\in\mathcal{M}(e)}\, z_i\right\}. \tag{3}$$

At test time, z_i is latent. Therefore the correct inference for an entity is to marginalize out the model's uncertainty over z_i:

$$\begin{aligned} P(y_e = 1|x_{\mathcal{M}(e)}) &= 1 - P(y_e = 0|x_{\mathcal{M}(e)}) && (4)\\ &= 1 - P(z_{\mathcal{M}(e)} = \vec{0} \mid x_{\mathcal{M}(e)}) && (5)\\ &= 1 - \prod_{i\in\mathcal{M}(e)} (1 - P(z_i = 1 \mid x_i)). && (6) \end{aligned}$$

Eq. 6 is the *noisyor* formula (Pearl, 1988; Craven and Kumlien, 1999). Procedurally, it counts strong probabilistic predictions as evidence, but can also incorporate a large number of weaker signals as positive evidence as well.[10]

In order to train these classifiers, we need mention-level labels (z_i) which we impute via two different distant supervision labeling methods: "hard" and "soft."

4.2 "Hard" distant label training

In "hard" distant labeling, labels for mentions in the training data are heuristically imputed and directly used for training. We use two labeling rules. First, **name-only:**

$$z_i = 1 \text{ if } \exists e \in \mathcal{G}^{(train)} : \text{name}(i) = \text{name}(e). \tag{7}$$

This is the direct unary predicate analogue of Mintz et al. (2009)'s *distant supervision assumption*, which assumes every mention of a gold-positive entity exhibits a description of a police killing.

This assumption is not correct. We manually analyze a sample of positive mentions and find 36 out of 100 name-only sentences did not express a police fatality event—for example, sentences contain commentary, or describe killings not by police. This is similar to the precision for distant supervision of binary relations found by Riedel et al. (2010), who reported 10–38% of sentences did not express the relation in question.

Our higher precision rule, **name-and-location**, leverages the fact that the location of the fatality is also in the Fatal Encounters database and requires both to be present:

$$\begin{aligned} &z_i = 1 \text{ if } \exists e \in \mathcal{G}^{(train)} :\\ &\text{name}(i) = \text{name}(e) \text{ and location}(e) \in x_i. \end{aligned} \tag{8}$$

We use this rule for training since precision is slightly better, although there is still a considerable level of noise.

4.3 "Soft" (EM) joint training

At training time, the *distant supervision assumption* used in "hard" label training is flawed: many positively-labeled mentions are in sentences that

[9] An alternative approach is to aggregate features across mentions into an entity-level feature vector (Mintz et al., 2009; Riedel et al., 2010); but here we opt to directly model at the mention level, which can use contextual information.

[10] In early experiments, we experimented with other, more ad-hoc aggregation rules with a "hard"-trained model. The maximum and arithmetic mean functions performed worse than *noisyor*, giving credence to the disjunction model. The sum rule ($\sum_i P(z_i = 1 \mid x_i)$) had similar ranking performance as *noisyor*—perhaps because it too can use weak signals, unlike mean or max—though it does not yield proper probabilities between 0 and 1.

do not assert the person was killed by a police officer. Alternatively, at training time we can treat z_i as a latent variable and assume, as our model states, that *at least one* of the mentions asserts the fatality event, but leave uncertainty over which mention (or multiple mentions) conveys this information. This corresponds to multiple instance learning (MIL; Dietterich et al. (1997)) which has been applied to distantly supervised relation extraction by enforcing the *at least one* constraint at training time (Bunescu and Mooney, 2007; Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012; Ritter et al., 2013). Our approach differs by using exact marginal posterior inference for the E-step.

With z_i as latent, the model can be trained with the EM algorithm (Dempster et al., 1977). We initialize the model by training on the "hard" distant labels (§4.2), and then learn improved parameters by alternating E- and M-steps.

The **E-step** requires calculating the marginal posterior probability for each z_i,

$$q(z_i) := P(z_i \mid x_{\mathcal{M}(e_i)}, y_{e_i}). \tag{9}$$

This corresponds to calculating the posterior probability of a disjunct, given knowledge of the output of the disjunction, and prior probabilities of all disjuncts (given by the mention-level classifier).

Since $P(z \mid x, y) = P(z, y \mid x)/P(y \mid x)$,

$$q(z_i = 1) = \frac{P(z_i = 1, y_{e_i} = 1 | x_{\mathcal{M}(e_i)})}{P(y_{e_i} = 1 | x_{\mathcal{M}(e_i)})}. \tag{10}$$

The numerator simplifies to the mention prediction $P(z_i = 1 \mid x_i)$ and the denominator is the entity-level *noisyor* probability (Eq. 6). This has the effect of taking the classifier's predicted probability and increasing it slightly (since Eq. 10's denominator is no greater than 1); thus the disjunction constraint implies a soft positive labeling. In the case of a negative entity with $y_e = 0$, the disjunction constraint implies all $z_{\mathcal{M}(e)}$ stay clamped to 0 as in the "hard" label training method.

The $q(z_i)$ posterior weights are then used for the **M-step**'s expected log-likelihood objective:

$$\max_\theta \sum_i \sum_{z \in \{0,1\}} q(z_i = z) \log P_\theta(z_i = z \mid x_i). \tag{11}$$

This objective (plus regularization) is maximized with gradient ascent as before.

This approach can be applied to any mention-level probabilistic model; we explore two in the next sections.

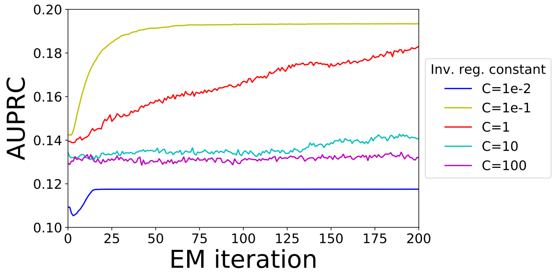


Figure 1: For soft-LR (EM), area under precision recall curve (AUPRC) results on the test set during training, for different inverse regularization values (C, the parameters' prior variance).

	Features
$D1$	length 3 dependency paths that include TARGET: word, POS, dep. label
$D2$	length 3 dependency paths that include TARGET: word and dep. label
$D3$	length 3 dependency paths that include TARGET: word and POS
$D4$	all length 2 dependency paths with word, POS, dep. labels
$N1$	n-grams length 1, 2, 3
$N2$	n-grams length 1, 2, 3 plus POS tags
$N3$	n-grams length 1, 2, 3 plus directionality and position from TARGET
$N4$	concatenated POS tags of 5-word window centered on TARGET
$N5$	word and POS tags for 5-word window centered on TARGET

Table 4: Feature templates for logistic regression grouped into syntactic dependencies (D) and N-gram (N) features.

4.4 Feature-based logistic regression

We construct hand-crafted features for regularized logistic regression (LR) (Table 4), designed to be broadly similar to the n-gram and syntactic dependency features used in previous work on feature-based semantic parsing (e.g. Das et al. (2014); Thomson et al. (2014)). We use randomized feature hashing (Weinberger et al., 2009) to efficiently represent features in 450,000 dimensions, which achieved similar performance as an explicit feature representation. The logistic regression weights (β in Eq. 2) are learned with *scikit-learn* (Pedregosa et al., 2011).[11] For EM (soft-LR) training, the test set's area under the precision recall curve converges after 96 iterations (Fig. 1).

[11] With *FeatureHasher*, L2 regularization, 'lbfgs' solver, and inverse strength $C = 0.1$, tuned on a development dataset in "hard" training; for EM training the same regularization strength performs best.

4.5 Convolutional neural network

We also train a convolutional neural network (CNN) classifier, which uses word embeddings and their nonlinear compositions to potentially generalize better than sparse lexical and n-gram features. CNNs have been shown useful for sentence-level classification tasks (Kim, 2014; Zhang and Wallace, 2015), relation classification (Zeng et al., 2014) and, similar to this setting, event detection (Nguyen and Grishman, 2015). We use Kim (2014)'s open-source CNN implementation,[12] where a logistic function makes the final mention prediction based on max-pooled values from convolutional layers of three different filter sizes, whose parameters are learned (γ in Eq. 2). We use pretrained word embeddings for initialization,[13] and update them during training. We also add two special vectors for the TARGET and PERSON symbols, initialized randomly.[14]

For training, we perform stochastic gradient descent for the negative expected log-likelihood (Eq. 11) by sampling with replacement fifty mention-label pairs for each minibatch, choosing each $(i, k) \in \mathcal{M} \times \{0, 1\}$ with probability proportional to $q(z_i = k)$. This strategy attains the same expected gradient as the overall objective. We use "epoch" to refer to training on 265,700 examples (approx. twice the number of mentions). Unlike EM for logistic regression, we do not run gradient descent to convergence, instead applying an E-step every two epochs to update q; this approach is related to incremental and online variants of EM (Neal and Hinton, 1998; Liang and Klein, 2009), and is justified since both SGD and E-steps improve the evidence lower bound (ELBO). It is also similar to Salakhutdinov et al. (2003)'s expectation gradient method; their analysis implies the gradient calculated immediately after an E-step is in fact the gradient for the marginal log-likelihood. We are not aware of recent work that uses EM to train latent-variable neural network models, though this combination has been explored (e.g. Jordan and Jacobs (1994))

4.6 Evaluation

On documents from the test period (Sept–Dec 2016), our models predict entity-level labels $P(y_e = 1 \mid x_{\mathcal{M}(e)})$ (Eq. 6), and we wish to evaluate whether retrieved entities are listed in Fatal Encounters as being killed during Sept–Dec 2016. We rank entities by predicted probabilities to construct a precision-recall curve (Fig. 4, Table 5). Area under the precision-recall curve (AUPRC) is calculated with a trapezoidal rule; F1 scores are shown for convenient comparison to non-ranking approaches (§5).

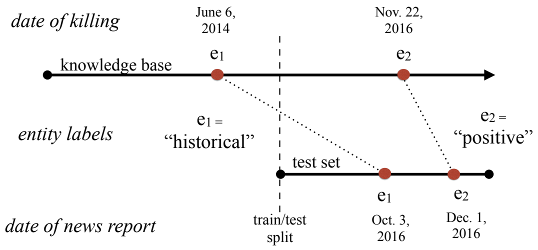


Figure 2: At test time, there are matches between the knowledge base and the news reports both for persons killed during the test period ("positive") and persons killed before it ("historical"). Historical cases are excluded from evaluation.

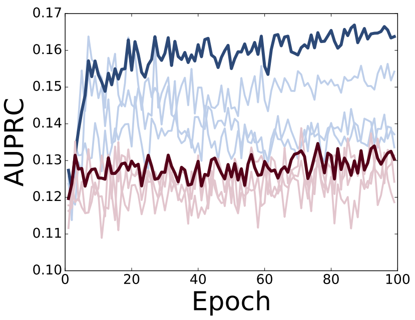


Figure 3: Test set AUPRC for three runs of soft-CNN (EM) (**blue**, higher in graph), and hard-CNN (**red**, lower in graph). Darker lines show performance of averaged predictions.

Excluding historical fatalities: Our model gives strong positive predictions for many people who were killed by police before the test period (i.e. before Sept 2016), when news articles contain discussion of historical police killings. We exclude these entities from evaluation, since we want to simulate an update to a fatality database (Fig 2). Our test dataset contains 1,148 such historical entities.

Data upper bound: Of the 452 gold entities in the FE database at test time, our news corpus only contained 258 (Table 2), hence the data up-

[12] https://github.com/yoonkim/CNN_sentence

[13] From the same *word2vec* embeddings used in §3.

[14] Training proceeds with ADADELTA (Zeiler, 2012). We tested several different settings of dropout and L2 regularization hyperparameters on a development set, but found mixed results, so used their default values.

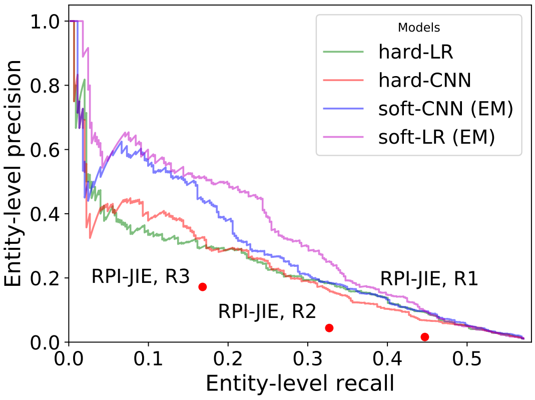


Figure 4: Precision-recall curves for the given models.

Model	AUPRC	F1
hard-LR, dep. feats.	0.117	0.229
hard-LR, n-gram feats.	0.134	0.257
hard-LR, all feats.	0.142	0.266
hard-CNN	0.130	0.252
soft-CNN (EM)	0.164	0.267
soft-LR (EM)	**0.193**	**0.316**
Data upper bound (§4.6)	0.57	0.73

Table 5: Area under precision-recall curve (AUPRC) and F1 (its maximum value from the PR curve) for entity prediction on the test set.

per bound of 0.57 recall, which also gives an upper bound of 0.57 on AUPRC. This is mostly a limitation of our news corpus; though we collect hundreds of thousands of news articles, it turns out Google News only accesses a subset of relevant web news, as opposed to more comprehensive data sources manually reviewed by Fatal Encounters' human experts. We still believe our dataset is large enough to be realistic for developing better methods, and expect the same approaches could be applied to a more comprehensive news corpus.

5 Off-the-shelf event extraction baselines

From a practitioner's perspective, a natural first approach to this task would be to run the corpus of police fatality documents through pre-trained, "off-the-shelf" event extractor systems that could identify killing events. In modern NLP research, a major paradigm for event extraction is to formulate a hand-crafted ontology of event classes, annotate a small corpus, and craft supervised learning systems to predict event parses of documents.

	Rule	Prec.	Recall	F1
SEMAFOR	R1	0.011	0.436	0.022
	R2	0.031	0.162	0.051
	R3	0.098	0.009	0.016
RPI-JIE	R1	0.016	0.447	0.030
	R2	0.044	0.327	0.078
	R3	0.172	0.168	**0.170**
Data upper bound (§4.6)		1.0	0.57	0.73

Table 6: Precision, recall, and F1 scores for test data using event extractors SEMAFOR and RPI-JIE and rules R1-R3 described below.

We evaluate two freely available, off-the-shelf event extractors that were developed under this paradigm: SEMAFOR (Das et al., 2014), and the RPI Joint Information Extraction System (RPI-JIE) (Li and Ji, 2014), which output semantic structures following the FrameNet (Fillmore et al., 2003) and ACE (Doddington et al., 2004) event ontologies, respectively.[15] Pavlick et al. (2016) use RPI-JIE to identify instances of gun violence.

For each mention $i \in \mathcal{M}$ we use SEMAFOR and RPI-JIE to extract event tuples of the form $t_i =$ (event type, agent, patient) from the sentence x_i. We want the system to detect (1) killing events, where (2) the killed person is the target mention i, and (3) the person who killed them is a police officer. We implement a small progression of these neo-Davidsonian (Parsons, 1990) conjuncts with rules to classify $z_i = 1$ if:[16]

- **(R1)** the event type is 'kill.'
- **(R2)** R1 holds and the patient token span contains e_i.

[15] Many other annotated datasets encode similar event structures in text, but with lighter ontologies where event classes directly correspond with lexical items—including PropBank, Prague Treebank, DELPHI-IN MRS, and Abstract Meaning Representation (Kingsbury and Palmer, 2002; Hajic et al., 2012; Oepen et al., 2014; Banarescu et al., 2013). We assume such systems are too narrow for our purposes, since we need an extraction system to handle different trigger constructions like "killed" versus "shot dead."

[16] For SEMAFOR, we use the FrameNet 'Killing' frame with frame elements 'Victim' and 'Killer'. For RPI-JIE, we use the ACE 'life/die' event type/subtype with roles 'victim' and 'agent'. SEMAFOR defines a token span for every argument; RPI-JIE/ACE defines two spans, both a head word and entity extent; we use the entity extent. SEMAFOR only predicts spans as event arguments, while RPI-JIE also predicts entities as event arguments, where each entity has a within-text coreference chain over one or more mentions; since we only use single sentences, these chains tend to be small, though they do sometimes resolve pronouns. For determining R2 and R3, we allow a match on any of an entity's extents from any of its mentions.

- **(R3)** R2 holds and the agent token span contains a police keyword.

As in §4.1 (Eq. 3), we aggregate mention-level z_i predictions to obtain entity-level predictions with a deterministic OR of $z_{\mathcal{M}(e)}$.

RPI-JIE under the full R3 system performs best, though all results are relatively poor (Table 6). Part of this is due to inherent difficulty of the task, though our task-specific model still outperforms (Table 5). We suspect a major issue is that these systems heavily rely on their annotated training sets and may have significant performance loss on new domains, or messy text extracted from web news, suggesting domain transfer for future work.

6 Results and discussion

Significance testing: We would like to test robustness of performance results to the finite datasets with bootstrap testing (Berg-Kirkpatrick et al., 2012), which can accomodate performence metrics like AUPRC. It is not clear what the appropriate unit of resampling should be—for example, parsing and machine translation research in NLP often resamples sentences, which is inappropriate for our setting. We elect to resample documents in the test set, simulating variability in the generation and retrieval of news articles. Standard errors for one model's AUPRC and F1 are in the range 0.004–0.008 and 0.008–0.010 respectively; we also note pairwise significance test results. See appendix for details.

Overall performance: Our results indicate our model is better than existing computational methods methods to extract names of people killed by police, by comparing to F1 scores of off-the-shelf extractors (Table 5 vs. Table 6; differences are statistically significant).

We also compare entities extracted from our test dataset to the Guardian's "The Counted" database of U.S. police killings during the span of the test period (Sept.–Dec., 2016),[17] and found 39 persons they did not include in the database, but who were in fact killed by police. This implies our approach could augment journalistic collection efforts. Additionally, our model could help practitioners by presenting them with sentence-level information in the form of Table 7; we hope this could decrease the amount of time and emotional toll required to maintain real-time updates of police fatality databases.

CNN: Model predictions were relatively unstable during the training process. Despite the fact that EM's evidence lower bound objective ($H(Q) + E_Q[\log P(Z, Y|X)]$) converged fairly well on the training set, test set AUPRC substantially fluctuated as much as 2% between epochs, and also between three different random initializations for training (Fig. 3). We conducted these multiple runs initially to check for variability, then used them to construct a basic ensemble: we averaged the three models' mention-level predictions before applying *noisyor* aggregation. This outperformed the individual models—especially for EM training—and showed less fluctuation in AUPRC, which made it easier to detect convergence. Reported performance numbers in Table 5 are with the average of all three runs from the final epoch of training.

LR vs. CNN: After feature ablation we found that hard-CNN and hard-LR with n-gram features (N1-N5) had comparable AUPRC values (Table 5). But adding dependency features (D1-D4) caused the logistic regression models to outperform the neural networks (albeit with bare significance: $p = 0.046$). We hypothesize these dependency features capture longer-distance semantic relationships between the entity, fatality trigger word, and police officer, which short n-grams cannot. Moving to sequence or graph LSTMs may better capture such dependencies.

Soft (EM) training: Using the EM algorithm gives substantially better performance: for the CNN, AUC improves from 0.130 to 0.164, and for LR, from 0.142 to 0.193. (Both improvements are statistically significant.) Logistic regression with EM training is the most accurate model. Examining the precision-recall curves (Fig. 4), many of the gains are in the higher confidence predictions (left side of figure). In fact, the soft EM model makes fewer strongly positive predictions: for example, hard-LR predicts $y_e = 1$ with more than 99% confidence for 170 out of 24,550 test set entities, but soft-LR does so for only 24. This makes sense given that the hard-LR model at training time assumes that many more positive entity mentions are evidence of a killing than they are in reality (§4.2).

Manual analysis: Manual analysis of false positives indicates misspellings or mismatches of names, police fatalities outside of the U.S., people who were shot by police but not killed, and names of police officers who were killed are com-

[17] https://www.theguardian.com/us-news/series/counted-us-police-killings, downloaded Jan. 1, 2017.

entity (e)	ment.(i) prob.	ment. text (x_i)
Keith Scott (true pos)	0.98	Charlotte protests Charlotte's Mayor Jennifer Roberts speaks to reporters the morning after protests against the police shooting of **Keith Scott**, in Charlotte, North Carolina .
Terence Crutcher (true pos)	0.96	Tulsa Police Department released video footage Monday, Sept. 19, 2016, showing white Tulsa police officer Betty Shelby fatally shooting **Terence Crutcher,** 40, a black man police later determined was unarmed.
Mark Duggan (false pos)	0.97	The fatal shooting of **Mark Duggan** by police led to some of the worst riots in England's recent history.
Logan Clarke (false pos)	0.92	**Logan Clarke** was shot by a campus police officer after waving kitchen knives at fellow students outside the cafeteria at Hug High School in Reno, Nevada, on December 7.

Table 7: Example of highly ranked entities, with selected mention predictions and text.

mon false positive errors (see detailed table in the appendix). This suggests many prediction errors are from ambiguous or challenging cases.[18]

Future work: While we have made progress on this application, more work is necessary for accuracy to be high enough to be useful for practitioners. Our model allows for the use of mention-level semantic parsing models; systems with explicit trigger/agent/patient representations, more like traditional event extraction systems, may be useful, as would more sophisticated neural network models, or attention models as an alternative to disjunction aggregation (Lin et al., 2016).

One goal is to use our model as part of a semi-automatic system, where people manually review a ranked list of entity suggestions. In this case, it is more important to focus on improving recall—specifically, improving precision at high-recall points on the precision-recall curve. Our best models, by contrast, tend to improve precision at lower-recall points on the curve. Higher recall may be possible through cost-sensitive training (e.g. Gimpel and Smith (2010)) and using features from beyond single sentences within the document.

Furthermore, our dataset could be used to contribute to communication studies, by exploring research questions about the dynamics of media attention (for example, the effect of race and geography on coverage of police killings), and discussions of historical killings in news—for example, many articles in 2016 discussed Michael Brown's 2014 death in Ferguson, Missouri. Improving NLP analysis of historical events would also be useful for the event extraction task itself, by delineating between recent events that require a database update, versus historical events that appear as "noise" from the perspective of the database update task. Finally, it may also be possible to adapt our model to extract other types of social behavior events.

Acknowledgments

This work was partially supported by the Amazon Web Services (AWS) Cloud Credits for Research program. Thanks to D. Brian Burghart for advice on police fatalities tracking, and to David Belanger, Trapit Bansal, Patrick Verga, Rajarshi Das, and Taylor Berg-Kirkpatrick for feedback.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, Sofia, Bulgaria, pages 178–186. http://www.aclweb.org/anthology/W13-2322.

Duren Banks, Paul Ruddle, Erin Kennedy, and Michael G. Planty. 2016. Arrest-related deaths program redesign study, 2015–16: Preliminary findings. Technical report, Technical Report NCJ 250112.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of EMNLP*.

Elizabeth Boschee, Premkumar Natarajan, and Ralph Weischedel. 2013. Automatic extraction of events from open source text for predictive forecasting. *Handbook of Computational Approaches to Counterterrorism* page 51.

Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of*

[18] We attempted to correct non-U.S. false positive errors by using CLAVIN, an open-source country identifier, but this significantly hurt recall.

ACL. Association for Computational Linguistics, Prague, Czech Republic, pages 576–583. http://www.aclweb.org/anthology/P07-1073.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*. pages 77–86.

Mark Craven, Andrew McCallum, Dan PiPasquo, Tom Mitchell, and Dayne Freitag. 1998. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of AAAI*.

Dipanjan Das, Desai Chen, Andre F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics* .

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (methodological)* pages 1–38.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence* 89(1):31–71.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ACE) program-tasks, data, and evaluation. In *LREC*. volume 2, page 1.

Charles J. Fillmore, Christopher R. Johnson, and Miriam R.L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography* .

Sean M Gerrish. 2013. *Applications of Latent Variable Models in Modeling Influence and Decision Making*. Ph.D. thesis, Princeton University.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin CRFs: Training log-linear models with cost functions. In *Proceedings of NAACL-HLT*. Association for Computational Linguistics, pages 733–736.

Jan Hajic, Eva Hajicová, Jarmila Panevová, Petr Sgall, Ondrej Bojar, Silvie Cinková, Eva Fucíková, Marie Mikulová, Petr Pajas, Jan Popelka, et al. 2012. Announcing prague czech-english dependency treebank 2.0. In *LREC*. pages 3153–3160.

Alex Hanna. 2017. MPEDS: Automating the generation of protest event data. *SocArXiv* .

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL*. Association for Computational Linguistics, Portland, Oregon, USA, pages 541–550. http://www.aclweb.org/anthology/P11-1055.

Eduard Hovy, Teruko Mitamura, Felisa Verdejo, Jun Araki, and Andrew Philpot. 2013. Events are not simple: Identity, non-identity, and quasi-identity. In *Workshop on Events: Definition, Detection, Coreference, and Representation*. Association for Computational Linguistics, Atlanta, Georgia, pages 21–28. http://www.aclweb.org/anthology/W13-1203.

Michael I Jordan and Robert A Jacobs. 1994. Hierarchical mixtures of experts and the em algorithm. *Neural computation* 6(2):181–214.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*.

Paul Kingsbury and Martha Palmer. 2002. From Tree-Bank to PropBank. In *LREC*. pages 1989–1993.

Alexander Konovalov, Benjamin Strauss, Alan Ritter, and Brendan O'Connor. 2017. Learning to extract events from knowledge base revisions. In *Proceedings of WWW*.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of EMNLP*.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of ACL*.

Percy Liang and Dan Klein. 2009. Online EM for unsupervised models. In *Proceedings of NAACL*. Boulder, Colorado. http://www.aclweb.org/anthology/N/N09/N09-1069.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*. Association for Computational Linguistics, Berlin, Germany, pages 2124–2133. http://www.aclweb.org/anthology/P16-1200.

Kristian Lum and Patrick Ball. 2015. Estimating undocumented homicides with two lists and list dependence. *Human Rights Data Analysis Group* https://hrdag.org/wp-content/uploads/2015/07/2015-hrdag-estimating-undoc-homicides.pdf.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL*. Suntec, Singapore. http://www.aclweb.org/anthology/P/P09/P09-1113.

Radford M Neal and Geoffrey E Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, Springer, pages 355–368.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of ACL*.

Brendan O'Connor, Brandon Stewart, and Noah A. Smith. 2013. Learning to extract international relations from political context. In *Proceedings of ACL*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of SemEval*. http://www.aclweb.org/anthology/S14-2008.

Terence Parsons. 1990. *Events in the Semantics of English*. Cambridge, MA: MIT Press.

Ellie Pavlick, Heng Ji, Xiaoman Pan, and Chris Callison-Burch. 2016. The Gun Violence Database: A new task and data set for NLP. In *Proceedings of EMNLP*. https://aclweb.org/anthology/D16-1106.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Kevin Reschke, Martin Jankowiak, Mihai Surdeanu, Christopher D. Manning, and Daniel Jurafsky. 2014. Event extraction using distant supervision. In *Language Resources and Evaluation Conference (LREC)*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pages 148–163.

Alan Ritter, Luke Zettlemoyer, Oren Etzioni, et al. 2013. Modeling missing data in distant supervision for information extraction. *TACL* .

Ruslan Salakhutdinov, Sam T Roweis, and Zoubin Ghahramani. 2003. Optimization with EM and expectation-conjugate-gradient. In *Proceedings of ICML*.

Philip A. Schrodt. 2012. Precedents, progress, and prospects in political event data. *International Interactions* 38(4):546–569.

Philip A. Schrodt and Deborah J. Gerner. 1994. Validity assessment of a machine-coded event data set for the Middle East, 1982-1992. *American Journal of Political Science* .

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP*. http://www.aclweb.org/anthology/D12-1042.

Sam Thomson, Brendan O'Connor, Jeffrey Flanigan, David Bamman, Jesse Dodge, Swabha Swayamdipta, Nathan Schneider, Chris Dyer, and Noah A. Smith. 2014. CMU: Arc-factored, discriminative semantic dependency parsing. In *Proceedings of SemEval*. http://www.aclweb.org/anthology/S14-2027.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of ICML*.

Bishan Yang, Claire Cardie, and Peter Frazier. 2015. A hierarchical distance-dependent Bayesian model for event coreference resolution. *TACL* 3.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extractionwithout labelled data. In *Proceedings of EMNLP*.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820* .

Asking too much? The rhetorical role of questions in political discourse

Justine Zhang,[1] **Arthur Spirling,**[2] **Cristian Danescu-Niculescu-Mizil**[1]
[1]Cornell University, [2]New York University
jz727@cornell.edu, arthur.spirling@nyu.edu, cristian@cs.cornell.edu

Abstract

Questions play a prominent role in social interactions, performing rhetorical functions that go beyond that of simple informational exchange. The surface form of a question can signal the intention and background of the person asking it, as well as the nature of their relation with the interlocutor. While the informational nature of questions has been extensively examined in the context of question-answering applications, their rhetorical aspects have been largely understudied.

In this work we introduce an unsupervised methodology for extracting surface motifs that recur in questions, and for grouping them according to their latent rhetorical role. By applying this framework to the setting of question sessions in the UK parliament, we show that the resulting typology encodes key aspects of the political discourse—such as the bifurcation in questioning behavior between government and opposition parties—and reveals new insights into the effects of a legislator's tenure and political career ambitions.

1 Introduction

> "We'd now like to open the floor to shorter speeches disguised as questions..."
>
> – Steve Macone, New Yorker cartoon caption

Why do we ask questions? Perhaps we are seeking factual information that others hold, or maybe we are requesting a favor. Alternatively we could be simply making a rhetorical point, perhaps at the start of an academic paper.

Questions play a prominent role in social interactions (Goffman, 1976), performing a multitude of rhetorical functions that go beyond mere factual information gathering (Kearsley, 1976). While the informational component of questions has been well-studied in the context of question-answering applications, there is relatively little computational work addressing the rhetorical and social role of these basic dialogic units.

One domain where questions have a particularly salient rhetorical role is politics. The ability to question the actions and intentions of governments is a crucial part of democracy (Pitkin, 1967), particularly in parliamentary systems. Consequently, scholars have studied parliamentary questions in detail, in terms of their origins (Chester and Bowring, 1962), their institutionalization (Eggers and Spirling, 2014) and their importance for oversight (Proksch and Slapin, 2011). In particular, the United Kingdom's House of Commons, renowned for theatrical questions periods, has been studied in some depth. However, those accounts are largely qualitative in nature (Bull and Wells, 2012; Bates et al., 2014).

The present work: methodology. In order to approach these problems computationally, we introduce an unsupervised framework to structure the space of questions according to their rhetorical role. First, we identify common ways in which questions are phrased. To this end, we automatically extract these recurring surface forms, or *motifs*, based on the lexico-syntactic structure of the questions posed (Section 4). To capture rhetorical aspects we then group these motifs according to their role, relying on the intuition that this role is encoded in the type of *answer* a question receives. To operationalize this intuition we construct a latent question-answer space in which question motifs triggering similar answers are mapped to the same region (Section 5).

The present work: application. We apply this general framework to the political discourse that occurs during parliamentary question sessions in

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1558–1572
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

the British House of Commons, a new dataset which we make publicly available (Section 3). Our framework extracts intuitive question types ranging from narrow factual queries to pointed criticisms disguised as questions (Section 5, Table 1). We validate our framework by aligning these types with prior understandings of parliamentary proceedings from the political science literature (Section 6). In particular, previous work (Bates et al., 2014) has categorized questions asked in Parliament according to the intentions of the asker (e.g., to help the answerer, or to adversarially put them on the spot); we find a clear, predictive mapping between these expert-coded categories and the induced typology. We further show that the types of questions specific legislators tend to ask vary with whether they are part of the governing or opposition party, consistent with well-established accounts of partisan differences (Cowley, 2002; Spirling and McLean, 2007; Eggers and Spirling, 2014). Concretely, government legislators exhibit a preference for overtly friendly questions, while the opposition slants towards more aggressive question types.

We then apply our methodology to provide new insights into how a legislator's questioning behavior varies with their career trajectory. The pressures faced by legislators at various stages in their career are cross-cutting, and multiple possible hypotheses emerge. Younger, more enthusiastic legislators may be motivated to ask harder-hitting questions, but risk being passed over for future promotion if they are too combative (Cowley, 2002). Older legislators, whose opportunities for promotion are largely behind them and hence have "less to lose", may act more aggressively (Benedetto and Hix, 2007); or simply seek a quiet path to retirement. Viewing each group's behavior through the questions they ask brings evidence for the latter hypothesis that more tenured legislators are more aggressive, even when questioning their own leaders. In this way, their presence in the House of Commons, and their refusal to simply 'keep their heads down', facilitates a core component of democracy.

2 Related Work

Question-answering. Computationally, questions have received considerable attention in the context of question-answering (QA) systems—for a survey see Gupta and Gupta (2012)—with an emphasis on understanding their information need (Harabagiu, 2008). Techniques have been developed to categorize questions based on the nature of these information needs in the context of the TREC QA challenge (Harabagiu et al., 2000), and to identify questions asking for similar information (Shtok et al., 2012; Zhang et al., 2017; Jeon et al., 2005); questions have also been classified by topic (Cao et al., 2010) and quality (Treude et al., 2011; Ravi et al., 2014). In contrast, our work is not concerned with the information need central to QA applications, and instead focuses on the rhetorical aspect of questions.

Question types. To facilitate retrieval of frequently asked questions, Lytinen and Tomuro (2002) manually developed a typology of surface question forms (e.g., 'what'- and 'why'-questions) starting from Lehnerts' conceptual question categories (Lehnert, 1978). Question types were also hand annotated for dialog-act labeling, distinguishing between yes-no, wh-, open-ended and rhetorical questions (Dhillon et al., 2004). To complement this line of work, this paper introduces a completely unsupervised methodology to automatically build a domain-tailored question typology, bypassing the need for human annotation.

Pragmatic dimensions. One important pragmatic dimension of questions that has been previously studied computationally is their level of politeness (Danescu-Niculescu-Mizil et al., 2013; Aubakirova and Bansal, 2016); in the specific context of making requests, politeness was shown to correlate with the social status of the asker. Sachdeva and Kumaraguru (2017) studied another rhetorical aspect by examining linguistic attributes distinguishing serviceable requests addressed to police on social media from general conversation. Previous research has also been directed at identifying rhetorical questions (Bhattasali et al., 2015) and understanding the motivations of their "askers" (Ranganath et al., 2016). Using the relationship between questions and answers, our work examines the rhetorical and social aspect of questions without predefining a pragmatic dimension and without relying on labeled data. We also complement these efforts in analyzing a broader range of situations in which questions may be posed without an information-seeking intent.

Political discourse. Finally, our work contributes to a rapidly growing area of NLP applications to political domains (Monroe et al., 2008; Card et al., 2016; Gonzalez-Bailon et al., 2010; Niculae et al., 2015; Grimmer and Stewart, 2013; Grimmer et al., 2012; Iyyer et al., 2014b, inter alia). Particularly relevant are applications to discourse in congressional and parliamentary settings (Thomas et al., 2006; Boydstun et al., 2014; Rheault et al., 2016).

3 Data: Parliamentary Question Periods

The bulk of our analysis focuses on the questions asked, and responses given during parliamentary question periods in the British House of Commons. Below, we provide a brief overview of key features of this political system in general, as well as a description of the question period setting.

Parliamentary systems. Legislators in the House of Commons (*Members of Parliament*, henceforth *MPs* or *members*) belong to two main voting and debating *affiliations*: a *government* party which controls the executive and holds a majority of the seats in the chamber, and a set of *opposition* parties.[1] The executive is headed by the Prime Minister (PM) and run by a cabinet of *ministers*, high-ranking government MPs responsible for various departments such as finance and education.

Question periods. The House of Commons holds weekly, moderated *question periods*, in which MPs of all affiliations take turns to ask questions to (and theoretically receive answers from) government ministers for each department regarding their specific domains. Such events are a primary way in which legislators hold senior policy-makers responsible for their decisions. In practice, beyond narrow requests for information about specific policy points, MPs use their questions to critique or praise the government, or to self-promote; indeed, certain sessions, such as Questions to the Prime Minister, have gained renown for their partisan clashes, often fueled by the (mis)handling of a current crisis. The following question, asked to the Prime Minister by an opposition MP about contamination of the meat supply in 2013, encapsulates this odd mix of purposes:

> *"The Prime Minister is rightly shocked by the revelations that many food products contain 100% horse. Does he share my concern that, if tested, many of his answers may contain 100% bull?"*[2]

The moderated, relatively rigid format of questions periods, along with the multifaceted array of underlying incentives and interpersonal relationships, yields a structurally controlled setting with a rich variety of social interactions, taking place in the realm of important policy discussions. This complexity makes question periods a particularly fruitful and consequential setting in which to study questions as social signals, and expand our understanding of their role beyond factual queries.

Dataset description. Our dataset covers question periods from May 1979 to December 2016, encompassing six different Prime Ministers. For each question period, we extract all question-answer pairs, along with the identity of the asker and answerer. Because our focus here is on how questions are posed in a social setting, and not on the subsequent dialogue, we ignore questions which were tabled prior to the session, as well as any followup back-and-forth dialogue between the asker and answerer.

We augment this collection with metadata about each asker and answerer, including their political party, the time when they first took office, and whether they were serving as a minister at a given point in time. Such information is used to validate our methodology and interpret our results in light of the social context in which the questions were asked, described further in Sections 6 and 7.

In total there are 216,894 question-answer pairs in our data, occurring over 4,776 days and 6 prime-ministerships. The questions cover 1,975 different askers, 1,066 different answerers, and a variety of government departments with responsibilities ranging from defense to transport. We make this dataset publicly available, along with the code implementing our methodology, as part of the Cornell Conversational Analysis Toolkit.[3]

4 Question Motifs

The first component of our framework identifies lexico-syntactic phrasing patterns recurring in a collection of questions, which we call *motifs*. Intuitively, motifs constitute wordings commonly used to pose questions. To find motifs in a given collection, we first extract relevant *fragments* from each question. We then group sets of frequently co-occurring fragments into motifs.

[1] We use *affiliation* to refer broadly to the government and opposition roles, independent of the identity of the current government and opposition parties. In subsequent analysis we only consider the largest, "official" opposition party as the opposition.

[2] MPs almost always address each other in 3rd person.

[3] `https://github.com/CornellNLP/Cornell-Conversational-Analysis-Toolkit`

Question fragments. Our goal is to find motifs which reflect functional characteristics of questions. Hence, we start by extracting the key *fragments* within a question which encapsulate its functional nature. Following the intuition that the bulk of this functional information is contained in the root of a question's dependency parse along with its outgoing arcs (Iyyer et al., 2014a), we take the fragments of a question to be the root of its parse tree, along with each (root, child) pair. To capture cases when the operational word in the question is not connected to its root (such as "What..."), we also consider the initial unigram and bigram of a question as fragments. The following question has 5 fragments: what, what is, going→*, is←going and going→do.

(1) **What** **is** the minister **going** to **do** about ... ?

Because our goal is to capture topic-agnostic patterns, we ignore all fragments which contain a noun phrase (NP) or pronoun. NP subtrees are identified based on their outgoing dependencies to the root;[4] in the event that an NP starts with a WH-determiner (WDT), we consider (root, WDT) to be a fragment and drop the remainder of the NP.[5]

Finally, we note that some questions consist of multiple sub-questions ("What does the Minister think [...], *and* why [...]?"). For such questions, we recursively extract fragments from each child subtree in the same manner, starting from their roots.

From fragments to motifs. We define *motifs* as sets of question fragments that frequently co-occur (in at least n questions). We find motifs by applying the apriori algorithm (Agrawal and Srikant, 1994) to find these common itemsets. This results in a collection of motifs $\mathcal{M}$ which correspond to different question phrasings.[6] Examples of motifs are shown in Table 1.

Motifs can identify phrasings to varying degrees of specificity. For example, the singleton motif $\{$what is$\}$ corresponds to all questions starting with that bigram, while $\{$what is, going→do$\}$ narrows these down to questions also containing the fragment going→do. To model the specificity relation between motifs, we structure $\mathcal{M}$ as a directed acyclic graph where an edge points from a motif m_1 to another motif m_2 if the latter has exactly one more fragment in addition to those in m_1, corresponding to a narrower set of phrasings.

Motif-representation of a question. Finally, a question q contains a motif if it includes all of the fragments comprising that motif. We can hence capture the phrasing of a given question q using the subset of motifs it contains, structured as the subgraph $\mathcal{M}_q \subset \mathcal{M}$ induced by this subset. This directed subgraph represents the question at multiple levels of specificity simultaneously; in particular, the set of sinks (i.e., nodes with outdegree 0; henceforth *sink motifs*) of $\mathcal{M}_q$ is the most fine-grained way to specify the phrasing of q. For example $\{$what is, is←going, going→do$\}$ is the only sink motif of the question in example (1); its entire subgraph is shown in Figure 3 in the appendix.

5 Latent Question Types

The second component of our framework structures the space of questions according to their functional roles, thus going beyond the lexico-syntactic representation captured via motifs. The main intuition is that the nature of the answer that a question receives provides a good indication of its intention. Therefore, if two questions are phrased differently but answered in similar ways, the parallels exhibited by their answers should reflect commonalities in the askers' intentions.

To operationalize this intuition, we first construct a latent space based on answers, and then map question motifs (Section 4) to the same space. Using the resultant latent representations, we can then cluster questions with similar rhetorical functions, even if their surface forms are different.

Constructing a space of answers. In line with our focus on functional characterizations, we extract the fragments from each sentence of an *answer*, defined in the same way as question fragments. We then construct a term-document matrix, where terms correspond to answer fragments, and documents correspond to individual answers in the corpus. We filter out infrequent fragments occurring less than n_A times, reweight the rows of this matrix with tf-idf reweighting, and scale to unit norm, producing a fragment-answer matrix $\mathcal{A}$. We perform singular value decomposi-

[4] We take as NPs subtrees connected to the root with the following: nsubj, nsubjpass, dobj, iobj, pobj, attr.

[5] In the particular case of the Parliament dataset, removing NPs also removes conventional, partisan address terms (e.g. "my hon. Friend").

[6] In some cases, a pair of motifs almost always co-occurs in the same questions, making them redundant. We treat two motifs m_1 and m_2 as equivalent if, for some probability p, $\Pr(m_1|m_2) > p$ and $\Pr(m_2|m_1) > p$; we keep the smaller of the two as the representative motif, or pick one of them arbitrarily if they are of equal sizes.

tion on $\mathcal{A}$ and obtain a low-rank representation $\mathcal{A} \approx \hat{\mathcal{A}} = U_A S V_A^T$, for some rank d, where rows of U_A correspond to answer fragments and rows of V_A correspond to answers.[7]

Latent projection of question motifs. We can draw a natural correspondence between a question motif m and answer term t if m occurs in a question whose answer contains t. This enables us to compute representations of question motifs in the same space as $\hat{\mathcal{A}}$. Concretely, we construct a motif-question matrix $\mathcal{Q} = (q_{ij})$ where $q_{ij} = 1$ if motif i occurred in question j; we scale rows of $\mathcal{Q}$ to unit norm. To represent $\mathcal{Q}$ in the latent answer space, we solve for $\hat{\mathcal{Q}}$ in $\mathcal{Q} = \hat{\mathcal{Q}} S V_A^T$ as $\hat{\mathcal{Q}} = \mathcal{Q} V_A S^{-1}$, again scaling rows to unit norm. Row i of $\hat{\mathcal{Q}}$ then gives a d-dimensional representation of motif i, denoted $\hat{q}_i$.

Grouping similar questions. Finally, we identify *question types*—broad groups of similar motifs. Intuitively, if two motifs m_i and m_j have vectors q_i and q_j which are close together, they elicit answers that are close in the latent space, so are functionally similar in this sense. We use the K-Means algorithm (Pedregosa et al., 2011) to cluster motif vectors into k clusters; these clusters then constitute the desired set of question types.

To determine the type of a *particular* question q^*, we transform it to a binary vector (q_i^*) where $q_i^* = 1$ if motif i is a sink motif of q^*; using only sink motifs at this stage allows us to characterize a question according to the most specific representation of its phrasing, thus avoiding spurious associations resulting from more general motifs. We scale q^*, project it to the latent space as before, and assign the resultant projection $\hat{q}^*$ to a cluster t, hence determining its type.

Since question motifs and answer fragments have both been mapped to the same latent space (as rows of $\hat{\mathcal{Q}}$ and U_A respectively), we can also assign each answer fragment to a question type. This further facilitates interpretability through characterizing the answers commonly triggered by a particular type of question.

6 Validation

We now apply our general framework to the particular setting of parliamentary question periods, structuring the space of questions posed within these sessions according to their rhetorical function. To validate the induced typology, we quantitatively show that it recovers asker intentions in an expert-coded dataset, and qualitatively aligns with prior findings in the political science literature on parliamentary dynamics.

Question types in Parliament. We apply our motif extraction and question type induction pipeline to the questions in the parliamentary dataset.[8] Over 90% of the questions in the dataset contain at least one of the resulting 2,817 motifs; in subsequent analyses we discard questions without a matching motif. We apply our pipeline to the questions in the parliamentary dataset, and induce a typology of $k = 8$ question types to capture the rich array of questions represented in this space while preserving interpretability.

Table 1 displays extracted types, along with example questions, answers, and motifs.[9] The second author, a political scientist with domain expertise in the UK parliamentary setting, manually investigated each type and provided interpretable labels. For example, in questions of type 4, the asker is aware that his main premise is supported by the minister, and thus will be met with a positive statement backing the thrust of the question; we call this the **agreement** cluster. Types 6 and 7 are much more combative: in type 6 questions the asker explicitly attempts to force the minister to **concede/accept** a point that would undermine some government stance, while type 7 contains **condemnatory** questions that prompt the minister to justify a policy that is self-evidently bad in the eyes of the asker. In contrast, type 2 constitutes tamer **narrow** queries that require the minister to simply report on non-partisan matters of policy. (Extended interpretations in the appendix.)

Quantitative validation. We compare our output to a dataset of 1,256 questions asked to various Prime Ministers labeled by Bates et al. (2014)

[7] We experimented with grouping answer fragments into motifs as well, but found that most of the motifs produced were one fragment large. While future work could focus more on understanding consistent phrasings of answers, we note that at least in our chosen corpus, answers are longer and encompass a much greater variation of possible phrasings.

[8] We consider questions to be sentences ending in question marks. If an utterance consists of multiple questions, we extract fragments sets from each question separately, and take the motifs of the utterance to be the union of motifs of each component question. We set $n = 100$, $p = 0.9$, $n_A = 100$ and $d = 25$. The choice of parameters was done via manual inspection of the dataset.

[9] Each type contains a few hundred question motifs and answer fragments.

Type	Question motifs	Answer fragments	Example question-answer pairs
0: Issue update (16,693)	{what, are←taking}, {will←update}	continue→work, met→discuss	Q: **What** steps **are** the Department **taking** to create a system for asylum-seekers? A: We **continue** to **work** with the Department of Education to ensure an equitable [...]

1: Shared concerns (35,954)	{will←take},
	{may←urge}
		grateful←am,
		shall←consider
			Q: **Will** he **take** steps to support other MPs to employ apprentices?
			A: I **am grateful** for that suggestion [...]

2: Narrow factual (16,467)	{what←made},
	{what←happen, will←happen}
		is←considering,
		have←discussed
			Q: **What** representations has the Minister **made** on the future of rural policing [in] Dyfed-Powys?
			A:The Home Office **is considering** the matter [...]

3: Prompt for comment (16,588)	{what←say, say→to}, {will←tell}	must←say, said→was	Q: **What** has the Prime Minister to **say to** President Reagan for sending troops to Honduras? A: [...] I **must say** that we deplore the reported incursion by Nicaraguan forces [...]
4: Agreement (32,835)	{does←agree, agree→is}, {is→important}	agree→with, agree→completely	Q: **Does** [he] **agree** that one of the best ways to improve the trade balance **is** to continue the Government's strong economic policies? A: I **agree with** my hon. Friend [...]
5: Self promotion (26,351)	{is→aware}, {will←consider}	will←appreciate, am→certain	Q: **Is** my Friend **aware** that members of my parish church are pleased to have received a grant [...] ? A: [My Friend] **will appreciate** the significant performance of parishes up and down the country [...]

6: Concede, accept (31,653)	{will←accept},
	{is→not, is→true}
		not←accept,
		not←believe
			Q: **Will** [he] **accept** that [the UK exiting the EU] would undermine our security [...]?
			A: No, I do **not accept** that [...]

7: Condemnatory (21,320)	{can←explain}, {how←justify, can←justify}	knows→well, is→wrong	Q: **Can** the Secretary **explain** why the Government are scrapping child poverty targets? A: The hon. Lady **is wrong** in what she says [...]

Table 1: Question types automatically extracted from the parliamentary question periods, along with representative motifs and question-answer pairs. The number of questions in our dataset assigned to each type is shown in parantheses. Interpretations and more examples in Tables 2 & 3 in the appendix.

(also included in our data distribution). Each question in this data is hand-coded by a domain expert with one of three labels indicating the rhetorical intention of the asker: compared to *standard* questions—denoting straightforward factual queries, *helpful* questions serve as prompts for the PM to talk favorably about their government, while *unanswerable* questions are effectively vehicles for delivering criticisms that the PM cannot respond to. Questions which are *unanswered* by the PM are also labeled. If our framework captures meaningful rhetorical dimensions, we expect a given label to be over-represented in some of our induced types, and under-represented in others.

Even though our clustering of questions is generated in an unsupervised fashion without any guidance from the coded rhetorical roles, we see that several of the types we discover closely align with these annotations. In particular, helpful questions are highly associated with the **agreement** type (constituting 28% of questions of that type compared to 14% over the entire dataset; binomial test $p < 0.01$), reinforcing our interpretation that this type captures MPs cheerleading their own government. Conversely, unanswerable questions are frequently of the **concede/accept** type (20% in-type vs. 11% overall), while **condemnatory** questions are often unanswered (43% vs. 24% overall), suggesting that questions of these types have an increased tendency to be posed as aggressive criticisms packaged as questions.

We also validate our framework in a prediction setting using these labels, in three binary classification tasks: distinguishing helpful vs. standard, unanswerable vs. standard, and unanswered vs. answered questions. (In each task, we balance the two classes.) To control for asker affiliation effects, we consider only questions asked by government MPs for the helpful task, and opposition questions in the unanswerable and unanswered tasks; we train on questions to Conservative PMs and evaluate on Labour PMs.[10] For each setting, we train logistic regression classifiers; as

[10] These choices are motivated by the number of questions from each affiliation and party in the dataset (see appendix for further details on this dataset).

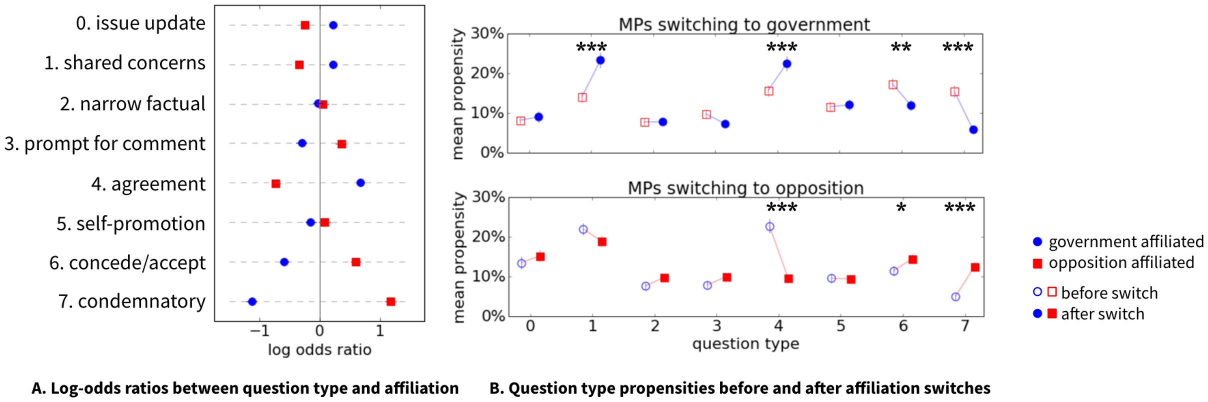


Figure 1: **A**: Log-odds ratios of questions of each type asked by government and opposition MPs, compared to MPs not of the respective affiliation; 95% confidence intervals (sometimes imperceptible) are depicted. **B**: Mean propensities for each question type, for MPs who switch from being in the opposition to being in the government (top) and vice-versa (bottom) after an election. Stars indicate statistically significant differences at the $p < 0.05$ (*), $p < 0.01$ (**) and $p < 0.001$ (***) levels (Wilcoxon test).

features we compare the latent representation of each question to a unigram BOW baseline.[11]

In the *unanswerable* and *unanswered* tasks, we find that the BOW features do not perform significantly better than a random (50%) baseline. However, the latent question features produced by our framework bring additional predictive signal and outperform the baseline when combined with BOW (binomial $p < 0.05$), achieving accuracies of 66% and 62% respectively (compared with 55% and 50% for BOW alone). This suggests that our representation captures useful rhetorical information that, given our train-test split, generalizes across parties. None of the models significantly outperform the random baseline on the *helpful* task, perhaps owing to the small data size.

Qualitative validation: question partisanship. We additionally provide a qualitative validation of our framework by comparing the question-asking activity of government and opposition-affiliated MPs—as viewed through the extracted question types—to well-established characterizations of these affiliations in the political science literature. In particular, prior work has examined the bifurcation in behavior between government and opposition members, in their differing focus on various issues (Louwerse, 2012), and in settings such as roll call votes (Cowley, 2002; Spirling and McLean, 2007; Eggers and Spirling, 2014). Since government MPs are elected on the same party ticket and manifesto, they primarily act to support the government's various policies and bolster the status of their cabinet, seldom airing disagreements publicly. In contrast, opposition members tend to offer trenchant partisan criticism of government policies, seeking to destabilize the government's relationship with its MPs and create negative press in the country at large. In characterizing the question-asking activity of government and opposition MPs, this friendly vs. adversarial behavior should also be reflected in a rhetorical typology of questions.[12]

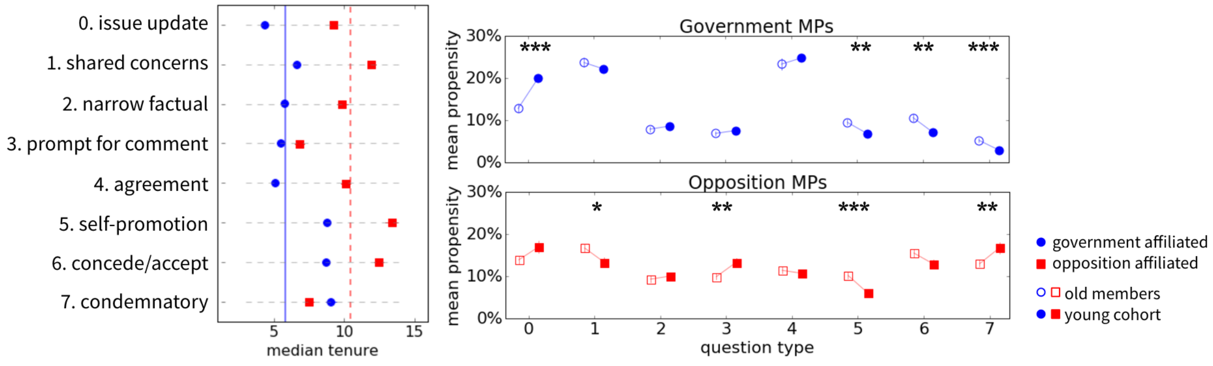
Concretely, to quantify the relationship between a particular question type t and asker affiliation $\mathcal{P}$, we compute the log-odds ratio of type t questions asked by MPs in $\mathcal{P}$, compared to MPs not in $\mathcal{P}$.[13]

Figure 1A shows the resultant log-odds ratios of each question type for government and opposition members. Notably, we see that **agreement**-type questions are significantly more likely to originate from government than from opposition MPs, while the opposite holds for **concede/accept** and **condemnatory** questions (binomial $p < 10^{-4}$ for each, comparing within-type to overall proportions of questions from an affiliation). No such

[11] We used tf-idf reweighting and excluded unigrams occurring less than 5 times.

[12] While we induce the typology over our entire dataset, we perform all subsequent analyses on a filtered subset of 50,152 questions. In particular, we omit utterances with multiple questions—i.e. multiple question marks—to ensure that we don't confound effects arising from different co-occurring question types. Our filtering decisions are also determined by the availability of information about the asker and answerers' roles in Parliament. Further information about these filtering choices can be found in the appendix.

[13] The log-odds values are not symmetric between government and opposition, because they includes questions asked by MPs not in the official opposition.



A. Median tenure of question askers

B. Question type propensities for old and young MPs

Figure 2: **A**: Median asker tenures over each question type, for government and opposition askers. Overall median tenures are also shown for reference (solid blue line for government, dashed red line for opposition). **B**: Mean propensities for newly elected MPs during the 1997 and 2010 elections, compared to re-elected MPs in the subsequent parliamentary sitting. Stars indicate statistically significant differences at the $p < 0.05$ (*), $p < 0.01$ (**) and $p < 0.001$ (***) levels (Mann Whitney U test).

slant is exhibited in the **narrow factual** type, further reinforcing the role of such questions as informational queries about relatively non-partisan issues. These results strongly cohere with the "textbook" accounts of parliamentary activity in the literature, as well as our interpretation of these types as bolstering or antagonistic.

Moreover, we find that the *same MP* shifts in her propensity for different question types as her affiliation changes. When a new political party is elected into office, MPs who were previously in the opposition now belong to the government party, and vice versa. Such a switch occurs within our data between the Major and Blair governments (Conservative to Labour, 1997), and between the Brown and Cameron governments (Labour to Conservative, 2010). For both switches, we consider all MPs who asked at least 5 questions both before and after the switch, resulting in 88 members who *became* government MPs and 102 who became opposition MPs. For an MP M we compute $P_{M,t}$, their *propensity* for a question type t, as the proportion of questions they ask which are from t. Comparing $P_{M,t}$ before and after a switch, we replicate the key differences observed above—for instance, we find that former opposition MPs who become government MPs decrease in their propensity for **condemnatory** questions, while newly opposition MPs move in the other direction (Wilcoxon $p < 0.001$, Figure 1B). This suggests that the general trends we observed before are driven by the shift in affiliation, and hence parliamentary role, of *individual* MPs.

7 Career Trajectory Effects

We now apply our framework to gain further insights into the nature of political discourse in Parliament, focusing on how questioning behavior varies with a member's tenure in the institution. As stated in the introduction, two alternative hypotheses arise: younger MPs may be more vigorously critical out of enthusiasm, but are potentially tempered by their stake in future promotion prospects compared to older members (Cowley, 2002, 2012). Alternatively, older MPs who have less at stake in terms of prospects of further promotion may ask more antagonistic questions. Throughout, *young* and *old* refer to tenure—i.e., how many years someone has served as an MP—rather than biological age.

In order to understand the extent to which young or old members contribute a specific type of question, for each question type t we compute the median tenure of askers of each question in t, and compare the median tenures of different question types, for each affiliation (Figure 2A).[14] We see that among both affiliations, more aggressive questions tend to originate more from older members, reflected in significantly higher median tenures (for types 6 in both affiliations, and 7 in government MPs; Mann Whitney U test $p < 0.001$ comparing within-type median tenure with outside-type median tenure); whereas standard **issue update** questions tend to come from younger

[14] Median tenures for opposition members are generally higher; winning an election tends to result in more newly-elected and therefore younger MPs (Webb and Farrell, 1999).

members ($p < 0.001$, both affiliations). Notably, the disproportionate aggressiveness of older members manifests even among *government* MPs who direct these questions towards their *own* government. This supports the "less to lose" intuition, offering a rhetorical parallel to previous findings about the increased tendency to vote contrary to party lines from MPs with little chance of ministerial promotion (Benedetto and Hix, 2007).

Interestingly, we find that these differential preferences across member tenure also manifest at a finer granularity than simply less to more aggressive. For instance, younger *opposition* members tend to contribute more **condemnatory** questions compared to older members (Mann Whitney U test $p < 0.01$), who disproportionately favor **concede/accept** questions. While further work is needed to fully explain these differences, we speculate that they are potentially reflective of strategic attempts by younger MPs to signal traits that could facilitate future promotion, such as partisan loyalty (Kam, 2009).

To discount the possibility of these effects being solely driven by a few very prolific young or old MPs, we also consider a setting where type propensities are macroaveraged over MPs. For each affiliation we compare the cohort of younger MPs who are newly voted in at the 1997 and 2010 elections, with older MPs who have been in office prior to the election.[15] We compute the type propensities of these two cohorts over the questions they asked during the *subsequent* parliamentary sitting, and replicate the tenure effects observed previously (Figure 2B). This suggests that these parliamentary career effects reflect behavioral changes at the level of individual MPs, whose incentives evolve over their tenure.

8 Conclusion and Future Work

In this work we introduced an unsupervised framework for structuring the space of questions according to their rhetorical role. We instantiated and validated our approach in the domain of parliamentary question periods, and revealed new interactions between questioning behavior and career trajectories.

We note that our methodology is not tied to a particular domain. It would be interesting to explore its potential in a variety of less structured domains where questions likewise play a crucial role. For example, examining how interviewers in high-profile media settings (e.g., Frost on Nixon) can use their questions to elicit substantive responses from influential people would aid us in the broader normative goal of holding elites to account, by gaining a better understanding of what and how to ask, and what (not) to accept as an answer.

From a technical standpoint, future work could also augment the representation of questions and answers presently used in our framework, beyond our heuristic of using root arcs without noun phrases. Richer linguistic representations, as well as more judicious ways of weighting different fragments and motifs, could enable us to capture a wider range of possible surface and rhetorical forms, especially in settings where phrasings are potentially less structured by institutional conventions. Additionally, as with most unsupervised methods, our approach is limited by the need to hand-select parameters such as the number of clusters, and manually interpret the typology's output. Having annotations of these corpora could better motivate the methodology and enable further evaluation and interpretation; we hope to encourage such annotation efforts by releasing the dataset.

Inevitably, drawing causal lessons from observational data is difficult. Moving forward, experimental tests of insights gathered through such explorations would enable us to establish causal effects of question-asking rhetoric, perhaps offering prescriptive insights into questioning strategies for objectives such as information-seeking (Dillman, 1978), request-making (Althoff et al., 2014; Mitra and Gilbert, 2014) and persuasion (Tan et al., 2016; Zhang et al., 2016; Wang et al., 2017).

Acknowledgements. The first author thanks John Bercow, the Speaker of the House, for suggesting she "calm [her]self down by taking up yoga" during the hectic deadline push (`https://youtu.be/AiAWdLAIj3c`). The authors thank the anonymous reviewers and Liye Fu for their comments and for their *helpful* questions. We are grateful to the organizers of the conference on New Directions in Text as Data for fostering the inter-disciplinary collaboration that led to this work, to Amber Boydstun and Philip Resnik for their insights on questions in the political domain, and to Stephen Bates, Peter Kerr and Christopher Byrne for sharing the labeled PMQ dataset. This research has been supported in part by a Discovery and Innovation Research Seed Award from the Office of the Vice Provost for Research at Cornell.

[15] This totals 272 new and 184 old government MPs, and 84 new and 179 old opposition MPs.

References

Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast algorithms for mining association rules. In *Proceedings of VLDB*.

Tim Althoff, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2014. How to ask for a favor: A case study on the success of altruistic requests. In *Proceedings of ICWSM*.

Malika Aubakirova and Mohit Bansal. 2016. Interpreting neural networks to improve politeness comprehension. In *Proceedings of EMNLP*.

Stephen R. Bates, Peter Kerr, Christopher Byrne, and Liam Stanley. 2014. Questions to the Prime Minister: A comparative study of PMQs from Thatcher to Cameron. *Parliamentary Affairs*.

Giacomo Benedetto and Simon Hix. 2007. The Rejected, the Ejected, and the Dejected: Explaining government rebels in the 2001-2005 British House of Commons. *Comparative Political Studies*.

Shohini Bhattasali, Jeremy Cytryn, Elana Feldman, and Joonsuk Park. 2015. Automatic identification of rhetorical questions. In *ACL*.

Amber E. Boydstun, Dallas Card, Justin H. Gross, Philip Resnik, and Noah A. Smith. 2014. Tracking the development of media frames within and across policy issues. In *Proceedings of the APSA*.

Peter Bull and Pam Wells. 2012. Adversarial discourse in Prime Ministers Questions. *Journal of Language and Social Psychology*.

Xin Cao, Gao Cong, Bin Cui, and Christian S Jensen. 2010. A generalized framework of exploring category information for question retrieval in community question answer archives. In *Proceedings of WWW*.

Dallas Card, Justin Gross, Amber Boydstun, and Noah Smith. 2016. Analyzing framing through the casts of characters in the news. In *Proceedings of EMNLP*.

Daniel Chester and Nona Bowring. 1962. *Questions in Parliament*. Clarendon Press, Oxford.

Philip Cowley. 2002. *The Rebels: How Blair Mislaid his Majority*. Politico's Publishing.

Philip Cowley. 2012. Arise, novice leader! The continuing rise of the career politician in Britain. *Politics*.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of ACL*.

Rajdip Dhillon, Bhagat Sonali, Carvey Hannah, and Shriberg Elizabeth. 2004. Meeting Recorder Project: Dialog Act Labeling Guide. Technical report.

Don A Dillman. 1978. *Mail and telephone surveys: The total design method*. Wiley New York.

Andrew Eggers and Arthur Spirling. 2014. Ministerial responsiveness in westminster systems: Institutional choices and House of Commons debate, 1832–1915. *American Journal of Political Science*.

Erving Goffman. 1976. Replies and responses. *Language in society*.

Sandra Gonzalez-Bailon, Andreas Kaltenbrunner, and Rafael E Banchs. 2010. The structure of political discussion networks: A model for the analysis of online deliberation. *Journal of Information Technology*.

Justin Grimmer, Solomon Messing, and Sean J Westwood. 2012. How words and money cultivate a personal vote: The effect of legislator credit claiming on constituent credit allocation. *American Political Science Review*.

Justin Grimmer and Brandon Stewart. 2013. Text as Data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*.

Poonam Gupta and Vishal Gupta. 2012. A survey of text question answering techniques. *International Journal of Computer Applications*.

Sanda M. Harabagiu. 2008. Questions and intentions. In *Advances in Open Domain Question Answering*.

Sanda M. Harabagiu, Dan I. Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Răzvan Bunescu, Corina R Gîrju, Vasile Rus, and Paul Morărescu. 2000. FALCON: Boosting knowledge for answer engines. In *Proceedings of TREC*.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé III. 2014a. A neural network for factoid question answering over paragraphs. In *Proceedings of EMNLP*.

Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014b. Political ideology detection using recursive neural networks. In *Proceedings of ACL*.

Jiwoon Jeon, W Bruce Croft, and Joon Ho Lee. 2005. Finding Semantically Similar Questions Based on Their Answers. In *Proceedings of SIGIR*.

Christopher Kam. 2009. *Party Discipline and Parliamentary Politics*. Cambridge University Press, Cambridge.

Greg P. Kearsley. 1976. Questions and question asking in verbal discourse: A cross-disciplinary review. *Journal of Psycholinguistic Research*.

Wendy G. Lehnert. 1978. *The process of question answering: A computer simulation of cognition*. Lawrence Erlbaum Associates.

Tom Louwerse. 2012. Mechanisms of issue congruence: The democratic party mandate. *West European Politics*.

Steven Lytinen and Noriko Tomuro. 2002. The use of question types to match questions in FAQFinder. In *AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*.

Tanushree Mitra and Eric Gilbert. 2014. The Language that Gets People to Give: Phrases that Predict Success on Kickstarter. In *Proceedings of CSCW*.

Burt L. Monroe, Michael P. Colaresi, and Kevin M. Quinn. 2008. Fightin' Words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*.

Vlad Niculae, Caroline Suen, Justine Zhang, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2015. QUOTUS: The structure of political media coverage as revealed by quoting patterns. In *Proceedings of WWW*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*.

Hanna F. Pitkin. 1967. *The Concept of Representation*. University of California Press, Oakland.

Sven-Oliver Proksch and Jonathan Slapin. 2011. Parliamentary questions and oversight in the European Union. *European Journal of Political Research*.

Suhas Ranganath, Xia Hu, Jiliang Tang, Suhang Wang, and Huan Liu. 2016. Identifying rhetorical questions in social media. In *ICWSM*.

Sujith Ravi, Bo Pang, Vibhor Rastogi, and Ravi Kumar. 2014. Great question! Question quality in community Q&A. In *Proceedings of ICWSM*.

Ludovic Rheault, Kaspar Beelen, Christopher Cochrane, and Graeme Hirst. 2016. Measuring emotion in parliamentary debates with automated textual analysis. *PloS One*.

Niharika Sachdeva and Ponnurangam Kumaraguru. 2017. Call for service: Characterizing and modeling police response to serviceable requests on Facebook. In *Proceedings of CSCW*.

Anna Shtok, Gideon Dror, Yoelle Maarek, and Idan Szpektor. 2012. Learning from the past: Answering new questions with past answers. In *Proceedings of WWW*.

Arthur Spirling and Iain McLean. 2007. UK OC OK? Interpreting optimal classification scores for the U.K. House of Commons. *Political Analysis*.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proceedings of WWW*.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get Out the Vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of EMNLP*.

Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. 2011. How do programmers ask and answer questions on the web? In *Proceedings of ICSE*.

Lu Wang, Nick Beauchamp, Sarah Shugars, and Kechen Qin. 2017. Winning on the merits: The joint effects of content and style on debate outcomes. *TACL*.

Paul Webb and David M Farrell. 1999. Party members and ideological change. *Critical Elections: British Parties and Voters in Long-Term Perspective*.

Justine Zhang, Ravi Kumar, Sujith Ravi, and Cristian Danescu-Niculescu-Mizil. 2016. Conversational flow in Oxford-style debates. In *Proceedings of NAACL*.

Wei Emma Zhang, Quan Z. Sheng, Jey Han Lau, and Ermyas Abebe. 2017. Detecting duplicate posts in programming QA communities via latent semantics and association rules. In *Proceedings of WWW*.

Appendix

A.1 Further examples of question types

Tables 2 and 3 provide further examples of representative questios and motifs from each of the eight question types we induce on our dataset of parliamentary question periods. Additionally, we include extended interpretations of each of these types, provided by the second author, a political scientist with domain expertise in the UK parliamentary setting.

A.2 Details about data filtering decisions

Here we provide further details about how we selected the subset of 50,152 questions which was used in the analyses described in Sections 6 & 7. First, we restrict our analysis to the questions which consist of only one question, as opposed to a series of questions (as delimited by multiple sentences ending in question marks; constituting 52% of the data). We omit these multi-question utterances in order to ensure that we don't confound effects arising from different co-occurring question types. Next, we only include questions for which information about the asker and answerers' party affiliations and ministership positions are available (such data is provided consistently from the Blair government onwards). Finally, we omit questions asked by opposition members who are specifically appointed by their party to *shadow* a government minister, and are hence obliged by their appointment to ask more critical questions. Our choice to omit such questions eliminates the possibility that our observed differences in question preference are driven by official appointment.

A.3 Details about the labeled PMQ dataset

Here we provide further details about the size of the labeled Questions to the Prime Minister (PMQ) dataset from Bates et al. (2014) used in the quantitative validation of our typology (Section 6). The dataset contains 931 standard, 186 helpful and 139 unanswerable questions. Additionally, 445 answers to questions are labeled as *answered*, while 305 are labeled as *not answered*; the rest are labeled as *deferred answer*, meaning the Prime Minister did not have the knowledge or capability to provide an answer.

We restrict all analysis done using the unanswered vs. answered labels to *standard* questions, i.e., questions for which the PM has an opportunity to provide a legitimate answer.

The *helpful vs. standard* classification task contains 264 train and 108 test examples; the *unanswerable vs. standard* classification task contains 190 train and 86 test examples; the *unanswered vs. answered* classification task contains 166 train and 84 test examples.

A.4 Example motif subgraph

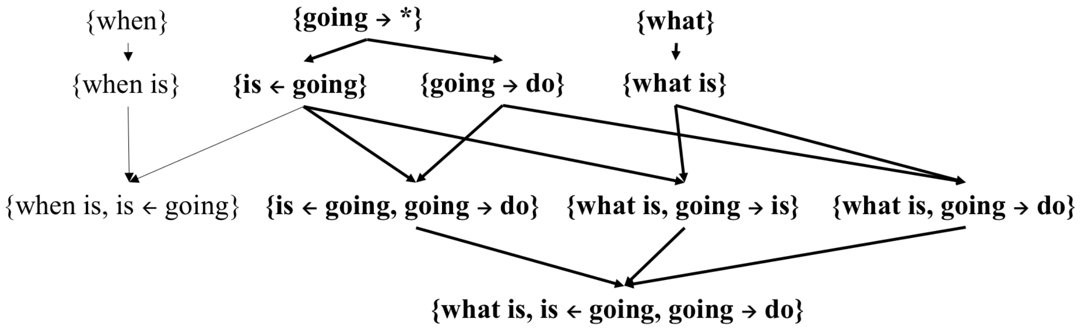
Figure 3 illustrates a section of the motif DAG $\mathcal{M}$ and highlights (in bold) the subgraph $\mathcal{M}_q$ corresponding to the phrasing of the question in example (1). Nodes that are higher in the graph serve as general representations, and capture similarities between broad sets of phrasings: e.g., {is←going going→do, is←going, what is} groups together example (1) with a question like "When is he going to get a grip on [the scandal]". By projecting such motifs into our latent question-answer space (Section 5) we capture characteristics shared between these phrasings and allow for generalizability. Nodes which are lower in the graph constitute more specific representations, disambiguating between phrasings. In particular, sink motifs serve as the most specific representation of a question, delineating the region of the latent space (and thus the question type) that best captures its phrasing. For instance, this additional specificity allows us to draw contrasts between "**What is** the minister **going** to **do** [about the policy]?" (sink motif: {what is, is←going, going→do}) and the more aggressive "**When is** he **going** to get a grip on [the scandal]?" (sink motif: {when is, is←going}).

Type	Interpretation and examples
0: Issue update	Requests for information or updates about a current event, issue, or policy. Typically the policy refers to a genuinely ‘national’ concern, rather than a partisan issue for which the major parties may have differing views. Q: **Will** the Minister also **update** the House on whether any decisions have been made on the post-2014 UK contribution to Afghanistan? Q: **What** more can the Government **do** to **ensure** that each pupil has a single point of contact [for mental health issues] throughout their education? Q: **What** further steps **can** we **take** to resolve [the] terrible situation [in Syria]? Question motifs: {what,are←taking}, {will←update}, {what←do, do→ensure}, {what←take, can←take}
1: Shared concerns	Straightforward factual question with no strong ideological underpinnings; answers are typically vague and involve explaining that the government takes it seriously, will continue to do so and will consult with the relevant stakeholders. Q: **May** I **urge** the Minister to concentrate on tough penalties for people who get involved in alcohol-induced antisocial behaviour? Q: **Will** the Secretary of State **look carefully at** reports that houses built to house the soldiers will block off the rising sun at the summer equinox [at] the Stonehenge? Q: **Will** [my hon. Friend] **raise** [the matter of] product placement **with** people in the film industry [when he meets them]? Question motifs: {will, take→*}, {may←urge}, {will←look, look→at, look→carefully}, {raise→will, raise→with}
2: Narrow factual	Narrow queries about relatively minor policy issues which are either extremely local in nature (perhaps referring to implications for a given constituency) or limited in scope (constituting a small issue within a much broader context of policy). Questions are precise if not especially penetrating. Answers typically explain the ministry response in narrow, non-ideological terms. Q: **What** funding will be **given to** the new postgraduate institute at the Edinburgh College of Dentistry? Q: **Will** the Minister **make** a statement **on** where we are with the website for the [National Health Service appraisals toolkit for general practitioners]? Q: **What will happen** to the French if they are found guilty of infringing the rules of the Commission? Question motifs: {what←made}, {what←happen, will←happen}, {what←given, given→to}, {will←make, make→on}
3: Prompt for comment	Requests for comments, or information, especially on a meeting that has taken place between a minister and constituents, colleagues, or opposite numbers, the contents of which would not normally be immediately accessible to MPs. The asker seeks that the minister clarifies policy where none might presently exist. Q: **Will** the Secretary of State **tell** us which Minister has been appointed to be responsible for green economic growth? Q: **Can** [the Secretary of State] **confirm** whether train platform capacity will **be** a part of the discussions between the city council and Network Rail? Q: **What would** the Prime Minister **say** to a borough council which is considering rejecting Government funding and instead [taxing] my constituents more? Question motifs: {what←say,say→to}, {will←tell}, {can←confirm, confirm→be}, {what←say, would←say}

Table 2: Interpretation for the first four types, with examples of representative questions and motifs.

Type	Interpretation and examples
4: Agreement	Airing a laudatory remark about a policy that the minister and MP clearly already agree on. Often these questions effectively serve as attempts to curry favor with the minister and bolster their (mutual) party. Q: **Is** it not **important** that the Department continues its excellent work [in] building flood defeneces? Q: **Does** [the Secretary of State] **agree with** me that part of protecting Britain's national interests is that Britain should develop relationships with emerging economies? Q: **Does** the Minister **agree** that UK taxpayers **need** to be considered at every single step of the way when it comes to our aid spending? Question motifs: {does←agree, agree→is}, {is→important}, {does←agree, agree→with}, {does←agree, agree→need}
5: Self-promotion	Here, "awareness" is entirely rhetorical: either the minister is aware and agrees or the minister is not aware and will investigate. These questions allow the question asker to be seen to be bringing local concerns to broader attention, but in a way that is more assertive than in type 2 (**narrow factual**). Q: **Has** the right hon. Gentleman **considered** compulsory postal voting , and moving polling day from a Thursday to the weekend? Q: In considering the role of local tribunals , **will** my hon Friend **take** account of the recommendation **in** the Oglesby report? Q: **Will** the Minister **reconsider** his reply to my [colleague], [in light of] the situation [on] school leavers applying for technical positions? Question motifs: {is→aware}, {considered←has}, {will←take, take→in}, {will←reconsider}
6: Concede, accept	Aggressive demand for minister to concede to, or accept, a fault. The premise of such questions is that the minister has been incompetent, or that the government has the wrong policy; these questions do not constitute a genuine attempt to obtain information. Q: **Is** it **not** now completely **true** that the Labour Government are out of touch with gut British instincts? Q: **Will** [the Secretary] **acknowledge** the importance of not completely abandoning the research on sustainable biofuels? Q: **Will** [the Deputy Prime Minister] now **concede** to the House that the Royal Mail was sold off too cheaply? Question motifs: {will←accept}, {is→not, is→true}, {will←acknowledge}, {will←concede}
7: Condemnatory	Similar to type 6 (**concede, accept**) but more aggressive and hectoring in tone, asking the minister to explain themselves and be contrite on the basis of very broad ideological premises that are difficult to answer without 'self-incrimination'. These questions often rely on rhetorical grandstanding and lacks any subtlety or policy detail on which minister can comment precisely. Q: When members of the armed forces are facing a pay freeze, **how can** the Secretary **justify** bonuses to senior offices in the civil service? Q: **Why does** the right hon Gentleman not have an industrial strategy to build that recovery? Q: **Will** the Government now **apologise** for their complacent decision to scrap the future jobs fund, [given that] long-term youth unemployment is rising? Question motifs: {can←explain}, {how←justify, can←justify}, {why does}, {will←apologise}

Table 3: Interpretation for the last four types, with examples of representative questions and motifs.



(1) What is the minister going to do about … ?

Figure 3: A section of the motif DAG $\mathcal{M}$ and the subgraph $\mathcal{M}_q$ (in bold) representing the phrasing of the question in example (1). For clarity, redundant and irrelevant nodes and edges are not shown.

Detecting Perspectives in Political Debates

David Vilares
Universidade da Coruña
Departamento de Computación
Campus de Elviña s/n, 15071
A Coruña, Spain
david.vilares@udc.es

Yulan He
School of Engineering and Applied Science
Aston University
United Kingdom
y.he@cantab.net

Abstract

We explore how to detect people's perspectives that occupy a certain proposition. We propose a Bayesian modelling approach where topics (or propositions) and their associated perspectives (or viewpoints) are modeled as latent variables. Words associated with topics or perspectives follow different generative routes. Based on the extracted perspectives, we can extract the top associated sentences from text to generate a succinct summary which allows a quick glimpse of the main viewpoints in a document. The model is evaluated on debates from the House of Commons of the UK Parliament, revealing perspectives from the debates without the use of labelled data and obtaining better results than previous related solutions under a variety of evaluations.

1 Introduction

Stance classification is binary classification to detect whether people is supporting or against a topic. Existing approaches largely rely on labelled data collected under specific topics for learning supervised classifiers for stance classification (Mohammad et al., 2016a). At most time, apart from detecting one's stance, we are interested in finding out the arguments behind the person's position. Perspectives, that state people's ideas or the facts known to one, can be *contrastive*, i.e. to be in favour of or against something (e.g. Brexit vs Bremain), or *non-contrastive*, i.e. independent discussions that share a common topic (e.g. unemployment and migration in the context of economy).

Recent years have seen increasing interests in argumentation mining which involves the automatic identification of argumentative structures, e.g., the claims and premises, and detection of argumentative relations between claims and premises or evidences. However, learning models for argumentation mining often require text labelled with components within argumentative structures and detailed indication of argumentative relations among them. Such labelled data is expensive to obtain in practice and it is also difficult to port models trained on one domain to another.

We are particularly interested in detecting different perspectives in political debates. Essentially, we would like to achieve somewhere in between stance classification and argumentation mining. Given a text document, we want to identify a speaker's key arguments, without the use of any labelled data. For example, in debates about '*Education*', we want to automatically extract sentences summarising the key perspectives and their arguments, e.g. 'our education system needs to promote excellence in *stem subjects*', 'teenagers need to be taught with *sexual and health education*' or '*grammar schools* promote inequality'. Similarly, if '*Brexit*' is being discussed in terms of leaving or remaining, we want to cluster arguments into those two viewpoints.

To do this, we introduce a Latent Argument Model (LAM) which assumes that words can be separated as topic words and argument words and follow different generative routes. While topic words only involve a sampling of topics, argument words involve a joint sampling of both topics and arguments. The model does not rely on labelled data as opposed to most existing approaches to stance classification or argument recognition. It is also different from cross-perspective topic models which assume the perspectives are observed (Fang et al., 2012). Quantitative and qualitative evaluations on debates from the House of Commons of United Kingdom show the utility of the approach and provide a comparison against related models.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1573–1582
Copenhagen, Denmark, September 7–11, 2017.

2 Related work

Our research is related to stance classification, argument recognition and topic modelling for sentiment/perspective detection.

2.1 Stance Classification

Stance detection aims to automatically detect from text whether the author is in favour of, against, or neutral towards a target. As previously reported in (Mohammad et al., 2016b), a person may express the same stance towards a target by using negative or positive language. Hence, stance detection is different from sentiment classification and sentiment features alone are not sufficient for stance detection. With the introduction of the shared task of stance detection in tweets in SemEval 2016 (Mohammad et al., 2016a), there have been increasing interests of developing various approaches for stance detection. But most of them focused on building supervised classifiers from labelled data. The best performing system (Zarrella and Marsh, 2016) made use of large unlabelled data by first learning sentence representations via a hashtag prediction auxiliary task and then fine tuning these sentence representations for stance detection on several hundred labelled examples. Nevertheless, labelled data are expensive to obtain and there is a lack of portability of classifiers trained on one domain to move to another domain.

2.2 Argument Recognition

Closely related to stance detection is argument recognition which can be considered as a more fine-grained task that it aims to identify text segments that contain premises that are against or in support of a claim. Cabrio and Villata (2012) combined textual entailment with argumentation theory to automatically extract the arguments from online debates. Boltuzic and Šnajder (2014) trained supervised classifiers for argument extraction from their manually annotated corpus by collecting comments from online discussions about two specific topics. Sardianos et al. (2015) proposed a supervised approach based on Conditional Random Fields for argument extraction from Greek news. Nguyen and Litman (2015) run an LDA model and post-processed the output, computing argument and domain weights for each of the topics, which were then used to extract argument and domain words. Their model outperformed traditional n-grams and lexical/syntactic rules on a collection of persuasive essays. Lippi and Torroni (2016a) hypothesized that vocal features of speech can improve argument mining and proposed to train supervised classifiers by combining features from both text and speech for claim detection from annotated political debates. Apart from claim/evidence detection, there has also been work focusing on identification of argument discourse structures such as the prediction of relations among arguments or argument components (Stab and Gurevych, 2014; Peldszus and Stede, 2015). A more recent survey of various machine learning approaches used for argumentation mining can be found in (Lippi and Torroni, 2016b). All these approaches have been largely domain-specific and rely on a small set of labelled data for supervised model learning.

2.3 Topic Modeling for Sentiment/Perspective Detection

Topic models can be modified to detect sentiments or perspectives. Lin and He (2009) introduced a joint sentiment topic (JST) model, which simultaneously extracts topics and topic-associated sentiments from text. Trabelsi and Zaıane (2014) proposed a joint topic viewpoint (JTV) model for the detection of latent viewpoints under a certain topic. This is essentially equivalent to the reparameterized version of the JST model called REVERSE-JST (Lin et al., 2012) in which sentiment label (or viewpoint) generation is dependent on topics, as opposed to JST where topic generation is conditioned on sentiment labels.

Fang et al. (2012) proposed a Cross-Perspective Topic Model (CPT) in which the generative processes for topic words (nouns) and opinion words (adjectives, adverbs and verbs) are different, as the opinion words are sampled independently from the topic. Also, CPT assumed perspectives are observed, which implies texts need to be annotated with the viewpoint they belong to. Awadallah et al. (2012) detected politically controversial topics by creating an opinion-base of opinion holders and their views. Das and Lavoie (2014) observed the editions and interactions of a user in Wikipedia pages to infer topics and points of view at the same time. Qiu et al. (2015) proposed a regression-based latent factor model which jointly models user arguments, interactions, and attributes for user stance prediction in online debates.

3 Latent Argument Model (LAM)

We assume that in a political debate, the speaker first decides on which topic she wants to comment on (e.g. *Education*). She then takes a stance (e.g. *remark the importance about stem subjects*) and elaborates her stance with arguments. It is worth noting that we do not consider the temporal dimension of documents here, i.e., our model is fed with a collection of unlabeled documents without temporal order.

We use a switch variable x to denote whether a word is a *background word* (shared across multiple topics), a *topic word* (relating to a certain topic) or an *argument word* (expressing arguments under a specific topic). Depending on the type of word, we follow a different generative process. For each word in a document, if it is a background word, we simply sample it from the background word distribution ϕ^b; if it is a topic word, we first sample a topic z from the document-specific topic distribution θ_d and then sample the word from the topic-word multinomial distribution ψ_z shared across all documents; if it is an argument word, we need to first jointly sample the topic-argument pair, (z, a), where z comes from the existing topics already sampled for the topic words in the document and a is sampled from the topic-specific argument distribution ω_z, and finally the word is sampled from the multinomial word distribution for the topic-specific argument $\psi_{z,a}$. The argument indicator here is a latent categorical variable. It can take a binary value to denote pro/con or positive/negative towards a certain topic. More generally, it could also take a value from multiple stance or perspective categories. We thus propose a Latent Argument Model (LAM) shown in Figure 1. Formally, the generative process is as follows:

- Draw a distribution over the word switch variable, $\phi \sim$ Dirichlet(γ), and background word distribution, $\psi^b \sim$ Dirichlet(β^b).
- For each topic $z \in \{1...T\}$, draw a multinomial topic-word distribution $\psi_z \sim$ Dirichlet(β^z).
  - For each argument $a \in \{1...A\}$ draw a multinomial topic-argument distribution $\omega_z \sim$ Dirichlet(δ) as well as a multinomial topic-argument-word distribution $\psi^v_{z,a} \sim$ Dirichlet(β^a).
- For each document $d \in \{1...D\}$:
  - Draw a multinomial topic distribution, $\theta_d \sim$ Dirichlet(α).
  - For each word $n \in \{1, .., N_d\}$ in d:
    - $*$ Choose $x_{d,n} \sim$ Multinomial(ϕ).
    - $*$ If $x_{d,n} = 0$, draw a background word $w_{d,n} \sim \psi^b$;
    - $*$ If $x_{d,n} = 1$, draw a topic $z \sim$ Multinomial(θ_d) and a word $w_{d,n} \sim$ Multinomial(ψ_z);
    - $*$ If $x_{d,n} = 2$, draw a topic $z \sim$ Multinomial(θ_d), an argument $a \sim$ Multinomial(ω_z) and a word $w_{d,n} \sim$ Multinomial$(\psi^a_{z,a})$.

Figure 1 shows its plate representation.

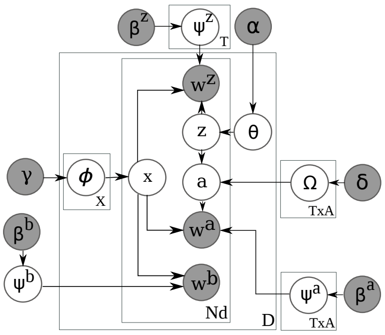


Figure 1: The plate notation for the LAM model. Shadowed elements represent the observed variables (words and prior distributions).

3.1 Inference and Parameter Estimation

We use Collapsed Gibbs Sampling (Casella and George, 1992) to infer the model parameters and the latent assignments of topics and arguments, given the observed data. Gibbs sampling is a Markov chain Monte Carlo method to iterative estimate latent parameters. In each iteration, a new sample of the hidden parameters is made based on the distribution of the previous epoch. Letting the index $t = (d, n)$ denote the nth word in document d and the subscript $-t$ denote a quantity that excludes data from the nth word position in document d, $\Lambda = \{\alpha, \beta^b, \beta^z, \beta^a, \gamma, \delta\}$, the conditional posterior for x_t is:

$$P(x_t = r|\boldsymbol{x}_{-t}, \boldsymbol{z}, \boldsymbol{a}, \boldsymbol{w}, \Lambda) \propto \frac{\{N^r_d\}_{-t} + \gamma}{\{N_d\}_{-t} + 3\gamma} \cdot \frac{\{N^r_{w_t}\}_{-t} + \beta^r}{\sum_{w'}\{N^r_{w'}\}_{-t} + W\beta^r}, \quad (1)$$

where r denotes different word types, either background word, topic word or argument word. N^r_d denotes the number of words in document d assigned to the word type r, N_d is the total number of words in the document d, $N^r_{w_t}$ is the number of times word w_t is sampled from the distribution for the word type r, W is the vocabulary size.

For topic words, the conditional posterior for z_t is:

$$P(z_t = k | \boldsymbol{z}_{-t}, \boldsymbol{w}, \Lambda) \propto \frac{N_{d,k}^{-t} + \alpha_k}{N_d^{-t} + \sum_k \alpha_k} \cdot \frac{N_{k,w_t}^{-t} + \beta^z}{N_k^{-t} + W\beta^z}, \quad (2)$$

where $N_{d,k}$ is the number of times topic k was assigned to some word tokens in document d, N_d is the total number of words in document d, N_{k,w_t} is the number of times word w_t appeared in topic k.

For argument words, the conditional posterior for z_t and a_t is:

$$P(z_t = k, a_t = j | \boldsymbol{z}_{-t}, \boldsymbol{a}_{-t}, \boldsymbol{w}, \Lambda) \propto \frac{N_{d,k}^{-t} + \alpha_k}{N_d^{-t} + \sum_k \alpha_k} \cdot \frac{N_{d,k,j}^{-t} + \delta_{k,j}}{N_{d,k}^{-t} + \sum_j \delta_{k,j}} \cdot \frac{N_{k,j,w_t}^{-t} + \beta^v}{N_{k,j}^{-t} + W\beta^v}, \quad (3)$$

where $N_{d,k,j}$ is the number of times a word from document d has been associated with topic k and argument j, N_{k,j,w_t} is the number of times word w_t appeared in topic k and with argument j, and $N_{k,j}$ is the number of words assigned to topic k and argument j.

Once the assignments for all the latent variables are known, we can easily estimate the model parameters $\{\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\rho}, \boldsymbol{\psi^b}, \boldsymbol{\psi^z}, \boldsymbol{\psi^a}, \boldsymbol{\omega}\}$. We set the symmetric prior $\gamma = 0.3$, $\epsilon = 0.01$, $\beta^b = \beta^z = 0.01$, $\delta = (0.05 \times L)/A$, where L is the average document length, A the is total number of arguments, and the value of 0.05 on average allocates 5% of probability mass for mixing. The asymmetric prior $\boldsymbol{\alpha}$ is learned directly from data using maximum-likelihood estimation (Minka, 2003) and updated every 40 iterations during the Gibbs sampling procedure. In this paper we only consider two possible stances, hence, $A = 2$. But the model can be easily extended to accommodate more than two stances or perspectives. We set the asymmetric prior β^a for the topic-argument-word distribution based on a subjectivity lexicon in hoping that contrastive perspectives can be identified based on the use of positive and negative words. We run Gibbs sampler for 1 000 iterations and stop the iteration once the log-likelihood of the training data converges.

3.2 Separating Topic and Perspective Words

Using the word type switch variable $\mathbf{x}$, we could separate topic and argument words in LAM based solely on the statistics gathered from data. We also explore another two methods to separate topic and argument words based on Part-of-Speech (POS) tags and with the incorporation of a subjectivity lexicon. For the first variant, we adopt the similary strategy as in (Fang et al., 2012) that nouns (NOUN) are topic words; adjectives (ADJ), adverbs (ADV) and verbs (VERB) are argument words; words with other POS tags are background words. Essentially, $\mathbf{x}$ is observed. We call this model LAM_POS.

For the second variant, instead of assuming $\mathbf{x}$ is observed, we incorporate the POS tags as prior information to modify the Dirichlet prior γ for the word type switch variable at the initialization step. In addition, we also consider a subjective lexicon[1], L, that if a word can be found in the lexicon, then it is very likely the word is used to convey an opinion or argument, although there is still a small probability that word could be either background or topic word. Assuming an asymmetric Dirichlet prior for $\mathbf{x}$ is parametrized by $\gamma^\intercal = [\gamma^b, \gamma^z, \gamma^a]$ for background, topic and argument words, it is modified by a transformation matrix λ, $\gamma^{\text{new}} = \lambda \times \gamma^\intercal$, where λ is defined by:

- If word $w \in L \wedge \text{POSTAG}(w) \neq \text{NOUN}$ then $\lambda^\intercal = [0.05, 0.05, 0.9]$
- else if $\text{POSTAG}(w) = \text{NOUN}$ then $\lambda^\intercal = [0.05, 0.9, 0.05]$
- else if $\text{POSTAG}(w) \in \{\text{ADJ,ADV,VERB}\}$ then $\lambda^\intercal = [0.05, 0.05, 0.9]$
- else $\lambda^\intercal = [0.9, 0.05, 0.05]$

The conditional probability for the switch variable x is modified by simultaneously considering the POS tag g for the word at position t:

$$P(x_t = r, y_t = g | \boldsymbol{x}_{-t}, \boldsymbol{z}, \boldsymbol{a}, \boldsymbol{w}, \Lambda) \propto \frac{\{N_d^r\}_{-t} + \gamma}{\{N_d\}_{-t} + 3\gamma} \cdot \frac{\{N_{w_t}^r\}_{-t} + \beta^r}{\sum_{w'} \{N_{w'}^r\}_{-t} + W\beta^r} \cdot \frac{\{N_g^r\} + \epsilon_g^r}{\{N_g\} + \sum_{g'} \epsilon_{g'}^r}, \quad (4)$$

where an additional term is added to the RHS of Equation 1. Here, N_g^r denotes the number of words with POS tag g assigned to the word type r, N_g is the total number of words assigned to the POS tag g, ϵ_g^r is the Dirichlet prior for the POS tag-word type distribution.

[1] In this work, we use the subjectivity lexicon presented at (Wiebe et al., 2005).

We call the second variant LAM_LEX. As both the POS tag information and the subjectivity lexicon are only incorporated in the initialisation step, LAM_LEX sits in-between LAM and LAM_POS that it performs soft clustering of topic words and argument words. That is, during the initialisation, nouns are more likely to be topic words, but there is still a small probability that they could be either argument or background words; and similarly for words tagged as adjectives, adverts and verbs.

4 House of Common Debates (HCD)

Debates from the UK parliament are archived and available for consulting.[2] A custom web-crawler was developed to obtain the records of every day that The House of Commons was in session between 2009 and 2016. Due to inconsistencies in the data format and volume of data, much of the analysis focuses on the recordings for the parliamentary year 2014-2015. The general structure of a single day of recording is as follows: a question will be put to the house (generally a Bill) and Members of Parliament (MPs) will discuss various aspects regarding the Bill or show stances about it. Each speech made by an MP is considered to be a single document. Multiple Bills will be discussed each day. The current item being discussed is clearly marked in the source data format, so linking documents to the current bill and MP is trivial. Each speech is labelled with a major (e.g. *education*) and a minor topic (e.g. *grammar schools*) and help us create a dataset with the desired needs.

The length that The House will be in session varies and the number of bills discussed also varies. In this paper, we considered debates occurred during March of 2015[3] and contains 1 992 speeches belonging to diverse domains: *justice*, *education*, *energy and climate change*, *treasury*, *transport*, *armed forces*, *foreign and commonwealth office*, *environment*, *transport*, *royal assent*, *work and pensions*, *northern Ireland*. This House of Commons Debates (HCD) dataset is made available for the research community.[4]

We followed a standard methodology to clean the texts: stopwords were removed, lemmatization was applied, and a naive negation treatment was considered for the particle 'not', by creating bigrams for words occurs in the subjectivity lexicon (e.g., 'not good' becomes 'not_good'). As topic models suffer from lack of robustness if large outliers are present, we also removed very frequent (above 99%) and rare words (below percentile 65%), assuming that word occurrences of the collection follow a Zip's law distribution.[5] Similar strategy was carried out for texts, in order to just consider texts of similar length. The preprocessed HCD contains a total of 1 598 speeches.

5 Experiments

This section evaluates LAM and its variants qualitatively and quantitatively (averaged over 5 runs).

The models for comparison are listed below:

- **LDA**. Latent Dirichlet Allocation (Blei et al., 2003).
- **CPT**. The Cross-perspective Topic Model (Fang et al., 2012) assumes perspectives are observed. To be able to run this model on the political speeches, we implemented a version that can manage latent perspectives and separately sample topics and viewpoints.
- **JTV**. Joint Topic-Viewpoint Model (Trabelsi and Zaıane, 2014) is essentially the reparameterized version of the Joint Sentiment-Topic (JST) model (Lin and He, 2009) called REVERSE-JST (Lin et al., 2012) in which sentiment label (or viewpoint) generation is dependent on topics. We implemented JTV as the reversed JST model.[6]
- **LAM**. Latent Argument Model from §3.
- **LAM_POS**. LAM with topic, argument or background words separated by POS tags.
- **LAM_LEX**. Both POS tags and a subjective lexicon are used to initialise the Dirichlet prior γ for the word type switch variable as described in §3.2.

5.1 Experimental Results

Results are evaluated in terms of both topic coherence and the quality of the extracted perspectives.

5.1.1 Topic Coherence

The CV metric[7] is used to measure the coherence of the topics generated by the models as it has been shown to give the results closest to human evaluation compared to other topic coherence metrics (Röder et al., 2015). In brief, given a set of words,

[2] https://hansard.parliament.uk

[3] Period of time what selected on a basis of existence of a large number of major topics.

[4] https://github.com/aghie/lam/blob/master/hcd.tsv

[5] Percentiles were selected on an empirical basis.

[6] We were not able to find a publicly available code of the JTV implementation.

[7] https://github.com/AKSW/Palmetto/

it gives an intuition of how likely those words co-occur compared to expected by chance.

Figure 2 plots the CV results[8] versus the number of topics on HCD for various models. For each topic z, we extract the top ten most representative words ranked by their respective normalised discriminative score defined by $\mathrm{DS}(w,z) = P(w|z)/[\max_{z' \neq z} P(w|z')]$. We chose this approach instead of simple $P(w|z)$ as it was observed to turn into higher quality topics. It is clear that LAM_LEX models outperform baselines and that all variants are learning well the topics from the data, showing that the three different mechanisms for the switch variable are effective to generate coherent topics. Also, our models work robustly under different number of topics. Moreover, LAM_LEX achieve better coherence scores than the original LAM and LAM_POS. This shows that it is more effective to use POS tags and a subjectivity lexicon to initialise the Dirichlet prior for the word type switch variable rather than simply relying on POS tags or subjectivity lexica to give hard discrimination between topic and argument words.

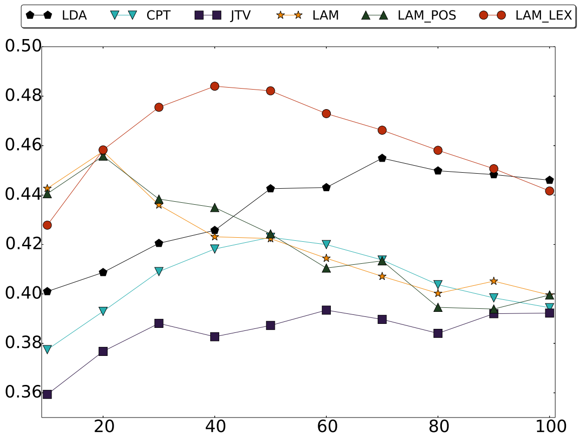


Figure 2: CV coherence vs the number of topics for different modeling approaches.

We also used the gold-standard major topic label assigned by Hansard to each speech to carry out an additional quantitative evaluation. For each topic z, we extract the top ten most representative sentences ranked by their respective normalised discriminative score defined by $\mathrm{DS}(s,z) = \sum_{w \in \boldsymbol{s}} \mathrm{DS}(w,z)/\mathrm{Length}(\boldsymbol{s})$.

If a particular model is clustering robustly, the top sentences it extracts should belong to speeches that discuss the same topic and share the same major and/or minor topic labels in the HCD corpus. Table 2 shows for the studied models the percentage of sentences where x out of top 10 topic sentences belong to the same major topic. The results reinforce the superior performance of the LAM_LEX approach in comparison with other models.

It is worth remarking that in cases where LAM_LEX cluster together sentences labelled with different major topics, some clustering results are actually quite sensible. Table 1 illustrates it with a representative case. These sentences were extracted from a cluster about *farmers* in which 9 out of 10 top topic sentences have "*environment, food and rural affairs*" as the gold major topic. The only discording sentence, belonging to *treasury* (major topic) and *infrastructure investment* (minor topic), is however closely related to *farmers* too and it makes sense to put it into the same cluster.

5.1.2 Perspectiven Summarisation

In this section we evaluate the quality of the relation of the arguments with respect to their topics.

In terms of a quantitative evaluation, we are interested in knowing how strongly the perspectives are related to their topic: it might be the case that the separate CV coherence for the topic and viewpoints is high, but there is no actual relation between them, which would be an undesirable behaviour. To determine whether this is happening or not in the studied models, for each perspective we compute a mixed topic-perspective CV value, by extracting the top 5 perspective words, concatenating them with the top 5 words of the corresponding topic and running Palmetto as in the previous section.[9] We then average the computed mixed topic-perspective CV values by $T \times A$. Following this methodology, a high average CV value means that the perspective words are likely to occur when discussing about that particular topic, and therefore a test of whether the model is learning perspectives that have to do with it. Figure 3 compares topic-perspective models evaluated following this methodology, showing that LAM_LEX gives the best overall coherence.

For a better understanding of what perspectives LAM_LEX is learning, we extract the top perspective sentences for a given topic based on normalised discriminative score of each sen-

[8] The CV results were calculated based on the top 10 words from each topic.

[9] Palmetto does not accept more than 10 words.

Sentence (extracted from a longer speech)	**Major topic**	**Minor topic**
I would add that HMRC can provide extra flexibility where there are particular impacts on particular farmers or other businesses	Treasury	Infrastructure Investment
I think milk prices will improve, but the banks need to support farmers in the meantime	Environment food and rural affairs	Topical questions

Table 1: Example sentences, belonging to speeches that were assigned in Hansard different major topics labels, were clustered together by LAM (and it is sensible to do so as they are both about "farmers").

Model	**#Topics**	**≥5**	**≥6**	**≥7**	**≥8**	**≥9**	**=10**
LDA	10	**0.720**	0.600	0.459	0.320	0.160	0.120
	20	**0.810**	**0.700**	0.570	0.430	0.310	0.180
	30	0.779	0.653	0.580	0.479	0.347	**0.233**
	40	0.620	0.550	0.475	0.360	0.290	0.145
	50	0.732	0.612	0.496	0.388	0.304	0.204
	100	0.654	0.486	0.374	0.306	0.216	0.139
CPT	10	0.580	0.440	0.320	0.260	0.220	0.140
	20	0.530	0.470	0.410	0.340	0.250	0.160
	30	0.473	0.394	0.313	0.273	0.193	0.147
	40	0.420	0.385	0.330	0.250	0.200	0.145
	50	0.464	0.340	0.292	0.220	0.156	0.104
	100	0.435	0.342	0.258	0.190	0.148	0.082
JTV	10	0.620	0.440	0.320	0.199	0.120	0.080
	20	0.634	0.486	0.377	0.303	0.229	0.110
	30	0.753	0.559	0.453	0.319	0.227	0.087
	40	0.705	0.580	0.460	0.370	0.250	0.145
	50	0.628	0.468	0.368	0.290	0.220	0.152
	100	0.636	0.508	0.364	0.274	0.184	0.126
LAM_LEX	10	**0.720**	**0.640**	**0.480**	**0.440**	**0.420**	**0.240**
	20	0.790	0.690	**0.610**	**0.520**	**0.340**	**0.220**
	30	**0.900**	**0.779**	**0.693**	**0.580**	**0.453**	0.213
	40	**0.850**	**0.770**	**0.650**	**0.550**	**0.444**	**0.260**
	50	**0.788**	**0.704**	**0.620**	**0.520**	**0.404**	**0.228**
	100	**0.656**	**0.544**	**0.452**	**0.348**	**0.278**	**0.186**

Table 2: Ratio of topics where x or more than x out of top 10 topic sentences ($\geq x$) belong to the same major topic.

tence[10], similar to what have been done in selecting the top topic sentences. In specific, we first define the discriminative score of word w under topic z and argument a by: $\mathrm{DS}(w,a,z) = \frac{P(w|z,a)}{\max_{z' \neq z, a' \neq a} P(w|z',a')}$. Then the sentence-level discriminative score is calculated based on the aggregated discriminative scores over all the words normalised by the sentence length: $\mathrm{DS}(s,z,a) = \sum_{w \in \boldsymbol{s}} \mathrm{DS}(w,a,z)/\mathrm{Length}(\boldsymbol{s})$. In order to have better correspondence between topics and their respective arguments, we perform two-stage selection: first ranking sentences based on topic-level discriminative scores DS(s,z), and then further ranking sentences based on topic-argument-level discriminative scores $\mathrm{DS}(s,z,a)$.

We can use these extracted top representative sentences together with the gold major topics from HCD to measure if perspectives are connected to their topic. We define the label-based accuracy (LA) as follows: let p_i be the gold major topics associated to the top 10 perspective sentences of a perspective i and let t be the gold major topics corresponding to the top 10 topic sentences; LA(t,p_i) = $\frac{|t \cap p_i|}{|t \cup p_i|}$ measures how many gold major topic labels are shared between topic and perspective sentences. LA also penalises the major topics that are not in common. Table 3 shows for different number of topics the averaged LA measure across all perspectives for three models. It can be observed that LAM_LEX obtains the best performance, followed by CPT.

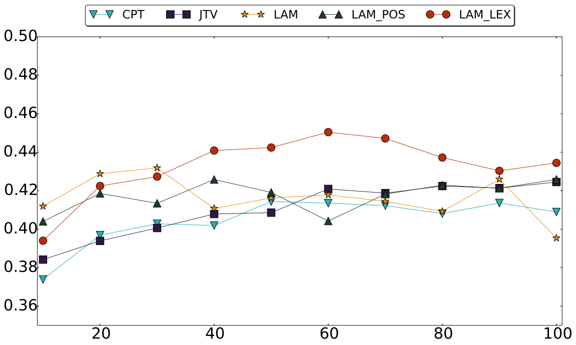


Figure 3: Average mixed topic-perspective CV coherence, across different number of topics.

Topics	**CPT**	**JTV**	**LAM_LEX**
10	0.254	0.308	**0.427**
20	0.369	0.366	**0.517**
30	0.401	0.389	**0.573**
40	0.426	0.394	**0.604**
50	0.431	0.408	**0.564**

Table 3: Averaged LA measure across all topic-perspectives for different models.

To compare the quality of perspectives inferred by LAM_LEX and CPT (over 30 topics) we also conducted human evaluation. To do this, topics and perspectives were represented as bag-of-words. Each perspective was also represented with its three most representative sentences. The outputs from the two models was first merged and shuffled. Two external annotators were then asked to answer ('yes' or 'no') for each topic if they could differentiate two perspectives. Co-

[10] We can also rank sentences for an argument a under a topic z based on the generative probability of sentences. But this consistently produce worse results.

hen's Kappa coefficient (Cohen, 1968) for inter-annotator agreement was 0.421. Table 4 shows the results of the evaluation and it is clear that LAM_LEX outperforms CPT.

Annotator	LAM_LEX	CPT
1	**0.63**	0.10
2	**0.67**	0.34
1&2	**0.53**	0.10

Table 4: Accuracy on detecting perspectives according to the human outputs. In 1&2 a 'yes' answer is only valid if marked by both annotators.

Table 5 shows the three most representative perspective sentences for some of the extracted topics by LAM_LEX and CPT, to illustrate how LAM_LEX obtains more coherent sentences.[11] The example involving the first topic shows a case where LAM_LEX learned non-contrastive perspectives: both deal with *Palestina*, but focusing in different aspects (*illegal settlements* vs. *Israel actions*). In contrast, CPT mixed perspectives about Israel/Palestina and other viewpoints about *GCSE* and *mortgages*. In the second topic, LAM_LEX ranks at the top sentences relating to *Sinn Fein & Northern Ireland*, that show two different stances (*positive* vs *negative)* meanwhile in CPT it is not possible to infer any clear perspective despite sentences contain semantically related terms.

Table 6 shows cases where LAM_LEX obtained a less-coherent output according to the annotators. The first topic deals with Shaker Aamer and the legality of its imprisonment in Guantanamo. Perspective 2 reflects this issue, but Perspective 1 includes other types of crimes. The second example discusses issues relating to transports. While Perspective 1 is all about the negotiation with the train company, First Great Western, on its franchise extension proposal, Perspective 2 contains sentences relating to a number of different issues under transports. To alleviate this problem, we hypothesise that additional levels of information (in addition to the topic and perspective levels), such as a Bill or a speaker, might be needed to better distinguish different topics and perspectives that share a significant proportion of vocabulary.

5.1.3 Discussion

LAM_LEX gave a glimpse of the perspectives that occupy a topic. However, in many cases those differ from the initial expectation given the priors used in our model. Despite of the use of the subjectivity lexicon to initialise the Dirichlet prior β^a for the topic-argument-word distribution, after a few iterations the initial distribution changes radically and turns instead into contrastive and non-contrastive perspectives, with the latter group being the most common one. We think this is due to factors that involve: (1) lack of contrastive speeches about very specific topics; and (2) jargon from the House of Commons that makes the task more challenging as stances are showed in subtle and polite way. This is also in line with what has been previously observed in (Mohammad et al., 2016b) that a person may express the same stance towards a target by using negative or positive language. This shows that LAM_LEX can infer perspectives from raw data, but we have little control on guiding the model on what perspectives to extract.

6 Conclusion and Future Work

We have presented LAM, a model able to provide a glimpse of what is going on in political debates, without relying on any labelled data and assuming the perspectives of a topic to be latent. It is implemented through a hierarchical Bayesian model considering that words can be separated as topic, argument or background words and follow different generative routes. Experiments show that our model obtains more coherent topics than related approaches and also extracts more interpretable perspectives. The code is made available at `https://github.com/aghie/lam`.

Although LAM can extract perspectives under a certain topic, there is little control in what kind of information to extract (e.g. we might want only contrastive or non-contrastive arguments). In future work, we plan to improve the model through complex priors or semantic similarity strategies. Also, adding a 'Bill' level could be beneficial as speeches about the same Bill should share the same high-level topic. But we need labels indicating to which Bill the text belongs to. Including a 'speaker' level to know which parliamentarians discuss which topics is another interesting path to follow.

Acknowledgments

We thank Charles Marshall for crawling the HCD data. DV was funded by MECD (FPU 13/01180), MINECO (FFI2014-51978-C2-2-R) and Inditex-UDC grants for research stays. YH is partly

[11] The examples were identified as two perspectives by at least one annotator. Its selection was made based on an existence of a similar topic both on LAM_LEX and CPT outputs.

	LAM_LEX	CPT
Topic 1	israel, iran, syria, settlement, relocation, counter-terrorism gaza, tpims, airline, metropolitan	israel, iran, middle, settlement, palestinian, israeli, gaza, negotiation, village, hamas
Perspective 1	a) It is contrary to international law in that sense, and any nation has obligations when dealing with occupied territories and their occupants. b) Again, I reiterate the difference between the two issues: one concerns the illegal settlements, and the other is a planning matter that we have raised concerns about. c) That is a slightly separate debate or concern if I can put it that way to the illegal settlements that have been put forward, but nevertheless we are concerned and are having a dialogue with Israel about that.	a) Does he agree that unless that happens it is difficult to envisage a unified and prosperous Palestinian state existing alongside Israel? b) Will the Minister discuss that issue with the Israeli Government, urge them to reconsider the upcoming evictions and demolitions due for next month, and instead consider villages co-existing side by side in the spirit of peace? c) That is caused partly by the security situation in Sinai and the Egyptian response to that, and partly by the situation between Israel and the Palestinians in Gaza.
Perspective 2	a) More to the point, the continual encroachment by the Israeli Government makes it impossible for East Jerusalem to become the capital of a Palestinian state. b) We know that 163 Palestinian children are being held in Israeli military detention, and that many are being held inside Israel in direct violation of the fourth Geneva c) We want to see the establishment of a sovereign and independent Palestinian state, living in peace and security alongside Israel.	a) I share the hon. Ladys desire that every school should offer three separate sciences at GCSE; that is very important. b) Everybody here will know, however, that a 1,000 monthly payment sustains a mortgage of 200,000. c) As I clarified, that is a different matter to the debate about the occupied Palestinian territories, but nevertheless we want a robust planning process that adequately.
Topic 2	stormont, sinn, fein, setback, scene, flag, belfast, backwards, surprise, feeling	northern, ireland, stormont, sinn, fein, fairly, poverty, corporation, molyneaux, monday
Perspective 1	a) Would that it was as simple as getting behind the democratic authority in Libyait is not clear that there is a democratic authority behind which we can get. b) It is very important for the Stormont House agreement to be implemented fully and fairly, including all the sections on welfare and budgets. c) The Stormont House agreement was a big step forward, and it is vital for all parties to work to ensure that it is implemented fully and fairly.	a) Following our two major reform programmes, spend has fallen to 1.7 billion in 2013-14 and is expected to fall to about 1.5 billion once the reforms have fully worked through the system. b) Universal credit is a major reform that will transform the welfare state in Britain for the better. c) We have put in place a five-year reform programme that will bring our courts into the 21st century.
Perspective 2	a) There is a clear disparity in political party funding in Northern Ireland, yet Sinn Fein Members continue to draw hundreds of thousands of pounds in allowances from this House, despite not taking their seats. b) In light of the reneging of Sinn Fein on the introduction of welfare reform, what implications does the Minister see in the devolution of corporation tax in Northern Ireland? c) There is no doubt that the announcement by Sinn Fein on Monday was a significant setback for the Stormont House agreement, but it is inevitable that there will be bumps in the road with agreements of this nature.	a) Will the National Crime Agency specifically target the organised criminal gangs that are engaging in subterfuge and in the organised criminal activity of fuel laundering along the border areas of Northern Ireland? b) This will ensure that the people of Northern Ireland are afforded the same protections from serious and organised crime as those in the rest of the United Kingdom. c) The Treasury has had meetings with the European Commission to discuss the reinstatement of the aggregate credit levy scheme for Northern Ireland, which could serve as a further tool of investment in infrastructure.

Table 5: Output sample for representative perspective sentences in non-contrastive and contrastive topics.

	LAM_LEX
Topic 1	aamer, shaker, bay, guantanamo, america, obama, american, timetable, embassy, harlington
Perspective 1	a) NSPCC research has shown that six in 10 teenagers have been asked for sexual images or videos online. b) Does my right hon. Friend agree that the report released last week that suggested that the punishments for online and offline crime should be equalised demonstrates that education is needed to show that the two sentences should be equal? c) I can confirm that the Government have announced that we are entering into a negotiation on a contract for difference for the Swansea bay lagoon to decide whether the project is affordable and represents value for money.
Perspective 2	a) This has been a helpful and constructive debate, and I join others in congratulating the hon. Member for Hayes and Harlington (John McDonnell) on securing it through the Backbench Business Committee. b) I thank the Backbench Business Committee for allocating time for this critical debate at an important time in the campaign to secure the release of Shaker Aamer. c) He has been one of the leading parliamentary campaigners for Mr Aamers release, and I acknowledge the presence of the hon. Member for Battersea (Jane Ellison) , who is the constituency MP for Mr Aamer and his familyindeed, this debate provides an important opportunity to follow up a Backbench Business Committee debate on the same subject that she initiated in April 2013.
Topic 2	passenger, franchise, fare, coast, connectivity, journey, gloucester, user, anglia, stagecoach
Perspective 1	a) Will my hon. Friend confirm when she expects the Departments negotiations with First Great Western on its franchise extension proposals, which include the improvements at Gloucester, to be completed? b) The hon. Gentleman will be pleased to learn that we expect to conclude negotiations with First Great Western and to finalise the second directly awarded franchise contract during this month, and expect the provision of services to start in September. c) My plans for the regeneration of the city of Gloucester include a new car park and entrance to Gloucester station, but they depend on a land sale agreement between the Ministry of Justice and the city council and the lands onward leasing to First Great Western.
Perspective 2	a) I do not want any young people to feel frightened of attending school or of their journey to and from school, and, sadly, that applies particularly to members of the Jewish community at present. b) Why, instead of real localism, have this Government presided over a failed record, with bus fares up 25% and 2,000 routes cut, and a broken bus market, which lets users down, but which Labour will fix in government? c) Last week we introduced the new invitation to tender for the Northern Rail and TransPennine Express services, and transferred East Coast back to the private sector.

Table 6: Output sample for non-representative perspective sentences in the LAM_LEX model.

funded by the Natural Science Foundation of China (61528302).

References

Rawia Awadallah, Maya Ramanath, and Gerhard Weikum. 2012. Opinions network for politically controversial topics. In *Proceedings of the first edition workshop on Politics, elections and data*, pages 15–22. ACM.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Filip Boltuzic and Jan Šnajder. 2014. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58. Citeseer.

Elena Cabrio and Serena Villata. 2012. Combining textual entailment and argumentation theory for supporting online debates interactions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL): Short Papers-Volume 2*, pages 208–212.

George Casella and Edward I George. 1992. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Sanmay Das and Allen Lavoie. 2014. Automated inference of point of view from user interactions in collective intelligence venues. In *ICML*, pages 82–90.

Yi Fang, Luo Si, Naveen Somasundaram, and Zhengtao Yu. 2012. Mining contrastive opinions on political texts using cross-perspective topic model. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 63–72. ACM.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 375–384. ACM.

Chenghua Lin, Yulan He, Richard Everson, and Stefan Ruger. 2012. Weakly supervised joint sentiment-topic detection from text. *IEEE Transactions on Knowledge and Data engineering*, 24(6):1134–1145.

Marco Lippi and Paolo Torroni. 2016a. Argument mining from speech: Detecting claims in political debates. In *Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*.

Marco Lippi and Paolo Torroni. 2016b. Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology (TOIT)*, 16(2):10.

Thomas P Minka. 2003. A comparison of numerical optimizers for logistic regression. *Unpublished draft*.

Saif M Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016a. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the International Workshop on Semantic Evaluation (SemEval)*, volume 16.

Saif M Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2016b. Stance and sentiment in tweets. *arXiv preprint arXiv:1605.01655*.

Huy Nguyen and Diane J Litman. 2015. Extracting argument and domain words for identifying argument components in texts. In *ArgMining@ HLT-NAACL*, pages 22–28.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 938–948.

Minghui Qiu, Yanchuan Sim, Noah A Smith, and Jing Jiang. 2015. Modeling user arguments, interactions, and attributes for stance prediction in online debate forums. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 855–863.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408. ACM.

Christos Sardianos, Ioannis Manousos Katakis, Georgios Petasis, and Vangelis Karkaletsis. 2015. Argument extraction from news. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 56–66.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56.

Amine Trabelsi and Osmar R Zaıane. 2014. Finding arguing expressions of divergent viewpoints in online debates. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)@ EACL*, pages 35–43.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Guido Zarrella and Amy Marsh. 2016. Mitre at semeval-2016 task 6: Transfer learning for stance detection. *arXiv preprint arXiv:1606.03784*.

“i have a feeling trump will win..................”: Forecasting Winners and Losers from User Predictions on Twitter

Sandesh Swamy, **Alan Ritter**
Computer Science & Engineering
The Ohio State University
Columbus, OH
swamy.14@osu.edu, aritter@cse.ohio-state.edu

Marie-Catherine de Marneffe
Department of Linguistics
The Ohio State University
Columbus, OH
mcdm@ling.ohio-state.edu

Abstract

Social media users often make explicit predictions about upcoming events. Such statements vary in the degree of certainty the author expresses toward the outcome: “Leonardo DiCaprio will win Best Actor” vs. “Leonardo DiCaprio may win” or “No way Leonardo wins!”. Can popular beliefs on social media predict who will win? To answer this question, we build a corpus of tweets annotated for veridicality on which we train a log-linear classifier that detects positive veridicality with high precision.[1] We then forecast uncertain outcomes using the *wisdom of crowds*, by aggregating users’ explicit predictions. Our method for forecasting winners is fully automated, relying only on a set of contenders as input. It requires no training data of past outcomes and outperforms sentiment and tweet volume baselines on a broad range of contest prediction tasks. We further demonstrate how our approach can be used to measure the reliability of individual accounts’ predictions and retrospectively identify surprise outcomes.

(a)	*Natalie Portman is gonna beat out Meryl Streep for best actress*
(b)	*La La Land doesn’t have lead actress and actor guaranteed. Natalie Portman will probably (and should) get best actress*
(c)	*Adored #LALALAND but it’s #NataliePortman who deserves the best actress #oscar #OscarNoms > superb acting*

Table 1: Examples of tweets expressing varying degrees of veridicality toward Natalie Portman winning an Oscar.

1 Introduction

In the digital era we live in, millions of people broadcast their thoughts and opinions online. These include predictions about upcoming events of yet unknown outcomes, such as the Oscars or election results. Such statements vary in the extent to which their authors intend to convey the event will happen. For instance, (a) in Table 1 strongly asserts the win of Natalie Portman over Meryl Streep, whereas (b) imbues the claim with uncertainty. In contrast, (c) does not say anything about the likelihood of Natalie Portman winning (although it clearly indicates the author would like her to win).

Prior work has made predictions about contests such as NFL games (Sinha et al., 2013) and elections using tweet volumes (Tumasjan et al., 2010) or sentiment analysis (O’Connor et al., 2010; Shi et al., 2012). Many such indirect signals have been shown useful for prediction, however their utility varies across domains. In this paper we explore whether the “wisdom of crowds” (Surowiecki, 2005), as measured by users’ explicit predictions, can predict outcomes of future events. We show how it is possible to accurately forecast winners, by aggregating many individual predictions that assert an outcome. Our approach requires no historical data about outcomes for training and can directly be adapted to a broad range of contests.

To extract users’ predictions from text, we present TwiVer, a system that classifies veridicality toward future contests with uncertain outcomes. Given a list of contenders competing in a contest (e.g., Academy Award for Best Actor), we use TwiVer to count how many tweets explicitly assert the win of each contender. We find that aggregating veridicality in this way provides

[1] The code and data can be found at `https://github.com/SandeshS/Twitter-Veridicality`

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1583–1592
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

an accurate signal for predicting outcomes of future contests. Furthermore, TwiVer allows us to perform a number of novel qualitative analyses including retrospective detection of *surprise outcomes* that were not expected according to popular belief (Section 4.5). We also show how TwiVer can be used to measure the number of correct and incorrect predictions made by individual accounts. This provides an intuitive measurement of the reliability of an information source (Section 4.6).

2 Related Work

In this section we summarize related work on text-driven forecasting and computational models of veridicality.

Text-driven forecasting models (Smith, 2010) predict future response variables using text written in the present: e.g., forecasting films' box-office revenues using critics' reviews (Joshi et al., 2010), predicting citation counts of scientific articles (Yogatama et al., 2011) and success of literary works (Ashok et al., 2013), forecasting economic indicators using query logs (Choi and Varian, 2012), improving influenza forecasts using Twitter data (Paul et al., 2014), predicting betrayal in online strategy games (Niculae et al., 2015) and predicting changes to a knowledge-graph based on events mentioned in text (Konovalov et al., 2017). These methods typically require historical data for fitting model parameters, and may be sensitive to issues such as concept drift (Fung, 2014). In contrast, our approach does not rely on historical data for training; instead we forecast outcomes of future events by directly extracting users' explicit predictions from text.

Prior work has also demonstrated that user sentiment online directly correlates with various real-world time series, including polling data (O'Connor et al., 2010) and movie revenues (Mishne and Glance, 2006). In this paper, we empirically demonstrate that veridicality can often be more predictive than sentiment (Section 4.1).

Also related is prior work on *detecting veridicality* (de Marneffe et al., 2012; Søgaard et al., 2015) and sarcasm (González-Ibánez et al., 2011). Soni et al. (2014) investigate how journalists frame quoted content on Twitter using predicates such as *think*, *claim* or *admit*. In contrast, our system TwiVer, focuses on the author's belief toward a claim and direct predictions of future events as opposed to quoted content.

Our approach, which aggregates predictions extracted from user-generated text is related to prior work that leverages explicit, positive veridicality, statements to make inferences about users' demographics. For example, Coppersmith et al. (2014; 2015) exploit users' self-reported statements of diagnosis on Twitter.

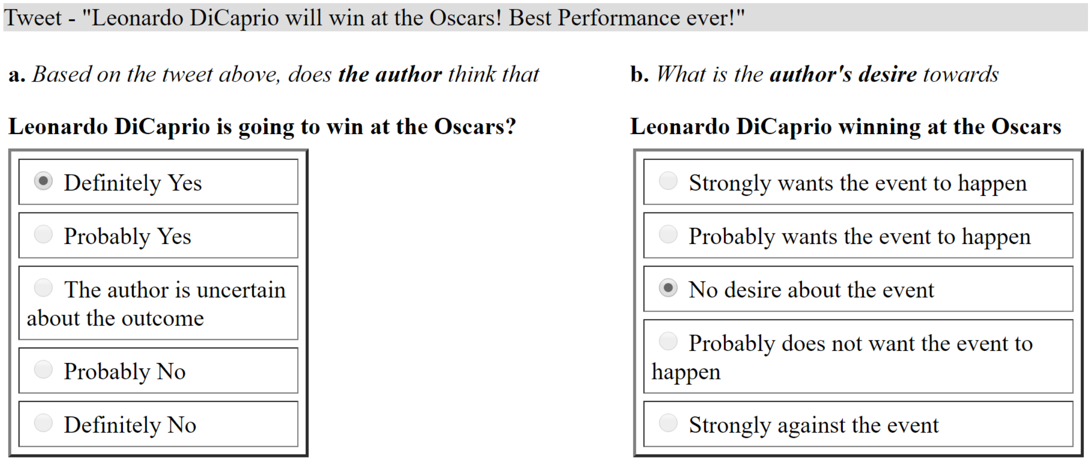
3 Measuring the Veridicality of Users' Predictions

The first step of our approach is to extract statements that make explicit predictions about unknown outcomes of future events. We focus specifically on *contests* which we define as events planned to occur on a specific date, where a number of *contenders* compete and a single *winner* is chosen. For example, Table 2 shows the contenders for Best Actor in 2016, highlighting the winner.

Actor	Movie
Leonardo DiCaprio	**The Revenant**
Bryan Cranston	Trumbo
Matt Damon	The Martian
Michael Fassbender	Steve Jobs
Eddie Redmayne	The Danish Girl

Table 2: Oscar nominations for Best Actor 2016.

To explore the accuracy of user predictions in social media, we gathered a corpus of tweets that mention events belonging to one of the 10 types listed in Table 4. Relevant messages were collected by formulating queries to the Twitter search interface that include the name of a contender for a given contest in conjunction with the keyword *win*. We restricted the time range of the queries to retrieve only messages written before the time of the contest to ensure that outcomes were unknown when the tweets were written. We include 10 days of data before the event for the presidential primaries and the final presidential elections, 7 days for the Oscars, Ballon d'Or and Indian general elections, and the period between the semi-finals and the finals for the sporting events. Table 3 shows several example queries to the Twitter search interface which were used to gather data. We automatically generated queries, using templates, for events scraped from various websites: 483 queries were generated for the presidential primaries based on events scraped from ballotpe-



Tweet - "Leonardo DiCaprio will win at the Oscars! Best Performance ever!"

a. *Based on the tweet above, does* ***the author*** *think that*

Leonardo DiCaprio is going to win at the Oscars?

● Definitely Yes
○ Probably Yes
○ The author is uncertain about the outcome
○ Probably No
○ Definitely No

b. *What is the* ***author's desire*** *towards*

Leonardo DiCaprio winning at the Oscars

○ Strongly wants the event to happen
○ Probably wants the event to happen
● No desire about the event
○ Probably does not want the event to happen
○ Strongly against the event

Figure 1: Example of one item to be annotated, as displayed to the Turkers.

dia[2] , 176 queries were generated for the Oscars,[3] 18 for Ballon d'Or,[4] 162 for the Eurovision contest,[5] 52 for Tennis Grand Slams,[6] 6 for the Rugby World Cup,[7] 18 for the Cricket World Cup,[8] 12 for the Football World Cup,[9] 76 for the 2016 US presidential elections,[10] and 68 queries for the 2014 Indian general elections.[11]

We added an event prefix (e.g., "Oscars" or the state for presidential primaries), a keyword ("win"), and the relevant date range for the event. For example, "Oscars Leonardo DiCaprio win since:2016-2-22 until:2016-2-28" would be the query generated for the first entry in Table 2.

Minnesota Rubio win since:2016-2-18 until:2016-3-1
Vermont Sanders win since:2016-2-18 until:2016-3-1
Oscars Sandra Bullock win since:2010-3-1 until:2010-3-7
Oscars Spotlight win since:2016-2-22 until:2016-2-28

Table 3: Examples of queries to extract tweets.

We restricted the data to English tweets only, as tagged by *langid.py* (Lui and Baldwin, 2012). Jaccard similarity was computed between messages to identify and remove duplicates.[12] We removed URLs and preserved only tweets that mention contenders in the text. This automatic post-processing left us with 57,711 tweets for all winners and 55,558 tweets for losers (contenders who did not win) across all events. Table 4 gives the data distribution across event categories.

Event	Number of tweets Winners	 Losers
2016 US Presidential primaries	20,347	17,873
Oscars (2009 – 2016)	1,498	872
Tennis Grand Slams (2011 – 2016)	10,785	19,745
Ballon d'Or Award (2010 – 2016)	3,492	3,285
Eurovision (2010 – 2016)	261	1,421
2016 US Presidential elections	9,679	3,966
2014 Indian general elections	920	736
Rugby World Cup (2010 – 2016)	272	379
Football World Cup (2010 – 2016)	8,129	5,489
Cricket World Cup (2010 – 2016)	2,328	1,792

Table 4: Number of tweets for each event category.

3.1 Mechanical Turk Annotation

We obtained veridicality annotations on a sample of the data using Amazon Mechanical Turk. For each tweet, we asked Turkers to judge veridicality toward a candidate winning as expressed in the tweet as well as the author's desire toward the event. For veridicality, we asked Turkers to rate whether the author believes the event will happen on a 1-5 scale ("Definitely Yes", "Probably Yes", "Uncertain about the outcome", "Probably No",

[2] https://ballotpedia.org/Main_Page

[3] https://en.wikipedia.org/wiki/Academy_Awards

[4] https://en.wikipedia.org/wiki/Ballon_d%27Or

[5] https://en.wikipedia.org/wiki/Eurovision_Song_Contest

[6] https://en.wikipedia.org/wiki/Grand_Slam_(tennis)

[7] https://en.wikipedia.org/wiki/Rugby_World_Cup

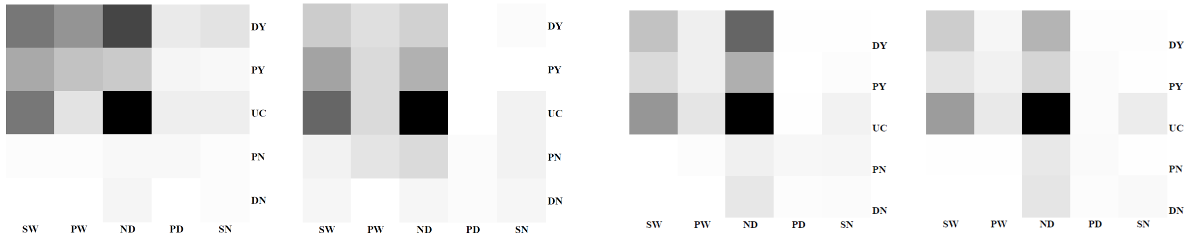
[8] https://en.wikipedia.org/wiki/Cricket_World_Cup

[9] https://en.wikipedia.org/wiki/FIFA_World_Cup

[10] https://en.wikipedia.org/wiki/United_States_presidential_election,_2016

[11] https://en.wikipedia.org/wiki/Indian_general_election,_2014

[12] A threshold of 0.7 was used.



(a) Oscar winners (b) Oscar losers (c) All events winners (d) All events losers

Figure 2: Heatmaps showing annotation distributions for one of the events - the Oscars and all event types, separating winners from losers. Vertical labels indicate veridicality (DY "Definitely Yes", PY "Probably Yes", UC "Uncertain about the outcome", PN "Probably No" and DN "Definitely No"). Horizontal labels indicate desire (SW "Strongly wants the event to happen", PW "Probably wants the event to happen", ND "No desire about the event outcome", PD "Probably does not want the event to happen", SN "Strongly against the event happening"). More data in the upper left hand corner indicates there are more tweets with positive veridicality and desire.

"Definitely No"). We also added a question about the author's desire toward the event to make clear the difference between veridicality and desire. For example, "I really want Leonardo to win at the Oscars!" asserts the author's desire toward Leonardo winning, but remains agnostic about the likelihood of this outcome, whereas "Leonardo DiCaprio will win the Oscars" is predicting with confidence that the event will happen.

Figure 1 shows the annotation interface presented to Turkers. Each HIT contained 10 tweets to be annotated. We gathered annotations for $1,841$ tweets for winners and $1,702$ tweets for losers, giving us a total of $3,543$ tweets. We paid $0.30 per HIT. The total cost for our dataset was $1,000. Each tweet was annotated by 7 Turkers. We used MACE (Hovy et al., 2013) to resolve differences between annotators and produce a single gold label for each tweet.

Figures 2a and 2c show heatmaps of the distribution of annotations for the winners for the Oscars in addition to all categories. In both instances, most of the data is annotated with "Definitely Yes" and "Probably Yes" labels for veridicality. Figures 2b and 2d show that the distribution is more diverse for the losers. Such distributions indicate that the veridicality of crowds' statements could indeed be predictive of outcomes. We provide additional evidence for this hypothesis using automatic veridicality classification on larger datasets in §4.

3.2 Veridicality Classifier

The goal of our system, TwiVer, is to automate the annotation process by predicting how veridical a tweet is toward a candidate winning a contest: is the candidate deemed to be winning, or is the author uncertain? For the purpose of our experiments, we collapsed the five labels for veridicality into three: positive veridicality ("Definitely Yes" and "Probably Yes"), neutral ("Uncertain about the outcome") and negative veridicality ("Definitely No" and "Probably No").

We model the conditional distribution over a tweet's veridicality toward a candidate c winning a contest against a set of opponents, O, using a log-linear model:

$$P(y = v|c, \text{tweet}) \propto \exp\left(\theta_v \cdot f(c, O, \text{tweet})\right)$$

where v is the veridicality (positive, negative or neutral).

To extract features $f(c, O, \text{tweet})$, we first preprocessed tweets retrieved for a specific event to identify named entities, using (Ritter et al., 2011)'s Twitter NER system. Candidate (c) and opponent entities were identified in the tweet as follows:

- TARGET (t). A target is a named entity that matches a contender name from our queries.
- OPPONENT (O). For every event, along with the current TARGET entity, we also keep track of other contenders for the same event. If a named entity in the tweet matches with one of other contenders, it is labeled as opponent.
- ENTITY (e): Any named entity which does not match the list of contenders.

Figure 3 illustrates the named entity labeling for a tweet obtained from the query "Oscars Leonardo DiCaprio win since:2016-2-22 until:2016-2-28". Leonardo DiCaprio is the TARGET, while the named entity tag for Bryan Cranston, one of the

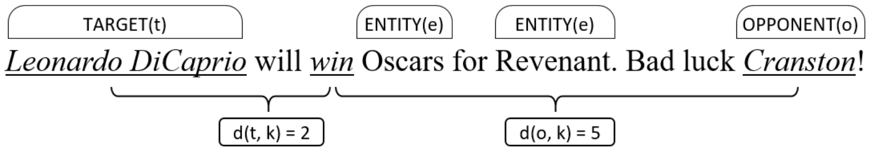

TARGET(t)
ENTITY(e)
ENTITY(e)
OPPONENT(o)
Leonardo DiCaprio will win Oscars for Revenant. Bad luck Cranston!
d(t, k) = 2
d(o, k) = 5

Figure 3: Illustration of the three named entity tags and distance features between entities and keyword *win* for a tweet retrieved by the query "Oscars Leonardo DiCaprio win since:2016-2-22 until:2016-2-28".

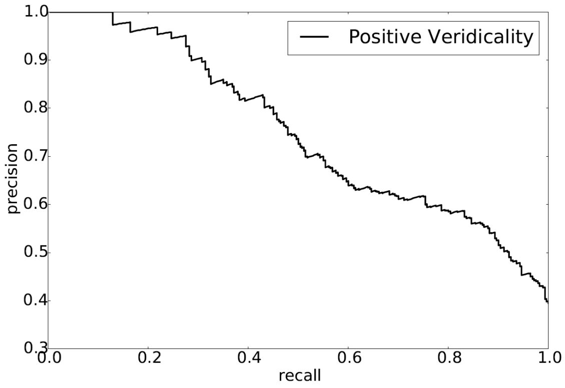


Figure 4: Precision/Recall curve showing TwiVer performance in identifying positive veridicality tweets in the test data.

	P	R	F1
− Context	47.7	**96.4**	63.8
− Distance	57.5	82.5	67.7
− Punctuation	53.4	88.2	66.6
− Dependency path	56.9	85.4	68.2
− Negated keyword	56.7	86.4	68.4
All features	**58.7**	83.1	**68.8**

Table 5: Feature ablation of the positive veridicality classifier by removing each group of features from the full set. The point of maximum F1 score is shown in each case.

losers for the Oscars, is re-tagged as OPPONENT. These tags provide information about the position of named entities relative to each other, which is used in the features.

3.3 Features

We use five feature templates: context words, distance between entities, presence of punctuation, dependency paths, and negated keyword.

Target and opponent contexts. For every TARGET (t) and OPPONENT ($o \in O$) entities in the tweet, we extract context words in a window of one to four words to the left and right of the TARGET ("Target context") and OPPONENT ("Opponent context"), e.g., *t will win*, *I'm going with* t, o *will win*.

Keyword context. For target and opponent entities, we also extract words between the entity and our specified keyword (k) (*win* in our case): t *predicted to* k, o *might* k.

Pair context. For the election type of events, in which two target entities are present (contender and state. e.g., *Clinton, Ohio*), we extract words between these two entities: e.g., t_1 *will win* t_2.

Distance to keyword. We also compute the distance of TARGET and OPPONENT entities to the keyword.

Punctuation. We introduce two binary features for the presence of exclamation marks and question marks in the tweet. We also have features which check whether a tweet ends with an exclamation mark, a question mark or a period. Punctuation, especially question marks, could indicate how certain authors are of their claims.

Dependency paths. We retrieve dependency paths between the two TARGET entities and between the TARGET and keyword (*win*) using the TweeboParser (Kong et al., 2014) after applying rules to normalize paths in the tree (e.g., "doesn't" → "does not").

Negated keyword. We check whether the keyword is negated (e.g., "not win", "never win"), using the normalized dependency paths.

We randomly divided the annotated tweets into a training set of 2,480 tweets, a development set of 354 tweets and a test set of 709 tweets. MAP parameters were fit using LBFGS-B (Zhu et al., 1997). Table 6 provides examples of high-weight features for positive and negative veridicality.

3.4 Evaluation

We evaluated TwiVer's precision and recall on our held-out test set of 709 tweets. Figure 4 shows the precision/recall curve for positive veridicality. By setting a threshold on the probability score to be greater than 0.64, we achieve a precision of

Positive Veridicality			Negative Veridicality		
Feature Type	Feature	Weight	Feature Type	Feature	Weight
Keyword context	TARGET *will* KEYWORD	0.41	Negated keyword	keyword is negated	0.47
Keyword dep. path	TARGET → *to* → KEYWORD	0.38	Keyword context	TARGET *won't* KEYWORD	0.41
Keyword dep. path	TARGET ← *is* → *going* → *to* → KEYWORD	0.29	Opponent context	OPPONENT *will win*	0.37
Target context	TARGET *is favored to win*	0.19	Keyword dep. path	TARGET ← *will* → *not* → KEYWORD	0.31
Keyword context	TARGET *are going to* KEYWORD	0.15	Distance to keyword	OPPONENT *closer to* KEYWORD	0.28
Target context	TARGET *predicted to win*	0.13	Target context	TARGET *may not win*	0.27
Pair context	TARGET1 *could win* TARGET2	0.13	Keyword dep. path	OPPONENT ← *will* → KEYWORD	0.23
Distance to keyword	TARGET *closer to* KEYWORD	0.11	Target context	TARGET *can't win*	0.18

Table 6: Some high-weight features for positive and negative veridicality.

Tweet	Gold	Predicted
The heart wants **Nadal** to win tomorrow but the mind points to a Djokovic win over 4 sets. Djokovic 7-5 4-6 7-5 6-4 **Nadal** for me.	negative	positive
Hopefully tomorrow Federer will win and beat that **Nadal** guy lol	neutral	negative
There is no doubt **India** have the tools required to win their second World Cup. Whether they do so will depend on ...	positive	neutral

Table 7: Some classification errors made by TwiVer. Contenders queried for are highlighted.

80.1% and a recall of 44.3% in identifying tweets expressing a positive veridicality toward a candidate winning a contest.

3.5 Performance on held-out event types

To assess the robustness of the veridicality classifier when applied to new types of events, we compared its performance when trained on all events vs. holding out one category for testing. Table 9 shows the comparison: the second and third columns give F1 score when training on all events vs. removing tweets related to the category we are testing on. In most cases we see a relatively modest drop in performance after holding out training data from the target event category, with the exception of elections. This suggests our approach can be applied to new event types without requiring in-domain training data for the veridicality classifier.

3.6 Error Analysis

Table 7 shows some examples which TwiVer incorrectly classifies. These errors indicate that even though shallow features and dependency paths do a decent job at predicting veridicality, deeper text understanding is needed for some cases. The opposition between "the heart ... the mind" in the first example is not trivial to capture. Paying attention to matrix clauses might be important too (as shown in the last tweet "There is no doubt ...").

4 Forecasting Contest Outcomes

We now have access to a classifier that can automatically detect positive veridicality predictions about a candidate winning a contest. This enables us to evaluate the accuracy of the crowd's wisdom by retrospectively comparing popular beliefs (as extracted and aggregated by TwiVer) against known outcomes of contests.

We will do this for each award category (Best Actor, Best Actress, Best Film and Best Director) in the Oscars from 2009 – 2016, for every state for both Republican and Democratic parties in the 2016 US primaries, for both the candidates in every state for the final 2016 US presidential elections, for every country in the finals of Eurovision song contest, for every contender for the Ballon d'Or award, for every party in every state for the 2014 Indian general elections, and for the contenders in the finals for all sporting events.

4.1 Prediction

A simple voting mechanism is used to predict contest outcomes: we collect tweets about each contender written before the date of the event,[13] and use TwiVer to measure the veridicality of users' predictions toward the events. Then, for each contender, we count the number of tweets that are labeled as positive with a confidence above 0.64, as well as the number of tweets with positive veridicality for all other contenders. Table 11 illustrates these counts for one contest, the Oscars Best Actress in 2014.

We then compute a simple prediction score, as follows:

$$\text{score} = (|T_c| + 1)/(|T_c| + |T_O| + 2) \qquad (1)$$

[13] These are a different set of tweets than those TwiVer was trained on.

Event	Veridicality			Sentiment			Frequency			#predictions
	P	R	F1	P	R	F1	P	R	F1	
Oscars	**80.6**	80.6	**80.6**	52.9	87.1	63.5	54.7	**93.5**	69.0	151
Ballon d'Or	**100.0**	100.0	**100.0**	85.7	100.0	92.2	85.7	100.0	92.2	18
Eurovision	**83.3**	71.4	**76.8**	38.5	71.4	50.0	50.0	57.1	53.3	87
Tennis Grand Slam	50.0	100.0	66.6	50.0	100.0	66.6	50.0	100.0	66.6	52
Rugby World Cup	**100.0**	100.0	**100.0**	50.0	100.0	66.6	50.0	100.0	66.6	4
Cricket World Cup	**66.7**	85.7	**75.0**	58.3	**100.0**	73.6	58.3	**100.0**	73.6	14
Football World Cup	**71.4**	100.0	**83.3**	62.5	100.0	76.9	**71.4**	100.0	**83.3**	10
Presidential primaries	**66.0**	**88.0**	**75.4**	58.9	82.5	68.7	63.4	78.7	70.2	211
2016 US presidential elections	60.9	**100.0**	**75.6**	63.3	73.8	68.1	**69.0**	69.0	69.0	84
2014 Indian general elections	**95.8**	100.0	**97.8**	65.6	91.3	76.3	56.1	100.0	71.8	52

Table 8: Performance of Veridicality, Sentiment baseline, and Frequency baseline on all event categories (%).

Event	Train on all	Train without held-out event	$\|T_t\|$
Oscars	69.5	63.8	64
Ballon d'Or	54.6	46.6	61
Eurovision	65.7	63.2	48
Tennis Grand Slams	52.1	45.5	44
Rugby World Cup	56.5	58.1	44
Cricket World Cup	61.9	66.8	49
Football World Cup	76.0	67.5	56
Presidential primaries	59.8	48.1	117
2016 US presidential elections	52.0	52.3	54
Indian elections	60.3	39.0	44

Table 9: F1 scores for each event when training on all events vs. holding out that event from training. $|T_t|$ is the number of tweets of that event category present in the test dataset.

where $|T_c|$ is the set of tweets mentioning positive veridicality predictions toward candidate c, and $|T_O|$ is the set of all tweets predicting any opponent will win. For each contest, we simply predict as winner the contender whose score is highest.

4.2 Sentiment Baseline

We compare the performance of our approach against a state-of-the-art sentiment baseline (Mohammad et al., 2013). Prior work on social media analysis used sentiment to make predictions about real-world outcomes. For instance, O'Connor et al. (2010) correlated sentiment with public opinion polls and Tumasjan et al. (2010) use political sentiment to make predictions about outcomes in German elections.

We use a re-implementation of (Mohammad et al., 2013)'s system[14] to estimate sentiment for tweets in our corpus. We run the tweets obtained for every contender through the sentiment analysis system to obtain a count of positive labels. Sentiment scores are computed analogously to veridicality using Equation (1). For each contest, the contender with the highest sentiment prediction score is predicted as the winner.

4.3 Frequency Baseline

We also compare our approach against a simple frequency (tweet volume) baseline. For every contender, we compute the number of tweets that has been retrieved. Frequency scores are computed in the same way as for veridicality and sentiment using Equation (1). For every contest, the contender with the highest frequency score is selected to be the winner.

4.4 Results

Table 8 gives the precision, recall and max-F1 scores for veridicality, sentiment and volume-based forecasts on all the contests. The veridicality-based approach outperforms sentiment and volume-based approaches on 9 of the 10 events considered. For the Tennis Grand Slam, the three approaches perform poorly. The difference in performance for the veridicality approach is quite lower for the Tennis events than for the other events. It is well known however that winners of tennis tournaments are very hard to predict. The performance of the players in the last minutes

[14] `https://github.com/ntietz/tweetment`

		Veridicality			Sentiment		
		Contender		Score	Contender		Score
OSCARS		**Leonardo DiCaprio**		0.97	**Julianne Moore**		0.85
		Natalie Portman		0.92	Mickey Rourke		0.83
		Julianne Moore		0.91	**Leonardo DiCaprio (2016)**		0.82
		Daniel Day-Lewis		0.90	**Kate Winslet**		0.75
		Slumdog Millionaire		0.75	Leonardo DiCaprio (2014)		0.69
		Matthew McConaughey		0.74	**Slumdog Millionaire**		0.67
	!	The Revenant		0.73	**Danny Boyle**		0.67
		Argo		0.71	**Daniel Day-Lewis**		0.66
		Brie Larson		0.70	**Brie Larson**		0.65
		The Artist		0.67	George Miller		0.63
PRIMARIES		**Trump**	South Carolina	0.96	**Sanders**	West Virginia	0.96
		Clinton	Iowa	0.90	**Clinton**	North Carolina	0.93
		Trump	Massachusetts	0.88	**Trump**	North Carolina	0.91
		Trump	Tennessee	0.88	**Sanders**	Wyoming	0.90
		Sanders	Maine	0.87	**Sanders**	Oklahoma	0.89
		Sanders	Alaska	0.87	**Sanders**	Hawaii	0.86
	!	Trump	Maine	0.87	Sanders	Arizona	0.86
		Sanders	Wyoming	0.86	**Sanders**	Maine	0.85
		Trump	Louisiana	0.86	**Trump**	Delaware	0.84
	!	Clinton	Indiana	0.85	**Trump**	West Virginia	0.83

Table 10: Top 10 predictions of winners for Oscars and primaries based on veridicality and sentiment scores. Correct predictions are highlighted. "!" indicates a loss which wasn't expected.

Contender	$\lvert T_c \rvert$	$\lvert T_O \rvert$
Cate Blanchett	73	46
Amy Adams	6	113
Sandra Bullock	22	97
Judi Dench	2	117
Meryl Streep	16	103

Table 11: Positive veridicality tweet counts for the Best Actress category in 2014: $|T_c|$ is the count of positive veridicality tweets for the contender under consideration and $|T_O|$ is the count of positive veridicality tweets for the other contenders.

of the match are decisive, and even professionals have a difficult time predicting tennis winners.

Table 10 shows the 10 top predictions made by the veridicality and sentiment-based systems on two of the events we considered - the Oscars and the presidential primaries, highlighting correct predictions.

4.5 Surprise Outcomes

In addition to providing a general method for forecasting contest outcomes, our approach based on veridicality allows us to perform several novel analyses including retrospectively identifying surprise outcomes that were unexpected according to popular beliefs.

In Table 10, we see that the veridicality-based approach incorrectly predicts *The Revenant* as winning Best Film in 2016. This makes sense, because the film was widely expected to win at the time, according to popular belief. Numerous sources in the press,[15,16,17] qualify *The Revenant* not winning an Oscar as a big surprise.

Similarly, for the primaries, the two incorrect predictions made by the veridicality-based approach were surprise losses. News articles[18,19,20] indeed reported the loss of Maine for Trump and the loss of Indiana for Clinton as unexpected.

4.6 Assessing the Reliability of Accounts

Another nice feature of our approach based on veridicality is that it immediately provides an intuitive assessment on the reliability of individual Twitter accounts' predictions. For a given account, we can collect tweets about past contests, and extract those which exhibit positive veridicality toward the outcome, then simply count how often

[15] www.forbes.com/sites/zackomalleygreenburg/2016/02/29/spotlight-best-picture-oscar-is-surprise-of-the-night/#52f546c2721a

[16] www.vox.com/2016/2/26/11115788/revenant-best-picture

[17] www.mirror.co.uk/tv/tv-news/spotlight-wins-best-picture-2016-7460633

[18] http://patch.com/us/across-america/maine-republican-caucus-live-results-trump-favored-win-0

[19] http://www.huffingtonpost.com/entry/ted-cruz-upset-win-maine-republican-caucus_us_56db461ee4b0ffe6f8e9a865

[20] https://news.vice.com/article/bernie-sanders-wins-indiana-primary-in-surprise-upset-over-hillary-clinton

User account	Accuracy(%)	User account	Accuracy(%)
User 1	100 (out of 6)	twitreporting	100 (out of 3)
Cr7Prince4ever	100 (out of 6)	User 3	100 (out of 3)
goal_ghana	100 (out of 4)	Naijawhatsup	100 (out of 3)
User 2	100 (out of 4)	1Mrfutball	90 (out of 10)
breakingnewsnig	100 (out of 4)	User 4	77 (out of 13)

Table 12: List of users sorted by how accurate they were in their Ballon d'Or predictions.

the accounts were correct in their predictions.

As proof of concept, we retrieved within our dataset, the user names of accounts whose tweets about Ballon d'Or contests were classified as having positive veridicality. Table 12 gives accounts that made the largest number of correct predictions for Ballon d'Or awards between 2010 to 2016, sorted by users' prediction accuracy. Usernames of non-public figures are anonymized (as user 1, etc.) in the table. We did not extract more data for these users: we only look at the data we had already retrieved. Some users might not make predictions for all contests, which span 7 years.

Accounts like "goal_ghana", "breakingnewsnig" and "1Mrfutball", which are automatically identified by our analysis, are known to post tweets predominantly about soccer.

5 Conclusions

In this paper, we presented TwiVer, a veridicality classifier for tweets which is able to ascertain the degree of veridicality toward future contests. We showed that veridical statements on Twitter provide a strong predictive signal for winners on different types of events, and that our veridicality-based approach outperforms a sentiment and frequency baseline for predicting winners. Furthermore, our approach is able to retrospectively identify surprise outcomes. We also showed how our approach enables an intuitive yet novel method for evaluating the reliability of information sources.

Acknowledgments

We thank our anonymous reviewers for their valuable feedback. We also thank the Clippers group and Wei Xu at The Ohio State University for their useful suggestions. This material is based upon work supported by the National Science Foundation under Grants No. IIS-1464128 to Alan Ritter and IIS-1464252 to Marie-Catherine de Marneffe. Alan Ritter is supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center in addition to the Office of the Director of National Intelligence (ODNI) and the Intelligence Advanced Research Projects Activity (IARPA) via the Air Force Research Laboratory (AFRL) contract number FA8750-16-C-0114. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, AFRL, NSF, or the U.S. Government.

References

Vikas Ganjigunte Ashok, Song Feng, and Yejin Choi. 2013. Success with style: Using writing style to predict the success of novels. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Hyunyoung Choi and Hal Varian. 2012. Predicting the present with Google Trends. *Economic Record*, 88(1):2–9.

Glen Coppersmith, Mark Dredze, Craig Harman, and Kristy Hollingshead. 2015. From ADHD to SAD: Analyzing the language of mental health on Twitter through self-reported diagnoses. In *Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*. North American Chapter of the Association for Computational Linguistics.

Glen Coppersmith, Craig Harman, and Mark Dredze. 2014. Measuring post traumatic stress disorder in Twitter. In *International Conference on Weblogs and Social Media*.

Kaiser Fung. 2014. Google flu trends failure shows good data > big data. In *Harvard Business Review/HBR Blog Network[Online]*.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in Twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers-Volume 2*.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of NAACL-HLT*.

Mahesh Joshi, Dipanjan Das, Kevin Gimpel, and Noah A Smith. 2010. Movie reviews and revenues:

An experiment in text regression. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proceedings of EMNLP*.

Alexander Konovalov, Benjamin Strauss, Alan Ritter, and Brendan O'Connor. 2017. Learning to extract events from knowledge base revisions. In *Proceedings of WWW*.

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*.

Marie-Catherine de Marneffe, Christopher D Manning, and Christopher Potts. 2012. Did it happen? The pragmatic complexity of veridicality assessment. *Computational linguistics*, 38(2):301–333.

Gilad Mishne and Natalie S Glance. 2006. Predicting movie sales from blogger sentiment. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises*.

Vlad Niculae, Srijan Kumar, Jordan Boyd-Graber, and Cristian Danescu-Niculescu-Mizil. 2015. Linguistic harbingers of betrayal: A case study on an online strategy game. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*.

Michael J Paul, Mark Dredze, and David Broniatowski. 2014. Twitter improves influenza forecasting. *PLOS Currents Outbreaks*, 6.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Lei Shi, Neeraj Agarwal, Ankur Agarwal, Rahul Garg, and Jacob Spoelstra. 2012. Predicting US primary elections with Twitter. In *Social Network and Social Media Analysis: Methods, Models and Applications, NIPS*.

Shiladitya Sinha, Chris Dyer, Kevin Gimpel, and Noah A. Smith. 2013. Predicting the NFL using Twitter. In *Proceedings of ECML/PKDD Workshop on Machine Learning and Data Mining for Sports Analytics*.

Noah A. Smith. 2010. *Text-driven forecasting*. http://www.cs.cmu.edu/ nasmith/papers/smith.whitepaper10.pdf.

Anders Søgaard, Barbara Plank, and Hector Martinez Alonso. 2015. Using frame semantics for knowledge extraction from Twitter. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Sandeep Soni, Tanushree Mitra, Eric Gilbert, and Jacob Eisenstein. 2014. Modeling factuality judgments in social media text. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

James Surowiecki. 2005. *The wisdom of crowds*. Anchor Books, New York, NY.

Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner, and Isabell M. Welpe. 2010. Predicting elections with Twitter: What 140 characters reveal about political sentiment. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*.

Dani Yogatama, Michael Heilman, Brendan O'Connor, Chris Dyer, Bryan R Routledge, and Noah A Smith. 2011. Predicting a scientific community's response to an article. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*.

A Question Answering Approach to Emotion Cause Extraction

Lin Gui[a,b], **Jiannan Hu**[a], **Yulan He**[c], **Ruifeng Xu**[a,d]†, **Qin Lu**[e], **Jiachen Du**[a]
[a]Shenzhen Graduate School, Harbin Institute of Technology, China
[b]College of Mathematics and Computer Science, Fuzhou University, China
[c]School of Engineering and Applied Science, Aston University, United Kingdom
[d]Guangdong Provincial Engineering Technology Research Center for Data Science, China
[e]Department of Computing, the Hong Kong Polytechnic University, Hong Kong
guilin.nlp@gmail.com, hujiannan0526@gmail.com,
y.he9@aston.ac.uk, xuruifeng@hit.edu.cn,
csluqin@comp.polyu.edu.hk, dujiachen@stmail.hitsz.edu.cn

Abstract

Emotion cause extraction aims to identify the reasons behind a certain emotion expressed in text. It is a much more difficult task compared to emotion classification. Inspired by recent advances in using deep memory networks for question answering (QA), we propose a new approach which considers emotion cause identification as a reading comprehension task in QA. Inspired by convolutional neural networks, we propose a new mechanism to store relevant context in different memory slots to model context information. Our proposed approach can extract both word level sequence features and lexical features. Performance evaluation shows that our method achieves the state-of-the-art performance on a recently released emotion cause dataset, outperforming a number of competitive baselines by at least 3.01% in F-measure.

1 Introduction

With the rapid growth of social network platforms, more and more people tend to share their experiences and emotions online. Emotion analysis of online text becomes a new challenge in Natural Language Processing (NLP). In recent years, studies in emotion analysis largely focus on emotion classification including detection of writers' emotions (Gao et al., 2013) as well as readers' emotions (Chang et al., 2015). There are also some information extraction tasks defined in emotion analysis (Chen et al., 2016; Balahur et al., 2011), such as extracting the feeler of an emotion (Das and Bandyopadhyay, 2010). These methods assume that emotion expressions are already observed. Sometimes, however, we care more about the stimuli, or the cause of an emotion. For instance, Samsung wants to know why people love or hate Note 7 rather than the distribution of different emotions.

Ex.1 我的手机昨天丢了，我现在很难过。

Ex.1 *Because I lost my phone yesterday, I feel sad now.*

In an example shown above, "sad" is an emotion word, and the cause of "sad" is "I lost my phone". The emotion cause extraction task aims to identify the reason behind an emotion expression. It is a more difficult task compared to emotion classification since it requires a deep understanding of the text that conveys an emotions.

Existing approaches to emotion cause extraction mostly rely on methods typically used in information extraction, such as rule based template matching, sequence labeling and classification based methods. Most of them use linguistic rules or lexicon features, but do not consider the semantic information and ignore the relation between the emotion word and emotion cause.

In this paper, we present a new method for emotion cause extraction. We consider emotion cause extraction as a question answering (QA) task. Given a text containing the description of an event which may or may not cause a certain emotion, we take an emotion word in context, such as "sad", as a query. The question to the QA system is: "Does the described event cause the emotion of sadness?". The expected answer is either "yes" or "no". (see Figure 1). We build our QA system based on a deep memory network. The memory network has two inputs: a piece of text, referred to as a story in QA systems, and a query. The story is represented using a sequence of word embeddings.

A recurrent structure is implemented to mine the deep relation between a query and a text. It

†Corresponding Author: xuruifeng@hit.edu.cn

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1593–1602
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

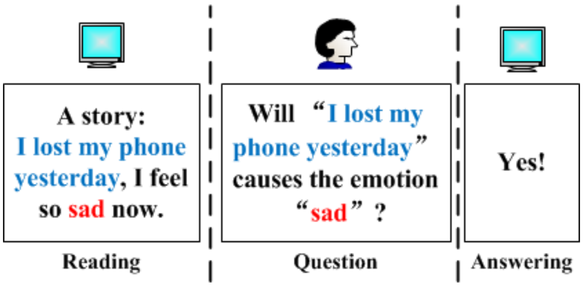

A story:
I lost my phone yesterday, I feel so sad now.
Reading
Will "I lost my phone yesterday" causes the emotion "sad" ?
Question
Yes!
Answering

Figure 1: An example of emotion cause extraction based on the QA framework.

measures the importance of each word in the text by an attention mechanism. Based on the learned attention result, the network maps the text into a low dimensional vector space. This vector is then used to generate an answer. Existing memory network based approaches to QA use weighted sum of attentions to jointly consider short text segments stored in memory. However, they do not explicitly model sequential information in the context. In this paper, we propose a new deep memory network architecture to model the context of each word simultaneously by multiple memory slots which capture sequential information using convolutional operations (Kim, 2014), and achieves the state-of-the-art performance compared to existing methods which use manual rules, common sense knowledge bases or other machine learning models.

The rest of the paper is organized as follows. Section 2 gives a review of related works on emotion analysis. Section 3 presents our proposed deep memory network based model for emotion cause extraction. Section 4 discusses evaluation results. Finally, Section 5 concludes the work and outlines the future directions.

2 Related Work

Identifying emotion categories in text is one of the key tasks in NLP (Liu, 2015). Going one step further, emotion cause extraction can reveal important information about what causes a certain emotion and why there is an emotion change. In this section, we introduce related work on emotion analysis including emotion cause extraction.

In emotion analysis, we first need to determine the taxonomy of emotions. Researchers have proposed a list of primary emotions (Plutchik, 1980; Ekman, 1984; Turner, 2000). In this study, we adopt Ekman's emotion classification scheme (Ekman, 1984), which identifies six primary emotions, namely *happiness, sadness, fear, anger, disgust* and *surprise*, known as the "Big6" scheme in the W3C Emotion Markup Language. This emotion classification scheme is agreed upon by most previous works in Chinese emotion analysis.

Existing work in emotion analysis mostly focuses on emotion classification (Li et al., 2013; Zhou et al., 2016) and emotion information extraction (Balahur et al., 2013). Xu et al. (2012) used a coarse to fine method to classify emotions in Chinese blogs. Gao et al. (2013) proposed a joint model to co-train a polarity classifier and an emotion classifier. Beck et al. (2014) proposed a Multi-task Gaussian-process based method for emotion classification. Chang et al. (2015) used linguistic templates to predict reader's emotions. Das and Bandyopadhyay (2010) used an unsupervised method to extract emotion feelers from Bengali blogs. There are other studies which focused on joint learning of sentiments (Luo et al., 2015; Mohtarami et al., 2013) or emotions in tweets or blogs (Quan and Ren, 2009; Liu et al., 2013; Hasegawa et al., 2013; Qadir and Riloff, 2014; Ou et al., 2014), and emotion lexicon construction (Mohammad and Turney, 2013; Yang et al., 2014; Staiano and Guerini, 2014). However, the aforementioned work all focused on analysis of emotion expressions rather than emotion causes.

Lee et al. (2010) first proposed a task on emotion cause extraction. They manually constructed a corpus from the Academia Sinica Balanced Chinese Corpus. Based on this corpus, Chen et al. (2010) proposed a rule based method to detect emotion causes based on manually define linguistic rules. Some studies (Gui et al., 2014; Li and Xu, 2014; Gao et al., 2015) extended the rule based method to informal text in Weibo text (Chinese tweets).

Other than rule based methods, Russo et al. (2011) proposed a crowdsourcing method to construct a common-sense knowledge base which is related to emotion causes. But it is challenging to extend the common-sense knowledge base automatically. Ghazi et al. (2015) used Conditional Random Fields (CRFs) to extract emotion causes. However, it requires emotion cause and emotion keywords to be in the same sentence. More recently, Gui et al. (2016) proposed a multi-kernel based method to extract emotion causes through learning from a manually annotated emotion cause dataset.

Most existing work does not consider the relation between an emotion word and the cause of such an emotion, or they simply use the emotion word as a feature in their model learning. Since emotion cause extraction requires an understanding of a given piece of text in order to correctly identify the relation between the description of an event which causes an emotion and the expression of that emotion, it can essentially be considered as a QA task. In our work, we choose the memory network, which is designed to model the relation between a story and a query for QA systems (Weston et al., 2014; Sukhbaatar et al., 2015). Apart from its application in QA, memory network has also achieved great successes in other NLP tasks, such as machine translation (Luong et al., 2015), sentiment analysis (Tang et al., 2016) or summarization (M. Rush et al., 2015). To the best of our knowledge, this is the first work which uses memory network for emotion cause extraction.

3 Our Approach

In this section, we will first define our task. Then, a brief introduction of memory network will be given, including its basic learning structure of memory network and deep architecture. Last, our modified deep memory network for emotion cause extraction will be presented.

3.1 Task Definition

The formal definition of emotion cause extraction is given in (Gui et al., 2016). In this task, a given document, which is a passage about an emotion event, contains an emotion word E and the cause of the event. The document is manually segmented in the clause level. For each clause $c = \{w_1, w_2, ...w_k\}$ consisting of k words, the goal is to identify which clause contains the emotion cause. For data representation, we can map each word into a low dimensional embedding space, a.k.a word vector (Mikolov et al., 2013). All the word vectors are stacked in a word embedding matrix $L \in \mathbb{R}^{d \times \|V\|}$, where d is the dimension of word vector and V is the vocabulary size.

For example, the sentence, "I lost my phone yesterday, I feel so sad now." shown in Figure 1, consists of two clauses. The first clause contains the emotion cause while the second clause expresses the emotion of sadness. Current methods to emotion cause extraction cannot handle complex sentence structures where the expression of an emotion and its cause are not adjacent. We envision that the memory network can better model the relation between a emotion word and its emotion causes in such complex sentence structures. In our approach, we only select the clause with the highest probability to be the emotion cause in each document.

3.2 Memory Network

We first present a basic memory network model for emotion cause extraction (shown in Figure 2). Given a clause $c = \{w_1, w_2, ..., w_k\}$, and an emotion word, we first obtain the emotion word's representation in an embedding space, denoted by E. For the clause, let the embedding representations of the words be denoted by $e_1, e_2, ..., e_k$. Here, both e_i and E are defined in $\mathbb{R}^d$. Then, we use the inner product to evaluate the correlation between each word i in a clause and the emotion word, denoted as m_i:

$$m_i = e_i \cdot E. \tag{1}$$

We then normalize the value of m_i to $[0, 1]$ using a softmax function, denoted by α_i as:

$$\alpha_i = \frac{\exp(m_i)}{\sum_{j=1}^{k} \exp(m_j)}, \tag{2}$$

where k is the length of the clause. k also serves as the size of the memory. Obviously, $\alpha_i \in [0, 1]$ and $\sum_{i=1}^{k} \alpha_i = 1$. α_i can serve as an attention weight to measure the importance of each word in our model.

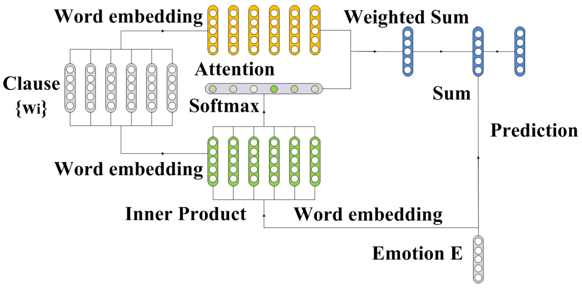


Figure 2: A single layer memory network.

Then, a sum over the word embedding e_i, weighted by the attention vector form the output of the memory network for the prediction of o:

$$o = \sum_{i=1}^{k} e_i \cdot \alpha_i + E. \tag{3}$$

The final prediction is an output from a softmax function, denoted as $\hat{o}$:

$$\hat{o} = \text{softmax}\left(W^T o\right). \quad (4)$$

Usually, W is a $d \times d$ weight matrix and T is the transposition. Since the answer in our task is a simple “yes” or “no”, we use a $d \times 1$ matrix for W. As the distance between a clause and an emotion words is a very important feature according to (Gui et al., 2016), we simply add this distance into the softmax function as an additional feature in our work.

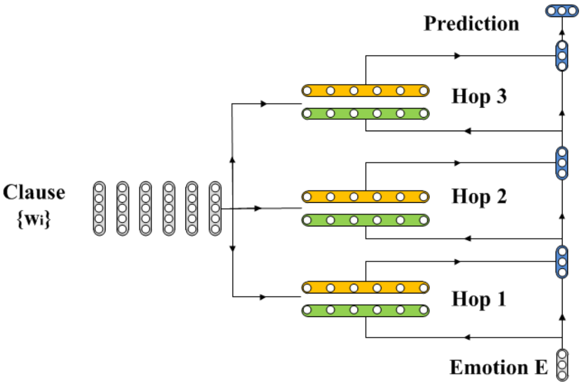


Figure 3: Deep memory network with three computational layers (hops).

The basic model can be extended to deep architecture consisting of multiple layers to handle L hop operations. The network is stacked as follows:

- For hop 1, the query is E and the prediction vector is o_1;
- For hop i, the query is the prediction vector of the previous hop and the prediction vector is o_i;
- The output vector is at the top of the network. It is a softmax function on the prediction vector from hop L: $\hat{o} = \text{softmax}\left(W^T o_L\right)$.

The illustration of a deep memory network with three layers is shown in Figure 3. Since a memory network models the emotion cause at a fine-grained level, each word has a corresponding weight to measure its importance in this task. Comparing to previous approaches in emotion cause extraction which are mostly based on manually defined rules or linguistic features, a memory network is a more principled way to identify the emotion cause from text. However, the basic memory network model does not capture the sequential information in context which is important in emotion cause extraction.

3.3 Convolutional Multiple-Slot Deep Memory Network

It is often the case that the meaning of a word is determined by its context, such as the previous word and the following word. Also, negations and emotion transitions are context sensitive. However, the memory network described in Section 3.2 has only one memory slot with size $d \times k$ to represent a clause, where d is the dimension of a word embedding and k is the length of a clause. It means that when the memory network models a clause, it only considers each word separately.

In order to capture context information for clauses, we propose a new architecture which contains more memory slot to model the context with a convolutional operation. The basic architecture of Convolutional Multiple-Slot Memory Network (in short: ConvMS-Memnet) is shown in Figure 4.

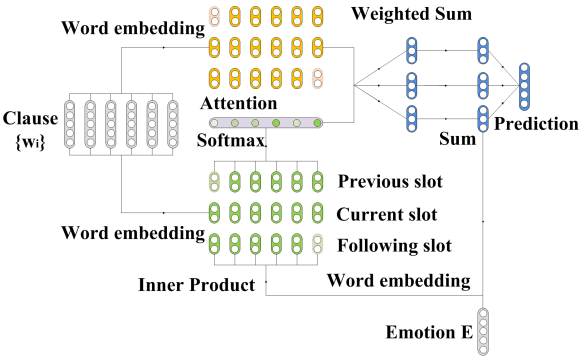


Figure 4: A single layer ConvMS-Memnet.

Considering the text length is usually short in the dataset used here for emotion cause extraction, we set the size of the convolutional kernel to 3. That is, the weight of word w_i in the i-th position considers both the previous word w_{i-1} and the following word w_{i+1} by a convolutional operation:

$$m'_i = \sum_{j=1}^{3} e_{i-2+j} \cdot E \quad (5)$$

For the first and the last word in a clause, we use zero padding, $w_0 = w_{k+1} = \vec{0}$, where k is the length of a clause. Then, the attention weight for each word position in the clause is now defined as:

$$\alpha_i' = \frac{\exp(m_i')}{\sum_{j=1}^{k} \exp\left(m_j'\right)} \tag{6}$$

Note that we obtain the attention for each position rather than each word. It means that the corresponding attention for the i-th word in the previous convolutional slot should be α_{i+1}. Hence, there are three prediction output vectors, namely, $o_{previous}$, $o_{current}$, $o_{following}$:

$$o_{previous} = \sum_{i=1}^{k} e_{i-1} \cdot \alpha_i' + E \tag{7}$$

$$o_{current} = \sum_{i=1}^{k} e_{i} \cdot \alpha_i' + E \tag{8}$$

$$o_{following} = \sum_{i=1}^{k} e_{i+1} \cdot \alpha_i' + E \tag{9}$$

At last, we concatenate the three vectors as $o = o_{previous} \bigoplus o_{current} \bigoplus o_{following}$ for the prediction by a softmax function:

$$\hat{o} = \text{softmax}\left(W_m^T o\right) \tag{10}$$

Here, the size of W_m is $(3 \cdot d) \times d$. Since the prediction vector is a concatenation of three outputs. We implement a concatenation operation rather than averaging or other operations because the parameters in different memory slots can be updated by back propagation. The concatenation of three output vectors forms a sequence-level feature which can be used in the training. Such a feature is important especially when the size of annotated training data is small.

For deep architecture with multiple layer training, the network is more complex (shown in Figure 5).

- For the first layer, the query is an embedding of the emotion word, E.

- In the next layer, there are three input queries since the previous layer has three outputs: $o^1_{previous}$, $o^1_{current}$, $o^1_{following}$. So, for the j-th layer ($j \neq 1$), we need to re-define the weight function (5) as:

$$m_i' = e_{i-1} \cdot o^{j-1}_{previous} + e_i \cdot o^{j-1}_{current} + e_{i+1} \cdot o^{j-1}_{following} \tag{11}$$

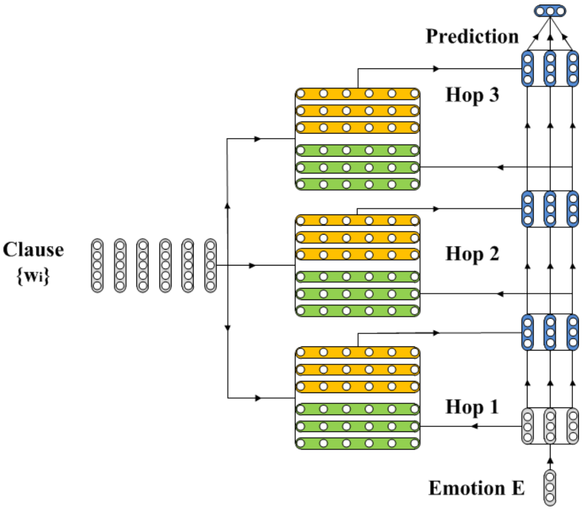


Figure 5: ConvMS-Memnet with three computational layers (hops).

- In the last layer, the concatenation of the three prediction vectors form the final prediction vector to generate the answer.

For model training, we use stochastic gradient descent and back propagation to optimize the loss function. Word embeddings are learned using a skip-gram model. The size of the word embedding is 20 since the vocabulary size in our dataset is small. The dropout is set to 0.4.

4 Experiments and Evaluation

We first presents the experimental settings and then report the results in this section.

4.1 Experimental Setup and Dataset

We conduct experiments on a simplified Chinese emotion cause corpus (Gui et al., 2016)[*], the only publicly available dataset on this task to the best of our knowledge. The corpus contains 2,105 documents from SINA city news[†]. Each document has only one emotion word and one or more emotion causes. The documents are segmented into clauses manually. The main task is to identify which clause contains the emotion cause.

Details of the corpus are shown in Table 1. The metrics we used in evaluation follows Lee et al. (2010). It is commonly accepted so that we can compare our results with others. If a proposed emotion cause clause covers the annotated answer, the word sequence is considered correct. The precision, recall, and F-measure are defined by

[*] Available at: http://hlt.hitsz.edu.cn/?page id=694

[†] http://news.sina.com.cn/society/

Item	Number
Documents	2,105
Clauses	11,799
Emotion Causes	2,167
Documents with 1 emotion	2,046
Documents with 2 emotions	56
Documents with 3 emotions	3

Table 1: Details of the dataset.

$$
\begin{aligned}
P &= \frac{\sum \text{correct causes } 1}{\sum \text{proposed causes } 1}, \\
R &= \frac{\sum \text{correct causes } 1}{\sum \text{annotated causes } 1}, \\
F &= \frac{2 \times P \times R}{P + R}.
\end{aligned}
$$

In the experiments, we randomly select 90% of the dataset as training data and 10% as testing data. In order to obtain statistically credible results, we evaluate our method and baseline methods 25 times with different train/test splits.

4.2 Evaluation and Comparison

We compare with the following baseline methods:

- **RB** (Rule based method): The rule based method proposed in (Lee et al., 2010).
- **CB** (Common-sense based method): This is the knowledge based method proposed by (Russo et al., 2011). We use the Chinese Emotion Cognition Lexicon (Xu et al., 2013) as the common-sense knowledge base. The lexicon contains more than 5,000 kinds of emotion stimulation and their corresponding reflection words.
- **RB+CB+ML** (Machine learning method trained from rule-based features and facts from a common-sense knowledge base): This methods was previously proposed for emotion cause classification in (Chen et al., 2010). It takes rules and facts in a knowledge base as features for classifier training. We train a SVM using features extracted from the rules defined in (Lee et al., 2010) and the Chinese Emotion Cognition Lexicon (Xu et al., 2013).
- **SVM**: This is a SVM classifier using the unigram, bigram and trigram features. It is a baseline previously used in (Li and Xu, 2014; Gui et al., 2016)

Method	P	R	F
RB	**0.6747**	0.4287	0.5243
CB	0.2672	**0.7130**	0.3887
RB+CB	0.5435	0.5307	0.5370
RB+CB+ML	0.5921	0.5307	0.5597
SVM	0.4200	0.4375	0.4285
Word2vec	0.4301	0.4233	0.4136
CNN	0.6215	0.5944	0.6076
Multi-kernel	0.6588	0.6927	**0.6752**
Memnet	0.5922	0.6354	0.6131
ConvMS-Memnet	**0.7076**	**0.6838**	**0.6955**

Table 2: Comparison with existing methods.

- **Word2vec**: This is a SVM classifier using word representations learned by Word2vec (Mikolov et al., 2013) as features.
- **Multi-kernel**: This is the state-of-the-art method using the multi-kernel method (Gui et al., 2016) to identify the emotion cause. We use the best performance reported in their paper.
- **CNN**: The convolutional neural network for sentence classification (Kim, 2014).
- **Memnet**: The deep memory network described in Section 3.2. Word embeddings are pre-trained by skip-grams. The number of hops is set to 3.
- **ConvMS-Memnet**: The convolutional multiple-slot deep memory network we proposed in Section 3.3. Word embeddings are pre-trained by skip-grams. The number of hops is 3 in our experiments.

Table 2 shows the evaluation results. The rule based RB gives fairly high precision but with low recall. CB, the common-sense based method, achieves the highest recall. Yet, its precision is the worst. RB+CB, the combination of RB and CB gives higher the F-measure But, the improvement of 1.27% is only marginal compared to RB.

For machine learning methods, RB+CB+ML uses both rules and common-sense knowledge as features to train a machine learning classifier. It achieves F-measure of 0.5597, outperforming RB+CB. Both SVM and word2vec are word feature based methods and they have similar performance. For word2vec, even though word representations are obtained from the SINA news raw corpus, it still performs worse than SVM trained using n-gram features only. The multi-kernel method (Gui et al., 2016) is the best performer

Word Embedding	P	R	F
Pre-trained	**0.7076**	**0.6838**	**0.6955**
Randomly initialized	0.6786	0.6608	0.6696

Table 3: Comparison of using pre-trained or randomly initialized word embedding.

Method	P	R	F
Hop 1	0.6597	0.6444	0.6520
Hop 2	0.6877	0.6718	0.6796
Hop 3	**0.7076**	**0.6838**	**0.6955**
Hop 4	0.6882	0.6722	0.6801
Hop 5	0.6763	0.6606	0.6683
Hop 6	0.6664	0.6509	0.6585
Hop 7	0.6483	0.6333	0.6407
Hop 8	0.6261	0.6116	0.6187
Hop 9	0.6161	0.6109	0.6089

Table 4: Performance with different number of hops in ConvMS-Memnet.

among the baselines because it considers context information in a structured way. It models text by its syntactic tree and also considers an emotion lexicon. Their work shows that the structure information is important for the emotion cause extraction task.

Naively applying the original deep memory network or convolutional network for emotion cause extraction outperforms all the baselines except the convolutional multi-kernel method. However, using our proposed ConvMS-Memnet architecture, we manage to boost the performance by 11.54% in precision, 4.84% in recall and 8.24% in F-measure respectively when compared to Memnet. The improvement is very significant with p-value less than 0.01 in t-test. The ConvMS-Memnet also outperforms the previous best-performing method, multi-kernel, by 3.01% in F-measure. It shows that by effectively capturing context information, ConvMS-Memnet is able to identify the emotion cause better compared to other methods.

4.3 More Insights into the ConvMS-Memnet

To gain better insights into our proposed ConvMS-Memnet, we conduct further experiments to understand the impact on performance by using: 1) pre-trained or randomly initialized word embedding; 2) multiple hops; 3) attention visualizations; 4) more training epochs.

4.3.1 Pre-trained Word Embeddings

In our ConvMS-Memnet, we use pre-trained word embedding as the input. The embedding maps each word into a lower dimensional real-value vector as its representation. Words sharing similar meanings should have similar representations. It enables our model to deal with synonyms more effectively.

The question is, "can we train the network without using pre-trained word embeddings?". We initialize word vectors randomly, and use an embedding matrix to update the word vectors in the training of the network simultaneously. Comparison results are shown in Table 3. It can be observed that pre-trained word embedding gives 2.59% higher F-measure compared to random initialization. This is partly due to the limited size of our training data. Hence using word embedding trained from other much larger corpus gives better results.

4.3.2 Multiple Hops

It is widely acknowledged that computational models using deep architecture with multiple layers have better ability to learn data representations with multiple levels of abstractions. In this section, we evaluate the power of multiple hops in this task. We set the number of hops from 1 to 9 with 1 standing for the simplest single layer network shown in Figure 4. The more hops are stacked, the more complicated the model is. Results are shown in Table 4. The single layer network has achieved a competitive performance. With the increasing number of hops, the performance improves. However, when the number of hops is larger than 3, the performance decreases due to overfitting. Since the dataset for this task is small, more parameters will lead to overfitting. As such, we choose 3 hops in our final model since it gives the best performance in our experiments.

4.3.3 Word-Level Attention Weights

Essentially, memory network aims to measure the weight of each word in the clause with respect to the emotion word. The question is, will the model really focus on the words which describe the emotion cause? We choose one example to show the attention results in Table 5:

Ex.2 家人/*family* 的/*'s* 坚持/*insistence* 更/*more* 让/*makes* 人/*people* 感动/*touched*

In this example, the cause of the emotion "touched" is "insistence". We show in Table 5 the distribution of word-level attention weights in different hops of memory network training. We can observe that in the first two hops, the highest attention weights centered on the word "more". However, from the third hop onwards, the highest atten-

previous slot	**current slot**	following slot	**Hop 1**	**Hop 2**	**Hop 3**	**Hop 4**	**Hop 5**
家人/family	的/’s	坚持/insisting	0.1298	0.3165	0.1781	0.2947	0.1472
的/’s	坚持/insistence	更/more	0.1706	0.2619	**0.7346**	**0.6412**	**0.8373**
坚持/insisting	更/more	让/makes	**0.5090**	**0.3070**	0.0720	0.0553	0.0145
更/more	让/makes	人/people	0.0327	0.0139	0.0001	0.0001	0.0000
让/makes	人/people	感动/touched	0.1579	0.0965	0.0145	0.0080	0.0008

Table 5: The distribution of attention in different hops.

Method	**P**	**R**	**F**
Memnet	0.5688	0.5588	0.5635
ConvMS-Memnet	**0.6250**	**0.6140**	**0.6195**

Table 6: Comparison of word level emotion cause extraction.

tion weight moves to the word sub-sequence centred on the word “insistence”. This shows that our model is effective in identifying the most important keyword relating to the emotion cause. Also, better results are obtained using deep memory network trained with at least 3 hops. This is consistent with what we observed in Section 4.3.2.

In order to evaluate the quality of keywords extracted by memory networks, we define a new metric on the keyword level of emotion cause extraction. The keyword is defined as the word which obtains the highest attention weight in the identified clause. If the keywords extracted by our algorithm is located within the boundary of annotation, it is treated as correct. Thus, we can obtain the precision, recall, and F-measure by comparing the proposed keywords with the correct keywords by:

$$P = \frac{\sum_{\text{correct keywords}} 1}{\sum_{\text{proposed keywords}} 1},$$

$$R = \frac{\sum_{\text{correct keywords}} 1}{\sum_{\text{annotated keywords}} 1},$$

$$F = \frac{2 \times P \times R}{P + R}.$$

Since the reference methods do not focus on the keywords level, we only compare the performance of Memnet and ConvMS-Memnet in Table 6. It can be observed that our proposed ConvMS-Memnet outperforms Memnet by 5.6% in F-measure. It shows that by capturing context features, ConvMS-Memnet is able to identify the word level emotion cause better compare to Memnet.

4.3.4 Training Epochs

In our model, the training epochs are set to 20. In this section, we examine the testing error using a case study. Due to the page length limit, we only choose one example from the corpus. The text below has four clauses:

Ex.3 *45*天，对于失去儿子的他们是多么的漫长，宝贝回家了，这个春节是多么幸福。

Ex.3 *45 days, it is long time for the parents who lost their baby. If the baby comes back home, they would become so happy in this Spring Festival.*

In this example, the cause of emotion “happy” is described in the third clause.

We show in Table 7 the probability of each clause containing an emotion cause in different training epochs. It is interesting to see that our model is able to detect the correct clause with only 5 epochs. With the increasing number of training epochs, the probability associated with the correct clause increases further while the probabilities of incorrect clauses decrease generally.

4.4 Limitations

We have shown in Section 4.3.4 a simple example consisting of only four clauses from which our model can identify the clause containing the emotion cause correctly. We notice that for some complex text passages which contain long distance dependency relations, negations or emotion transitions, our model may have a difficulty in detecting the correct clause containing the emotion causes. It is a challenging task to properly model the discourse relations among clauses. In the future, we will explore different network architecture with consideration of various discourse relations possibly through transfer learning of larger annotated data available for other tasks.

Another shortcoming of our model is that, the answer generated from our model is simply “yes” or “no”. The main reason is that the size of the annotated corpus is too small to train a model which can output natural language answers in full sentences. Ideally, we would like to develop a model which can directly give the cause of an emotion

Clause	5 Epochs	10 Epochs	15 Epochs	20 Epochs
45 Days	0.0018	0.0002	0.0000	0.0000
it is ... baby	0.3546	0.6778	0.5457	0.3254
If the ... back home	**0.7627**	**0.7946**	**0.8092**	**0.9626**
they ... Spring Festival	0.2060	0.0217	0.0004	0.0006

Table 7: The probability of a clause containing the emotion cause in different iterations in the multiple-slot memory network.

expressed in text. However, since the manual annotation of data is too expensive for this task, we need to explore feasible ways to automatically collect annotate data for emotion cause detection. We also need to study effective evaluation mechanisms for such QA systems.

5 Conclusions

In this work, we treat emotion cause extraction as a QA task and propose a new model based on deep memory networks for identifying the emotion causes for an emotion expressed in text. The key property of this approach is the use of context information in the learning process which is ignored in the original memory network. Our new memory network architecture is able to store context in different memory slots to capture context information in proper sequence by convolutional operation. Our model achieves the state-of-the-art performance on a dataset for emotion cause detection when compared to a number of competitive baselines. In the future, we will explore effective ways to model discourse relations among clauses and develop a QA system which can directly output the cause of emotions as answers.

Acknowledgments

This work was supported by the National Natural Science Foundation of China 61370165, U1636103, 61632011, 61528302, National 863 Program of China 2015AA015405, Shenzhen Foundational Research Funding JCYJ20150625142543470, JCYJ20170307150024907 and Guangdong Provincial Engineering Technology Research Center for Data Science 2016KF09.

References

Alexandra Balahur, Jesús M. Hermida, Andrés Montoyo, and Rafael Muñoz. 2011. Emotinet: A knowledge base for emotion detection in text built on the appraisal theories. In *Proceedings of International Conference on Applications of Natural Language to Information Systems*, pages 27–39.

Alexandra Balahur, Jesús M. Hermida, and Hristo Tanev. 2013. Detecting implicit emotion expressions from text using ontological resources and lexical learning. *Theory and Applications of Natural Language Processing*, pages 235–255.

Daniel Beck, Trevor Cohn, and Lucia Specia. 2014. Joint emotion analysis via multi-task gaussian processes. In *EMNLP*, pages 1798–1803.

Yung-Chun Chang, Cen-Chieh Chen, Yu-Lun Hsieh, and WL Hsu. 2015. Linguistic template extraction for recognizing reader-emotion and emotional resonance writing assistance. In *ACL*, pages 775–780.

Wei Fan Chen, Mei Hua Chen, Ming Lung Chen, and Lun Wei Ku. 2016. A computer-assistance learning system for emotional wording. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1–1.

Ying Chen, Sophia Yat Mei Lee, Shoushan Li, and Chu-Ren Huang. 2010. Emotion cause detection with linguistic constructions. In *COLING*, pages 179–187.

Dipankar Das and Sivaji Bandyopadhyay. 2010. Finding emotion holder from bengali blog texts—an unsupervised syntactic approach. In *Proceedings of Pacific Asia Conference on Language, Information and Computation*, pages 621–628.

Paul Ekman. 1984. Expression and the nature of emotion. *Approaches to Emotion*, 3:19–344.

Kai Gao, Hua Xu, and Jiushuo Wang. 2015. A rule-based approach to emotion cause detection for chinese micro-blogs. *Expert Systems with Applications*, 42(9):4517–4528.

Wei Gao, Shoushan Li, Sophia Yat Mei Lee, Guodong Zhou, and Chu-Ren Huang. 2013. Joint learning on sentiment and emotion classification. In *CIKM*, pages 1505–1508. ACM.

Diman Ghazi, Diana Inkpen, and Stan Szpakowicz. 2015. Detecting emotion stimuli in emotion-bearing sentences. In *Computational Linguistics and Intelligent Text Processing*, pages 152–165. Springer.

Lin Gui, Dongyin Wu, Ruifeng Xu, Qin Lu, and Yu Zhou. 2016. Event-driven emotion cause extraction with corpus construction. In *EMNLP*, pages 1639–1649.

Lin Gui, Li Yuan, Ruifeng Xu, Bin Liu, Qin Lu, and Yu Zhou. 2014. Emotion cause detection with linguistic construction in chinese weibo text. In *Natural Language Processing and Chinese Computing*, pages 457–464. Springer.

Takayuki Hasegawa, Nobuhiro Kaji, Naoki Yoshinaga, and Masashi Toyoda. 2013. Predicting and eliciting addressee's emotion in online dialogue. In *ACL*, pages 964–972.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Sophia Yat Mei Lee, Ying Chen, and Chu-Ren Huang. 2010. A text-driven rule-based system for emotion cause detection. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 45–53. Association for Computational Linguistics.

Shoushan Li, Lei Huang, Rong Wang, and Guodong Zhou. 2013. Sentence-level emotion classification with label and context dependence. In *ACL*, pages 1045–1053.

Weiyuan Li and Hua Xu. 2014. Text-based emotion classification using emotion cause extraction. *Expert Systems with Applications*, 41(4):1742–1749.

Bing Liu. 2015. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press.

Huanhuan Liu, Shoushan Li, Guodong Zhou, Chu-Ren Huang, and Peifeng Li. 2013. Joint modeling of news reader's and comment writer's emotions. In *ACL*, pages 511–515.

Kun-Hu Luo, Zhi-Hong Deng, Liang-Chen Wei, and Hongliang Yu. 2015. Jeam: A novel model for cross-domain sentiment classification based on emotion analysis. In *EMNLP*, pages 2503–2508.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, pages 1412–1421.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*, pages 379–389.

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

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

Mitra Mohtarami, Man Lan, and Chew Lim Tan. 2013. Probabilistic sense sentiment similarity through hidden emotions. In *ACL*, pages 983–992.

Gaoyan Ou, Wei Chen, Tengjiao Wang, Zhongyu Wei, Binyang Li, Dongqing Yang, and Kam-Fai Wong. 2014. Exploiting community emotion for microblog event detection. In *EMNLP*, pages 1159–1168.

Robert Plutchik. 1980. Emotion: A psychoevolutionary synthesis.

Ashequl Qadir and Ellen Riloff. 2014. Learning emotion indicators from tweets: Hashtags, hashtag patterns, and phrases. In *EMNLP*, pages 1203–1209.

Changqin Quan and Fuji Ren. 2009. Construction of a blog emotion corpus for chinese emotional expression analysis. In *EMNLP*, pages 1446–1454.

Irene Russo, Tommaso Caselli, Francesco Rubino, Ester Boldrini, and Patricio Martínez-Barco. 2011. Emocause: an easy-adaptable approach to emotion cause contexts. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pages 153–160.

Jacopo Staiano and Marco Guerini. 2014. Depechemood: a lexicon for emotion analysis from crowd-annotated news. *arXiv preprint arXiv:1405.1605*.

Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. 2015. End-to-end memory networks. In *NIPS*, pages 2431–2439.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *EMNLP*, pages 214–225.

Jonathan H Turner. 2000. *On the origins of human emotions: A sociological inquiry into the evolution of human affect*. Stanford University Press Stanford, CA.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Jun Xu, Ruifeng Xu, Qin Lu, and Xiaolong Wang. 2012. Coarse-to-fine sentence-level emotion classification based on the intra-sentence features and sentential context. In *CIKM*, pages 2455–2458. ACM.

Ruifeng Xu, Chengtian Zou, Yanzhen Zheng, Xu Jun, Lin Gui, Bin Liu, and Xiaolong Wang. 2013. A new emotion dictionary based on the distinguish of emotion expression and emotion cognition. *Journal of Chinese Information Processing*, 27(6):82–90.

Min Yang, Dingju Zhu, and Kam-Pui Chow. 2014. A topic model for building fine-grained domain-specific emotion lexicon. In *ACL(2)*, pages 421–426.

Deyu Zhou, Xuan Zhang, Yin Zhou, Quan Zhao, and Xin Geng. 2016. Emotion distribution learning from texts. In *EMNLP*, pages 638–647.

Story Comprehension for Predicting What Happens Next

Snigdha Chaturvedi   Haoruo Peng   Dan Roth
University of Illinois, Urbana-Champaign
{snigdha,hpeng7,danr}@illinois.edu

Abstract

Automatic story comprehension is a fundamental challenge in Natural Language Understanding, and can enable computers to learn about social norms, human behavior and commonsense. In this paper, we present a story comprehension model that explores three distinct semantic aspects: (i) the sequence of events described in the story, (ii) its emotional trajectory, and (iii) its plot consistency. We judge the model's understanding of real-world stories by inquiring if, like humans, it can develop an expectation of what will happen next in a given story. Specifically, we use it to predict the correct ending of a given short story from possible alternatives. The model uses a hidden variable to weigh the semantic aspects in the context of the story. Our experiments demonstrate the potential of our approach to characterize these semantic aspects, and the strength of the hidden variable based approach. The model outperforms the state-of-the-art approaches and achieves best results on a publicly available dataset.

1 Introduction

Narratives are a fundamental part of human language and culture. They serve as vehicles to share experiences, information and goals. For these reasons, automatically understanding stories is an interesting but challenging task for Computational Linguists (Mani, 2012). Story comprehension involves not only an array of NLP capabilities, but also some common sense knowledge and an understanding of normative social behavior (Charniak, 1972). Past research has focused on various aspects of story understanding such as identifying character personas (Bamman et al., 2014; Valls-Vargas et al., 2015), interpersonal relationships (Chaturvedi, 2016), plot-patterns (Jockers, 2013), narrative structures (Finlayson, 2012). There has also been an interest in predicting what is expected to happen next in a piece of text (Chambers and Jurafsky, 2008). Human readers are good at *filling-in-the-gaps* or inferring information that is not explicitly stated in the text. However, computers are not yet able to match their performance on predicting what could be the likely next step in a given sequence of events described in a story.

Context: One day Wesley's auntie came over to visit. He was happy to see her, because he liked to play with her. When she started to give his little sister attention, he got jealous. He got angry at his auntie and bit her hand when she wasn't looking.

Incorrect Ending: She gave him a cookie for being so nice.
Correct Ending: He was scolded.

Figure 1: Example from the story-cloze task: predict the correct ending to a given short story out of provided options.

Recently, Mostafazadeh et al. (2016) introduced the story-cloze task for testing this ability, albeit without the aspect of language generation. This task requires choosing the correct ending to a given four sentences long story (also referred to as *context*) from two provided alternatives. Fig. 1 shows an example story consisting of a short context, and two ending options.

In this work we address this story-cloze task. While the short nature and third person narrative style of these stories help us circumvent the problem of speaker identification and processing long

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1603–1614
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

dialogues, the crowdsourced dataset ensures that they reflect real-world and commonsense stories. Our approach emphasizes the joint contribution of multiple aspects to story understanding, which future research can build upon.

In this paper we explore three semantic aspects of story understanding: (i) the sequence of events described in the story, (ii) the evolution of sentiment and emotional trajectories, and (iii) topical consistency. The first aspect is motivated from approaches in semantic script induction, and evaluates if events described in an ending-alternative *are likely* to occur within the sequence of events described in the preceding context. For example, in the story in Fig. 1, Wesley gets angry and bites his sister's hand. So, a next likely step might suggest that he would be scolded. However, there are multiple semantic aspects to story understanding beyond analyzing events and scripts. Stories often describe characters (e.g. Wesley) who need to be viewed as social and emotional agents. They not only describe events involving these characters, but also reflect their social lives and emotional states. Our model captures this by evaluating if the sentiment described in an ending option *makes sense* considering the context of the story. For example, in the story in Fig. 1, the general sentiment of being *scolded* is better aligned with the sentiment of Wesley being *angry* and *jealous*, compared to that of *being nice*. Also, stories generally revolve around coherent themes and topics. Our model accounts for that by analyzing if the topic of an ending option is consistent with the preceding context. We present a log-linear model that is used to weigh the various aspects of the story using a hidden variable. It then uses this hidden variable to predict the correct ending for the given story.

We demonstrate the strength of our approach by comparing it with the existing state-of-the-art methods for this task. We first validate the predictive potential of the features that correspond to the three semantic aspects through a simple classifier trained using these features. We then demonstrate the benefit of using our hidden variable approach by showing that it significantly outperforms the above mentioned classifier and other baselines, and achieves an accuracy of 77.60% on the task. Our key contributions are:

- We model story understanding as a joint model over multiple semantic aspects, and utilize the idea for predicting a story's end.
- We design linguistic features that incorporate world knowledge and narrative awareness.
- We present a hidden variable approach to weigh these aspects in a story's context.
- We empirically demonstrate that our approach significantly outperforms state-of-the-art methods.

2 Predicting Story Ending

Given an L sentences long context, $\mathbf{c} = \langle c_1, c_2, c_3 \ldots c_L \rangle$, and two ending-options, o_1 and o_2, we aim to predict which ending option forms an inconsistent story. This is a binary classification task. We assume that the inconsistency can arise from one (or more) of certain semantic aspects. In this section, we first describe the intuition behind using these aspects and the features that we designed to capture them (Sec. 2.1). We then describe our model which uses a latent variable to weigh these aspects in light of the story, and then predicts its ending (Sec. 2.2).

2.1 Measuring Consistency

Our approach analyzes the following aspects of story understanding: Event-sequence, Sentiment-trajectory, and Topical Consistency.

Event-sequence: For a story, or any piece of text, to be coherent, it needs to describe a meaningful or 'mutually entailing' sequence of events (Chatman, 1980). For instance, in Figure 1 *Wesley got angry* → *Wesley bit her hand* → *Wesley was scolded* describes a more coherent sequence of events, as compared to *Wesley got angry* → *Wesley bit her hand* → *Wesley got a cookie*

Prior work in script-learning attempts to model such *prototypical* sequence of events (usually captured through verbs). For this task, we wanted to model events at an abstraction level that would be generalizable and yet semantically meaningful. Peng and Roth (2016) recently proposed a neural SemLM approach, to model such sequence of events using a language model of FrameNet (Baker et al., 1998) frames that are evoked in the given text. It represents an event using the corresponding predicate frame and its sense, obtained using a Sematic Role Labeler (Punyakanok et al., 2004). It also extends the frame definition to include explicit discourse markers (such as *but*, *and*) since they model relationships between frames. For example, in Figure 1, the SemLM representation for the last sentence of the context is 'Get.01-and-bit.01'. Here, '01' indicates specific predicate senses for verbs 'get' and 'bit' with 'and' being a discourse marker. Also, it produces 'scold.01' and 'give.01' for the correct and incorrect endings respectively. We train this language model using a log bilinear language model (Mnih and Hinton, 2007) on a collection of unannotated short stories (see Sec. 3.1) and also 20 years of New York Times data[1].

Given a sequence of frames evoked in the context, such a trained language model can then be used to get the conditional probabilities of the frame(s) evoked in each of the two ending-options. The option with more probable frame(s) is likely to be the appropriate ending. With this intuition in mind, for each of the two ending-option, o_i, we design features whose values are probabilities of frames evoked in that option (f_{o_i}), given the sequence of frames, $\langle f_1, f_2, \ldots f_D \rangle$, evoked in the context, $\mathbf{c} = \langle c_1, c_2, c_3 \ldots c_L \rangle$. We consider increasingly longer frame-contexts for conditional probability computation, i.e. for each option, o_i, we extract the following features: $P(f_{o_i}|f_D)$, $P(f_{o_i}|f_D f_{D-1})$, $\ldots$ $P(f_{o_i}|f_D f_{D-1} \ldots f_1)$. For each of these features, we additionally also include a comparative binary feature whose value is 1 if the conditional probability of one of the options (o_2) is greater than the corresponding conditional probability of the other option (o_1) (E.g. $P(o_2|f_D) > P(o_1|f_D)$), and -1 otherwise. Our preliminary experiments indicated that these features were helpful for supervised classification.

Sentiment-trajectory: As mentioned before, stories are different from objective texts such as news articles, as they additionally describe sentiments or emotions. Some stories can be categorized as happy stories while others as sad. However, most stories depict evolving sentiments in their plots as they progress (Vonnegut, 1981).

With the goal of modeling such sentiment trajectories, we assumed that a story can be divided into the following narrative-segments: a *beginning*, a *body*, a *climax*, and an *ending*. While this narrative-segmentation process warrants deeper research, in this paper we adopt a simple methodology. We treat the first sentence of the L sentences long context as the *beginning*, the next $L-2$ sentences are treated as the *body*, the last sentence of the context forms the *climax*, and the two options form the (possible) *ending*[2]. We then assigned a positive, negative, or neutral sentiment to each segment, represented as $\mathbb{S}(segment) = \text{sign}($number of positive words - number of negative words$)$ in the segment. The sentiment polarity of a word was determined by a look-up from pre-trained sentiment lexica (Liu et al., 2005; Wilson et al., 2005)[3]. Thus, the L length context can now be viewed as a sequence of its segment's sentiments. Lastly, we learn sentiment trajectories in form of N-gram language models from an unannotated corpus of short stories (Sec. 3.1) that learn: (i) $P(\mathbb{S}(ending)|\mathbb{S}(climax), \mathbb{S}(body), \mathbb{S}(beginning))$; (ii) $P(\mathbb{S}(ending)|\mathbb{S}(climax), \mathbb{S}(body))$; and (iii) $P(\mathbb{S}(ending)|\mathbb{S}(climax))$.

The process described above learns typical sentiment trajectories over narrative-segments. However, it does not model a story's *overall* sentiment (i.e. whether it is a happy or a sad story, *in general*). To capture this notion, we train another language model to learn $P(\mathbb{S}(ending)|\mathbb{S}(context))$, where $\mathbb{S}(context)$ is the sentiment of the full context (without segmentation).

Finally, for each ending option, we extract features whose values are the four conditional probabilities described above. As before we also consider four comparative binary features.

Topical Consistency: This aspect is motivated by the idea that stories are topically cohesive (Bamberg, 2012), and in a typical story, new topics (concepts, entities or ideas) are not introduced towards the end because it does not allow the story-writer enough narrative space and time to develop and describe them (Jovchelovitch and Bauer, 2000). We capture the notion of topic of a sentence using *topic-words* (the nouns and verbs appearing in it (Lapata and Barzilay, 2005)). For each option, we first *align* each of its topic-words with the most similar topic-word in one of the context-sentences, while defining the alignment score as this similarity value. We measure similarity between two words using the cosine similarity of their vector space representations (using

[1] Owing to the large size of the training data and the fact that we abstract to the frame-semantic (and not verb) level, we cover most instances (76%) in our dataset.

[2] The reported segmentation process made sense from qualitative analysis on a random sample, and also led to superior performance compared to alternate strategies.

[3] Polarities of 'negated' word were reversed (determined from *neg* dependency relation in the corresponding sentence).

pretrained GloVe (Pennington et al., 2014) vectors). We then quantify the *topical-closeness* of an ending option with the context using averaged alignment score of its topic-words[4]. For each ending option, we extract one feature whose value is this *topical-closeness* with the context. As before, we also include a binary comparative feature.

2.2 Hidden Coherence Model

Sec. 2.1 described the three semantic consistency aspects and the corresponding features. We now describe our model which uses these features (represented as $\vec{f}_{co}$ in the rest of this paper) to identify the (in)coherent ending-option. The model is also dependent on another feature set, $\vec{\phi}_{co}$, which will be discussed later in this section.

Formally, our model addresses the following binary classification problem: given the multi-sentence context, $\mathbf{c}$, and two ending-options, o_1 and o_2, predict the answer, $a \in \{0, 1\}$. The correct ending for the story is o_1 when $a = 0$ and o_2 otherwise. Our training data consists of instances (context and ending options) labeled with corresponding answers a. It does not contain any other annotation (like semantic consistency aspects).

The model proceeds by assuming that there are K different semantic consistency aspects and that an ending-option can lead to an incoherent story by violating any of these aspects (our implementation uses $K = 3$ corresponding to the three aspects described in Sec. 2.1). The model achieves this by assuming that each instance belongs to a latent category, $z \in \{1, 2, 3 \ldots K\}$, which advises the model on the importance of these aspects for the given instance. Using these definitions and assumptions, the probability of an answer given the context and the ending-options can be modeled as:

$$P(a|\mathbf{c}, o_1, o_2) = \sum_{z}^{K} P(z|\mathbf{c}, o_1, o_2) P(a|z, \mathbf{c}, o_1, o_2)$$

We parameterize $P(z|\mathbf{c}, o_1, o_2)$ as:

$$P(z|\mathbf{c}, o_1, o_2) = \frac{e^{-\vec{\lambda}_z \vec{\phi}_{co}}}{\sum_k e^{-\vec{\lambda}_k \vec{\phi}_{co}}}$$

where, $\vec{\phi}_{co}$ is the feature vector used for assigning a value to the hidden variable for an instance, and $\vec{\lambda}_z$ is the weight vector of the log-linear model for the z^{th} aspect. There are K weight vectors, one corresponding to each of the K aspects.

For predicting the answer, a, we assume that each aspect has a separate logistic-regression based prediction model parameterized as:

$$P(a|z, \mathbf{c}, o_1, o_2) = \frac{(e^{-\vec{w}_z \vec{f}^z_{co}})^{1-a}}{1 + e^{-\vec{w}_z \vec{f}^z_{co}}}$$

where $\vec{f}^z_{co}$ is the feature vector constructed from the context and ending-options for the z^{th} aspect, and $\vec{w}_z$ are the corresponding weights.

Training: The model parameters, $\vec{w}_z$ and $\vec{\lambda}_z$, are learned during the training process by maximizing the log-likelihood of the data. We use Expectation-Maximization (Dempster et al., 1977) for training. During the E-step we compute the expectations for latent variable assignments using parameter values from the previous iteration as:

$$< z_n^k > \propto \frac{e^{-\vec{\lambda}_k \vec{\phi}_{co}}}{\sum_{k'}^{K} e^{-\vec{\lambda}_{k'} \vec{\phi}_{co}}} P(a_n | z_n^k, \mathbf{c}_n, o_{1n}, o_{2n})$$

where, a subscript of n represents the n^{th} training instance out of a total of N instances. z_n^k represents n^{th} instance getting assigned to the k^{th} aspect, and $<>$ denotes expected values.

In the M-step, given the expected assignments, we maximize the following expected log complete likelihood with respect to the model parameters using gradient ascent:

$$\begin{aligned} < L > \quad = \quad & \sum_{n}^{N} \sum_{k}^{K} < z_n^k > \Big(\log \frac{e^{-\vec{\lambda}_k \vec{\phi}_{co}}}{\sum_{k'}^{K} e^{-\vec{\lambda}_{k'} \vec{\phi}_{co}}} \\ & + \quad \log \frac{(e^{-\vec{w}_k \vec{f}^k_{co}})^{1-a_n}}{1 + e^{-\vec{w}_k \vec{f}^k_{co}}} \Big) \end{aligned}$$

Features: Our model uses two types of features: (i) for aspect-specific prediction model, $\vec{f}^k_{co}$, and (ii) for hidden aspect assignment, $\vec{\phi}_{co}$. The features extracted for each of the $K = 3$ aspects, $\vec{f}^k_{co}$, were described in Sec. 2.1. For the hidden aspect assignment, we needed features that could analyze the two options in light of the given context, and characterize the importance of various aspects for the given instance. One way to measure an aspect's importance is by quantifying how different the two options are with respect to that aspect. The underlying assumption is that the option that

[4] An alternative would be to compute similarity between averaged vector representations of the topic-words of the context and the ending-option(s). However, that assumes that a story is strictly about a single topic. Instead they reflect interplay of multiple related and 'narrow topics. E.g. a story describing a teacher walking in rain is about topics like 'teacher', 'walk', 'rain', etc. The correct ending option describes a passer-by helping the teacher. 'passer-by' was far from an average of all topics but close to the 'walk' topic.

leads to an inconsistent story, by compromising on one of the aspects, would differ significantly from the other option in that aspect. We quantify an aspect's importance using the normalized $L1$ distance between the corresponding features, $\vec{f_{co}^k}$, extracted for the two options in Sec. 2.1 (ignoring the comparative binary features, and normalizing by the number of features). Specifically, for an aspect k, lets represent the feature extracted for the two options by $\vec{f_1^k}$ and $\vec{f_2^k}$ (each of length n) then the corresponding 'importance feature' for this aspect $= |\vec{f_1^k} - \vec{f_2^k}|/n$. For example, for topical-consistency, for each option, we extracted 1 feature measuring its *topical-closeness* to the context. For $\vec{\phi}_{co}$ computation we consider the absolute difference between this value for the two options. To summarize, for each instance, we define $\vec{\phi}_{co}$ as a set of $K + 1$ features: K of these measure the importance of each of the aspects, while the last one is an additional always-one feature which captures the context-insensitive bias in the data.

3 Empirical Evaluation

In this section we describe our experiments.

3.1 Dataset

For our experiments, we have used a publicly available collection of commonsense short stories released by Mostafazadeh et al. (2016). It consists of about 100K unannotated five-sentences long stories. For collecting these stories, Amazon Mechanical Turk workers were asked to compose novel five-sentence long stories on everyday topics. They were prompted to write coherent stories with a specific beginning and ending, with *something* happening in between. This resulted in a wide variety in topics with causal and temporal links between the events described in the story. Also, the workers were asked to limit the length of individual sentences to 70 characters which yielded short and succinct sentences, and to not use informal language or quotations.

The dataset also contains an additional set of $3,742$ four-sentences long stories (context) with two ending options, only one of which is correct. Each instance is annotated with this correctness information. This set was collected by asking Amazon Mechanical Turk workers to write a coherent and an incoherent ending to a given short story. The workers were asked to ensure that both the options shared at least one character from the story, and that the options, in isolation, made sense. This resulted in non-trivial alternative endings, and was also validated by other human subjects for high quality. This set was divided by Mostafazadeh et al. (2016) into validation and test sets of 1871 instances each for the Story-Cloze Task, and were used for training and evaluating our model.

3.2 Baselines

We use the following baselines in our experiments:
DSSM: (Mostafazadeh et al., 2016) It trains two deep neural networks (Huang et al., 2013) to project the context and the ending-options into the same vector space. Based on these vector representations, it predicts the ending-option with the largest cosine similarity with the context.
Msap: The task addressed in this paper was also a shared task for an EACL'17 workshop and this baseline (Schwartz et al., 2017) represents the best performance reported on its leaderboard (Mostafazadeh et al., 2017). It trains a logistic regression based on stylistic and language-model based features.
LR: Our next baseline is a simple logistic regression model which is agnostic to the fact that there are multiple types of aspects. Given a context and ending-options, it predicts the answer using the same features (Sec. 2.1) as the Hidden Coherence model but clubs them all into one feature-vector.
Majority Vote: This ensemble method uses the features extracted for each of the $K = 3$ aspects, to train K separate logistic regression models. It then makes a prediction by taking a majority vote of these K classifiers.
Soft Voting: This baseline also learns K different aspect-specific classifiers. However, instead of taking a majority vote, it computes a score for each option, o_i, as $\Pi_k^K P_k(ending = o_i|\mathbf{c}, o_1, o_2)$. Here P_k represents the probability obtained from the k^{th} logistic regression. The final prediction corresponds to the option with greater score.
Aspect-aware Ensemble: Like the voting methods, this baseline also trains K different aspect-specific classifiers. However, it makes the final prediction by training another logistic regression over their predictions.

3.3 Quantitative Results

Table 1 shows accuracies of various models on the held-out test set. An always-one classifier would get 51.3% accuracy on the task and human performance is reported to be 100% (Mostafazadeh

Model	Accuracy
DSSM (Mostafazadeh et al., 2016)	58.5%
Msap (Schwartz et al., 2017)	75.2%
Majority Voting	69.5% *
Aspect aware ensemble	71.5% *
LR	74.4% *
Soft Voting	75.1%
Hidden Coherence Model	77.6% *

Table 1: Test-set accuracies of various models. Our Hidden Coherence Model outperforms competitive baselines and state-of-the-art system.

et al., 2016). A * indicates that the model's accuracy was significantly better than the previous best model in the table (using McNemar's test with $\alpha = 0.1$). We can see that the logistic regression, LR, outperforms the DSSM model indicating the strength of the features extracted for the various story-consistency aspects. Also, the Soft Voting approach gives us slight benefit over the LR model, possibly because of increased expressivity which includes better *organization* of features into groups or aspects. Majority Vote, in spite of sharing a similar classifier structure, does not perform as well. This might happen because it takes a *hard* vote of individual classifiers, which might be detrimental to model performance if one of the classifiers is weak. Our analysis in Sec. 3.4 shows that the topical-consistency features indeed result in a relatively weak classifier. The Aspect aware Ensemble performs better possibly because of its ability to weight the aspects (though not in context of the story).

Lastly, we can see that the proposed Hidden Coherence model, with an accuracy of 77.60%, outperforms all other models. The superior performance of our model indicates the benefit of the context-sensitive weighing of individual consistency aspects.

3.4 Ablation Study

We now investigate the predictive value of the various aspect-specific features. Table 2 shows the performance of a logistic regression model trained using all the features (All) and then using individual feature-groups. We can see that the features extracted from the aspect analyzing the event-sequence have the strongest predictive power, followed by those characterizing Sentiment-trajectory. The features measuring topical consistency result in lowest accuracy but they still perform better than random on the task.

Features	Accuracy
All	74.4%
Event-sequence	71.6%
Sentiment	64.5%
Topic	55.2%

Table 2: Performance comparison of various aspect features. Our event-sequence based features are most helpful followed by Sentiment-trajectory and then Topical Consistency based features.

3.5 Qualitative Results

Table 3 shows example stories, and weights given to the three aspects. An aspect's weight is its contribution towards the predicted output, and is shown as a bar of vertically stacked blocks in the last column. A block's height is proportional to its aspect's weight. Light grey block represents Event-sequence, and dark grey and black blocks represent Sentiment-trajectory and Topical consistency respectively.

The first row describes the story of a man hurting himself. A human reader can guess from commonsense knowledge that people usually recover (correct ending) after being hurt and do not repeat their mistake (incorrect ending). Accordingly, our model also primarily used the aspect analyzing events in this story, which is indicated by the long light grey block in its weight bar. Also, we can see that the topic of both the options is consistent with the story, and the model gave a very small weight to the Topical Consistency aspect indicated by the almost indiscernible black block in its weight bar. Similarly, the second row describes the story of Pam being proud of her yard work. There is a striking sentimental contrast between the two options (*upset* versus *satisfied*), and the model relies primarily on sentiments (dark grey). The last row, describes the story of Maria making candy apples. The incorrect ending introduces a new entity/idea, *apple pie*, resulting in topical incoherence of this option with the rest of the story. The model relies primarily on topic (black) and events (light grey). Reliance on events makes sense because it is likely for a person to *enjoy* what they fondly *cook*. The model gave a weight of 40% to the topical aspect, which is high as compared to its average weight across the dataset.

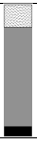
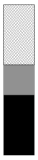
Context	Incorrect Ending	Correct Ending	Weights
He didn't know how the television worked. He tried to fix it, anyway. He climbed up on the roof and fiddled with the antenna. His foot **slipped** on the wet shingles and he went tumbling down.	He decided that was fun and to try tumbling again.	Thankfully, he **recovered**.	
Pam thought her front yard looked boring. So she decided to buy several plants. And she placed them in her front yard. She was **proud** of her work.	Pam was **upset** at herself.	Pam was **satisfied**.	
Maria smelled the fresh Autumn air and decided to celebrate. She wanted to make candy apples. She picked up the ingredients at a local market and headed home. She cooked the candy and prepared the apples.	Maria's apple **pie** was delicious.	She enjoyed the candy apples.	

Table 3: Examples of stories, ending-options, and aspect weights learned by our model. Aspect weights are shown as bars of stacked blocks in the last column (light grey, dark grey and black represent Event-sequence, Sentiment-trajectory and Topical Consistency respectively). A block's height is proportional to its component's weight. Black blocks are sometimes not visible because there were too small.

3.6 Discussion

Error Analysis: Table 4 shows examples of stories for which our model could not predict the correct ending. We believe that many of these stories require a deeper understanding of language and commonsense. For example, in the story described in the first row, the protagonist accepted an invitation from his friends to go to a club but danced terribly, and so he was asked to stay home the next time. To make the correct prediction in this story, the model not only needs to understand that if one does not dance well at a club they are likely to be not invited in the future, but also that staying home is the same as not getting invited. Similarly, the second row shows a story in which Johnny asks Anita out, but she makes an excuse. He later sees her with another guy and decides not to ask her out again. This example requires identifying that Anita's excuse was a lie indicating her disinterest in Johnny, which makes it unlikely for Johnny to invite her again. It also needs an understanding of inter-personal relationships, i.e. seeing a potential lover with another person leads to estrangement.

Social Analysis: To further explore the significance of social relations in stories, we consider the special case of romantic stories. We use a deterministic heuristic to identify romantic stories using lexical matches with a handcrafted list containing words like *marry*, *proposal*, *girlfriend*, *ask out*, etc. We then applied the following two rules: (i) if a story contains two characters, then output the option whose sentiment matches that of the context, (ii) if a story contains three characters, then output the option with negative sentiment. Most stories in our dataset contained few characters. These rules are motivated by the intuition that a romantic story between two people can have a happy or sad ending depending on the context. However, a romantic story with three people is likely to describe a love triangle, and so not end well. Expectedly, these rules had low coverage (of about 60 stories), but a considerably high accuracy (70%) when active. Furthermore, a closer analysis revealed that most errors resulted from incorrect coreference resolutions (leading to incorrect count of characters). This indicates the utility of understanding semantics of social relationships for story comprehension and it could potentially be another aspect to consider while solving such tasks.

Sentiment Analysis: We now explore the insights obtained by modeling sentiments in stories. Mostafazadeh et al. (2016) presented two baselines for this task whose outputs were simply the *ending* whose sentiment agreed with (i) the complete story, or (ii) the *climax* (last sentence of the story). While their performances were close to random, our sentiment based features yield a much higher accuracy of 64.5% (see Table 2). This could possibly be attributed to our approach's ability to learn such rules from the data itself, rather

Context	Incorrect Ending	Correct Ending
My friends all love to go to the club to dance. They think it's a lot of fun and always invite. I finally decided to tag along last Saturday. I danced terribly and broke a friend's toe.	The next weekend, I was asked to please stay home.	My friends decided to keep inviting me as I am so much fun.
Johnny thought Anita was the girl for him, but he was wrong. He invited her out but she said she didn't feel well. Johnny decided to go to a club, just to drink and listen to music. At midnight, he looked back and saw Anita dancing with another guy.	Johnny did not ask Anita out again.	Johnny wanted to ask Anita out again.

Table 4: Examples of stories incorrectly predicted by our model.

than making hard assumptions. For instance, our language model of overall narrative sentiments indicates that while happy stories mostly have happy endings (with a conditional probability of 74%), the reverse is not true. In particular, sad stories (with overall negative sentiments) end with a negative sentiment in only 52% of the cases. We made similar observations regarding sentimental conformity between endings and climaxes.

Our features' superior performance can also be attributed to their deeper understanding of not just overall sentiments but also their trajectories. Our language models indicate that stories that exhibit a positive sentiment in all three narrative segments (beginning, body, and climax) have very high chance of happy endings (83%). Similarly, stories with negative sentiments in the three segments also have a fair chance of having sad endings (60%). This is different from stories with an overall negative sentiment, in which case the sentiment may be exhibited in only certain narrative segments. The language models also identify a pattern of *hopeful* stories, in which the sentiment begins as negative but moves towards positive in the body and climax, resulting in mostly happy endings ($\sim$ 70%). This was not true for the reverse case: *pessimistic* stories with positive beginning but negative body (and/or climax) were equally likely to have positive or negative endings (52%). Supplementary material contains sample stories for each of the above observations.

4 Related Work

We now review previous work done in this field. Our work touches upon several research areas.

4.1 Story understanding:

Our work is most closely related to the field of narrative understanding. Apart from event-centric understanding of narrative plots (Lehnert, 1981; McIntyre and Lapata, 2010; Goyal et al., 2010; Elsner, 2012; Finlayson, 2012), recent methods have focused on understanding narratives from the perspective of characters (Wilensky, 1978) mentioned in them. These methods study character personas (Bamman et al., 2013, 2014) or Proppian (Propp, 1968) roles (Valls-Vargas et al., 2014, 2015), inter-character relationships (Iyyer et al., 2016; Chaturvedi et al., 2016, 2017), and social networks of characters (Elson et al., 2010; Elson, 2012; Agarwal et al., 2013, 2014; Krishnan and Eisenstein, 2015; Srivastava et al., 2016).

4.2 Events-centered learning:

Our Entity-sequence component is closely related to semantic script learning. Script learning focuses on representing text using a prototypical sequences of events, their participants and causal relationships between them, called scripts (Schank and Abelson, 1977; Mooney and DeJong, 1985). Several statistical methods have been proposed to automatically learn scripts or scripts-like structures from unstructured text (Chambers and Jurafsky, 2008, 2009; Jans et al., 2012; Orr et al., 2014; Pichotta and Mooney, 2014). Such methods for script-learning also include Bayesian approaches (Bejan, 2008; Frermann et al., 2014), sequence alignment algorithms (Regneri et al., 2010) and neural networks (Modi and Titov, 2014; Granroth-Wilding and Clark, 2016; Pichotta and Mooney, 2016). There has also been work on representing events in a structured manner using schemas, which are learned probabilistically (Chambers, 2013; Cheung et al., 2013;

Nguyen et al., 2015), using graphs (Balasubramanian et al., 2013) or neural approaches (Titov and Khoddam, 2015). Recently, Ferraro and Durme (2016) presented a unified Bayesian model for scripts and frames.

4.3 Textual Coherence:

Our work is also related to the study of coherence in discourse. A significant amount of prior work is primarily based on the Centering Theory Framework (Grosz et al., 1995) and focus on entities and their syntactic roles (Karamanis, 2003; Karamanis et al., 2004; Lapata and Barzilay, 2005; Barzilay and Lapata, 2008; Elsner and Charniak, 2008). Other approaches measure coherence using topic drift within a domain (Barzilay and Lee, 2004; Fung and Ngai, 2006), co-occurrence of words (Lapata, 2003; Soricut and Marcu, 2006), syntactic patterns (Louis and Nenkova, 2012) and discourse relations (Pitler and Nenkova, 2008; Lin et al., 2011). The nature of the tasks addressed by these works (such as determining the correct arrangement order for a set of sentences) makes them focus on learning sequential order of the various discourse components (entities, ideas, etc.). Our goal, instead, is to choose between alternatives of discourse components themselves (and not just their order) to produce a consistent story.

5 Conclusion

Story comprehension is a complex Natural Language Understanding task involving linguistic intelligence as well as a semantic and social knowledge of the real world. This paper studies story comprehension from the perspective of learning what is likely to happen next in a story. We present a model that given a short story, predicts its correct ending. It incorporates three aspects of story-understanding, that are based on an analysis of the events, sentiments and topics described in the story. While this is the best-performing model till date on this task, our analysis indicates a need for even deeper analysis of human behavior and societal norms to further improve our understanding. This work emphasizes that there are multiple aspects to story understanding, which future research can build upon.

Acknowledgement

This work is partly supported by the IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM Cognitive Horizon Network; by the US Defense Advanced Research Projects Agency (DARPA) under contract FA8750-13-2-0008; and by the Army Research Laboratory (ARL) under agreement W911NF-09-2-0053. The views expressed are those of the authors and do not reflect the official policy or position of IBM, the Department of Defense or the U.S. Government.

References

Apoorv Agarwal, Sriramkumar Balasubramanian, Anup Kotalwar, Jiehan Zheng, and Owen Rambow. 2014. Frame semantic tree kernels for social network extraction from text. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014*, pages 211–219.

Apoorv Agarwal, Anup Kotalwar, and Owen Rambow. 2013. Automatic extraction of social networks from literary text: A case study on alice in wonderland. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing, IJCNLP 2013*, pages 1202–1208.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, COLING-ACL 1998*, pages 86–90.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*, pages 1721–1731.

Michael Bamberg. 2012. Narrative analysis. In D. L. Long A. T. Panter D. Rindskopf H. Cooper, P. M. Camic and K. Sher, editors, *APA handbook of research methods in psychology*, volume 2, pages 85–102. American Psychological Association.

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning Latent Personas of Film Characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013*, pages 352–361.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A Bayesian Mixed Effects Model of Literary Character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, pages 370–379.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2004*, pages 113–120.

Cosmin Adrian Bejan. 2008. Unsupervised discovery of event scenarios from texts. In *Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference*, pages 124–129.

Nathanael Chambers. 2013. Event Schema Induction with a Probabilistic Entity-Driven Model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013*, pages 1797–1807.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised Learning of Narrative Schemas and their Participants. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, ACL 2009*, pages 602–610.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, ACL 2008*, pages 789–797.

Eugene Charniak. 1972. Toward a model of children's story comprehension. Technical report, Massachusetts Institute of Technology.

Seymour Chatman. 1980. *Story and Discourse*. Cornell University Press.

Snigdha Chaturvedi. 2016. *Structured Approaches for Exploring Interpersonal Relationships in Natural Language Text*. Ph.D. thesis, University of Maryland, College Park.

Snigdha Chaturvedi, Mohit Iyyer, and Hal Daumé III. 2017. Unsupervised Learning of Evolving Relationships Between Literary Characters. In *Proceedings of the Thirty First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 3159–3165.

Snigdha Chaturvedi, Shashank Srivastava, Hal Daumé III, and Chris Dyer. 2016. Modeling evolving relationships between characters in literary novels. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, AAAI'16, pages 2704–2710.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic Frame Induction. In *Proceedings of the Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 837–846.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38.

Micha Elsner. 2012. Character-based Kernels for Novelistic Plot Structure. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics EACL 2012*, pages 634–644.

Micha Elsner and Eugene Charniak. 2008. Coreference-inspired coherence modeling. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, ACL 2008*, pages 41–44.

David K. Elson. 2012. *Modeling Narrative Discourse*. Ph.D. thesis, Columbia University.

David K. Elson, Nicholas Dames, and Kathleen McKeown. 2010. Extracting Social Networks from Literary Fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL 2010*, pages 138–147.

Francis Ferraro and Benjamin Van Durme. 2016. A unified bayesian model of scripts, frames and language. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2601–2607.

Mark Alan Finlayson. 2012. *Learning Narrative Structure from Annotated Folktales*. Ph.D. thesis, Massachusetts Institute of Technology.

Lea Frermann, Ivan Titov, and Manfred Pinkal. 2014. A hierarchical bayesian model for unsupervised induction of script knowledge. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014*, pages 49–57.

Pascale Fung and Grace Ngai. 2006. One story, one flow: Hidden markov story models for multilingual multidocument summarization. *TSLP*, 3(2):1–16.

Amit Goyal, Ellen Riloff, and Hal Daumé III. 2010. Automatically Producing Plot Unit Representations for Narrative Text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010*, pages 77–86.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2727–2733.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, CIKM 2013*, pages 2333–2338.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan L. Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1534–1544.

Bram Jans, Steven Bethard, Ivan Vulic, and Marie-Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2012*, pages 336–344.

Matthew L. Jockers. 2013. *Macroanalysis: Digital Methods and Literary History*. Topics in the Digital Humanities. University of Illinois Press.

Sandra Jovchelovitch and Martin W. Bauer. 2000. Narrative interviewing. In Martin W. Bauer and G. Gaskell, editors, *Qualitative Researching With Text, IMAge and Sound : a Practical Handbook*, pages 57–74. SAGE, London, UK.

Nikiforos Karamanis. 2003. *Entity Coherence for Descriptive Text Structuring*. Ph.D. thesis, Division of Informatics, University of Edinburgh.

Nikiforos Karamanis, Massimo Poesio, Chris Mellish, and Jon Oberlander. 2004. Evaluating centering-based metrics of coherence. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, ACL 2004*, pages 391–398.

Vinodh Krishnan and Jacob Eisenstein. 2015. "You're Mr. Lebowski, I'm the Dude": Inducing Address Term Formality in Signed Social Networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2015*, pages 1616–1626.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, ACL 2003*, pages 545–552.

Mirella Lapata and Regina Barzilay. 2005. Automatic evaluation of text coherence: Models and representations. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI 2005*, pages 1085–1090.

Wendy G. Lehnert. 1981. Plot Units and Narrative Summarization. *Cognitive Science*, 5(4):293–331.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2011. Automatically evaluating text coherence using discourse relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 997–1006.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion Observer: Analyzing and Comparing Opinions on the Web. In *Proceedings of the 14th international conference on World Wide Web, WWW 2005*, pages 342–351.

Annie Louis and Ani Nenkova. 2012. A coherence model based on syntactic patterns. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012*, pages 1157–1168.

Inderjeet Mani. 2012. *Computational Modeling of Narrative*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Neil McIntyre and Mirella Lapata. 2010. Plot Induction and Evolutionary Search for Story Generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL 2010*, pages 1562–1572.

A. Mnih and G. Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning, ICML 2007*, pages 641–648.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning, CoNLL 2014*, pages 49–57.

Raymond J. Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence, IJCAI 1985*, pages 681–687.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies NAACL HLT 2016*, pages 839–849.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. Lsdsem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51.

Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. 2015. Generative event schema induction with entity disambiguation. In

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, pages 188–197.

John Walker Orr, Prasad Tadepalli, Janardhan Rao Doppa, Xiaoli Fern, and Thomas G. Dietterich. 2014. Learning Scripts as Hidden Markov Models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1565–1571.

Haoruo Peng and Dan Roth. 2016. Two discourse driven language models for semantics. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, pages 290–300.

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

Karl Pichotta and Raymond J. Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014*, pages 220–229.

Karl Pichotta and Raymond J. Mooney. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2800–2806.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008*, pages 186–195.

Vladimir Iakovlevich Propp. 1968. *Morphology of the Folktale*. University of Texas.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th International Conference on Computational Linguistics, COLING 2004*.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning Script Knowledge with Web Experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL 2010*, pages 979–988.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, Plans, Goals, and Understanding: An Inquiry Into Human Knowledge Structures (Artificial Intelligence Series)*, 1 edition. Psychology Press.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, and Noah A. Smith. 2017. Story cloze task: UW NLP system. In *Proceedings of LS-DSem*, pages 52–55.

Radu Soricut and Daniel Marcu. 2006. Discourse generation using utility-trained coherence models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, ACL 2006*, pages 1105–1112.

Shashank Srivastava, Snigdha Chaturvedi, and Tom M. Mitchell. 2016. Inferring interpersonal relations in narrative summaries. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2807–2813.

Ivan Titov and Ehsan Khoddam. 2015. Unsupervised induction of semantic roles within a reconstruction-error minimization framework. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2015*, pages 1–10.

Josep Valls-Vargas, Jichen Zhu, and Santiago Ontañón. 2014. Toward automatic role identification in unannotated folk tales. In *Proceedings of the Tenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2014*, pages 188–194.

Josep Valls-Vargas, Jichen Zhu, and Santiago Ontañón. 2015. Narrative hermeneutic circle: Improving character role identification from natural language text via feedback loops. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015*, pages 2517–2523.

Kurt Vonnegut. 1981. *Palm Sunday*. RosettaBooks LLC, New York.

Robert Wilensky. 1978. *Understanding Goal-Based Stories*. Outstanding Dissertations in the Computer Sciences. Garland Publishing, New York.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, HLT/EMNLP 2005*, pages 347–354.

Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm

Bjarke Felbo[1], **Alan Mislove**[2], **Anders Søgaard**[3], **Iyad Rahwan**[1], **Sune Lehmann**[4]

[1]Media Lab, Massachusetts Institute of Technology
[2]College of Computer and Information Science, Northeastern University
[3]Department of Computer Science, University of Copenhagen
[4]DTU Compute, Technical University of Denmark

Abstract

NLP tasks are often limited by scarcity of manually annotated data. In social media sentiment analysis and related tasks, researchers have therefore used binarized emoticons and specific hashtags as forms of distant supervision. Our paper shows that by extending the distant supervision to a more diverse set of noisy labels, the models can learn richer representations. Through emoji prediction on a dataset of 1246 million tweets containing one of 64 common emojis we obtain state-of-the-art performance on 8 benchmark datasets within emotion, sentiment and sarcasm detection using a single pretrained model. Our analyses confirm that the diversity of our emotional labels yield a performance improvement over previous distant supervision approaches.

1 Introduction

A variety of NLP tasks are limited by scarcity of manually annotated data. Therefore, co-occurring emotional expressions have been used for distant supervision in social media sentiment analysis and related tasks to make the models learn useful text representations before modeling these tasks directly. For instance, the state-of-the-art approaches within sentiment analysis of social media data use positive/negative emoticons for training their models (Deriu et al., 2016; Tang et al., 2014). Similarly, hashtags such as #anger, #joy, #happytweet, #ugh, #yuck and #fml have in previous research been mapped into emotional categories for emotion analysis (Mohammad, 2012).

Distant supervision on noisy labels often enables a model to obtain better performance on the target task. In this paper, we show that extending the distant supervision to a more diverse set of noisy labels enables the models to learn richer representations of emotional content in text, thereby obtaining better performance on benchmarks for detecting sentiment, emotions and sarcasm. We show that the learned representation of a single pretrained model generalizes across 5 domains.

Table 1: Example sentences scored by our model. For each text the top five most likely emojis are shown with the model's probability estimates.

I love mom's cooking	😋 49.1%	😍 8.8%	❤ 3.1%	☺️ 3.0%	♡ 2.9%
I love how you never reply back..	😒 14.0%	😑 8.3%	😠 6.3%	😐 5.4%	💔 5.1%
I love cruising with my homies	😎 34.0%	👌 6.6%	✌ 5.7%	😌 4.1%	💯 3.8%
I love messing with yo mind!!	😜 17.2%	😈 11.8%	😏 8.0%	😉 6.4%	🙊 5.3%
I love you and now you're just gone..	💔 39.1%	😔 11.0%	😞 7.3%	😪 5.3%	😢 4.5%
This is shit	😠 7.0%	😡 6.4%	😞 6.0%	😒 6.0%	😤 5.8%
This is the shit	🎧 10.9%	🎶 9.7%	👌 6.5%	😎 5.7%	😏 4.8%

Emojis are not always a direct labeling of emotional content. For instance, a positive emoji may serve to disambiguate an ambiguous sentence or to complement an otherwise relatively negative text. Kunneman et al. (2014) discuss a similar duality in the use of emotional hashtags such as *#nice* and *#lame*. Nevertheless, our work shows that emojis can be used to classify the emotional content of texts accurately in many cases. For instance, our DeepMoji model captures varied usages of the word 'love' as well as slang such as 'this is the shit' being a positive statement (see Table 1). We provide an online demo at deepmoji.mit.edu to allow others to explore the predictions of our model.

Contributions We show how millions of readily available emoji occurrences on Twitter can be used to pretrain models to learn a richer emotional

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1615–1625
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

representation than traditionally obtained through distant supervision. We transfer this knowledge to the target tasks using a new layer-wise fine-tuning method, obtaining improvements over the state-of-the-art within a range of tasks: emotion, sarcasm and sentiment detection. We present multiple analyses on the effect of pretraining, including results that show that the diversity of our emoji set is important for the transfer learning potential of our model. Our pretrained DeepMoji model is released with the hope that other researchers can use it for various NLP tasks[1].

2 Related work

Using emotional expressions as noisy labels in text to counter scarcity of labels is not a new idea (Read, 2005; Go et al., 2009). Originally, binarized emoticons were used as noisy labels, but later also hashtags and emojis have been used. To our knowledge, previous research has always manually specified which emotional category each emotional expression belong to. Prior work has used theories of emotion such as Ekman's six basic emotions and Plutchik's eight basic emotions (Mohammad, 2012; Suttles and Ide, 2013).

Such manual categorization requires an understanding of the emotional content of each expression, which is difficult and time-consuming for sophisticated combinations of emotional content. Moreover, any manual selection and categorization is prone to misinterpretations and may omit important details regarding usage. In contrast, our approach requires no prior knowledge of the corpus and can capture diverse usage of 64 types of emojis (see Table 1 for examples and Figure 3 for how the model implicitly groups emojis).

Another way of automatically interpreting the emotional content of an emoji is to learn emoji embeddings from the words describing the emoji-semantics in official emoji tables (Eisner et al., 2016). This approach, in our context, suffers from two severe limitations: a) It requires emojis at test time while there are many domains with limited or no usage of emojis. b) The tables do not capture the dynamics of emoji usage, i.e., drift in an emoji's intended meaning over time.

Knowledge can be transferred from the emoji dataset to the target task in many different ways. In particular, multitask learning with simultaneous training on multiple datasets has shown promising results (Collobert and Weston, 2008). However, multitask learning requires access to the emoji dataset whenever the classifier needs to be tuned for a new target task. Requiring access to the dataset is problematic in terms of violating data access regulations. There are also issues from a data storage perspective as the dataset used for this research contains hundreds of millions of tweets (see Table 2). Instead we use transfer learning (Bengio et al., 2012) as described in §3.3, which does not require access to the original dataset, but only the pretrained classifier.

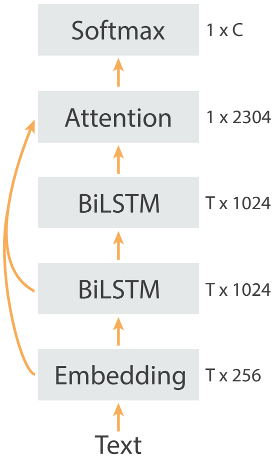


Figure 1: Illustration of the DeepMoji model with S being text length and C the number of classes.

3 Method

3.1 Pretraining

In many cases, emojis serve as a proxy for the emotional contents of a text. Therefore, pretraining on the classification task of predicting which emoji were initially part of a text can improve performance on the target task (see §5.3 for an analysis of why our pretraining helps). Social media contains large amounts of short texts with emojis that can be utilized as noisy labels for pretraining. Here, we use data from Twitter from January 1st 2013 to June 1st 2017, but any dataset with emoji occurrences could be used.

Only English tweets without URL's are used for the pretraining dataset. Our hypothesis is that the content obtained from the URL is likely to be important for understanding the emotional content of the text in the tweet. Therefore, we expect emojis associated with these tweets to be noiser labels

[1] Available with preprocessing code, examples of usage, benchmark datasets etc. at github.com/bfelbo/DeepMoji

than for tweets without URLs, and the tweets with URLs are thus removed.

Proper tokenization is important for generalization. All tweets are tokenized on a word-by-word basis. Words with 2 or more repeated characters are shortened to the same token (e.g. 'loool' and 'looooool' are tokenized such that they are treated the same). Similarly, we use a special token for all URLs (only relevant for benchmark datasets), user mentions (e.g. '@acl2017' and '@emnlp2017' are thus treated the same) and numbers. To be included in the training set the tweet must contain at least 1 token that is not a punctuation symbol, emoji or special token[2].

Many tweets contain multiple repetitions of the same emoji or multiple different emojis. In the training data, we address this in the following way. For each unique emoji type, we save a separate tweet for the pretraining with that emoji type as the label. We only save a single tweet for the pretraining per unique emoji type regardless of the number of emojis associated with the tweet. This data preprocessing allows the pretraining task to capture that multiple types of emotional content are associated with the tweet while making our pretraining task a single-label classification instead of a more complicated multi-label classification.

To ensure that the pretraining encourages the models to learn a rich understanding of emotional content in text rather than only emotional content associated with the most used emojis, we create a balanced pretraining dataset. The pretraining data is split into a training, validation and test set, where the validation and test set is randomly sampled in such a way that each emoji is equally represented. The remaining data is upsampled to create a balanced training dataset.

3.2 Model

With the millions of emoji occurrences available, we can train very expressive classifiers with limited risk of overfitting. We use a variant of the Long Short-Term Memory (LSTM) model that has been successful at many NLP tasks (Hochreiter and Schmidhuber, 1997; Sutskever et al., 2014). Our DeepMoji model uses an embedding layer of 256 dimensions to project each word into a vector space. A hyperbolic tangent activation function is used to enforce a constraint of each embedding dimension being within $[-1, 1]$. To capture the context of each word we use two bidirectional LSTM layers with 1024 hidden units in each (512 in each direction). Finally, an attention layer that take all of these layers as input using skip-connections is used (see Figure 1 for an illustration).

The attention mechanism lets the model decide the importance of each word for the prediction task by weighing them when constructing the representation of the text. For instance, a word such as 'amazing' is likely to be very informative of the emotional meaning of a text and it should thus be treated accordingly. We use a simple approach inspired by (Bahdanau et al., 2014; Yang et al., 2016) with a single parameter pr. input channel:

$$\begin{aligned} e_t &= h_t w_a \\ a_t &= \frac{exp(e_t)}{\sum_{i=1}^{T} exp(e_i)} \\ v &= \sum_{i=1}^{T} a_i h_i \end{aligned}$$

Here h_t is the representation of the word at time step t and w_a is the weight matrix for the attention layer. The attention importance scores for each time step, a_t, are obtained by multiplying the representations with the weight matrix and then normalizing to construct a probability distribution over the words. Lastly, the representation vector for the text, v, is found by a weighted summation over all the time steps using the attention importance scores as weights. This representation vector obtained from the attention layer is a high-level encoding of the entire text, which is used as input to the final Softmax layer for classification. We find that adding the attention mechanism and skip-connections improves the model's capabilities for transfer learning (see §5.2 for more details).

The only regularization used for the pretraining task is a L2 regularization of $1E{-}6$ on the embedding weights. For the finetuning additional regularization is applied (see §4.2). Our model is implemented using Theano (Theano Development Team, 2016) and we make an easy-to-use version available that uses Keras (Chollet et al., 2015).

3.3 Transfer learning

Our pretrained model can be fine-tuned to the target task in multiple ways with some approaches 'freezing' layers by disabling parameters updates to prevent overfitting. One common approach is

[2] Details available at github.com/bfelbo/deepmoji

to use the network as a feature extractor (Donahue et al., 2014), where all layers in the model are frozen when fine-tuning on the target task except the last layer (hereafter referred to as the *'last'* approach). Alternatively, another common approach is to use the pretrained model as an initialization (Erhan et al., 2010), where the full model is unfrozen (hereafter referred to as *'full'*).

We propose a new simple transfer learning approach, *'chain-thaw'*, that sequentially unfreezes and fine-tunes a single layer at a time. This approach increases accuracy on the target task at the expense of extra computational power needed for the fine-tuning. By training each layer separately the model is able to adjust the individual patterns across the network with a reduced risk of overfitting. The sequential fine-tuning seems to have a regularizing effect similar to what has been examined with layer-wise training in the context of unsupervised learning (Erhan et al., 2010).

More specifically, the chain-thaw approach first fine-tunes any new layers (often only a Softmax layer) to the target task until convergence on a validation set. Then the approach fine-tunes each layer individually starting from the first layer in the network. Lastly, the entire model is trained with all layers. Each time the model converges as measured on the validation set, the weights are reloaded to the best setting, thereby preventing overfitting in a similar manner to early stopping (Sjöberg and Ljung, 1995). This process is illustrated in Figure 2. Note how only performing step a) in the figure is identical to the 'last' approach, where the existing network is used as a feature extractor. Similarly, only doing step d) is identical to the 'full' approach, where the pretrained weights are used as an initialization for a fully trainable network. Although the chain-thaw procedure may seem extensive it is easily implemented with only a few lines of code. Similarly, the additional time spent on fine-tuning is limited when the target task uses GPUs on small datasets of manually annotated data as is often the case.

A benefit of the chain-thaw approach is the ability to expand the vocabulary to new domains with little risk of overfitting. For a given dataset up to 10000 new words from the training set are added to the vocabulary. §5.3 contains analysis on the added word coverage gained from this approach.

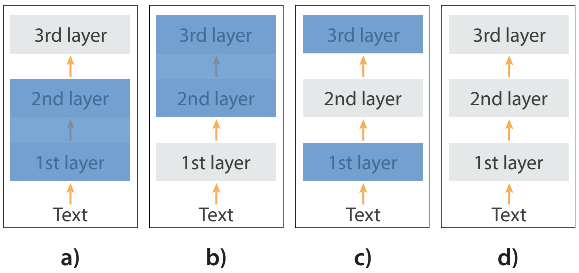


Figure 2: Illustration of the chain-thaw transfer learning approach, where each layer is fine-tuned separately. Layers covered with a blue rectangle are frozen. Step a) tunes any new layers, b) then tunes the 1st layer and c) the next layer until all layers have been fine-tuned individually. Lastly, in step d) all layers are fine-tuned together.

Table 2: The number of tweets in the pretraining dataset associated with each emoji in millions.

😂	❤️	😍	😭	😩	😊	😘	😒	👌	💕	♥️	☺️	😏	😁	😔	🙌
233.7	82.2	79.5	78.1	60.8	54.7	54.6	51.7	50.5	44.0	39.5	39.1	34.8	34.4	32.1	28.1
🙏	😳	💯	👍	✌️	👏	🙈	😌	😴	😉	😎	💁	🎶	💖	💜	😅
24.8	23.4	21.6	21.0	20.5	20.3	19.9	19.6	18.9	17.5	17.0	16.9	16.1	15.3	15.2	15.0
👀	💙	😑	😋	😢	😫	🙊	😕	😄	😜	✋	😞	😡	😐	✨	💀
14.9	14.3	14.2	14.2	12.9	12.4	12.0	12.0	11.7	11.7	11.3	11.2	11.1	11.0	11.0	10.8
😈	👊	😪	💪	🔫	😤	💔	♡	😷	😣	😓	😖	🙅	😬	🎧	😠
10.2	9.6	9.5	9.3	9.2	8.9	8.7	8.6	8.1	6.3	6.0	5.7	5.6	5.5	5.4	5.1

4 Experiments

4.1 Emoji prediction

We use a raw dataset of 56.6 billion tweets, which is then filtered to 1.2 billion relevant tweets (see details in §3.1). In the pretraining dataset a copy of a single tweet is stored once for each unique emoji, resulting in a dataset consisting of 1.6 billion tweets. Table 2 shows the distribution of tweets across different emoji types. To evaluate performance on the pretraining task a validation set and a test set both containing 640K tweets (10K of each emoji type) are used. The remaining tweets are used for the training set, which is balanced using upsampling.

The performance of the DeepMoji model is evaluated on the pretraining task with the results shown in Table 3. Both top 1 and top 5 accuracy is used for the evaluation as the emoji labels are noisy with multiple emojis being potentially correct for any given sentence. For comparison we also train a version of our DeepMoji model with smaller LSTM layers and a bag-of-words classifier, fastText, that has recently shown competitive results (Joulin et al., 2016). We use 256 dimen-

Table 3: Accuracy of classifiers on the emoji prediction task. d refers to the dimensionality of each LSTM layer. Parameters are in millions.

	Params	Top 1	Top 5
Random	–	1.6%	7.8%
fasttext	12.8	12.8%	36.2%
DeepMoji (d = 512)	15.5	16.7%	43.3%
DeepMoji (d = 1024)	22.4	17.0%	43.8%

sions for this fastText classifier, thereby making it almost identical to only using the embedding layer from the DeepMoji model. The difference in top 5 accuracy between the fastText classifier (36.2%) and the largest DeepMoji model (43.8%) underlines the difficulty of the emoji prediction task. As the two classifiers only differ in that the DeepMoji model has LSTM layers and an attention layer between the embedding and Softmax layer, this difference in accuracy demonstrates the importance of capturing the context of each word.

4.2 Benchmarking

We benchmark our method on 3 different NLP tasks using 8 datasets across 5 domains. To make for a fair comparison, we compare DeepMoji to other methods that also utilize external data sources in addition to the benchmark dataset. An averaged F1-measure across classes is used for evaluation in emotion analysis and sarcasm detection as these consist of unbalanced datasets while sentiment datasets are evaluated using accuracy.

An issue with many of the benchmark datasets is data scarcity, which is particularly problematic within emotion analysis. Many recent papers proposing new methods for emotion analysis such as (Staiano and Guerini, 2014) only evaluate performance on a single benchmark dataset, SemEval 2007 Task 14, that contains 1250 observations. Recently, criticism has been raised concerning the use of correlation with continuous ratings as a measure (Buechel and Hahn, 2016), making only the somewhat limited binary evaluation possible. We only evaluate the emotions {Fear, Joy, Sadness} as the remaining emotions occur in less than 5% of the observations.

To fully evaluate our method on emotion analysis against the current methods we thus make use of two other datasets: A dataset of emotions in tweets related to the Olympic Games created by Sintsova et al. that we convert to a single-label classification task and a dataset of self-reported emotional experiences created by a large group of psychologists (Wallbott and Scherer, 1986). See the supplementary material for details on the datasets and the preprocessing. As these two datasets do not have prior evaluations, we evaluate against a state-of-the-art approach, which is based on a valence-arousal-dominance framework (Buechel and Hahn, 2016). The scores extracted using this approach are mapped to the classes in the datasets using a logistic regression with parameter optimization using cross-validation. We release our preprocessing code and hope that these 2 two datasets will be used for future benchmarking within emotion analysis.

We evaluate sentiment analysis performance on three benchmark datasets. These small datasets are chosen to emphasize the importance of the transfer learning ability of the evaluated models. Two of the datasets are from SentiStrength (Thelwall et al., 2010), SS-Twitter and SS-Youtube, and follow the relabeling described in (Saif et al., 2013) to make the labels binary. The third dataset is from SemEval 2016 Task4A (Nakov et al., 2016). Due to tweets being deleted from Twitter, the SemEval dataset suffers from data decay, making it difficult to compare results across papers. At the time of writing, roughly 15% of the training dataset for SemEval 2016 Task 4A was impossible to obtain. We choose not to use review datasets for sentiment benchmarking as these datasets contain so many words pr. observation that even bag-of-words classifiers and unsupervised approaches can obtain a high accuracy (Joulin et al., 2016; Radford et al., 2017).

The current state of the art for sentiment analysis on social media (and winner of SemEval 2016 Task 4A) uses an ensemble of convolutional neural networks that are pretrained on a private dataset of tweets with emoticons, making it difficult to replicate (Deriu et al., 2016). Instead we pretrain a model with the hyperparameters of the largest model in their ensemble on the positive/negative emoticon dataset from Go et al. (2009). Using this pretraining as an initialization we finetune the model on the target tasks using early stopping on a validation set to determine the amount of training. We also implemented the Sentiment-Specific Word Embedding (SSWE) using the embeddings available on the authors' website (Tang et al., 2014), but found that it performed worse

Table 4: Description of benchmark datasets. Datasets without pre-existing training/test splits are split by us (with splits publicly available). Data used for hyperparameter tuning is taken from the training set.

Identifier	Study	Task	Domain	Classes	N_{train}	N_{test}
SE0714	(Strapparava and Mihalcea, 2007)	Emotion	Headlines	3	250	1000
Olympic	(Sintsova et al., 2013)	Emotion	Tweets	4	250	709
PsychExp	(Wallbott and Scherer, 1986)	Emotion	Experiences	7	1000	6480
SS-Twitter	(Thelwall et al., 2012)	Sentiment	Tweets	2	1000	1113
SS-Youtube	(Thelwall et al., 2012)	Sentiment	Video Comments	2	1000	1142
SE1604	(Nakov et al., 2016)	Sentiment	Tweets	3	7155	31986
SCv1	(Walker et al., 2012)	Sarcasm	Debate Forums	2	1000	995
SCv2-GEN	(Oraby et al., 2016)	Sarcasm	Debate Forums	2	1000	2260

Table 5: Comparison across benchmark datasets. Reported values are averages across five runs. Variations refer to transfer learning approaches in §3.3 with 'new' being a model trained without pretraining.

Dataset	Measure	State of the art	DeepMoji (new)	DeepMoji (full)	DeepMoji (last)	DeepMoji (chain-thaw)
SE0714	F1	.34 [Buechel]	.21	.31	.36	**.37**
Olympic	F1	.50 [Buechel]	.43	.50	**.61**	**.61**
PsychExp	F1	.45 [Buechel]	.32	.42	.56	**.57**
SS-Twitter	Acc	.82 [Deriu]	.62	.85	.87	**.88**
SS-Youtube	Acc	.86 [Deriu]	.75	.88	.92	**.93**
SE1604	Acc	.51 [Deriu]	.51	.54	**.58**	**.58**
SCv1	F1	.63 [Joshi]	.67	.65	.68	**.69**
SCv2-GEN	F1	.72 [Joshi]	.71	.71	.74	**.75**

than the pretrained convolutional neural network. These results are therefore excluded.

For sarcasm detection we use the sarcasm dataset version 1 and 2 from the Internet Argument Corpus (Walker et al., 2012). Note that results presented on these benchmarks in e.g. Oraby et al. (2016) are not directly comparable as only a subset of the data is available online.[3] A state-of-the-art baseline is found by modeling the embedding-based features from Joshi et al. (2016) alongside unigrams, bigrams and trigrams with an SVM. GoogleNews word2vec embeddings (Mikolov et al., 2013) are used for computing the embedding-based features. A hyperparameter search for regularization parameters is carried out using cross-validation. Note that the sarcasm dataset version 2 contains both a quoted text and a sarcastic response, but to keep the models identical across the datasets only the response is used.

For training we use the Adam optimizer (Kingma and Ba, 2015) with gradient clipping of the norm to 1. Learning rate is set to 1E$-$3 for training of all new layers and 1E$-$4 for finetuning any pretrained layers. To prevent overfitting on the small datasets, 10% of the channels across all words in the embedding layer are dropped out during training. Unlike e.g. (Gal and Ghahramani, 2016) we do not drop out entire words in the input as some of our datasets contain observations with so few words that it could change the meaning of the text. In addition to the embedding dropout, L2 regularization for the embedding weights is used and 50% dropout is applied to the penultimate layer.

Table 5 shows that the DeepMoji model outperforms the state of the art across all benchmark datasets and that our new 'chain-thaw' approach consistently yields the highest performance for the transfer learning, albeit often only slightly better or equal to the 'last' approach. Results are averaged across 5 runs to reduce the variance. We test the statistical significance of our results by comparing the performance of DeepMoji (chain-thaw) vs. the state of the art. Bootstrap testing with 10000 samples is used. On all datasets are our results statistically significantly better than the state of the art with $p < 0.001$.

Our model is able to out-perform the state-of-

[3] We contacted the authors, but were unable to obtain the full dataset for neither version 1 or version 2.

the-art on datasets that originate from domains that differ substantially from the tweets on which it was pretrained. A key difference between the pretraining dataset and the benchmark datasets is the length of the observations. The average number of tokens pr. tweet in the pretraining dataset is 11, whereas e.g. the board posts from the Internet Argument Corpus version 1 (Oraby et al., 2016) has an average of 66 tokens with some observations being much longer.

5 Model Analysis

5.1 Importance of emoji diversity

One of the major differences between this work compared to previous papers using distant supervision is the diversity of the noisy labels used (see §2). For instance, both Deriu et al. (2016) and Tang et al. (2014) only used positive and negative emoticons as noisy labels. Other instances of previous work have used slightly more nuanced sets of noisy labels (see §2), but to our knowledge our set of noisy labels is the most diverse yet. To analyze the effect of using a diverse emoji set we create a subset of our pretraining data containing tweets with one of 8 emojis that are similar to the positive/negative emoticons used by Tang et al. (2014) and Hu et al. (2013) (the set of emoticons and corresponding emojis are available in the supplemental material). As the dataset based on this reduced set of emojis contains 433M tweets, any difference in performance on benchmark datasets is likely linked to the diversity of labels rather than differences in dataset sizes.

We train our DeepMoji model to predict whether the tweets contain a positive or negative emoji and evaluate this pretrained model across the benchmark datasets. We refer to the model trained on the subset of emojis as *DeepMoji-PosNeg* (as opposed to *DeepMoji*). To test the emotional representations learned by the two pretrained models the 'last' transfer learning approach is used for the comparison, thereby only allowing the models to map already learned features to classes in the target dataset. Table 6 shows that DeepMoji-PosNeg yields lower performance compared to DeepMoji across all 8 benchmarks, thereby showing that the diversity of our emoji types encourage the model to learn a richer representation of emotional content in text that is more useful for transfer learning.

Table 6: Benchmarks using a smaller emoji set (Pos/Neg emojis) or a classic architecture (standard LSTM). Results for DeepMoji from Table 5 are added for convenience. Evaluation metrics are as in Table 5. Reported values are the averages across five runs.

Dataset	Pos/Neg emojis	Standard LSTM	DeepMoji
SE0714	.32	.35	.36
Olympic	.55	.57	.61
PsychExp	.40	.49	.56
SS-Twitter	.86	.86	.87
SS-Youtube	.90	.91	.92
SE1604	.56	.57	.58
SCv1	.66	.66	.68
SCv2-GEN	.72	.73	.74

Many of the emojis carry similar emotional content, but have subtle differences in usage that our model is able to capture. Through hierarchical clustering on the correlation matrix of the DeepMoji model's predictions on the test set we can see that the model captures many similarities that one would intuitively expect (see Figure 3). For instance, the model groups emojis into overall categories associated with e.g. negativity, positivity or love. Similarly, the model learns to differentiate within these categories, mapping sad emojis in one subcategory of negativity, annoyed in another subcategory and angry in a third one.

5.2 Model architecture

Our DeepMoji model architecture as described in §3.2 use an attention mechanism and skip-connections to ease the transfer of the learned representation to new domains and tasks. Here we compare the DeepMoji model architecture to that of a standard 2-layer LSTM, both compared using the 'last' transfer learning approach. We use the same regularization and training parameters.

As seen in Table 6 the DeepMoji model performs better than a standard 2-layer LSTM across all benchmark datasets. The two architectures performed equally on the pretraining task, suggesting that while the DeepMoji model architecture is indeed better for transfer learning, it may not necessarily be better for single supervised classification task with ample available data.

A reasonable conjecture is that the improved transfer learning performance is due to two factors: a) the attention mechanism with skip-connections provide easy access to learned low-

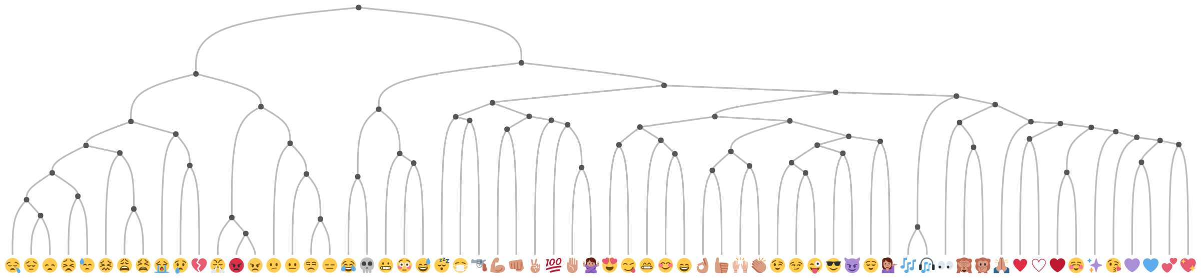


Figure 3: Hierarchical clustering of the DeepMoji model's predictions across categories on the test set. The dendrogram shows how the model learns to group emojis into overall categories and subcategories based on emotional content. The y-axis is the distance on the correlation matrix of the model's predictions measured using average linkage. More details are available in the supplementary material.

level features for any time step, making it easy to use this information if needed for a new task b) the improved gradient-flow from the output layer to the early layers in the network due to skip-connections (Graves, 2013) is important when adjusting parameters in early layers as part of transfer learning to small datasets. Detailed analysis of whether these factors actually explain why our architecture outperform a standard 2-layer LSTM is left for future work.

5.3 Analyzing the effect of pretraining

Performance on the target task benefits strongly from pretraining as shown in Table 5 by comparing DeepMoji (new) to DeepMoji (chain-thaw). In this section we experimentally decompose the benefit of pretraining into 2 effects: word coverage and phrase coverage. These two effects help regularize the model by preventing overfitting (see the supplementary details for an visualization of the effect of this regularization).

There are numerous ways to express a specific sentiment, emotion or sarcastic comment. Consequently, the test set may contain specific language use not present in the training set. The pretraining helps the target task models attend to low-support evidence by having previously observed similar usage in the pretraining dataset. We first examine this effect by measuring the improvement in word coverage on the test set when using the pretraining with word coverage being defined as the % of words in the test dataset seen in the training/pretraining dataset (see Table 7). An important reason why the 'chain-thaw' approach outperforms other transfer learning approaches is can be used to tune the embedding layer with limited risk of overfitting. Table 7 shows the increased word coverage from adding new words to the vocabulary as part of that tuning.

Note that word coverage can be a misleading metric in this context as for many of these small datasets a word will often occur only once in the training set. In contrast, all of the words in the pretraining vocabulary are present in thousands (if not millions) of observations in the emoji pretraining dataset thus making it possible for the model to learn a good representation of the emotional and semantic meaning. The added benefit of pretraining for learning word representations therefore likely extends beyond the differences seen in Table 7.

Table 7: Word coverage on benchmark test sets using only the vocabulary generated by finding words in the training data ('own'), the pretraining vocabulary ('last') or a combination of both vocabularies ('full / chain-thaw').

Dataset	Own	Last	Full / Chain-thaw
SE0714	41.9%	93.6%	94.0%
Olympic	73.9%	90.3%	96.0%
PsychExp	85.4%	98.5%	98.8%
SS-Twitter	80.1%	97.1%	97.2%
SS-Youtube	79.6%	97.2%	97.3%
SE1604	86.1%	96.6%	97.0%
SCv1	88.7%	97.3%	98.0%
SCv2-GEN	86.5%	97.2%	98.0%

To examine the importance of capturing phrases and the context of each word, we evaluate the accuracy on the SS-Youtube dataset using a fastText classifier pretrained on the same emoji dataset as our DeepMoji model. This fastText classifier is almost identical to only using the embedding layer from the DeepMoji model. We evaluate the representations learned by fine-tuning the models as feature extractors (i.e. using the 'last' transfer learning approach). The fastText model achieves an accuracy of 63% as compared to 93% for our DeepMoji model, thereby emphasizing the importance of phrase coverage. One concept that the LSTM layers likely learn is negation, which is known to be important for sentiment analysis (Wiegand et al., 2010).

5.4 Comparing with human-level agreement

To understand how well our DeepMoji classifier performs compared to humans, we created a new dataset of random tweets annotated for sentiment. Each tweet was annotated by a minimum of 10 English-speaking Amazon Mechanical Turkers (MTurk's) living in USA. Tweets were rated on a scale from 1 to 9 with a 'Do not know' option, and guidelines regarding how to rate the tweets were provided to the human raters. The tweets were selected to contain only English text, no mentions and no URL's to make it possible to rate them without any additional contextual information. Tweets where more than half of the evaluators chose 'Do not know' were removed (98 tweets).

For each tweet, we select a MTurk rating random to be the 'human evaluation', and average over the remaining nine MTurk ratings are averaged to form the ground truth. The 'sentiment label' for a given tweet is thus defined as the overall consensus among raters (excluding the randomly-selected 'human evaluation' rating). To ensure that the label categories are clearly separated, we removed neutral tweets in the interval $[4.5, 5.5]$ (roughly 29% of the tweets). The remaining dataset consists of 7 347 tweets. Of these tweets, 5000 are used for training/validation and the remaining are used as the test set. Our DeepMoji model is trained using the chain-thaw transfer learning approach.

Table 8 shows that the agreement of the random MTurk rater is 76.1%, meaning that the randomly selected rater will agree with the average of the nine other MTurk-ratings of the tweet's polarity 76.1% of the time. Our DeepMoji model achieves 82.4% agreement, which means it is better at capturing the average human sentiment-rating than a single MTurk rater.

Table 8: Comparison of agreement between classifiers and the aggregate opinion of Amazon Mechanical Turkers on sentiment prediction of tweets.

	Agreement
Random	50.1%
fastText	71.0%
MTurk	76.1%
DeepMoji	**82.4%**

6 Conclusion

We have shown how the millions of texts on social media with emojis can be used for pretraining models, thereby allowing them to learn representations of emotional content in texts. Through comparison with an identical model pretrained on a subset of emojis, we find that the diversity of our emoji set is important for the performance of our method. We release our pretrained DeepMoji model with the hope that other researchers will find good use of them for various emotion-related NLP tasks[4].

Acknowledgments

The authors would like to thank Janys Analytics for generously allowing us to use their dataset of human-rated tweets and the associated code to analyze it. Furthermore, we would like to thank Max Lever, who helped design the website, and Han Thi Nguyen, who helped code the software that is provided alongside the pretrained model.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR)*.

Yoshua Bengio et al. 2012. Deep learning of representations for unsupervised and transfer learning. In *29th International Conference on Machine learning (ICML) – Workshop on Unsupervised and Transfer Learning*, volume 27, pages 17–36.

Sven Buechel and Udo Hahn. 2016. Emotion analysis as a regression problem - dimensional models and their implications on emotion representation and metrical evaluation. In *22nd European Conference on Artificial Intelligence (ECAI)*.

[4] Available with preprocessing code, examples of usage, benchmark datasets etc. at github.com/bfelbo/deepmoji

François Chollet et al. 2015. Keras. https://github.com/fchollet/keras.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *25th International Conference on Machine learning (ICML)*, pages 160–167.

Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurelien Lucchi, Valeria De Luca, and Martin Jaggi. 2016. Swisscheese at semeval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. *Proceedings of SemEval*, pages 1124–1128.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *31th International Conference on Machine Learning (ICML)*, volume 32, pages 647–655.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. In *4th International Workshop on Natural Language Processing for Social Media (SocialNLP)*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research (JMLR)*, 11:625–660.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *30th Conference on Neural Information Processing Systems (NIPS)*, pages 1019–1027.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

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

Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. 2013. Unsupervised sentiment analysis with emotional signals. In *Proceedings of the 22nd international conference on World Wide Web (WWW)*, pages 607–618. ACM.

Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. 2016. Are word embedding-based features useful for sarcasm detection? In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*.

FA Kunneman, CC Liebrecht, and APJ van den Bosch. 2014. The (un)predictability of emotional hashtags in twitter. In *52th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *27th Conference on Neural Information Processing Systems (NIPS)*, pages 3111–3119.

Saif Mohammad. 2012. #emotional tweets. In *The First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 246–255. Association for Computational Linguistics.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *10th International Workshop on Semantic Evaluation (SemEval)*, pages 1–18.

Shereen Oraby, Vrindavan Harrison, Lena Reed, Ernesto Hernandez, Ellen Riloff, and Marilyn Walker. 2016. Creating and characterizing a diverse corpus of sarcasm in dialogue. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, page 31.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Jonathon Read. 2005. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *ACL student research workshop*, pages 43–48. Association for Computational Linguistics.

Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. 2013. Evaluation datasets for twitter sentiment analysis: a survey and a new dataset, the sts-gold. In *Workshop: Emotion and Sentiment in Social and Expressive Media: approaches and perspectives from AI (ESSEM) at AI*IA Conference*.

Valentina Sintsova, Claudiu-Cristian Musat, and Pearl Pu. 2013. Fine-grained emotion recognition in olympic tweets based on human computation. In *4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*.

Jonas Sjöberg and Lennart Ljung. 1995. Overtraining, regularization and searching for a minimum, with application to neural networks. *International Journal of Control*, 62(6):1391–1407.

Jacopo Staiano and Marco Guerini. 2014. Depechemood: A lexicon for emotion analysis from crowd-annotated news. In *52th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *4th International Workshop on Semantic Evaluations (SemEval)*, pages 70–74. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *28th Conference on Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Jared Suttles and Nancy Ide. 2013. Distant supervision for emotion classification with discrete binary values. In *International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, pages 121–136. Springer.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *52th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1555–1565.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2012. Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology (JASIST)*, 63(1):163–173.

Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558.

Marilyn A Walker, Jean E Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012. A corpus for research on deliberation and debate. In *International Conference on Language Resources and Evaluation (LREC)*, pages 812–817.

Harald G Wallbott and Klaus R Scherer. 1986. How universal and specific is emotional experience? evidence from 27 countries on five continents. *International Social Science Council*, 25(4):763–795.

Michael Wiegand, Alexandra Balahur, Benjamin Roth, Dietrich Klakow, and Andrés Montoyo. 2010. A survey on the role of negation in sentiment analysis. In *Workshop on Negation and Speculation in Natural Language Processing (NeSp-NLP)*, pages 60–68. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*.

Opinion Recommendation Using A Neural Model[*]

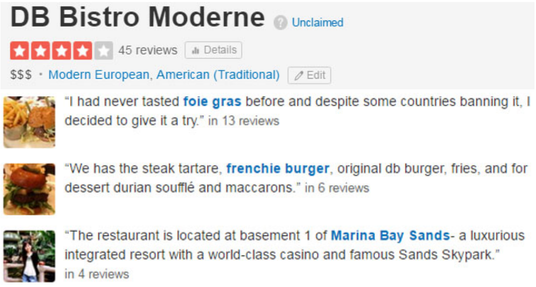
Zhongqing Wang[†,‡] and **Yue Zhang**[‡]
[†]Soochow University, Suzhou, China
[‡]Singapore University of Technology and Design, Singapore
wangzq.antony@gmail.com, yue_zhang@sutd.edu.sg,

Abstract

We present opinion recommendation, a novel task of jointly generating a review with a rating score that a certain user would give to a certain product which is unreviewed by the user, given existing reviews to the product by other users, and the reviews that the user has given to other products. A characteristic of opinion recommendation is the reliance of multiple data sources for multi-task joint learning. We use a single neural network to model users and products, generating customised product representations using a deep memory network, from which customised ratings and reviews are constructed jointly. Results show that our opinion recommendation system gives ratings that are closer to real user ratings on Yelp.com data compared with Yelp's own ratings. our methods give better results compared to several pipelines baselines.

1 Introduction

Offering a channel for customers to share opinions and give scores to products and services, review websites have become a highly influential information source that customers refer to for making purchase decisions. Popular examples include IMDB.com on the movie domain, Epinions.com on the product domain, and Yelp.com on the service domain. Figure 1 shows a screenshot of a restaurant review page on Yelp.com, which offers two main types of information. First, an overall rating score is given under the restaurant name; second, detailed user reviews are listed below the rating.



DB Bistro Moderne

★★★★☆ 45 reviews

$$$ · Modern European, American (Traditional)

"I had never tasted **foie gras** before and despite some countries banning it, I decided to give it a try." in 13 reviews

"We has the steak tartare, **frenchie burger**, original db burger, fries, and for dessert durian soufflé and maccarons." in 6 reviews

"The restaurant is located at basement 1 of **Marina Bay Sands**- a luxurious integrated resort with a world-class casino and famous Sands Skypark." in 4 reviews

Figure 1: A restaurant review on Yelp.com.

Though offering useful overview and details about a product or service, such information has several limitations for a user who has *not* used the product or service. First, the overall rating is general and not necessarily agreeable to the taste of an individual customer. Being a simple reflection of all customer scores, it serves an average customer well, but can be rather inaccurate for individuals. For example, the authors themselves often find highly rated movies being tedious. Second, there can be hundreds of reviews for a product or service, which makes it infeasible for exhaustive reading. It would be useful to have a brief summary of all reviews, which ideally should be customized to the reader.

To address the limitations above, we propose a new task called **opinion recommendation**, which is to generate a *customized review score* of the product that the user is likely to give, as well as a *customized review* that the user would have written for the target product, if the user had reviewed the product. The proposed opinion recommendation task is closely related to several existing lines of work in NLP. The first is *sentiment analysis* (Hu and Liu, 2004; Pang and Lee, 2008) and *opinion summarization* (Nishikawa et al., 2010; Wang and Ling, 2016), which is to give a rating score or summary based on existing customer reviews. Our

[*] This work has been done when the first author worked at SUTD.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1626–1637
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

task is different in that we aim to generate user rating scores and review of a product unreviewed by the user. The second is *recommendation* (Su and Khoshgoftaar, 2009; Yang et al., 2014), which is to give a ranking score to a certain product or service based on the purchase history of the user and other customers who have purchased the target product. Our task is different in the source of input, which is *textual* customer reviews and ratings rather than *numerical* purchase history.

There are two types of inputs for our task, namely existing reviews of the target product, and the reviews of the user on other products, and two types of outputs, namely a customized rating score and a customized review. The ideal solution should consider the interaction between all given types of information, jointly predicting the two types of outputs. This poses significant challenges to statistical models, which require manually defined features to capture relevant patterns from training data. Deep learning is a relatively more feasible choice, offering viabilities of information fusion by fully connected hidden layers (Collobert et al., 2011; Henderson et al., 2013; Zhang and Weiss, 2016; Chen et al., 2016a). We leverage this advantage in building our model.

In particular, we use a sub RNN to model the semantic content of each review. A sub *product model* is used to consolidate existing reviews for the target product, and a *user model* is built by consolidating the reviews of the given user into a single vector form. To address potential sparsity of a user's history reviews, neighbor users are identified by collaborative filtering (Ding et al., 2006), and a vector representation is learned by using a neural *neighborhood model*. Finally, a deep memory network is utilized to find the association between the user and target product, jointly yielding the rating score and customised review. Experiments on a Yelp dataset show that the model outperforms several pipelined baselines. We make our source code publicly available under GPL at https://github.com/wangzq870305/opinion_recommend.

2 Related Work

Sentiment Analysis. Our task is related to document-level sentiment classification (Pang and Lee, 2008) for various neural network models have been used, including convolutional neural networks (Kim, 2014), recursive neural network (Socher et al., 2013) and recurrent neural network (Teng et al., 2016; Tai et al., 2015), Review rating prediction aims to predict the numeric rating of a given review. Pang and Lee (2005) pioneered this task by regarding it as a classification/regression problem. Most subsequent work focuses on designing effective textural features of reviews (Qu et al., 2010; Li et al., 2011; Wan, 2013).

User information has been widely investigated in sentiment analysis. Gao et al. (2013) developed user-specific features to capture user leniency, and Li et al. (2014) incorporated textual topic and user-word factors through topic modeling. For integrating user information into neural network models, Tang et al. (2015) predicted the rating score given a review by using both lexical semantic information and a user embedding model. Chen et al. (2016b) proposed a neural network to incorporate global user and product information for sentiment classification via an attention mechanism.

Different from the above research, which focuses on predicting the opinion on existing reviews, our task is to recommend the score that a user would give to a new product *without* knowing his review text. The difference originates from the objective. Previous research aims to predict opinions on reviewed products, while our task is to recommend opinion on new products, which the user has not reviewed.

Opinion Summarization. Our work also overlaps with to the area of opinion summarization, which constructs natural language summaries for multiple product reviews (Hu and Liu, 2004). Most previous work extracts opinion words and aspect terms. Typical approaches include association mining of frequent candidate aspects (Hu and Liu, 2004; Qiu et al., 2011), sequence labeling based methods (Jakob and Gurevych, 2010; Yang and Cardie, 2013), as well as topic modeling techniques (Lin and He, 2009). Recently, word embeddings and recurrent neural networks are also used to extract aspect terms (Irsoy and Cardie, 2014; Liu et al., 2015). While all the methods above are *extractive*, Ganesan et al. (2010) presented a graph-based summarization framework to generate concise abstractive summaries of highly redundant opinions, and Wang and Ling (2016) used an attention-based neural network model to absorb information from multiple text units and generate summaries of movie reviews. We also perform abstractive summarization. However, different from the above research, which summarize existing reviews, we generate *customized* reviews for a unreviewed product.

Recommendation. has been solved on mainly purchase history. There are two main approaches, which are content-based and collaborative-filtering (CF) based (Adomavicius and Tuzhilin, 2005; Yang et al., 2014), respectively. Most existing social recommendation systems are CF-based, and can be further grouped into model-based CF and neighborhood-based CF (Kantor et al., 2011; Su and Khoshgoftaar, 2009). Matrix Factorization (MF) is one of the most popular models for CF. In recent MF-based social recommendation works, user-user social trust information is integrated with user-item feedback history (e.g., ratings, clicks, purchases) to improve the accuracy of traditional recommendation systems, which only factorize user-item feedback data (Ding et al., 2006; Koren, 2008; He et al., 2016).

There has been work integrating sentiment analysis and recommendation systems, which use recommendation strategies such as matrix factorization to improve the performance of sentiment analysis (Leung et al., 2006; Singh et al., 2011). These methods typically use ensemble learning (Singh et al., 2011) or probabilistic graph models (Wu and Ester, 2015). For example, Zhang et al. (2014) proposed a factor graph model to recommend opinion rating scores by using explicit product features as hidden variables. Different from the above research, we recommend user opinions.

Neural Network Models. Multi-task learning has been recognised as a strength of neural network models for natural language processing (Collobert et al., 2011; Henderson et al., 2013; Zhang and Weiss, 2016; Chen et al., 2016a), where hidden feature layers are shared between different tasks that have common basis. Our work can be regarded as an instance of such multi-tasks learning via shared parameters, which has been widely used in the research community recently.

Dynamic memory network models have been applied for NLP tasks such as question answering (Sukhbaatar et al., 2015; Kumar et al., 2016), language modeling (Tran et al., 2016) and machine translation (Wang et al., 2016). There are typically used to find abstract semantic representations of texts towards certain tasks, which are consistent with our main need, namely abstracting the representation of a product that is biased towards the taste of a certain user. We use a variation of the memory network model for obtaining user-specific review representation.

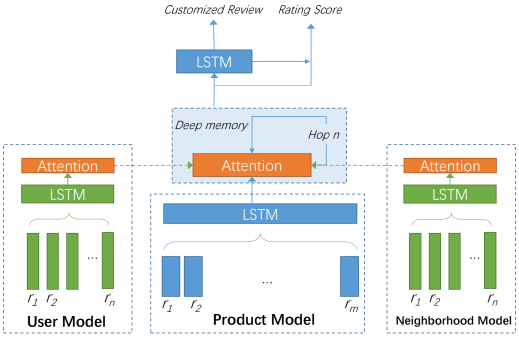


Figure 2: Overview of proposed model.

3 Model

Formally, the input to our model is a tuple $\langle R_T, R_U, R_N\rangle$, where $R_T = \{r_{T_1}, r_{T_2}, ..., r_{T_{n_t}}\}$ is the set of existing reviews of a target product, $R_U = \{r_{U_1}, r_{U_2}, ..., r_{U_{n_u}}\}$ is the set of user's history reviews, and $R_N = \{r_{N_1}, r_{N_2}, ..., r_{N_{n_n}}\}$ is the set of the user's neighborhood reviews. All the reviews are sorted with temporal order. The output is a pair $\langle Y_S, Y_R\rangle$, where Y_S is a real number between 0 and 5 representing the customized rating score of the target product, and Y_R is a customised review. A characteristic of our model is that Y_S and Y_R are generated on a product that the user has not reviewed.

For capturing both general and personalized information, we first build a *product model*, a *user model*, and a *neighborhood model*, respectively, and using a memory network model to integrate these three types of information, constructing a *customized product model*. Finally, we predict a customized rating score and a review collectively using neural stacking framework. The overall architecture of the model is shown in Figure 2.

3.1 Review Model

A review is the foundation of our model, based on which we derive representations of both a user and a target product. In particular, a user profile can be achieved by modeling all the reviews R_U of the user, and a target product profile can be obtained by using all existing reviews R_T of the product. We use the average of word embeddings to model a review. Formally, given a review $r = \{x_1, x_2, ..., x_m\}$, where m is the length of the review, each word x_k is represented with a K-dimensional embedding e_k^w (Mikolov et al., 2013). We use the $\sum_k(e_k^w)/m$ for the representation of the review $e^d(r)$.

3.2 User Model

A standard LSTM (Hochreiter and Schmidhuber, 1997) is used to learn the hidden states of an user's reviews to build the user model. Denoting the recurrent function at step t as $\text{LSTM}(x_t, h_{t-1})$, we obtain a sequence of hidden state vectors $\{h_{U_1}, h_{U_2}, ..., h_{U_{n_u}}\}$ recurrently by feeding $\{e^d(r_{U_1}), e^d(r_{U_2}), ..., e^d(r_{U_{n_u}})\}$ as inputs, where $h_{U_i} = \text{LSTM}(e^d(r_{U_i}), h_{U_{i-1}})$. The initial state and all standard LSTM parameters are randomly initialized and tuned during training.

Not all reviews contribute equally to the representation of a user. We use the attention mechanism (Bahdanau et al., 2014; Yang et al., 2016) to extract the reviews that are relatively more important, aggregating the representation of reviews to form a vector. Taking the hidden state $\{h_{U_1}, ...h_{U_2}, ..., h_{U_{n_u}}\}$ of user model as input, the attention model outputs, a continuous vector $v_U \in \mathbb{R}^{d\times 1}$, which is computed as a weighted sum of each hidden state h_{U_i}, namely

$$v_U = \sum_i^{n_u} \alpha_i h_{U_i} \tag{1}$$

where n_u is the hidden variable size, $\alpha_i \in [0, 1]$ is the weight of h_{U_i}, and $\sum_i \alpha_i = 1$.

For each piece of hidden state h_{U_i}, the scoring function is calculated by

$$u_i = \tanh(W_U h_{U_i} + b_U) \tag{2}$$

$$\alpha_i = \frac{\exp(u_i)}{\sum_j \exp(u_j)} \tag{3}$$

where W_U and b_U are model parameters. The attention vector v_U is used to represent the user for the *User Model*.

3.3 Neighborhood Model

We use neighborhood reviews to improve the user model, since a user may not have sufficient reviews to construct a reliable model. Here a neighbor refers to a user that has similar tastes to the target user (Koren, 2008; Desrosiers and Karypis, 2011). The same as the user model, we construct the *neighborhood model* v_N using the neighborhood reviews $R_N = \{r_{N_1}, r_{N_2}, ..., r_{N_{n_n}}\}$ with an attention recurrent network.

A key issue in building the neighborhood model is how to find neighbors of a certain user. In this study, we use matrix factorization (Koren, 2008) to detect neighbors, which is a standard approach for recommendation (Ding et al., 2006; Li et al., 2009; He et al., 2016). In particular, users' rating scores of products are used to build a product-users matrix $M \in \mathbb{R}^{n_t \times n_u}$ with n_t products and n_u users. We approximate it using three factors, specifying soft membership of products and users (Ding et al., 2006) by finding:

$$\begin{gathered}\min_{F,S,T} ||M - FST^T|| \\ s.t. S \geq 0, F \geq 0, T \geq 0\end{gathered} \tag{4}$$

where $F \in \mathbb{R}^{n_t \times K}$ represents the posterior probability of K topic clusters for each product; $S \in \mathbb{R}^{K\times K}$ encodes the distribution of each topic k; and $T \in \mathbb{R}^{K\times n_u}$ indicates the posterior probability of K topic clusters for each user.

As a result of matrix factorization, we directly obtain the probability of each user on each topic from the person-topic matrix T. To infer T, the optimization problem in Eq.4 can be solved using the following updating rule:

$$T_{jk} \leftarrow T_{jk} \frac{(M^T FS)_{jk}}{(TT^T M^T FS)_{jk}} \tag{5}$$

With the user-topic matrix T, we measure the implicit connection between two users using:

$$sim(i, j) = \sum_{k=1}^{k} T_{ik} T_{jk} \tag{6}$$

where $sim(i, j)$ measure the implicit connection degree between users i and j. If $sim(i, j)$ is higher than a threshold η, we consider user j as the neighbor of user i.

3.4 Product Model

Given the representations of existing reviews $\{e^d(r_{T_1}), e^d(r_{T_2}), ..., e^d(r_{T_{n_t}})\}$ of the product, we use LSTM to model their temporal orders, obtaining a sequence of hidden vectors $h_T = \{h_{T_1}, h_{T_2}, ..., h_{T_{n_t}}\}$ by recurrently feeding $\{e^d(r_{T_1}), e^d(r_{T_2}), ..., e^d(r_{T_{n_t}}\}$ as inputs. The hidden state vectors h_T are used to represent the product.

Customized Product Model. The product model represents salient information of existing reviews in their temporal order, yet do not reflect the taste of a particular user. We build the customised product model to integrate user information and product information (as reflected by the product model), resulting in a single vector that represents a customised product. From this vector we are able to synthesis both a customised review and a customised rating score. In particular, we use the user representation v_U and the neighbour representation v_N to transform the target product representation $h_T = \{h_{T_1}, h_{T_2}, ..., h_{T_{n_t}}\}$ into a customised product representation v_C, which is tailored to the taste of the user.

A naive model of yielding v_C could utilise the attention mechanism over h_t, deriving a weighted sum according to user information. On the other hand, dynamic memory networks have been shown highly useful for deriving abstract semantic information compared with simple attention, and hence we follow Sukhbaatar et al. (2015) and Xiong et al. (2016), building a variation of DMN to iteratively find increasingly abstract representations of h_t, by injecting v_U and v_N information.

The memory model consists of multiple dynamic computational layers (hops), each of which contains an attention layer and a linear layer. In the first computational layer (hop 1), we take the hidden variables h_{T_i} $(0 \leq i \leq n_t)$ of product model as input, adaptively selecting important evidences through one attention layer using v_U and v_N. The output of the attention layer gives a linear interpolation of h_T, and the result is considered as input to the next layer (hop 2). In the same way, we stack multiple hops and run the steps multiple times, so that more abstract representations of the target product can be derived.

The attention model outputs a continuous vector $v_C \in \mathbb{R}^{d\times 1}$, which is computed as a weighted sum of h_{T_i} $(0 \leq i \leq n_t)$, namely

$$v_C = \sum_{i}^{n_t} \beta_i h_{T_i} \tag{7}$$

where n_t is the hidden variable size, $\beta_i \in [0, 1]$ is the weight of h_{T_i}, and $\sum_i \beta_i = 1$. For each piece of hidden state h_{T_i}, we use a feed forward neural network to compute its semantic relatedness with the abstract representation v_C. The scoring function is calculated as follows at hop t:

$$\begin{aligned} u_i^t = \tanh(&W_T h_{T_i} + W_C v_C^{t-1} \\ &+ W_U v_U + W_N v_N + b) \end{aligned} \tag{8}$$

$$\beta_i^t = \frac{\exp(u_i^t)}{\sum_j \exp(u_j^t)} \tag{9}$$

The vector v_C is used to represent the customized product model. At the first hop, we define $V_C^0 = \sum_i h_{T_i}/n_t$.

The product model h_{T_i} $(0 \leq i \leq n_t)$ represents salient information of existing reviews in their temporal order, they do not reflect the taste of a particular user. We use the customised product model to integrate user information and product information (as reflected by the product model), resulting in a single vector that represents a customised product. From this vector we are able to synthesis both a customised review and a customised rating score.

3.5 Customized Review Generation

The goal of customized review generation is to generate a review Y_R from the customized product representation v_C, composed by a sequence of words $y_{R_1}, ..., y_{R_{n_r}}$. We use a standard LSTM decoder (Rush et al., 2015) to decompose the prediction of Y_R into a sequence of word-level predictions:

$$\begin{aligned} &\log P(Y_R|v_C) = \\ &\quad \sum_j P(y_{R_j}|y_{R_1}, ..., y_{R_{j-1}}, v_C) \end{aligned} \tag{10}$$

where each word y_{R_j} is predicted conditional on the previously generated $y_{R_1}, ..., y_{R_{j-1}}$ and the customized product vector v_C. The probability is estimated by using standard word softmax:

$$\begin{aligned} P(y_{R_j}|y_{R_1}, ..., y_{R_{j-1}}, v_C) = \\ \mathrm{softmax}(h_{R_j}) \end{aligned} \tag{11}$$

where h_{R_j} is the hidden state variable at timestamp j, which is modeled as $LSTM(u_{j-1}, h_{Rj})$. Here a LSTM is used to generate a new state h_{R_j} from the representation of the previous state $h_{R_{j-1}}$ and u_{j-1}. u_{j-1} is the concatenation of previously generated word $y_{R_{j-1}}$ and the input representation of customized model v_C.

3.6 Customized Rating Prediction

A straightforward approach to predicting the rating score of a product is to take the average of existing review scores. However, the drawback is that it cannot reflect the the variance in user tastes. In order to integrate user preferences into the rating, we instead take a user-based weighted average of existing rating scores, so that the scores of reviews that are closer to the user preference are given higher weights. However, existing ratings can be all different from a users personal rating, if the existing reviews do not come from the user's neighbours. We thus use the customized product vector v_c as a bias of the weighted average of existing rating scores.

Formally, given the rating scores $s_1, s_2, ..., s_n$ of existing reviews, and the the customized product representation v_C, we calculate:

$$Y_S = \sum_{i}^{n} \beta_i \cdot s_i + \mu \tanh(W_S v_C + b_S) \qquad (12)$$

In the left term $\sum_i^n \beta_i \cdot s_i$, we use attention weights β_i in Eq.9 to measure the important of each rating score s_i. The right term $\tanh(W_S v_C + b_S)$ is a review-based shift, weighted by μ.

Since the result of customized review generation can be helpful for rating score prediction, we use neural stacking additionally feeding the last hidden state h_{R_n} of review generation model as input for Y_S prediction, resulting in

$$\begin{aligned} Y_S = & \sum_{i}^{n} \alpha_i \cdot s_i + \\ & + \mu \tanh(W_S(v_C \oplus h_{R_n}) + b_S) \end{aligned} \qquad (13)$$

where $\oplus$ denotes vector concatenation.

3.7 Training

For our task, there are two joint training objectives, for review scoring and review summarisation, respectively. For review scoring, the loss function is defined as:

$$L(\Theta) = \sum_{i=1}^{N} (Y^*_{S_i} - Y_{S_i})^2 + \frac{\lambda}{2} ||\Theta||^2 \qquad (14)$$

where $Y^*_{S_i}$ is the predicted rating score, Y_{S_i} is the rating score in the training data, Θ is the set of model parameters and λ is a parameter for L2 regularization.

	Amount
Business	15,584
Review	334,997
User	303,032

Table 1: Statistics of the dataset.

For customized review generation, loss is defined by maximizing the log probability of Eq.10 (Sutskever et al., 2014; Rush et al., 2015). The two loss functions for score and review prediction share the representation vectors under v_C, hence forming multi-task learning.

Standard back propagation is performed to optimize parameters, where gradients also propagate from the scoring objective to the review generation objective due to neural stacking (Eq.13). We apply online training, where model parameters are optimized by using AdaGrad (Duchi et al., 2011). Word embeddings are trained using the *Skip-gram* algorithm (Mikolov et al., 2013)[1].

4 Experiments

4.1 Experimental Settings

Our data are collected from the yelp academic dataset[2], provided by Yelp.com, a popular restaurant review website. The data set contains three types of objects: *business*, *user*, and *review*, where business objects contain basic information about local businesses (i.e. restaurants), review objects contain review texts and star rating, and user objects contain aggregate information about a single user across all of Yelp. Table 1 illustrates the general statistics of the dataset.

For evaluating our model, we choose 4,755 user-product pairs from the dataset. The user-product pairs are extracted by following criterions: for each selected user-product pair, the user should have written 10 reviews at least, and the product should contain 100 reviews at least. In addition, the gold-standard review that the user write for the corresponding product should contain 10 helpful hits at least. We did not try alternative data selection rules. We will give the detail in our draft.

For each pair, the existing reviews of the target service (restaurant) are used for the product model. The rating score given by each user to the target service is considered as the gold customized rating score, and the review of the target service given by

[1] https://code.google.com/p/word2vec/

[2] https://www.yelp.com/academic_dataset

each user is used as the gold-standard customized review for the user. The remaining reviews of each user are used for training the user model. We use 3,000 user-product pairs to train the model, 1,000 pairs as testing data, and remaining data for development.

We use the ROUGE-1.5.5 (Lin, 2004) toolkit for evaluating the performance of customized review generation, and report unigram overlap (ROUGE-1) as a means of assessing informativeness. Mean Square Error (MSE) (Wan, 2013; Tang et al., 2015) is used as the evaluation metric for measuring the performance of customized rating score prediction. MSE penalizes more severe errors more heavily.

4.2 Hyper-parameters

There are several important hyper-parameters in our models, and we tune their values using the development dataset. We set the regularization weight $\lambda = 10^{-8}$ and the initial learning rate to 0.01. We set the size of word vectors to 128, the size of hidden vectors in LSTM to 128. In order to avoid over-fitting, dropout (Hinton et al., 2012) is used for word embedding with a ratio of 0.2. The neighbor similarity threshold η is set to 0.25.

4.3 Development Experiments

4.3.1 Ablation Test

Effects of various configurations of our model, are shown on Table 2, where *Joint* is the full model of this paper, *-user* ablates the user model, *-neighbor* ablates the neighbor model, *-rating* is a single-task model that generates a review without the rating score, and *-generation* yields only the rating.

By comparing "Joint" and "-user,-neighbor", we can find that customized information have significant influence on both the rating and review generation results ($p - value < 0.01$ using t-test). In addition, comparison between "-Joint" and "-user", and between "-user" and "-user, -neighbor" shows that both the user information and the neighbour user information of the user are effective for improving the results. A users neighbours can indeed alleviate scarcity of user reviews.

Finally, comparison between "Joint" and "-generation", and between "Joint" and "-rating" shows that multi-task learning by parameter sharing is highly useful.

	Rating	**Generation**
Joint	0.904	0.267
-user	1.254	0.220
-neighbor	1.162	0.245
-user,-neighbor	1.342	0.205
-rating	-	0.254
-generation	1.042	-

Table 2: Feature ablation tests.

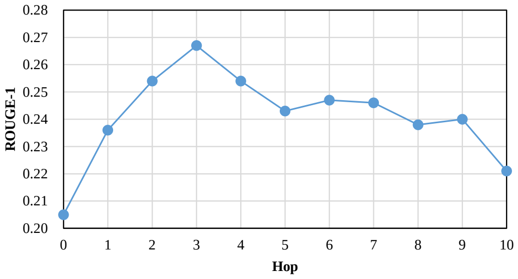


Figure 3: Influence of hops.

4.3.2 Influence of Hops

We show the influence of hops of memory network for customized review generation on Figure 3. When $hop = 0$, the model considers only the general product reviews ($-user, -neighbor$). When $hop \geq 1$, customized product information is leveraged. From the figure we can find that, when $hop = 3$, the performance is the best. It indicates that multiple hops can capture more abstract evidences from external memory to improve the performance. However, too many hops leads to over-fitting, thereby harms the performance. As a result, we choose 3 as the number of hops in our final test.

4.3.3 Influence of μ

We show the influence of the bias weight parameter μ for rating prediction in Figure 4. With μ being 0, the model uses the weighted sum of existing reviews to score the product. When μ is very large, the system tends to use only the customized product representation v_c to score the product, hence ignoring existing review scores, which are a useful source of information. Our results show that when μ is 1, the performance is optimal, thus indicating both existing review scores and review contents are equally useful.

4.4 Final Results

We show the final results for opinion recommendation, comparing our proposed model with the

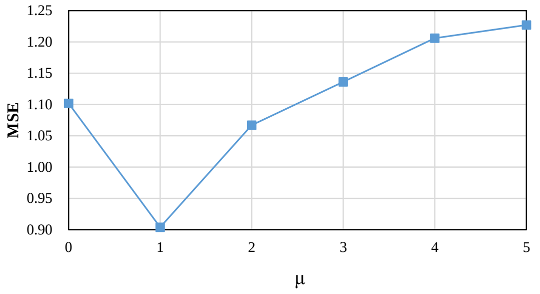


Figure 4: Influence of bias score.

	Rating	Generation
RS-Average-Yelp	1.280	-
RS-Linear	1.234	-
RS-Item	1.364	-
RS-MF	1.143	-
Sum-Opinosis	-	0.183
Sum-LSTM-Att	-	0.196
Joint	**1.023**	**0.250**

Table 3: Final results.

following state-of-the-art baseline systems:

- *RS-Average-Yelp* is the widely-adopted baseline (e.g., by Yelp.com), using the averaged review scores as the final score.
- *RS-Linear* estimates the rating score that a user would give by $s_{ui} = s_{all} + s_u + s_i$ (Ricci et al., 2011), where s_u and s_i are the the training deviations of rating score of the user u and the product i, respectively.
- *RS-Item* applies kNN to estimate the rating score (Sarwar et al., 2001). We choose the cosine similarity between v_c to measure the distance between product.
- *RS-MF* is a state-of-the-art recommendation model, which uses matrix factorisation to predict rating score (Ding et al., 2006; Li et al., 2009; He et al., 2016).
- *Sum-Opinosis* uses a graph-based framework to generate abstractive summarisation given redundant opinions (Ganesan et al., 2010).
- *Sum-LSTM-Att* is a state-of-the-art neural abstractive summariser, which uses an attentional neural model to consolidate information from multiple text sources, generating summaries using LSTM decoding (Rush et al., 2015; Wang and Ling, 2016).

Being non-opinion recommendation methods, all the baselines are single-task models, without considering rating and summarisation prediction jointly. The results are shown in Table 3. Our model (“ Joint”) significantly outperforms both “RS-Average-Yelp” and “RS-Linear” ($p - value < 0.01$ using t-test). Note that, our proposed rating recommendation for the user are significantly closer individual real user rating compared with Yelp’s rating.

Our proposed model also significantly outperforms state-of-the-art recommendation systems (RS-Item and RS-MF) ($p - value < 0.01$ using t-test), indicating that textual information are a useful addition to the rating scores themselves for recommending a product.

Finally, comparison between our proposed model and state-of-the-art summarisation techniques (Sum-Opinosis and Sum-LSTM-Att) shows the advantage of leveraging user information to enhance customised review generation, and also the strength of joint learning.

4.5 Example Output

Table 4 shows example outputs of rating scores and reviews. *Ref.* is the rating score and review written by user her/himself, and *Base* is the baseline model, that generates the rating score by RS-MF, and review by Sum-LSTM-Att. From these examples, we can find that, both rating score and review which generated by the proposed Joint model is closer to the real user. In particular, in the first example, the baseline system correctly identifies the main both price and quality information, which the target user wrote in the review, yet the baseline model did not yield comments about the price based on reviews of other users. Associating reviews and ratings closely, the joint model gives a rating score that is much closer to the real user score compared to the score given by the recommendation model MF. In addition, we can also find some habits of certain users from their customized reviews, for example, Mexican food, cheap and clean restaurant.

5 Conclusion

We proposed a novel task called opinion recommendation, which is to generate the review and rating score that a certain user would give to an unreviewed product or service. In particular, a

	Rating	Review
Ref.	5.0	Amazing beer selection, enough food choices, and a much smaller bill than I was expecting ...
Base	4.0	Boulders is JAM, favorite neighborhood bar, have amazing food ...
Joint	4.6	Bar is cheap, food is good enough ...
Ref.	4.0	This is one of my favorite Mexican fast food restaurants. It's clean and cool in the summer...
Base	5.0	The restaurant is great, This Chipotle is a great location, Their medium salsa good ...
Joint	4.2	Mexican food my favorite, place is clean ...

Table 4: Example outputs.

deep memory network was utilized to find the association between the user and the product, jointly yielding the rating score and customised review. Results show that our methods are better results compared to several pipelines baselines using state-of-the-art sentiment rating and summarisation systems. Review scores given by the opinion recordation system are closer to real user review scores compared to the review scores which Yelp assigns to target products.

Acknowledgments

The corresponding author is Yue Zhang. We are grateful for the help of Xuanyi Li for his initial exploration. We thank our anonymous reviewers for their constructive comments, which helped to improve the paper. This work is supported by the Temasek Lab grant IGDST1403012 at Singapore University of Technology and Design, and supported by the National Natural Science Foundation of China (No.61402314).

References

Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Hongshen Chen, Yue Zhang, and Qun Liu. 2016a. Neural network for heterogeneous annotations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 731–741.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016b. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1650–1659.

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

Christian Desrosiers and George Karypis. 2011. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender Systems Handbook*, pages 107–144.

Chris Ding, Tao Li, Wei Peng, and Haesun Park. 2006. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 126–135. ACM.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics*, pages 340–348. Association for Computational Linguistics.

Wenliang Gao, Naoki Yoshinaga, Nobuhiro Kaji, and Masaru Kitsuregawa. 2013. Modeling user leniency and product popularity for sentiment classification. In *IJCNLP*, pages 1107–1111.

Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 549–558.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4):949–998.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pages 168–177.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *EMNLP*, pages 720–728.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single and cross-domain setting with conditional random fields. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1035–1045.

Paul B Kantor, Lior Rokach, Francesco Ricci, and Bracha Shapira. 2011. Recommender systems handbook.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1378–1387.

Cane WK Leung, Stephen CF Chan, and Fu-lai Chung. 2006. Integrating collaborative filtering and sentiment analysis: A rating inference approach. In *Proceedings of the ECAI 2006 workshop on recommender systems*, pages 62–66.

Fangtao Li, Nathan Nan Liu, Hongwei Jin, Kai Zhao, Qiang Yang, and Xiaoyan Zhu. 2011. Incorporating reviewer and product information for review rating prediction. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 1820–1825.

Fangtao Li, Sheng Wang, Shenghua Liu, and Ming Zhang. 2014. Suit: A supervised user-item based topic model for sentiment analysis. In *AAAI*, pages 1636–1642.

Tao Li, Yi Zhang, and Vikas Sindhwani. 2009. A non-negative matrix tri-factorization approach to sentiment classification with lexical prior knowledge. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 244–252. Association for Computational Linguistics.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 375–384.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Pengfei Liu, Shafiq R. Joty, and Helen M. Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1433–1443.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Hitoshi Nishikawa, Takaaki Hasegawa, Yoshihiro Matsuo, and Gen-ichiro Kikui. 2010. Optimizing informativeness and readability for sentiment summarization. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, Short Papers*, pages 325–330.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 115–124.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27.

Lizhen Qu, Georgiana Ifrim, and Gerhard Weikum. 2010. The bag-of-opinions method for review rating prediction from sparse text patterns. In *COLING*

2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China, pages 913–921.

Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. 2011. *Recommender Systems Handbook*. Springer.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 379–389.

Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 285–295.

Vivek Kumar Singh, Mousumi Mukherjee, and Ghanshyam Kumar Mehta. 2011. Combining collaborative filtering and sentiment classification for improved movie recommendations. In *Multi-disciplinary Trends in Artificial Intelligence - 5th International Workshop, MIWAI 2011, Hyderabad, India, December 7-9, 2011. Proceedings*, pages 38–50.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL.*

Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1556–1566.

Duyu Tang, Bing Qin, Ting Liu, and Yuekui Yang. 2015. User modeling with neural network for review rating prediction. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1340–1346.

Zhiyang Teng, Duy-Tin Vo, and Yue Zhang. 2016. Context-sensitive lexicon features for neural sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1629–1638.

Ke M. Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 321–331.

Xiaojun Wan. 2013. Co-regression for cross-language review rating prediction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers*, pages 526–531.

Lu Wang and Wang Ling. 2016. Neural network-based abstract generation for opinions and arguments. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 47–57.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2016. Memory-enhanced decoder for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 278–286.

Yao Wu and Martin Ester. 2015. FLAME: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*, pages 199–208.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 2397–2406.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association* *for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 1640–1649.

Xiwang Yang, Yang Guo, Yong Liu, and Harald Steck. 2014. A survey of collaborative filtering based social recommender systems. *Computer Communications*, 41:1–10.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL 2016, 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, US*.

Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, pages 83–92.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

CRF Autoencoder for Unsupervised Dependency Parsing*

Jiong Cai, Yong Jiang and **Kewei Tu**
{caijiong,jiangyong, tukw}@shanghaitech.edu.cn
School of Information Science and Technology
ShanghaiTech University, Shanghai, China

Abstract

Unsupervised dependency parsing, which tries to discover linguistic dependency structures from unannotated data, is a very challenging task. Almost all previous work on this task focuses on learning generative models. In this paper, we develop an unsupervised dependency parsing model based on the CRF autoencoder. The encoder part of our model is discriminative and globally normalized which allows us to use rich features as well as universal linguistic priors. We propose an exact algorithm for parsing as well as a tractable learning algorithm. We evaluated the performance of our model on eight multilingual treebanks and found that our model achieved comparable performance with state-of-the-art approaches.

1 Introduction

Unsupervised dependency parsing, which aims to discover syntactic structures in sentences from unlabeled data, is a very challenging task in natural language processing. Most of the previous work on unsupervised dependency parsing is based on generative models such as the dependency model with valence (DMV) introduced by Klein and Manning (2004). Many approaches have been proposed to enhance these generative models, for example, by designing advanced Bayesian priors (Cohen et al., 2008), representing dependencies with features (Berg-Kirkpatrick et al., 2010), and representing discrete tokens with continuous vectors (Jiang et al., 2016).

Besides generative approaches, Grave and Elhadad (2015) proposed an unsupervised discriminative parser. They designed a convex quadratic objective function under the discriminative clustering framework. By utilizing global features and linguistic priors, their approach achieves state-of-the-art performance. However, their approach uses an approximate parsing algorithm, which has no theoretical guarantee. In addition, the performance of the approach depends on a set of manually specified linguistic priors.

Conditional random field autoencoder (Ammar et al., 2014) is a new framework for unsupervised structured prediction. There are two components of this model: an encoder and a decoder. The encoder is a globally normalized feature-rich CRF model predicting the conditional distribution of the latent structure given the observed structured input. The decoder of the model is a generative model generating a transformation of the structured input from the latent structure. Ammar et al. (2014) applied the model to two sequential structured prediction tasks, part-of-speech induction and word alignment and showed that by utilizing context information the model can achieve better performance than previous generative models and locally normalized models. However, to the best of our knowledge, there is no previous work applying the CRF autoencoder to tasks with more complicated outputs such as tree structures.

In this paper, we propose an unsupervised discriminative dependency parser based on the CRF autoencoder framework and provide tractable algorithms for learning and parsing. We performed experiments in eight languages and show that our approach achieves comparable results with previous state-of-the-art models.

*This work was supported by the National Natural Science Foundation of China (61503248).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1638–1643
Copenhagen, Denmark, September 7–11, 2017.

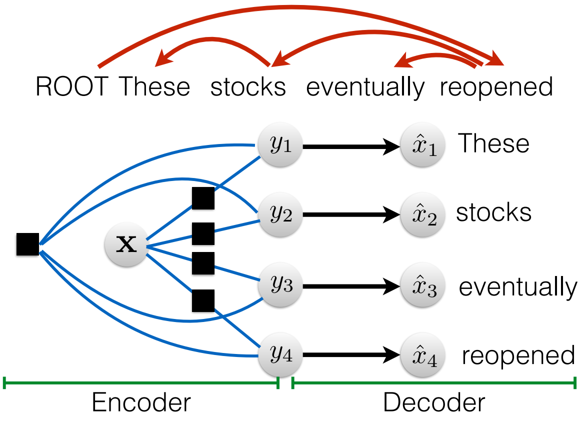


Figure 1: The CRF Autoencoder for the input sentence "These stocks eventually reopened" and its corresponding parse tree (shown at the top). $\mathbf{x}$ and $\hat{\mathbf{x}}$ are the original and reconstructed sentence. $\mathbf{y}$ is the dependency parse tree represented by a sequence where y_i contains the token and index of the parent of token x_i in the parse tree, e.g., $y_1 = \langle \text{stocks}, 2\rangle$ and $y_2 = \langle \text{reopened}, 4\rangle$. The encoder is represented by a factor graph (with a global factor specifying valid parse trees) and the decoder is represented by a Bayesian net.

2 Method

2.1 Model

Figure 1 shows our model with an example input sentence. Given an input sentence $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, we regard its parse tree as the latent structure represented by a sequence $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ where y_i is a pair $\langle t_i, h_i\rangle$, t_i is the head token of the dependency connecting to token x_i in the parse tree, and h_i is the index of this head token in the sentence. The model also contains a reconstruction output, which is a token sequence $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$. Throughout this paper, we set $\hat{\mathbf{x}} = \mathbf{x}$.

The encoder in our model is a log-linear model represented by a first-order dependency parser. The score of a dependency tree can be factorized as the sum of scores of its dependencies. For each dependency arc $(\mathbf{x}, i, j)$, where i and j are the indices of the head and child of the dependency, a feature vector $\mathbf{f}(\mathbf{x}, i, j)$ is specified. The score of a dependency is defined as the inner product of the feature vector and a weight vector $\mathbf{w}$,

$$\phi(\mathbf{x}, i, j) = \mathbf{w}^T \mathbf{f}(\mathbf{x}, i, j)$$

The score of a dependency tree $\mathbf{y}$ of sentence $\mathbf{x}$ is

$$\phi(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \phi(\mathbf{x}, h_i, i)$$

We define the probability of parse tree $\mathbf{y}$ given sentence $\mathbf{x}$ as

$$P(\mathbf{y}|\mathbf{x}) = \frac{\exp(\phi(\mathbf{x}, \mathbf{y}))}{Z(\mathbf{x})}$$

$Z(\mathbf{x})$ is the partition function,

$$Z(\mathbf{x}) = \sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{x})} \exp(\phi(\mathbf{x}, \mathbf{y}'))$$

where $\mathcal{Y}(\mathbf{x})$ is the set of all valid parse trees of $\mathbf{x}$. The partition function can be efficiently computed in $O(n^3)$ time using a variant of the inside-outside algorithm (Paskin, 2001) for projective tree structures, or using the Matrix-Tree Theorem for non-projective tree structures (Koo et al., 2007).

The decoder of our model consists of a set of categorical conditional distributions $\theta_{x|t}$, which represents the probability of generating token x conditioned on token t. So the probability of the reconstruction output $\hat{\mathbf{x}}$ given the parse tree $\mathbf{y}$ is

$$P(\hat{\mathbf{x}}|\mathbf{y}) = \prod_{i=1}^{n} \theta_{\hat{x}_i|t_i}$$

The conditional distribution of $\hat{\mathbf{x}}, \mathbf{y}$ given $\mathbf{x}$ is

$$P(\mathbf{y}, \hat{\mathbf{x}}|\mathbf{x}) = P(\mathbf{y}|\mathbf{x})P(\hat{\mathbf{x}}|\mathbf{y})$$

2.1.1 Features

Following McDonald et al. (2005) and Grave et al. (2015), we define the feature vector of a dependency based on the part-of-speech tags (POS) of the head, child and context words, the direction, and the distance between the head and child of the dependency. The feature template used in our parser is shown in Table 1.

2.1.2 Parsing

Given parameters $\mathbf{w}$ and $\boldsymbol{\theta}$, we can parse a sentence $\mathbf{x}$ by searching for a dependency tree $\mathbf{y}$ which has the highest probability $P(\hat{\mathbf{x}}, \mathbf{y}|\mathbf{x})$.

$$\begin{aligned}\mathbf{y}^* &= \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \log P(\hat{\mathbf{x}}, \mathbf{y}|\mathbf{x}) \\ &= \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \sum_{i=1}^{n} \left(\phi(\mathbf{x}, h_i, i) + \log \theta_{\hat{x}_i|t_i}\right)\end{aligned}$$

$\text{POS}_i \times dis \times dir$
$\text{POS}_j \times dis \times dir$
$\text{POS}_i \times \text{POS}_j \times dis \times dir$
$\text{POS}_i \times \text{POS}_{i-1} \times \text{POS}_j \times dis \times dir$
$\text{POS}_i \times \text{POS}_{i+1} \times \text{POS}_j \times dis \times dir$
$\text{POS}_i \times \text{POS}_j \times \text{POS}_{j-1} \times dis \times dir$
$\text{POS}_i \times \text{POS}_j \times \text{POS}_{j+1} \times dis \times dir$

Table 1: Feature template of a dependency, where i is the index of the head, j is the index of the child, $dis = |i - j|$, and dir is the direction of the dependency.

For projective dependency parsing, we can use Eisners algorithm (1996) to find the best parse in $O(n^3)$ time. For non-projective dependency parsing, we can use the Chu-Liu/Edmond algorithm (Chu and Liu, 1965; Edmonds, 1967; Tarjan, 1977) to find the best parse in $O(n^2)$ time.

2.2 Parameter Learning

2.2.1 Objective Function

Spitkovsky et al. (2010) shows that Viterbi EM can improve the performance of unsupervised dependency parsing in comparison with EM. Therefore, instead of using negative conditional log likelihood as our objective function, we choose to use negative conditional Viterbi log likelihood,

$$-\sum_{i=1}^{N}\log\left(\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x}_i)} P(\hat{\mathbf{x}}_i,\mathbf{y}|\mathbf{x}_i)\right)+\lambda\Omega(\mathbf{w}) \tag{1}$$

where $\Omega(\mathbf{w})$ is a L1 regularization term of the encoder parameter $\mathbf{w}$ and λ is a hyper-parameter controlling the strength of regularization.

To encourage learning of dependency relations that satisfy universal linguistic knowledge, we add a soft constraint on the parse tree based on the universal syntactic rules following Naseem et al. (2010) and Grave et al. (2015). Hence our objective function becomes

$$-\sum_{i=1}^{N}\log\left(\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x}_i)} P(\hat{\mathbf{x}}_i,\mathbf{y}|\mathbf{x}_i)Q^{\alpha}(\mathbf{x}_i,\mathbf{y})\right)+\lambda\Omega(\mathbf{w})$$

where $Q(\mathbf{x},\mathbf{y})$ is a soft constraint factor over the parse tree, and α is a hyper-parameter controlling the strength of the constraint factor. The factor Q is also decomposable by edges in the same way as the encoder and the decoder, and therefore our parsing algorithm can still be used with this factor

VERB → VERB	NOUN → NOUN
VERB → NOUN	NOUN → ADJ
VERB → PRON	NOUN → DET
VERB → ADV	NOUN → NUM
VERB → ADP	NOUN → CONJ
ADJ → ADV	ADP → NOUN

Table 2: Universal linguistic rules

taken into account.

$$Q(\mathbf{x},\mathbf{y})=\exp\left(\sum_i \mathbb{1}[(t_i\rightarrow x_i)\in\mathcal{R}]\right)$$

where $\mathbb{1}[(t_i \rightarrow x_i)\in\mathcal{R}]$ is an indicator function of whether dependency $t_i \rightarrow x_i$ satisfies one of the universal linguistic rules in $\mathcal{R}$. The universal linguistic rules that we use are shown in Table 2 (Naseem et al., 2010).

2.2.2 Algorithm

We apply coordinate descent to minimize the objective function, which alternately updates $\mathbf{w}$ and $\boldsymbol{\theta}$. In each optimization step of $\mathbf{w}$, we run two epochs of stochastic gradient descent, and in each optimization step of $\boldsymbol{\theta}$, we run two iterations of the Viterbi EM algorithm.

To update $\mathbf{w}$ using stochastic gradient descent, for each sentence $\mathbf{x}$, we first run the parsing algorithm to find the best parse tree $\mathbf{y}^* = \arg\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})}(P(\hat{\mathbf{x}},\mathbf{y}|\mathbf{x})Q^{\alpha}(\mathbf{x},\mathbf{y}))$; then we can calculate the gradient of the objective function based on the following derivation,

$$\begin{aligned}
&\frac{\partial \log P(\hat{\mathbf{x}},\mathbf{y}^*|\mathbf{x})}{\partial \mathbf{w}} \\
&= \frac{\partial \log P(\mathbf{y}^*|\mathbf{x})}{\partial \mathbf{w}} + \frac{\partial \log P(\hat{\mathbf{x}}|\mathbf{y}^*)}{\partial \mathbf{w}} \\
&= \frac{\partial \log P(\mathbf{y}^*|\mathbf{x})}{\partial \mathbf{w}} \\
&= \frac{\partial\left(\sum_{i=1}^{n}\mathbf{w}^T\mathbf{f}(\mathbf{x},h_i,i)-\log Z(\mathbf{x})\right)}{\partial \mathbf{w}} \\
&= \sum_{(i,j)\in\mathbf{D}(\mathbf{x})} f(\mathbf{x},i,j)\left(\mathbb{1}[y_j^*=\langle i,x_i\rangle]-\mu(\mathbf{x},i,j)\right)
\end{aligned}$$

where $\mathbf{D}(\mathbf{x})$ is the set of all possible dependency arcs of sentence $\mathbf{x}$, $\mathbb{1}[\cdot]$ is the indicator function, and $\mu(\mathbf{x},i,j)$ is the expected count defined as follows,

$$\mu(\mathbf{x},i,j)=\sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})}\left(\mathbb{1}[y_j=\langle i,x_i\rangle]P(\mathbf{y}|\mathbf{x})\right)$$

	Basque	Czech	Danish	Dutch	Portuguese	Slovene	Swedish	Avg
Length: $\leq$ 10								
DMV(EM)	41.1	31.3	50.8	47.1	36.7	36.7	43.5	41.0
DMV(Viterbi)	47.1	27.1	39.1	37.1	32.3	23.7	42.6	35.5
Neural DMV (EM)	46.5	33.1	**55.6**	**49.0**	30.4	42.2	44.3	43.0
Neural DMV (Viterbi)	48.1	28.6	39.8	37.2	36.5	39.9	47.9	39.7
Convex-MST (No Prior)	29.4	36.5	49.3	31.3	46.4	33.7	35.5	37.4
Convex-MST (With Prior)	30.0	46.1	51.6	35.3	55.4	**63.7**	50.9	47.5
CRFAE (No Prior)	49.0	33.9	28.8	39.3	47.6	34.7	51.3	40.6
CRFAE (With Prior)	**49.9**	**48.1**	53.4	43.9	**68.0**	52.5	**64.7**	**54.3**
Length: All								
DMV(EM)	31.2	28.1	40.3	44.2	23.5	25.2	32.0	32.0
DMV(Viterbi)	40.9	20.4	32.6	33.0	26.9	16.5	36.2	29.5
Neural DMV (EM)	38.5	29.3	**46.1**	**46.2**	16.2	36.6	32.8	35.1
Neural DMV (Viterbi)	**41.8**	23.8	34.2	33.6	29.4	30.8	40.2	33.4
Convex-MST (No Prior)	30.5	33.4	44.2	29.3	38.3	32.2	28.3	33.7
Convex-MST (With Prior)	30.6	**40.0**	45.8	35.6	46.3	**51.8**	40.5	41.5
CRFAE (No Prior)	39.8	25.4	24.2	35.2	52.2	26.4	40.0	34.7
CRFAE (With Prior)	41.4	36.8	40.5	38.6	**58.9**	43.3	**48.5**	**44.0**

Table 4: Parsing accuracy on seven languages. Our model is compared with DMV (Klein and Manning, 2004), Neural DMV (Jiang et al., 2016), and Convex-MST (Grave and Elhadad, 2015)

Methods	**WSJ10**	**WSJ**
Basic Setup		
Feature DMV (Berg-Kirkpatrick et al., 2010)	63.0	-
UR-A E-DMV (Tu and Honavar, 2012)	71.4	57.0
Neural E-DMV (Jiang et al., 2016)	69.7	52.5
Neural E-DMV (Good Init) (Jiang et al., 2016)	72.5	57.6
Basic Setup + Universal Linguistic Prior		
Convex-MST (Grave and Elhadad, 2015)	60.8	48.6
HDP-DEP (Naseem et al., 2010)	71.9	-
CRFAE	71.7	55.7
Systems Using Extra Info		
LexTSG-DMV (Blunsom and Cohn, 2010)	67.7	55.7
CS (Spitkovsky et al., 2013)	72.0	64.4
MaxEnc (Le and Zuidema, 2015)	73.2	65.8

Table 3: Comparison of recent unsupervised dependency parsing systems on English. Basic setup is the same as our setup except that linguistic prior is not used. Extra info includes lexicalization, longer training sentences, etc.

The expected count can be efficiently computed using the Matrix-Tree Theorem (Koo et al., 2007) for non-projective tree structures or using a variant of the inside-outside algorithm for projective tree structures (Paskin, 2001).

To update $\boldsymbol{\theta}$ using Viterbi EM, in the E-step we again use the parsing algorithm to find the best parse tree $\mathbf{y}^*$ for each sentence $\mathbf{x}$; then in the M-step we update $\boldsymbol{\theta}$ by maximum likelihood estimation.

$$\theta_{c|t} = \frac{\sum_{\mathbf{x}} \sum_i \mathbb{1}[x_i = c, y_i^* = \langle \cdot, t \rangle]}{\sum_{c'} \sum_{\mathbf{x}} \sum_i \mathbb{1}[x_i = c', y_i^* = \langle \cdot, t \rangle]}$$

3 Experiments

3.1 Setup

We experimented with projective parsing and used the informed initialization method proposed by Klein and Manning (2004) to initialize our model before learning. We tested our model both with and without using the universal linguistic rules. We used AdaGrad to optimize $\mathbf{w}$. We used POS tags of the input sentence as the tokens in our model. We learned our model on training sentences of length $\leq$ 10 and reported the directed dependency accuracy on test sentences of length $\leq$ 10 and on all test sentences.

3.2 Results on English

We evaluated our model on the Wall Street Journal corpus. We trained our model on section 2-21, tuned the hyperparameters on section 22, and tested our model on section 23. Table 3 shows the directed dependency accuracy of our model (CRFAE) compared with recently published results. It can be seen that our method achieves a comparable performance with state-of-the-art systems.

We also compared the performances of CRF autoencoder using an objective function with negative log likelihood vs. using our Viterbi version of the objective function (Eq.1). We find that the Viterbi version leads to much better performance (55.7 vs. 41.8 in parsing accuracy of WSJ), which echoes Spitkovsky et al. 's findings on Viterbi EM (2010).

3.3 Multilingual Results

We evaluated our model on seven languages from the PASCAL Challenge on Grammar Induction (Gelling et al., 2012). We did not use the Arabic corpus because the number of training sentences with length ≤ 10 is less than 1000. The result is shown in Table 4. The accuracies of DMV and Neural DMV are from Jiang et.al (2016). Both our model (CRFAE) and Convex-MST were tuned on the validation set of each corpus. It can be seen that our method achieves the best results on average. Besides, our method outperforms Convex-MST both with and without linguistic prior. From the results we can also see that utilizing universal linguistic prior greatly improves the performance of Convex-MST and our model.

4 Conclusion

In this paper, we propose a new discriminative model for unsupervised dependency parsing based on CRF autoencoder. Both learning and inference of our model are tractable. We tested our method on eight languages and show that our model is competitive to the state-of-the-art systems.

The code is available at `https://github.com/caijiong/CRFAE-Dep-Parser`

References

Waleed Ammar, Chris Dyer, and Noah A Smith. 2014. Conditional random field autoencoders for unsupervised structured prediction. In *Advances in Neural Information Processing Systems*, pages 3311–3319.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590. Association for Computational Linguistics.

Phil Blunsom and Trevor Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1204–1213. Association for Computational Linguistics.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396.

Shay B Cohen, Kevin Gimpel, and Noah A Smith. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In *Advances in Neural Information Processing Systems*, pages 321–328.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345. Association for Computational Linguistics.

Douwe Gelling, Trevor Cohn, Phil Blunsom, and Joao Graça. 2012. The pascal challenge on grammar induction. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 64–80. Association for Computational Linguistics.

Edouard Grave and Noémie Elhadad. 2015. A convex and feature-rich discriminative approach to dependency grammar induction. In *ACL (1)*, pages 1375–1384.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 763–771, Austin, Texas. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 478. Association for Computational Linguistics.

Terry Koo, Amir Globerson, Xavier Carreras Pérez, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150.

Phong Le and Willem Zuidema. 2015. Unsupervised dependency parsing: Let's use supervised parsers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 651–661, Denver, Colorado. Association for Computational Linguistics.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 91–98. Association for Computational Linguistics.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244. Association for Computational Linguistics.

Mark A Paskin. 2001. *Cubic-time parsing and learning algorithms for grammatical bigram models*. Citeseer.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *EMNLP*, pages 1983–1995.

Valentin I Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D Manning. 2010. Viterbi training improves unsupervised dependency parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 9–17. Association for Computational Linguistics.

Robert Endre Tarjan. 1977. Finding optimum branchings. *Networks*, 7(1):25–35.

Kewei Tu and Vasant Honavar. 2012. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1324–1334. Association for Computational Linguistics.

Efficient Discontinuous Phrase-Structure Parsing via the Generalized Maximum Spanning Arborescence

Caio Corro   **Joseph Le Roux**   **Mathieu Lacroix**
Laboratoire d'Informatique de Paris Nord,
Université Paris 13 – SPC, CNRS UMR 7030,
F-93430, Villetaneuse, France
{corro,leroux,lacroix}@lipn.fr

Abstract

We present a new method for the joint task of tagging and non-projective dependency parsing. We demonstrate its usefulness with an application to discontinuous phrase-structure parsing where decoding lexicalized spines and syntactic derivations is performed jointly. The main contributions of this paper are (1) a reduction from joint tagging and non-projective dependency parsing to the Generalized Maximum Spanning Arborescence problem, and (2) a novel decoding algorithm for this problem through Lagrangian relaxation. We evaluate this model and obtain state-of-the-art results despite strong independence assumptions.

1 Introduction

Discontinuous phrase-structure parsing relies either on formal grammars such as LCFRS, which suffer from a high complexity, or on reductions to non-projective dependency parsing with complex labels to encode phrase combinations. We propose an alternative approach based on a variant of spinal TAGs, which allows parses with discontinuity while grounding this work on a lexicalized phrase-structure grammar. Contrarily to previous approaches, (Hall and Nivre, 2008; Versley, 2014; Fernández-González and Martins, 2015), we do not model supertagging nor spine interactions with a complex label scheme. We follow Carreras et al. (2008) but drop projectivity.

We first show that our discontinuous variant of spinal TAG reduces to the Generalized Maximum Spanning Arborescence (GMSA) problem (Myung et al., 1995). In a graph where vertices are partitioned into clusters, GMSA consists in finding the arborescence of maximum weight incident to exactly one vertex per cluster. This problem is NP-complete even for arc-factored models. In order to bypass complexity, we resort to Lagrangian relaxation and propose an efficient resolution based on dual decomposition which combines a simple non-projective dependency parser on a contracted graph and a local search on each cluster to find a global consensus.

We evaluated our model on the discontinuous PTB (Evang and Kallmeyer, 2011) and the Tiger (Brants et al., 2004) corpora. Moreover, we show that our algorithm is able to quickly parse the whole test sets.

Section 2 presents the parsing problem. Section 3 introduces GMSA from which we derive an effective resolution method in Section 4. In Section 5 we define a parameterization of the parser which uses neural networks to model local probabilities and present experimental results in Section 6. We discuss related work in Section 7.

2 Joint Supertagging and Spine Parsing

In this section we introduce our problem and set notation. The goal of phrase-structure parsing is to produce a derived tree by means of a sequence of operations called a derivation. For instance in context-free grammars the derived tree is built from a sequence of substitutions of a non-terminal symbol with a string of symbols, whereas in tree adjoining grammars (TAGs) a derivation is a sequence of substitutions and adjunctions over elementary trees. We are especially interested in building discontinuous phrase-structure trees which may contain constituents with gaps.[1]

We follow Shen (2006) and build derived trees from adjunctions performed on spines. Spines are lexicalized unary trees where each level represents

[1] Although we will borrow concepts from TAGs, we do not require derivations to be TAG compatible (i.e. well-nested dependencies with a bounded number of gaps).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1644–1654
Copenhagen, Denmark, September 7–11, 2017.

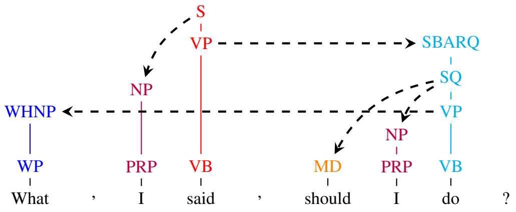


Figure 1: A derivation with spines and adjunctions (dashed arrows). The induced dependency tree is non-projective. Each color corresponds to a spine. We omit punctuation to simplify figures.

a lexical projection of the anchor. Carreras et al. (2008) showed how spine-based parsing could be reduced to dependency parsing: since spines are attached to words, equivalent derivations can be represented as a dependency tree where arcs are labeled by spine operations, an adjunction together with information about the adjunction site. However, we depart from previous approaches (Shen and Joshi, 2008; Carreras et al., 2008) by relaxing the projectivity constraint to represent all discontinuous phrase-structure trees (see Figure 1).

We assume a finite set of spines S. A spine s can be defined as a sequence of grammatical categories, beginning at root. For a sentence $\mathbf{w} = (w_0, w_1, \ldots, w_n)$ where w_k is the word at position k and w_0 is a dummy root symbol, a derivation is a triplet $(\mathbf{d}, \mathbf{s}, \mathbf{l})$ defined as follows. Adjunctions are described by a dependency tree rooted at 0 written as a sequence of arcs $\mathbf{d}$. If $(h, m) \in \mathbf{d}$ with $h \in \{0, \ldots, n\}$ and $m \in \{1, \ldots, n\}$, then the derivation contains an adjunction of the root of the spine at position m to a node from the spine at position h. Supertagging, the assignment of a spine to each word, is represented by a sequence $\mathbf{s} = (s_0, s_1, \ldots, s_n)$ of $n + 1$ spines, each spine s_k being assigned to word w_k. Finally, labeling $\mathbf{l} = (l_1, \ldots, l_n)$ is a sequence where l_k is the label of the k^{th} arc (h, m) of $\mathbf{d}$. The label consists of a couple (op, i) where op is the type of adjunction, here *sister* or *regular*[2], and i is the index of the adjunction node in s_h.

Each derivation is assigned an arc-factored score σ which is given by:

$$\sigma(\mathbf{d}, \mathbf{s}, \mathbf{l}; \mathbf{w}) = \sum_{(h,m) \in \mathbf{d}} \Omega(h, m, s_h, s_m, l_{hm}; \mathbf{w})$$

For instance, following score functions developed in (Carreras et al., 2008), this function could read $s_h[i]$, $s_h[i+1]$ and $s_m[0]$, where $s[i]$ denotes the i-th grammatical category of the spine s. The score of the derivation in Figure 1 could then reflect that the spine `WHNP-WP` associated with *What* is adjoined on the spine `SBARQ-SQ-VP-VB` associated with *do* on a site with the grammatical triple `[VP WHNP VB]`.

We assume that Ω accounts for the contribution of arcs, spines and labels to the score. The details of the contribution depend on the model. We choose the following:

$$\begin{aligned}\sigma(\mathbf{d}, \mathbf{s}, \mathbf{l}; \mathbf{w}) = \sum_{(h,m) \in \mathbf{d}} & (\alpha(h, m; \mathbf{w}) \\ & +\nu(s_m; h, m, \mathbf{w}) \\ & +\gamma(l_{hm}; h, m, s_h, \mathbf{w}))\end{aligned}$$

where α is the score related to the dependency tree, ν is the supertagging score and γ the labeling score. Note that functions α, ν and γ have access to the entire input string $\mathbf{w}$. Score function σ can be parameterized in many ways and we discuss our implementation in Section 5. In this setting, parsing a sentence $\mathbf{w}$ amounts to finding the highest-scoring derivation $(\mathbf{d}^*, \mathbf{s}^*, \mathbf{l}^*) = \arg\max_{(\mathbf{d},\mathbf{s},\mathbf{l})} \sigma(\mathbf{d}, \mathbf{s}, \mathbf{l}; \mathbf{w})$.

Recovering the derived tree from a derivation is performed by recursively mapping each spine and its dependencies to a possibly gappy constituent. Given a spine s_h and site index i, we look for the leftmost s_l and rightmost s_r dependents attached with regular adjunction. If any, we insert a new node between $s_h[i]$ and $s_h[i+1]$ with the same grammatical category as the first one. This new node fills the role of the foot node in TAGs. Every dependent of $s_h[i]$ with anchor in interval $[l+1, r-1]$ is moved to the newly created node. Remaining sister and regular adjunctions are simply attached to $s_h[i]$.

The complexity of the parsing problem depends on the type of dependency trees. In the case of projective trees, it has been shown (Eisner, 2000; Carreras et al., 2008; Li et al., 2011) that this could be performed in cubic worst-case time complexity with dynamic programming, whether supertags are fixed beforehand or not. However, the modification of the original Eisner algorithm requires that chart cells must be indexed not only by spans, or pairs of positions, but also by pairs of supertags. In practice the problem is intractable unless heavy

[2] The distinction is not crucial for the exposition. We refer readers to (Shen and Joshi, 2008; Carreras et al., 2008).

pruning is performed first in order to select a subset of spines at each position.

In the case of non-projective dependency trees, the problem has quadratic worst-case time complexity when supertags are fixed, since the problem then amounts to non-projective parsing and reduces to the Maximum Spanning Arborescence problem (MSA) as in (McDonald et al., 2005). Unfortunately, the efficient algorithm for MSA is greedy and does not *store* potential substructure candidates. Hence, when supertags are not fixed beforehand, a new arborescence must be recomputed for each choice of supertags. This problem can be seen as instance of the Generalized Maximum Spanning Arborescence problem, an NP-complete problem, which we review in the next section. Note that arc labels do not impact the asymptotic complexity of an arc-factored model. Indeed, only the labeled arc with maximum weight between two vertices is considered when parsing.

3 The Generalized Maximum Spanning Arborescence

In this section, we first define GMSA introduced by Myung et al. (1995). We formulate this problem as an integer linear program. We then explain the reduction from the joint supertagging and spine parsing task to this problem.[3]

3.1 Problem definition

Let $D = (V, A)$ be a directed graph. Given a subset $T \subseteq A$ of arcs, $V[T]$ denotes the set of vertices of V which are the tail or the head of at least one arc of T. These vertices are said to *be covered* by T. A subset $T \subseteq A$ of arcs is called an *arborescence* if the graph $(V[T], T)$ is connected, acyclic and each vertex has at most one entering arc. The vertex with no entering arc is called the *root* of T. An arborescence covering all vertices is called a *spanning arborescence*.

Let $\pi = \{V_0, \dots, V_n\}$, $n \in \mathbb{N}$ be a partition of V. Each element of π is called a *cluster*. An arborescence T of D covering exactly one vertex per cluster of π is called a *generalized spanning arborescence* (GSA). Figure 2 gives an example of a GSA. The partition of V is composed of a cluster having one vertex and six clusters having four vertices. Each cluster is depicted by a hatched area. The GSA is depicted by the dashed arcs.

Let W be a vertex subset of V. We denote $\delta^-(W)$ (resp. $\delta^+(W)$) the set of arcs entering (resp. leaving) W and $\delta(W) = \delta^-(W) \cup \delta^+(W)$.[4] Contracting W consists in replacing in D all vertices in W by a new vertex w, replacing each arc $uv \in \delta^-(W)$ by the arc uw and each arc $vu \in \delta^+(W)$ by wu. Let D^π be the graph obtained by contracting each cluster of π in D. Note that a GSA of D and π induces a spanning arborescence of D^π.[5] For instance, contracting each cluster in the graph given by Figure 2 leads to a graph D^π having 7 vertices and the set of dashed arcs corresponds to a spanning arborescence of D^π.

Given arc weights $\phi \in \mathbb{R}^A$, the weight of an arborescence T is $\sum_{a \in T} \phi_a$. Given (D, π, ϕ), the *Generalized Maximum Spanning Arborescence problem (GMSA)* consists in finding a GSA of D and π of maximum weight whose root is in V_0.

3.2 Integer linear program

Given a set S, $z \in \mathbb{R}^S$ is a vector indexed by elements in S. For $S' \subseteq S$, $z(S') = \sum_{s \in S'} z_s$.

A GSA $T \subseteq A$ is represented by variables $x \in \{0,1\}^V$ and $y \in \{0,1\}^A$ such that x_v (resp. y_a) is equal to 1 iff $v \in V[T]$ (resp. $a \in T$).

Since a GSA of D and π induces a spanning arborescence of D^π, the arc-incidence vector $y \in \{0,1\}^A$ of a GSA with root in V_0 satisfies the following, adapted from MSA (Schrijver, 2003):

$$y(\delta^-(V_0)) = 0 \tag{1}$$

$$y(\delta^-(V_k)) = 1 \qquad \forall 1 \leq k \leq n, \tag{2}$$

$$y(\delta^-(\bigcup_{V_k \in \pi'} V_k)) \geq 1 \quad \forall \pi' \subseteq \pi \setminus \{V_0\}. \tag{3}$$

Let $\mathcal{Y}$ denote all the arc-incidence vectors on D corresponding to a spanning arborescence in D^π whose root is the contraction of V_0. Then,

$$\mathcal{Y} = \{y \in \{0,1\}^A | y \text{ satisfies (1)-(3)}\}.$$

GMSA can be formulated with the following integer linear program:

$$\max_{x,y} \quad \phi \cdot y \tag{4}$$

$$\text{s.t.} \quad y \in \mathcal{Y} \tag{5}$$

$$x_v \geq y_a \qquad \forall v \in V, a \in \delta(v), \tag{6}$$

$$x_v(V_k) = 1 \qquad \forall 0 \leq k \leq n, \tag{7}$$

$$x_v \in \{0,1\} \qquad \forall v \in V. \tag{8}$$

[3] A similar reduction can be obtained in the reverse direction, thus proving the NP-completeness of our problem.

[4] By an abuse of notation, we identify any singleton $\{v\}$ with its element v.

[5] The converse does not hold: an arc subset of A corresponding to a spanning arborescence of D^π may not be a GSA of D and π since it may not induce a connected graph.

Let W and T be the vertex and arc sets given by $x_v = 1$ and $y_a = 1$ respectively. Since T is a spanning arborescence of D^π by (5), $(V[T], T)$ is an acyclic directed graph with n arcs such that V_0 has no entering arc and V_i, $i \in \{1, \dots, n\}$, has one entering arc. By constraints (7), W contains one vertex per cluster of π. Moreover, by inequalities (6), $V[T] \subseteq W$. Since $|W| = n + 1$ and $|T| = n$, $W = V[T]$ and $(V[T], T)$ is connected, so it is a GSA. Because its root is in V_0 by (5), it is an optimal solution for GMSA by (4).

3.3 Reduction from joint parsing to GMSA

Given an instance of the joint parsing problem, we construct an instance of GMSA as follows. With every spine s of every word w_k different from w_0, we associate a vertex v. For $k = 1, \dots, n$, we denote by V_k the set of vertices associated with the spines of w_k. We associate with w_0 a set V_0 containing only one vertex and V_0 will now refer both the cluster and the vertex it contains depending on the context. Let $\pi = \{V_0, \dots, V_n\}$ and $V = \cup_{k=0}^{n} V_k$. For every couple u, v of vertices such that $u \in V_h$ and $v \in V_m$, $h \neq m$ and $m \neq 0$, we associate an arc uv corresponding to the best adjunction of the root of spine s_m associated with v of V_m to spine s_h associated with vertex u of V_h. The weight of this arc is given by

$$\phi_{uv} = \alpha(h, m; \mathbf{w}) + \nu(s_m; h, m, \mathbf{w}) + \max_{l_{hm}} \gamma(l_{hm}; h, m, s_h, \mathbf{w})$$

which is the score of the best adjunction of s_m to s_h. This ends the construction of (D, π, ϕ).

There is a 1-to-1 correspondence between the solutions to GMSA and those to the joint supertagging and spine parsing task in which each adjunction is performed with the label maximizing the score of the adjunction. Indeed, the vertices covered by a GSA T with root V_0 correspond to the spines on which the derivation is performed. By definition of GSAs, one spine per word is chosen. Each arc of T corresponds to an adjunction. The score of the arborescence is the sum of the scores of the selected spines plus the sum of the scores of the best adjunctions with respect to T. Hence, one can solve GMSA to perform joint parsing.

As an illustration, the GSA depicted in Figure 2 represents the derivation tree of Figure 1: the vertices of $V \setminus V_0$ covered by the GSA are those associated with the spines of Figure 1 and the arcs represent the different adjunctions. For instance the arc from V_3 to V_2 represents the adjunction of spine `NP-PRP` to spine `S-VP-VB` at index 0.

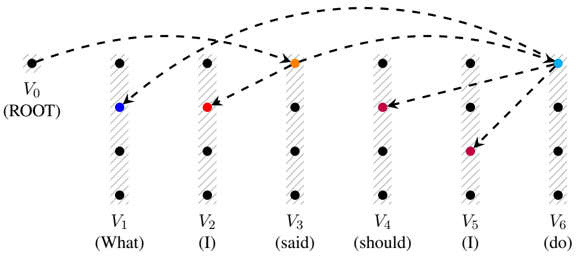


Figure 2: The generalized spanning arborescence inducing the derivation tree in Figure 1.

4 Efficient Decoding

Lagrangian relaxation has been successfully applied to various NLP tasks (Koo et al., 2010; Le Roux et al., 2013; Almeida and Martins, 2013; Das et al., 2012; Corro et al., 2016). Intuitively, given an integer linear program, it consists in relaxing some linear constraints which make the program difficult to solve and penalizing their violation in the objective function.

We propose a new decoding method for GMSA based on dual decomposition, a special flavor of Lagrangian relaxation where the problem is decomposed in several independent subproblems.

4.1 Dual decomposition

To perform the dual decomposition, we first reformulate the integer linear program (4)-(8) before relaxing linear constraints. For this purpose, we replace the variables y by three copies $\{y^i\} = \{y^0, y^1, y^2\}$, $y^i \in \{0,1\}^A$. We also consider variables $z \in \mathbb{R}^A$. Let ϕ^0, ϕ^1 and ϕ^2 be arc weight vectors such that $\sum_i \phi^i = \phi$.[6] GMSA can then be reformulated as:

$$\max_{x, \{y^i\}, z} \quad \sum_i \phi^i \cdot y^i \tag{9}$$

$$\text{s.t.} \quad y^0 \in \mathcal{Y} \tag{10}$$

$$x_v \geq y^1_a \qquad \forall v \in V, a \in \delta^-(v), \tag{11}$$

$$x_v \geq y^2_a \qquad \forall v \in V, a \in \delta^+(v), \tag{12}$$

$$x_v(V_k) = 1 \qquad \forall 0 \leq k \leq n, \tag{13}$$

$$x_v \in \{0,1\} \qquad \forall v \in V, \tag{14}$$

$$z = y^i \qquad \forall i. \tag{15}$$

[6] In our implementation, we choose $\phi^0 = \phi^1 = \phi^2 = \frac{1}{3}\phi$.

Note that variables z only appear in equations (15). Their goal is to ensure equality between copies y^0, y^1 and y^2. Variables z are usually called *witness variables* (Komodakis et al., 2007). Equality between y^0, y^1 and y^2 implies that (10)-(12) are equivalent to (5) and (6).

We now relax constraints (15) and build the dual objective (Lemaréchal, 2001) $\mathcal{L}^*(\{\lambda^i\})$:

$$\begin{aligned} \max_{x,\{y^i\},z} \quad & \sum_i \phi^i \cdot y^i + \sum_{i\in\{0,1,2\}} \lambda^i \cdot (z - y^i) \\ \text{s.t.} \quad & (10) - (14) \end{aligned}$$

where $\{\lambda^i\} = \{\lambda^0, \lambda^1, \lambda^2\}$, $\lambda^i \in \mathbb{R}^A$ for $i = 0, 1, 2$, is the set of Lagrangian multipliers. The dual problem is then:

$$\min_{\{\lambda^i\}} \mathcal{L}^*(\{\lambda^i\})$$

Note that, as there is no constraint on z, if $\sum_i \lambda^i \neq \mathbf{0}$ then $\mathcal{L}^*(\{\lambda^i\}) = +\infty$. Therefore, we can restrict the domain of $\{\lambda^i\}$ in the dual problem to the set $\Lambda = \{\{\lambda^i\} | \sum_i \lambda^i = \mathbf{0}\}$. This implies that z may be removed in the dual objective. This latter can be rewritten as:

$$\begin{aligned} \mathcal{L}^*(\{\lambda^i\}) = \max_{x,\{y^i\}} & \sum_i \bar{\phi}^i \cdot y^i \\ \text{s.t.} \quad & (10) - (14) \end{aligned}$$

where $\bar{\phi}^i = \phi^i - \lambda^i$ for $i = 0, 1, 2$.

4.2 Computing the dual objective

Given $\{\lambda^i\} \in \Lambda$, computing the dual objective $\mathcal{L}^*(\{\lambda^i\})$ can be done by solving the two following distinct subproblems:

$$\begin{aligned} P_1(\bar{\phi}^0) = & \max_{y^0} \bar{\phi}^0 \cdot y^0 \\ & \text{s.t.} \quad y^0 \in \mathcal{Y} \\ P_2(\bar{\phi}^1, \bar{\phi}^2) = & \max_{x,y^1,y^2} \bar{\phi}^1 \cdot y^1 + \bar{\phi}^2 \cdot y^2 \\ & \text{s.t.} \quad (11) - (14) \\ & \qquad y^i_a \in \{0, 1\} \qquad \forall a \in A, i = 1, 2. \end{aligned}$$

Subproblem P_1 can be solved by simply running the MSA algorithm on the contracted graph D^π.

Subproblem P_2 can be solved in a combinatorial way. Indeed, observe that each value of y^1 and y^2 is only constrained by a single value of x. The problem amounts to selecting for each cluster a vertex as well as all the arcs with positive weight covering it. More precisely, for each vertex $v \in V$, compute the *local weight* c_v defined by:

$$\sum_{a\in\delta^-(v)} \max\{0, \bar{\phi}^1\} + \sum_{a\in\delta^+(v)} \max\{0, \bar{\phi}^2\}.$$

Let $V^{\max}$ be the set of vertices defined as follows. For $k = 0, \ldots, n$, add in $V^{\max}$ the vertex $v \in V_k$ with the maximum weight c_v. Let A^1 and A^2 be the sets of arcs such that A^1 (resp. A^2) contains all the arcs with positive weights entering (resp. leaving) a vertex of $V^{\max}$. The vectors x, y^1 and y^2 corresponding respectively to the incidence vectors of $V^{\max}$, A^1 and A^2 form an optimal solution to P_2.

Hence, both supbroblems can be be solved with a $O(|\pi|^2)$ time complexity, that is quadratic w.r.t. the length of the input sentence.[7]

4.3 Decoding algorithm

Our algorithm seeks for a solution to GMSA by solving the dual problem since its solution is optimal to GMSA whenever it is a GSA. If not, a solution is constructed by returning the highest GSA on the spines computed during the resolution of the dual problem.

We solve the dual problem using a projected subgradient descent which consists in iteratively updating $\{\lambda^i\}$ in order to reduce the distance to the optimal assignment. Let $\{\lambda^{i,t}\}$ denotes the value of $\{\lambda^i\}$ at iteration t. $\{\lambda^{i,0}\}$ is initially set to $\mathbf{0}$. At each iteration, the value of $\{\lambda^{i,t+1}\}$ is computed from the value of $\{\lambda^{i,t}\}$ thanks to a subgradient of the dual objective. More precisely, we have

$$\{\lambda^{i,t+1}\} = \{\lambda^{i,t}\} - \eta^t \times \nabla\mathcal{L}^*(\{\lambda^{i,t}\})\}$$

where $\nabla\mathcal{L}^*(\{\lambda^{i,t}\})$ is a subgradient of $\mathcal{L}^*(\{\lambda^{i,t}\})$ and $\eta^t \in \mathbb{R}$ is the stepsize at iteration t. We use the projected subgradient from Komodakis et al. (2007). Hence, at iteration t, we must solve reparameterized subproblems P_1 and P_2 to obtain the current solution $(\bar{x}^t, \bar{y}^{0,t}, \bar{y}^{1,t}, \bar{y}^{2,t})$ of the dual objective. Then each multiplier is updated following

$$\lambda^{i,t+1} = \lambda^{i,t} - \eta^t \times \left(\bar{y}^{i,t} - \sum_{j=0}^{2} \frac{\bar{y}^{j,t}}{3} \right).$$

Note that for any value of $\{\lambda^i\}$, $\mathcal{L}^*(\{\lambda^i\})$ gives an upper bound for GMSA. So, whenever the

[7] In the general case, the time complexity is $O(|V|^2)$. But in our problem, the number of vertices per cluster is bounded by the grammar size: $O(|V|^2) = O(|S\pi|^2) = O(|\pi|^2)$.

optimal solution $\bar{x}^t, \{\bar{y}^{i,t}\}$ to the dual objective $\mathcal{L}^*(\{\lambda^{i,t}\})$ at iteration t is a primal feasible solution, that is $\bar{y}^{0,t} = \bar{y}^{1,t} = \bar{y}^{2,t}$, it is an optimal solution to GMSA and the algorithm ends. Otherwise, we construct a *pipeline solution* by performing a MSA on the vertices given by $\bar{x}^t$.

If after a fixed number of iterations we have not found an optimal solution to GMSA, we return the pipeline solution with maximum weight.

4.4 Lagrangian enhancement

The previsouly defined Lagrangian dual is valid but may lead to slow convergence. Thus, we propose three additional techniques which empirically improve the decoding time and the convergence rate: constraint tightening, arc reweighing and problem reduction.

Constraint tightening: In subproblem P_2, we consider a vertex and all of its adjacent arcs of positive weight. However, we know that our optimal solution must satisfy tree-shape constraints (5). Thus, every cluster except the root must have exactly one incoming arc and there is at most one arc between two clusters. Both constraints are added to P_2 without hurting its time complexity.

Reweighing: By modifying weights such that less incoming arcs have a positive weight, the solution of P_2 tends to be an arborescence. For each cluster $V_k \in \pi \setminus V_0$, let $\hat{A}_k$ be the set of incoming arcs with the highest weight $\hat{\phi}_k$. Then, let γ_k be a value such that $\phi_a - \gamma_k$ is positive only for arcs in $\hat{A}_k$. Subtracting γ_k from the weight ϕ_a of each arc of $\delta^-(V_k)$ and adding γ_k to the objective score does not modify the weight of the solution because only one entering arc per cluster is selected.

Problem reduction: We use the pipeline solutions computed at each iteration to set the value of some variables. Let $\bar{x}, \{\bar{y}^i\}$ be the optimal solution of $\mathcal{L}^*(\{\lambda^i\})$ computed at any iteration of the subgradient algorithm. For $k = 1, \dots, n$, let $\bar{v}$ be the vertex of V_k such that $\bar{x}_{\bar{v}} = 1$. Using the local weights (Section 4.2), for all $v \in V_k \setminus \{\bar{v}\}$, $\mathcal{L}^*(\{\lambda^i\}) + c_v - c_{\bar{v}}$ is an upper bound on the weight of any solution (x, y) to GMSA with $x_v = 1$. Hence, if it is lower than the weight of the best pipeline solution found so far, we can guarantee that $x_v = 0$ in any optimal solution. We can check the whole graph in linear time if we keep local weights c in memory.

5 Neural Parameterization

We present a probabilistic model for our framework. We implement our probability distributions with neural networks, more specifically we build a neural architecture on top of bidirectional recurrent networks that compute context sensitive representations of words. At each step, the recurrent architecture is given as input a concatenation of word and part-of-speech embeddings. We refer the reader to (Kiperwasser and Goldberg, 2016; Dozat and Manning) for further explanations about bidirectional LSTMs (Hochreiter and Schmidhuber, 1997). In the rest of this section, b_m denotes the context sensitive representation of word w_m.

We now describe the neural network models used to learn and assign weight functions α, ν and γ under a probabilistic model. Given a sentence $\mathbf{w}$ of length n, we assume a derivation $(\mathbf{d}, \mathbf{s}, \mathbf{l})$ is generated by three distinct tasks. By chain rule, $P(\mathbf{d}, \mathbf{s}, \mathbf{l}|\mathbf{w}) = P_\alpha(\mathbf{d}|\mathbf{w}) \times P_\nu(\mathbf{s}|\mathbf{d}, \mathbf{w}) \times P_\gamma(\mathbf{l}|\mathbf{d}, \mathbf{s}, \mathbf{w})$. We follow a common approach in dependency parsing and assign labels $\mathbf{l}$ in a post-processing step, although our model is able to incorporate label scores directly. Thus, we are left with jointly decoding a dependency structure and assigning a sequence of spines. We note s_i the i^{th} spine:[8]

$$\begin{aligned} & P_\alpha(\mathbf{d}|\mathbf{w}) \times P_\nu(\mathbf{s}|\mathbf{d}, \mathbf{w}) \\ &= \prod_{(h,m)\in\mathbf{d}} P_\alpha(h|m, \mathbf{w}) \times P_\nu(s_m|m, \mathbf{d}, \mathbf{w}) \\ &= \prod_{(h,m)\in\mathbf{d}} P_\alpha(h|m, \mathbf{w}) \times P_\nu(s_m|m, h, \mathbf{w}) \end{aligned}$$

We suppose that adjunctions are generated by an arc-factored model, and that a spine prediction depends on both current position and head position.

Then parsing amounts to finding the most probable derivation and can be realized in the log space, which gives following weight functions:

$$\begin{aligned} \alpha(h, m; \mathbf{w}) &= \log P_\alpha(h|m, \mathbf{w}) \\ \nu(s_m; h, m, \mathbf{w}) &= \log P_\nu(s_m|m, h, \mathbf{w}) \end{aligned}$$

where α represents the arc contribution and ν the spine contribution (cf. Section 2).

Word embeddings b_k are first passed through specific feed-forward networks depending on the

[8] We assume that the spine for the root w_0 is unique.

distribution and role. The result of the feed-forward transformation parameterized by set of parameters ρ of a word embedding b_s is a vector denoted $b_s^{(\rho)}$. We first define a biaffine attention networks weighting dependency relations (Dozat and Manning):

$$o^{(\alpha)}_{h,m} = b^{(\alpha_1)\top}_m W^{(\alpha)} b^{(\alpha_2)}_h + V^{(\alpha)} b^{(\alpha_2)}_h$$

where $W^{(\alpha)}$ and $V^{(\alpha)}$ are trainable parameters. Moreover, we define a biaffine attention classifier networks for class c as:

$$\begin{aligned} o^{(\tau)}_{c,h,m} &= b^{(\tau_1)\top}_m W^{(\tau_c)} b^{(\tau_2)}_h \\ &\quad + V^{(\tau_c)} \left(b^{(\tau_1)}_m \oplus b^{(\tau_2)}_h \right) \\ &\quad + u^{(\tau_c)} \end{aligned}$$

where $\oplus$ is the concatenation. $W^{(\tau_c)}$, $V^{(\tau_c)}$ and $u^{(\tau_c)}$ are trainable parameters. Then, we define the weight of assigning spine s to word at position m with head h as $o^{(\nu)}_{s,h,m}$.

Distributions P_α and P_ν are parameterized by these biaffine attention networks followed by a softmax layer:

$$P_\alpha(h|m,\mathbf{w}) = \frac{\exp o^{(\alpha)}_{h,m}}{\sum_{h'} \exp o^{(\alpha)}_{h',m}}$$

$$P_\nu(s|h,m,\mathbf{w}) = \frac{\exp o^{(\nu)}_{s,h,m}}{\sum_{s'} \exp o^{(\nu)}_{s',h,m}}$$

Now we move on to the post-processing step predicting arc labels. For each adjunction of spine s at position m to spine t at position h, instead of predicting a site index i, we predict the non-terminal nt at $t[i]$ with a biaffine attention classifier.[9] The probability of the adjunction of spine s at position m to a site labeled with nt on spine t at position h with type $a \in \{\text{regular}, \text{sister}\}$ is:

$$\begin{aligned} P_\gamma(nt, a|h,m) &= P_{\gamma'}(nt|h,m,\mathbf{w}) \\ &\quad \times P_{\gamma''}(a|h,m\mathbf{w}) \end{aligned}$$

P_γ and $P_{\gamma''}$ are again defined as distributions from the exponential family using biaffine attention classifiers:

$$P_{\gamma'}(nt|h,m,t) = \frac{\exp o^{(\gamma')}_{nt,h,m}}{\sum_{nt'} \exp o^{(\gamma')}_{nt,h,m}}$$

$$P_{\gamma''}(a|h,m,t) = \frac{\exp o^{(\gamma'')}_{t,h,m}}{\sum_{a'} \exp o^{(\gamma'')}_{a',h,m}}$$

We use embeddings of size 100 for words and size 50 for parts-of-speech tags. We stack two bidirectional LSTMs with a hidden layer of size 300, resulting in a context sensitive embedding of size 600. Embeddings are shared across distributions. All feed-forward networks have a unique elu-activated hidden layer of size 100 (Clevert et al., 2016). We regularize parameters with a dropout ratio of 0.5 on LSTM input. We estimate parameters by maximizing the likelihood of the training data through stochastic subgradient descent using Adam (Kingma and Ba, 2015). Our implementation uses the Dynet library (Neubig et al., 2017) with default parameters.

6 Experiments

We ran a series of experiments on two corpora annotated with discontinuous constituents.

English We used an updated version of the Wall Street Journal part of the Penn Treebank corpus (Marcus et al., 1994) which introduces discontinuity (Evang and Kallmeyer, 2011). Sections 2-21 are used for training, 22 for developpement and 23 for testing. We used gold and predicted POS tags by the Stanford tagger,[10] trained with 10-jackknifing. Dependencies are extracted following the head-percolation table of Collins (1997).

German We used the Tiger corpus (Brants et al., 2004) with the split defined for the SPMRL 2014 shared task (Maier, 2015; Seddah et al., 2013). Following Maier (2015) and Coavoux and Crabbé (2017), we removed sentences number 46234 and 50224 as they contain annotation errors. We only used the given gold POS tags. Dependencies are extracted following the head-percolation table distributed with Tulipa (Kallmeyer et al., 2008).

We emphasize that long sentences are not filtered out. Our derivation extraction algorithm is similar to the one proposed in Carreras et al. (2008). Regarding decoding, we use a beam of size 10 for spines w.r.t. $P_\nu(s_m|m,\mathbf{w}) = \sum_h P_\nu(s_m|h,m,\mathbf{w}) \times P_\alpha(h|m,\mathbf{w})$ but allow every possible adjunction. The maximum number of iterations of the subgradient descent is set to 500 and the stepsize η^t is fixed following the rule of Polyak (1987).

Parsing results and timing on short sentences only ($\leq$ 40 words) and full test set using the de-

[9] If a spine contains repeated non-terminal sequences, we select the lowest match.

[10] http://nlp.stanford.edu/software/tagger.shtml

fault discodop[11] eval script are reported on Table 1 and Table 2.[12] We report labeled recall (LR), precision (LP), F-measure (LF) and time measured in minutes. We also report results published by van Cranenburgh et al. (2016) for the discontinuous PTB and Coavoux and Crabbé (2017) for Tiger. Moreover, dependency unlabeled attachment scores (UAS) and tagging accuracies (Spine acc.) are given on Table 3. We achieve significantly better results on the discontinuous PTB, while being roughly 36 times faster together with a low memory footprint.[13] On the Tiger corpus, we achieve on par results. Note however that Coavoux and Crabbé (2017) rely on a greedy parser combined with beam search.

Fast and efficient parsing of discontinuous constituent is a challenging task. Our method can quickly parse the whole test set, without any parallelization or GPU, obtaining an optimality certificate for more than 99% of the sentences in less than 500 iterations of the subgradient descent. When using a non exact decoding algorithm, such as a greedy transition based method, we may not be able to deduce the best opportunity for improving scores on benchmarks, such as the parameterization method or the decoding algorithm. Here the behavior may be easier to interpret and directions for future improvement easier to see. We stress that our method is able to produce an optimality certificate on more than 99% of the test examples thanks to the enhancement presented in Section 4.4.

	LR	LP	LF	Time
		Short sentences only		
This work	90.63	91.01	90.82	≈ 4
This work†	89.57	90.13	89.85	≈ 4
VC2016†			87.00	≈ 180
		Full test set		
This work	89.89	90.29	90.09	≈ 6.5
This work†	88.90	89.45	89.17	≈ 5.5

Table 1: Parsing results and processing time on the english discontinuous PTB corpus. Results marked with † use predicted part-of-speech tags. VC2016 indicates results of van Cranenburgh et al. (2016).

	LR	LP	LF	Time
		Short sentences only		
This work	82.69	84.68	83.67	≈ 7.5
		Full test set		
This work	80.66	82.63	81.63	≈ 11
C2017			81.60	≈ 2.5

Table 2: Parsing results and processing time on the german Tiger corpus. C2017 indicates results of Coavoux and Crabbé (2017).

7 Related Work

Spine-based parsing has been investigated in (Shen and Joshi, 2005) for Lexicalized TAGs with a left-to-right shift-reduce parser which was subsequently extended to a bidirectional version in (Shen and Joshi, 2008). A graph-based algorithm was proposed in (Carreras et al., 2008) for second-order projective dependencies, and for a form of non-projectivity occurring in machine translation (i.e. projective parses of permutated input sentences) in (Carreras and Collins, 2009).

Discontinuous phrase-structure parsing through dependencies in contexts other that TAGs have been explored in (Hall and Nivre, 2008; Versley, 2014; Fernández-González and Martins, 2015). The first two encode spine information as arc labels while the third one relaxes spine information by keeping only the root and height of the adjunction, thus avoiding combinatorial explosion. Labeling is performed as a post-processing step in these approaches, since the number of labels can be very high. Our model also performs labeling after structure construction, but it could be performed jointly without major issue. This is one way our model could be improved.

GMSA has been studied mostly as a way to solve the non directed version (i.e. with symetric arc weights) (Myung et al., 1995), see (Pop, 2009; Feremans et al., 1999) for surveys on resolution methods. Myung et al. (1995) proposed an exact decoding algorithm through branch-and-bound using a dual ascent algorithm to compute bounds. Pop (2002) also used Lagrangian relaxation – in the non directed case – where a single subproblem is solved in polynomial time. However, the relaxed constraints are inequalities: if the dual objective returns a valid primal solution, it is not a sufficient condition in order to guarantee that

[11] https://github.com/andreasvc/disco-dop/

[12] C2017 processing time is 137.338 seconds plus approximatively 30 seconds for model and corpus loading (personnal communication).

[13] Execution times are not directly comparable because we report our experimental conditions and published results.

	UAS	Spine acc.
English	93.70	97.32
English†	93.04	96.81
German	92.25	96.49

Table 3: Dependency parsing and tagging results. Results marked with † use predicted part-of-speech tags.

it is the optimal solution (Beasley, 1993), and thus the stopping criterion for the subgradient descent is usually slow to obtain. To our knowledge, our system is the first time that GMSA is used to solve a NLP problem.

Dual decomposition has been used to derive efficient practical resolution methods in NLP, mostly for machine translation and parsing, see (Rush et al., 2010) for an overview and (Koo et al., 2010) for an application to dependency parsing.

To accelerate the resolution, our method relies heavily on problem reduction (Beasley, 1993), which uses the primal/dual bounds to filter out suboptimal assignments. Exact pruning based on duality has already been studied in parsing, with branch and bound (Corro et al., 2016) or column generation (Riedel et al., 2012) and in machine translation with beam search (Rush et al., 2013).

8 Conclusion

We presented a novel framework for the joint task of supertagging and parsing by a reduction to GMSA. Within this framework we developed a model able to produce discontinuous constituents. The scoring model can be decomposed into tagging and dependency parsing and thus may rely on advances in those active fields.

This work could benefit from several extensions. Bigram scores on spines could be added at the expense of a third subproblem in the dual objective. High-order scores on arcs like grandparent or siblings can be handled in subproblem P_2 with the algorithms described in (Koo et al., 2010). In this work, the parameters are learned as separate models. Joint learning in the max-margin framework (Komodakis, 2011; Komodakis et al., 2015) may model interactions between vertex and arc weights better and lead to improved accuracy. Finally, we restricted our grammar to spinal trees but it could be possible to allow full lexicalized TAG-like trees, with substitution nodes and even obligatory adjunction sites. Derivations compatible with the TAG formalism (or more generally LCFRS) could be recovered by the use of a constrained version of MSA (Corro et al., 2016).

Acknowledgements

We thank the anonymous reviewers for their insightful comments. We thank Laura Kallmeyer and Kilian Evang for providing us with the script for the discontinuous PTB. First author is supported by a public grant overseen by the French National Research Agency (ANR) as part of the Investissements d'Avenir program (ANR-10-LABX-0083). Second author, supported by a public grant overseen by the French ANR (ANR-16-CE33-0021), completed this work during a CNRS research leave at LIMSI, CNRS / Université Paris Saclay.

References

Miguel Almeida and Andre Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 196–206, Sofia, Bulgaria. Association for Computational Linguistics.

John Beasley. 1993. *Modern heuristic techniques for combinatorial problems*, chapter Lagrangian relaxation. John Wiley & Sons, Inc.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. Tiger: Linguistic interpretation of a german corpus. *Research on language and computation*, 2(4):597–620.

Xavier Carreras and Michael Collins. 2009. Non-projective parsing for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 200–209, Singapore. Association for Computational Linguistics.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 9–16, Manchester, England. Coling 2008 Organizing Committee.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In *Proceedings of the 2016 International Conference on Learning Representations*.

Maximin Coavoux and Benoit Crabbé. 2017. Incremental discontinuous phrase structure parsing with the gap transition. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1259–1270. Association for Computational Linguistics.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain. Association for Computational Linguistics.

Caio Corro, Joseph Le Roux, Mathieu Lacroix, Antoine Rozenknop, and Roberto Wolfler Calvo. 2016. Dependency parsing with bounded block degree and well-nestedness via lagrangian relaxation and branch-and-bound. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 355–366, Berlin, Germany. Association for Computational Linguistics.

Andreas van Cranenburgh, Remko Scha, and Rens Bod. 2016. Data-oriented parsing with discontinuous constituents and function tags. *Journal of Language Modelling*, 4(1):57–111.

Dipanjan Das, André F. T. Martins, and Noah A. Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *The First Joint Conference on Lexical and Computational Semantics*, pages 209–217, Montréal, Canada. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. Deep biaffine attention for neural dependency parsing. *Proceedings of the 2017 International Conference on Learning Representations*.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *New Developments in Natural Language Parsing*, pages 29–62. Kluwer Academic Publishers.

Kilian Evang and Laura Kallmeyer. 2011. PLCFRS parsing of english discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 104–116, Dublin, Ireland. Association for Computational Linguistics.

Corinne Feremans, Martine Labbé, and Gilbert Laporte. 1999. The generalized minimum spanning tree: Polyhedra and branch-and-cut. *Electronic Notes in Discrete Mathematics*, 3:45–50.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1523–1533, Beijing, China. Association for Computational Linguistics.

Johan Hall and Joakim Nivre. 2008. Parsing discontinuous phrase structure with grammatical functions. In *Advances in Natural Language Processing: 6th International Conference, GoTAL 2008 Gothenburg, Sweden, August 25-27, 2008 Proceedings*, pages 169–180, Berlin, Heidelberg. Springer Berlin Heidelberg.

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

Laura Kallmeyer, Timm Lichte, Wolfgang Maier, Yannick Parmentier, Johannes Dellert, and Kilian Evang. 2008. Tulipa: Towards a multi-formalism parsing environment for grammar engineering. In *Coling 2008: Proceedings of the workshop on Grammar Engineering Across Frameworks*, pages 1–8, Manchester, England. Coling 2008 Organizing Committee.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of The International Conference on Learning Representations (ICLR)*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Nikos Komodakis. 2011. Efficient training for pairwise or higher order crfs via dual decomposition. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1841–1848. IEEE.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2007. MRF optimization via dual decomposition: Message-passing revisited. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE.

Nikos Komodakis, Bo Xiang, and Nikos Paragios. 2015. A framework for efficient structured max-margin learning of high-order mrf models. *IEEE transactions on pattern analysis and machine intelligence*, 37(7):1425–1441.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA. Association for Computational Linguistics.

Joseph Le Roux, Antoine Rozenknop, and Jennifer Foster. 2013. Combining PCFG-LA models with dual decomposition: A case study with function labels and binarization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1158–1169, Seattle, Washington, USA. Association for Computational Linguistics.

Claude Lemaréchal. 2001. Lagrangian relaxation. In *Computational combinatorial optimization*, pages 112–156. Springer.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese pos tagging and dependency parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1180–1191, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Wolfgang Maier. 2015. Discontinuous incremental shift-reduce parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1202–1212. Association for Computational Linguistics.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *HLT'94: Proceedings of the workshop on Human Language Technology*, pages 114–119, Morristown, NJ, USA. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Young-Soo Myung, Chang-Ho Lee, and Dong-Wan Tcha. 1995. On the generalized minimum spanning tree problem. *Networks*, 26(4):231–241.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Boris T Polyak. 1987. Introduction to optimization. *Optimization Software*.

Petrica Claudiu Pop. 2002. *The generalized minimum spanning tree problem*. Twente University Press.

Petrica Claudiu Pop. 2009. A survey of different integer programming formulations of the generalized minimum spanning tree problem. *Carpathian Journal of Mathematics*, 25(1):104–118.

Sebastian Riedel, David Smith, and Andrew McCallum. 2012. Parse, price and cut—delayed column and row generation for graph based parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 732–743, Jeju Island, Korea. Association for Computational Linguistics.

Alexander Rush, Yin-Wen Chang, and Michael Collins. 2013. Optimal beam search for machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 210–221, Seattle, Washington, USA. Association for Computational Linguistics.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Cambridge, MA. Association for Computational Linguistics.

A. Schrijver. 2003. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, D. Jinho Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Villemonte Eric de la Clergerie. 2013. *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, chapter Overview of the SPMRL 2013 Shared Task: A Cross-Framework Evaluation of Parsing Morphologically Rich Languages. Association for Computational Linguistics.

Libin Shen. 2006. *Statistical LTAG Parsing*. Ph.D. thesis, University of Pennsylvania.

Libin Shen and Aravind Joshi. 2005. Incremental ltag parsing. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 811–818, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Libin Shen and Aravind Joshi. 2008. LTAG dependency parsing with bidirectional incremental construction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 495–504, Honolulu, Hawaii. Association for Computational Linguistics.

Yannick Versley. 2014. Experiments with easy-first nonprojective constituent parsing. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 39–53, Dublin, Ireland. Dublin City University.

Incremental Graph-based Neural Dependency Parsing

Xiaoqing Zheng
School of Computer Science, Fudan University, Shanghai, China
Shanghai Key Laboratory of Intelligent Information Processing
zhengxq@fudan.edu.cn

Abstract

Very recently, some studies on neural dependency parsers have shown advantage over the traditional ones on a wide variety of languages. However, for graph-based neural dependency parsing systems, they either count on the long-term memory and attention mechanism to implicitly capture the high-order features or give up the global exhaustive inference algorithms in order to harness the features over a rich history of parsing decisions. The former might miss out the important features for specific headword predictions without the help of the explicit structural information, and the latter may suffer from the error propagation as false early structural constraints are used to create features when making future predictions. We explore the feasibility of explicitly taking high-order features into account while remaining the main advantage of global inference and learning for graph-based parsing. The proposed parser first forms an initial parse tree by head-modifier predictions based on the first-order factorization. High-order features (such as grandparent, sibling, and uncle) then can be defined over the initial tree, and used to refine the parse tree in an iterative fashion. Experimental results showed that our model (called INDP) archived competitive performance to existing benchmark parsers on both English and Chinese datasets.

1 Introduction and Motivation

The rise of machine learning methods in natural language processing (NLP) coupled with the availability of treebanks (Buchholz and Marsi, 2006) for a wide variety of languages has led to a rapid increase in research on data-driven dependency parsing. Two predominant paradigms for the data-driven dependency parsing are often called graph-based and transition-based dependency parsing (McDonald and Nivre, 2007, 2011). The first category learns the parameters to score correct dependency subgraphs over incorrect ones, typically by factoring the graphs into their component directed arcs, and performs parsing by searching the highest-scoring graph for a given sentence. The second category of parsing systems instead learns to predict one transition from one parse state to the next given a parse history, and performs parsing by taking the predicted transitions at each parse state until a complete dependency graph is derived.

Empirical studies show that the graph-based and transition-based models exhibit no statistically significant difference in accuracy on a variety of languages, although they are very different theoretically (McDonald and Nivre, 2011). Graph-based models are usually trained by maximizing the difference in score between the entire correct dependency graph and all incorrect ones for every training sentence. However, exhaustive inference is generally NP-hard when the score is factored over any extended scope of the dependency subgraph beyond a single arc (McDonald and Satta, 2007), which is the primary shortcoming of the graph-based systems. In transition-based parsing, the feature representations are not restricted to a small number of arcs in the graph but can be derived from all the dependency subgraphs built so far, while the main disadvantage of these models is that the local greedy parsing strategy may lead to the error propagation because false early predictions can eliminate valid parse trees.

With a few exceptions (Zeman and Žabokrtskỳ, 2005; Zhang and Clark, 2008; Zhang et al., 2014), the graph-based parsers usually require global

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1655–1665
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

learning and inference, but define features over a limited scope of the dependency graph, while the transition-based ones typically use local, greedy training and inference, but introduce a rich feature space based on the history of parsing decisions.

Many approaches have been proposed to overcome the weaknesses of traditional graph-based or transition-based models. There are at least three ways for potential improvement: ensemble—weighting the predictions of multiple parsing systems (Sagae and Lavie, 2006; Hall et al., 2007), feature integration—combining the two models by allowing the output of one model to define features for the other (Martins et al., 2008; Nivre and McDonald, 2008; McDonald and Nivre, 2011), and novel approaches—changing the underlying model structure directly by constructing globally trained transition-based parsers (Zhang and Clark, 2008; Huang and Sagae, 2010) or graph-based parsers with rich features (Riedel and Clarke, 2006; Nakagawa, 2007; Smith and Eisner, 2008; Martins et al., 2009).

Very recently, some studies on the deep architectures have shown advantage over the shallow ones on a wide variety of dependency parsing benchmarks. Deep neural networks were used to replace the classifiers for predicting optimal transitions in transition-based parers (Chen and Manning, 2014) or the scoring functions for ranking the subgraphs in graph-based rivals (Kiperwasser and Goldberg, 2016a,b). There are several recent developments in neural dependency parsing (Weiss et al., 2015; Zhou et al., 2015; Dyer et al., 2015), which can be viewed as targeting the weaknesses of locally greedy algorithms in transition-based models by using the beam search and conditional random field loss objective, although using the beam search instead of strictly deterministic parsing can to some extent alleviate the error propagation problem but does not eliminate it.

For graph-based neural dependency parsing systems, they either count on the long-term memory and neural attention to implicitly capture the high-order features (Kiperwasser and Goldberg, 2016b; Cheng et al., 2016; Dozat and Manning, 2017) or give up the global inference algorithms in order to introduce features over a rich history of parsing decisions by a greedy, bottom-up method (Kiperwasser and Goldberg, 2016a). The former might miss out the important information for specific headword predictions without the help of the structural features derived from the entire parse tree, while the latter may suffer from the error propagation as false structural constraints are used to create features when making future predictions. In this study, we explore the feasibility of explicitly taking advantage of high-order features while remaining the strength of global exhaustive inference and learning as a graph-based parser.

The proposed parser first encodes each word in a sentence by distributed embeddings using a convolutional neural network and constructs an initial parse graph by head-modifier predictions with a maximum directed spanning tree algorithm based on the first-order features (i.e. the score is factored over the arcs in a graph). Once an initial parse graph is built, the high-order features (such as grandparent, sibling, and uncle) can be defined, and used to refine the structure of the parse tree in an iterative way. Theoretically, the refinement will continue until no change is made in the iteration. But experimental results demonstrated that pretty good performance can be achieved with no more than twice updates because many dependencies are determined by independent arc prediction and a few head-modifier pairs need to be re-estimated after one update (i.e. only a few changes above and beyond the dominant first-order scores). We call this proposed model an incremental neural dependency parsing (INDP)[1].

2 Incremental Neural Dependency Parser

Given an input sentence x, we denote the set of all valid dependency parse trees that can be constructed from x as $\mathcal{Y}(x)$. Assuming there exists a graph scoring function s, the dependency parsing problem can be formulated as finding the highest scoring directed spanning tree for the sentence x.

$$y^*(x) = \operatorname*{argmax}_{\hat{y} \in \mathcal{Y}(x)} s(x, \hat{y}; \theta) \tag{1}$$

where $y^*(x)$ is the parse tree with the highest score, and θ is a set of the parameters used to compute the scores. To make the search tractable, the score of a graph is usually factorized into the sum of its arc (head-modifier) scores (McDonald et al., 2005a).

$$s(x, \hat{y}; \theta) = \sum_{(h,m) \in A(\hat{y})} s(h, m; \theta) \tag{2}$$

[1] The source code is available at http://homepage.fudan.edu.cn/zhengxq/deeplearning/

where $A(\hat{y})$ represents a set of directed arcs in the parse tree $\hat{y}$. The score of an arc (h, m) represents the likelihood of creating a dependency from head h to modifier (or dependent) m in a dependency tree. If each arc score is estimated independently, we call it a first-order factorization. When the scoring is based on two or more arcs, second- or high-order factorizations are applied.

In traditional approaches, this score is commonly defined to be the product of a high dimensional feature representation of the arc and a learned weighting parameter vector. The performance of those systems is heavily dependent on the choice of features. For that reason, much effort in designing such systems goes into the feature engineering, which is important but labor-intensive, mainly first based on human ingenuity and linguistic intuition, and then confirmed or refined by empirical analyses. In this study, a neural network is designed instead to estimate the arc scores using the high-order features. In the following, we first describe how the word representations are produced. Then, the key components of the INDP, direction-specific scoring with special normalization and incremental refinement with high-order features, are discussed in detail. Finally, we present the entire parsing algorithm of the INDP.

2.1 Word Feature Representations

In graph-based neural dependency parsing work, such as (Kiperwasser and Goldberg, 2016a,b; Dozat and Manning, 2017), recurrent neural network (RNN) is a popular statistical learner used to produce the continuous vector representations for each word in a sentence due to its ability to bridge long time lags between relevant inputs. We chose to use one-dimensional convolution instead as a building block because it is good enough to capture the interactions of word feature representations in a context window with less computational cost. Such a design makes the parameters of our first-order parser to be optimized efficiently, which will be augmented with the high-order features (i.e. long distance dependencies) at incremental refinement stages.

The words are fed into the network as indices that are used by a lookup operation to transform words into their feature vectors. We consider a fixed-sized word dictionary $\mathcal{D}$[2]. The vector representations are stored in a word embedding matrix $E^{word} \in \mathbb{R}^{d \times |\mathcal{D}|}$, where d is the dimensionality of the vector space (a hyper-parameter to be chosen) and $|\mathcal{D}|$ is the size of the dictionary. Like (Chen and Manning, 2014; Dyer et al., 2015; Weiss et al., 2015; Cheng et al., 2016), we also map part-of-speech (POS) tags to another q-dimensional vector space, and provide POS type features for words. Formally, assume we are given a sentence $x_{[1:n]}$ that is a sequence of n words $x_i, 1 \leq i \leq n$. For each word $x_i \in \mathcal{D}$ that has an associated index k_i into the column of the matrix E^{word}, and is labeled as a POS tag of type l_i, its feature representation is obtained by concatenating both word and POS tag embeddings as:

$$E(x_i) = E^{word} e_{k_i} \oplus E^{pos} e_{l_i} \tag{3}$$

where $E^{pos} \in \mathbb{R}^{q \times |\mathcal{P}|}$ is a POS tag embedding matrix and $|\mathcal{P}|$ is the size of POS tag set $\mathcal{P}$ (fine-grained POS tags are used if available). Binary e_{k_i} and e_{l_i} are one-hot encoding vectors for the ith word in the sentence.

The lookup table layer extracts features for each single word, but the meaning of a word is strongly related to its surrounding words. Given a word, we consider a fixed size window w (another hyper-parameter) of words around it. More precisely, given an input sentence $x_{[1:n]}$, the feature window produced by the first lookup table layer at position x_i can be written as:

$$f^{win}_{x_i} = (E(x_{i-w/2}) \cdots E(x_i) \cdots E(x_{i+w/2})) \tag{4}$$

where the word feature window is a matrix $f^{win} \in \mathbb{R}^{(d+q) \times w}$, and each column of the matrix is the word feature vector in the context window. A one-dimensional convolution is used to yield another feature vector by taking the dot product of filter vectors with the rows of the matrix f^{win} at the same dimension. After each row of f^{win} is convolved with the corresponding column of a filter matrix W^1, some non-linear function $\phi(\cdot)$ will be applied as:

$$f^{con} = \phi(f^{win} \odot W^1) \tag{5}$$

where the weights in the matrix $W^1 \in \mathbb{R}^{w \times (d+q)}$ are the parameters to be trained, and the output $f^{con} \in \mathbb{R}^{(d+q)}$ is a vector. We choose a hyperbolic tangent as the non-linear function ϕ. The word feature vectors from a window of text can be computed efficiently thanks to the speed advantage of the one-dimensional convolution (Kalchbrenner et al., 2014).

[2] Unless otherwise specified, the word dictionary is extracted from the training set. Unknown words are mapped to a special symbol that is not used elsewhere.

2.2 Direction-Specific Scoring

For the same head-modifier arc (h, m), the head word h may occur on the left size of m (i.e. left-arc) in some sentences while it also can appear on the right size of m (i.e. right-arc) in other ones. Considering two English sentences excerpted from the Penn Treebank (Buchholz and Marsi, 2006): “A group of workers exposed to it.”, and “Mr. Vinken is chairman of Elsevier, the Dutch publishing group.”, they have the same (group, of) head-modifier arc, but those two words occur in different orders. This would not be problem in the traditional models, such as (McDonald et al., 2005a; Nivre and McDonald, 2008), in which the arc directions are directly used as features by their structured learning algorithms. However, it is hard to train a single neural network that gives a higher score to the left-arc case than the right-arc one in some situations while reverses in others because of the symmetries in weight space (Note that we cannot tell which case is correct in advance, and both cases need to be scored). It would be more serious when the first-order factorization is applied due to the lack of context information.

Based on the above observations, we use a multi-layer perceptron (MLP) to score the left-arc cases, and another MLP to score the right-arc ones. Those two MLPs share the word and POS tag embeddings, and can update them when necessary during the training process. Formally, if a MLP with one hidden layer is used, the score of each possible head-modifier arc is computed as:

$$s(h, m; \theta) = W^3(\phi(W^2(f_h^{con} \oplus f_m^{con} \oplus f_{h,m}^{dis}) + b^2)) \quad (6)$$

where the convolutional outputs of the head and dependent words are concatenated with a bucketed distance between the head and modifier, denoted by $f_{h,m}^{dis}$, in buckets of 0 (root), 1, 2, 3-5, and 6+, and feed into the MLP for scoring. The weights in the hidden and output layers are denoted by W^2 and W^3 respectively, and the corresponding bias by b^2. Once every possible arc is scored, we obtain a matrix like Figure 1, in which the element at the row i and column j is the score for (x_i, x_j) arc, denoted by $s(i, j)$. An artificial word, x_0, has been inserted at the beginning of a sentence that will always serve as the single root of the graph and is primarily a means to simplify computation. The scores at the lower (or upper) triangular are computed by the left-arc (or right-arc) MLP, and the shaded elements do not need to be calculated. We can treat $s(i, j)$ as a score of the corresponding arc and then search for the highest scoring directed spanning tree to form a dependency parse tree as proposed in (McDonald et al., 2005b). This problem can be solved using the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967), which can be implemented in $O(n^2)$.

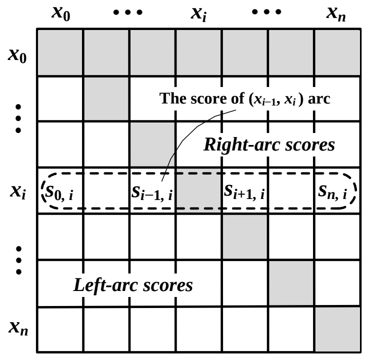


Figure 1: Scoring matrix for possible head and modifier arcs, in which the element at the row i and column j is the score for (x_i, x_j) arc, denoted by $s(i, j)$. A dependency tree can be formed by finding the highest scoring directed spanning tree over the scoring matrix.

The left-arc and right-arc MLPs should carefully collaborate with each other; otherwise, one MLP would be overwhelmed by another (i.e. the maximum score produced by one MLP is less than the minimum by another). To overcome this bias problem, we use the partition function by summing over the elements in each row of the scoring matrix, namely the scores/probabilities are normalized across the two MLPs. The conditional probability of arc (x_i, x_j) given a sentence $x_{[1:n]}$ is defined as:

$$\begin{gathered} p((x_i, x_j)|x_{[1:n]}; \theta) = \frac{\exp s(x_i, x_j; \theta)}{Z_i(x_{[1:n]}; \theta)} \\ \text{where } Z_i(x_{[1:n]}; \theta) = \sum_{i \in \{0 \ldots n\}, i \neq j} \exp s(x_i, x_j; \theta) \end{gathered} \quad (7)$$

Each $Z_i(x_{[1:n]}; \theta)$ is a normalization term used to predict x_i’s head word.

2.3 Incremental Refinement with High-order Features

Given an input sentence, once the initial dependency tree is built using the first-order factorization, we can define the high-order features over the resulting tree. For each head-modifier arc, the modifier’s left sibling, right sibling, leftmost child, and rightmost child vector representations are concatenated with the inputs of Equation (6), which are then feed into two new left-arc and right-arc

MLPs to update the scoring matrix. Like the head and modifier, those additional feature representations are added as the results produced by the convolution layer. As shown in Figure 2, commonly-used high-order features have been take into account, such as consecutive sibling (H, B, S), tri-siblings (B, M, S), and grandparent (H, M, R). The missing feature vectors are replaced by one of four special vectors, namely “left-sibling”, “right-sibling”, “leftmost-child”, and “rightmost-child” according to their relations to the modifier word.

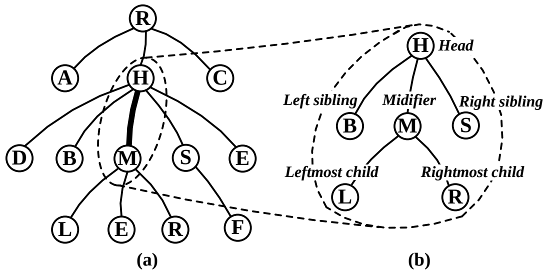


Figure 2: High-order features. (a) An example dependency parse tree. (b) The subgraph used to define high-order features for the head-modifier arc (H, M).

Although high-order features are used, the highest scoring parse tree still can be founded efficiently in $O(n^2)$ by the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967). The main rationale is that, even in the presence of high-order features, the resulting scores remain based on single head-modifier arcs. The higher-order features are derived from the parse tree obtained with first-order inference, and because that tree is already pretty good, these higher-order features end up being a good approximation, and such approximation can be further improved by incremental refinements upon the parse tree. Thus, the high-order features used by the scoring MLPs can offer deliberate refinement above and beyond the first-order results. Theoretically, the refinement can be made until there is no update in the scoring matrix. However, experimental results show that comparable performance can be achieved with no more than twice high-order refinements (see Section 3).

We add a softmax layer to the network (after removing the last scoring layer) to predict syntactic labels for each arc. Labeling is trained by minimizing the cross-entropy error of the softmax layer using backpropagation. The network performs the structure prediction and labeling jointly. The two tasks shared the several layers (from the input to convolutional layers) of the network. When minimizing the cross-entropy error of the softmax layer, the error will also backpropagate and influence both the network parameters and the embeddings. We list our incremental neural dependency parsing algorithm in Figure 3. Staring with an initial tree formed using the first-order features, the algorithm makes changes to the parse tree with the high-order refinements in an attempt to climb the objective function.

```
Inputs:
 $\theta$: neural network parameters.
 $x$: an input sentence.
 $T$: maximum number of iterations.
Output: optimal dependency tree $y^*$.
Algorithm:
form an initial tree using the first-order features;
$t = 0$;
repeat
  update the scoring matrix using the high-order features;
  find the highest scoring tree $y$ by Chu-Liu-Edmonds
  algorithm;
  $t = t + 1$;
until no change in this iteration or $t \geq T$;
predict syntactic labels based on the parse tree $y$;
return $y^* = y$;
```

Figure 3: Incremental neural dependency parsing (INDP) algorithm.

2.4 Training

Given a training example (x, y), we defined a structured margin $\Delta(x, y, \hat{y})$ loss for proposing a parse $\hat{y}$ for sentence x when y is the true parse. This penalty is proportional to the number of unlabeled arcs on which the two parse trees do not agree. In general, $\Delta(x, y, \hat{y})$ is equal to 0 if $y = \hat{y}$. The loss function is defined as a penalization of incorrect arcs:

$$\Delta(x, y, \hat{y}) = \sum_{(h,m) \in A(\hat{y})} \kappa \mathbf{1}\{(h, m) \notin A(y)\} \qquad (8)$$

where κ is a penalization term to each incorrect arc, and $A(y)$ is a set of arcs in the true parse y.

For a training set, we seek a function with small expected loss on unseen sentences. The function we consider take the following form as Equation (1). The score of a tree $\hat{y}$ is higher if the algorithm is more confident that the structure of the tree is correct. In the max-margin estimation framework, we want to ensure that the highest scoring tree is the true parse for all training instances (x^i, y^i), $i = 1, \cdots, h$, and it’s score to be larger up to a margin defined by the loss. For all i in the training data:

$$s(\theta, x^i, y^i) \geq s(\theta, x^i, \hat{y}) + \Delta(x^i, y^i, \hat{y}) \qquad (9)$$

These lead us to minimize the following regularized objective for h training instances:

$$\begin{aligned} J(\theta) &= \frac{1}{h}\sum_{i=1}^{h} E^{i}(\theta) + \frac{\lambda}{2}||\theta||^{2}, \text{where} \\ E_i(\theta) &= \max_{\hat{y}\in\mathcal{Y}(x^i)} (0, (s(\theta, x^i, \hat{y}) + \Delta(x^i, y^i, \hat{y})) \\ &\quad - s(\theta, x^i, y^i)) \end{aligned} \tag{10}$$

where the coefficient λ governs the relative importance of the regularization term compared with the error. The trees are penalized more by the loss when they deviate from the correct one. Minimizing this objective maximizes the score of the correct tree, and minimizes that of the highest scoring but incorrect parse tree. The objective is not differentiable due to the hinge loss. We use the subgradient method to compute a gradient-like direction for minimizing the objective function.

3 Experiments

We conducted three sets of experiments. The first one is to test several variants of the INDP on the development set, to gain some understanding of how the choice of hyper-parameters impacts upon the performance. The goal of the second one is to see how well the incremental approach enhanced with the high-order features to improve the first-order results by analysing parsing errors relative to sentence length. In the third set, we compared the performance of the INDP with existing state-of-the-art models on both English and Chinese datasets. We report unlabeled attachment scores (UAS) and labeled attachment scores (LAS) with punctuations being omitted from evaluation.

3.1 Datasets

We show test results for the proposed model on the English Penn Treebank (PTB), converted into Stanford dependencies using version 3.3.0 of the Stanford dependency converter, and the Chinese Penn Treebank (CTB). We follow the standard splits of PTB, using section 2-21 for training, section 22 as development set and 23 as test set. We use POS tags generated from the Stanford POS tagger (Toutanova et al., 2003); for the Chinese PTB dataset, we use gold word segmentation and POS tags.

3.2 Training Strategy

Previous work demonstrated that the performance can be improved by using word embeddings learned from large-scale unlabeled data in many NLP tasks both in English (Collobert et al., 2011; Socher et al., 2011) and Chinese (Zheng et al., 2013). Unsupervised pretraining guides the learning towards basins of attraction of minima that support better generalization (Erhan et al., 2010). We leveraged large unlabeled corpus to learn word embeddings, and then used these improved embeddings to initialize the word embedding matrices of the neural networks. English and Chinese Wikipedia documents were used to train the word embeddings by Word2Vec tool[3] proposed in (Mikolov et al., 2013).

Previous studies show that a joint solution (i.e., performing several tasks at the same time) usually leads to the improvement in accuracy over pipelined systems because the error propagation is avoided and the various information normally used in the different steps of pipelined systems can be integrated. The INDP networks are also trained in a joint way, but adopting three-step strategy. The parameters of the parsing neural network using the first-order factorization are first learned, and when its unlabeled parsing accuracy exceeds a given threshold (e.g. 85%), we start to train the high-order parsing network. The weights already trained in the first step will remain unchanged for the first several epochs, and they are in fact used to generate the high-order features. After the parsing accuracy reaches another threshold (e.g. 90%), all the parameters for the first-order, and high-order predictions as well as labeling are trained jointly.

3.3 Hyper-parameter Choices

Hyper-parameters was tuned with the PTB 3.3.0 development set by trying only a few different networks. Generally, the dimensionality of the embeddings, and the numbers of hidden units, provided they are large enough, have a limited impact on the generalization performance. In the following experiments, the window size was set to 5, the learning rate to 0.02, and the number of hidden layer to 300. The embedding size of words was set to 50, and that of tags to 30, which achieved a good trade-off between speed and performance. All experiments were run on a computer equipped with an Intel Xeon processor working at 2.2GHz, with 16GB RAM and a NVIDIA Titan GPU. The parsing speed of the INDP is around 250-300 sents/sec in average on the PTB dataset.

[3] Available at http://code.google.com/p/word2vec/

3.4 Sentence Length Factors

It is well known that dependency parsers tend to have lower accuracies for longer sentences because the increased presence of complex syntactic structures. In order to get a better understanding of how well the incremental strategy and high-order features benefit the models, Figure 4 shows the accuracy of our neural dependency parser using the first-order features only (indicated with "NDP + First-order") and INDP with at most twice high-order refinements (indicated with "INDP + High-order + M2") on the English PTB develop set. For simplicity, the experiments report unlabeled parsing accuracy, and identical experiments using labeled parsing accuracy did not reveal any additional information.

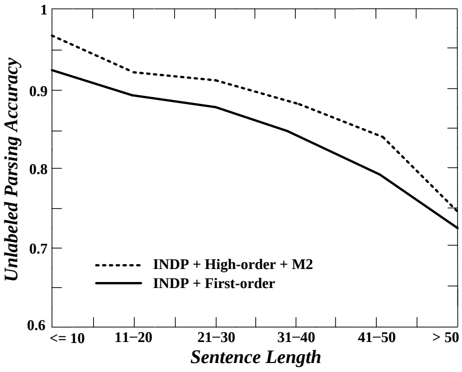


Figure 4: Accuracy relative to sentence length.

The INDP with the high-order refinements is more precise than the parser using only the first-order features. Due to the fact that longer dependencies are typically harder to parse, there is still a degradation in performance for our INDP. However, the accuracy curve for INDP is slightly flatter than its reduced version in which the high-order features and incremental recipe are not applied when the sentence length is within 11-50. This behavior can be explained by the reasons that the feature representations are not restricted to a limited number of graph arcs, but can take into account with the (almost) entire dependency graph built so far at the refinement stages of the INDP, and it do offer substantial refinements.

3.5 Results

We report the experimental results on the English PTB and Chinese CTB datasets in Table 1 and 2 respectively, in which our networks are denoted by "INDP". The "M1" indicates that the results are obtained by the INDP with just one refinement over the parse graphs built using the first-order features, and similarly, the "M2" indicates the results are achieved by the INDP with at most twice high-order refinements, while the "UNC" in the last row indicates that the refinements will continue until no change is made in the structure predictions (see the algorithm listed in Figure 3). All compared transition-based parsing systems are indicated by a "‡", and graph-based ones by "§".

Table 1: Results on the English PTB dataset.

Model	UAS	LAS
Zhou et al (2015)‡	93.28	92.35
Weiss et al (2015)‡	94.26	92.41
Ballesteros et al (2016)‡	93.56	91.42
Kiperwasser and Goldberg (2016b)‡	93.90	91.90
Andor et al (2016)‡	94.61	92.79
Kuncoro et al (2016)‡	**95.80**	**94.60**
Kiperwasser and Goldberg (2016a)§	93.00	90.90
Cheng et al (2016)§	94.10	91.49
Hashimoto et al (2016)§	94.67	92.90
Dozat and Manning (2017)§	95.74	94.08
NDP + First-order	90.88	88.93
INDP + High-order + M1	93.31	91.51
INDP + High-order + M2	94.76	93.12
INDP + High-order + UNC	95.53	93.94

From these numbers, a handful of trends are readily apparent. Firstly, we note that the "full-fledged" INDP (indicated with "UNC") is superior to that without the high-order refinements by a fairly significant margin (5.01% for English and 6.55% for Chinese in LAS). Another striking result of these experiments is that comparable performance can be obtained by no more than twice refinements with high-order features, and "INDP + High-order + M2" achieves a good trade-off between the performance and parsing complexity.

Table 2: Results on the Chinese CTB dataset.

Model	UAS	LAS
Ballesteros et al (2016)‡	87.65	86.21
Kiperwasser and Goldberg (2016b)‡	87.60	86.10
Kiperwasser and Goldberg (2016a)§	87.10	85.50
Cheng et al (2016)§	88.10	85.70
Dozat and Manning (2017)§	89.30	**88.23**
NDP + First-order	82.97	81.39
INDP + High-order + M1	87.35	85.82
INDP + High-order + M2	88.78	87.28
INDP + High-order + UNC	**89.42**	87.94

Our INDP gets nearly the same performance on the English PTB as the current models of (Kuncoro et al., 2016) and (Dozat and Manning, 2017) in spite of its simpler architectures, and gets state-of-the-art UAS accuracy on the Chinese CTB. The INDP lags behind in LAS, indicating one of a few possibilities. Firstly, we tried only a few different network configurations, and there are many ways (such as using deeper architectures, and recruiting bi-directional recurrent neural networks to produce word feature representations) that we could improve it further. Secondly, the model of (Kuncoro et al., 2016) is particularly designed to capture phrase compositionality, and thus, another possible improvement is to capture such compositionality by optimizing the network architectures, which may also lead to a better label score.

4 Related Work

Dependency-based syntactic representations of sentences have been found to be useful for various NLP tasks, especially for those involving natural language understanding in some way. We briefly review prior work both on graph-based and transition-based neural dependency parsers.

In transition-based parsing, we learn a model for scoring transitions from one state to the next, conditioned on the parse history, and parse a sentence by taking the highest-scoring transition out of every state until a complete dependency graph has been derived. Chen and Manning (2014) made the first successful attempt at introducing deep learning into a transition-based dependency parser. At each step, the feed-forward neural network assigns a probability to every action the parse can take from certain state (words on the stack and buffer). Some researchers have attempted to address the limitations of (Chen and Manning, 2014) by augmenting it with additional complexity.

A beam search and a conditional random field loss function were incorporated into the transition-based neural network models (Weiss et al., 2015; Zhou et al., 2015; Andor et al., 2016), which allow the parsers to keep the top-k partial parse trees and revoke previous actions once it finds evidence that they may have been incorrect by locally greedy choices. Dyer et al (2015) used three LSTMs to represent the buffer, stack, and parsing history, getting state-of-the-art results on Chinese and English dependency parsing tasks.

Graph-based parsers use machine learning for scoring each possible edge for a given sentence, typically by factoring the graphs into their component arcs, and constructing the parse tree with the highest score from these weighted edges. Kiperwasser and Goldberg (2016b) presented a neural graph-based parser in which the bi-directional L-STM's recurrent output vector for each word is concatenated with each possible head's vector (also produced by the same biLSTM), and the result is used as input to a multi-layer perceptron (MLP) for scoring this modifier-head pair. Given the scores of the arcs, the highest scoring tree is constructed using Eisner's decoding algorithm (Eisner, 1996). Labels are predicted similarly, with each word's recurrent output vector and its head's vector being used in a multi-class MLP.

Kiperwasser and Goldberg (2016a) also proposed a hierarchical tree LSTM to model the dependency tree structures in which each word is represented by the concatenation of its left and right modifier (child) vectors, and the modifier vectors are generated by two (leftward or rightward) recurrent neural networks. The tree representations were produced in a bottom-up recursive way with the (greedy) easy-first parsing algorithm (Goldberg and Elhadad, 2010). Similarly, Cheng et al (2016) proposed a graph-based neural dependency parser that is able to predict the scores for the next arc, conditioning on previous parsing decisions. In addition to using one bi-directional recurrent network that produces a recurrent vector for each word, they also have uni-directional recurrent neural networks (left-to-right and right-to-left) that keep track of the probabilities of each previous parsing actions.

In their many-task neural model, Hashimoto et al (2016) included a graph-based dependency parse in which the traditional MLP-based method that Kiperwasser and Goldberg (2016b) used was replaced with a bilinear one. Dozat and Manning (2017) modified the neural graph-based approach of (Kiperwasser and Goldberg, 2016b) in a few ways to improve the performance. In addition to building a network that is larger and uses more regularization, they replace the traditional MLP-based attention mechanism and affine label classifier with biaffine ones.

This work is most closely related to the graph-based parsing approaches with multiple high-order refinements (Rush and Petrov, 2012; Zhang et al., 2014), although the neural networks were not used in their parsers. Rush and Petrov (2012) proposed a multi-pass coarse-to-fine approach in which a coarse model was used to prune the search space in order to make the inference with up to third-order features practical. They start with a linear-time vine pruning pass and build up to high-order models. Zhang et al (2014) introduced a randomized greedy algorithm for dependency parsing in which they begin with a tree drawn from the uniform distribution and use hill-climbing strategy to find the optimal parse tree. Although they reported that drawing the initial tree randomly results in the same performance as when initialized from a trained first-order distribution, but multiple random restarts are required to avoid getting stuck in a locally optimal solution. Their greedy algorithm breaks the parsing into a sequence of local steps, which correspond to choosing the head for each modifier word (one arc at a time) in the bottom-up order relative to the current tree. In contrast, we employed the global inference algorithm to change the entire tree (all at a time) in each refinement step, which makes the improvement more efficient.

5 Conclusion

Graph-based parsers cannot easily condition on any extended scope of the dependency parse tree beyond a single arc, which is their primary shortcoming relative to transition-based competitors. We have shown that a simple, generally applicable incremental neural dependency parsing algorithm can deliver close to state-of-the-art parsing performance, which allows the high-order features to be taken into account without hurting the advantage of global exhaustive inference and learning as a member of graph-based parsing systems. Future work will involve exploring ways of augmenting the parser with a more innovative architecture than the relatively simple one used in current neural graph-based parsers.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This work was partly supported by a grant from Shanghai Municipal Natural Science Foundation (No. 15511104303).

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL'16).*

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack-LSTM parser. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'16).*

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the International Conference on Computational Natural Language Learning (CoNLL'06).*

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'14).*

Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional attention with agreement for dependency parsing. In *arXiv: 1608.02076.*

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

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

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the International Conference on Learning Representations (ICLR'17).*

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency paring with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL'15).*

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71:233–240.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96).*

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, and Pascal Vincent. 2010. Why does unsupervised pre-training help deep learning. *Journal of Machine Learning Research*, 11:625–660.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'10).*

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single Malt or blended? a study in multilingual parser optimization. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL'07)*.

Kazuma Hashimoto, Yoshimasa Tsuruoka Caiming Xiong, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple NLP tasks. In *arXiv: 1611.01587*.

Liang Huang and Kenjie Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL'10)*.

Nal Kalchbrenner, Edward Grefenstette, and Fernando Pereira. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14)*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016a. Easy-first dependency parsing with hierarchical tree LSTMs. *Transactions of the Association for Computational Linguistics*, 4:445–461.

Eliyahu Kiperwasser and Yoav Goldberg. 2016b. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Adhiguna Kuncoro, Miguel Ballesteros, Chris Dyer Lingpeng Kong, Graham Neibig, and Noah A. Smith. 2016. What do recurrent neural network grammars learn about syntax? In *arXiv: 1611.05774*.

André F. T. Martins, Dipanian Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'08)*.

André F. T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer liear programming formulations for dependency parsing. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing (ACL-IJCNLP'09)*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL'07)*.

Ryan McDonald and Joakim Nivre. 2011. Analyzing and integrating dependency parsers. *Computational Lingustics*, 37:197–230.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hanjič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Human Language Technology Conferece and the International Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP'05)*.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the International Workshop on Parsing Technologies (IWPT'07)*.

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

Tetsuji Nakagawa. 2007. Multilingual depdendency parsing using global features. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL'07)*.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL'08:HLT)*.

Sebastian Riedel and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'06)*.

Alexander M. Rush and Slav Petrov. 2012. Vine pruning for efficient multi-pass dependency parsing. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL'12)*.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL'06)*.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'08)*.

Richard Socher, Cliff C-Y. Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the International Conference on Machine learning (ICML'11)*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'03)*.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Stuctured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL'15)*.

Daniel Zeman and Zdeněk Žabokrtskỳ. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of the International Workshop on Parsing Technologies (IWPT'05)*.

Yuan Zhang, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2014. Greed is good if randomized: New inference for dependency parsing. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*.

Yue Zhang and Stephen Clark. 2008. A tale of two parses: Investigating and bombining graph-based and transition-based dependency parsing. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'08)*.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and pos tagging. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL'15)*.

Neural Discontinuous Constituency Parsing

Miloš Stanojević and **Raquel G. Alhama**
Institute for Logic, Language and Computation (ILLC)
University of Amsterdam
{m.stanojevic, rgalhama}@uva.nl

Abstract

One of the most pressing issues in discontinuous constituency transition-based parsing is that the relevant information for parsing decisions could be located in any part of the stack or the buffer. In this paper, we propose a solution to this problem by replacing the structured perceptron model with a recursive neural model that computes a global representation of the configuration, therefore allowing even the most remote parts of the configuration to influence the parsing decisions. We also provide a detailed analysis of how this representation should be built out of sub-representations of its core elements (words, trees and stack). Additionally, we investigate how different types of swap oracles influence the results. Our model is the first neural discontinuous constituency parser, and it outperforms all the previously published models on three out of four datasets while on the fourth it obtains second place by a tiny difference.

1 Introduction

Research on constituency parsing has been mostly concentrated on projective trees, which can be modeled with Context-Free Grammars (CFGs). One of the main reasons for this is that modeling non-projective trees often requires richer grammar formalisms, which in practice implies slower runtime. For instance, the parsing algorithms for binary LCFRS—the most prominent grammar-based approach to parsing non-projective constituency trees—have computational complexity $O(n^{3k})$, where k is the *fan-out* of the grammar. For this reason, researchers turned to faster approximate methods. Approximations can be done in two ways: either on the types of structures that are predicted or on the parsing algorithm.

The first approach approximates discontinuous constituency structures with simpler structures for which more efficient algorithms exist. This method works as a pipeline: it converts the input to a simpler formalism, parses with it, and then converts it back. Relevant examples are the parsers by Hall and Nivre (2008) and Fernández-González and Martins (2015), who convert discontinuous constituents to dependencies, and Versley (2016), who also applied a conversion but in this case to the projective constituency trees.

The second approach—approximation on the parsing algorithm—consists of an approximate search for the most probable parse. This is analogous to the search done by transition-based parsers, which greedily search through the space of all possible parses, resulting in very fast models. The first transition-based discontinuous constituency parser of this sort was presented in Versley (2014), and it consists of a shift-reduce parser that handles discontinuities with swap transitions. This parser was very similar to dependency parsers with swap transitions (Nivre, 2009; Nivre et al., 2009), but unlike its dependency equivalents, it did not exhibit higher accuracy. Later work on discontinuous transition-based parsing was largely focused on finding alternative transitioning systems to handle discontinuity. Maier (2015) and Maier and Lichte (2016) proposed new types of swap operations (*CompoundSwap* and *SkipShift*) to make the transition sequences shorter—and therefore easier to learn. Coavoux and Crabbé (2017) went even further by modifying not only the transitions but the whole configuration structure by introducing an additional stack.

Over the years the transitioning system has seen some progress, but the learning model has remained the same : a sparse linear model trained

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1666–1676
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

with structured perceptron and early update strategy (Collins, 2002; Collins and Roark, 2004; Huang et al., 2012). This model requires heavy feature engineering and has a limited capacity in modeling interaction between the features.

Maier and Lichte (2016) argue that one of the biggest problems of transition based systems is precisely their greedy search, because they cannot recover from the bad decisions made in earlier parsing steps. Some researchers try to account for this problem by increasing the beam size, but there is a limit on how much the beam can be increased while remaining efficient for practical use (Coavoux and Crabbé, 2017).

The solution we propose is to use a probabilistic model that exploits the information from the whole configuration structure when making the decision for the next action. This can be achieved by using recurrent neural models that allow information to flow all the way from the individual characters, up trough the words, POS tags, subtrees, stack and buffer until the final configuration representation. Thanks to using a neural network model, which removes the need for feature engineering, we can concentrate on the question of which representations are more relevant for the model at each step of the flow. Thus, we reflect on how alternative representations should impact the task, and we report their relative contribution in an ablation study.

In our work, we also reduce the number of swap transitions by trying to postpone them as much as possible, in a style similar to the lazy-swap used in Nivre et al. (2009) —albeit with an even lower number of swaps. This change influences the model indirectly by introducing a helpful inductive bias.

Our model gets state-of-the-art results on Negra, Negra-30 and TigerSPMRL datasets, and on the TigerHN achieves the second best published result. To the best of our knowledge this is the first work that uses neural networks in the context of discontinuous constituency parsing.

2 Transition System

We base our transitioning system on the shift-promote-adjoin transitions proposed in Cross and Huang (2016), because they remove the need for explicit binarization. Transition-based parsers consist of two components: a configuration that represents a parsing state and a set of transitions between configurations.

The configuration consists of two data structures: a stack S that contains all the constituents built so far, and a buffer B of words that remain to be processed. The initial configuration consists of a buffer filled with words and an empty stack—presented as the *axiom* in Figure 1. The objective is to find a sequence of transitions that lead to a *goal* state in which the buffer is empty and the stack contains only one constituent with the ROOT label. The *shift* transition moves the first element from the buffer to the top of the stack. The *pro(X)* transition "promotes" the topmost element of the stack: it replaces it with a tree that has nonterminal X and the topmost element of the stack as its only child, which also becomes its head constituent. The *adj*$\curvearrowleft$ transition adjoins the second topmost element of the stack as a leftmost child of the topmost element of the stack. The *adj*$\curvearrowright$ transition is a mirror transition of the *adj*$\curvearrowleft$.

The transitions described so far are enough to handle projective constituency structures, and have been used with success for this task in Cross and Huang (2016). To make the parser able to process discontinuous constituents we need an additional transition that allows for constituents that are far apart on the stack to become close, so that they can be adjoined into a new constituent. For this we use the *swap* transition from Nivre (2009). This transition takes the second topmost element from the stack and puts it back to the buffer. To prevent infinite loops of *shift-swap* transitions, we put a constraint that *swap* can be applied only to constituents that have not been swapped before. To do this we use the linear ordering of constituents $<_{ind}$ based on the position of the leftmost word in their yield (Maier and Lichte, 2016).

2.1 Oracle

In the case of non-projective parsing, the extraction of the oracle is not trivial because there can be many possible oracles that would derive the same tree. Therefore it is common practice to use some heuristic to extract only one of the possible oracles.

To construct the oracle, we start with the initial configuration and apply the first transition whose conditions are satisfied. We keep applying transitions to the resulting configurations until the goal is reached. The transitions are determined as follows: first, we apply *adj*$\curvearrowleft$, *adj*$\curvearrowright$ or *pro(X)* if

$$\begin{array}{rl}
\textit{axiom} & \langle\ [],\ [w_1, w_2, ..., w_n]\ \rangle \\[1ex]
\textit{shift} & \dfrac{\langle\ S,\ x|B\ \rangle}{\langle\ S|x,\ B\ \rangle} \\[2ex]
\textit{pro(X)} & \dfrac{\langle\ S|t,\ B\ \rangle}{\langle\ S|X(t),\ B\ \rangle} \\[2ex]
\textit{adj}\curvearrowleft & \dfrac{\langle\ S|t|X(t_1 \dots t_k),\ B\ \rangle}{\langle\ S|X(t, t_1 \dots t_k),\ B\ \rangle} \\[2ex]
\textit{adj}\curvearrowright & \dfrac{\langle\ S|X(t_1 \dots t_k)|t,\ B\ \rangle}{\langle\ S|X(t_1 \dots t_k, t),\ B\ \rangle} \\[2ex]
\textit{swap} & \dfrac{\langle\ S|t_1|t_2,\ B\ \rangle}{\langle\ S|t_2,\ t_1|B\ \rangle}\ t_1 <_{ind} t_2 \\[2ex]
\textit{goal} & \langle\ ROOT,\ \epsilon\ \rangle
\end{array}$$

Figure 1: Transition System

one of those produces a constituent that is found in the tree; in case of failure, we check the condition for applying *swap*, which varies depending on the type of oracle, as we define next. If all these checks fail then a *shift* transition is performed.

2.1.1 Eager Oracle

Nivre (2009) introduced *swap* transitions with a very simple oracle. We can define the swapping condition for the extraction of the Eager oracle transition sequence as:

$$s_1 <_G s_0 \qquad (1)$$

where s_0 and s_1 are the topmost and second topmost elements of the stack respectively, and $<_G$ is the *projective ordering* of the nodes in the tree. That ordering can be computed by visiting the nodes in the tree in the postorder traversal.

This is the technique that has been used in most previous proposals on discontinuous constituency parsing (Maier, 2015; Maier and Lichte, 2016).

2.1.2 Lazy Oracle

Eager swapping strategy produces a large number of *swap* transitions which makes them difficult to predict. For this reason, Nivre et al. (2009) introduced a *lazy-swap* operation that postpones swapping by having an additional condition during the construction of an oracle. This technique was used successfully in Versley (2014) to improve over the eager swapping baseline. As an example, in Figure 2a word $w1$ should shift and swap many times to get to word $w5$ in order to construct constituent C. In contrast, a Lazy oracle would postpone swapping until constituent B is built so that only one swap operation over node B would be enough for word $w1$ to get to word $w5$.

In order to define that condition in the context of discontinuous constituency parsing, we need to define a few other terms. First of all, we call a *projective constituent* any constituent that yields a continuous span of words (marked with blue color in Figure 2). Note that a projective constituent might contain non-projective constituents as its descendants. A *fully projective constituent* is a constituent that is projective and whose descendants are all projective (marked with red in Figure 2). Finally, a *maximal fully projective constituent* is a fully projective constituent whose parent is not a fully projective constituent (marked green in Figure 2). Finally, we define a function $MPC(x)$ that returns the closest maximally projective constituent that is ascendant of a constituent x if there is one; otherwise, it returns x.

The condition for the lazy swap can now be expressed as:

$$s_1 <_G s_0 \quad \wedge \quad MPC(s_0) \neq MPC(b_0) \qquad (2)$$

where s_0 and b_0 are the topmost elements of the stack and buffer, respectively. This means that we do not allow *swap* to penetrate into maximally projective constituents, so swapping can be delayed until the maximally projective constituent has been built.

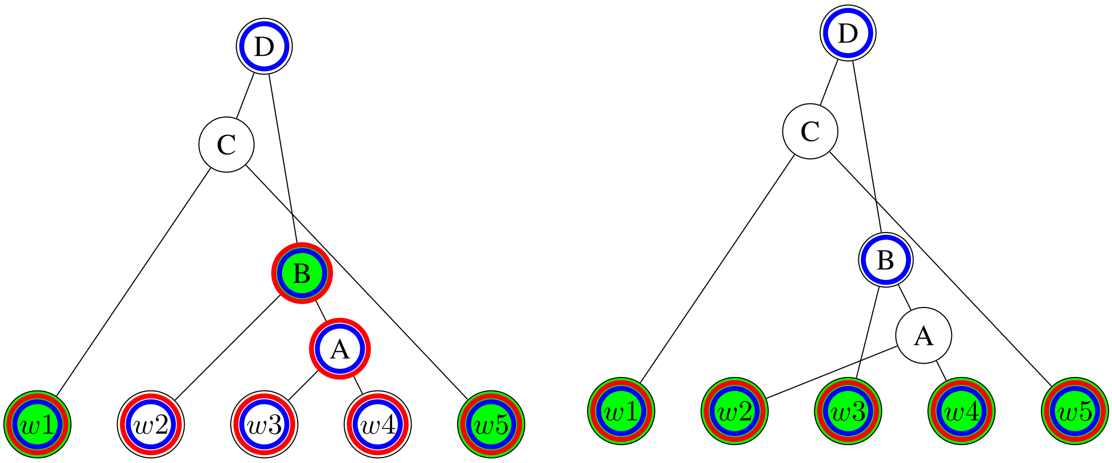
2.1.3 Lazier Oracle

The standard Lazy swap strategy helps in cases where MPC constituents exist, like in Figure 2a. But in cases like Figure 2b there are no MPC constituents (except for words), so Lazy would not show any improvement over Eager. Still, even in this case it is visible that swapping $w1$ should be postponed until B is built. We introduce an oracle strategy called Lazier that implements the heuristic of postponing swapping over projective constituents.[1]

Let a function $CPC(x)$ return the closest projective constituent ascendant of a constituent x. The condition for swap operation can now be expressed with:

$$s_1 <_G s_0 \quad \wedge \quad CPC(s_0) = CPC(s_1) \qquad (3)$$

[1] The same intuition is followed in the Barriers strategy of Versley (2014).



(a) Tree with Maximal Fully Projective Node B

(b) Tree without any Maximal Fully Projective Nodes (other than words themselves)

Figure 2: Example tree structures

This constraint prevents *swap* from allowing constituents to escape their closest projective ancestor. If we know that the *swap* operation can be performed, i.e. if $s_1 <_G s_0$, it is easy to show that in that case $CPC(s_0) = CPC(s_1) \implies MPC(s_0) \neq MPC(b_0)$ or in other words that Lazy is a special case of Lazier. There are are only two cases to consider about $CPC(s_0)$: case a) s_0 and $CPC(s_0)$ are separated by a non-empty sequence of non-projective constituents and case b) s_0 is the immediate child of $CPC(s_0)$. In case a) from definition of maximal projective constituents follows that $MPC(s_0) = s_0$ and therefore $MPC(s_0) \neq MPC(b_0)$ since s_0 and b_0 are non-overlapping. In case b) we need to consider two possible options: b1) $CPC(s_0)$ is fully projective and b2) $CPC(s_0)$ is not fully projective. Case b1) is not possible because it leads to contradiction with original condition $s_1 <_G s_0$. Case b2) leads again to $MPC(s_0) = s_0$ and by that $MPC(s_0) \neq MPC(b_0)$.

3 Model

As mentioned before, our goal is to have a model that can have a global representation of the parsing state. In order to define this global representation of the configuration, we first need to analyze what are the proper representations of its subparts.

3.1 How to Represent Terminal Nodes?

The representations induced by neural networks are continuous vectors that encode the information that is relevant for the loss function. The initial nodes in the computation graph are often embeddings that represent the atomic inputs in the model. In our model, the embedding of a terminal node is computed by concatenating the following four embeddings and then applying the affine transformation to compress the result into a smaller vector:

- a trained word embedding
- a trained POS tag embedding
- a pre-trained word embedding
- a trained character embedding of the word

Trained embeddings (both word and POS tag embedding) are automatically trained by our model to better suit the task that we are solving. The usage of pre-trained embeddings has become standard in neural parsing models: these presentations are helpful because they bring additional contextual information from a bigger non-annotated corpora. The embeddings that we use in this work are the ones distributed with the Polyglot package (Al-Rfou et al., 2013).

The character embedding representation of a word is computed by composing the representations of each character in the word form. This can be useful to recover some of the morphological features present in the word, such as suffixes or prefixes. We compose character embeddings by running a bi-directional LSTM (Bi-LSTM) over the characters (Ling et al., 2015; Ballesteros et al., 2015).

The embeddings composed in this way express the properties of a word, but they ignore the context in which the word appears in the actual sentence. To address this we compute the final representation of the word by running a separate Bi-LSTM model over the initial vectors of the terminals in the same way as done by Kiperwasser and Goldberg (2016) and Cross and Huang (2016).

3.2 How to Represent Non-Terminal Nodes?

During the parsing process we need to produce the representations of the full subtrees that are going to be placed on the stack. In the dependency parsing literature, many approaches for representing dependency subtrees use the representation of the head word. If the representation of the head word is computed using a model that takes context into account, such as Bi-LSTM models, then this simple architecture can give good results (Kiperwasser and Goldberg, 2016; Cross and Huang, 2016). However, we believe that this is not the right approach for discontinuous constituency parsing. The reason is that, for the parser to know to which constituents it should attach the current constituent, it needs to know which arguments have already been attached and which ones are missing. In other words, even if the head of two different constituents is the same, their representation should be different because they have different requirements.

To address this we use a "composition function" approach where we recursively compute the representation of the constituent. Recursive neural networks (RecNN) (Goller and Küchler, 1996) are one way of accomplishing this. Dyer et al. (2015) use RecNN to compute the representation of the subtrees in the dependency structure. We adapt this model to our case in the following way. For binary constituents (i.e. outputs of *adj*$\curvearrowleft$ and *adj*$\curvearrowright$) the composition function takes the representation of the head constituent h_{head}, the representation of the complement h_{comp} and one single bit that represents the directionality of the $e_{\curvearrowleft}$ in the adjoining operation (0 for *adj*$\curvearrowleft$ and 1 for *adj*$\curvearrowright$). The resulting h_{new} representation is computed as follows:

$$h_{new} = tanh(W_{adj}[h_{head}; h_{comp}; e_{\curvearrowleft}] + b_{adj})$$

Here, semi-colon (;) represents vector concatenation, and W_{adj} and b_{adj} are the weight matrix and the bias vector that are trained together with the rest of the model, to optimize the desired loss function.

The transition *pro(X)* also creates new trees and its composition function can be seen as a function of a Simple RNN model:

$$h_{new} = tanh(W_{pro}[h_{head}; e_{nt}] + b_{pro})$$

Here e_{nt} is the embedding for the non-terminal to which constituent gets promoted. W_{pro} and b_{pro} are again the weight matrix and the bias vector whose values are estimated during training.

Simple RNN models have been shown to suffer from vanishing gradient problem, and for that reason they have been largely replaced with LSTM models (Hochreiter and Schmidhuber, 1997). The same holds for recursive neural network models. Le and Zuidema (2016) have shown that, for deep and complex hierarchical structures, the models that have a memory akin to the memory in LSTM are much more robust towards the vanishing gradient problem. Thus, in our work we use the Tree-LSTM neural architecture from Tai et al. (2015), but the alternative recursive version of LSTM by Le and Zuidema (2015) could be used as well.

In the Tree-LSTM model each constituent is represented by the hidden state h and the *memory cell* c. The composition function for the binary constituents with representations h_{head}, c_{head}, h_{comp} and c_{comp} computes the new representations h_{new} and c_{new} in the following way:

$$\begin{aligned}
f_{head} &= \sigma(W_{11}^{(f)} h_{head} + W_{12}^{(f)} h_{comp} + b_{\curvearrowleft}^{(f)}) \\
f_{comp} &= \sigma(W_{21}^{(f)} h_{head} + W_{22}^{(f)} h_{comp} + b_{\curvearrowleft}^{(f)}) \\
i &= \sigma(W_{1}^{(i)} h_{head} + W_{2}^{(i)} h_{comp} + b_{\curvearrowleft}^{(i)}) \\
o &= \sigma(W_{1}^{(o)} h_{head} + W_{2}^{(o)} h_{comp} + b_{\curvearrowleft}^{(o)}) \\
u &= tanh(W_{1}^{(u)} h_{head} + W_{2}^{(u)} h_{comp} + b_{\curvearrowleft}^{(u)}) \\
c_{new} &= i \odot u + f_{head} \odot c_{head} + f_{comp} \odot c_{comp} \\
h_{new} &= o \odot tanh(c_{new})
\end{aligned}$$

All the W matrices and the bias vectors b are trained parameters of the composition function. For each equation above there is an alternative equation that instead of bias $b_{\curvearrowleft}$ uses bias $b_{\curvearrowright}$. Which equation/bias will be used depends on the directionality of the adjoining operation.

For the promote transition, since it creates only one unary node, we can use almost the same computation as in the standard LSTM:

$$
\begin{aligned}
f &= \sigma(W^{(f)} h_{head} + W_{nt}^{(f)} e_{nt} + b^{(f)}) \\
i &= \sigma(W^{(i)} h_{head} + W_{nt}^{(i)} e_{nt} + b^{(i)}) \\
o &= \sigma(W^{(o)} h_{head} + W_{nt}^{(o)} e_{nt} + b^{(o)}) \\
u &= tanh(W^{(u)} h_{head} + W_{nt}^{(u)} e_{nt} + b^{(u)}) \\
c_{new} &= i \odot u + f \odot c_{head} \\
h_{new} &= o \odot tanh(c_{new})
\end{aligned}
$$

The main difference from the standard LSTM is that here we additionally use the information from the non-terminal embedding e_{nt} to which the constituent is promoted.

3.3 How to Represent a Configuration?

We have covered how to represent syntactic objects (terminal and non-terminal nodes) that are stored in the stack and the buffer, but we still need to decide how to combine these representations to make a final decision about the next transition.

One possibility is to first find a suitable representation for the stack and the buffer individually, concatenate these representations and then apply a multi-layer perceptron (MLP) to produce the probabilities for the next action.

The stack and the buffer can be seen as the same type of data structure: the buffer can be interpreted as a stack that is filled by pushing the words in a sentence from the last to the first. Therefore, we can use same approach for modeling stack and buffer.

The most common approach for representing a stack structure in transition based parsers (both in perceptron and neural models) is to take the representations of the first few top constituents on the top of the stack. Thus, this approach assumes that only the top of the stack and buffer are relevant for deciding the next action. Even though this assumption seems reasonable in the context of continuous constituency parsing, for discontinuous parsing it can be very harmful because the constituents that we want to merge might be very far from each other in the stack, as argued in (Maier and Lichte, 2016).

In our work, we explore an alternative model that could address this problem; namely, the Stack-LSTM model proposed in (Dyer et al., 2015). This model consists of an LSTM that processes the whole stack as a sequence, to obtain in this way a representation of the stack that includes *all* of its elements. This approach gave good results on continuous dependency parsing, but its properties should be even more important for discontinuous parsing, since it allows to keep in the stack a representation of all the constituents.

Given the stack h_{stack} and buffer h_{buffer} representations computed by Stack-LSTMs, we compute the configuration representation h_{conf} by concatenating these vectors and then applying an affine transformation followed by a $ReLU(\cdot)$ nonlinearity:

$$h_{conf} = ReLU(W_{conf}[h_{stack}; h_{buffer}] + b_{conf})$$

This vector representation encodes the whole configuration: the information flow passes trough every character, every POS tag, every constituent in the stack and in the buffer. From this vector representation we can compute the probability of the transition z from the set of possible transitions Z by applying one final softmax layer:

$$p(z|h_{conf}) = \frac{exp(w_z^T h_{conf} + b_{z_i})}{\sum_{z_i \in Z} exp(w_{z_i}^T h_{conf} + b_{z_i})}$$

The probability of the whole sequence of transitions is defined as the product of the probabilities of its transitions:

$$p(\mathbf{z}|\mathbf{w}) = \prod_{i=1}^{|z|} p(z_i|h_{conf\_i})$$

The parameters are optimized for maximum likelihood of the oracle sequence of transitions.

4 Experiments

We empirically test the performance of our parser on two German constituency treebanks: Negra and Tiger. The preprocessing applied to these treebanks follows the same methods used in other discontinuous constituency parsing literature, as described in Maier (2015) and implemented in the *tree-tools* software[2].

We use two different versions of the Negra treebank. The first version is filtered for the sentences up to 30 words, in order to remain comparable to previous grammar-based models; the second version includes sentences of all lengths. As for the Tiger treebank, we use two different splits: TigerHN (Hall and Nivre, 2008) and TigerSPMRL

[2] https://github.com/wmaier/treetools

name	**value**
trained word embedding dim.	100
pretrained word embedding dim.	64
POS embedding dim.	20
character embedding dim.	100
character Bi-LSTM layers	1
word Bi-LSTM layers	2
(non-)terminal node repr. dim.	40
configuration repr. dim.	100
stack LSTM dim.	100
stack LSTM layers	2
buffer LSTM dim.	100
buffer LSTM layers	2
optimizer	Adam
optimizer parameter b1	0.9
optimizer parameter b2	0.999
beam size	16

Table 1: Hyper-parameters of the model

	#swaps	jump size
Eager	43.17 ±49.21	1.00 ± 0.0
Lazy	10.96 ± 9.96	6.88 ±5.54
Lazier	5.40 ± 3.05	10.03 ±11.05

Table 2: Average number of swaps and jump sizes per sentence

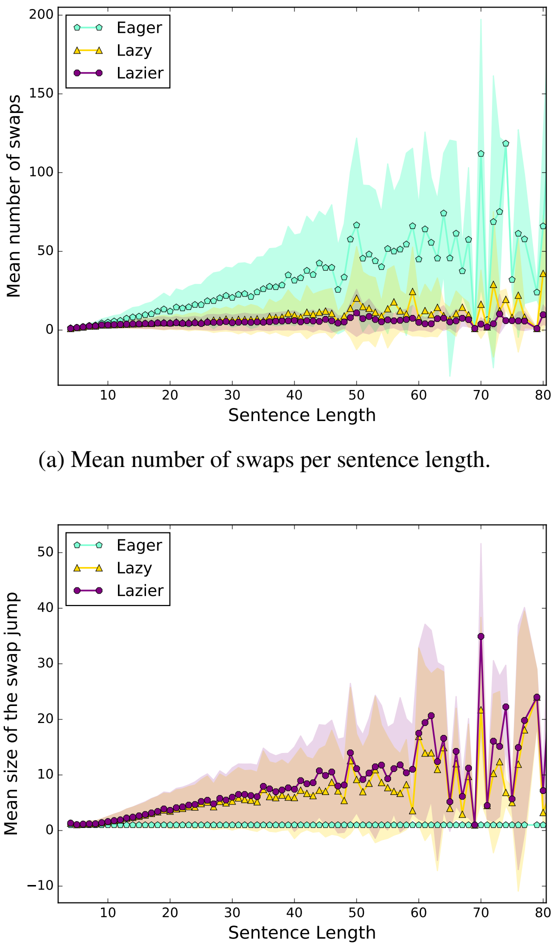
(Maier, 2015). We evaluated the model with the evaluation module of *discodop*[3] parser.

Our model is implemented with DyNet (Neubig et al., 2017) and the code is available at `https://github.com/stanojevic/BadParser`. The concrete hyper-parameters of our model are shown in Table 1. We optimize the parameters with Adam optimizer on the training set, for 10 iterations with 100 random restarts, and we do model selection on the validation set for the F-score. During test time we use beam search with beam of size 16.

We conducted the development of our model on the TigerHN train and development sets. First we will analyze the effect of different model design decisions and then we show the results over the test set. The development set scores on TigerHN are shown in Table 3.

4.1 Which oracle is better?

The results in Table 3 show that the Eager oracle works better than Lazy for discontinuous constituents, but for continuous constituents (and over all constituents on average) Lazy works better. This can be explained by Lazy being very conservative about swaps: since their number is significantly reduced, the transition becomes difficult to predict, and thus the model gives up on predicting swaps and concentrates on the statistics for the projective operations. In other words, Lazy predicts swaps only if the statistical evidence for swaps is high. This can be seen by the contrast between high precision but very low recall on discontinuous constituents.



(a) Mean number of swaps per sentence length.

(b) Mean size of the swap jumps per sentence length.

Figure 3: The effect of different swap strategies of sentences with up to 80 words

Eager works in the opposite direction: since it has observed many swaps it has a strong bias to predict them, which leads to a high recall but low precision. Lazier strikes a good balance between precision and recall on the discontinuous constituents, and because of that it outperforms both Eager and Lazy on F-score for both all constituents and discontinuous constituents.

The good result of Lazier cannot be subscribed only to the shorter transition sequences being easier to predict, because if it was up to the transition

[3] `https://github.com/andreasvc/disco-dop`

Composition function	Bi-LSTM layers	Stack-LSTM	Oracle	All				Discontinuous			
				P	R	F	E	P	R	F	E
Tree-LSTM	**2**	✓	**Lazier**	**84.85**	**83.81**	**84.33**	**46.64**	55.67	**46.54**	**50.70**	**37.48**
Head	2	✓	Lazier	61.86	53.91	57.61	13.87	16.43	6.63	9.45	3.49
RecNN	2	✓	Lazier	83.57	82.84	83.21	43.55	57.30	38.17	45.82	32.94
Tree-LSTM	0	✓	Lazier	81.31	80.30	80.80	39.85	48.17	33.67	39.64	27.37
Tree-LSTM	1	✓	Lazier	83.86	83.67	83.77	44.46	**58.12**	42.57	49.15	35.08
Tree-LSTM	2	top3	Lazier	82.29	81.07	81.68	40.64	54.39	31.49	39.89	26.57
Tree-LSTM	2	✓	Lazy	84.80	83.39	84.09	45.65	59.35	36.24	45.00	31.88
Tree-LSTM	2	✓	Eager	83.74	83.04	83.39	43.81	50.15	43.68	46.69	33.49

Table 3: Precision (P), Recall (R), F-score (F) and Exact (E), for our best model and ablated versions.

		Negra	**Negra-All**		**TigerHN**		**TigerSPMRL**
		All	**L≤40**	**All**	**L≤40**	**All**	
conv2dep	Hall and Nivre (2008)	-	-	-	79.93	-	-
	Fernández-González and Martins (2015)	82.56 ⋆	81.08	80.52	**85.53**	**84.22**	80.62◇
LCFRS	van Cranenburgh (2012)	-	72.33	71.08	-	-	-
	van Cranenburgh and Bod (2013)	-	76.8	-	-	-	-
	Kallmeyer and Maier (2013)	75.75	-	-	-	-	-
Transition-Based	Versley (2014)	-	-	-	74.23	-	-
	Maier (2015)	76.95	-	-	79.52	-	74.71
	Maier and Lichte (2016)	-	-	-	80.02	-	76.46
	Coavoux and Crabbé (2017)	82.46	82.76	82.16	85.11	84.01	81.60
	This work	**83.29**	**83.39**	**82.87**	85.25	84.06	**81.64**

Table 4: Final results on test set, computed with *discodop* evaluation module. ⋆Trained on Negra-All. ◇Evaluated with SPRML scripts.

sequences length alone then Lazy would work better than Eager on the discontinuous constituents. The more likely explanation is that Lazier introduces an inductive bias in the model that is useful for generalization, and that allows the model to generalize better than Eager and Lazy.

We also quantified how many swaps are made by Eager, Lazy and Lazier. Figure 3 shows the statistics over the TigerHN training set for different sentence lengths; the aggregated statistics over all sentence lengths can be read in Table 2. We can observe in Figure 3(a) that, in the case of short sentences, all the swapping strategies give similar results, but as sentences get longer the number of swaps in Eager gets much higher and more unstable than lazier alternatives. We found that for some sentences Lazy and Lazier do with 2 swaps what Eager does with 126 swaps. Compared to Lazy, Lazier is much more stable in terms of the number of swaps, which can be seen by the standard deviation in Table 2. In Figure 3(b) shows a similar trend for the size of the jump of swap transitions. All the swaps of Eager make a jump of size 1, while the jumps of Lazier can go up to 91 words.

4.2 What is the best word representation?

We have tested whether the representation of a word based solely on its embeddings is enough to get good results or, instead, this representation should be refined by the bi-directional LSTM. Table 3 shows that adding layers to the bi-directional LSTM consistently improves the scores. The difference between not using a bi-directional LSTM and using 2 layers of bi-directional LSTM is 3.63

F-score, which is a big margin. Adding a third layer did not improve scores significantly.

4.3 What is the best composition function?

We have tried three options for composition functions: Head (use only the head word embedding instead of a composition function), RecNN and TreeLSTM – all presented in Section 3.2. As we expected, the head representation alone did not perform well, which shows that some type of composition function is needed. We find that using a recursive model with a memory cell improves results by 1.12 F-score, and thus we settle for the TreeLSTM composition function.

4.4 What is the best configuration representation?

We tested two configuration representations: the first one – top3 – takes the 3 topmost elements from the stack and the buffer as the representatives, while the second one – Stack-LSTM – models the whole content of the configuration via recurrent neural models. In line with our intuitions, the Stack-LSTM, thanks to considering the whole stack and buffer structure instead of only a few elements, outperforms top3 by a margin of 2.65 F-score points.

4.5 Comparison with other models

We took the version of our model that performed the best on the TigerHN development set and compared it on the four different datasets (two treebanks with two different splits) with other parsers.

In Table 4 we show the results compared to the other works published on these datasets. Our parser outperforms all the previously published models on all datasets except TigerHN, where it ends up second best after Fernández-González and Martins (2015). As shown in our previous analysis, exploring alternative representations of the different components has allowed us to construct a better model. We must also notice that, when comparing to other models, one influential cause of the good performance may be the capacity of our model, provided by the neural architectures. Neural networks allow modeling relations from input to output that are much more complex than those captured by the approaches we compare to, most of which use linear models based on perceptron or simple PCFG type of generative models.

We have also tested our model on the predicted POS tags from TigerSPMRL split, as provided in the shared task (Seddah et al., 2013). The results are shown in Table 5. The biggest strength of our model—its capacity— is in this case its biggest weakness: it causes the model parameters to overfit the noisy predicted tags during training, because we have not used any form of regularization. Model combinations like the one in Versley (2016) do not suffer from this because they implicitly do strong regularization. Our model could probably achieve better results on this dataset with stronger regularization, which we leave for future research.

TigerSPRML	F1 (spmrl.prm)	
	≤ 70	All
Versley (2014)	73.90	–
This work	77.25	76.96
F&M (2015)	77.72	77.32
Coavoux&Crabbé (2017)	79.44	79.26
Versley (2016)	79.84	79.50

Table 5: Results on SPMRL data with predicted tags.

5 Conclusion

We have presented the first neural model for discontinuous constituency parsing that achieves state-of-the-art results in three out of four standard datasets for discontinuous parsing with gold POS tags. Our findings suggest that i) bidirectional LSTM should be used for refining the representations of terminals even in the cases when they are going to be combined by a recursive model, ii) the performance of the composition function depends to a big extent on the availability of the memory cells, to prevent the vanishing gradient, iii) it is crucial to use all the elements in the stack and buffer in the decision process instead of just few elements on the top and iv) Lazier oracle gives better and more stable results than Eager and Lazy oracles on both continuous and discontinuous constituents.

Acknowledgments

We are thankful to Andreas van Cranenburgh and Wolfgang Maier for sharing the data and preprocessing scripts, and to the anonymous reviewers for suggestions that improved the paper. We are grateful to the ILLC for providing us with traveling funds to attend the conference.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Sofia, Bulgaria, pages 183–192. http://www.aclweb.org/anthology/W13-3520.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 349–359. http://aclweb.org/anthology/D15-1041.

Maximin Coavoux and Benoit Crabbé. 2017. Incremental discontinuous phrase structure parsing with the gap transition. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Valencia, Spain.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, pages 1–8.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 111.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 32–37. http://anthology.aclweb.org/P16-2006.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 334–343. http://www.aclweb.org/anthology/P15-1033.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1523–1533. http://www.aclweb.org/anthology/P15-1147.

Christoph Goller and Andreas Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the International Conference on Neural Networks*. IEEE, pages 347–352.

Johan Hall and Joakim Nivre. 2008. Parsing discontinuous phrase structure with grammatical functions. In *Proceedings of the 6th International Conference on Natural Language Processing (GoTAL)*. pages 169–180.

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

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL HLT '12, pages 142–151. http://dl.acm.org/citation.cfm?id=2382029.2382049.

Laura Kallmeyer and Wolfgang Maier. 2013. Data-driven parsing using probabilistic linear context-free rewriting systems. *Computational Linguistics* 39(1):87–119.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics* 4:313–327. https://transacl.org/ojs/index.php/tacl/article/view/885.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, Denver, Colorado, pages 10–19. http://www.aclweb.org/anthology/S15-1002.

Phong Le and Willem Zuidema. 2016. Quantifying the vanishing gradient and long distance dependency problem in recursive neural networks and recursive LSTMs. In *ACL Workshop on Representation Learning for NLP*.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1520–1530. http://aclweb.org/anthology/D15-1176.

Wolfgang Maier. 2015. Discontinuous incremental shift-reduce parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint*

Conference on Natural Language Processing (Volume 1: Long Papers). Association for Computational Linguistics, Beijing, China, pages 1202–1212. http://www.aclweb.org/anthology/P15-1116.

Wolfgang Maier and Timm Lichte. 2016. Discontinuous parsing with continuous trees. In *Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing*. Association for Computational Linguistics, San Diego, California, pages 47–57. http://www.aclweb.org/anthology/W16-0906.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. pages 351–359. http://www.aclweb.org/anthology/P/P09/P09-1040.pdf.

Joakim Nivre, Marco Kuhlmann, and Johan Hall. 2009. An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT)*. pages 73–76.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*. Association for Computational Linguistics, Seattle, Washington, USA, pages 146–182. http://www.aclweb.org/anthology/W13-4917.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1556–1566. http://www.aclweb.org/anthology/P15-1150.

Andreas van Cranenburgh. 2012. Efficient parsing with linear context-free rewriting systems. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Avignon, France, pages 460–470. http://www.aclweb.org/anthology/E12-1047.

Andreas van Cranenburgh and Rens Bod. 2013. Discontinuous parsing with an efficient and accurate DOP model. In *Proceedings of the 13th International Conference on Parsing Technologies (IWPT)*.

Yannick Versley. 2014. Experiments with easy-first nonprojective constituent parsing. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*. Dublin City University, Dublin, Ireland, pages 39–53. http://www.aclweb.org/anthology/W14-6104.

Yannick Versley. 2016. Discontinuity reˆ2-visited: A minimalist approach to pseudoprojective constituent parsing. In *Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing*. Association for Computational Linguistics, San Diego, California, pages 58–69. http://www.aclweb.org/anthology/W16-0907.

Stack-based Multi-layer Attention for Transition-based Dependency Parsing

Zhirui Zhang[1], **Shujie Liu**[2], **Mu Li**[2], **Ming Zhou**[2], **Enhong Chen**[1]
University of Science and Technology of China, Hefei, China[1]
Microsoft Research Asia, Beijing, China[2]
zr011036@mail.ustc.edu.cn cheneh@ustc.edu.cn[1]
{shujie,muli,mingzhou}@microsoft.com[2]

Abstract

Although sequence-to-sequence (seq2seq) network has achieved significant success in many NLP tasks such as machine translation and text summarization, simply applying this approach to transition-based dependency parsing cannot yield a comparable performance gain as in other state-of-the-art methods, such as stack-LSTM and head selection. In this paper, we propose a stack-based multi-layer attention model for seq2seq learning to better leverage structural linguistics information. In our method, two binary vectors are used to track the decoding stack in transition-based parsing, and multi-layer attention is introduced to capture multiple word dependencies in partial trees. We conduct experiments on PTB and CTB datasets, and the results show that our proposed model achieves state-of-the-art accuracy and significant improvement in labeled precision with respect to the baseline seq2seq model.

1 Introduction

Deep learning models have been proven very effective in solving various NLP problems such as language modeling, machine translation and syntactic parsing. For dependency parsing, one line of research aims to incrementally integrate distributed word representations into classic dependency parsing (Chen and Manning, 2014; Weiss et al., 2015; Andor et al., 2016; Cross and Huang, 2016; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2016). Another line of research attempts to leverage end-to-end neural network to perform dependency parsing, such as stack-LSTM and sequence-to-sequence (seq2seq) model (Dyer et al., 2015; Zhang et al., 2017; Wiseman and Rush, 2016). Recently seq2seq model has made significant success in many NLP tasks, such as machine translation and text summarization (Cho et al., 2014; Sutskever et al., 2014; Rush et al., 2015). Unfortunately, to our best knowledge, simply applying seq2seq model to transition-based dependency parsing cannot achieve comparable results as in other state-of-the-art methods like stack-LSTM and head selection (Dyer et al., 2015; Zhang et al., 2017).

One issue with the simple seq2seq neural network for dependency parsing is that structural linguistic information, which plays a key role in classic transition-based or graph-based dependency parsing models, cannot be explicitly employed. For example, classic transition-based parsing algorithm utilizes a stack to manage the heads of partial sub-trees and leverages these evidents for action selection, while such information is missing from current seq2seq models. Another problem is related to the limit of the conventional attention network being used in seq2seq network, which is unable to capture dependencies between words in the input. As a matter of fact, various types of features (word unigram, bigram, trigram, ...) traditionally adopted by transition-based parsing algorithm are usually ignored by the current attention mechanism, but they are very important to capture word dependencies in generated partial trees.

In this paper, we propose a stack-based multi-layer attention mechanism to solve the above problems. To simulate the stack used in the transition-based dependency parsing, we introduce two binary vectors, one indicates whether a word is pushed into the stack, and another indicates whether a word is popped out from it. To model the complex structural information, we propose a multi-layer attention based on the stack information, previous action and input sentence.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1677–1682
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

The multi-layer attention aims to capture multiple word dependencies in partial trees for action prediction.

We evaluate our model on English and Chinese datasets. Experimental results show that our proposed model can significantly outperform the basic seq2seq model with 1.87 UAS (English) and 1.61 UAS (Chinese), matching the state-of-the-art parsing performance. With 4 models ensembled, we obtain further improvements with accuracies of 94.16 UAS (English) and 87.97 UAS (Chinese).

2 Neural Model for Sequence-to-Sequence Learning

In this work, we follow the encoder-decoder architecture proposed by Bahdanau et al. (2015). The whole architecture can be divided into three components: encoder, decoder and attention.

Encoder: The encoder reads in the source sentence $X = (x_1, x_2, \ldots, x_T)$ and transforms it into a sequence of hidden states $h = (h_1, h_2, \ldots, h_T)$, using a bi-directional recurrent neural network that is usually implemented as Gated Recurrent Unit (GRU) (Cho et al., 2014) or Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997).

Attention Mechanism: The context vector c_i is a weighted sum of the hidden states $(h_1, h_2, \ldots, h_T)$ with the coefficients $\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,T}$ computed by

$$\alpha_{i,t} = \frac{\exp(e_{i,t})}{\sum_k \exp(e_{i,k})} \qquad (1)$$

$$e_{i,t} = v_a^\top \tanh(W_a z_{i-1} + U_a h_t) \qquad (2)$$

where v_a,W_a, U_a are the weight matrices.

Decoder: The decoder uses another recurrent neural network to generate a corresponding target sequence $Y = (y_1, y_2, \ldots, y_{T'})$ based on the encoded sequence of hidden state h. At each time i, the conditional probability of target symbol y_i is computed by

$$z_i = \mathrm{RNN}([y_{i-1}; c_i], z_{i-1}) \qquad (3)$$

$$p(y_i | y_{<i}, h) = \mathrm{softmax}(g(y_{i-1}, z_i, c_i)) \qquad (4)$$

where g is a non-linear function, z_i is the i_{th} hidden state of the decoder, and it is calculated conditional on the previous hidden state z_{i-1}, previous target symbol y_{i-1} and source context vector c_i.

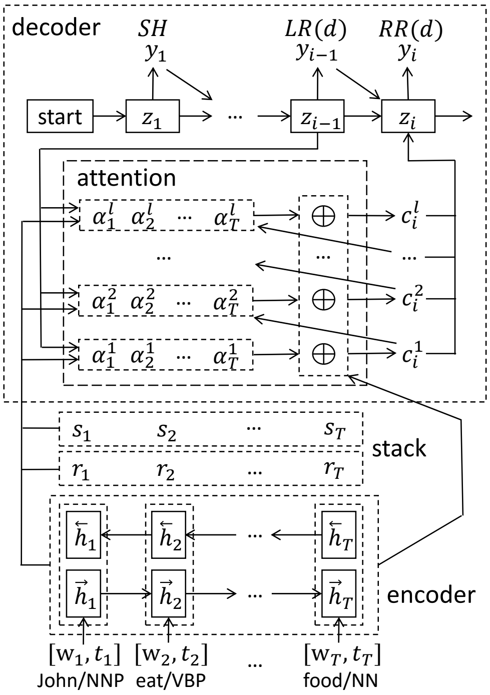

decoder
SH
y_1
$LR(d)$
y_{i-1}
$RR(d)$
y_i
start
z_1
...
z_{i-1}
z_i
attention
$\alpha_1^l \; \alpha_2^l \; \cdots \; \alpha_T^l$
c_i^l
...
$\alpha_1^2 \; \alpha_2^2 \; \cdots \; \alpha_T^2$
c_i^2
$\alpha_1^1 \; \alpha_2^1 \; \cdots \; \alpha_T^1$
c_i^1
s_1 s_2 ... s_T
stack
r_1 r_2 ... r_T
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ ... $\overleftarrow{h}_T$
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ ... $\overrightarrow{h}_T$
encoder
$[w_1, t_1]$ John/NNP
$[w_2, t_2]$ eat/VBP
...
$[w_T, t_T]$ food/NN

Figure 1: The architecture of sequence-to-sequence parsing model. SH, LR(d), RR(d) denote the SHIFT, LEFT-ARC(d), RIGHT-ARC(d) transitions in arc-standard algorithm and d is arc-label.

3 Sequence-to-Sequence Parsing Model

Transition-based dependency parsing conceptualizes the process of transforming a sentence into a dependency tree as a sequence of actions. It can be formulated as a sequence-to-sequence problem, and seq2seq framework can be applied. Compared with other tasks, such as machine translation, dependency parsing not only considers the previous action and input sentence, but also requires many structure information, such as the subtree structure during the parsing. Such information plays an important role in transition-based dependency parsing, so traditional methods adopt a stack to save structure information and design different type of features (word unigram, bigram, trigram, ...) to model them. However, vanilla seq2seq models have no explicit structure to model these necessary structure information. To better leverage the structure information, we extend the basic seq2seq architecture with a simulated stack and multi-layer attention, as illustrated in Figure 1. The main structure (encoder, decoder and attention part in Figure 1) of our parsing model is detailed below.

Encoder: As shown in the encoder part of Figure 1, to utilize POS tag information, each word w_i is additionally represented by x_i, the concatenation of two vectors corresponding to w_i's lexical and POS tag t_i embedding: $x_i = [W_e * e(w_i); W_t * e(t_i)]$, where $e(w_i)$ and $e(t_i)$ are one-hot vector representations of token w_i and its POS tag t_i, W_e and W_t are embedding matrices. The rest part of the encoder is the same with the basic seq2seq model.

Attention Mechanism: We improve the attention part in two aspects: introduction of stack information and multi-layer attention structure.

Stack information, which plays an essential role in the conventional algorithm, is simulated with two binary vectors $s = (s_1, \dots, s_T)$ and $r = (r_1, \dots, r_T)$ to record the state of each word w_i and initialized to zero. When parser pushes word w_i into stack, s_i is assigned to 1, while r_i is assigned to 1 only if word w_i is removed from stack. Intuitively, at each time step i in the decoding phase, stack information serves as an additional input to the attention model, which provides complementary information of that the source words is in the stack or not. We expect the stack information would guide the attention model to focus more on words in the stack. More formally, the coefficients $\alpha_1, \alpha_2, \dots, \alpha_T$ used in attention mechanism can be rewritten as

$$\alpha_{i,t} = \frac{\exp(e_{i,t}) * (1 - r_t)}{\sum_k \exp(e_{i,k}) * (1 - r_k)} \tag{5}$$

$$e_{i,t} = v_a^\top \tanh(W_a z_{i-1} + U_a h_t + S_a s_t) \tag{6}$$

where S_a is the weight matrix.

To extract complex structure information to help action prediction, we apply a l-layers network structure for attention mechanism as shown in the attention part of Figure 1. To further enhance connection between adjacent layers, we replace the state z_{i-1} in Equation 6 by the concatenation of z_{i-1} and context vector c_i^{m-1} at each layer $m(m > 1)$. The Equation 6 can be rewritten as:

$$e_{i,t}^m = v_a^\top \tanh(W_a^m [z_{i-1}; c_i^{m-1}] + U_a h_t + S_a s_t) \tag{7}$$

where W^m is the weight matrix. With this network structure, we obtain different context vectors $(c_i^1, c_i^2, \dots, c_i^l)$, and the final context vector c_i, which is considered as complex context information, is replaced by the concatenation of those vectors: $c_i = [c_i^1; c_i^2; \dots; c_i^l]$.

Decoder: Unlike machine translation and text summarization in which seq2seq model is widely applied, a sequence of action in dependency parsing must satisfy some constraints so that they can generate a dependency tree. Following the arc-standard algorithm (Nivre, 2004), the precondition can be categorized as 1) SHIFT(SH): There exists at least one word that is not pushed into the stack; 2) LEFT-ARC(LR(d)) and RIGHT-ARC(RR(d)): There are at least two words in the stack. These two constraints can be defined as indicator functions

$$I(y_i) = \begin{cases} 0 & y_i = \text{SH}, W_c \le 0 \\ 0 & y_i = \text{LR(d) or RR(d)}, S_c < 2 \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

where S_c represents the number of words in the stack and W_c is the number of source words that are not pushed into the stack. To introduce these constraints, the conditional probability of each target symbol y_i can be rewritten as

$$p(y_i | y_{<i}, h) = \frac{\exp(g_i) * I(y_i)}{\sum_k \exp(g_k) * I(y_k)} \tag{9}$$

where g_i is the ith element of $g(y_{i-1}, z_i, c_i)$.

4 Experiments

In this section, we evaluate our parsing model on the English and Chinese datasets. Following Dyer et al. (2015), Stanford Dependencies (de Marneffe and Manning, 2008) conversion of the Penn Treebank (PTB) (Marcus et al., 1993) and Chinese Treebank 5.1 (CTB) are adopted. We leverage the arc-standard algorithm for our dependency parsing. In addition, we limit the vocabulary to contain up to 20k most frequent words and convert remaining words into the <unk> token.

4.1 Setup

For our model, 3-layers GRU is used for encoder and decoder. The dimension of word embedding is 300, the dimension of POS-tag/action embedding is 32, and the size of hidden units in GRU is 500. 3-layers attention structure is adopted in our model. Following Chen and Manning (2014); Dyer et al. (2015), we used 300-dimensional pretrained GloVe vectors (Pennington et al., 2014) to initialize our word embedding matrix. Other model parameters are initialized using a normal distribution with a mean of 0 and a variance of

Parser	PTB-SD				CTB			
	Dev		Test		Dev		Test	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Z&N11	-	-	93.00	90.95	-	-	86.00	84.40
C&M14	92.20	89.70	91.80	89.60	84.00	82.40	83.90	82.40
ConBSO	-	-	91.57	87.26	-	-	-	-
Dyer15	93.20	90.90	93.10	90.90	87.20	85.90	87.20	85.70
Weiss15	-	-	93.99	92.05	-	-	-	-
K&G16	-	-	93.99	91.90	-	-	87.60	86.10
DENSE	**94.30**	91.95	94.10	91.90	87.35	85.85	87.84	86.15
seq2seq	92.02	89.10	91.84	88.84	86.21	83.80	85.80	83.53
Our model	93.65	91.52	93.71	91.60	87.28	85.30	87.41	85.40
Ensemble	94.24	**92.01**	**94.16**	**92.13**	**88.06**	**86.30**	**87.97**	**86.18**

Table 1: Results of various state-of-the-art parsing systems on English dataset (PTB with Stanford Dependencies) and Chinese dataset (CTB). The numbers reported from different systems are taken from: Z&N11 (Zhang and Nivre, 2011); C&M14 (Chen and Manning, 2014); ConBSO (Wiseman and Rush, 2016); Dyer15 (Dyer et al., 2015); Weiss15 (Weiss et al., 2015); K&G16 (Kiperwasser and Goldberg, 2016); DENSE (Zhang et al., 2017).

$\sqrt{6/(d_{row}+d_{col})}$, where d_{row} and d_{col} are the number of rows and columns (Glorot and Bengio, 2010). Our models are trained on a Tesla K40m GPU and optimized with vanilla SGD algorithm with mini-batch size 64 for English dataset and 32 for Chinese dataset. The initial learning rate is set to 2 and will be halved when unlabeled attachment scores (UAS) on the development set do not increase for 900 batches. To alleviate the gradient exploding problem, we rescale the gradient when its norm exceeds 1. Dropout (Srivastava et al., 2014) is applied to our model with the strategy recommended in Zaremba et al. (2014) and the dropout rate is 0.2. For testing, beam search is employed to find the best action sequence with beam size 8. For evaluation, we report unlabeled (UAS) and labeled attachment scores (LAS) on the development and test sets. Following Chen and Manning (2014), the punctuation is excluded from the evaluation.

4.2 Main Results

Table 1 lists the accuracies of our parsing models, compared to other state-of-the-art parsers. For the baseline, seq2seq model employs the same encoder and decoder network structure with our model. We can see that our proposed model can significantly outperform the basic seq2seq model with 1.87 UAS (English) and 1.61 UAS (Chinese) improvements on the test set. This demonstrates the effectiveness of our proposed multi-layer attention mechanism. Besides, our model achieves better UAS accuracy than Z&N11, C&M14, ConBSO and Dyer15 on development and test set, while slightly lower than Weiss15, K&G16 and DENSE. Weiss15 adopts a structured training procedure which can be easily applied to our model as well, and it will further improve the performance of our model. K&G16 uses 11 bidirectional LSTM vectors as features, which will be fed to a transition-based parser. It suggests a new direction that combines our model with feature engineering of the traditional transition-based parser to gain better performance. DENSE formalizes dependency parsing as head selection and applies MST algorithms to correct non-tree outputs, while our model doesn't require any post-processing at test time. Dozat and Manning (2016) use deep biaffine attention instead of traditional attention in the graph-based architectures of K&G16, achieving 95.74 UAS and 89.30 UAS on PTB-SD and CTB datasets respectively. For ensemble, we train 4 models using the same network with different random initialization. When we ensemble these 4 models, we simply average the output probabilities from different models and obtain the better result with accuracies of 94.16 UAS (English) and 87.97 UAS (Chinese) as shown in the Table 1.

4.3 Impact of l

The hyper-parameter l represents the number of layers in our proposed multi-layer attention mech-

	Dev		Test	
	UAS	LAS	UAS	LAS
seq2seq	92.02	89.10	91.84	88.84
$l = 1$	92.85	90.44	92.70	90.40
$l = 2$	93.30	91.13	93.21	90.98
$l = 3$	**93.65**	**91.52**	**93.71**	**91.60**
$l = 4$	93.49	91.29	93.42	91.24

Table 2: Impact of l on English PTB dataset.

	Dev		Test	
	UAS	LAS	UAS	LAS
Our model	93.65	91.52	93.71	91.60
–pretraining	93.19	90.92	93.22	91.11
–POS	92.73	89.86	92.57	90.05
–s vector	93.18	90.68	93.02	90.89
–r vector	93.16	90.90	93.27	91.02

Table 3: Impact of the different components on English PTB dataset.

anism. Larger l would bring more capacity, but lead to more computational complexity and aggravate the risk of over-fitting.

We conduct a group of experiments to investigate the impact of l. The results are shown in Table 2. Seq2seq model can be viewed as a special case of our model without any stack information. With $l = 1$, we can see that the introduction of stack information can strongly improve the parsing performance, especially for LAS. When l is small ($l < 4$), the general trend is that larger l leads to better result. However, further increasing l bring slightly damages to the parsing performance due to the over-fitting problem.

Although larger l would bring more capacity, multiple layers structure will double the training time compared with the vanilla seq2seq. In our implementation, our model costs about 500 seconds for a round of training data on English PTB dataset, while the vanilla costs about 260 seconds.

4.4 Additional Results

We perform some ablation experiments in order to quantify the effect of the different components on our models. As shown in Table 3, the POS-tag information plays the most important role in our model. We note that, different from Dyer et al. (2015), we don't utilize an external word embedding to tackle OOV problem, and it may cause our model to be more dependent on the POS-tag information. For s and r vectors, same as discussion in last section, we find that the introduction of stack information can strongly improve the parsing performance.

5 Conclusion

In order to leverage structure information for seq2seq based dependency parsing, in this paper, we propose a stack based multi-layer attention method, in which, stack is simulated with two binary vectors, and multi-layer attention is introduced to capture multiple word dependencies in partial trees. Experimental results demonstrate that our proposed model significantly outperforms the basic seq2seq model, and achieves a state-of-the-art parsing performance.

In the future, we plan to apply our approach in more languages and other transition-based system, such as arc-eager or arc-hybrid. Another direction we are interested in is to train our model with complex training approaches proposed in Weiss et al. (2015) and Andor et al. (2016).

Acknowledgments

We appreciate Dongdong Zhang, Nan Yang, Shuangzhi Wu and Qingyu Zhou for the fruitful discussions. We also thank the anonymous reviewers for their careful reading of our paper and insightful comments.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *CoRR*, abs/1603.06042.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations (ICLR)*.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37, Berlin, Germany. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256.

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

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *CoRR*, abs/1603.04351.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. Stanford typed dependencies manual.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57. Association for Computational Linguistics.

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

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

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

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *ACL*.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency parsing as head selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 665–676, Valencia, Spain. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *ACL*.

Dependency Grammar Induction with Neural Lexicalization and Big Training Data[*]

Wenjuan Han, Yong Jiang and **Kewei Tu**
{hanwj, jiangyong ,tukw}@shanghaitech.edu.cn
School of Information Science and Technology
ShanghaiTech University, Shanghai, China

Abstract

We study the impact of big models (in terms of the degree of lexicalization) and big data (in terms of the training corpus size) on dependency grammar induction. We experimented with L-DMV, a lexicalized version of Dependency Model with Valence (Klein and Manning, 2004) and L-NDMV, our lexicalized extension of the Neural Dependency Model with Valence (Jiang et al., 2016). We find that L-DMV only benefits from very small degrees of lexicalization and moderate sizes of training corpora. L-NDMV can benefit from big training data and lexicalization of greater degrees, especially when enhanced with good model initialization, and it achieves a result that is competitive with the current state-of-the-art.

1 Introduction

Grammar induction is the task of learning a grammar from a set of unannotated sentences. In the most common setting, the grammar is unlexicalized with POS tags being the tokens, and the training data is the WSJ10 corpus (the Wall Street Journal corpus with sentences no longer than 10 words) containing no more than 6,000 training sentences (Cohen et al., 2008; Berg-Kirkpatrick et al., 2010; Tu and Honavar, 2012).

Lexicalized grammar induction aims to incorporate lexical information into the learned grammar to increase its representational power and improve the learning accuracy. The most straightforward approach to encoding lexical information is full lexicalization (Pate and Johnson, 2016; Spitkovsky et al., 2013). A major problem with full lexicalization is that the grammar becomes much larger and thus learning is more data demanding. To mitigate this problem, Headden et al. (2009) and Blunsom and Cohn (2010) used partial lexicalization in which infrequent words are replaced by special symbols or their POS tags. Another straightforward way to mitigate the data scarcity problem of lexicalization is to use training corpora larger than the standard WSJ corpus. For example, Pate and Johnson (2016) used two large corpora containing more than 700k sentences; Marecek and Straka (2013) utilized a very large corpus based on Wikipedia in learning an unlexicalized dependency grammar. Finally, smoothing techniques can be used to reduce the negative impact of data scarcity. One example is Neural DMV (NDMV) (Jiang et al., 2016) which incorporates neural networks into DMV and can automatically smooth correlated grammar rule probabilities.

Inspired by this background, we conduct a systematic study regarding the impact of the degree of lexicalization and the training data size on the accuracy of grammar induction approaches. We experimented with a lexicalized version of Dependency Model with Valence (L-DMV) (Klein and Manning, 2004) and our lexicalized extension of NDMV (L-NDMV). We find that L-DMV only benefits from very small degrees of lexicalization and moderate sizes of training corpora. In comparison, L-NDMV can benefit from big training data and lexicalization of greater degrees, especially when it is enhanced with good model initialization. The performance of L-NDMV is competitive with the current state-of-the-art.

2 Methods

2.1 Lexicalized DMV

We choose to lexicalize an extended version of DMV (Gillenwater et al., 2010). We adopt a sim-

[*]This work was supported by the National Natural Science Foundation of China (61503248).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1683–1688
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

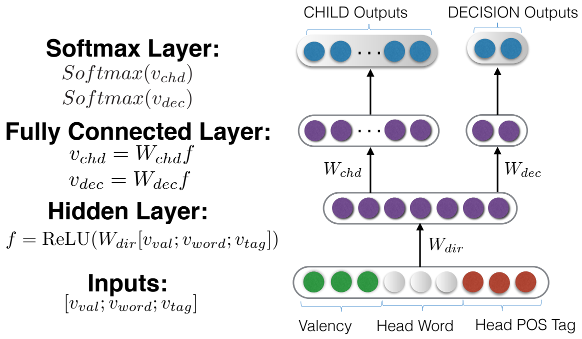


Figure 1: The structure of the neural networks in the L-NDMV model. It predicts the probabilities of the CHILD rules and DECISION rules.

ilar approach to that of Spitkovsky et al. (2013) and Blunsom and Cohn (2010) and represent each token as a word/POS pair. If a pair appears infrequently in the corpus, we simply ignore the word and represent it only with the POS tag. We control the degree of lexicalization by replacing words that appear less than a cutoff number in the WSJ10 corpus with their POS tags. With a very large cutoff number, the grammar is virtually unlexicalized; but when the cutoff number becomes smaller, the grammar becomes closer to be fully lexicalized. Note that our method is different from previous practice that simply replaces rare words with a special "unknown" symbol (Headden III et al., 2009). Using POS tags instead of the "unknown" symbol to represent rare words can be helpful in the neural approach introduced below in that the learned word vectors are more informative.

2.2 Lexicalized NDMV

With a larger degree of lexicalization, the grammar contains more tokens and hence more parameters (i.e., grammar rule probabilities), which require more data for accurate learning. Smoothing is a useful technique to reduce the demand for data in this case. Here we employ a neural approach to smoothing. Specifically, we propose a lexicalized extension of neural DMV (Jiang et al., 2016) and we call the resulting approach L-NDMV.

Extended Model: The model structure of L-NDMV is similar to that of NDMV except for the representations of the head and the child of the CHILD and DECISION rules. The network structure for predicting the probabilities of CHILD rules $[p_{c_1}, p_{c_2}, ..., p_{c_m}]$ (m is the vocabulary size; c_i is the i-th token) and DECISION rules $[p_{stop}, p_{continue}]$ given the head word, head POS tag, direction and valence is shown in Figure 1. We denote the input continuous representations of the head word, head POS tag and valence by v_{word}, v_{tag} and v_{val} respectively. By concatenating these vectors we get the input representation to the neural network: $[v_{val}; v_{word}; v_{tag}]$. We map the input representation to the hidden layer f using the direction-specific weight matrix W_{dir} and the ReLU activation function. We represent all the child tokens with matrix $W_{chd} = [W_{word}, W_{tag}]$ which contains two parts: child word matrix $W_{word} \in \mathbb{R}^{m \times k}$ and child POS tag matrix $W_{tag} \in \mathbb{R}^{m \times k'}$, where k and k' are the pre-specified dimensions of output word vectors and tag vectors respectively. The i-th rows of W_{word} and W_{tag} represent the output continuous representations of the i-th word and its POS tag respectively. Note that for two words with the same POS tag, the corresponding POS tag representations are the same. We take the product of f and the child matrix W_{chd} and apply a softmax function to obtain the CHILD rule probabilities. For DECISION rules, we replace W_{chd} with the decision weight matrix W_{dec} and follow the same procedure.

Extended Learning Algorithm: The original NDMV learning method is based on hard-EM and is very time-consuming when applied to L-NDMV with a large training corpus. We propose two improvements to achieve significant speedup. First, at each EM iteration we collect grammar rule counts from a different batch of sentences instead of from the whole training corpus and train the neural network using only these counts. Second, we train the same neural network across EM iterations without resetting. More details can be found in the supplementary material. Our algorithm can be seen as an extension of online EM (Liang and Klein, 2009) to accommodate neural network training.

2.3 Model Initialization

It was previously shown that the heuristic KM initialization method by Klein and Manning (2004) does not work well for lexicalized grammar induction (Headden III et al., 2009; Pate and Johnson, 2016) and it is very helpful to initialize learning with a model learned by a different grammar induction method (Le and Zuidema, 2015; Jiang et al., 2016). We tested both KM initialization and the following initialization method: we first learn an unlexicalized DMV using the grammar induction method of Naseem et al. (2010) and use it to parse the training corpus; then, from the parse trees we run maximum likelihood estimation to produce the initial lexicalized model.

3 Experimental Setup

For English, we used the BLLIP corpus[1] in addition to the regular WSJ corpus in our experiments. Note that the BLLIP corpus is collected from the same news article source as the WSJ corpus, so it is in-domain and is ideal for training grammars to be evaluated on the WSJ test set. In order to solve the compatibility issue as well as improve the POS tagging accuracy, we used the Stanford tagger (Toutanova et al., 2003) to retag the BLLIP corpus and selected the sentences for which the new tags are consistent with the original tags, which resulted in 182244 sentences with length less than or equal to 10 after removing punctuations. We used this subset of BLLIP and section 2-21 of WSJ10 for training, section 22 of WSJ for validation and section 23 of WSJ for testing. We used training sets of four different sizes: WSJ10 only (5779 sentences) and 20k, 50k, and all sentences from the BLLIP subset. For Chinese, we obtained 4762 sentences for training from Chinese Treebank 6.0 (CTB) after converting data to dependency structures via Penn2Malt (Nivre, 2006) and then stripping off punctuations. We used the recommended validation and test data split described in the documentation.

We trained the models with different degrees of lexicalization. We control the degree of lexicalization by replacing words that appear less than a cutoff number in the WSJ10 or CTB corpus with their POS tags. For each degree of lexicalization, we tuned the dimension of the hidden layer of the neural network on the validation dataset. For English, we tested nine word cutoff numbers: 100000, 500, 200, 100, 80, 70, 60, 50, and 40, which resulted in vocabulary sizes of 35, 63, 98, 166, 203, 226, 267, 306, and 390 respectively; for Chinese, the word cutoff numbers are 100000, 100, 70, 50, 40, 30, 20, 12, and 10. Ideally, with higher degrees of lexicalization, the hidden layer dimension should be larger in order to accommodate the increased number of tokens. For the neural network of L-NDMV, we initialized the word and tag vectors in the neural network by learning a CBOW model using the Gensim package (Řehůřek and Sojka, 2010). We set the dimension of input and output word vectors to 100 and the dimension of input and output tag vectors to 20. We trained the neural network with learning rate 0.03, mini-batch size 200 and momentum 0.9. Because some of the neural network weights are randomly initialized, the model converges to a different local minimum in each run of the learning algorithm. Therefore, for each setup we ran our learning algorithm for three times and reported the average accuracy. More detail of the experimental setup can be found in the supplementary material.

4 Experimental Results

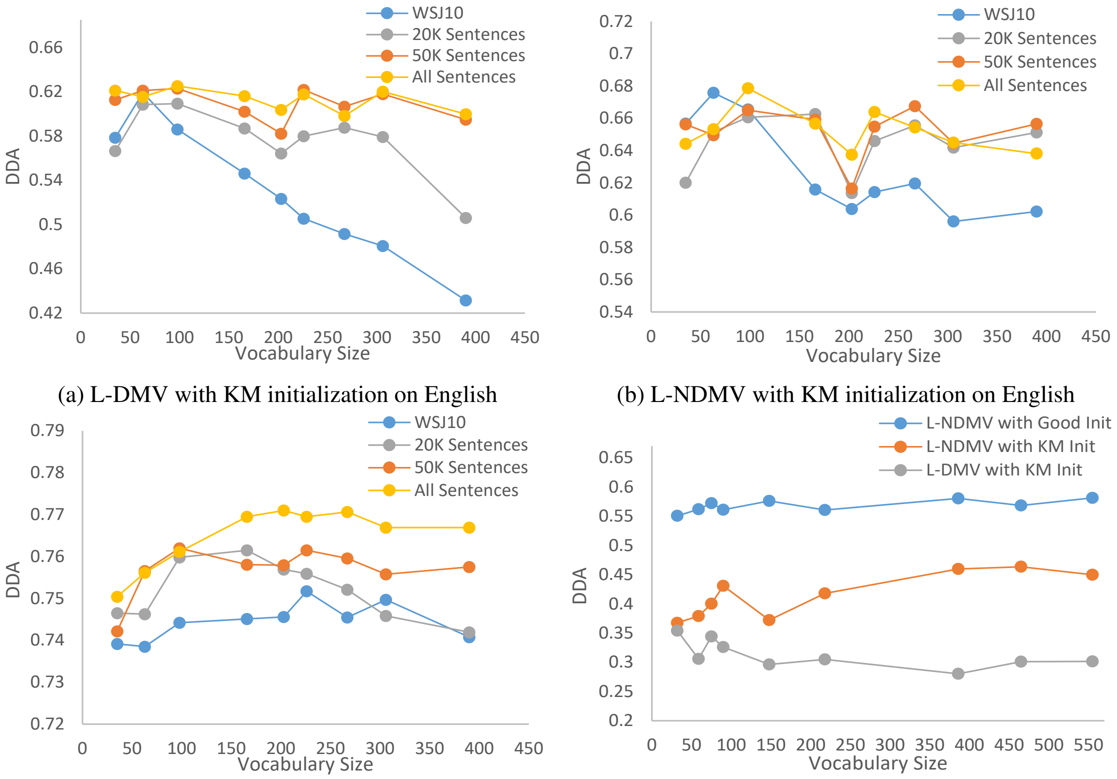
4.1 Results on English

Figure 2(a) shows the directed dependency accuracy (DDA) of the learned lexicalized DMV with KM initialization. It can be seen that on the smallest WSJ10 training corpus, lexicalization improves learning only when the degree of lexicalization is small; with further lexicalization, the learning accuracy significantly degrades. On the three larger training corpora, the impact of lexicalization on the learning accuracy is still negative but is less severe. Overall, lexicalization seems to be very data demanding and even our largest training corpora could not bring about the benefit of lexicalization. Increasing the training corpus size is helpful regardless of the degree of lexicalization, but the learning accuracies with the 50K dataset are almost identical to those with the full dataset, suggesting diminishing return of more data.

Figure 2(b) shows the results of L-NDMV with KM initialization. The parsing accuracy is improved under all the settings, showing the advantage of NDMV. The range of lexicalization degrees that improve learning becomes larger, and the degradation in accuracy with large degrees of lexicalization becomes much less severe. Diminishing return of big data as seen in the first figure can still be observed.

Figure 2(c) shows the results of L-NDMV with the initialization method described in section 2.3. It can be seen that lexicalization becomes less data demanding and the learning accuracy does not decrease until the highest degrees of lexicalization. Larger training corpora now lead to significantly better learning accuracy and support lexicalization

[1] Brown Laboratory for Linguistic Information Processing (BLLIP) 1987-89 WSJ Corpus Release 1



(a) L-DMV with KM initialization on English

(b) L-NDMV with KM initialization on English

(c) L-NDMV with good initialization on English

(d) L-DMV and L-NDMV on Chinese

Figure 2: The impact of the training corpus size and the degree of lexicalization on L-DMV and L-NDMV with different initialization methods on English and Chinese.

of greater degrees than smaller corpora. Diminishing return of big data is no longer observed, which implies further increase in accuracy with even more data.

Table 1 compares the result of L-NDMV (with the largest corpus and the vocabulary size of 203 which was selected on the validation set) with previous approaches to dependency grammar induction. It can be seen that L-NDMV is competitive with previous state-of-the-art approaches. We did some further analysis of the learned word vectors in L-NDMV in the supplementary material.

4.2 Results on Chinese

Figure 2(d) shows the results of the three approaches on the Chinese treebank. Because the corpus is relatively small, we did not study the impact of the corpus size. Similar to the case of English, the accuracy of lexicalized DMV degrades with more lexicalization. However, the accuracy with L-NDMV increases significantly with more lexicalization even without good model initialization. Adding good initialization further boosts the performance of L-NDMV, but the benefit of lexicalization is less significant (from 0.55 to 0.58).

Methods	**WSJ10**	**WSJ**
Unlexicalized Approaches, with WSJ10		
EVG (Headden III et al., 2009)	65.0	-
TSG-DMV (Blunsom and Cohn, 2010)	65.9	53.1
PR-S (Gillenwater et al., 2010)	64.3	53.3
HDP-DEP (Naseem et al., 2010)	73.8	-
UR-A E-DMV (Tu and Honavar, 2012)	71.4	57.0
Neural E-DMV(Jiang et al., 2016)	72.5	57.6
Systems Using Lexical Information and/or More Data		
LexTSG-DMV (Blunsom and Cohn, 2010)	67.7	55.7
L-EVG (Headden III et al., 2009)	68.8	-
CS (Spitkovsky et al., 2013)	72.0	64.4
MaxEnc (Le and Zuidema, 2015)	73.2	**65.8**
L-NDMV + WSJ	75.1	59.5
L-NDMV + Large Corpus	**77.2**	63.2

Table 1: Comparison of recent grammar induction systems.

5 Effect of Grammar Rule Probability Initialization

We compare four initialization methods to L-NDMV: uniform initialization, random initialization, KM initialization (Klein and Manning, 2004), and good initialization as described in section 2.3 in Figure 3. Here we trained the L-NDMV model on the WSJ10 corpus with the same experimental setup as in section 3.

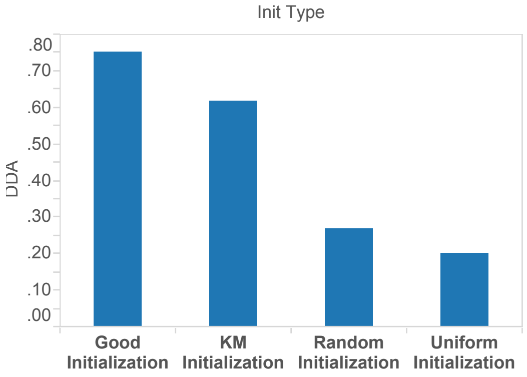


Figure 3: Comparison of four initialization methods to L-NDMV: uniform initialization, random initialization, KM initialization and good initialization.

Again, we find that good initialization leads to better performance than KM initialization, and both good initialization and KM initialization are significantly better than random and uniform initialization. Note that our results are different from those by Pate and Johnson (2016), who found that uniform initialization leads to similar performance to KM initialization. We speculate that it is because of the difference in the learning approaches (we use neural networks which may be more sensitive to initialization) and the training and test corpora (we use news articles while they use telephone scripts).

6 Conclusion and Future Work

We study the impact of the degree of lexicalization and the training data size on the accuracy of dependency grammar induction. We experimented with lexicalized DMV (L-DMV) and our lexicalized extension of Neural DMV (L-NDMV). We find that L-DMV only benefits from very small degrees of lexicalization and moderate sizes of training corpora. In contrast, L-NDMV can benefit from big training data and lexicalization of greater degrees, especially when enhanced with good model initialization, and it achieves a result that is competitive with the state-of-the-art.

In the future, we plan to study higher degrees of lexicalization or full lexicalization, as well as even larger training corpora (such as the Wikipedia corpus). We would also like to experiment with other grammar induction approaches with lexicalization and big training data.

References

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590. Association for Computational Linguistics.

Phil Blunsom and Trevor Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1204–1213. Association for Computational Linguistics.

Shay B Cohen, Kevin Gimpel, and Noah A Smith. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In *Advances in Neural Information Processing Systems*, pages 321–328.

Jennifer Gillenwater, Kuzman Ganchev, Joao Graça, Fernando Pereira, and Ben Taskar. 2010. Sparsity in dependency grammar induction. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 194–199. Association for Computational Linguistics.

William P Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109. Association for Computational Linguistics.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 763–771, Austin, Texas. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Phong Le and Willem Zuidema. 2015. Unsupervised dependency parsing: Let's use supervised parsers. *arXiv preprint arXiv:1504.04666*.

Percy Liang and Dan Klein. 2009. Online em for unsupervised models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 611–619, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Marecek and Milan Straka. 2013. Stop-probability estimates computed on a large corpus improve unsupervised dependency parsing. In *ACL (1)*, pages 281–290.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244. Association for Computational Linguistics.

Joakim Nivre. 2006. *Inductive dependency parsing*. Springer.

John K Pate and Mark Johnson. 2016. Grammar induction from (lots of) words alone.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2013. Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *EMNLP*, pages 1983–1995.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Kewei Tu and Vasant Honavar. 2012. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1324–1334. Association for Computational Linguistics.

Combining Generative and Discriminative Approaches to Unsupervised Dependency Parsing via Dual Decomposition[*]

Yong Jiang, Wenjuan Han and **Kewei Tu**
{jiangyong,hanwj,tukw}@shanghaitech.edu.cn
School of Information Science and Technology
ShanghaiTech University, Shanghai, China

Abstract

Unsupervised dependency parsing aims to learn a dependency parser from unannotated sentences. Existing work focuses on either learning generative models using the expectation-maximization algorithm and its variants, or learning discriminative models using the discriminative clustering algorithm. In this paper, we propose a new learning strategy that learns a generative model and a discriminative model jointly based on the dual decomposition method. Our method is simple and general, yet effective to capture the advantages of both models and improve their learning results. We tested our method on the UD treebank and achieved a state-of-the-art performance on thirty languages.

1 Introduction

Dependency parsing is an important task in natural language processing. It identifies dependencies between words in a sentence, which have been shown to benefit other tasks such as semantic role labeling (Lei et al., 2015) and sentence classification (Ma et al., 2015). Supervised learning of a dependency parser requires annotation of a training corpus by linguistic experts, which can be time and resource consuming. Unsupervised dependency parsing eliminates the need for dependency annotation by directly learning from unparsed text.

Previous work on unsupervised dependency parsing mainly focuses on learning generative models, such as the dependency model with valence (DMV) (Klein and Manning, 2004) and combinatory categorial grammars (CCG) (Bisk and Hockenmaier, 2012). Generative models have many advantages. For example, the learning objective function can be defined as the marginal likelihood of the training data, which is typically easy to compute in a generative model. In addition, many types of inductive bias, such as those favoring short dependency arcs (Smith and Eisner, 2006), encouraging correlations between POS tags (Cohen et al., 2008; Cohen and Smith, 2009; Berg-Kirkpatrick et al., 2010; Jiang et al., 2016), and limiting center embedding (Noji et al., 2016), can be incorporated into generative models to achieve better parsing accuracy. However, due to the strong independence assumption in most generative models, it is difficult for these models to utilize context information that has been shown to benefit supervised parsing.

Recently, a feature-rich discriminative model for unsupervised parsing is proposed that captures the global context information of sentences (Grave and Elhadad, 2015). Inspired by discriminative clustering, learning of the model is formulated as convex optimization of both the model parameters and the parses of training sentences. By utilizing language-independent rules between pairs of POS tags to guide learning, the model achieves state-of-the-art performance on the UD treebank dataset.

In this paper we propose to jointly train two state-of-the-art models of unsupervised dependency parsing: a generative model called LC-DMV (Noji et al., 2016) and a discriminative model called Convex-MST (Grave and Elhadad, 2015). We employ a learning algorithm based on the dual decomposition (Dantzig and Wolfe, 1960) inference algorithm, which encourages the two models to influence each other during training.

We evaluated our method on thirty languages and found that the jointly trained models surpass their separately trained counterparts in parsing accuracy. Further analysis shows that the two models positively influence each other during joint train-

[*]This work was supported by the National Natural Science Foundation of China (61503248).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1689–1694
Copenhagen, Denmark, September 7–11, 2017.

ing by implicitly sharing the inductive bias.

2 Preliminaries

2.1 DMV

The dependency model with valence (DMV) (Klein and Manning, 2004) is the first generative model that outperforms the left-branching baseline in unsupervised dependency parsing. In DMV, a sentence is generated by recursively applying three types of grammar rules to construct a parse tree from the top down. The probability of the generated sentence and parse tree is the probability product of all the rules used in the generation process. To learn the parameters (rule probabilities) of DMV, the expectation maximization algorithm is often used. Noji et al. (2016) exploited two universal syntactic biases in learning DMV: restricting the center-embedding depth and encouraging short dependencies. They achieved a comparable performance with state-of-the-art approaches.

2.2 Convex-MST

Convex-MST (Grave and Elhadad, 2015) is a discriminative model for unsupervised dependency parsing based on the first-order maximum spanning tree dependency parser (McDonald et al., 2005). Given a sentence, whether each possible dependency exists or not is predicted based on a set of handcrafted features and a valid parse tree closest to the prediction is identified by the minimum spanning tree algorithm.

For each sentence $\mathbf{x}$, a first-order dependency graph is built over the words of the sentence. The weight of each edge is calculated by $\mathbf{w}^T\mathbf{f}(\mathbf{x}, i, j)$, where $\mathbf{w}$ is the parameters and $\mathbf{f}(\mathbf{x}, i, j)$ is the handcrafted feature vector of the dependency from the i-th word to the j-th word in sentence $\mathbf{x}$. For sentence $\mathbf{x}$ of length n, we can represent it as matrix $\mathbf{X}$ where each raw is a feature vector. The parse tree $\mathbf{y}$ is a spanning tree of the graph and can be represented as a binary vector with length $n \times n$ where each element is 1 if the corresponding arc is in the tree and 0 otherwise.

Learning is based on discriminative clustering with the following objective function:

$$\frac{1}{N}\sum_{\alpha=1}^{N}\left(\frac{1}{2n_\alpha}||\mathbf{y}_\alpha - \mathbf{X}_\alpha\mathbf{w}||_2^2 - \mu\mathbf{v}^T\mathbf{y}_\alpha\right)+\frac{\lambda}{2}||\mathbf{w}||_2^2$$

where $\mathbf{X}_\alpha$ is a matrix where each row is a feature representation $\mathbf{f}(\mathbf{x}_\alpha, i, j)$ of an edge in the dependency graph of sentence $\mathbf{x}_\alpha$, $\mathbf{v}$ represents whether each dependency arc in $\mathbf{y}_\alpha$ satisfies a set of prespecified linguistic rules, and λ and μ are hyperparameters. The Frank-Wolfe algorithm is employed to optimize the objective function.

2.3 Dual Decomposition

Dual decomposition (Dantzig and Wolfe, 1960), a special case of Lagrangian relaxation, is an optimization method that decomposes a hard problem into several small sub-problems. It has been widely used in machine learning (Komodakis et al., 2007) and natural language processing (Koo et al., 2010; Rush and Collins, 2012).

Komodakis et al. (2007) proposed using dual decomposition to do MAP inference for Markov random fields. Koo et al. (2010) proposed a new dependency parser based on dual decomposition by combining a graph based dependency model and a non-projective head automata. In the work of Rush et al. (2010), they showed that dual decomposition can effectively integrate two lexicalized parsing models or two correlated tasks.

2.4 Agreement based Learning

Liang et al. (2008) proposed agreement based learning that trains several tractable generative models jointly and encourages them to agree on certain latent variables. To effectively train the system, a product EM algorithm was used. They showed that the joint model can perform better than each independent model on the accuracy or convergence speed. They also showed that the objective function of the work of Klein and Manning (2004) is a special case of the product EM algorithm for grammar induction. Our approach has a similar motivation to agreement based learning but has two important advantages. First, while their approach only combines generative models, our approach can make use of both generative and discriminative models. Second, while their approach requires the sub-models to share the same dynamic programming structure when performing decoding, our approach does not have such restriction.

3 Joint Training

We minimize the following objective function that combines two different models of unsupervised dependency parsing:

$$J(\mathbf{M_F}, \mathbf{M_G}) = \sum_{\alpha=1}^{N} \min_{\mathbf{y}_\alpha \in \mathcal{Y}_\alpha} \left(F(\mathbf{x}_\alpha, \mathbf{y}_\alpha; \mathbf{M_F}) + G(\mathbf{x}_\alpha, \mathbf{y}_\alpha; \mathbf{M_G})\right)$$

where N is the size of training data, $\mathbf{M_F}$ and $\mathbf{M_G}$ are the parameters of the first and second model respectively, F and G are their respective learning objectives, and $\mathcal{Y}_\alpha$ is the set of valid dependency parses of sentence $\mathbf{x}_\alpha$. While in principle this objective can be used to combine many different types of models, here we consider two state-of-the-art models of unsupervised dependency parsing, a generative model LC-DMV (Noji et al., 2016) and a discriminative model Convex-MST (Grave and Elhadad, 2015). We denote the parameters of LC-DMV by Θ and the parameters of Convex-MST by $\mathbf{w}$. Their respective objective functions are,

$$F(\mathbf{x}_\alpha, \mathbf{y}_\alpha; \Theta) = -\log\left(P_\Theta(\mathbf{x}_\alpha, \mathbf{y}_\alpha) f(\mathbf{x}_\alpha, \mathbf{y}_\alpha)\right)$$

$$G(\mathbf{x}_\alpha, \mathbf{y}_\alpha; \mathbf{w}) = \frac{1}{2n_\alpha}||\mathbf{y}_\alpha - \mathbf{X}_\alpha \mathbf{w}||_2^2 + \frac{\lambda}{2N}||\mathbf{w}||_2^2 - \mu \mathbf{v}^T \mathbf{y}$$

where $P_\Theta(\mathbf{x}_\alpha, \mathbf{y}_\alpha)$ is the joint probability of sentence $\mathbf{x}_\alpha$ and parse $\mathbf{y}_\alpha$, f is a constraint factor, and the notations in the second objective function are explained in section 2.2.

3.1 Learning

We use coordinate descent to optimize the parameters of the two models. In each iteration, we first fix the parameters and find the best dependency parses of the training sentences (see section 3.2); we then fix the parses and optimize the parameters. The detailed algorithm is shown in Algorithm 1.

Pretraining of the two models is done by running their original learning algorithms separately. When the parses of the training sentences are fixed, it is easy to show that the parameters of the two models can be optimized separately. Updating the parameters Θ of LC-DMV can be done by simply counting the number of times each rule is used in the parse trees and then normalizing the counts to get the maximum-likelihood probabilities. The parameters $\mathbf{w}$ of Convex-MST can be updated by stochastic gradient descent. After updating Θ and $\mathbf{w}$ at each iteration, we additionally train each model separately for three iterations, which we find further improves learning.

Algorithm 1 Parameter Learning

```
Input: Training sentence $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$
Pre-train $\Theta$ and $\mathbf{w}$
repeat
    Fix $\Theta$ and $\mathbf{w}$ and solve the decoding problem
        to get $\mathbf{y}_\alpha, \alpha = 1, 2, \ldots, N$
    Fix the parses and update $\Theta$ and $\mathbf{w}$
until Convergence
```

Algorithm 2 Decoding via Dual Decomposition

```
Input: Sentence $\mathbf{x}$, fixed parameters $\mathbf{w}$ and $\Theta$
Initialize vector $\mathbf{u}$ of size $n \times n$ to $\mathbf{0}$
repeat
    $\hat{\mathbf{y}} = \arg\min_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}, \mathbf{y}; \Theta) + \mathbf{u}^T \mathbf{y}$
    $\hat{\mathbf{z}} = \arg\min_{\mathbf{z} \in \mathcal{Y}} G(\mathbf{x}, \mathbf{z}; \mathbf{w}) - \mathbf{u}^T \mathbf{z}$
    if $\hat{\mathbf{y}} = \hat{\mathbf{z}}$ then
        return $\hat{\mathbf{y}}$
    else
        $\mathbf{u} = \mathbf{u} - \tau(\hat{\mathbf{y}} - \hat{\mathbf{z}})$
    end if
until Convergence
```

3.2 Joint Decoding

Given a training sample $\mathbf{x}$ and parameters $\mathbf{w}, \Theta$, the goal of decoding is to find the best parse tree:

$$\hat{\mathbf{y}} = \arg\min_{\mathbf{y} \in \mathcal{Y}} \frac{1}{2n}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 - \mu \mathbf{v}^T \mathbf{y} - \log P_\Theta(\mathbf{x}, \mathbf{y})$$

We employ the dual decomposition algorithm to solve this problem (shown in Algorithm 2), where τ represents the step size.

The most important part of the algorithm is solving the two separate decoding problems:

$$\hat{\mathbf{y}} = \arg\min_{\mathbf{y} \in \mathcal{Y}} -\log(P_\Theta(\mathbf{x}, \mathbf{y}) f(\mathbf{x}, \mathbf{y})) + \mathbf{u}^T \mathbf{y}$$

$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z} \in \mathcal{Y}} \frac{1}{2n}||\mathbf{z} - \mathbf{X}\mathbf{w}||_2^2 - \mu \mathbf{v}^T \mathbf{z} - \mathbf{u}^T \mathbf{z}$$

The first decoding problem can be solved by a modified CYK parsing algorithm that takes into account the information in vector $\mathbf{u}$. The second decoding problem can be solved using the same algorithm of Grave and Elhadad (2015) (we use the projective version in our approach).

4 Experiments

4.1 Setup

We use UD Treebank 1.4 as our datasets. We sorted the datasets in the treebank by the number of training sentences of length $\leq$ 15 and selected the top thirty datasets, which is similar to the setup of Noji et al. (2016). For each dataset, we trained our method on the training data with length $\leq$ 15 and tested our method on the testing data with length $\leq$ 40. We tuned the hyper-parameters of our method on the dataset of the English language and reported the results on the thirty datasets without any further parameter tuning. We compared our method with four baselines. The first two baselines are Convex-MST and LC-DMV that are independently trained. To construct the third baseline, we used the independently trained Convex-MST baseline to parse all the training sentences and then used the parses to initialize the training of LC-DMV. This can be seen as a simple method to combine two different approaches. On the other hand, we did not use the LC-DMV baseline to initialize Convex-MST training because the objective function of Convex-MST is convex and therefore the initialization does not matter.

4.2 Results

In Table 1, we compare our jointly trained models with the four baselines. We can see that with joint training and independent decoding, LC-DMV and Convex-MST can achieve superior overall performance than when they are separately trained with or without mutual initialization. Joint decoding with our jointly trained models performs worse than independent decoding. We made the same observation when applying joint decoding to the separately trained models (not shown in the table). We believe this is because unsupervised parsers have relatively low accuracy and forcing them to reconcile would not lead to better parses. On the other hand, joint decoding during training helps propagate useful inductive biases between models and thus leads to better trained models.

4.3 Analysis of Parsing Results

We analyze the parsing results from the two models to see how they benefit each other with joint training. Note that LC-DMV limits the depth of center embedding and encourages shorter dependency length, while Convex-MST encourages dependencies satisfying pre-specified linguistic rules. Therefore, we would like to see whether the jointly-trained LC-DMV produces more dependencies satisfying the linguistic priors than its separately-trained counterpart, and whether the jointly-trained Convex-MST produces parse trees with less center embedding and shorter dependencies than its separately-trained counterpart.

Language	M	D	D-I	M-J	D-J	DD
A_Greek	43.4	33.1	38.8	44.2	**44.9**	38.9
A_Greek-P	50.4	43.0	44.7	50.8	**52.9**	44.9
Basque	50.0	45.4	54.2	52.1	**55.7**	50.2
Bulgarian	61.6	62.4	60.3	64.7	**73.8**	64.8
Czech	48.6	17.4	53.9	48.7	**54.0**	53.5
Czech-CAC	50.4	53.0	53.9	55.6	**62.3**	50.2
Dutch	45.3	34.1	**56.7**	48.2	43.5	40.7
Dutch-LS	42.4	27.0	16.4	**43.2**	41.2	36.3
English	54.0	56.0	49.8	57.3	**60.1**	53.4
Estonian	**49.4**	31.8	47.5	48.7	44.0	44.4
Finnish	**44.7**	26.9	39.0	44.2	43.5	31.2
Finnish-FTB	**49.9**	31.0	47.9	47.7	48.0	36.5
French	**62.0**	48.6	57.0	54.5	57.0	55.5
German	51.4	50.5	54.1	49.3	**55.7**	48.6
Gothic	52.7	49.9	47.3	**59.6**	56.4	58.0
Hindi	56.8	54.2	48.4	52.1	**60.0**	49.1
Italian	69.1	**71.1**	67.4	62.8	70.3	64.5
Japanese	44.8	43.8	43.8	42.8	**45.8**	41.0
Latin-ITTB	38.8	38.6	42.3	**47.0**	42.2	40.3
Latin-PROIEL	44.3	34.8	38.7	**46.8**	41.8	42.9
Norwegian	55.3	45.5	51.4	57.4	**60.8**	46.6
Old_Church_S	56.4	26.6	51.3	58.3	**58.6**	42.0
Polish	63.4	63.7	61.5	70.7	**74.2**	68.9
Portuguese	57.9	**67.2**	60.1	56.1	62.9	57.4
Portuguese-BR	59.3	63.1	62.0	65.5	**68.8**	58.3
Russian-STR	47.6	51.7	56.5	52.1	**64.4**	52.6
Slovak	57.4	59.3	51.9	61.7	**65.9**	58.7
Slovenian	54.0	49.5	56.3	65.5	**69.6**	56.1
Spanish	61.9	61.9	60.3	57.4	**68.0**	60.2
Spanish-AC	59.4	59.5	56.4	56.8	**65.2**	57.6
Average	52.7	47.2	50.3	54.2	**56.5**	49.6
Average $\leq$ 15	55.4	48.9	54.9	57.3	**60.2**	53.8

Table 1: Directed dependency accuracy on thirty datasets with test sentences of length $\leq$ 40. The last row indicates the average directed accuracy on sentences of length $\leq$ 15. M (Convex-MST) and D (LC-DMV) are the independently trained baselines. D-I is the third baseline in which the LC-DMV training is initialized by the parses produced from the trained Convex-MST model. With our jointly trained models, M-J and D-J denote separate decoding and DD denotes joint decoding.

Figure 1 shows the percentages of dependencies satisfying linguistic rules when using the separately and jointly trained LC-DMV to parse the test sentences in the English dataset. As we can see, with joint training, LC-DMV is indeed influenced by Convex-MST and produces more dependencies satisfying linguistic rules.

Table 2 shows the average dependency length when using the separately and jointly trained Convex-MST to parse the English test dataset. The dependency length can be seen to decrease with joint training, showing the influence from LC-DMV. As to center embedding depth, we find that separately trained Convext-MST already produces very few center embeddings of depth 2 or more,

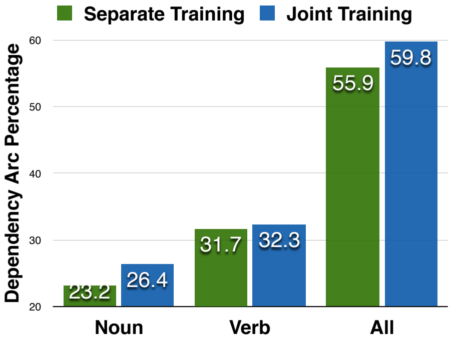


Figure 1: Percentages of dependencies satisfying linguistic rules in the LC-DMV parses of the English test dataset. Noun and Verb denote dependencies headed by nouns and verbs.

Methods	Average Dependency Length
Separate Training	1.673
Joint Training	**1.627**

Table 2: Average dependency length in the Convex-MST parses of the English test dataset.

so the influence from the center embedding constraint of LC-DMV during joint training is not obvious. We note that the influence on Convex-MST from LC-DMV during joint training is relatively small, which may contribute to the much smaller accuracy improvement (1.5%) of Convex-MST with joint training in comparison with the 9.3% improvement of LC-DMV. We conducted an additional experiment that scaled down the Convex-MST objective in joint training in order to increase the influence of LC-DMV. The results show that LC-DMV indeed influences Convex-MST to a greater degree, but the parsing accuracies of the two models decrease.

5 Conclusion

In this paper, we proposed a new learning strategy for unsupervised dependency parsing that learns a generative model and a discriminative model jointly based on dual decomposition. We show that with joint training, two state-of-the-art models can positively influence each other and achieve better performance than their separately trained counterparts.

References

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590. Association for Computational Linguistics.

Yonatan Bisk and Julia Hockenmaier. 2012. Simple robust grammar induction with combinatory categorial grammars.

Shay B Cohen, Kevin Gimpel, and Noah A Smith. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In *Advances in Neural Information Processing Systems*, pages 321–328.

Shay B Cohen and Noah A Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–82. Association for Computational Linguistics.

George B Dantzig and Philip Wolfe. 1960. Decomposition principle for linear programs. *Operations research*, 8(1):101–111.

Edouard Grave and Noémie Elhadad. 2015. A convex and feature-rich discriminative approach to dependency grammar induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1375–1384, Beijing, China. Association for Computational Linguistics.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 763–771, Austin, Texas. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 478. Association for Computational Linguistics.

Nikos Komodakis, Nikos Paragios, and Georgios Tziritas. 2007. Mrf optimization via dual decomposition: Message-passing revisited. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

Terry Koo, Alexander M Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298. Association for Computational Linguistics.

Tao Lei, Yuan Zhang, Lluís Màrquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proceedings of the 2015 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1150–1160, Denver, Colorado. Association for Computational Linguistics.

Percy S Liang, Dan Klein, and Michael I. Jordan. 2008. Agreement-based learning. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 913–920. Curran Associates, Inc.

Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. 2015. Dependency-based convolutional neural networks for sentence embedding. *arXiv preprint arXiv:1507.01839*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 91–98. Association for Computational Linguistics.

Hiroshi Noji, Yusuke Miyao, and Mark Johnson. 2016. Using left-corner parsing to encode universal structural constraints in grammar induction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 33–43, Austin, Texas. Association for Computational Linguistics.

Alexander M. Rush and Michael Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *J. Artif. Int. Res.*, 45(1):305–362.

Alexander M Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1–11. Association for Computational Linguistics.

Noah A Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 569–576. Association for Computational Linguistics.

Effective Inference for Generative Neural Parsing

Mitchell Stern **Daniel Fried** **Dan Klein**
Computer Science Division
University of California, Berkeley
{mitchell,dfried,klein}@cs.berkeley.edu

Abstract

Generative neural models have recently achieved state-of-the-art results for constituency parsing. However, without a feasible search procedure, their use has so far been limited to reranking the output of external parsers in which decoding is more tractable. We describe an alternative to the conventional action-level beam search used for discriminative neural models that enables us to decode directly in these generative models. We then show that by improving our basic candidate selection strategy and using a coarse pruning function, we can improve accuracy while exploring significantly less of the search space. Applied to the model of Choe and Charniak (2016), our inference procedure obtains 92.56 F1 on section 23 of the Penn Treebank, surpassing prior state-of-the-art results for single-model systems.

1 Introduction

A recent line of work has demonstrated the success of generative neural models for constituency parsing (Dyer et al., 2016; Choe and Charniak, 2016). As with discriminative neural parsers, these models lack a dynamic program for exact inference due to their modeling of unbounded dependencies. However, while discriminative neural parsers are able to obtain strong results using greedy search (Dyer et al., 2016) or beam search with a small beam (Vinyals et al., 2015), we find that a simple action-level approach fails outright in the generative setting. Perhaps because of this, the application of generative neural models has so far been restricted to reranking the output of external parsers.

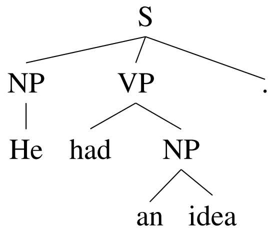
Intuitively, because a generative parser defines a joint distribution over sentences and parse trees, probability mass will be allocated unevenly between a small number of common structural actions and a large vocabulary of lexical items. This imbalance is a primary cause of failure for search procedures in which these two types of actions compete directly. A notion of equal competition among hypotheses is then desirable, an idea that has previously been explored in generative models for constituency parsing (Henderson, 2003) and dependency parsing (Titov and Henderson, 2010; Buys and Blunsom, 2015), among other tasks. We describe a related state-augmented beam search for neural generative constituency parsers in which lexical actions compete only with each other rather than with structural actions. Applying this inference procedure to the generative model of Choe and Charniak (2016), we find that it yields a self-contained generative parser that achieves high performance.

Beyond this, we propose an enhanced candidate selection strategy that yields significant improvements for all beam sizes. Additionally, motivated by the look-ahead heuristic used in the top-down parsers of Roark (2001) and Charniak (2010), we also experiment with a simple coarse pruning function that allows us to reduce the number of states expanded per candidate by several times without compromising accuracy. Using our final search procedure, we surpass prior state-of-the-art results among single-model parsers on the Penn Treebank, obtaining an F1 score of 92.56.

2 Common Framework

The generative neural parsers of Dyer et al. (2016) and Choe and Charniak (2016) can be unified under a common shift-reduce framework. Both systems build parse trees in left-to-right depth-first order by executing a sequence of actions, as illustrated in Figure 1. These actions can be grouped

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1695–1700
Copenhagen, Denmark, September 7–11, 2017.



(S (NP He NP) (VP had (NP an idea NP) VP) . S)

Figure 1: A parse tree and the action sequence that produced it, corresponding to the sentence "He had an idea." The tree is constructed in left-to-right depth-first order. The tree contains only nonterminals and words; part-of-speech tags are not included. OPEN(X) and CLOSE(X) are rendered as "(X" and "X)" for brevity.

into three major types: OPEN(X) and CLOSE(X), which open and close a constituent with nonterminal X,[1] respectively, and SHIFT(x), which adds the word x to the current constituent. The probability of an action sequence $(a_1, \ldots, a_T)$ is

$$\begin{aligned} P(a_1, \ldots, a_T) &= \prod_{t=1}^{T} P(a_t \mid a_1, \ldots, a_{t-1}) \\ &= \prod_{t=1}^{T} [\text{softmax}(\mathbf{W}\mathbf{u}_t + \mathbf{b})]_{a_t}, \end{aligned}$$

where $\mathbf{u}_t$ is a continuous representation of the parser's state at time t, and $[\mathbf{v}]_j$ denotes the jth component of a vector $\mathbf{v}$. We refer readers to the respective authors' papers for the parameterization of $\mathbf{u}_t$ in each model.

In both cases, the decoding process reduces to a search for the most probable action sequence that represents a valid tree over the input sentence. For a given hypothesis, this requirement implies several constraints on the successor set (Dyer et al., 2016); e.g., SHIFT(x) can only be executed if the next word in the sentence is x, and CLOSE(X) cannot be executed directly after OPEN(X).

3 Model and Training Setup

We reimplemented the generative model described in Choe and Charniak (2016) and trained it on the Penn Treebank (Marcus et al., 1993) using their published hyperparameters and preprocessing. However, rather than selecting the final model based on reranking performance, we instead perform early stopping based on development set perplexity. We use sections 2-21 of the Penn Treebank for training, section 22 for development, and section 23 for testing. The model's action space consists of 26 matching pairs of OPEN and CLOSE actions, one for each nonterminal, and 6,870 SHIFT actions, one for each preprocessed word type. While we use this particular model for our experiments, we note that our subsequent discussion of inference techniques is equally applicable to any generative parser that adheres to the framework described above in Section 2.

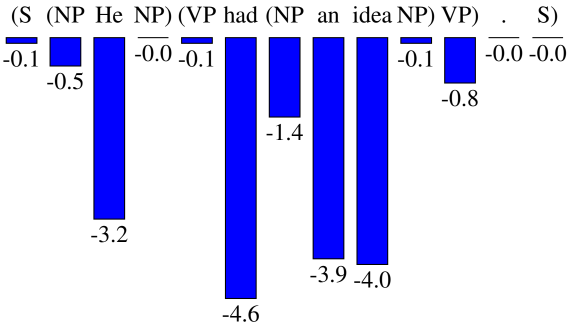


Figure 2: A plot of the action log probabilities $\log P(a_t \mid a_1, \ldots, a_{t-1})$ for the example in Figure 1 under our main model. We observe that OPEN and CLOSE actions have much higher probability than SHIFT actions. This imbalance is responsible for the failure of standard action-level beam search.

4 Action-Level Search

Given that ordinary action-level search has been applied successfully to discriminative neural parsers (Vinyals et al., 2015; Dyer et al., 2016), it offers a sensible starting point for decoding in generative models. However, even for large beam sizes, the following pathological behavior is encountered for generative decoding, preventing reasonable parses from being found. Regardless of the sequence of actions taken so far, the generative model tends to assign much higher probabilities to structural OPEN and CLOSE actions than it does lexical SHIFT actions, as shown in Figure 2. The model therefore prefers to continually open new constituents until a hard limit is reached, as the alternative at each step is to take the low-probability action of shifting the next word. The resulting

[1] The model described in Dyer et al. (2016) has only a single CLOSE action, whereas the model described in Choe and Charniak (2016) annotates CLOSE(X) actions with their nonterminals. We present the more general version here.

Beam Size k	200	400	600	800	1000	2000
$k_w = k$	87.47	89.86	90.98	91.62	91.97	92.74
$k_w = k/10$	89.25	91.16	91.83	92.12	92.38	92.93

Table 1: Development F1 scores using word-level search with various beam sizes k and two choices of word beam size k_w.

sequence typically has much lower overall probability than a plausible parse, but the model's myopic comparison between structural and lexical actions prevents reasonable candidates from staying on the beam. Action-level beam search with beam size 1000 obtains an F1 score of just 52.97 on the development set.

5 Word-Level Search

The imbalance between the probabilities of structural and lexical actions suggests that the two kinds of actions should not compete against each other within a beam. This leads us to consider an augmented state space in which they are kept separate by design, as was done by Fried et al. (2017). In conventional action-level beam search, hypotheses are grouped by the length of their action history $|A|$. Letting A_i denote the set of actions taken since the ith shift action, we instead group hypotheses by the pair $(i, |A_i|)$, where i ranges between 0 and the length of the sentence.

Let k denote the target beam size. The search process begins with the empty hypothesis in the $(0, 0)$ bucket. Word-level steps are then taken according to the following procedure for $i = 0, 1, \ldots,$ up to the length of the sentence (inclusive). Beginning with the $(i, 0)$ bucket, the successors of each hypothesis are pooled together, sorted by score, and filtered down to the top k. Of those that remain, successors obtained by taking an OPEN or CLOSE action advance to the $(i, 1)$ bucket, whereas successors obtained from a SHIFT action are placed in the $(i + 1, 0)$ bucket if i is less than the sentence length, or the completed list if i is equal to the sentence length. This process is repeated for the $(i, 1)$ bucket, the $(i, 2)$ bucket, and so forth, until the $(i+1, 0)$ bucket contains at least k hypotheses. If desired, a separate word beam size $k_w < k$ can be used at word boundaries, in which case each word-level step terminates when the $(i + 1, 0)$ bucket has k_w candidates instead of k. This introduces a bottleneck that can help to promote beam diversity.

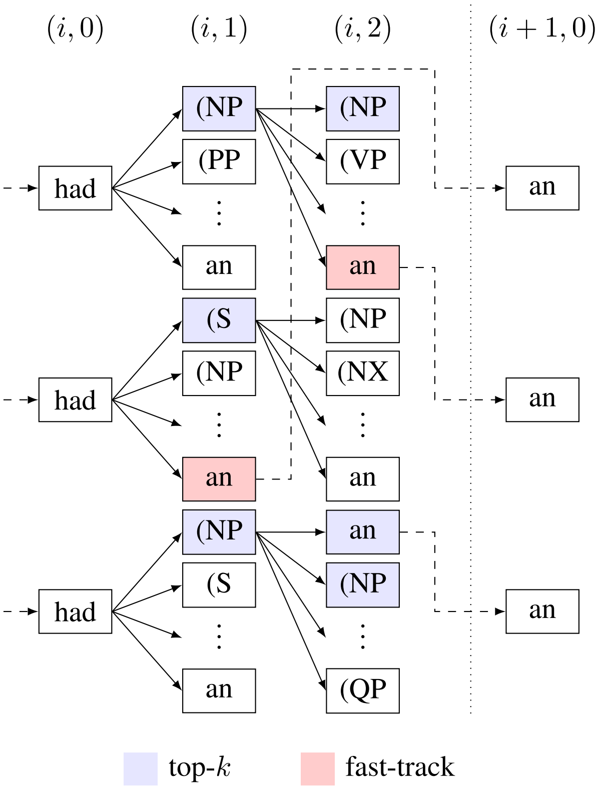


Figure 3: One step of word-level search with fast-track candidate selection (Sections 5 and 6) for the example in Figure 1. Grouping candidates by the current word i ensures that low-probability lexical actions are kept separate from high-probability structural actions at the beam level. Fast-track selection mitigates competition between the two types of actions within a single pool of successors.

Development set results for word-level search with a variety of beam sizes and with $k_w = k$ or $k_w = k/10$ are given in Table 1. We observe that performance in both cases increases steadily with beam size. Word-level search with $k_w = k/10$ consistently outperforms search without a bottleneck at all beam sizes, indicating the utility of this simple diversity-inducing modification. The top result of 92.93 F1 is already quite strong compared to other single-model systems.

6 Fast-Track Candidate Selection

The word-level beam search described in Section 5 goes one step toward ameliorating the issue that causes action-level beam search to fail, namely the direct competition between common structural actions with high probabilities and low-frequency shift actions with low probabilities. However, the issue is still present to some extent, in that successors of both types from a given bucket are pooled

Beam Size k	200	400	600	800	1000	2000
$k_w = k$	91.33	92.17	92.51	92.73	92.89	93.05
$k_w = k/10$	91.41	92.34	92.70	92.94	93.09	93.18

Table 2: Development F1 scores using the settings from Table 1, together with the fast-track selection strategy from Section 6 with $k_s = k/100$.

together and filtered down as a single collection before being routed to their respective destinations. We therefore propose a more direct solution to the problem, in which a small number $k_s \ll k$ of SHIFT successors are fast-tracked to the next word-level bucket *before* any filtering takes place. These fast-tracked candidates completely bypass competition with potentially high-scoring OPEN or CLOSE successors, allowing for higher-quality results in practice with minimal overhead. See Figure 3 for an illustration.

We repeat the experiments from Section 5 with $k_s = k/100$ and report the results in Table 2. Note that the use of fast-tracked candidates offers significant gains under all settings. The top result improves from 92.93 to 93.18 with the use of fast-tracked candidates, surpassing prior single-model systems on the development set.

7 OPEN Action Pruning

At any point during the trajectory of a hypothesis, either 0 or all 26 of the OPEN actions will be available, compared with at most 1 CLOSE action and at most 1 SHIFT action. Hence, when available, OPEN actions comprise most or all of a candidate's successor actions. To help cut down on this portion of the search space, it is natural to consider whether some of these actions could be ruled out using a coarse model for pruning.

7.1 Coarse Model

We consider a class of simple pruning models that condition on the $c \geq 0$ most recent actions and the next word in the sentence, and predict a probability distribution over the next action. In the interest of efficiency, we collapse all SHIFT actions into a single unlexicalized SHIFT action, significantly reducing the size of the output vocabulary.

The input $\mathbf{v}_t$ to the pruning model at time t is the concatenation of a vector embedding for each action in the context $(a_{t-c}, a_{t-c+1}, \ldots, a_{t-1})$ and a vector embedding for the next word w:

$$\mathbf{v}_t = [\mathbf{e}_{a_{t-c}}; \mathbf{e}_{a_{t-c+1}}; \ldots; \mathbf{e}_{a_{t-1}}; \mathbf{e}_w],$$

where each $\mathbf{e}_j$ is a learned vector embedding. The pruning model itself is implemented by feeding the input vector through a one-layer feedforward network with a ReLU non-linearity, then applying a softmax layer on top:

$$\begin{aligned}
&P(a_t = a \mid a_1, \ldots, a_{t-1}, \text{next-word} = w) \\
&= P(a_t = a \mid a_{t-c}, \ldots, a_{t-1}, \text{next-word} = w) \\
&= [\operatorname{softmax}(\mathbf{W}_2 \max(\mathbf{W}_1 \mathbf{v}_t + \mathbf{b}_1, 0) + \mathbf{b}_2)]_a.
\end{aligned}$$

The pruning model is trained separately from the main parsing model on gold action sequences derived from the training corpus, with log-likelihood as the objective function and a cross entropy loss.

7.2 Strategy and Empirical Lower Bound

Once equipped with a coarse model, we use it for search reduction in the following manner. As mentioned above, when a hypothesis is eligible to open a new constituent, most of its successors will be obtained through OPEN actions. Accordingly, we use the coarse model to restrict the set of OPEN actions to be explored. When evaluating the pool of successors for a given collection of hypotheses during beam search, we run the coarse model on each hypothesis to obtain a distribution over its next possible actions, and gather together all the coarse scores of the would-be OPEN successors. We then discard the OPEN successors whose coarse scores lie below the top $1-p$ quantile for a fixed $0 < p < 1$, guaranteeing that no more than a p-fraction of OPEN successors are considered for evaluation. Taking $p = 1$ corresponds to the unpruned setting.

This strategy gives us a tunable hyperparameter p that allows us to trade off between the amount of search we perform and the quality of our results. Before testing our procedure, however, we would first like to investigate whether there is a principled bound on how low we can expect to set p without a large drop in performance. A simple estimate arises from noting that the pruning fraction p should be set to a value for which most or all of the outputs encountered in the training set are retained. Otherwise, the pruning model would prevent the main model from even recreating the training data, let alone producing good parses for new sentences.

To this end, we collect training corpus statistics on the occurrences of inputs to the pruning function and their corresponding outputs. We then

c	1	2	3	4	5	6	7	8	9	10
0	20.0	58.4	82.4	91.0	94.9	96.8	97.9	98.6	98.9	99.2
1	54.9	80.5	91.1	95.9	97.7	98.8	99.5	99.8	99.9	100.0
2	61.2	85.0	93.8	97.4	98.6	99.5	99.8	99.9	100.0	100.0

Table 3: Cumulative distributions of the number of unique OPEN outputs per input for an order-c pruning function, computed over pruning inputs with at least one OPEN output.

p	6/26	7/26	8/26	9/26	10/26	11/26	1
Dev F1	92.78	93.00	93.08	93.13	93.19	93.19	93.18

Table 4: Results when the best setting from Section 6 is rerun with OPEN action pruning with context size $c = 2$ and various pruning fractions p. Lower values of p indicate more aggressive pruning, while $p = 1$ means no pruning is performed.

compute the number of unique OPEN actions associated with inputs occurring at least 20 times, and restrict our attention to inputs with at least one OPEN output. The resulting cumulative distributions for context sizes $c = 0, 1, 2$ are given in Table 3. If we require that our pruning fraction p be large enough to recreate at least 99% of the training data, then since there are 26 total nonterminals, approximate[2] lower bounds for p are $10/26 \approx 0.385$ for $c = 0$, $7/26 \approx 0.269$ for $c = 1$, and $6/26 \approx 0.231$ for $c = 2$.

7.3 Pruning Results

We reran our best experiment from Section 6 with an order-2 pruning function and pruning fractions $p = 6/26, \dots, 11/26$. The results are given in Table 4. We observe that performance is on par with the unpruned setup (at most 0.1 absolute difference in F1 score) for p as low as $8/26 \approx 0.308$. Setting p to $7/26 \approx 0.269$ results in a drop of 0.18, and setting p to $6/26 \approx 0.231$ results in a drop of 0.40. Hence, degradation begins to occur right around the empirically-motivated threshold of $6/26$ given above, but we can prune $1 - 8/26 \approx 69.2\%$ of OPEN successors with minimal changes in performance.

8 Final Results and Conclusion

We find that the best overall settings are a beam size of $k = 2000$, a word beam size of $k_w = 200$, and $k_s = 20$ fast-track candidates per step, as this setup achieves both the highest probabilities under the model and the highest development F1. We report our test results on section 23 of the Penn Treebank under these settings in Table 5 both with and without pruning, as well as a number of other recent results. We achieve F1 scores of 92.56 on the test set without pruning and 92.53 when $1 - 8/26 \approx 69.2\%$ of OPEN successors are pruned, obtaining performance well above the previous state-of-the-art scores for single-model parsers. This demonstrates that the model of Choe and Charniak (2016) works well as an accurate, self-contained system. The fact that we match the performance of their reranking parser using the same generative model confirms the efficacy of our approach. We believe that further refinements of our search procedure can continue to push the bar higher, such as the use of a learned heuristic function for forward score estimation, or a more sophisticated approximate decoding scheme making use of specific properties of the model. We look forward to exploring these directions in future work.

Parser	LR	LP	F1
Vinyals et al. (2015)	–	–	88.3
Shindo et al. (2012)	–	–	91.1
Cross and Huang (2016)	90.5	92.1	91.3
Dyer et al. (2016)	–	–	91.7
Liu and Zhang (2017)	91.3	92.1	91.7
Stern et al. (2017)	90.63	92.98	91.79
Our Best Result	92.57	92.56	92.56
Our Best Result (with pruning)	92.52	92.54	92.53
Vinyals et al. (2015) (ensemble)	–	–	90.5
Shindo et al. (2012) (ensemble)	–	–	92.4
Choe and Charniak (2016) (rerank)	–	–	92.6
Dyer et al. (2016) (rerank)	–	–	93.3
Fried et al. (2017) (ensemble, rerank)	–	–	94.25

Table 5: Comparison of F1 scores on section 23 of the Penn Treebank. Here we only include models trained without external silver training data. Results in the first two sections are for single-model systems.

Acknowledgments

MS is supported by an NSF Graduate Research Fellowship. DF is supported by an NDSEG fellowship.

References

Jan Buys and Phil Blunsom. 2015. Generative incremental dependency parsing with neural networks. In *Proceedings of the 53rd Annual Meeting of the*

[2] These thresholds are not exact due to the fact that our pruning procedure operates on collections of multiple hypotheses' successors at inference time rather than the successors of an individual hypothesis.

Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 863–869, Beijing, China. Association for Computational Linguistics.

Eugene Charniak. 2010. Top-down nearly-context-sensitive parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 674–683. Association for Computational Linguistics.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Vancouver, Canada. Association for Computational Linguistics.

James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 24–31. Association for Computational Linguistics.

Jiangming Liu and Yue Zhang. 2017. Shift-reduce constituent parsing with neural lookahead features. *Transactions of the Association for Computational Linguistics*, 5:45–58.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational linguistics*, 27(2):249–276.

Hiroyuki Shindo, Yusuke Miyao, Akinori Fujino, and Masaaki Nagata. 2012. Bayesian symbol-refined tree substitution grammars for syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–448, Jeju Island, Korea. Association for Computational Linguistics.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada. Association for Computational Linguistics.

Ivan Titov and James Henderson. 2010. A latent variable model for generative dependency parsing. In *Trends in Parsing Technology*, pages 35–55. Springer.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Semi-Supervised Structured Prediction with Neural CRF Autoencoder

Xiao Zhang[†], Yong Jiang[‡], Hao Peng[†], Kewei Tu[‡] and Dan Goldwasser[†]

[†]Department of Computer Science, Purdue University, West Lafayette, USA
{zhang923, pengh, dgoldwas}@cs.purdue.edu
[‡]School of Information Science and Technology, ShanghaiTech University, Shanghai, China
{jiangyong, tukw}@shanghaitech.edu.cn

Abstract

In this paper we propose an end-to-end neural CRF autoencoder (NCRF-AE) model for semi-supervised learning of sequential structured prediction problems. Our NCRF-AE consists of two parts: an encoder which is a CRF model enhanced by deep neural networks, and a decoder which is a generative model trying to reconstruct the input. Our model has a unified structure with different loss functions for labeled and unlabeled data with shared parameters. We developed a variation of the EM algorithm for optimizing both the encoder and the decoder simultaneously by decoupling their parameters. Our experimental results over the Part-of-Speech (POS) tagging task on eight different languages, show that the NCRF-AE model can outperform competitive systems in both supervised and semi-supervised scenarios.

1 Introduction

The recent renaissance of deep learning has led to significant strides forward in several AI fields. In Natural Language Processing (NLP), characterized by highly structured tasks, promising results were obtained by models that combine deep learning methods with traditional structured learning algorithms (Chen and Manning, 2014; Durrett and Klein, 2015; Andor et al., 2016; Wiseman and Rush, 2016). These models combine the strengths of neural models, that can score local decisions using a rich non-linear representation, with efficient inference procedures used to combine the local decisions into a coherent global decision. Among these models, neural variants of the Conditional Random Fields (CRF) model (Lafferty et al., 2001) are especially popular. By replacing the linear potentials with non-linear potential using neural networks these models were able to improve performance in several structured prediction tasks (Andor et al., 2016; Peng and Dredze, 2016; Lample et al., 2016; Ma and Hovy, 2016; Durrett and Klein, 2015).

Despite their promise, wider adoption of these algorithms for new structured prediction tasks can be difficult. Neural networks are notoriously susceptible to over-fitting unless large amounts of training data are available. This problem is exacerbated in the structured settings, as accounting for the dependencies between decisions requires even more data. Providing it through manual annotation is often a difficult labor-intensive task.

In this paper we tackle this problem, and propose an end-to-end neural CRF autoencoder (NCRF-AE) model for semi-supervised learning on sequence labeling problems.

An autoencoder is a special type of neural net, modeling the conditional probability $P(\hat{X}|X)$, where X is the original input to the model and $\hat{X}$ is the reconstructed input (Hinton and Zemel, 1994). Autoencoders consist of two parts, an *encoder* projecting the input to a hidden space, and a *decoder* reconstructing the input from it.

Traditionally, autoencoders are used for generating a compressed representation of the input by projecting it into a dense low dimensional space. In our setting the hidden space consists of discrete variables that comprise the output structure. These generalized settings are described in Figure 1a. By definition, it is easy to see that the encoder (lower half in Figure 1a) can be modeled by a discriminative model describing $P(Y|X)$ directly, while the decoder (upper half in Figure 1a) naturally fits as a generative model, describing $P(\hat{X}|Y)$, where Y is the label. In our model, illustrated in Figure 1b, the encoder is a CRF model with neural networks

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1701–1711
Copenhagen, Denmark, September 7–11, 2017.

as its potential extractors, while the decoder is a generative model, trying to reconstruct the input.

Our model carries the merit of autoencoders, which can exploit valuable information from unlabeled data. Recent works (Ammar et al., 2014; Lin et al., 2015) suggested using an autoencoder with a CRF model as an encoder in an unsupervised setting. We significantly expand on these works and suggest the following contributions:

1. We propose a unified model seamlessly accommodating both unlabeled and labeled data. While past work focused on unsupervised structured prediction, neglecting the discriminative power of such models, our model easily supports learning in both fully supervised and semi-supervised settings. We developed a variation of the Expectation-Maximization (EM) algorithm, used for optimizing the encoder and the decoder of our model simultaneously.

2. We increase the expressivity of the traditional CRF autoencoder model using neural networks as the potential extractors, thus avoiding the heavy feature engineering necessary in previous works.

Interestingly, our model's predictions, which unify the discriminative neural CRF encoder and the generative decoder, have led to an improved performance over the highly optimized neural CRF (NCRF) model alone, even when trained in the supervised settings over the same data.

3. We demonstrate the advantages of our model empirically, focusing on the well-known Part-of-Speech (POS) tagging problem over 8 different languages, including low resource languages. In the supervised setting, our NCRF-AE outperformed the highly optimized NCRF. In the semi-supervised setting, our model was able to successfully utilize unlabeled data, improving on the performance obtained when only using the labeled data, and outperforming competing semi-supervised learning algorithms.

Furthermore, our newly proposed algorithm is directly applicable to other sequential learning tasks in NLP, and can be easily adapted to other structured tasks such as dependency parsing or constituent parsing by replacing the forward-backward algorithm with the inside-outside algorithm. All of these tasks can benefit from semi-supervised learning algorithms.[1]

2 Related Work

Neural networks were successfully applied to many NLP tasks, including tagging (Ma and Hovy, 2016; Mesnil et al., 2015; Lample et al., 2016), parsing (Chen and Manning, 2014), text generation (Sutskever et al., 2011), machine translation (Bahdanau et al., 2015), sentiment analysis (Kim, 2014) and question answering (Andreas et al., 2016). Most relevant to this work are structured prediction models capturing dependencies between decisions, either by modeling the dependencies between the hidden representations of connected decisions using RNN/LSTM (Vaswani et al., 2016; Katiyar and Cardie, 2016), by explicitly modeling the structural dependencies between output predictions (Durrett and Klein, 2015; Lample et al., 2016; Andor et al., 2016), or by combining the two approaches (Socher et al., 2013; Wiseman and Rush, 2016).

In contrast to supervised latent variable models, such as the Hidden Conditional Random Fields in (Quattoni et al., 2007), which utilize additional latent variables to infer for supervised structure prediction, we do not presume any additional latent variables in our NCRF-AE model in both supervised and semi-supervised setting.

The difficulty of providing sufficient supervision has motivated work on semi-supervised and unsupervised learning for many of these tasks (McClosky et al., 2006; Spitkovsky et al., 2010; Subramanya et al., 2010; Stratos and Collins, 2015; Marinho et al., 2016; Tran et al., 2016), including several that also used autoencoders (Ammar et al., 2014; Lin et al., 2015; Miao and Blunsom, 2016; Kociský et al., 2016; Cheng et al., 2017). In this paper we expand on these works, and suggest a neural CRF autoencoder, that can leverage both labeled and unlabeled data.

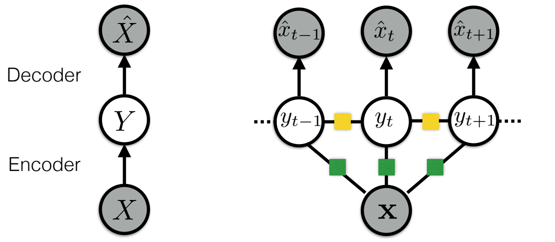
3 Neural CRF Autoencoder

In semi-supervised learning the algorithm needs to utilize both labeled and unlabeled data. Autoencoders offer a convenient way of dealing with both types of data in a unified fashion.

A generalized autoencoder (Figure 1a) tries to reconstruct the input $\hat{X}$ given the original input X, aiming to maximize the log probability $P(\hat{X}|X)$ without knowing the latent variable Y explicitly. Since we focus on sequential structured prediction problems, the encoding and decoding processes are no longer for a single data point (x, y) (x if

[1] Our code and experimental set up will be available at https://github.com/cosmozhang/NCRF-AE

unlabeled), but for the whole input instance and output sequence $(\boldsymbol{x}, \boldsymbol{y})$ ($\boldsymbol{x}$ if unlabeled). Additionally, as our main purpose in this study is to reconstruct the input with precision, $\hat{\boldsymbol{x}}$ is just a copy of $\boldsymbol{x}$.



(a) A generalized autoencoder.

(b) The neural CRF autoencoder model in this work.

Figure 1: On the left is a generalized autoencoder, of which the lower half is the encoder and the upper half is the decoder. On the right is an illustration of the graphical model of our NCRF-AE model. The yellow squares are interactive potentials among labels, and the green squares represent the unary potentials generated by the neural networks.

As shown in Figure 1b, our NCRF-AE model consists of two parts: the encoder (the lower half) is a discriminative CRF model enhanced by deep neural networks as its potential extractors with encoding parameters $\boldsymbol{\Lambda}$, describing the probability of a predicted sequence of labels given the input; the decoder (the upper half) is a generative model with reconstruction parameters $\boldsymbol{\Theta}$, modeling the probability of reconstructing the input given a sequence of labels. Accordingly, we present our model mathematically as follows:

$$\begin{aligned}P_{\boldsymbol{\Theta},\boldsymbol{\Lambda}}(\hat{\boldsymbol{x}}|\boldsymbol{x}) &= \sum_{\boldsymbol{y}} P_{\boldsymbol{\Theta},\boldsymbol{\Lambda}}(\hat{\boldsymbol{x}}, \boldsymbol{y}|\boldsymbol{x}) \\ &= \sum_{\boldsymbol{y}} P_{\boldsymbol{\Theta}}(\hat{\boldsymbol{x}}|\boldsymbol{y}) P_{\boldsymbol{\Lambda}}(\boldsymbol{y}|\boldsymbol{x}),\end{aligned}$$

where $P_{\boldsymbol{\Lambda}}(\boldsymbol{y}|\boldsymbol{x})$ is the probability given by the neural CRF encoder, and $P_{\boldsymbol{\Theta}}(\hat{\boldsymbol{x}}|\boldsymbol{y})$ is the probability produced by the generative decoder.

When making a prediction, the model tries to find the most probable output sequence by performing the following inference procedure using the Viterbi algorithm:

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} P_{\boldsymbol{\Theta},\boldsymbol{\Lambda}}(\hat{\boldsymbol{x}}, \boldsymbol{y}|\boldsymbol{x}).$$

To clarify, as we focus on POS tagging problems in this study, in the unsupervised setting where the true POS tags are unknown, the labels used for reconstruction are actually the POS tags being induced. The labels induced here are corresponding to the hidden nodes in a generalized autoencoder model.

3.1 Neural CRF Encoder

In a CRF model, the probability of predicted labels $\boldsymbol{y}$, given sequence $\boldsymbol{x}$ as input is modeled as

$$P_{\boldsymbol{\Lambda}}(\boldsymbol{y}|\boldsymbol{x}) = \frac{e^{\Phi(\boldsymbol{x},\boldsymbol{y})}}{Z},$$

where $Z = \sum_{\tilde{\boldsymbol{y}}} e^{\Phi(\boldsymbol{x},\tilde{\boldsymbol{y}})}$ is the partition function that marginalize over all possible assignments to the predicted labels of the sequence, and $\Phi(\boldsymbol{x}, \boldsymbol{y})$ is the scoring function, which is defined as:

$$\Phi(\boldsymbol{x}, \boldsymbol{y}) = \sum_t \phi(\boldsymbol{x}, y_t) + \psi(y_{t-1}, y_t).$$

The partition function Z can be computed efficiently via the forward-backward algorithm. The term $\phi(\boldsymbol{x}, y_t)$ corresponds to the score of a particular tag y_t at position t in the sequence, and $\psi(y_{t-1}, y_t)$ represents the score of transition from the tag at position $t-1$ to the tag at position t. In our NCRF-AE model, $\phi(\boldsymbol{x}, y_t)$ is described by deep neural networks while $\psi(y_{t-1}, y_t)$ by a transition matrix. Such a structure allows for the use of distributed representations of the input, for instance, the word embeddings on a continuous vector space (Mikolov et al., 2013).

Typically in our work, $\phi(\boldsymbol{x}, y_t)$ is modeled jointly by a multi-layer perceptron (MLP) that utilizes the word-level information, and a bi-directional long-short term memory (LSTM) neural network (Hochreiter and Urgen Schmidhuber, 1997) that captures the character level information within each word. A bi-directional structure can extract character level information from both directions, with which we expect to catch the prefix and suffix information of words in an end-to-end system, rather than using hand-engineered features. The bi-directional LSTM neural network consumes character embeddings $\boldsymbol{e}_c \in \mathbb{R}^{k_1}$ as input, where k_1 is the dimensionality of the character embeddings. A normal LSTM can be denoted as:

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_{ei}\boldsymbol{e}_{c_t} + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i),\\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_{ef}\boldsymbol{e}_{c_t} + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f),\\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_{eo}\boldsymbol{e}_{c_t} + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o),\\
\boldsymbol{g}_t &= Relu(\boldsymbol{W}_{ec}\boldsymbol{e}_{c_t} + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c),\\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t,\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot tanh(\boldsymbol{c}_t),
\end{aligned}$$

where $\odot$ denotes element-wise multiplication. Then a bi-directional LSTM neural network extends it as follows, by denoting the procedure of generating $\boldsymbol{h}_t$ as $\mathcal{H}$:

$$\begin{aligned}
\overrightarrow{\boldsymbol{h}}_t &= \mathcal{H}(\boldsymbol{W}_{e\overrightarrow{h}}\boldsymbol{e}_{c_t} + \boldsymbol{W}_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_{\overrightarrow{h}}),\\
\overleftarrow{\boldsymbol{h}}_t &= \mathcal{H}(\boldsymbol{W}_{e\overleftarrow{h}}\boldsymbol{e}_{c_t} + \boldsymbol{W}_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{\boldsymbol{h}}_{t-1} + \boldsymbol{b}_{\overleftarrow{h}}),
\end{aligned}$$

where $\boldsymbol{e}_{c_t}$ here is the character embedding for character c in position t in a word.

The inputs to the MLP are word embeddings $\boldsymbol{e}_v \in \mathbb{R}^{k_2}$ for each word v, where k_2 is the dimensionality of the vector, concatenated with the final representation generated by the bi-directional LSTM over the characters of that word: $\boldsymbol{u} = [\boldsymbol{e}_v; \overrightarrow{\boldsymbol{h}}_v; \overleftarrow{\boldsymbol{h}}_v]$. In order to leverage the capacity of the CRF model, we use a word and its context together to generate the unary potential. More specifically, we adopt a concatenation $\boldsymbol{v}_t = [\boldsymbol{u}_{t-(w-1)/2}; \cdots; \boldsymbol{u}_{t-1}; \boldsymbol{u}_t; \boldsymbol{u}_{t+1}; \cdots; \boldsymbol{u}_{t+(w-1)/2}]$ as the inputs to the MLP model, where t denotes the position in a sequence, and w being an odd number indicates the context size. Further, in order to enhance the generality of our model, we add a dropout layer on the input right before the MLP layer as a regularizer. Notice that different from a normal MLP, the activation function of the last layer is no more a softmax function, but a linear function generates the log-linear part $\phi_t(\boldsymbol{x}, y_t)$ of the CRF model:

$$\begin{aligned}
\boldsymbol{h}_t &= Relu(\boldsymbol{W}\boldsymbol{v}_t + \boldsymbol{b})\\
\phi_t &= \boldsymbol{w}_y^{\intercal}\boldsymbol{h}_t + b_y.
\end{aligned}$$

The transition score $\psi(y_{t-1}, y_t)$ is a single scalar representing the interactive potential. We use a transition matrix $\boldsymbol{\Psi}$ to cover all the transitions between different labels, and $\boldsymbol{\Psi}$ is part of the encoder parameters $\boldsymbol{\Lambda}$.

All the parameters in the neuralized encoder are updated when the loss function is minimized via error back-propagation through all the structures of the neural networks and the transition matrix.

The detailed structure of the neural CRF encoder is demonstrated in Fig 2. Note that the MLP layer is also interchangeable with a recurrent neural network (RNN) layer or LSTM layer. But in our pilot experiments, we found a single MLP structure yields better performance, which we conjecture is due to over-fitting caused by the high complexity of those alternatives.

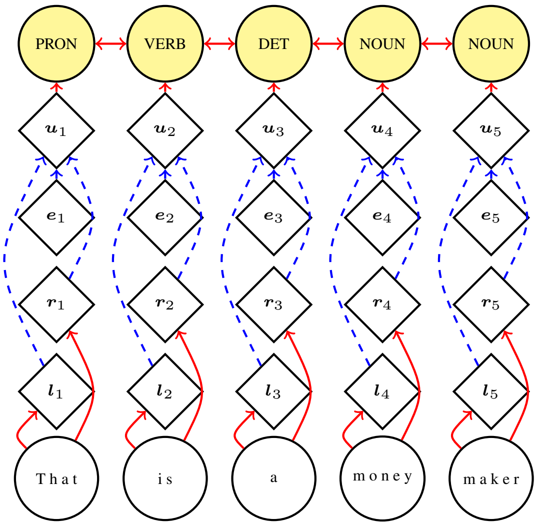


Figure 2: A demonstration of the neural CRF encoder. $\boldsymbol{l}_t$ and $\boldsymbol{r}_t$ are the output of the forward and backward character-level LSTM of the word at position t in a sentence, and $\boldsymbol{e}_t$ is the word-level embedding of that word. $\boldsymbol{u}_t$ is the concatenation of $\boldsymbol{e}_t, \boldsymbol{l}_t$ and $\boldsymbol{r}_t$, denoted by blue dashed arrows.

3.2 Generative Decoder

In our NCRF-AE, we assume the generative process follows several multinomial distributions: each label y has the probability $\theta_{y\to x}$ to reconstruct the corresponding word x, i.e., $P(x|y) = \theta_{y\to x}$. This setting naturally leads to a constraint $\sum_x \theta_{y\to x} = 1$. The number of parameters of the decoder is $|\mathcal{Y}| \times |\mathcal{X}|$. For a whole sequence, the reconstruction probability is $P_{\boldsymbol{\Theta}}(\hat{\boldsymbol{x}}|\boldsymbol{y}) = \prod_t P(\hat{x}_t|y_t)$.

4 A Unified Learning Framework

We first constructed two loss functions for labeled and unlabeled data using the same model. Our model is trained in an on-line fashion: given a labeled or unlabeled sentence, our NCRF-AE optimizes the loss function by choosing the corresponding one. In an analogy to coordinate descent, we optimize the loss function of the NCRF-

AE by alternatively updating the parameters $\mathbf{\Theta}$ in the decoder and the parameters $\mathbf{\Lambda}$ in the encoder. The parameters $\mathbf{\Theta}$ in the decoder are updated via a variation of the Expectation-Maximization (EM) algorithm, and the the parameters $\mathbf{\Lambda}$ in the encoder are updated through a gradient-based method due to the non-convexity of the neuralized CRF. In contrast to the early autoencoder models (Ammar et al., 2014; Lin et al., 2015), our model has two distinctions: First, we have two loss functions to model labeled example and unlabeled examples; Second, we designed a variant of EM algorithm to alternatively learn the parameters of the encoder and the decoder at the same time.

4.1 Unified Loss Functions for Labeled and unlabeled Data

For a sequential input with labels, the complete data likelihood given by our NCRF-AE is

$$\begin{aligned} P_{\mathbf{\Theta},\mathbf{\Lambda}}(\hat{\boldsymbol{x}},\boldsymbol{y}|\boldsymbol{x}) &= P_{\mathbf{\Theta}}(\hat{\boldsymbol{x}}|\boldsymbol{y})P_{\mathbf{\Lambda}}(\boldsymbol{y}|\boldsymbol{x}) \\ &= \left[\prod_t P(\hat{x}_t|y_t)\right]\frac{e^{\Phi(\boldsymbol{x},\boldsymbol{y})}}{Z} \\ &= \frac{e^{\sum_t s_t(\boldsymbol{x},\boldsymbol{y})}}{Z}, \end{aligned}$$

where

$$s_t(\boldsymbol{x},\boldsymbol{y}) = \log P(x_t|y_t) + \phi(\boldsymbol{x},y_t) + \psi(y_{t-1},y_t).$$

If the input sequence is unlabeled, we can simply marginalize over all the possible assignment to labels. The probability is formulated as

$$\begin{aligned} P_{\mathbf{\Theta},\mathbf{\Lambda}}(\hat{\boldsymbol{x}}|\boldsymbol{x}) &= \sum_{\boldsymbol{y}} P(\hat{\boldsymbol{x}},\boldsymbol{y}|\boldsymbol{x}) \\ &= \frac{U}{Z}, \end{aligned}$$

where $U = \sum_{\boldsymbol{y}} e^{\sum_t s_t(\boldsymbol{x},\boldsymbol{y})}$.

Our formulation have two advantages. First, term U is different from but in a similar form as term Z, such that to calculate the probability $P(\hat{\boldsymbol{x}}|\boldsymbol{x})$ for an unlabeled sequence, the forward-backward algorithm to compute the partition function Z can also be applied to compute U efficiently. Second, our NCRF-AE highlights a unified structure of different loss functions for labeled and unlabeled data with shared parameters. Thus during training, our model can address both labeled and unlabeled data well by alternating the loss functions. Using negative log-likelihood as our loss function, if the data is labeled, the loss function is:

$$\begin{aligned} loss_l &= -\log P_{\mathbf{\Theta},\mathbf{\Lambda}}(\hat{\boldsymbol{x}},\boldsymbol{y}|\boldsymbol{x}) \\ &= -(\sum_t s_t(\boldsymbol{x},\boldsymbol{y}) - \log Z) \end{aligned}$$

If the data is unlabeled, the loss function is:

$$\begin{aligned} loss_u &= -\log P_{\mathbf{\Theta},\mathbf{\Lambda}}(\hat{\boldsymbol{x}}|\boldsymbol{x}) \\ &= -(\log U - \log Z). \end{aligned}$$

Thus, during training, based on whether the encountered data is labeled or unlabeled, our model can select the appropriate loss function for learning parameters. In practice, we found for labeled data, using a combination of $loss_l$ and $loss_u$ actually yields better performance.

5 Mixed Expectation-Maximization Algorithm

The Expectation-Maximization (EM) algorithm (Dempster et al., 1977) was applied to a wide range of problems. Generally, it establishes a lower-bound of the objective function by using Jensen's Inequality. It first tries to find the posterior distribution of the latent variables, and then based on the posterior distribution of the latent variables, it maximizes the lower-bound. By alternating expectation (E) and maximization (M) steps, the algorithm iteratively improves the lower-bound of the objective function.

In this section we describe the mixed Expectation-Maximization (EM) algorithm used in our study. Parameterized by the encoding parameters $\mathbf{\Lambda}$ and the reconstruction parameters $\mathbf{\Theta}$, our NCRF-AE consists of the encoder and the decoder, which together forms the log-likelihood a highly non-convex function. However, a careful observation shows that if we fix the encoder, the lower bound derived in the E step, is convex with respect to the reconstruction parameters $\mathbf{\Theta}$ in the M step. Hence, in the M step we can analytically obtain the global optimum of $\mathbf{\Theta}$. In terms of the reconstruction parameters $\mathbf{\Theta}$ by fixing $\mathbf{\Lambda}$, we describe our EM algorithm in iteration t as follows:

In the E-step, we let $Q(\boldsymbol{y}_i) = P(\boldsymbol{y}_i|\boldsymbol{x}_i,\hat{\boldsymbol{x}}_i)$, and treat $\boldsymbol{y_i}$ the latent variable as it is not observable in unlabeled data. We derive the lower-bound of the log-likelihood using $Q(\boldsymbol{y}_i)$:

$$\begin{aligned}\sum_i \log P(\hat{\boldsymbol{x}}_i|\boldsymbol{x}_i) &= \sum_i \log \sum_{\boldsymbol{y}_i} Q(\boldsymbol{y}_i) \frac{P(\hat{\boldsymbol{x}}_i, \boldsymbol{y}_i|\boldsymbol{x}_i)}{Q(\boldsymbol{y}_i)} \\ &\geq \sum_i \sum_{\boldsymbol{y}_i} Q(\boldsymbol{y}_i) \log \frac{P(\hat{\boldsymbol{x}}_i, \boldsymbol{y}_i|\boldsymbol{x}_i)}{Q(\boldsymbol{y}_i)},\end{aligned}$$

where $Q(\boldsymbol{y}_i)$ is computed using parameters $\boldsymbol{\Theta}^{(t-1)}$ in the previous iteration $t-1$.

In the M-step, we try to improve the aforementioned lower-bound using all examples:

$$\begin{aligned}&\arg\max_{\boldsymbol{\Theta}^{(t)}} \sum_i \sum_{\boldsymbol{y}_i} Q(\boldsymbol{y}_i) \log \frac{P_{\boldsymbol{\Theta}^{(t)}}(\hat{\boldsymbol{x}}_i|\boldsymbol{y}_i) P_{\boldsymbol{\Lambda}}(\boldsymbol{y}_i|\boldsymbol{x}_i)}{Q(\boldsymbol{y}_i)} \\ &\arg\max_{\boldsymbol{\Theta}^{(t)}} \sum_i \sum_{\boldsymbol{y}_i} Q(\boldsymbol{y}_i) \log P_{\boldsymbol{\Theta}^{(t)}}(\hat{\boldsymbol{x}}_i|\boldsymbol{y}_i) + const \\ &\arg\max_{\boldsymbol{\Theta}^{(t)}} \sum_{y \to x} \log \theta^{(t)}_{y \to x} \sum_y Q(y) C(y,x) \\ &\arg\max_{\boldsymbol{\Theta}^{(t)}} \sum_{y \to x} \log \theta^{(t)}_{y \to x} \mathrm{E}_{y \sim Q}[C(y,x)] \\ &\qquad s.t. \sum_x \theta^{(t)}_{y \to x} = 1.\end{aligned}$$

In this formulation, *const* is a constant with respect to the parameters we are updating. $Q(y)$ is the distribution of a label y at any position by marginalizing labels at all other positions in a sequence. By denoting $C(y,x)$ as the number of times that (x,y) co-occurs, $\mathrm{E}_{y \sim Q_{\boldsymbol{\Theta}^{(t-1)}}}[C(y,x)]$ is the expected count of a particular reconstruction at any position, which can also be calculated using Baum-Welch algorithm (Welch, 2003), and can be summed over for all examples in the dataset (In the labeled data, it is just a real count). The algorithm we used to calculate the expected count is described in Algorithm 1. Therefore, it can be shown that the aforementioned global optimum can be calculated by simply normalizing the expected counts. In terms of the encoder's parameters $\boldsymbol{\Lambda}$, they are first updated via a gradient-based optimization before each EM iteration. Based on the above discussion, our Mixed EM Algorithm is presented in Algorithm 2.

Algorithm 1 Obtain Expected Count (T_e)

```
Require: the expected count table $T_e$
for an unlabeled data example $\boldsymbol{x}_i$ do
    Compute the forward messages: $\alpha(y,t)$  $\forall y,t$.    ▷ $t$ is the position in a sequence.
    Compute the backward messages: $\beta(y,t)$  $\forall y,t$.
    Calculate the expected count for each $x$ in $\boldsymbol{x}_i$: $P(y_t|x_t) \propto \alpha(y,t) \times \beta(y,t)$.
    $T_e(x_t,y_t) \leftarrow T_e(x_t,y_t) + P(y_t|x_t)$    ▷ $T_e$ is the expected count table.
end for
```

Algorithm 2 Mixed Expectation-Maximization

```
Initialize expected count table $T_e$ using labeled data $\{\boldsymbol{x},\boldsymbol{y}\}^l_i$ and use it as $\boldsymbol{\Theta}^{(0)}$ in the decoder.
Initialize $\boldsymbol{\Lambda}^{(0)}$ in the encoder randomly.
for $t$ in epochs do
    Train the encoder on labeled data $\{\boldsymbol{x},\boldsymbol{y}\}^l$ and unlabeled data $\{\boldsymbol{x}\}^u$ to update $\boldsymbol{\Lambda}^{(t-1)}$ to $\boldsymbol{\Lambda}^{(t)}$.
    Re-initialize expected count table $T_e$ with 0s.
    Use labeled data $\{\boldsymbol{x},\boldsymbol{y}\}^l$ to calculate real counts and update $T_e$.
    Use unlabeled data $\{\boldsymbol{x}\}^u$ to compute the expected counts with parameters $\boldsymbol{\Lambda}^{(t)}$ and $\boldsymbol{\Theta}^{(t-1)}$ and update $T_e$.
    Obtain $\boldsymbol{\Theta}^{(t)}$ globally and analytically based on $T_e$.
end for
```

This mixed EM algorithm is a combination of the gradient-based approach to optimize the encoder by minimizing the negative log-likelihood as the loss function, and the EM approach to update the decoder's parameters by improving the lower-bound of the log-likelihood.

6 Experiments

6.1 Experimental Settings

Dataset We evaluated our model on the POS tagging task, in both the supervised and semi-supervised learning settings, over eight different languages from the UD (Universal Dependencies) 1.4 dataset (Mcdonald et al., 2013). The task is defined over 17 different POS tags, used across the different languages. We followed the original

	English	French	German	Italian	Russian	Spanish	Indonesian	Croatian
Tokens	254830	391107	293088	272913	99389	423346	121923	139023
Training	12543	14554	14118	12837	4029	14187	4477	5792
Development	2002	1596	799	489	502	1552	559	200
Testing	2077	298	977	489	499	274	297	297

Table 1: Statistics of different UD languages used in our experiments, including the number of tokens, and the number of sentences in training, development and testing set respectively.

UD division for training, development and testing in our experiments. The statistics of the data used in our experiments are described in table 1. The UD dataset includes several low-resource languages which are of particular interest to our semi-supervised model.

Input Representation and Neural Architecture Our model uses word embeddings as input. In our pilot experiments, we compared the performance on the English dataset of the pre-trained embedding from Google News (Mikolov et al., 2013) and the embeddings we trained directly on the UD dataset using the skip-gram algorithm (Mikolov et al., 2013). We found these two types of embeddings yield very similar performance on the POS tagging task. So in our experiments, we used embeddings of different languages directly trained on the UD dataset as input to our model, whose dimension is 200. For the MLP neural network layer, the number of hidden nodes in the hidden layer is 20, which is the same for the hidden layer in the character-level LSTM. The dimension of the character-level embeddings sent into the LSTM layer is 15, which is randomly initialized. In order to incorporate the global information of the input sequence, we set the context window size to 3. The dropout rate for the dropout layer is set to 0.5.

Learning We used ADADELTA (Zeiler, 2012) to update parameters $\mathbf{\Lambda}$ in the encoder, as ADADELTA dynamically adapts learning rate over time using only first order information and has minimal computational overhead beyond vanilla stochastic gradient descent (SGD). The authors of ADADELTA also argue this method appears robust to noisy gradient information, different model architecture choices, various data modalities and selection of hyper-parameters. We observed that ADADELTA indeed had faster convergence than vanilla SGD optimization. In our experiments, we include word embeddings and character embeddings as parameters as well. We used Theano to implement our algorithm, and all the experiments were run on NVIDIA GPUs. To prevent over-fitting, we used the "early-stop" strategy to determine the appropriate number of epochs during training. We did not take efforts to tune those hyper-parameters and they remained the same in both our supervised and semi-supervised learning experiments.

6.2 Supervised Learning

In these settings our Neural CRF autoencoder model had access to the full amount of annotated training data in the UD dataset. As described in Section 5, the decoder's parameters $\mathbf{\Theta}$ were estimated using real counts from the labeled data.

We compared our model with existing sequence labeling models including HMM, CRF, LSTM and neural CRF (NCRF) on all the 8 languages. Among these models, the NCRF can be most directly compared to our model, as it is used as the base of our model, but without the decoder (and as a result, can only be used for supervised learning).

The results, summarized in Table 2, show that our NCRF-AE consistently outperformed all other systems, on all the 8 languages, including Russian, Indonesian and Croatian which had considerably less data compared to other languages. Interestingly, the NCRF consistently came second to our model, which demonstrates the efficacy of the expressivity added to our model by the decoder, together with an appropriate optimization approach.

To better understand the performance difference by different models, we performed error analysis, using an illustrative example, described in Figure 3.

In this example, the LSTM incorrectly predicted the POS tag of the word "*search*" as a verb, instead of a noun (part of the NP "*nice search engine*"), while predicting correctly the preceding word, "*nice*", as an adjective. We attribute the error to LSTM lacking an explicit output transition scoring function, which would penalize the ungrammatical transition between "ADJ" and "VERB".

Models	English	French	German	Italian	Russian	Spanish	Indonesian	Croatian
HMM	86.28%	91.23%	85.59%	92.03%	79.82%	91.31%	89.40%	86.98%
CRF	89.96%	93.40%	86.83%	94.07%	83.38%	91.47%	88.63%	86.90%
LSTM	90.50%	94.16%	88.40%	94.96%	84.87%	93.17%	89.42%	88.95%
NCRF	91.52%	95.07%	90.27%	96.20%	93.37%	93.34%	92.32%	93.85%
NCRF-AE	**92.50%**	**95.28%**	**90.50%**	**96.64%**	**93.60%**	**93.86%**	**93.96%**	**94.32%**

Table 2: Supervised learning accuracy of POS tagging on 8 UD languages using different models

Models	English	French	German	Italian	Russian	Spanish	Indonesian	Croatian
NCRF (OL)	88.01%	93.38%	90.43%	91.75%	86.63%	91.22%	88.35%	86.11%
NCRF-AE (OL)	88.41%	93.69%	90.75%	92.17%	87.82%	91.70%	89.06%	87.92%
HMM-EM	79.92%	88.15%	77.01%	84.57%	72.96%	86.77%	83.61%	77.20%
NCRF-AE (HEM)	86.79%	92.83%	89.78%	90.68%	86.39%	91.30%	88.86%	86.55%
NCRF-AE	**89.43%**	**93.89%**	**90.99%**	**92.85%**	**88.93%**	**92.17%**	**89.41%**	**89.14%**

Table 3: Semi-supervised learning accuracy of POS tagging on 8 UD languages. HEM means hard-EM, used as a self-training approach, and OL means only 20% of the labeled data is used and no unlabeled data is used.

Text	*Google*	*is*	*a*	*nice*	*search*	*engine*	*.*
Gold	PROPN	VERB	DET	ADJ	NOUN	NOUN	PUNCT
NCRF-AE	PROPN	VERB	DET	ADJ	NOUN	NOUN	PUNCT
NCRF	NOUN	VERB	DET	ADJ	NOUN	NOUN	PUNCT
LSTM	PROPN	VERB	DET	ADJ	VERB	NOUN	PUNCT

Figure 3: An example from the test set to compare the predicted results of our NCRF-AE model, the NCRF model and the LSTM model.

The NCRF, which does score such transitions, correctly predicted that word. However, it incorrectly predicted "*Google*" as a noun rather than a proper-noun. This is a subtle mistake, as the two are grammatically and semantically similar. This mistake appeared consistently in the NCRF results, while NCRF-AE predictions were correct.

We attribute this success to the superior expressivity of our model: The prediction is done jointly by the encoder and the decoder, as the reconstruction decision is defined over all output sequences, picking the jointly optimal sequence. From another perspective, our NCRF-AE model is a combination of discriminative and generative models, in that sense the decoder can be regarded as a soft constraint that supplements the encoder. Such that, the decoder performs as a regularizer to check-balance the choices made by the encoder.

6.3 Semi-supervised Learning

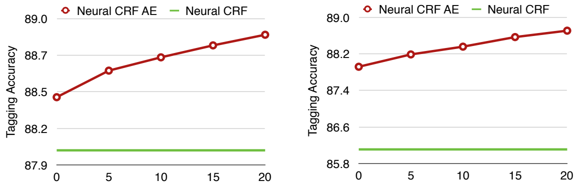
In the semi-supervised settings we compared our models with other semi-supervised structured prediction models. In addition, we studied how varying the amount of unlabeled data would change the performance of our model.

As described in Sec. 5, the decoder's parameters $\boldsymbol{\Theta}$ are initialized by the labeled dataset using real counts and updated in training.

6.3.1 Varying Unlabeled Data Proportion

We first experimented with varying the proportion of unlabeled data, while fixing the amount of labeled data. We conducted these experiments over two languages, English and low-resource language Croatian. We fixed the proportion of labeled data at 20%, and gradually added more unlabeled data from 0% to 20% (from full supervision to semi-supervision). The unlabeled data was sampled from the same dataset (without overlapping with the labeled data), with the labels removed. The results are shown in Figure 4.

The left most point of both sub-figures is the accuracy of fully supervised learning with 20% of the whole data. As we can observe, the tagging accuracy increased as the proportion of unlabeled data increased.



(a) English

(b) Croatian

Figure 4: UD English and Croatian POS tagging accuracy versus increasing proportion of unlabeled sequences using 20% labeled data. The green straight line is the performance of the neural CRF, trained over the labeled data.

6.3.2 Semi-supervised POS Tagging on Multiple Languages

We compared our NCRF-AE model with other semi-supervised learning models, including the HMM-EM algorithm and the hard-EM version of our NCRF-AE. The hard EM version of our model can be considered as a variant of self-training, as it infers the missing labels using the current model in the E-step, and uses the real counts of these labels to update the model in the M-step. To contextualize the results, we also provide the results of the NCRF model and the supervised version our NCRF-AE model trained on 20% of the data. We set the proportion of labeled data to 20% for each language and set the proportion of unlabeled data to 50% of the dataset. There was no overlap between labeled and unlabeled data.

The results are summarized in Table 3. Similar to the supervised experiments, the supervised version of our NCRF-AE, trained over 20% of the labeled data, outperforms the NCRF model. Our model was able to successfully use the unlabeled data, leading to improved performance in all languages, over both the supervised version of our model, as well as the HMM-EM and Hard-EM models that were also trained over both the labeled and unlabeled data.

6.3.3 Varying Sizes of Labeled Data on English

As is known to all, semi-supervised approaches tend to work well when given a small size of labeled training data. But with the increase of labeled training data size, we might get diminishing effectiveness. To verify this conjecture, we conducted additional experiments to show how varying sizes of labeled training data affect the effectiveness of our NCRF-AE model. In these experiments, we gradually increased the proportion of labeled data, and in accordance decreased the proportion of unlabeled data.

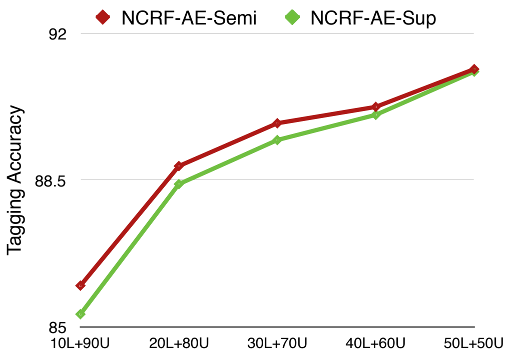


Figure 5: Performance of the NCRF-AE model on different proportion of labeled and unlabeled data. The green line shows the results on only labeled data, and the red line on both labeled and unlabeled data. The difference between the red line and the green line are gradually vanishing.

The results of these experiments are demonstrated in Figure 5. As we speculated, we observed diminishing effectiveness when increasing the proportion of labeled data in training.

7 Conclusion

We proposed an end-to-end neural CRF autoencoder (NCRF-AE) model for semi-supervised sequence labeling. Our NCRF-AE is an integration of a discriminative model and generative model which extends the generalized autoencoder by using a neural CRF model as its encoder and a generative decoder built on top of it. We suggest a variant of the EM algorithm to learn the parameters of our NCRF-AE model.

We evaluated our model in both supervised and semi-supervised scenarios over multiple languages, and show it can outperform other supervised and semi-supervised methods. Additional experiments suggest how varying sizes of labeled training data affect the effectiveness of our model.

These results demonstrate the strength of our model, as it was able to utilize the small amount of labeled data and exploit the hidden information from the large amount of unlabeled data, without additional feature engineering which is often needed in order to get semi-supervised and weakly-supervised systems to perform well. The superior performance on the low resource language also suggests its potential in practical use.

References

Waleed Ammar, Chris Dyer, and Noah A Smith. 2014. Conditional random field autoencoders for unsupervised structured prediction. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 1–12.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 2442–2452.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 1545–1554.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. International Conference on Learning Representation (ICLR)*.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Yong Cheng, Yang Liu, and Wei Xu. 2017. Maximum reconstruction estimation for generative latent-variable models. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B*, 39(1):1–38.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. pages 302–312.

Geoffrey E Hinton and Richard S Zemel. 1994. Autoencoders, minimum description length and helmholtz free energy. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 3–10.

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

Arzoo Katiyar and Claire Cardie. 2016. Investigating lstms for joint extraction of opinion entities and relations. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 919–929, Berlin, Germany.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 1746–1751.

Tomás Kociský, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 1078–1087.

John D Lafferty, Andrew McCallum, and Fernando C N Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 282–289, San Francisco, CA, USA.

Guillaume Lample, Miguel Ballesteros, Kazuya Kawakami, Sandeep Subramanian, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 1–10.

Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. 2015. Unsupervised pos induction with word embeddings. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 1311–1316.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 1064–1074, Berlin, Germany.

Z. Marinho, A. F. T. Martins, S. B. Cohen, and N. A. Smith. 2016. Semi-supervised learning of sequence models with the method of moments. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 152–159, Stroudsburg, PA, USA.

Ryan Mcdonald, Joakim Nivre, Yvonne Quirmbach-brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Tckstrm, Claudia Bedini, Nria Bertomeu, and Castell Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*.

G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu, and G. Zweig. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 319–328, Austin, Texas.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 1–9.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for chinese social media with word segmentation representation learning. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 149–155.

Ariadna Quattoni, Sybor Wang, Louis-Philippe Morency, Michael Collins, and Trevor Darrell. 2007. Hidden conditional random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(10):1848–1852.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proc. of the Annual Meeting of the Association Computational Linguistics (ACL)*, pages 455–465, Sofia, Bulgaria.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How less is more in unsupervised dependency parsing. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 751–759.

Karl Stratos and Michael Collins. 2015. Simple semi-supervised pos tagging. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

Amarnag Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 1017–1024, New York, NY, USA.

Ke M. Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. 2016. Unsupervised neural hidden markov models. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 63–71, Austin, TX.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with lstms. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 232–237.

Lloyd R Welch. 2003. Hidden markov models and the baum-welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4):1,10–13.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*, pages 1296–1306, Austin, Texas.

Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701.

TAG Parsing with Neural Networks and Vector Representations of Supertags

Jungo Kasai
Dept. of Computer Science
Yale University
jungo.kasai@yale.edu

Robert Frank
Dept. of Linguistics
Yale University
robert.frank@yale.edu

R. Thomas McCoy
Dept. of Linguistics
Yale University
richard.mccoy@yale.edu

Owen Rambow
DSI
Columbia University
rambow@ccls.columbia.edu

Alexis Nasr
LIF
Université Aix Marseille
Alexis.Nasr@lif.univ-mrs.fr

Abstract

We present supertagging-based models for Tree Adjoining Grammar parsing that use neural network architectures and dense vector representation of supertags (elementary trees) to achieve state-of-the-art performance in unlabeled and labeled attachment scores. The shift-reduce parsing model eschews lexical information entirely, and uses only the 1-best supertags to parse a sentence, providing further support for the claim that supertagging is "almost parsing." We demonstrate that the embedding vector representations the parser induces for supertags possess linguistically interpretable structure, supporting analogies between grammatical structures like those familiar from recent work in distributional semantics. This dense representation of supertags overcomes the drawbacks for statistical models of TAG as compared to CCG parsing, raising the possibility that TAG is a viable alternative for NLP tasks that require the assignment of richer structural descriptions to sentences.

1 Introduction

Recent work has applied Combinatory Categorial Grammar (CCG, Steedman and Baldridge (2011)) to the problem of broad-coverage parsing in order to derive grammatical representations that are sufficiently rich to support tasks requiring deeper representation of a sentence's meaning (Lewis et al., 2015; Reddy et al., 2016; Nadejde et al., 2017). Yet CCG is only one of a number of mildly context-sensitive grammar formalisms that can provide such rich representations, and each has distinct advantages. In this paper we explore the applicability of another formalism, Tree Adjoining Grammar (TAG, Joshi and Schabes (1997)), to the task of broad-coverage parsing.

TAG and CCG share the property of lexicalization: words are associated with elementary units of grammatical structure which are composed during a derivation using one of a small set of operations to produce a parse tree. The task of parsing involves the construction of a derivation tree that encodes the application of this set of actions to a set of elementary lexically-associated objects. TAG differs from CCG in having an even richer set of lexical units, so that the identification of these units in a derivation could be even more informative for subsequent tasks involving semantic interpretation, translation and the like, which have been the focus of CCG-based work.

The elementary units of CCG and TAG (categories for CCG, and elementary trees for TAG) determine a word's combinatory potential, in a way that is not the case for the usual part-of-speech tags used in parsing. Indeed, the assignment of elementary objects to the words in a sentence almost determines the possible parse for a sentence. The near uniqueness of a parse given a sequence of lexical units motivated Bangalore and Joshi (1999) to decompose the parsing problem into two phases: *supertagging*, where elementary objects, or supertags, are assigned to each word, and *stapling*, where these supertags are combined together. They claim that given a perfect supertagger, a parse of a sentence follows from syntactic features provided by the supertags, and therefore, supertagging is "almost parsing." This claim has been confirmed in subsequent work: it has been shown that the task of parsing given a gold sequence of supertags can achieve high accuracy (TAG: (Bangalore et al., 2009; Chung et al., 2016), CCG: (Lewis et al., 2016)). However, it has also been revealed that the difficulty of supertagging, because of the large set of possible supertags, re-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1712–1722
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

sults in inaccuracies that prevent us from effectively utilizing syntactic information provided by the imperfect set of supertags that are assigned. This problem is even more severe for TAG parsing. TAG differs from CCG in having a smaller set of combinatory operations, but a more varied set of elementary objects: the TAG-annotated version of the Penn Treebank that we use (Chen, 2001) includes 4727 distinct supertags (2165 occur once) while the CCG-annotated version (Hockenmaier and Steedman, 2007) includes 1286 distinct supertags (439 occur once). As a result, building a robust, broad-coverage TAG parser has proven difficult.

In this work, we show that robust supertagging-based parsing of TAG is indeed possible by using a dense representation of supertags that is induced using neural networks. In the first half of the paper, we present a neural network supertagger based on a bi-directional LSTM (BLSTM) architecture, inspired by the work of Xu (2015) and Lewis et al. (2016) in CCG, and we make crucial use of synchronized dropout (Gal and Ghahramani, 2016). This supertagger achieves the state-of-the-art accuracy on the WSJ Penn Treebank. When combined with an existing TAG chart parser (Bangalore et al., 2009), the LSTM-based supertagger already yields state-of-the-art unlabeled and labeled attachment scores.

In the second half of the work, we present a shift-reduce parsing model based on a feed-forward neural network that makes use of dense supertag embeddings. Although this approach has much in common with the approach to shift-reduce CCG parsing taken by Zhang and Clark (2011), it differs in its additive structures in supertag embeddings. When a CCG operation combines two supertags (categories), it yields a resulting category that is typically distinct from the two that are combined, and CCG shift-reduce parsers (e.g. Xu (2015)) make use of this result to guide subsequent actions. When the resulting category is the same as some lexical category assignment (for example when function application over $(S\backslash NP)/NP$ NP yields $S\backslash NP$, the same as an intransitive verb), the parser will benefit from sharing statistics across these contexts. For TAG however, substitution or adjoining of one elementary tree into another does not change the nature of the elementary tree into which the operation has taken place. Consequently, the results of partial derivations are not identified with atomic lexical entries, resulting in sparser data. We propose a solution to this problem for TAG by introducing vector representations that are added to the supertag embedding when an operation has been applied to an elementary tree. Not only does this result in a TAG-parser with the best known performance over the WSJ Penn Treebank, but the resulting supertag embeddings turn out to contain linguistically sensible linear structure that we illustrate.

2 TAG Parsing and Dependency Parsing

TAG is a tree-rewriting system, and typically the elementary structures of a TAG are phrase structure trees. Thus, TAG-derived structures are also phrase structure trees. In addition, a TAG derivation also yields a record of the derivational operations (substitutions, adjunctions) used to produce the derived tree. Since these operations are context-free, this record also forms a tree, called the *derivation tree*, whose nodes are the elementary objects and the edges are combinatory operations. If we assume the TAG is lexicalized (i.e., each elementary structure is anchored by at least one terminal symbol), then we can label the nodes of the derivation tree with the tree names and also the anchors of the elementary trees, and we obtain what is *formally* a dependency tree.[1] Since each node is also labeled with an elementary tree from the grammar, we can associate rich linguistic structure with that node, such as passive voice or empty subject.

In addition, it has long been observed that the derivation tree can also be interpreted *linguistically* as a dependency tree (Rambow and Joshi, 1997), if certain assumptions are made about the shape of the elementary trees in the grammar (Frank, 2001). The substitution operation corresponds to the obligatory addition of an argument, and adjunction is used to add adjuncts, as well as function words to a lexical head. The one exception is the treatment of long distance *wh*-movement in TAG. Here, a matrix clause is represented by a *predicative auxiliary tree* which is adjoined into the embedded clause, so that the *wh*-element moved from the embedded clause can still be substituted locally into the tree headed by its verb. As a result, the dependency between the matrix and embedded verbs is inverted relative to the

[1] For the difference between formal and linguistic dependency, see Rambow (2010).

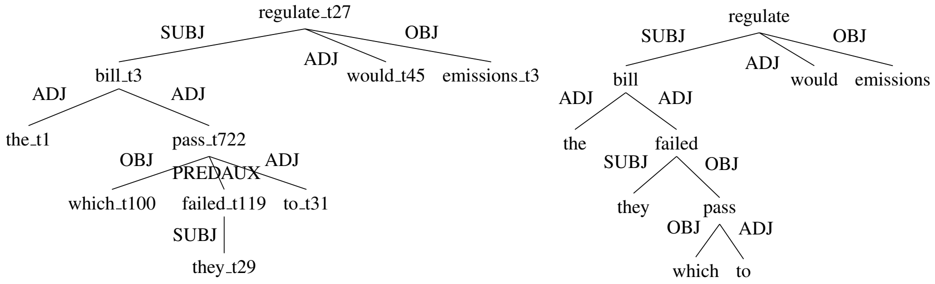


Figure 1: TAG derivation tree (left) and closely related dependency tree (right) for *The bill, which they failed to pass, would regulate emissions*. Substitution edges are labeled SUBJ or OBJ, predicative auxiliary edges are labeled PREDAUX, while all other adjoining edges are labeled ADJ. We use the same edge labels in the dependency tree. The derivation tree also carries the name of the elementary tree used during the derivation, which can be used to look up rich syntactic information about that word in context.

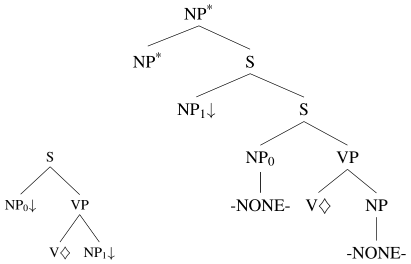


Figure 2: The elementary trees for t27 (left) and t722 (right).

normally assumed dependency. This can be seen in Figure 1, where in the linguistically motivated dependency tree (right) *pass* depends on *failed* as the latter's object, while in the TAG derivation tree (left), *failed* depends on *pass*, linked by an arc marked PREDAUX for predicative auxiliary. These cases can be detected automatically because of the trees used; as a result of this inversion, there is almost no non-projectivity in English. In summary, TAG parsing into derivation trees is very closely related to dependency parsing. In this paper, we are interested in extracting derivation trees, not the derived trees (which can be recovered from the derivation trees).

The corpus we use is obtained by extracting a TAG grammar from the WSJ part of the Penn Treebank corpus, resulting in a grammar and derivation trees labeled with the grammar Chen (2001). For example, in Figure 1, t27 is the basic tree for a transitive verb (*regulate*), while t722 is the tree for a transitive verb which forms an object relative clause with an overt relative pronoun but an empty subject (Figure 2).[2] The corpus and grammar were iteratively refined to obtain linguistically plausible derivation trees which could serve as input for a generation task (Bangalore and Rambow, 2000). As a result, the dependency structure is similar to Universal Dependency (Nivre et al., 2016), apart from the different treatment of long-distance *wh*-movement noted above: the primary dependencies are between the core meaning-bearing lexical words, while function words (auxiliaries, determiners, complementizers) depend on their lexical head and have no dependents.[3] We label verbal argument arcs with deep dependency labels: Subject, Object, and Indirect Object normalized for passive and dative shift. All other arcs are labeled as Adjuncts. This means that our label set is small, but determining the argument labels requires detection of voice alternations and dative shift.

3 TAG Supertagging

3.1 Long-Distance Dependencies and LSTMs

In CCG, a transitive verb is uniformly associated with the category $(S\backslash NP)/NP$, and variation in the word order of a clause is addressed through the use of different combinatory operations. This results in greater parsing ambiguity given a sequence of categories. In TAG, the set of operations is more restricted. While this has the positive effect of reducing parsing ambiguity given a sequence of elementary trees, it necessitates a proliferation in the number of elementary trees. For example, a TAG will associate different elementary trees for the same transitive verb in order to derive canonical clauses, subject and object relatives, and subject and object questions. Not only does this lead to a larger number of supertags, it

[2] Our full grammar is shown at http://mica.lif.univ-mrs.fr/d6.clean2-backup.pdf

[3] One difference should be noted: UD considers prepositions always to be function words, while our TAG grammar treats them as core words unless the Penn Treebank marks them as closely related to the verb.

also means that the determination of the correct supertag requires sensitivity to long-distance dependencies. For example, in the question *Who does Bill think Harry likes?*, the category of the verb *like* requires sensitivity to the first word of the sentence. To address this problem, we make use of a supertagging model that is based on a recurrent network architecture, the Long Short-Term Memory (LSTM, Hochreiter and Schmidhuber (1997)), which is constructed so that its update rule avoids the vanishing/exploding gradient problem.

3.2 Supertagger Model

The model architecture we adopt is depicted in Figure 3, a BLSTM. The input for each word is represented via the concatenation of a 100-dimensional embedding of the word, a 5-dimensional embedding of a predicted part of speech tag, and a 10-dimensional embedding of a suffix vector (which encodes the presence of 1 and 2 character suffixes of the word). We initialize the word embeddings to be the pre-trained GloVe vectors (Pennington et al., 2014); for words which do not have a corresponding GloVe vector, we initialize their embedding to a zero vector. The other embeddings are randomly initialized. Features for each word are fed into the BLSTMs. To produce an output for the network, we concatenate the output vectors from the two LSTM directions and apply an affine transformation before the softmax function to obtain a probability distribution over the 4727 supertags. We train this network, including the embeddings, by optimizing the negative log-likelihood of the observed sequences of supertags in a mini-batch stochastic fashion with the Adam optimization algorithm with $l = 0.001$ (Kingma and Ba, 2015).

Since neural networks have numerous parameters, regularization plays a key role in training. This is typically accomplished by using dropout (Srivastava et al., 2014). Although dropout training has been successful on feed-forward neural networks, performing dropout on recurrent neural networks has been problematic (Gal and Ghahramani, 2016). Armed with a novel interpretation of dropout based on variational inference on parameters, Gal and Ghahramani (2016) propose that dropout noise should be shared across the time steps. We apply this technique to the training of our LSTM network, and achieve an improvement of approximately 2% in accuracy.

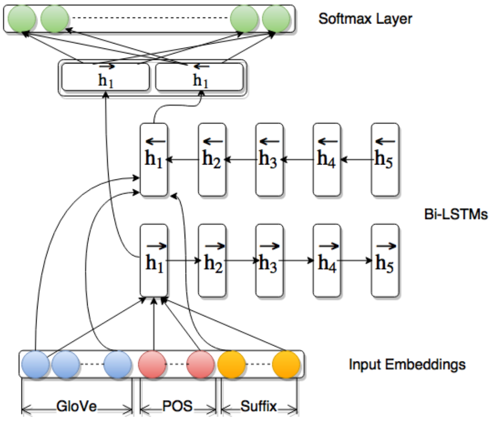

Softmax Layer
$\overrightarrow{h}_1$
$\overleftarrow{h}_1$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_4$ $\overleftarrow{h}_5$
Bi-LSTMs
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_4$ $\overrightarrow{h}_5$
Input Embeddings
GloVe
POS
Suffix

Figure 3: BLSTM Supertagger Architecture.

4 Transition-based Parsing for TAG

As discussed in Section 2, TAG parsing into derivation trees is closely related to dependency parsing; it is natural to make use of techniques from dependency parsing to reconstruct a TAG derivation tree. We make use of this approach here, eschewing complete chart-based parsing algorithms in favor of greedy or beam-search-based explorations of possible parses.

4.1 Shift-Reduce Parsing Algorithm

We employ the arc-eager system of shift-reduce parsing, familiar from the MALT parser (Nivre et al., 2006). In this system, an oracle is trained to predict a sequence of transition operations from an initial state to a terminal state for each sentence. Each state is represented by $c = (s, b, A)$ where s, b, and A denote the stack, buffer and set of dependency relations derived so far. Therefore, our objective is to predict a transition operation given the configuration set c. The initial configuration is defined as $s = [ROOT]$, $b = [w_1, \cdots w_n]$, and $A = \emptyset$ where n is the number of tokens in the sentence $w_1 w_2 \cdots w_n$. At a particular state, denote the top ith element of the stack and the buffer by s_i and b_i respectively. The arc-eager system defines four types of operations with corresponding preconditions: LEFT-ARC, RIGHT-ARC, SHIFT and REDUCE. For the present parser, the LEFT-ARC and RIGHT-ARC operations are each further divided into seven different types depending on the derivational operation involved and the location: Substitution 0-4, Adjoining, and Co-anchor attachment. Substitution n represents an instance of substitution into an argument slot of an elementary tree that is uniquely annotated with the number n (we discuss the interpretation of such numbers below). Adjoining represents an application of the adjoining operation. It is not further subdivided, as the current parser does not distinguish among different loci of adjoining within an elementary tree. Co-anchor attachment represents the substitution into a node that is construed as a cohead of an elementary tree. An example of this is the insertion of a particle into a verbally headed tree associated with a verb-particle construction, such as the insertion of *up* into the *pick*-headed tree to generate *I picked up the book*. The transitions terminate when the buffer is empty.

This system will fail to capture non-projective TAG derivation structures. However, as noted in Section 2, there is almost no non-projectivity in TAG derivation structures of English. Concretely, we find that WSJ Sections 01-22 contain only 26 non-projective sentences (0.065%), and those sentences are discarded during training. WSJ Section 00 does not have any non-projective sentences.

On the other hand, WSJ Sections 01-22 contain 0.6% of non-projective sentences in dependency grammar (Chen and Manning, 2014), an order of magnitude more than non-projectivity for TAG. This suggests that the problem of TAG parsing is more compatible with standard shift-reduce parsing than dependency grammar parsing is.[4]

4.2 Parser Model

In this work, we use a non-lexicalized parser, which does not have access to the identities of the words of the sentence to be parsed.[5] Instead, the parser's decisions will be guided by the supertags of the top k elements from the stack and the first k elements of the buffer. Using these features as input, we build a two-layer feed-forward network to predict the action for the parser to take. As noted above, the identity of the supertag does not allow the parser to encode whether a particular node in an elementary tree has already been targeted by a substitution operation. In order to overcome this deficiency, we augment the parser's state with *substitution memory*, which encodes for each possible substitution site (from 0 to 4) in a supertag T whether that substitution has already applied in T.

Each supertag is mapped to a d-dimensional vector by an embedding matrix $E \in \mathbb{R}^{d\times(N+2)}$ where N denotes the number of supertags; we also have additional vectors representing the empty state and $ROOT$. Substitution memory is similarly transformed, with a *substitution memory embedding matrix* $M \in \mathbb{R}^{d\times 5}$, to a d-dimensional vector that encodes in a distributed manner where substitution has applied. Each column in M is the vector corresponding to a specific substitution type. Each element from the stack and buffer is then represented by adding the supertag T embedding to the embedding associated with each vector from M corresponding to the substitution operations already performed on T, if any. Mathematically, suppose that we are at the configuration $c = (s, b, A)$, and $p^{(i)} \in \mathbb{R}^5$ denotes the substitution history of s_i. $p^{(i)}$ is an indicator vector that $p^{(i)}_j = 1$ if and only if we have already performed substitution j into s_i in the parser, and 0 otherwise. Define $p^{(k+1)}$ in the same way for b_1.[6] Then, the input vector to the network can be expressed as

$$[Es_1 + Mp^{(1)}; \cdots ; Es_k + Mp^{(k)};\\ Eb_1 + Mp^{(k+1)}; \cdots ; Eb_k]$$

This model with the additive substitution memory has several conceptual advantages. First, the additive structure gives us an unbounded scope of the past transition, avoiding making decisions that lead to substitution collisions without a computationally expensive architecture such as an ensemble of LSTMs (Xu, 2015). Moreover, as TAG supertags encode rich syntactic features, the parsing data for some supertags tend to become scarce. The most common 300 supertags in the Penn Tree Bank WSJ Sections 01-22 cover 96.8% of the data. In a situation of such data sparsity, it becomes crucial to link, for example, the behaviors of intransitive verbs with those of transitive verbs. With substitution memory, the network can develop representations under which addition of appropriate substitution vectors serves to transform

[4] We recognize alternatives to shift-reduce parsing. For instance, Dozat and Manning (2017) propose a graph-based parser that accommodates non-projectivity. It remains open whether such alternatives will work for TAG parsing, and we leave this for the future. We emphasize, however, that because of the nature of TAG derivations, the issue of non-projectivity is much less severe than dependency parsing.

[5] We have tried adding word embeddings as inputs to the parser with different choices of hyperparameters (e.g., the number of embedding dimensions). Unfortunately, our experiments yielded degraded performance. It should be noted, however, that TAG supertags typically provide enough information for deriving correct parses; the only cases that supertags cannot disambiguate are ambiguous attachments to identical nonterminals (e.g. *The picture of my friend with green eyes*).

[6] Notice that no substitution should have happened on $b_2, b_3, \ldots$ by construction.

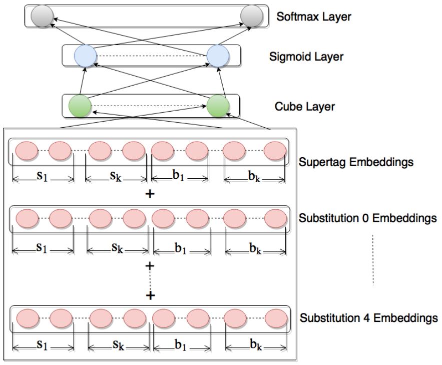


Figure 4: Shift-Reduce Parser Neural Network Architecture.

one supertag into another, allowing the generalization across these contexts. Indeed, as we will show in a later section, the substitution memory embeddings and supertag embeddings turn out to yield interpretable and linguistically sensible structures.

Finally, we concatenate the vectors associated with the relevant elements from the stack and buffer into a $2dk$ dimensional vector and feed it to the network to obtain a probability distribution over the possible transition actions. The architecture is visualized in Figure 4. Following Chen and Manning (2014), we use the cube activation function for the first layer, which could better capture interactions. We, again, optimize the negative log-likelihood in a mini-batch stochastic fashion with the Adam optimization algorithm with $l = 0.001$ (Kingma and Ba, 2015). With regards to decoding, we consider both greedy parsing as well as a beam search algorithm, where we keep transition action hypotheses at each time step, in the experiments we report below.

4.3 Supertag Input to the Parser

We consider three types of supertag inputs to the neural network parser: gold supertags, 1-best supertags from the BLSTM supertagger, and 1-best supertags from the MICA chart parser (Bangalore et al., 2009). MICA searches through n-best supertags with their corresponding probabilities and produces a full parse forest that abides by the TAG grammar. To generate the 1-best supertags from MICA, we first feed 10-best supertags from the BLSTM supertagger to the MICA chart parser, and retain only the supertags of the best parse. These supertags have the special property that there exists a feasible parse in the TAG grammar for every sentence, which does not necessarily hold for the 1-best supertags from the BLSTM supertagger.

5 Experiments

5.1 Experimental Setups

In order to ensure comparability with past work on TAG parsing, we follow the protocol of Bangalore et al. (2009) and Chung et al. (2016), and use the grammar and the TAG-annotated WSJ Penn Tree Bank described in Section 2. Following that work, we use Sections 01-22 as the training set, Section 00 as the development set, and Section 23 as the test set. The training, development, and test sets comprise 39832, 1921, and 2415 sentences, respectively. The development set contains 177 sentences with at least one supertag that was absent from the training set. We implement the networks in TensorFlow (Abadi et al., 2015). During training, we shuffle the order of the sentences in the training set to form mini-batches. Each mini-batch consists of 100 sentences, except the last which contains 32 sentences.

For supertagging, we first generate predicted POS tags for both the training set and the development set. The POS-tagger architecture is similar to that of the supertagger shown in Figure 3, except that, obviously, we do not feed it POS embeddings. The BLSTMs each contain 128 units, and we do not apply dropout at this stage. To derive predicted POS tags for the supertagger training set, we perform 10-fold jackknife training over the training set. For the supertagger, each direction of LSTM computation involves two layers, and each LSTM contains 512 units. The hidden units, layer-to-layer, and input units dropout rates are 0.5, 0.5, and 0.2 respectively. After each training epoch, we test the parser on the development set. When classification accuracy does not improve on two consecutive epochs, we end the training.

For the parser, we initialize the supertag embedding matrix E and the substitution memory embedding matrix M according to $Uniform(-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}})$. For all of the experiments reported here, we fix the hyper-parameters as follows: the embedding dimensions d for the supertag and substitution memory embeddings are 50, the number of units is 200 on both of the two hidden layers, and the input dropout rate is 0.2 and the hidden dropout rate is 0.3. We choose $k = 3$ or 5 for the stack/buffer scope. After each training epoch, we test the parser on the development set, and when the greedy accuracy fails to improving on two consecutive epochs, we terminate the training.

5.2 Supertagging Results

We achieve on Section 00 supertagging accuracy of 89.32%, 90.67% if we disregard the 177 sentences that contain an unseen supertag. This performance surpasses previous results on this task: Bangalore et al. (2009) report 88.52% accuracy using a maxent supertagger combined with a chart parser (MICA), which is the best result over a tag set of this complexity, though better results are reported for considerably smaller tag sets (on the order of 300 supertags). The n-best and β pruning accuracy (Clark and Curran, 2007) are given in Figure 5. In the β pruning scheme, we pick supertags whose probabilities are greater than β times the probability of the most likely supertag. We show the results for $\beta \in [0.075, 0.03, 0.01, 0.005, 0.001, 0.0001]$. It is noteworthy that with $\beta = 0.005$, the average number of supertags picked for each token (ambiguity level) is about 2, but the accuracy surpasses 98%, suggesting that incorporating the β pruning method in the *stapling* phase of TAG parsing will enhance the parser. We also obtain comparable accuracy of 89.44% on Section 23.

As discussed above, TAG supertags alone provide rich syntactic information. In order to understand how much such information our supertagger sucessfully captures, we analyze the 1-best supertag results on the basis of the syntactic properties of the elementary trees defined in Chung et al. (2016). Extending the notion of binary precision and recall, we define the macro-averaging precision and recall as the simple average over precision or recall corresponding to each class (Sokolova and Lapalme, 2009). We also compute accuracy, which is simply the ratio of correctly classified examples to the entire number of examples. Table 1 shows the results along with those for the maxent supertagger (Bangalore et al., 2009). Recall tends to be lower than precision; we can attribute this pattern to the nature of the macro-averaging scheme that equally treats each class regardless of the size; poor recall performance on a small class, such as the class of dative shift verbs, influences the overall recall as much as performance on a large class. Observe, however, that the BLSTM supertagger yields significantly better performance on recall in general, and it outperforms the maxent supertagger by a large margin in handling long dependencies of *wh*-movement and relativization.

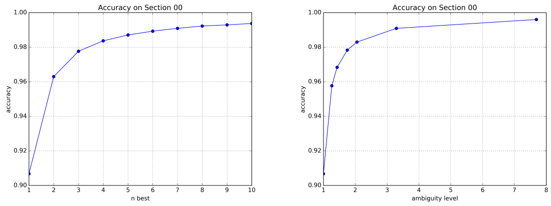


Figure 5: Section 00 n-best accuracy (left), and β pruning accuracy (right). Sentences with unseen supertags are disregarded.

		MICA			BLSTM		
Property	#	Prec	Rec	Acc	Pre.	Rec	Acc
root	42	**88.8**	68.3	95.4	80.4	**72.9**	**95.9**
coanc	4	**89.7**	64.0	**99.2**	65.3	**64.7**	**99.2**
modif	28	**82.2**	57.3	92.4	73.6	**63.6**	**93.7**
dir	3	95.9	95.9	96.0	**96.7**	**96.6**	**96.7**
predaux	2	80.0	67.6	**100.0**	**83.3**	**85.3**	**100.0**
pred	2	**93.3**	90.6	99.6	**93.3**	**93.1**	**99.7**
comp	3	92.9	61.4	99.7	**95.9**	**63.3**	**99.8**
particle	3	**94.0**	92.1	97.1	92.8	**92.5**	**97.5**
particleShift	3	**89.3**	**77.7**	**99.9**	77.5	77.0	**99.9**
voice	4	**94.6**	92.7	**99.4**	94.4	**94.7**	**99.4**
wh	4	93.0	79.6	97.1	**94.5**	**84.7**	**97.6**
rel	6	68.4	71.2	96.5	**88.9**	**73.6**	**97.2**
esubj	3	94.0	94.0	96.9	**95.4**	**95.3**	**97.4**
datshift	3	92.8	45.3	**99.9**	**96.9**	**53.3**	**99.9**

Table 1: 1-best Supertag Analysis on Section 00. # indicates the number of possible classes in a property. The Prec and Rec columns show macro-averaging precision and recall. The Acc columns indicate simple accuracy. For a complete description of the properties, see Chung et al. (2016).

Lastly, we interpret our supertagging performance in the context of prepositional phrase (PP) attachment ambiguity. Normally, in dependency parsing, PP attachment is resolved by the parser. However, in our case, it can be resolved before parsing, during the supertagging step. This is because the supertags for prepositions vary depending on the type of constituent modified by the PP containing the preposition; for example, $t4$ is the supertag for a preposition whose PP modifies an NP, while $t13$ is the supertag for a preposition whose PP modifies a VP.

To test how well our supertagger resolves PP attachment ambiguity, we used the dataset from Ratnaparkhi et al. (1994) (derived from the PTB WSJ) to extract a test set of sentences with PPs that are ambiguous between attaching to a VP or to an NP.[7]

[7] We were unable to use the full test set because, in order to run the supertagger on the test set, we had to map the test examples back to their full sentences, but some of those original sentences are no longer available in PTB3.

We then supertagged these sentences and checked whether the supertag for the preposition in the ambiguous PP is a VP modifier or an NP modifier. Of our test set of 1951 sentences, 1616 had supertags modifying the correct part of speech, to give an accuracy of 0.826. Table 2 compares this result to past work. The supertagger outperforms all other models besides the Word Vector model. Since this Word Vector model (like the MaxEnt model) is specifically trained for this task, and given that our supertagger is not trained for this particular task, the accuracy is reasonably encouraging. This result suggests that TAG supertagging is a reasonable intermediate level between only resolving PP attachment and conducting full parsing.

System	PP Attachment Accuracy
Malt (Nivre et al., 2006)	79.7*
MaxEnt (Ratnaparkhi et al., 1994)	81.6*
Word Vector (Belinkov et al., 2014)	**88.7***
Parsey McParseface (Andor et al., 2016)	82.3
BLSTM Supertagger	82.6

Table 2: Various PP attachment results. * denotes the results on a different dataset.

5.3 Parsing Results

Parsing results and comparison with prior models are summarized in Tables 3, 4 (Section 00), and 5 (Section 23). From Table 4, we see that the combination of the BLSTM supertagger, MICA chart parser, and the neural network parser achieves state-of-the-art performance, even compared to parsers that make use of lexical information, POS tags, and hand-engineered features. With gold supertags, the neural network parser with beam size 16 performs slightly better than the chart parser. As shown in Table 5, our supertag-based parser outperforms SyntaxNet (Andor et al., 2016) with the computationally expensive global normalization. This suggests that, besides providing the grammars and linguistic features that can be used in downstream tasks in addition to derivation trees (Semantic Role Labeling: (Chen and Rambow, 2003), Textual Entailments: (Xu et al., 2017)), supertagging also improves parsing performance.

5.4 Learned Vector Representation

We motivated the use of embeddings in the parser to encode properties of the supertags and the substitution operations performed on them. We can examine their structure in a way similar to what Mikolov et al. (2013) did for word embeddings by performing analogy tests on the learned supertag embeddings. Consider, for example, the analogy that an elementary tree representing a clause headed by a transitive verb (t27) is to a clause headed by an intransitive verb (t81) as a subject relative clause headed by a transitive verb (t99) is to a subject relative headed by an intransitive verb (t109). Following Mikolov et al. (2013), we can express this analogy with the equation $t27 - t81 \approx t99 - t109$, which can be rearranged as $t27 - t81 + t109 \approx t99$. By seeing if this approximate equality holds when the embeddings of the relevant supertags have been added and subtracted, we can test how well the embeddings capture syntactic properties of the supertags.

To create a set of such analogies, we extracted all pairs (stag1, stag2) such that stag2 is the result of excising exactly one substitution node from stag1. The idea here is that, once a substitution node is filled within a supertag, the result behaves like a supertag without that substitution node; for example, a transitive verb with its object filled behaves like an intransitive verb. We then create analogies by choosing two such pairs, (stag1, stag2) and (stag3, stag4), chosen so that stag1 and stag2 are related in the same way that stag3 and stag4 are related. From these two pairs we then form an equation of the form $\mathrm{stag1} - \mathrm{stag2} + \mathrm{stag4} \approx \mathrm{stag3}$.

We considered three different criteria for choosing which pairs of pairs can form analogies: **A-1**, where both pairs must have the same deep syntactic role (Drole) for the excised substitution node; **A-2**, where both pairs must have the same Drole and POS for the excised substitution node; and **A-3**, where both pairs must have the same Drole and same POS for the excised substitution node, and the heads of all supertags in the analogy must have the same POS. For each analogy generated, we computed the left hand side by adding and subtracting the relevant supertag embeddings and used cosine similarity to determine the most similar embeddings to the result and whether the intended right hand side was among the closest neighbors. We used four metrics for evaluation: *Acc*, the proportion of analogies for which the closest neighbor was the correct supertag; *Acc-300*, the proportion of analogies for which the closest neighbor amongst the 300 most common supertags was the correct supertag; *Avg Rank*, the average position of the correct choice in the ranked list of the closest neighbors; and *Avg Rank-300*,

		Gold Stags		BLSTM		BLSTM+Chart	
k	B	UAS	LAS	UAS	LAS	UAS	LAS
3	1	$96.74_{\pm 0.06}$	$96.47_{\pm 0.06}$	$89.54_{\pm 0.03}$	$88.06_{\pm 0.04}$	$90.03_{\pm 0.02}$	$88.56_{\pm 0.02}$
3	16	$97.62_{\pm 0.06}$	$97.42_{\pm 0.07}$	$90.31_{\pm 0.04}$	$88.85_{\pm 0.04}$	$90.85_{\pm 0.02}$	$89.38_{\pm 0.02}$
5	1	$96.96_{\pm 0.19}$	$96.67_{\pm 0.20}$	$89.63_{\pm 0.03}$	$88.12_{\pm 0.04}$	$90.07_{\pm 0.06}$	$88.60_{\pm 0.06}$
5	16	$\mathbf{97.68}_{\pm 0.06}$	$\mathbf{97.46}_{\pm 0.05}$	$\underline{90.38}_{\pm 0.05}$	$\underline{88.92}_{\pm 0.04}$	$\mathbf{90.88}_{\pm 0.06}$	$\mathbf{89.39}_{\pm 0.06}$

Table 3: Parsing Results on Section 00. k is # of elements from stack and buffer used as input, B is the beam size. We show mean and standard deviation over 5 trials with different initialization for each configuration. BLSTM+Chart shows results obtained by feeding the 1-best supertag inputs from the MICA chart parser discussed in Section 4.3.

		Gold Stags		Predicted Stags		
Parser	Features	UAS	LAS	Stag Acc	UAS	LAS
MALT-Stag	Words, POS, Stags (1-best)	97.20*	96.90*	88.52	88.50*	86.80*
Maxent+Chart (MICA)	Stags (10-best)	97.60	97.30	88.52	87.60	85.80
P3	Words, POS, Stags (1-best), Stag features	97.46*	96.51*	87.88	89.96*	87.86*
BLSTM+Chart	Stags (10-best)			**89.32**	90.05	88.32
BLSTM+NN	Stags (1-best)	$\mathbf{97.68}_{\pm 0.06}$	$\mathbf{97.46}_{\pm 0.05}$	**89.32**	$90.38_{\pm 0.05}$	$88.92_{\pm 0.04}$
BLSTM+Chart+NN	Stags (1-best)	–	–	89.31	$\mathbf{90.88}_{\pm 0.06}$	$\mathbf{89.39}_{\pm 0.06}$

Table 4: Section 00 Performance Comparison with Prior Models. The P3 results are from Chung et al. (2016). P3 is based on the model described in Nivre et al. (2004). * denotes the results with gold POS tags. For the NN parser, k=5 and B=16.

Model	Stag Acc	UAS	LAS
SyntaxNet	–	$90.47_{\pm 0.05}$	$88.99_{\pm 0.06}$
Maxent+Chart	86.85	86.66	84.90
BLSTM+Chart	89.44	90.20	88.66
BLSTM+NN	89.44	$90.31_{\pm 0.03}$	$88.98_{\pm 0.03}$
BLSTM+Chart+NN	**89.71**	$\mathbf{90.97}_{\pm 0.03}$	$\mathbf{89.68}_{\pm 0.03}$

Table 5: Supertagging and Parsing Results on Section 23. For the NN parser, k=5 and B=16 throughout. We trained Syntaxnet (Andor et al., 2016) with global normalization beam size 16 using the TensorFlow toolkit.

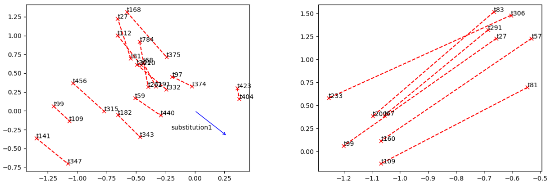
Type	Acc	Acc-300	Avg Rank	Avg Rank-300
A-1	0.20	0.28	49.5	10.2
A-2	0.44	0.60	17.4	3.68
A-3	0.61	**0.81**	2.26	**1.38**

Table 6: Analogy Task Results.

the average position of the correct choice in the ranked list of the closest neighbors amongst the 300 most common supertags.

We expect that the embeddings for common supertags would be better representations than embeddings for rare supertags. Thus, we restricted our experiment to analogies between supertags among the 300 most common ones in the training set. (Indeed, experiments that included rare supertags in the analogies produced poor results.)

Table 6 provides the results for the 3 types of analogies, which are very promising, particularly type A-3. We can visualize these results by performing PCA on the embedding vectors. Figure 6a shows the first 2 PCA components of A-3 analogies involving supertags containing transitive and intransitive predicates across a variety of structures. We see that virtually all pairs differ from one another by a similar vector, and in fact this difference is essentially the vector associated with substitution 1 in the substitution embedding memory (shown in blue). Figure 6b shows the case of pairs of canonical sentence elementary trees (*read* in *I read the book*) and their subject relative analogs (*read* in *the guy who read the book*). This again shows a systematic mapping between grammatically related embeddings, suggesting that the embeddings encode relevant structural properties.



(a) Transitive/intransitive (b) Declarative/relative

Figure 6: Embedding vector alignments.

6 Conclusions and Future Work

We presented a state-of-the-art TAG supertagger and parser, the former based on a BLSTM architecture, and the latter on a non-lexicalized shift-reduce parser using a feed-forward network. The parser makes crucial use of supertag embeddings that provide linguistically interpretable vector representations of the supertags. These positive results suggest that TAG can provide the foundation of NLP systems for tasks requiring deeper analysis than current dependency parsers provide, and we will apply our parser to such tasks in the future. Nonetheless, a large discrepancy remains in parser performance with gold supertags and predicted supertags, indicating that supertagging is still a bottleneck. We will explore ways to leverage our supertagger's high β-pruning accuracy in parsing.

References

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

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of Association for Computational Linguistics*.

Srinivas Bangalore, Pierre Boullier, Alexis Nasr, Owen Rambow, and Benoît Sagot. 2009. MICA: A Probabilistic Dependency Parser Based on Tree Insertion Grammars. In *NAACL HLT 2009 (Short Papers)*.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Computational Linguistics* 25:237–266.

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*. Saarbrücken, Germany.

Yonatan Belinkov, Tao Lei, Regina Barzilay, and Amir Globerson. 2014. Exploring compositional architectures and word vector representations for prepositional phrase attachment. *Transactions of the Association for Computational Linguistics* (2):561–572.

Danqi Chen and Christopher D Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

John Chen. 2001. *Towards Efficient Statistical Parsing Using Lexicalized Grammatical Information*. Ph.D. thesis, University of Delaware.

John Chen and Owen Rambow. 2003. Use of Deep Linguistics Features for the Recognition and Labeling of Semantic Arguments. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Wonchang Chung, Suhas Siddhesh Mhatre, Alexis Nasr, Owen Rambow, and Srinivas Bangalore. 2016. Revisiting supertagging and parsing: How to use supertags in transition-based parsing. In *Proceedings of the 12th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+12)*. pages 85–92.

Stephen Clark and James R. Curran. 2007. Wide-coverage semantic representations from a CCG parser. *Computational Linguistics* 4.

Timothy Dozat and Christopher Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR*.

Robert Frank. 2001. *Phrase Structure Composition and Syntactic Dependencies*. MIT Press, Cambridge, Mass.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, Curran Associates, Inc., pages 1019–1027. http://papers.nips.cc/paper/6241-a-theoretically-grounded-application-of-dropout-in-recurrent-neural-networks.pdf.

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

Julia Hockenmaier and Mark Steedman. 2007. CCG-bank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics* 33(3):355–396.

Aravind K. Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Volume 3: Beyond Words*, Springer, New York, pages 69–124.

Diederik P. Kingma and Jimmy Lei Ba. 2015. ADAM: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint a* CCG parsing and semantic role labeling". In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG Parsing. In *Proceedings of NAACL-HLT 2016*. pages 221–231.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 3111–3119. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Maria Nadejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch. 2017. Syntax-aware neural machine translation using CCG. ArXiv Preprint 1702.01147v1.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*. pages 49–56.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *LREC*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1532–1543.

Owen Rambow. 2010. The simple truth about dependency and phrase structure representations: An opinion piece. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, pages 337–340. http://www.aclweb.org/anthology/N10-1049.

Owen Rambow and Aravind Joshi. 1997. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In Leo Wanner, editor, *Recent Trends in Meaning-Text Theory*, John Benjamins, Amsterdam and Philadelphia, pages 167–190.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *ARPA Human Language Technology Workshop*. pages 250–255.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics* 4:127–140.

Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing and Management* 45:427–437.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research 15* (15):1929–1958.

Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. In Robert Borsley and Kersti Börjars, editors, *Non-Transformational Syntax: Formal and Explicit Models of Grammar*, Wiley-Blackwell.

Pauli Xu, Robert Frank, Jungo Kasai, and Owen Rambow. 2017. TAG parsing evaluation using textual entailments. In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+13)*.

Wenduan Xu. 2015. LSTM shift-reduce CCG parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1754–1764.

Yue Zhang and Stephen Clark. 2011. Shift-Reduce CCG Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 683–692.

Global Normalization of Convolutional Neural Networks for Joint Entity and Relation Classification

Heike Adel and **Hinrich Schütze**
Center for Information and Language Processing (CIS)
LMU Munich, Germany
heike@cis.lmu.de

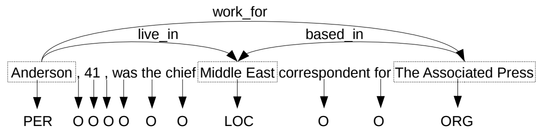
Abstract

We introduce globally normalized convolutional neural networks for joint entity classification and relation extraction. In particular, we propose a way to utilize a linear-chain conditional random field output layer for predicting entity types and relations between entities at the same time. Our experiments show that global normalization outperforms a locally normalized softmax layer on a benchmark dataset.

1 Introduction

Named entity classification (EC) and relation extraction (RE) are important topics in natural language processing. They are relevant, e.g., for populating knowledge bases or answering questions from text, such as "Where does X live?"

Most approaches consider the two tasks independent from each other or treat them as a sequential pipeline by first applying a named entity recognition tool and then classifying relations between entity pairs. However, named entity types and relations are often mutually dependent. If the types of entities are known, the search space of possible relations between them can be reduced and vice versa. This can help, for example, to resolve ambiguities, such as in the case of "Mercedes", which can be a person, organization and location. However, knowing that in the given context, it is the second argument for the relation "live_in" helps concluding that it is a location. Therefore, we propose a single neural network (NN) for both tasks. In contrast to joint training and multitask learning, which calculate task-wise costs, we propose to learn a *joint classification layer* which is *globally normalized* on the outputs of both tasks. In particular, we train the NN parameters based on the loss of a linear-chain conditional random field (CRF) (Lafferty et al., 2001). CRF layers for NNs have been introduced for token-labeling tasks like named entity recognition (NER) or part-of-speech tagging (Collobert et al., 2011; Lample et al., 2016; Andor et al., 2016). Instead of labeling each input token as in previous work, we model the joint entity and relation classification problem as a sequence of length three for the CRF layer. In particular, we identify the types of two candidate entities (words or short phrases) given a sentence (we call this entity classification to distinguish it from the token-labeling task NER) as well as the relation between them. To the best of our knowledge, this architecture for combining entity and relation classification in a single neural network is novel. Figure 1 shows an example of how we model the task: For each sentence, candidate entities are identified. Every possible combination of candidate entities (query entity pair) then forms the input to our model which predicts the classes for the two query entities as well as for the relation between them.



Model inputs (query entity pairs) ⇒ Model outputs:
("Anderson", ",") ⇒ PER - N - O
("Anderson", "41") ⇒ PER - N - O
...
("Anderson", "chief") ⇒ PER - N - O
("Anderson", "Middle East") ⇒ PER - live_in - LOC
...
("was", "for") ⇒ O - N - O
...
("for", "The Associated Press") ⇒ O - N - ORG

Figure 1: Examples of our task

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1723–1729
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

To sum up, our contributions are as follows: We introduce globally normalized convolutional neural networks for a sentence classification task. In particular, we present an architecture which allows us to model joint entity and relation classification with a single neural network and classify entities and relations at the same time, normalizing their scores globally. Our experiments confirm that a CNN with a CRF output layer outperforms a CNN with locally normalized softmax layers. Our source code is available at `http://cistern.cis.lmu.de`.

2 Related Work

Some work on joint entity and relation classification uses distant supervision for building their own datasets, e.g., (Yao et al., 2010; Yaghoobzadeh et al., 2016). Other studies, which are described in more detail in the following, use the "entity and relation recognition" (ERR) dataset from (Roth and Yih, 2004, 2007) as we do in this paper. Roth and Yih (2004) develop constraints and use linear programming to globally normalize entity types and relations. Giuliano et al. (2007) use entity type information for relation extraction but do not train both tasks jointly. Kate and Mooney (2010) train task-specific support vector machines and develop a card-pyramid parsing algorithm to jointly model both tasks. Miwa and Sasaki (2014) use the same dataset but model the tasks as a table filling problem (see Section 4.2). Their model uses both a local and a global scoring function. Recently, Gupta et al. (2016) apply recurrent neural networks to fill the table. They train them in a multitask fashion. Previous work also uses a variety of linguistic features, such as part-of-speech tags. In contrast, we use convolutional neural networks and only word embeddings as input. Furthermore, we are the first to adopt global normalization of neural networks for this task.

Several studies propose different variants of non-neural CRF models for information extraction tasks but model them as token-labeling problems (Sutton and McCallum, 2006; Sarawagi et al., 2004; Culotta et al., 2006; Zhu et al., 2005; Peng and McCallum, 2006). In contrast, we propose a simpler linear-chain CRF model which directly connects entity and relation classes instead of assigning a label to each token of the input sequence. This is more similar to the factor graph by Yao et al. (2010) but computationally simpler. Xu and Sarikaya (2013) also apply a CRF layer on top of continuous representations obtained by a CNN. However, they use it for a token labeling task (semantic slot filling) while we apply the model to a sentence classification task, motivated by the fact that a CNN creates single representations for whole phrases or sentences.

3 Model

3.1 Modeling Context and Entities

Figure 2 illustrates our model.

Input. Given an input sentence and two query entities, our model identifies the *types of the entities* and the *relation* between them; see Figure 1. The input tokens are represented by word embeddings trained on Wikipedia with word2vec (Mikolov et al., 2013). For identifying the class of an entity e_k, the model uses the context to its left, the words constituting e_k and the context to its right. For classifying the relation between two entities e_i and e_j, the sentence is split into six parts: left of e_i, e_i, right of e_i, left of e_j, e_j, right of e_j.[1] For the example sentence in Figure 1 and the entity pair ("Anderson", "chief"), the context split is: [] [Anderson] [, 41 , was the chief Middle ...] [Anderson , 41 , was the] [chief] [Middle East correspondent for ...]

Sentence Representation. For representing the different parts of the input sentence, we use convolutional neural networks (CNNs). CNNs are suitable for RE since a relation is usually expressed by the semantics of a whole phrase or sentence. Moreover, they have proven effective for RE in previous work (Vu et al., 2016). We train one CNN layer for convolving the entities and one for the contexts. Using two CNN layers instead of one gives our model more flexibility. Since entities are usually shorter than contexts, the filter width for entities can be smaller than for contexts. Furthermore, this architecture simplifies changing the entity representation from words to characters in future work.

After convolution, we apply k-max pooling for both the entities and the contexts and concatenate the results. The concatenated vector $c_z \in \mathbb{R}^{C_z}$, $z \in \{EC, RE\}$ is forwarded to a task-specific hidden layer of size H_z which learns patterns across the different input parts:

$$h_z = \tanh(V_z^T c_z + b_z) \qquad (1)$$

[1] The ERR dataset we use provides boundaries for entities to concentrate on the classification task (Roth and Yih, 2004).

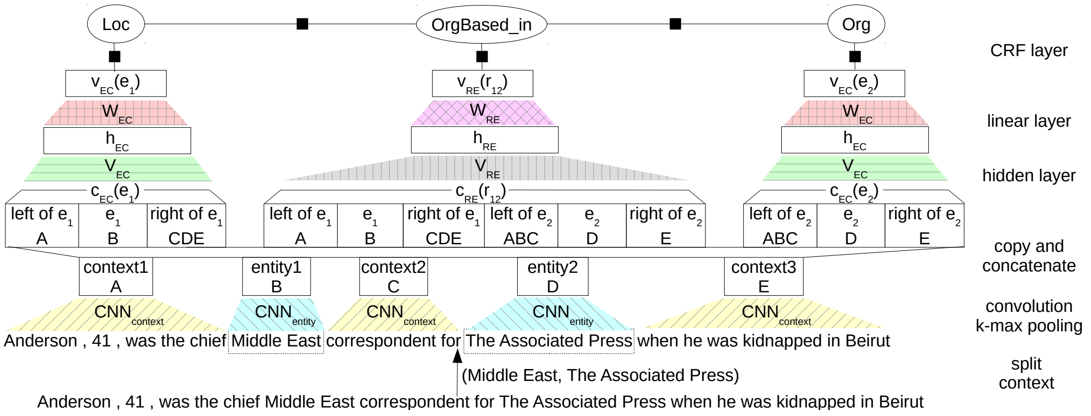


Figure 2: Model overview; the colors/shades show which model parts share parameters

with weights $V_z \in \mathbb{R}^{C_z \times H_z}$ and bias $b_z \in \mathbb{R}^{H_z}$.

3.2 Global Normalization Layer

For global normalization, we adopt the linear-chain CRF layer by Lampple et al. (2016).[2] It expects scores for the different classes as input. Therefore, we apply a linear layer first which maps the representations $h_z \in \mathbb{R}^{H_z}$ to a vector v_z of the size of the output classes $N = N_{EC} + N_{RE}$:

$$v_z = W_z^T h_z \tag{2}$$

with $W_z \in \mathbb{R}^{H_z \times N}$. For a sentence classification task, the input sequence for the CRF layer is not inherentely clear. Therefore, we propose to model the joint entity and relation classification problem with the following sequence of scores (cf., Figure 2):

$$d = [v_{EC}(e_1), v_{RE}(r_{12}), v_{EC}(e_2)] \tag{3}$$

with r_{ij} being the relation between e_i und e_j. Thus, we approximate the joint probability of entity types T_{e_1}, T_{e_2} and relations $R_{e_1 e_2}$ as follows:

$$\begin{aligned} &P(T_{e_1} R_{e_1 e_2} T_{e_2}) \\ \approx &P(T_{e_1}) \cdot P(R_{e_1 e_2} | T_{e_1}) \cdot P(T_{e_2} | R_{e_1 e_2}) \end{aligned} \tag{4}$$

Our intuition is that the dependence between relation and entities is stronger than the dependence between the two entities.

The CRF layer pads its input of length $n = 3$ with begin and end tags and computes the following score for a sequence of predictions y:

$$s(y) = \sum_{i=0}^{n} Q_{y_i y_{i+1}} + \sum_{i=1}^{n} d_{i,y_i} \tag{5}$$

with $Q_{k,l}$ being the transition score from class k to class l and $d_{p,q}$ being the score of class q at position p in the sequence. The scores are summed because all the variables of the CRF layer live in the log space. The matrix of transition scores $Q \in \mathbb{R}^{(n+2)\times(n+2)}$ is learned during training.[3] For training, the forward algorithm computes the scores for all possible label sequences Y to get the log-probability of the correct label sequence $\hat{y}$:

$$log(p(\hat{y})) = \frac{e^{s(\hat{y})}}{\sum_{\tilde{y} \in Y} e^{s(\tilde{y})}} \tag{6}$$

For testing, Viterbi is applied to obtain the label sequence y^* with the maximum score:

$$y^* = \arg\max_{\tilde{y} \in Y} s(\tilde{y}) \tag{7}$$

4 Experiments and Analysis

4.1 Data and Evaluation Measure

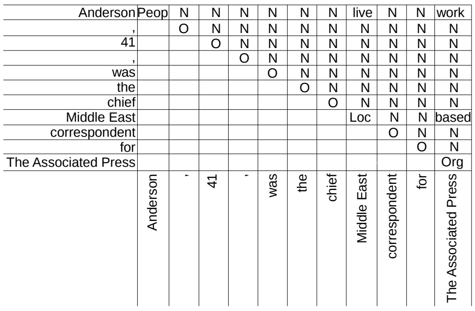
We use the "entity and relation recognition" (ERR) dataset from (Roth and Yih, 2004)[4] with the train-test split by Gupta et al. (2016). We tune the parameters on a held-out part of train. The data is labeled with entity types and relations (see Table 1). For entity pairs without a relation, we use the label N. Dataset statistics and model parameters are provided in the appendix.

Following previous work, we compute F_1 of the individual classes for EC and RE, as well as a task-wise macro F_1 score. We also report the average of scores across tasks (Avg EC+RE).

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

[3] 2 is added because of the padded begin and end tag

[4] http://cogcomp.cs.illinois.edu/page/resource_view/43



Anderson	Peop	N	N	N	N	N	N	live	N	N	work
,		O	N	N	N	N	N	N	N	N	N
41			O	N	N	N	N	N	N	N	N
,				O	N	N	N	N	N	N	N
was					O	N	N	N	N	N	N
the						O	N	N	N	N	N
chief							O	N	N	N	N
Middle East								Loc	N	N	based
correspondent									O	N	N
for										O	N
The Associated Press											Org
	Anderson	,	41	,	was	the	chief	Middle East	correspondent	for	The Associated Press

Figure 3: Entity-relation table

4.2 Experimental Setups

Setup 1: Entity Pair Relations. Roth and Yih (2004, 2007); Kate and Mooney (2010) train separate models for EC and RE on the ERR dataset. For RE, they only identify relations between named entity pairs. In this setup, the query entities for our model are only named entity pairs. Note that this facilitates EC in our experiments.

Setup 2: Table Filling. Following Miwa and Sasaki (2014); Gupta et al. (2016), we also model the joint task of EC and RE as a table filling task. For a sentence with length m, we create a quadratic table. Cell (i, j) contains the relation between word i and word j (or N for no relation). A diagonal cell (k, k) contains the entity type of word k. Following previous work, we only predict classes for half of the table, i.e. for $m(m + 1)/2$ cells. Figure 3 shows the table for the example sentence from Figure 1. In this setup, each cell (i, j) with $i \neq j$ is a separate input query to our model. Our model outputs a prediction for cell (i, j) (the relation between i and j) and predictions for cells (i, i) and (j, j) (the types of i and j). To fill the diagonal with entity classes, we aggregate all predictions for the particular entity by using majority vote. Section 4.4 shows that the individual predictions agree with the majority vote in almost all cases.

Setup 3: Table Filling Without Entity Boundaries. The table from setup 2 includes one row/column per multi-token entity, utilizing the given entity boundaries of the ERR dataset. In order to investigate the impact of the entity boundaries on the classification results, we also consider another table filling setup where we ignore the boundaries and assign one row/column per token. Note that this setup is also used by prior work on table filling (Miwa and Sasaki, 2014; Gupta et al., 2016). For evaluation, we follow Gupta et al. (2016) and score a multi-token entity as correct if at least one of its comprising cells has been classified correctly.

Comparison. The most important difference between setup 1 and setup 2 is the number of entity pairs with no relation (test set: ≈3k for setup 1, ≈121k for setup 2). This makes setup 2 more challenging. The same holds for setup 3 which considers the same number of entity pairs with no relation as setup 2. To cope with this, we randomly subsample negative instances in the train set of setup 2 and 3. Setup 3 considers the most query entity pairs in total since multi-token entities are split into their comprising tokens. However, setup 3 represents a more realistic scenario than setup 1 or setup 2 because in most cases, entity boundaries are not given. In order to apply setup 1 or 2 to another dataset without entity boundaries, a preprocessing step, such as entity boundary recognition or chunking would be required.

4.3 Experimental Results

Table 1 shows the results of our globally normalized model in comparison to the same model with locally normalized softmax output layers (one for EC and one for RE). For setup 1, the CRF layer performs comparable or better than the softmax layer. For setup 2 and 3, the improvements are more apparent. We assume that the model can benefit more from global normalization in the case of table filling because it is the more challenging setup. The comparison between setup 2 and setup 3 shows that the entity classification suffers from not given entity boundaries (in setup 3). A reason could be that the model cannot convolve the token embeddings of the multi-token entities anymore when computing the entity representation (context B and D in Figure 2). Nevertheless, the relation classification performance is comparable in setup 2 and setup 3. This shows that the model can internally account for potentially wrong entity classification results due to missing entity boundaries.

The overall results (Avg EC+RE) of the CRF are better than the results of the softmax layer for all three setups. To sum up, the improvements of the linear-chain CRF show that (i) joint EC and RE benefits from global normalization and (ii) our way of creating the input sequence for the CRF for joint EC and RE is effective.

Comparison to State of the Art. Table 2 shows our results in the context of state-of-the-art results: (Roth and Yih, 2007), (Kate and Mooney, 2010),

	Setup 1		Setup 2		Setup 3	
	softmax	CRF	softmax	CRF	softmax	CRF
Peop	**95.24**	94.95	93.99	**94.47**	91.46	**92.21**
Org	**88.94**	87.56	78.95	**79.37**	67.29	**67.91**
Loc	93.25	**93.63**	90.69	**90.80**	85.99	**86.20**
Other	**90.38**	89.54	73.78	**73.97**	**62.67**	61.19
Avg EC	**91.95**	91.42	84.35	**84.65**	76.85	**76.88**
Located_in	55.03	**57.72**	51.03	**55.13**	44.96	**52.29**
Work_for	**71.23**	70.67	52.89	**61.42**	52.63	**65.31**
OrgBased_in	53.25	**59.38**	56.96	**59.12**	46.15	**57.65**
Live_in	**59.57**	58.94	**64.29**	60.12	**64.09**	61.45
Kill	74.70	**79.55**	69.14	**74.73**	**82.93**	75.86
Avg RE	62.76	**65.25**	58.86	**62.10**	58.15	**62.51**
Avg EC+RE	77.36	**78.33**	71.61	**73.38**	67.50	**69.69**

Table 1: F_1 results for entity classification (EC) and relation extraction (RE) in the three setups

Model	S	Feats	EC	RE	EC+RE
R & Y 2007	1	yes	85.8	58.1	72.0
K & M 2010	1	yes	91.7	62.2	77.0
Ours (NN CRF)	1	no	**92.1**	**65.3**	**78.7**
Ours (NN CRF)	2	no	**88.2**	**62.1**	**75.2**
M & S 2014	3	yes	92.3	**71.0**	**81.7**
G et al. 2016 (1)	3	yes	**92.4**	69.9	81.2
G et al. 2016 (2)	3	no	**88.8**	58.3	**73.6**
Ours (NN CRF)	3	no	82.1	**62.5**	72.3

Table 2: Comparison to state of the art (S: setup)

(Miwa and Sasaki, 2014), (Gupta et al., 2016).[5] Note that the results are not comparable because of the different setups and different train-test splits.[6]

Our results are best comparable with (Gupta et al., 2016) since we use the same setup and train-test splits. However, their model is more complicated with a lot of hand-crafted features and various iterations of modeling dependencies among entity and relation classes. In contrast, we only use pre-trained word embeddings and train our model end-to-end with only one iteration per entity pair. When we compare with their model without additional features (G et al. 2016 (2)), our model performs worse for EC but better for RE and comparable for Avg EC+RE.

4.4 Analysis of Entity Type Aggregation

As described in Section 4.2, we aggregate the EC results by majority vote. Now, we analyze their disagreement. For our best model, there are only 9 entities (0.12%) with disagreement in the test data. For those, the max, min and median disagreement with the majority label is 36%, 2%, and 8%, resp. Thus, the disagreement is negligibly small.

	N	Based_in	Live_in	Kill	Located_in	Work_for
O	✗					
Other	✗					
Peop			✗	✗		✗
Org		✗				✗
Loc		✗	✗		✗	

Figure 4: Most strongly correlated entity types and relations according to CRF transition matrix

4.5 Analysis of CRF Transition Matrix

To analyze the CRF layer, we extract which transitions have scores above 0.5. Figure 4 shows that the layer has learned correct correlations between entity types and relations.

5 Conclusion and Future Work

In this paper, we presented the first study on global normalization of neural networks for a sentence classification task without transforming it into a token-labeling problem. We trained a convolutional neural network with a linear-chain conditional random field output layer on joint entity and relation classification and showed that it outperformed using a locally normalized softmax layer.

An interesting future direction is the extension of the linear-chain CRF to jointly normalize all predictions for table filling in a single model pass. Furthermore, we plan to verify our results on other datasets in future work.

Acknowledgments

Heike Adel is a recipient of the Google European Doctoral Fellowship in Natural Language Processing and this research is supported by this fellowship. This work was also supported by DFG (SCHU 2246/4-2).

[5] We only show results of single models, no ensembles. Following previous studies, we omit the entity class "Other" when computing the EC score.

[6] Our results on EC in setup 1 are also not comparable since we only input named entities into our model.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

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

Aron Culotta, Andrew McCallum, and Jonathan Betz. 2006. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 296–303, New York City, USA. Association for Computational Linguistics.

Claudio Giuliano, Alberto Lavelli, and Lorenza Romano. 2007. Relation extraction and the influence of automatic named-entity recognition. *ACM Trans. Speech Lang. Process.*, 5(1):2:1–2:26.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2537–2547, Osaka, Japan. The COLING 2016 Organizing Committee.

Rohit J. Kate and Raymond Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 203–212, Uppsala, Sweden. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at 1st International Conference on Learning Representations (ICLR)*, Scottsdale, Arizona, USA.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, Doha, Qatar. Association for Computational Linguistics.

Fuchun Peng and Andrew McCallum. 2006. Information extraction from research papers using conditional random fields. *Information processing & management*, 42(4):963–979.

D. Roth and W. Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. In *Introduction to Statistical Relational Learning*. MIT Press.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 1–8, Boston, Massachusetts, USA. Association for Computational Linguistics.

Sunita Sarawagi, William W Cohen, et al. 2004. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, volume 17, pages 1185–1192.

Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, pages 93–128.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 534–539, San Diego, California. Association for Computational Linguistics.

Puyang Xu and Ruhi Sarikaya. 2013. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 78–83, Olomouc, Czech Republic. IEEE.

Yadollah Yaghoobzadeh, Heike Adel, and Hinrich Schütze. 2016. Noise mitigation for neural entity typing and relation extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, Valencia, Spain. Association for Computational Linguistics.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1013–1023, Cambridge, MA. Association for Computational Linguistics.

Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. 2005. 2d conditional random fields for web information extraction. In *Proceedings of the 22nd international conference on Machine learning*, pages 1044–1051. ACM.

A Dataset Statistics

Table 3 provides statistics of the data composition in our different setups which are described in the paper. The N class of setup 2 and setup 3 has been subsampled in the training and development set as described in the paper.

	train	dev	test
Peop	1146	224	321
Org	596	189	198
Loc	1204	335	427
Other	427	110	125
O	20338	5261	6313
Located_in	243	66	94
Work_for	243	82	76
OrgBased_in	239	106	105
Live_in	342	79	100
Kill	203	18	47
N (setup 1)	10742	2614	3344
N (setup 2/3)	123453	30757	120716

Table 3: Dataset statistics for our different experimental setups

Note that the sum of numbers of relation labels is slightly different to the numbers reported in (Roth and Yih, 2004). According to their website https://cogcomp.cs.illinois.edu/page/resource_view/43, they have updated the corpus.

B Hyperparameters

Setup	Output layer	nk_C	nk_E	h_C	h_E
1	softmax	500	100	100	50
2	softmax	500	100	100	50
3	softmax	500	100	100	50
1	CRF	200	50	100	50
2	CRF	500	100	200	50
3	CRF	500	100	100	50

Table 4: Hyperparameter optimization results

Table 4 provides the hyperparameters we optimized on dev (nk_C: number of convolutional filters for the CNN convolving the contexts, nk_E: number of convolutional filters for the CNN convolving the entities; h_C: number of hidden units for creating the final context representation, h_E: number of hidden units for creating the final entity representation).

For all models, we use a filter width of 3 for the context CNN and a filter width of 2 for the entity CNN (tuned in prior experiments and fixed for the optimization of the parameters in Table 4).

For training, we apply gradient descent with a batch size of 10 and an initial learning rate of 0.1. When the performance on dev decreases, we halve the learning rate. The model is trained with early stopping on dev, with a maximum number of 20 epochs. We apply L2 regularization with $\lambda = 10^{-3}$.

End-to-End Neural Relation Extraction with Global Optimization

Meishan Zhang[1] and **Yue Zhang**[2] and **Guohong Fu**[1]
1. School of Computer Science and Technology, Heilongjiang University, China
2. Singapore University of Technology and Design
mason.zms@gmail.com,
yue_zhang@sutd.edu.sg,
ghfu@hotmail.com

Abstract

Neural networks have shown promising results for relation extraction. State-of-the-art models cast the task as an end-to-end problem, solved incrementally using a local classifier. Yet previous work using statistical models have demonstrated that global optimization can achieve better performances compared to local classification. We build a globally optimized neural model for end-to-end relation extraction, proposing novel LSTM features in order to better learn context representations. In addition, we present a novel method to integrate syntactic information to facilitate global learning, yet requiring little background on syntactic grammars thus being easy to extend. Experimental results show that our proposed model is highly effective, achieving the best performances on two standard benchmarks.

1 Introduction

Extracting entities (Florian et al., 2006, 2010) and relations (Zhao and Grishman, 2005; Jiang and Zhai, 2007; Sun et al., 2011; Plank and Moschitti, 2013) from unstructured texts have been two central tasks in information extraction (Grishman, 1997; Doddington et al., 2004). Traditional approaches to relation extraction take entity recognition as a predecessor step in a pipeline (Zelenko et al., 2003; Chan and Roth, 2011), predicting relations between given entities.

In recent years, there has been a surge of interest in performing end-to-end relation extraction, jointly recognizing entities and relations given free text inputs (Li and Ji, 2014; Miwa and Sasaki, 2014; Miwa and Bansal, 2016; Gupta et al., 2016). End-to-end learning prevents error propagation in the pipeline approach, and allows cross-task dependencies to be modeled explicitly for entity recognition. As a result, it gives better relation extraction accuracies compared to pipelines.

Miwa and Bansal (2016) were among the first to use neural networks for end-to-end relation extraction, showing highly promising results. In particular, they used bidirectional LSTM (Graves et al., 2013) to learn hidden word representations under a sentential context, and further leveraged tree-structured LSTM (Tai et al., 2015) to encode syntactic information, given the output of a parser. The resulting representations are then used for making local decisions for entity and relation extraction incrementally, leading to much improved results compared with the best statistical model (Li and Ji, 2014). This demonstrates the strength of neural representation learning for end-to-end relation extraction.

On the other hand, Miwa and Bansal (2016)'s model is trained locally, without considering structural correspondences between incremental decisions. This is unlike existing statistical methods, which utilize well-studied structured prediction methods to address the problem (Li and Ji, 2014; Miwa and Sasaki, 2014). As has been commonly understood, learning local decisions for structured prediction can lead to label bias (Lafferty et al., 2001), which prevents globally optimal structures from receiving optimal scores by the model. We address this potential issue by building a structural neural model for end-to-end relation extraction, following a recent line of efforts on globally optimized models for neural structured prediction (Zhou et al., 2015; Watanabe and Sumita, 2015; Andor et al., 2016; Wiseman and Rush, 2016).

In particular, we follow Miwa and Sasaki (2014), casting the task as an end-to-end table-filling problem. This is different from the action-based method of Li and Ji (2014), yet has shown to

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1730–1740
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

be more flexible and accurate (Miwa and Sasaki, 2014). We take a different approach to representation learning, addressing two potential limitations of Miwa and Bansal (2016).

First, Miwa and Bansal (2016) rely on external syntactic parsers for obtaining syntactic information, which is crucial for relation extraction (Culotta and Sorensen, 2004; Zhou et al., 2005; Bunescu and Mooney, 2005; Qian et al., 2008). However, parsing errors can lead to encoding inaccuracies of tree-LSTMs, thereby hurting relation extraction potentially. We take an alternative approach to integrating syntactic information, by taking the hidden LSTM layers of a bi-affine attention parser (Dozat and Manning, 2016) to augment input representations. Pretrained for parsing, such hidden layers contain rich syntactic information on each word, yet do not explicitly represent parsing decisions, thereby avoiding potential issues caused by incorrect parses.

Our method is also free from a particular syntactic formalism, such as dependency grammar, constituent grammar or combinatory categorial grammar, requiring only hidden representations on word that contain syntactic information. In contrast, the method of Miwa and Bansal (2016) must consider tree LSTM formulations that are specific to grammar formalisms, which can be structurally different (Tai et al., 2015).

Second, Miwa and Bansal (2016) did not explicitly learn the representation of segments when predicting entity boundaries or making relation classification decisions, which can be intuitively highly useful, and has been investigated in several studies (Wang and Chang, 2016; Zhang et al., 2016). We take the LSTM-Minus method of Wang and Chang (2016), modelling a segment as the difference between its last and first LSTM hidden vectors. This method is highly efficient, yet gives as accurate results as compared to more complex neural network structures to model a span of words (Cross and Huang, 2016).

Evaluation on two benchmark datasets shows that our method outperforms previous methods of Miwa and Bansal (2016), Li and Ji (2014) and Miwa and Sasaki (2014), giving the best reported results on both benchmarks. Detailed analysis shows that our integration of syntactic features is as effective as traditional approaches based on discrete parser outputs. We make our code publicly available under Apache License 2.0.[1]

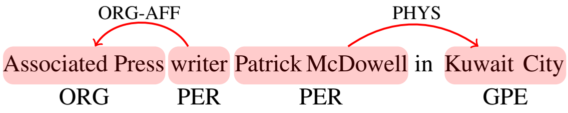


Figure 1: Relation extraction. The example is chosen from the ACE05 dataset, where ORG, PER and GPE denote organization, person and geo-political entities, respectively; ORG-AFF and PHYS denote organization affiliation and physical relations, respectively.

2 Model

2.1 Task Definition

As shown in Figure 1, the goal of relation extraction is to mine relations from raw texts. It consists of two sub-tasks, namely entity detection, which recognizes valid entities, and relation classification, which determines the relation categories over entity pairs. We follow recent studies and recognize entities and relations as one single task.

2.2 Method

We follow Miwa and Sasaki (2014) and Gupta et al. (2016), treating relation extraction as a table-filling problem, performing entity detection and relation classification using a single incremental model, which is similar in spirit to Miwa and Bansal (2016) by performing the task end-to-end.

Formally, given a sentence $w_1 w_2 \cdots w_n$, we maintain a table $T^{n \times n}$, where $T(i,j)$ denotes the relation between w_i and w_j. When $i = j$, $T(i,j)$ denotes an entity boundary label. We map entity words into labels under the BILOU (Begin, Inside, Last, Outside, Unit) scheme, assuming that there are no overlapping entities in one sentence (Li and Ji, 2014; Miwa and Sasaki, 2014; Miwa and Bansal, 2016). Only the upper triangular table is necessary for indicating the relations.

We adopt the close-first left-to-right order (Miwa and Sasaki, 2014) to map the two-dimensional table into a sequence, in order to fill the table incrementally. As shown in Figure 2, first $\{T(i,i)\}$ are filled by growing i, and then the sequence $\{T(i,i+1)\}$ is filled, and then $\{T(i,i+2)\}, \cdots, \{T(i,i+n)\}$ are filled incrementally, until the table is fully annotated.

During the table-filling process, we take two label sets for entity detection ($i = j$) and relation

[1] https://github.com/zhangmeishan/NNRelationExtraction

	Associated	Press	writer	Patrick	McDowell	in	Kuwait	City
Associated	1 B-ORG	9 ⊥	16 ⊥	22 ⊥	27 ⊥	31 ⊥	34 ⊥	36 ⊥
Press		2 L-ORG	10 $\overleftarrow{\text{ORG-AFF}}$	17 ⊥	23 ⊥	28 ⊥	32 ⊥	35 ⊥
writer			3 U-PER	11 ⊥	18 ⊥	24 ⊥	29 ⊥	33 ⊥
Patrick				4 B-PER	12 ⊥	19 ⊥	25 ⊥	30 ⊥
McDowell					5 L-PER	13 ⊥	20 ⊥	26 $\overrightarrow{\text{PHYS}}$
in						6 O	14 ⊥	21 ⊥
Kuwait							7 B-GPE	15 ⊥
City								8 L-GPE

Figure 2: Table-filling example, where numbers indicate the filling order.

classification ($i < j$), respectively. The labels for entity detection include {B-*, I-*, L-*, O, U-* }, where * denotes the entity type, and the labels for relation classification are $\{\overrightarrow{*}, \overleftarrow{*}, \perp\}$, where * denotes the relation category and $\perp$ denotes a NULL relation.[2]

At each step, given a partially-filled table T, we determine the most suitable label l for the next step using a scoring function:

$$\text{score}(T, l) = W_l h_T, \tag{1}$$

where W_l is a model parameter and h_T is the vector representation of T. Based on the function, we aim to find the best label sequence $l_1 \cdots l_m$, where $m = \frac{n(n+1)}{2}$, and the resulting sequence of partially-filled tables is $T_0 T_1 \cdots T_m$, where $T_i = \text{FILL}(T_{i-1}, l_i)$, and T_0 is an empty table. Different from previous work, we investigate a structural model that is optimized for the label sequence $l_1 \cdots l_m$ globally, rather than for each l_i locally.

2.3 Representation Learning

At the ith step, we determine the label l_i of the next table slot based on the current hypothesis T_{i-1}. Following Miwa and Bansal (2016), we use a neural network to learn the vector representation of T_{i-1}, and then use Equation 1 to rank candidate next labels. There are two types of input features, including the word sequence $w_1 w_2 \cdots w_n$, and the readily filled label sequence $l_1 l_2 \cdots l_{i-1}$. We build a neural network to represent T_{i-1}.

2.3.1 Word Representation

Shown in Figure 3, we represent each word w_i by a vector h_i^w using its word form, POS tag and characters. Two different forms of embeddings are used based on the word form, one being obtained by using a randomly initialized look-up table E_w, tuned during training and represented by e_w, and the other being a pre-trained external word embedding from E'_w, which is fixed and represented by e'_w.[3] For a POS tag t, its embedding e_t is obtained from a look-up table E_t similar to E_w.

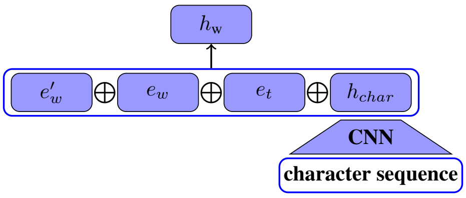


Figure 3: Word representations.

The above two components have also been used by Miwa and Bansal (2016). We further enhance the word representation by using its character sequence (Ballesteros et al., 2015; Lample et al., 2016), taking a convolution neural network (CNN) to derive a character-based word representation h_{char}, which has been demonstrated effective for several NLP tasks (dos Santos and Gatti, 2014). We obtain the final h_i^w based on a non-linear feed-forward layer on $e'_w \oplus e_w \oplus e_t \oplus h_{char}$, where $\oplus$ denotes concatenation.

2.3.2 Label Representation

In addition to the word sequence, the history label sequence $l_1 l_2 \cdots l_{i-1}$, and especially the labels representing detected entities, are also useful disambiguation. For example, the previous entity boundary label can be helpful to deciding the boundary label of the current word. During relation classification, the types of the entities involved can indicate the relation category between them. We exploit the diagonal label sequence of partial table T, which denotes entity boundaries, to enhance the representation learning. A word's entity boundary label embedding e_l is obtained by

[2] We remove the illegal table-filling labels during decoding for training and testing. For example, $T(i, j)$ must be $\perp$ if $T(i, i)$ or $T(j, j)$ equals O.

[3] We use the set of pre-trained glove word embeddings available at http://nlp.stanford.edu/data/glove.6B.zip as external word embeddings.

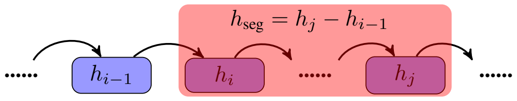


Figure 4: Segment representation.

using a randomly initialized looking-up table E_l.

2.3.3 LSTM Features

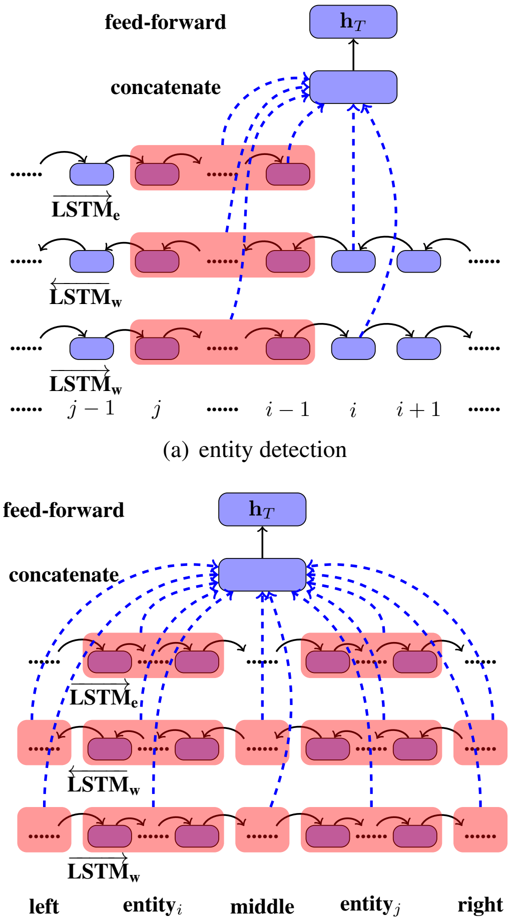
We follow Miwa and Bansal (2016), learning global context representations using LSTMs. Three *basic* LSTM structures are used: a left-to-right word LSTM ($\overrightarrow{\text{LSTM}}_w$), a right-to-left word LSTM ($\overleftarrow{\text{LSTM}}_w$) and a left-to-right entity boundary label LSTM ($\overrightarrow{\text{LSTM}}_e$). Each LSTM derives a sequence of hidden vectors for inputs. For example, for $w_1 w_2 \cdots w_n$, $\overrightarrow{\text{LSTM}}_w$ gives $h_1^{w,\rightarrow} h_2^{w,\rightarrow} \cdots h_n^{w,\rightarrow}$.

Different from Miwa and Bansal (2016), who use the output hidden vectors $\{h_i\}$ of LSTMs to represent words, we exploit *segment* representations as well. In particular, for a segment of text $[i, j]$, the representation is computed by using LSTM-Minus (Wang and Chang, 2016), shown by Figure 4, where $h_j - h_{i-1}$ in a left-to-right LSTM and $h_i - h_{j+1}$ in a right-to-left LSTM are used to represent the segment $[i, j]$. The segment representations can reflect entities in a sentence, and thus can be potentially useful for both entity detection and relation extraction.

2.3.4 Feature Representation

We use separate feature representations for entity detection and relation classification, both of which are extracted from the above three LSTM structures. In particular, we first extract a set of base neural features, and then concatenate them and feed them into a non-linear neural layer for entity detection and relation classification, respectively. Figure 5 shows the overall representation.

[Entity Detection] Figure 5(a) shows the feature representation for the entity detection. First, we extract six feature vectors from the three basic LSTMs, three of which are word features, namely $h_i^{w,\rightarrow}$, $h_i^{w,\leftarrow}$ and $h_{i-1}^{e,\rightarrow}$, and the remaining are segment features, namely $h_{[j,i-1]}^{w,\rightarrow}$, $h_{[j,i-1]}^{w,\leftarrow}$ and $h_{[j,i-1]}^{e,\rightarrow}$, where j denotes the start position of the previous entity.[4] The segment features are computed dynamically from the partial outputs of entity detection, according to the boundaries of the lastly-formed entity during the decoding. The six vectors are concatenated and then fed into a non-linear layer for entity detection.



(a) entity detection

(b) relation classification

Figure 5: Feature representation.

[Relation Classification] Figure 5(b) shows the feature representation for relation classification. Similar to entity detection, we extract a set of features from the basic LSTMs ($\overrightarrow{\text{LSTM}}_w$, $\overleftarrow{\text{LSTM}}_w$ and $\overrightarrow{\text{LSTM}}_e$), and then concatenate them for a non-linear classification layer. The differences between relation classification with entity detection lie in the range of hidden layers from LSTMs. For relation classification between i and j, we split each LSTM into five segments according to the two entities ended with i and j. Formally, let $[s(i), i]$ and $[s(j), j]$ denote the two entities above, where $s(\cdot)$ denotes the start position of an entity, the resulted segments are $[0, s(i)-1]$ (i.e., **left**, in Figure 5(b)), $[s(i), i]$ (i.e., **entity**$_i$), $[i+1, s(j)-1]$ (i.e., **middle**), $[s(j), j]$ (i.e., **entity**$_j$) and $[j+1, n]$ (i.e., **right**), respectively. For the word LSTMs, we extract all five segment features, while the en-

[4] The non-entity word is treated as a special unit entity to extract segmental features.

Models	Encoder	LAS
S-LSTM (2015)	1-Layer LSTM	90.9
K&G (2016)	2-Layer Bi-LSTM	91.9
D&M (2016)	4-Layer Bi-LSTM	**93.8**

Table 1: Encoder structures and performances of three state-of-the-art dependency parsers, where S-LSTM (2015) refers to Dyer et al. (2015), K&G (2016) refers to the best parser of Kiperwasser and Goldberg (2016), D&M (2016) refers to Dozat and Manning (2016), and LAS (labeled attachment score) is the major evaluation metric.

tity label LSTM, we only use the segment features of $\textbf{entity}_i$ and $\textbf{entity}_j$.

2.3.5 Syntactic Features

Previous work has shown that syntactic features are useful for relation extraction (Zhou et al., 2005). For example, the shortest dependency path has been used by several relation extraction models (Bunescu and Mooney, 2005; Miwa and Bansal, 2016). Here we propose a novel method to integrate syntax, without need for prior knowledge on concrete syntactic structures.

In particular, we take state-of-the-art syntactic parsers that use encoder-decoder neural models (Buys and Blunsom, 2015; Kiperwasser and Goldberg, 2016), where the encoder represents the syntactic features of the input sentences. For example, LSTM hidden states over the input word/tag sequences has been used frequently as syntactic features (Kiperwasser and Goldberg, 2016). Such features represent input words with syntactic information. The parser decoder also leverages partially-parsed results, such as features from partial syntactic trees, although we do not use explicit output features. Table 1 shows the encoder structures of three state-of-the-art dependency parsers.

Our method is to leverage trained syntactic parsers, dumping the encoder feature representations given our inputs, using them directly as part of input embeddings in our proposed model. Denoting the dumped syntactic features on each word as $h_1^{\text{syn}} h_2^{\text{syn}} \cdots h_n^{\text{syn}}$, we feed them into a nonlinear neural layer, and then generate two LSTMs (bi-directional) based on the outputs, namely $\overrightarrow{\text{LSTM}}_{syn}$ and $\overleftarrow{\text{LSTM}}_{syn}$, respectively, augmenting the original three LSTMs into five LSTMs. Features are extracted from the two new LSTMs in the same way as from the basic bi-directional word LSTMs.

In this paper, we exploit the parser of Dozat and Manning (2016), since it achieves the current best performance for dependency parsing. Our method can be easily generalized to other parsers, which are potentially useful for our task as well. For example, we can use a constituent parser in the same way by dumping the implicit encoder features.

Our exploration of syntactic features has two main advantages over the method of Miwa and Bansal (2016), where dependency path LSTMs are used for relation classification. On the one hand, incorrect dependency paths between entity pairs can propagate to relation classification in Miwa and Bansal (2016), because these paths rely on explicit discrete outputs from a syntactic parser. Our method can avoid the problem since we do not compute parser outputs. On the other hand, the computation complexity is largely reduced by using our method since sequential LSTMs are based on inputs only, while the dependency path LSTMs should be computed based on the dynamic entity detection outputs. When beam search is exploited during decoding, increasing number of dependency paths can be used by a surge of entity pairs from beam outputs.

Our method can be extended into neural stacking Wang et al. (2017), by doing back-propagation training of the parser parameters during model training, which are leave for future work.

2.4 Training and Search

2.4.1 Local Optimization

Previous work (Miwa and Bansal, 2016; Gupta et al., 2016) trains model parameters by modeling each step for labeling one input sentence separately. Given a partial table T, its neural representation h_T is first obtained, and then compute the next label scores $\{l_1, l_2, \cdots, l_s\}$ using Equation 1. The output scores are regularized into a probability distribution $\{p_{l_1}, p_{l_2}, \cdots, p_{l_s}\}$ by using a softmax layer. The training objective is to minimize the cross-entropy loss between this output distribution with the gold-standard distribution:

$$\text{loss}(T, l_i^g, \Theta) = -\log p_{l_i^g}, \qquad (2)$$

where l_i^g is the gold-standard next label for T, and Θ is the set of all model parameters. We refer this training method as *local optimization*, because it maximizes the score of the gold-standard label at each step locally.

Algorithm 1 Beam-search.

```
$agenda \leftarrow \{$ ($empty\ table$, score=0.0) $\}$
for $i$ in $1 \cdots$ max-step
    $next\_scored\_tables \leftarrow \{\ \}$
    for $scored\_table$ in $agenda$
        $labels \leftarrow$ NEXTLABELS($scored\_table$)
        for next_label in labels
            new $\leftarrow$ FILL(scored_table, $next\_label$)
            ADDITEM($next\_scored\_tables$, $new$)
    $agenda \leftarrow$ TOP-B($next\_scored\_tables$, $B$)
```

During the decoding phase, the greedy search strategy is applied in consistence with the training. At each step, we find the highest-scored label based on the current partial table, before going on to the next step.

2.4.2 Global Optimization

We exploit the global optimization strategy of Zhou et al. (2015) and Andor et al. (2016), maximizing the cumulative score of the gold-standard label sequence for one sentence as a unit. Global optimization has achieved success for several NLP tasks under the neural setting (Zhou et al., 2015; Watanabe and Sumita, 2015). For relation extraction, global learning gives the best performances under the discrete setting (Li and Ji, 2014; Miwa and Sasaki, 2014). We study such models here for neural network models.

Given a label sequence of $l_1 l_2 \cdots l_i$, the score of T_i is defined as follows:

$$\begin{aligned}\text{score}(T_i) &= \sum_{j=0}^{i} \text{score}(T_{j-1}, l_j) \\ &= \text{score}(T_{i-1}) + \text{score}(T_{i-1}, l_i),\end{aligned} \quad (3)$$

where $\text{score}(T_0) = 0$ and $\text{score}(T_{i-1}, l_i)$ is computed by Equation 1. By this definition, we maximize the scores of all gold-standard partial tables.

Again cross-entropy loss is used to perform model updates. At each step i, the objective function is defined by:

$$\begin{aligned}\text{loss}(x, T_i^g, \Theta) &= -\log p_{T_i^g} \\ &= -\log \frac{\text{score}(T_i^g)}{\sum_{T_i'} \text{score}(T_i')},\end{aligned} \quad (4)$$

where x denotes the input sentence, T_i^g denotes the gold-standard state at step i, and T_i' are all partial tables that can be reached at step i.

The major challenge is to compute $p_{T_i^g}$, because we cannot traverse all partial tables that are valid at step i, since their count increases exponentially by the step number. We follow Andor et al. (2016), approximating the probability by using beam search and early-update.

Shown in Algorithm 1, we use standard beam search, maintaining the B highest-scored partially-filled tables in an agenda at each step. When each action of table filling is taken, all hypotheses in the agenda are expanded by enumerating the next labels, and the B highest-scored resulting tables are used to replace the agenda for the next step. Search begins with the agenda containing an empty table, and finishes when all cells of the tables in the agenda have been filled. When the beam size is 1, the algorithm is the same as greedy decoding. When the beam size is larger than 1, however, error propagation is alleviated. For training, the same beam search algorithm is applied to training examples, and early-update (Collins and Roark, 2004) is used to fix search errors.

3 Experiments

3.1 Data and Evaluation

We evaluate the proposed model on two datasets, namely the ACE05 data and the corpus of Roth and Yih (2004) (CONLL04), respectively. The ACE05 dataset defines seven coarse-grained entity types and six coarse-grained relation categories, while the CONLL04 dataset defines four entity types and five relation categories.

For the ACE05 dataset, we follow Li and Ji (2014) and Miwa and Bansal (2016), splitting and preprocessing the dataset into training, development and test sets.[5] For the CONLL04 dataset, we follow Miwa and Sasaki (2014) to split the data into training and test corpora, and then divide 10% of the training corpus for development.

We use the micro F1-measure as the major metric to evaluate model performances, treating an entity as correct when its head region and type are both correct,[6] and regard a relation as correct when the argument entities and the relation category are all correct. We exploit pairwise t-test for measuring significance values.

[5] https://github.com/tticoin/LSTM-ER/.

[6] For the ACE05 dataset, the head region is defined by the corpus, and for the CONLL04 dataset, the head region covers the entire scope of an entity.

Network Structure	Size
Word Embedding	200
Tag Embedding	50
Char Embedding	50
Entity Label Embedding	50
Input/Output of Word LSTMs	250
Input/Output of Entity Label LSTMs	100
Table Representation	300

Table 2: Dimension sizes.

Model	Entity F1	Relation F1
baseline	**81.5**	**50.9**
-character	80.9	50.2
-segment (entity detection)	80.2	49.8

Table 3: Feature ablation tests.

Model	Beam	Relation F1	Speed
Local	1	50.9	**95.6**
Local(+SS)	1	51.2	95.1
Global	1	51.4	95.3
	3	51.8	52.0
	5	**52.6**	36.9

Table 4: Comparisons between local and global models, where SS denotes scheduled sampling, and speed is measured by the number of sentences per second.

3.2 Parameter Tuning

We update all model parameters by back propagation using Adam (Kingma and Ba, 2014) with a learning rate 10^{-3}, using gradient clipping by a max norm 10 and l_2-regularization by a parameter 10^{-5}. The dimension sizes of various vectors in neural network structure are shown in Table 2. All the hyper-parameters are tuned by development experiments. All experiments are conducted using gcc version 4.9.4 (Ubuntu 4.9.4-2ubuntu1 14.04.1), on an Intel(R) Xeon(R) CPU E5-2670 @ 2.60GHz.

Online training is used to learn parameters, traversing over the entire training examples by 300 iterations. We select the best iteration number according to the development results. In particular, we exploit pre-training techniques (Wiseman and Rush, 2016) to learn better model parameters. For the local model, we follow Miwa and Bansal (2016), training parameters only for entity detection during the first 20 iterations. For the global model, we pretrain our model using local optimization for 40 iterations, before conducting beam global optimization.

3.3 Development Experiments

We conduct several development experiments on the ACE05 development dataset.

3.3.1 Feature Ablation Tests

We consider the baseline system with no syntactic features using local training. Compared with Miwa and Bansal (2016), we introduce character-level features, and in addition exploit segmental features for entity detection. Feature ablation experiments are conducted for the two types of features. Table 3 shows the experimental results, which demonstrate that the character-level features and the segment features we use are both useful for relation extraction.

3.3.2 Local v.s. Global Training

We study the influence of training strategies for relation extraction without using syntactic features. For the local model, we apply scheduled sampling (Bengio et al., 2015), which has been shown to improve the performance of relation extraction by Miwa and Bansal (2016).

Table 4 shows the results. Scheduled sampling achieves improved F-measure scores for the local model. With the same greedy search strategy, the globally normalized model gives slightly better results than the local model with scheduled sampling. The performance of the global model increases with a larger beam size. When beam size 5 is exploited, we obtain a further gain of 1.2% on the relation F-measure, which is significantly better than our baseline local model with scheduled sampling ($p \approx 10^{-4}$). However, the decoding speed becomes intolerably slow when the beam size increases beyond 5. Thus we exploit a beam size of 5 for global training considering both performance and efficiency.

3.3.3 Syntactic Features

We examine the effectiveness of the proposed implicit syntactic features. Table 5 shows the development results using both local and global optimization. The proposed features improve the relation performances significantly under both settings ($p < 10^{-4}$), demonstrating that our use of syntactic features is highly effective.

We also compare our feature integration method with the traditional methods based on syntactic

Model	Features	Entity F1	Relation F1
Local	all	**81.6**	**53.0**
	-syn	81.5	50.9
Global	all	**81.9**	**54.2**
	-syn	81.6	52.6

Table 5: The influence of syntactic features.

model	ACE05		CONLL04	
	Entity	Relation	Entity	Relation
Our Model	**83.6**	**57.5**	**85.6**	**67.8**
M&B (2016)	83.4	55.6	—	—
L&J (2014)	80.8	49.5	—	—
M&S (2014)	—	—	80.7	61.0

Table 6: Final results on the test datasets.

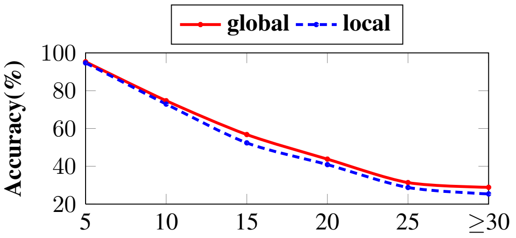


Figure 6: Sentence-level accuracies with respect to sentence length.

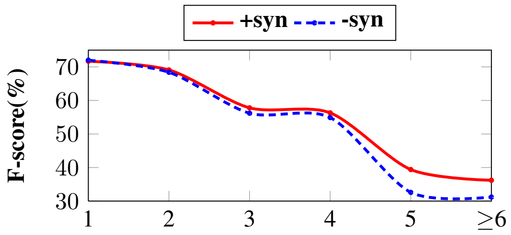


Figure 7: F-scores with respect to the distance between entity pairs.

outputs which Miwa and Bansal (2016) and all previous methods use. We use the same parser of Dozat and Manning (2016), building features on its dependency outputs. We exploit the bidirectional tree LSTM of Teng and Zhang (2016) to extract neural features, and then exploit a nonlinear feed-forward neural network to combine the two features. Similarly, we extract segment features but by using max pooling instead over the sequential outputs of the feed-forward layer, since the vector minus is nonsense here. The final relation results are 53.1% and 53.9% for the local and global models, respectively, which have no significantly differences compared with our models. On the other hand, our method is relatively more efficient, and flexible to the grammar formalism.

3.4 Final Results

Table 6 shows the final results on the test datasets of ACE05 and CONLL04. We show several top-performing systems in the table as well, where M&B (2016) refers to Miwa and Bansal (2016), who exploit end-to-end LSTM neural networks with local optimization, and L&J (2014) and M&S (2014) refer to Li and Ji (2014) and Miwa and Sasaki (2014), respectively, which are both globally optimized models using discrete features, giving the top F-scores among statistical models.[7]

Overall, neural models give better performances than statistical models, and global optimization can give improved performances as well. Our final model achieves the best performances on both datasets. Compared with the best reported results, our model gives improvements of 1.9% on ACE05, and 6.8% on CONLL04.

3.5 Analysis

We conduct analysis on the ACE05 test dataset in order to better understand our models, on its two major contributions, first examining the influences of global optimization, and then studying the gains by using the proposed syntactic features.

Intuitively global optimization should give better accuracies at the sentence level. We verify this by examining the sentence-level accuracies, where one sentence is regarded as correct when all the labels in the resulted table are correct. Figure 6 shows the result, which is consistent with our intuition. The sentence-level accuracies of the globally normalized model are consistently better than the local model. In addition, the accuracy decreases sharply as the sentence length increases, with the local model suffering more severely from larger sentences.

To understand the effectiveness of the proposed syntactic features, we examine the relation F-scores with respect to entity distances. Miwa and Bansal (2016) exploit the shortest dependency path, which can make the distance between two entities closer compared with their sequential dis-

[7] Gupta et al. (2016) proposed a locally optimized model but used a different test dataset from CONLL04 and a different evaluation method, reporting entity and relation F-scores of 93.6% and 72.1%, respectively. Their results are not directly comparable to the results in Table 6. In particular, they regard an entity as correct if at least one token is tagged correctly, which influences the results significantly since multi-word entities accounts for over 50% of all entities.

tance, thus facilitating relation extraction. We verify whether the proposed syntactic features can benefit our model similarly. As shown in Figure 7, the F-scores of entity-pairs with large distances see apparent improvements, demonstrating that our use of syntactic features has a similar effect compared to the shortest dependency path.

4 Related Work

Entity recognition (Florian et al., 2004, 2006; Ratinov and Roth, 2009; Florian et al., 2010; Kuru et al., 2016) and relation extraction (Zhao and Grishman, 2005; Jiang and Zhai, 2007; Zhou et al., 2007; Qian and Zhou, 2010; Chan and Roth, 2010; Sun et al., 2011; Plank and Moschitti, 2013; Verga et al., 2016) have received much attention in the NLP community. The dominant methods treat the two tasks separately, where relation extraction is performed assuming that entity boundaries have been given (Zelenko et al., 2003; Miwa et al., 2009; Chan and Roth, 2011; Lin et al., 2016).

Several studies find that extracting entities and relations jointly can benefit both tasks. Early work conducts joint inference for separate models (Ji and Grishman, 2005; Roth and Yih, 2004, 2007). Recent work shows that joint learning and decoding with a single model brings more benefits for the two tasks (Li and Ji, 2014; Miwa and Sasaki, 2014; Miwa and Bansal, 2016; Gupta et al., 2016), and we follow this line of work in the study.

LSTM features have been extensively exploited for NLP tasks, including tagging (Huang et al., 2015; Lample et al., 2016), parsing (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2016), relation classification (Xu et al., 2015; Vu et al., 2016; Miwa and Bansal, 2016) and sentiment analysis (Li et al., 2015; Teng et al., 2016). Based on the output of LSTM structures, Wang and Chang (2016) introduce segment features, and apply it to dependency parsing. The same method is applied to constituent parsing by Cross and Huang (2016). We exploit this segmental representation for relation extraction.

Global optimization and normalization has been successfully applied on many NLP tasks that involve structural prediction (Lafferty et al., 2001; Collins, 2002; McDonald et al., 2010; Zhang and Clark, 2011), using traditional discrete features. For neural models, it has recently received increasing interests (Zhou et al., 2015; Andor et al., 2016; Xu, 2016; Wiseman and Rush, 2016), and improved performances can be achieved with global optimization accompanied by beam search. Our work is in line with these efforts. To our knowledge, we are the first to apply globally optimized neural models for end-to-end relation extraction, achieving the best results on standard benchmarks.

5 Conclusion

We investigated a globally normalized end-to-end relation extraction model using neural network, based on the table-filling framework proposed by Miwa and Sasaki (2014). Feature representations are learned from several LSTM structures over the inputs, and a novel simple method is used to integrate syntactic information. Experiments show the effectiveness of both global normalization and syntactic features. Our final model achieved the best performances on two benchmark datasets.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, which help to improve the paper, and Zhiyang Teng for dumping intermediate outputs from the bi-affine parser. This work is supported by National Natural Science Foundation of China (NSFC) grants 61602160 and 61672211, Natural Science Foundation of Heilongjiang Province (China) grant F2016036, Special business expenses in Heilongjiang Province (China) grant 2016-KYYWF-0183, the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301 and SRG ISTD 2012 038 from Singapore University of Technology and Design. Yue Zhang is the corresponding author.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *ACL*, pages 2442–2452.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the EMNLP*, pages 349–359.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, pages 1171–1179.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *EMNLP*, pages 724–731. Association for Computational Linguistics.

Jan Buys and Phil Blunsom. 2015. Generative incremental dependency parsing with neural networks. In *Proceedings of the 53rd ACL*, pages 863–869.

Yee Seng Chan and Dan Roth. 2010. Exploiting background knowledge for relation extraction. In *COLING*, pages 152–160.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *ACL*, pages 551–560.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *ACL*, pages 111–118.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *EMNLP*, pages 1–11.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *ACL*, pages 423–429.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, volume 2, page 1.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL*, pages 334–343.

R Florian, H Hassan, A Ittycheriah, H Jing, N Kambhatla, X Luo, N Nicolov, and S Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *NAACL*, pages 1–8.

Radu Florian, Hongyan Jing, Nanda Kambhatla, and Imed Zitouni. 2006. Factorizing complex models: A case study in mention detection. In *COLING/ACL*, pages 473–480.

Radu Florian, John Pitrelli, Salim Roukos, and Imed Zitouni. 2010. Improving mention detection robustness to noisy input. In *EMNLP*, pages 335–345.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *ICASSP*, pages 6645–6649. IEEE.

Ralph Grishman. 1997. Information extraction: Techniques and challenges. In *Information extraction a multidisciplinary approach to an emerging information technology*, pages 10–27. Springer.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016*, pages 2537–2547.

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

Heng Ji and Ralph Grishman. 2005. Improving name tagging by reference resolution and relation detection. In *ACL*, pages 411–418.

Jing Jiang and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *NAACL*, pages 113–120.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *TACL*, 4:313–327.

Onur Kuru, Ozan Arkan Can, and Deniz Yuret. 2016. Charner: Character-level named entity recognition. In *Proceedings of COLING 2016*, pages 911–921.

John Lafferty, Andrew McCallum, Fernando Pereira, et al. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, volume 1, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*, pages 260–270.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the EMNLP*, pages 2304–2314.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the Association for Computational Linguistics*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *ACL*, pages 2124–2133.

Ryan McDonald, Keith Hall, and Gideon Mann. 2010. Distributed training strategies for the structured perceptron. In *NAACL*, pages 456–464.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *ACL*, pages 1105–1116.

Makoto Miwa, Rune Sætre, Yusuke Miyao, and Jun'ichi Tsujii. 2009. A rich feature vector for protein-protein interaction extraction from multiple corpora. In *EMNLP*, pages 121–130.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *EMNLP*, pages 1858–1869.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *ACL*, pages 1498–1507.

Longhua Qian and Guodong Zhou. 2010. Clustering-based stratified seed sampling for semi-supervised relation classification. In *EMNLP*, pages 346–355.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Coling 2008*, pages 697–704.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *CoNLL*, pages 1–8.

Dan Roth and Wen-tau Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. *Introduction to statistical relational learning*, pages 553–580.

Cicero dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014*, pages 69–78.

Ang Sun, Ralph Grishman, and Satoshi Sekine. 2011. Semi-supervised relation extraction with large-scale word clustering. In *ACL*, pages 521–529.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, pages 1556–1566.

Zhiyang Teng, Duy Tin Vo, and Yue Zhang. 2016. Context-sensitive lexicon features for neural sentiment analysis. In *Proceedings of the EMNLP*, pages 1629–1638.

Zhiyang Teng and Yue Zhang. 2016. Bidirectional tree-structured lstm with head lexicalization. *arXiv preprint arXiv:1611.06788*.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *Proceedings of the 2016 NAACL*, pages 886–896.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of the NAACL*, pages 534–539.

Hongmin Wang, Yue Zhang, GuangYong Leonard Chan, Jie Yang, and Hai Leong Chieu. 2017. Universal dependencies parsing for colloquial singaporean english. *CoRR*, abs/1705.06463.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *ACL*, pages 2306–2315.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *ACL*, pages 1169–1179.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *EMNLP*, pages 1296–1306.

Wenduan Xu. 2016. Lstm shift-reduce ccg parsing. In *EMNLP*, pages 1754–1764.

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

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-based neural word segmentation. In *Proceedings of ACL*, pages 421–431, Berlin, Germany.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *ACL*, pages 419–426.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *ACL*, pages 427–434.

GuoDong Zhou, Min Zhang, DongHong Ji, and QiaoMing Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *EMNLP-CoNLL*, pages 728–736.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd ACL*, pages 1213–1222.

KGEval: Accuracy Estimation of Automatically Constructed Knowledge Graphs

Prakhar Ojha
Indian Institute of Science
prakhar@iisc.ac.in

Partha Talukdar
Indian Institute of Science
ppt@iisc.ac.in

Abstract

Automatic construction of large knowledge graphs (KG) by mining web-scale text datasets has received considerable attention recently. Estimating accuracy of such automatically constructed KGs is a challenging problem due to their size and diversity and has largely been ignored in prior research. In this work, we try to fill this gap by proposing KGEval. KGEval uses *coupling constraints* to bind facts and crowdsource those few that can *infer* large parts of the graph. We demonstrate that the objective optimized by KGEval is submodular and NP-hard, allowing guarantees for our approximation algorithm. Through experiments on real-world datasets, we demonstrate that KGEval best estimates KG accuracy compared to other baselines, while requiring significantly lesser number of human evaluations.

1 Introduction

Automatic construction of Knowledge Graphs (KGs) from Web documents has received significant interest over the last few years, resulting in the development of several large KGs consisting of hundreds of predicates (e.g., *isCity*, *stadiumLocatedInCity(Stadium, City)*) and millions of their instances called beliefs (e.g., *(Joe Luis Arena, stadiumLocatedInCity, Detroit)*). Examples of such KGs include NELL (Mitchell et al., 2015), Knowledge-Vault (Dong et al., 2014) etc.

Due to imperfections in the automatic KG construction process, many incorrect beliefs are also found in these KGs. Knowing accuracy for each predicate in the KG can provide targeted feedback for improvement and highlight its strengths from weaknesses, whereas overall accuracy of a KG can quantify the effectiveness of its construction-process. Knowing accuracy at predicate-level granularity is immensely helpful for Question-Answering (QA) systems that integrate opinions from multiple KGs (Samadi et al., 2015). For such systems, being aware that a particular KG is more accurate than others in a certain domain, say sports, helps in restricting the search over relevant and accurate subsets of KGs, thereby improving QA-precision and response time. In comparison to the large body of recent work focused on construction of KGs, the important problem of accuracy estimation of such large KGs is unexplored – we address this gap in this paper.

True accuracy of a predicate may be estimated by aggregating human judgments on correctness of each and every belief in the predicate[1]. Even though crowdsourcing marketplaces such as Amazon Mechanical Turk (AMT) provide a convenient way to collect human judgments, accumulating such judgments at the scale of larges KGs is prohibitively expensive. We shall refer to the task of manually classifying a single belief as true or false as a Belief Evaluation Task (BET). Thus, the crucial problem is: *How can we select a subset of beliefs to evaluate which will best estimate the true (but unknown) accuracy of KG and its predicates?*

A naive and popular approach is to evaluate randomly sampled subset of beliefs from the KG. Since random sampling ignores relational-couplings present among the beliefs, it usually results in oversampling and poor accuracy estimates. Let us motivate this through an example.

[1] Note that belief evaluation can not be completely automated and will require human-judgment. If an algorithm could accurately predict correctness of a belief, then it may as well be used during KG construction rather than during evaluation.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1741–1750
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Motivating example: We motivate efficient accuracy estimation through the KG fragment shown in Figure 1. Here, each belief is an edge-triple in the graph, for example (*RedWings, isA, SportsTeam*). There are six correct and two incorrect beliefs (the two incident on *Taj Mahal*), resulting in an overall accuracy of $75\% (= 6/8)$ which we would like to estimate. Additionally, we would also like to estimate accuracies of the predicates: *homeStadiumOf*, *homeCity*, *stadiumLocatedInCity*, *cityInState* and *isA*.

We now demonstrate how evaluation labels of beliefs are constrained by each other. *Type consistency* is one such coupling constraint. For instance, we know from KG ontology that the *homeStadiumOf* predicate connects an entity from *Stadium* category to another entity in *Sports Team* category. Now, if *(Joe Louis Arena, homeStadiumOf, Red Wings)* is evaluated to be correct, then from these *type constraints* we can infer that *(Joe Louis Arena, isA, Stadium)* and *(Red Wings, isA, Sports Team)* are also correct. Similarly, by evaluating *(Taj Mahal, isA, State)* as false, we can infer that *(Detroit, cityInState, TajMahal)* is incorrect.

Additionally, we have *Horn-clause coupling constraints* (Mitchell et al., 2015; Lao et al., 2011), such as *homeStadiumOf(x, y)* ∧ *homeCity(y, z)* → *stadiumLocatedInCity(x, z)*. By evaluating *(Red Wings, homeCity, Detroit)* and applying this horn-clause to the already evaluated facts mentioned above, we infer that *(Joe Louis Arena, stadiumLocatedInCity, Detroit)* is also correct. We explore generalized forms of these constraints in Section 3.1.

Thus, evaluating only three beliefs, and exploiting constraints among them, we exactly estimate the *overall* true accuracy as 75% and also cover all predicates. In contrast, the empirical accuracy by randomly evaluating three beliefs, averaged over 5 trials, is 66.7%.

Our contributions in this paper are the following: (1). Systematic study into the important problem of evaluation of automatically constructed Knowledge Graphs. (2). A novel crowdsourcing-based system KGEval to estimate accuracy of large knowledge graphs (KGs) by exploiting dependencies among beliefs for more accurate and faster KG accuracy estimation. (3). Extensive experiments on real-world KGs to demonstrate KGEval's effectiveness and also evaluate its robustness and scalability.

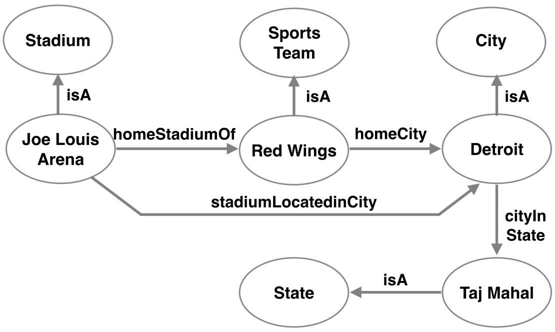


Figure 1: Sample knowledge-graph (KG) fragment that is consistent but has erroneous beliefs.

All the data and code used in the paper are available at http://talukdar.net/mall-lab/KGEval

2 Overview and Problem Statement

2.1 KGEval: Overview

We try to estimate correctness of as many beliefs as possible while evaluating only a subset of them through crowdsourcing. KGEval achieves this goal using an iterative algorithm which alternates between the following two stages:

- **Control Mechanism** (Section 3.4): In this step, KGEval selects the belief which is to be evaluated next using crowdsourcing.
- **Inference Mechanism** (Section 3.3): Coupling constraints are applied over evaluated beliefs to automatically estimate correctness of additional beliefs.

This iterative process is repeated until there are no more beliefs to be evaluated. Single iteration of KGEval over the KG fragment from Figure 1 is shown in Figure 2 where, belief *(John Louis Arena, homeStadiumOf, Red Wings)* is selected and evaluated by crowdsourcing. Subsequently, the inference mechanism uses type coupling constraints to infer *(JL Arena, isA, Stadium)* and *(R. Wings, isA, Team)* also as true. Next, we formalize the notations used in this paper.

2.2 Notations and Problem Statement

We are given a KG with n beliefs. Evaluating a single belief as true or false forms a Belief Evaluation Task (BET). Coupling constraints are derived by determining relationships among BETs, which we further discuss in Section 3.1. Notations are also summarized in Table 1.

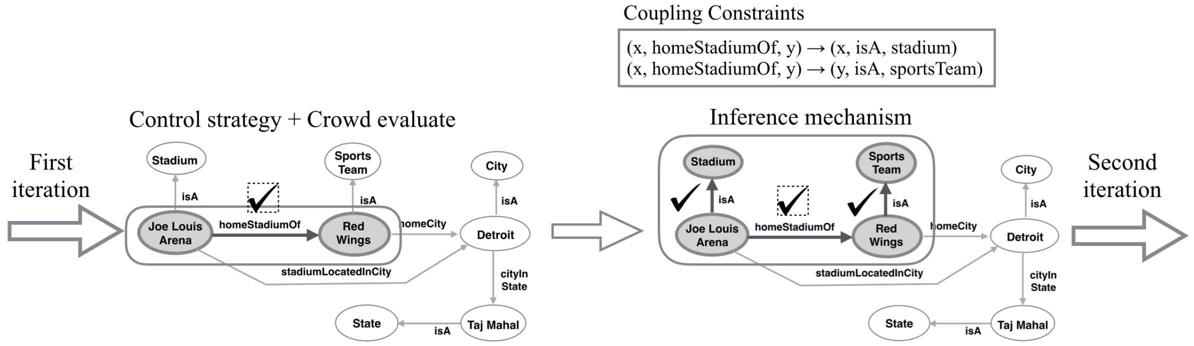


Figure 2: Demonstration of one iteration of KGEval. Control mechanism selects a belief whose correctness is evaluated from crowd. In the above example, *(J.L. Arena, homeStadiumOf, Red Wings)* is crowd-evaluated to be true (indicated by tick with dotted square). (Section 2.1 and Section 3).

Symbol	Description
$\mathcal{H} = \{h_1, \ldots, h_n\}$	Set of all n Belief Evaluation Tasks (BETs)
$c(h) \in \mathbb{R}_+$	Cost of labeling h from crowd
$\mathcal{C} = \{(\mathcal{C}_i, \theta_i)\}$	Coupling constraints $\mathcal{C}_i$ with weights $\theta_i \in \mathbb{R}_+$
$t(h) \in \{0, 1\}$	True label of h
$l(h) \in \{0, 1\}$	Estimated label of h
$\mathcal{H}_i = \text{Dom}(\mathcal{C}_i)$	$\mathcal{H}_i \subseteq \mathcal{H}$ which participate in $\mathcal{C}_i$
$G = (\mathcal{H} \cup \mathcal{C}, \mathcal{E})$	Evaluation Coupling Graph, $e \in \mathcal{E}$ between $\mathcal{H}_j$ and $\mathcal{C}_j$ denotes $\mathcal{H}_j \in \text{Dom}(\mathcal{C}_j)$.
$\mathcal{Q} \subseteq \mathcal{H}$	BETs evaluated using crowd
$\mathcal{I}(G, \mathcal{Q}) \subseteq \mathcal{H}$	Inferable set for evidence $\mathcal{Q}$:
$\Phi(\mathcal{Q}) = \frac{1}{\lvert\mathcal{Q}\rvert} \sum_{h \in \mathcal{Q}} t(h)$	True accuracy of evaluated BETs $\mathcal{Q}$

Table 1: Summary of notations used (Section 2.2).

Inference algorithm helps us work out evaluation labels of other BETs using constraints $\mathcal{C}$. For a set of already evaluated BETs $\mathcal{Q} \subseteq \mathcal{H}$, we define *inferable set* $\mathcal{I}(G, \mathcal{Q}) \subseteq \mathcal{H}$ as BETs whose evaluation labels can be deduced by the inference algorithm. We calculate the average true accuracy of a given set of evaluated BETs $\mathcal{Q} \subseteq \mathcal{I}(G, \mathcal{Q}) \subseteq \mathcal{H}$ by $\Phi(\mathcal{Q}) = \frac{1}{|\mathcal{Q}|} \sum_{h \in \mathcal{Q}} t(h)$.

KGEval aims to sample and crowdsource a BET set $\mathcal{Q}$ with the largest inferable set, and solves the optimization problem:

$$\arg\max_{\mathcal{Q} \subseteq \mathcal{H}} \left|\mathcal{I}(G, \mathcal{Q})\right| \qquad (1)$$

3 KGEval: Method Details

In this section, we describe various components of KGEval.

3.1 Coupling Constraints

The evaluation labels of beliefs are often dependent on each other due to rich relational structure of KGs. In this work, we derive coupling constraints $\mathcal{C}$ from the KG ontology and link-prediction algorithms, such as PRA (Lao et al., 2011) over NELL and AMIE (Galárraga et al., 2013) over Yago. These rules are jointly learned over entire KG with millions of facts and are assumed true.

We use conjunction-form first-order-logic rules and refer to them as *Horn clauses*. Examples of a few such coupling constraints are shown below.

$\mathcal{C}_2$: *(x, homeStadiumOf, y) → (y, isA, sportsTeam)*
$\mathcal{C}_5$: *(x, homeStadiumOf, y) ∧ (y, homeCity, z) →*
(x, stadiumLocatedInCity, z)

Each coupling constraint $\mathcal{C}_i$ operates over $\mathcal{H}_i \subseteq \mathcal{H}$ to the left of its arrow and infers label of the BET on the right of its arrow. $\mathcal{C}_2$ enforces *type consistency* and $\mathcal{C}_5$ is an instance of PRA path. These constraints have also been successfully employed earlier during knowledge extraction (Mitchell et al., 2015) and integration (Pujara et al., 2013). Note that the constraints are *directional* and inference propagates in forward direction.

3.2 Evaluation Coupling Graph (ECG)

To combine all beliefs and constraints at a common place, for all $\mathcal{H}$ and $\mathcal{C}$, we construct a graph with two types of nodes: (1) a node for each BET $h \in \mathcal{H}$, and (2) a node for each constraint $\mathcal{C}_i \in \mathcal{C}$. Each $\mathcal{C}_i$ node is connected to all h nodes that participate in it. We call this graph the *Evaluation Coupling Graph* (*ECG*), represented as $G = (\mathcal{H} \cup \mathcal{C}, \mathcal{E})$ with set of edges $\mathcal{E} = \{(\mathcal{C}_i, h) \mid h \in \text{Dom}(\mathcal{C}_i)\ \forall \mathcal{C}_i \in \mathcal{C}\}$. Note that ECG is a bipartite *factor graph* (Kschischang et al., 2001) with h as variable-nodes and $\mathcal{C}_i$ as factor-nodes.

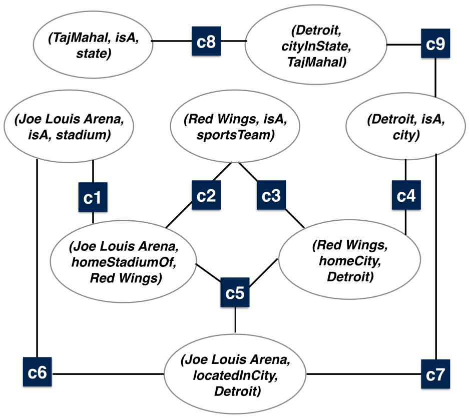

(TajMahal, isA, state)
c8
(Detroit, cityInState, TajMahal)
c9
(Joe Louis Arena, isA, stadium)
(Red Wings, isA, sportsTeam)
(Detroit, isA, city)
c1
c2
c3
c4
(Joe Louis Arena, homeStadiumOf, Red Wings)
(Red Wings, homeCity, Detroit)
c5
c6
(Joe Louis Arena, locatedInCity, Detroit)
c7

Figure 3: Evaluation Coupling Graph (ECG) constructed for example in Figure 1. (Section 3.2)

Figure 3 shows ECG constructed out of the motivating example in Figure 1 with $|\mathcal{C}| = 8$ and separate nodes for each of the edges (beliefs or BETs) in KG. We pose the KG evaluation problem as classification of BET nodes in the ECG by allotting them a label of 1 or 0 to represent true or false respectively.

3.3 Inference Mechanism

Inference mechanism helps propagate true/false labels of evaluated beliefs to other non-evaluated beliefs using available coupling constraints (Bragg et al., 2013). We use Probabilistic Soft Logic (PSL), (Broecheler et al., 2010) as our inference engine.Below we briefly describe the internal workings of our inference engine for accuracy estimation problem.

Potential function ψ_j is defined for each $\mathcal{C}_j$ using Lukaseiwicz t-norm and it depicts how satisfactorily constraint $\mathcal{C}_j$ is satisfied. For example, $\mathcal{C}_5$ mentioned earlier is transformed from first-order logical form to a real valued number by

$$\psi_j(\mathcal{C}_5) = \Big(\max\{0, h_x + h_y - 1 - h_w\}\Big)^2 \quad (2)$$

where $\mathcal{C}_5 = h_x \wedge h_y \rightarrow h_w$, where h_x denotes the evaluation score $\in [0, 1]$ associated with the BETs.

The probability distribution over label assignment is so structured such that labels which satisfy more coupling constraints are more probable. Probability of any label assignment $\Omega(\mathcal{H}) \in \{0, 1\}^{|\mathcal{H}|}$ over BETs in G is given by

$$\mathbb{P}(\Omega(\mathcal{H})) = \frac{1}{Z} \exp\Big[-\sum_{j=1}^{|\mathcal{C}|} \theta_j \psi_j(\mathcal{C}_j)\Big] \quad (3)$$

where Z is the normalizing constant and ψ_j corresponds to potential function acting over BETs $h \in \text{Dom}(\mathcal{C}_j)$. Final assignment of $\Omega(\mathcal{H})_{PSL} \in \{1, 0\}^{|\mathcal{H}|}$ labels is obtained by solving the *maximum a-posteriori (MAP)* optimization problem

$$\Omega(\mathcal{H})_{PSL} = \underset{\Omega(\mathcal{H})}{\arg\max}\ \mathbb{P}\Big(\Omega(\mathcal{H})\Big)$$

We denote by $M_{PSL}(h, \gamma) \in [0, 1]$ the PSL-estimated score for label $\gamma \in \{1, 0\}$ on BET h in the optimization above.

Inferable Set using PSL: We define estimated label for each BET h as shown below.

$$l(h) = \begin{cases} 1 \text{ if } M_{PSL}(h, 1) \geq \tau \\ 0 \text{ if } M_{PSL}(h, 0) \geq \tau \\ \varnothing \text{ otherwise} \end{cases}$$

where threshold τ is system hyper-parameter. Inferable set is composed of BETs for which inference algorithm (PSL) confidently propagates labels.

$$\mathcal{I}(G, \mathcal{Q}) = \{h \in \mathcal{H} \mid l(h) \neq \varnothing\}$$

Note that two BET nodes from ECG can interact with varying strengths through different constraint nodes; this multi-relational structure requires soft probabilistic propagation.

3.4 Control Mechanism

Control mechanism selects the BET to be crowd-evaluated at every iteration. We first present the following two theorems involving KGEval's optimization in Equation (1). Please refer Appendix for proofs of both theorems.

Theorem 1. [Submodularity] *The function optimized by KGEval (Equation (1)) using the PSL-based inference mechanism is submodular (Lovász, 1983).*

The proof follows from the fact that all pairs of BETs satisfy the regularity condition (Jegelka and Bilmes, 2011; Kolmogorov and Zabih, 2004), further used by a proven conjecture (Mossel and Roch, 2007).

Theorem 2. [NP-Hardness] *The problem of selecting optimal solution in KGEval's optimization (Equation (1)) is NP-Hard.*

Proof follows by reducing NP-complete Set-cover Problem (SCP) to selecting $\mathcal{Q}$ which covers $\mathcal{I}(G, \mathcal{Q})$.

Justification for Greedy Strategy: From Theorem 1 and 2, we observe that the function optimized by KGEval is NP-hard and submodular.

Algorithm 1 KGEval: Accuracy Estimation of Knowledge Graphs

```
Require: $\mathcal{H}$: BETs, $\mathcal{C}$: coupling constraints, $\mathbb{B}$:
    assigned budget, $\mathcal{S}$: seed set, $c(h)$: BET cost
 1: $G = \text{BUILDECG}(\mathcal{H}, \mathcal{C})$
 2: $B_r = \mathbb{B}$
 3: $\mathcal{Q}_0 = \mathcal{S}, t = 1$
 4: repeat
 5:     $h^* = \arg\max_{h \in \mathcal{H}-\mathcal{Q}} |\mathcal{I}(G, \mathcal{Q}_{t-1} \cup \{h\})|$
 6:     $\text{CROWDEVALUATE}(h^*)$
 7:     $\text{RUNINFERENCE}(\mathcal{Q}_{t-1} \cup h^*)$
 8:     $\mathcal{Q}_t = \mathcal{I}(G, \mathcal{Q}_{t-1} \cup \{h^*\})$
 9:     $B_r = B_r - c(h^*)$
10:     $\mathcal{Q} = \mathcal{Q} \cup \mathcal{Q}_t$
11:     if $\mathcal{Q} \equiv \mathcal{H}$ then
12:         EXIT
13:     end if
14:     $\mathrm{Acc_t} = \frac{1}{|\mathcal{Q}|} \sum_{h \in \mathcal{Q}} l(h)$
15:     $t = t + 1$
16: until CONVERGENCE
17: return $\mathrm{Acc_t}$
```

Results from (Nemhauser et al., 1978) prove that greedy hill-climbing algorithms solve such maximization problem within an approximation factor of $(1-1/e) \approx 63\%$ of the optimal solution. Hence we iteratively select the next BET which gives the greatest increase in size of inferable set.

We acknowledge the importance of crowdsourcing aspects such as label-aggregation, worker's quality estimation etc. Appendix A.1 presents a mechanism to handle noisy crowd workers under limited budget.

3.5 Bringing it All Together

Algorithm 1 presents KGEval. In Lines 1-3, we build the Evaluation Coupling Graph $G = (\mathcal{H} \cup \mathcal{C}, \mathcal{E})$ and use the labels of seed set $\mathcal{S}$ to initialize G. In lines 4-16, we repetitively run our inference mechanism, until either the accuracy estimates have converged, or all the BETs are covered. In each iteration, the BET with the largest inferable set is identified and evaluated using crowdsourcing (Lines 5-6). The new inferable set $\mathcal{Q}_t$ is estimated. These automatically annotated nodes are added to $\mathcal{Q}$ (Lines 7-10).

Convergence: In this paper, we define convergence whenever the variance of sequence of accuracy estimates [$\mathrm{Acc_{t-k}}, \ldots, \mathrm{Acc_{t-1}}, \mathrm{Acc_t}$] is less than α. We set $k = 9$ and $\alpha = 0.002$ for our experiments.

4 Experiments

To assess the effectiveness of KGEval, we ask the following questions: (1).How effective is KGEval in estimating KG accuracy, both at predicate-level and at overall KG-level? (Section 4.3). (2). What is the importance of coupling constraints on its performance? (Section 4.4). (3). And lastly, how robust is KGEval to estimating accuracy of KGs with varying quality? (Section 4.5).

4.1 Model Description

Evaluation set	$\mathcal{H}_N$	$\mathcal{H}_Y$
#BETs	1860	1386
#Constraints	$\lvert\mathcal{C}_N\rvert = 130$	$\lvert\mathcal{C}_Y\rvert = 28$
#Predicates	18	16
Gold Acc.	91.34%	99.20%

Table 2: Details of BET subsets used for accuracy evaluation. (Section 4.1.2).

4.1.1 Setup

Datasets: For experiments, we consider two KGs: NELL and Yago2. From NELL (NELL), we choose a sub-graph of sports related beliefs **NELL-sports**, mostly pertaining to athletes, coaches, teams, leagues, stadiums etc. We construct coupling constraints set $\mathcal{C}_\mathbf{N}$ using top-ranked PRA inference rules for available predicate-relations (Lao et al., 2011). The confidence score returned by PRA are used as weights θ_i. We use NELL-ontology's predicate-signatures to get information for *type* constraints. Please note that PSL is capable of handling weighted constraints and also learn their weights (relative importance). So, it is not critical to provide absolutely correct constraints. We also select **YAGO2-sample** (YAGO) , which unlike NELL-sports, is not domain specific. We use AMIE horn clauses (Galárraga et al., 2013) to construct multi-relational coupling constraints $\mathcal{C}_\mathbf{Y}$. For each $\mathcal{C}_i$, the score returned by AMIE is used as rule weight θ_i. Table 2 reports the statistics of datasets used, their true accuracy and number of coupling constraints. Obtaining gold-labels for millions of facts is non-trivial and expensive as crowdsourcing over full KG incurs significant cost.

Size of evaluation set: NELL-sport consists of $23,422$ beliefs with $13,290$ unique entities and 53 unique predicates. Whereas YAGO-sample has $31,720$ beliefs, with unique $32,103$ entities and 17 predicates. In order to calculate accuracy, we require gold evaluation of all beliefs in the evaluation set. Since obtaining gold evaluation of the entire (or large subsets of) NELL and Yago2 KGs will be prohibitively expensive, we take subset of these KGs for evaluation. (KGEval) consists of datasets used, their crowdsourced labels, coupling constraints and code for inference/control.

Initialization: Algorithm 1 requires initial seed set $\mathcal{S}$ which we generate by randomly evaluating $|\mathcal{S}| = 50$ beliefs from $\mathcal{H}$. To maintain fairness, all baselines start from $\mathcal{S}$. For asserting true (or false) value for beliefs, we set a high soft label confidence threshold at $\tau = 0.8$ (see Section 3.3).

4.1.2 Crowdsourcing of BETs

To compare KGEval predictions against human evaluations, we evaluate all BETs $\{\mathcal{H}_N \cup \mathcal{H}_Y\}$ on AMT. For the ease of workers, we translate each *entity-relation-entity* belief into human readable format before posting to crowd.

We published BETs on AMT under 'classification project' category. We hired AMT recognized master workers for high quality labels and paid \$0.01 per BET. To compare between 'master' and 'noisy' workers, we correlated their labels individually to expert labels on random subset and observed that master workers were better correlated (93%) as compared to three non-masters (89%). Consequently we consider votes of master workers for $\{\mathcal{H}_N \cup \mathcal{H}_Y\}$ as gold labels, which we would like our inference algorithm to be able to predict. As the average turnaround time for AMT tasks runs into a few minutes (Dupuis et al., 2013), KGEval is effectively real-time within such turnaround time range.

4.1.3 Performance Evaluation Metrics

Performance of various methods are evaluated using the following two metrics. To capture accuracy at the predicate level, we define $\Delta_{predicate}$ as the average of difference between gold and estimated accuracy of each of the R predicates in KG.

$$\Delta_{predicate} = \frac{1}{|R|}\left(\sum_{\forall r \in R}\left|\Phi(\mathcal{H}_r) - \frac{1}{|\mathcal{H}_r|}\sum_{\forall h \in \mathcal{H}_r} l(h)\right|\right)$$

We define $\Delta_{overall}$ as the difference between gold and estimated accuracy over the entire evaluation set.

$$\Delta_{overall} = \left|\Phi(\mathcal{H}) - \frac{1}{|\mathcal{H}|}\sum_{\forall h \in \mathcal{H}} l(h)\right|$$

Above, $\Phi(\mathcal{H})$ is the overall gold accuracy, $\Phi(\mathcal{H}_r)$ is the gold accuracy of predicate r and $l(h)$ is the label assigned by the currently evaluated method. $\Delta_{overall}$ treats entire KG as a single bag of BETs whereas $\Delta_{predicate}$ segregates beliefs based on their type of predicate-relation. For both metrics, lower is better.

4.2 Baseline Methods

Since accuracy estimation of large multi-relational KGs is a relatively unexplored problem, there are no well established baselines for this task (apart from random sampling). We present below the baselines which we compared against KGEval.

Random: Randomly sample a BET $h \in \mathcal{H}$ without replacement and crowdsource for its correctness. Selection of every subsequent BET is independent of previous selections.

Max-Degree: Sort the BETs in decreasing order of their degrees in ECG and select them from top for evaluation; this method favors selection of more centrally connected BETs first.

Independent Cascade: This method is based on contagion transmission model where nodes only infect their immediate neighbors (Kempe et al., 2003). At every time iteration t, we choose a BET which is not evaluated yet, crowdsource for its label and let it propagate its evaluation label in ECG.

KGEval: Method proposed in Algorithm 1.

4.3 Effectiveness of KGEval

Experimental results of all methods comparing $\Delta_{overall}$ and $\Delta_{predicate}$ at convergence, are presented in Table 3. We observe that KGEval is able to achieve the best estimate across both datasets and metrics. Due to the significant positive bias in $\mathcal{H}_Y$ (see Table 2), all methods do fairly well as per $\Delta_{overall}$ on this dataset, even though KGEval still outperforms others. Also, KGEval is able to estimate KG accuracy most closely while utilizing least number of crowd-evaluated queries. This clearly demonstrates KGEval's effectiveness.

Nodes in coupling graph with higher degrees are those which participate in large number of constraints. In real KGs, such facts tend to be correct as they interact with several other facts. Hence, MaxDegree overestimates the accuracy by propagating True label. In contrast, Random samples True and False labels in unbiased fashion.

Predicate-level Analysis: Here, we consider the top two baselines from Table 3, viz., Random and

NELL sports dataset ($\mathcal{H}_N$)			
Method	$\Delta_{predicate}$ (%)	$\Delta_{overall}$ (%)	# Queries
Random	4.9	1.3	623
Max-Deg	7.7	2.9	1370
Ind-Casc	9.8	0.8	232
KGEval	**3.6**	**0.5**	**140**
Yago dataset ($\mathcal{H}_Y$)			
Random	1.3	0.3	513
Max-Deg	1.7	0.5	550
Ind-Casc	1.1	0.7	649
KGEval	**0.7**	**0.1**	**204**

Table 3: $\Delta_{predicate}(\%)$ and $\Delta_{overall}(\%)$ estimates (lower is better) of various methods with number of crowd-evaluated queries (BET evaluations) to reach the $\Delta_{overall}$ converged estimate. (See Section 4.3)

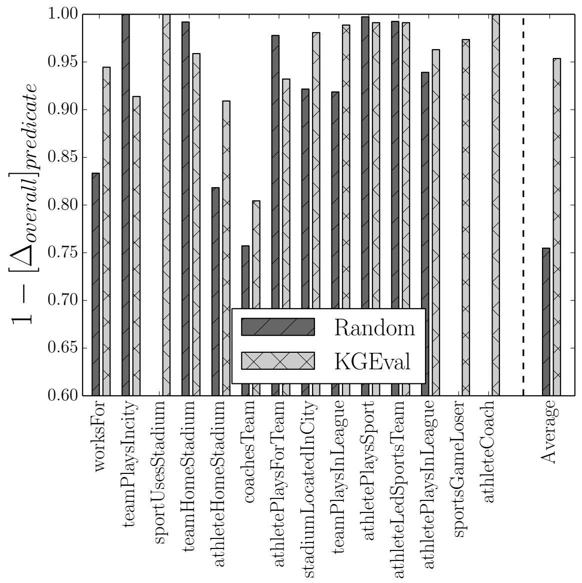


Figure 4: Comparing $(1-[\Delta_{overall}]_{predicate})$ of individual predicates (higher is better) in $\mathcal{H}_N$ between KGEval and Random, the two top performing systems in Table 3. (see Section 4.3)

KGEval, and compare performance on the $\mathcal{H}_N$ dataset. We use $(1-[\Delta_{overall}]_{predicate})$ as the metric, which means $\Delta_{overall}$ computed over individual predicates. Here, we are interested in evaluating how well the two methods have estimated per-predicate accuracy when KGEval's $\Delta_{overall}$ has converged. Comparison of per-predicate performances of the two methods is shown in Figure 4. Observe that KGEval significantly outperforms Random baseline. Its advantage lies in exploiting the coupling constraints among beliefs, where evaluating a belief from certain predicate helps infer beliefs from other predicates.

Constraint Set	Iterations to Convergence	$\Delta_{overall}(\%)$
$\mathcal{C}$	**140**	**0.5**
$\mathcal{C}-\mathcal{C}_{b3}$	259	0.9
$\mathcal{C}-\mathcal{C}_{b3}-\mathcal{C}_{b2}$	335	1.1

Table 4: Performance of KGEval with ablated constraint sets. Additional constraints help in better estimation with lesser iterations.(see Section 4.4)

4.4 Importance of Coupling Constraints

This paper is largely motivated by the thesis – *exploiting richer relational couplings among BETs may result in faster and more accurate evaluations*. To evaluate this thesis, we successively ablated Horn clause coupling constraints of body-length 2 and 3 from $\mathcal{C}_N$.

We observe that with the full (non-ablated) constraint set $\mathcal{C}_N$, KGEval takes least number of crowd evaluations of BETs to convergence, while providing best accuracy estimate. Whereas with ablated constraint sets, KGEval takes up to 2.4x more crowd-evaluation queries for convergence; thus validating our thesis.

4.5 Additional Experiments

Other Baselines along with Inference: In order to evaluate how Random and Max-degree perform in conjunction with inference mechanism, we replaced KGEval's greedy control mechanism in Line 5 of Algorithm 1 with these two control mechanisms. In our experiments, we observed that both *Random+inference* and *Max-degree+inference* are able to estimate accuracy more accurately than their control-only variants. Secondly, even though the accuracies estimated by *Random+inference* and *Max-degree+inference* were comparable to that of KGEval, they required larger number of crowd-evaluation queries – 1.2x and 1.35x more, respectively. This shows effectiveness of greedy mechanism.

Rate of Coverage: In case of large KGs with scarce budget, it is imperative to have a mechanism which covers greater parts of KG with lesser number of crowdsource queries. Figure 5 shows the fraction of total beliefs whose evaluations were automatically inferred by different methods as a function of number of crowd-evaluated beliefs. We observe that KGEval infers evaluation for the largest number of BETs at each supervision level.

Robustness to Noise: In order to test robustness of the methods in estimating accuracies

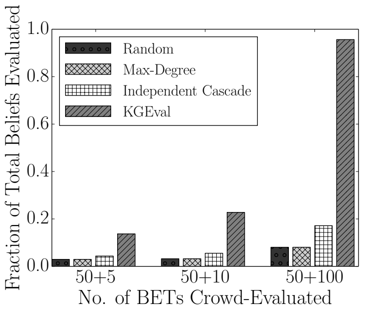


Figure 5: Fraction of total beliefs whose evaluation where automatically inferred by different methods for varying number of crowd-evaluated queries (x-axis) in $\mathcal{H}_N$.

NELL sports + 5% noise ($\mathcal{H}_{N5}$)		
Method	$\Delta_{overall}$ (%)	# Queries
Random	1.8	563
Max-Degree	4.2	1249
Ind-Cascade	1.2	370
KGEval	**0.8**	**106**
NELL sports + 10% noise ($\mathcal{H}_{N10}$)		
Random	1.8	728
Max-Degree	6.2	1501
Ind-Cascade	1.2	406
KGEval	**0.2**	**115**

Table 5: Accuracy estimate (higher is better) over entire KG by various baselines in the presence of noise.

of KGs with different gold accuracies, we artificially added noise to $\mathcal{H}_N$ by flipping a fixed fraction of edges, otherwise following the same evaluation procedure as in Section 3.5. We analyze $\Delta_{overall}$ (and not $\Delta_{predicate}$) because flipping edges in KG distorts predicate-relations dependencies and present in Table 5. We evaluated all the methods and observed that while performance of other methods degraded significantly with diminishing KG quality (more noise), KGEval was significantly robust to noise.

Scalability comparisons with MLN: Markov Logic Networks (Richardson and Domingos, 2006) can serve as a candidate for Inference Mechanism. We compared the runtime performance of KGEval with PSL and MLN as inference engines. While PSL took 320 seconds to complete one iteration, the MLN implementation (PyMLN) could not finish grounding the rules even after 7 hours. This justifies our choice of PSL as the inference engine for KGEval.

5 Related Work

Even though Knowledge Graph (KG) construction is an active area of research, we are not aware of any previous research which systematically studies the important problem of estimating accuracy of such automatically constructed KGs. Random sampling has traditionally been the most preferred way for large-scale KG accuracy estimation (Mitchell et al., 2015).

Traditional crowdsourcing research has typically considered atomic allocation of tasks where the requester posts them *independently*. KGEval operates in a rather novel crowdsourcing setting as it exploits dependencies among its tasks (BETs or belief evaluations). Our notion of interdependence (coupling constraints) among tasks is more general and different than related ideas explored in the crowdsourcing literature before (Kolobov et al., 2013; Bragg et al., 2013; Sun et al., 2015). Even though coupling constraints have been used for KG construction (Nakashole et al., 2011; Galárraga et al., 2013; Mitchell et al., 2015), they have so far not been exploited for KG evaluation. We address this gap in this paper.

The task of knowledge corroboration (Kasneci et al., 2010) proposes probabilistic model to utilize a fixed set of basic first-order logic rules for label propagation and is closely aligned with our motivations. However, unlike KGEval, it does not try to reduce the number of queries to crowdsource or maximize coverage.

6 Conclusion

In this paper, we have initiated a systematic study into the important problem of evaluation of automatically constructed Knowledge Graphs. In order to address this challenge, we have proposed KGEval, an instance of a novel crowdsourcing paradigm where dependencies among tasks presented to humans (BETs) are exploited. To the best of our knowledge, this is the first method of its kind. We demonstrated that the objective optimized by KGEval is in fact NP-Hard and submodular, and hence allows for the application of simple greedy algorithms with guarantees. Through extensive experiments on real datasets, we demonstrated effectiveness of KGEval. We hope to extend KGEval to incorporate varying evaluation cost, and also explore more sophisticated evaluation aggregation.

Acknowledgments

This research has been supported by MHRD (Govt. of India) and in part by a gift from Google.

A Appendix

Submodularity: A real valued function f, which acts over subsets of any finite set $\mathcal{H}$, is said to be *submodular* if $\forall R, S \subset \mathcal{H}$ it fulfills

$$f(R) + f(S) \geq f(R \cup S) + f(R \cap S).$$

We call potential function ψ as pairwise *regular* if for all pairs of BETs $\{p, q\} \in \mathcal{H}$ it satisfies

Proof. **(for Theorem 1)** The additional utility, in terms of label inference, obtained by adding a BET to larger set is lesser than adding it to any smaller subset. By construction, any two BETs which share a common factor node $\mathcal{C}_j$ are encouraged to have similar labels in G.

Potential functions ψ_j of Equation (3) satisfy pairwise regularity property i.e., for all BETs $\{p, q\} \in \mathcal{H}$

$$\psi(0,1) + \psi(1,0) \geq \psi(0,0) + \psi(1,1) \tag{4}$$

where $\{1, 0\}$ represent true/false. Equivalence of *submodular* and *regular* properties are established (Kolmogorov and Zabih, 2004; Jegelka and Bilmes, 2011). Using non-negative summation property (Lovász, 1983), $\sum_{j \in C} \theta_j \psi_j$ is submodular for positive weights $\theta_j \geq 0$.

We consider a BET h to be confidently inferred when the soft score of its label assignment in $\mathcal{I}(G, \mathcal{Q})$ is greater than threshold $\tau_h \in [0, 1]$. From above we know that $\mathbb{P}(l(h)|\mathcal{Q})$ is submodular with respect to fixed initial set $\mathcal{Q}$. Although $\max$ or $\min$ of submodular functions are not submodular in general, but (Kempe et al., 2003) conjectured that global function of Equation (1) is submodular if local threshold function $\mathbb{P}(h|\mathcal{Q}) \geq \tau_h$ respected submodularity, which holds good in our case of Equation (3). This conjecture was further proved in (Mossel and Roch, 2007) and thus making our global optimization function of Equation (1) submodular. □

Proof. **(for Theorem 2)** We reduce KGEval to NP-complete Set-cover Problem (SCP) so as to select $\mathcal{Q}$ which covers $\mathcal{I}(G, \mathcal{Q})$. For the proof to remain consistent with earlier notations, we define SCP by collection of subsets $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_m$ from set $\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$ and we want to determine if there exist k subsets whose union equals $\mathcal{H}$. We define a bipartite graph with $m + n$ nodes corresponding to $\mathcal{I}_i$'s and h_j's respectively and construct edge $(\mathcal{I}_i, h_j)$ if $h_j \in \mathcal{I}_i$. We need to find a set $\mathcal{Q}$, with cardinality k, such that $|\mathcal{I}(G, \mathcal{Q})| \geq n + k$.

Choosing our BET-set $\mathcal{Q}$ from SCP solution and further inferring evaluations of other remaining BETs using PSL will solve the problem in hand. □

A.1 Noisy Crowd Workers and Budget

Here, we provide a scheme to allot crowd workers so as to remain within specified budget and upper bound total error on accuracy estimate. We have not integrated this mechanism with Algorithm 1 to maintain its simplicity.

We resort to majority voting in our analysis and assume that crowd workers are not adversarial. So expectation over responses $r_h(u)$ for a task h with respect to multiple workers u is close to its true label $t(h)$ (Tran-Thanh et al., 2013), i.e.,

$$\left| \mathbb{E}_{u \sim \mathbb{D}(u,h)}[r_h(u)] - t(h) \right| \leq \frac{1}{2} \tag{5}$$

where $\mathbb{D}$ is joint probability distribution of workers u and tasks h.

Our key idea is that we want to be more confident about BETs h with larger inferable set (as they impact larger parts of KG) and hence allocate them more budget to post to more workers. We determine the number of workers $\{w_{h_1}, \ldots, w_{h_n}\}$ for each task such that h_t with larger inference set have higher w_{h_t}. For total budget B, we allocate

$$w_{h_t} = \left\lfloor \tfrac{B\ i_t\ (1-\gamma)}{c\ i_{max}} \right\rfloor$$

where i_t denotes the cardinality of inferable set $\mathcal{I}(G, \mathcal{Q} \cup h_t)$, c the cost of querying crowd worker, i_{max} the size of largest inferable set and $\gamma \in [0, 1]$ constant.

This allocation mechanism easily integrates with Algorithm 1; in (Line 8) we determine size of inferable set $i_t = |\mathcal{Q}_t|$ for task h and allocate w_h crowd workers. Budget depletion (Line 9) is modified to $B_r = B_r - w_h c(h)$. The following theorem bounds the error with such allocation scheme.

Theorem 3. [Error bounds] *The allocation scheme of redundantly posing h_t to w_{h_t} workers does not exceed the total budget B and its expected estimation error is upper bounded by $e^{-O(i_t)}$, keeping other parameters fixed. The expected estimation error over all tasks is upper bounded by $e^{-O(B)}$.*

Proof. Let $\gamma \in [0, 1]$ control the reduction in size of inferable set by $i_{t+1} = \gamma\, i_t$. By allocating w_{h_t} redundant workers for task $h_t, \forall t \in \{1 \cdots n\}$ with size of inferable set i_t, we incur total cost of

$$\begin{aligned}
\sum_{t=1}^{n} c\, w_{h_t} &= \sum_{t=1}^{n} \frac{B\ i_t\ (1-\gamma)}{c\ i_{max}} \cdot c \\
&= \left(\sum_{t=1}^{n} i_t\right) \cdot \left(\frac{B\ (1-\gamma)}{i_{max}}\right) \\
&= \left(\frac{i_{max}\ (1-\gamma^T)}{(1-\gamma)}\right) \cdot \left(\frac{B\ (1-\gamma)}{i_{max}}\right) \\
&\leq B
\end{aligned}$$

Note that the above geometric approximation helps in estimating summation $\sum_{t=1}^{n} i_t$ at iteration $t \leq n$.

Error Bounds: Here we show that the expected error of estimating of h_t for any time t decreases exponentially in the size of inferable set i_t. We use majority voting to aggregate w_{h_t} worker responses for h_t, denoted by $\hat{r}_{h_t} \in \{0, 1\}$

$$\hat{r}_{h_t} = \left\lfloor \frac{1}{w_{h_t}} \sum_{k=1}^{w_{h_t}} r_{h_t}(u_k) - \frac{1}{2} \right\rfloor + 1 \tag{6}$$

where $r_{h_t}(u_k)$ is the response by k^{th} worker for h_t. The error from aggregated response can be given by $\Delta(h_t) = |\hat{r}_{h_t} - t(h_t)|$, where $t(h_t)$ is its true label. From Equation (5) and Hoeffding-Azuma bounds over w_{h_t} i.i.d responses and error margin ε_t, we have

$$\begin{aligned}
\Delta(h_t) &= \mathbb{P}\left\{ \left| \frac{1}{w_{h_t}} \sum_{k=1}^{w_{h_t}} r_{h_t}(u_k) - \mathbb{E}(r_h(u)) \right| \geq \varepsilon_t \right\} \\
&= 2 \exp\left(-2\, \frac{B\ i_t\ (1-\gamma)}{c\ i_{max}}\ \varepsilon_t^2 \right)
\end{aligned}$$

For fixed budget B and given error margin ε_t, we have $\Delta(h_t) = e^{-O(i_t)}$. Summing up over all tasks t, by union bounds we get the total expected error from absolute truth as $\Delta(B) = \sum_{t=1}^{n} \Delta(h_t)$.

$$\begin{aligned}
\Delta(B) &\leq \sum_{t=1}^{n} 2 \exp\left(-2\, \frac{B\ i_t\ (1-\gamma)}{c\ i_{max}}\ \varepsilon_t^2 \right) \\
&\leq n \cdot 2 \exp\left(-2\, \frac{B\ i_{min}\ (1-\gamma)}{c\ i_{max}}\ \varepsilon_{min}^2 \right)
\end{aligned}$$

The accuracy estimation error will decay exponentially with increase in total budget for fixed parameters. □

References

AMT. https://www.mturk.com.

Jonathan Bragg, Daniel S Weld, et al. 2013. Crowdsourcing multi-label classification for taxonomy creation. In *HCOMP*.

Matthias Broecheler, Lilyana Mihalkova, and Lise Getoor. 2010. Probabilistic similarity logic. In *UAI*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD*, pages 601–610.

Marc Dupuis, Barbara Endicott-Popovsky, and Robert Crossler. 2013. An analysis of the use of amazons mechanical turk for survey research in the cloud. In *ICCSM2013-Proceedings of the International Conference on Cloud Security Management: ICCSM 2013*, page 10. Academic Conferences Limited.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, pages 413–422.

Stefanie Jegelka and Jeff Bilmes. 2011. Submodularity beyond submodular energies: coupling edges in graph cuts. In *CVPR*, pages 1897–1904.

Gjergji Kasneci, Jurgen Van Gael, Ralf Herbrich, and Thore Graepel. 2010. Bayesian knowledge corroboration with logical rules and user feedback. In *Machine Learning and Knowledge Discovery in Databases*, pages 1–18.

David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *SIGKDD*.

KGEval. http://talukdar.net/mall-lab/KGEval.

Vladimir Kolmogorov and Ramin Zabih. 2004. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159.

Andrey Kolobov, Daniel S Weld, et al. 2013. Joint crowdsourcing of multiple tasks. In *HCOMP*.

Frank R Kschischang, Brendan J Frey, and H-A Loeliger. 2001. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, pages 529–539.

László Lovász. 1983. Submodular functions and convexity. In *Mathematical Programming The State of the Art*, pages 235–257. Springer.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of AAAI*.

Elchanan Mossel and Sebastien Roch. 2007. On the submodularity of influence in social networks. In *ACM symposium on Theory of computing*, pages 128–134.

Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. 2011. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of WSDM*.

NELL. http://rtw.ml.cmu.edu/rtw/resources.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294.

PSL. http://www.psl.umiacs.umd.edu.

Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. 2013. Knowledge graph identification. In *The Semantic Web–ISWC 2013*, pages 542–557. Springer.

PyMLN. http://ias.cs.tum.edu/people/jain/mlns.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Mehdi Samadi, Partha Talukdar, Manuela Veloso, and Tom Mitchell. 2015. Askworld: budget-sensitive query evaluation for knowledge-on-demand. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 837–843. AAAI Press.

Yuyin Sun, Adish Singla, Dieter Fox, and Andreas Krause. 2015. Building hierarchies of concepts via crowdsourcing. *arXiv preprint arXiv:1504.07302*.

Long Tran-Thanh, Matteo Venanzi, Alex Rogers, and Nicholas R Jennings. 2013. Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks. In *AAMAS*.

YAGO. https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga.

Sparsity and Noise:
Where Knowledge Graph Embeddings Fall Short

Jay Pujara and **Eriq Augustine** and **Lise Getoor**
Department of Computer Science
University of California, Santa Cruz
jay@cs.umd.edu, eaugustine@ucsc.edu, getoor@soe.ucsc.edu

Abstract

Knowledge graph (KG) embedding techniques use structured relationships between entities to learn low-dimensional representations of entities and relations. One prominent goal of these approaches is to improve the quality of knowledge graphs by removing errors and adding missing facts. Surprisingly, most embedding techniques have been evaluated on benchmark datasets consisting of dense and reliable subsets of human-curated KGs, which tend to be fairly complete and have few errors. In this paper, we consider the problem of applying embedding techniques to KGs extracted from text, which are often incomplete and contain errors. We compare the sparsity and unreliability of different KGs and perform empirical experiments demonstrating how embedding approaches degrade as sparsity and unreliability increase.

1 Introduction

Recently knowledge graphs (KGs), structured representations of knowledge bases, have become an essential component of systems that perform question-answering (Berant et al., 2013), provide decision support, and enable exploration and discovery (Dong et al., 2014). Initial efforts to create KGs focused on structured information sources or relied extensively on manual curation. However, the diversity of knowledge available on resources like the World Wide Web have spurred many projects that tackle the more difficult task of automatically constructing KGs (Nickel et al., 2016a).

Unfortunately, information extraction approaches for KG construction must overcome complex, unreliable, and incomplete data. Many machine learning methods have been proposed to address the challenge of cleaning and completing KGs. One popular class of methods learn embeddings that translate entities and relationships into a latent subspace, then use this latent representation to derive additional, unobserved facts and score existing facts (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015).

Embedding methods have shown state-of-the-art results on several benchmark datasets. However, by construction, these benchmark datasets differ from data in real KGs. First, benchmark datasets have largely been restricted to the most frequently occurring entities in the KG. However in most KGs, entities are associated with a sparse set of observations. Second, benchmark datasets consist only of highly reliable facts from curated knowledge bases. In contrast, many KG construction projects extract knowledge from noisy data such as text or images, which introduces unreliable information.

In this paper, we evaluate popular KG embedding approaches on KGs that have sparse entities and unreliable candidate facts. We apply embedding methods to an extracted KG and modify existing benchmarks by varying the sparsity and reliability of training data used to learn embedding models. Using this suite of datasets, we characterize where embedding approaches are successful and the conditions that result in degrading results. Based on our insights, we provide recommendations for improving embedding models and identify promising areas of future exploration.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1751–1756
Copenhagen, Denmark, September 7–11, 2017.

Dataset	Triples	$\|E\|$	$\|R\|$	EE	RE	ED	RD	prec
Freebase	1B	124M	15K	14	3.2	16	68K	1
WordNet	380K	116K	27	21	2.3	7	21K	1
NELL1000	92M	4.8M	435	21	4.9	19	210K	0.45
FB15K	592K	15K	1.3K	16	5.1	79	440	1
WN18	151K	40K	18	19	2.1	7	8.4K	1
NELL165	1M	820K	221	25	1.5	3	4.7K	0.35

Table 1: Statistics of knowledge graphs and extracted datasets. Triples are the number of individual facts in the knowledge graph. $\|E\|$ and $\|R\|$ are unique entities and relations in the KG, respectively. EE and RE are measures of entropy, ED and RD measures of density, and prec is the precision of triples.

2 Background and Related Work

Diverse strategies for knowledge base construction include manually-crafted ontologies for common-sense reasoning (Lenat, 1995), community-driven collaborative efforts (Bollacker et al., 2008), ontology-based extraction from structured and textual sources (Mitchell et al., 2015), and "open" approaches that rely on textual information (Mausam et al., 2012). In this paper, we contrast the properties of two knowledge graphs that have clean, human-vetted facts with two knowledge graphs that are extracted from textual data.

Semantically meaningful embeddings of text have been a longstanding topic of study in NLP research (Turney and Pantel, 2010). More recently, knowledge graphs, which capture structured relationships between entities, has inspired methods such as matrix factorization (Riedel et al., 2013), tensor factorization (Nickel et al., 2011), and deep learning (Socher et al., 2013) that embed entities while preserving this relationship structure. We consider four state-of-the-art embedding methods (Bordes et al., 2013; Wang et al., 2014; Nickel et al., 2016b; Nguyen et al., 2016) and assess their performance on knowledge graphs with different properties.

3 Comparing Properties of KGs

In Table 1, we introduce three knowledge graphs and a parallel set of benchmark datasets derived from these KGs. Each KG takes the form of triples that specify a relationship between a subject and an object. The first two KGs, Freebase and WordNet, benefit from human curation that results in precisely defined entities and relationships and highly reliable facts. The third KG, NELL, is extracted from a large Web text corpus using an iterative co-training process and a pre-defined set of relations and types. Due to the iterative nature, NELL is a dynamic dataset and the table reports statistics of the 1000th iteration. FB15K and WN18, derived from Freebase and WordNet, respectively, have been used to train and evaluate many embedding strategies. NELL165, based on an earlier iteration of NELL, has been used as a benchmark for probabilistic models. We compare the vital statistics of these six datasets.

3.1 Size and Sampling

Despite the reliance on curation, Freebase is the largest KG with more facts ($\|T\|$), unique entities ($\|E\|$), and relationship types ($\|R\|$) than others. NELL, is a tenth the size of Freebase with substantially fewer entities and limited relations. WordNet, focused on NLP, is the smallest and expresses only 27 relationships between different words. The derived benchmark datasets are substantially smaller than the source KGs, with the largest, NELL, containing 1M facts. FB15K is generated by sampling a subset of the KG centered around 15K entities. WN18 is generated by restricting to 18 relations. NELL165 performs no sampling, but is limited by the comprehensiveness of patterns learned during training.

3.2 Diversity

To understand the distribution of entities and relationships in the KG, we introduce an entropy-based measure using the probability an entity or relation will occur in a randomly selected triple. For triples T of the form (s, p, o), relations R, entities E, We define the entity and relation probabilities as the probability that a randomly selected triple will contain a particular relation or entity. More formally, we define these probabilities:

$$P(r) = \frac{|t.p = r|}{\|T\|}\ ;\quad P(e) = \frac{|t.s = e| + |t.o = e|}{\|T\|}$$

Using these definitions, we define:

$$RE = \sum_{r \in R} -P(r) \log P(r)$$

We compute entity entropy (EE) and relation entropy (RE) for each dataset. Higher entropy values indicate more uniform distributions of facts across entities and relations, lower values signal biases in the facts. For example, the low RE values for Freebase and NELL165 are due to an abundance of facts specifying entity types (such as person), relative to other relations between entities. While Freebase has the most facts and entities, these facts are less diverse compared to other KGs. Through sampling, FB15K rebalances Freebase, increasing the diversity of entities and relations. In contrast, WordNet and WN18 have similar diversity statistics. Compared to NELL1000, NELL165 has a more diverse set of entities and a less diverse set of relations. All KGs have much higher EE than RE, since they use a manually defined set of relations but include many diverse entities.

3.3 Sparsity

In addition to diversity, KGs have differing levels of factual information for each entity or relation. One sparsity metric is information density, defined as the average triples per entity or relation. We formally define densities:

$$RD = \frac{\|T\|}{\|R\|}; \quad ED = \frac{2\|T\|}{\|E\|}$$

We compare the datasets using entity density (ED) and relational density (RD). Most datasets have a similar ED, but the benchmark dataset FB15K has much higher entity density while the benchmark dataset NELL165 has a much lower entity density. NELL1000 has the highest RD, since extractions are focused on a small set of relations, while FB15K has a particularly low RD value due to the entity-centric approach to construction. We note that FB15K has much higher ED and much lower RD than parent Freebase, due to the sampling choices made during its construction.

3.4 Reliability

Embedding approaches rely on using facts that are reliable. Human-curated KGs generally have high precision due to strong oversight. In contrast, extracted KGs are far noisier, including erroneous relationships between entities. Extracted KGs are often evaluated on small, manually-labeled evaluation sets to estimate precision. In recent evaluations (Mitchell et al., 2015) using 11K annotations, NELL facts had a precision of ranging from 0.75-0.85 for confident extractions and 0.35-0.45 across the broader set of extractions.

4 Empirical Evaluation

To better understand embedding performance with sparse and unreliable data, we select four popular embedding approaches and perform four empirical analyses. We evaluate embedding techniques TransE (Bordes et al., 2013), TransH (Wang et al., 2014), HolE (Nickel et al., 2016b), and STransE (Nguyen et al., 2016), that use increasingly sophisticated learning methods to represent entities and relations. To learn embeddings, we used the public implementations of Lin et al. (2015); Nickel et al. (2016b); Nguyen et al. (2016). We conduct four experiments to characterize the performance of these embeddings methods. The first set of experiments evaluate the performance of embeddings on the extracted NELL165 KG. The second set of experiments modify the existing FB15K benchmark to isolate the impact of sparsity on embedding quality. The third set of experiments decrease the reliability of FB15K and determine how performance degrades as a result. The final experiments explore the tradeoff between sparsity and reliability by beginning with a sparse trainng set and incrementally adding unreliable triples at differing noise levels.[1]

[1] Code for experiments is available at `https://www.github.com/linqs/pujara-emnlp17`

Method	AUPRC	F1
Baseline	0.873	0.828
NELL	0.765	0.673
TransH	0.701	0.783
HolE	0.710	0.783
TransE	0.726	0.783
STransE	0.784	0.783
PSL-KGI	**0.891**	**0.848**

Table 2: Embedding performance on the sparse and noisy NELL165 benchmark is poor, failing to beat a baseline that simply selects the top extractions, and substantially underperforming probabilistic models.

4.1 Extracted Knowledge Graphs

In Section 3, we noted that the extracted NELL165 dataset is sparse, with fewer (candidate) facts per relation or entity than the FB15K benchmarks. Moreover, the precision of these candidates can be far lower than benchmark datasets. To evaluate whether embeddings can succeed under such challenging conditions, we applied four state-of-the-art embedding techniques,

We evaluated all methods on 4.5K manually-labeled facts (Jiang et al., 2012), reporting the area under the precision-recall curve (AUPRC) and the F1 score, computed with parameters that maximize performance on the labeled training set. We compare against a baseline that simply applies a threshold to NELL extractor confidences (but cannot score novel facts), the NELL promotion strategy, and a probabilistic approach PSL-KGI (Pujara et al., 2015), that reasons collectively about KG facts using ontological constraints and supports open-world reasoning. The results, in Table 2, suggest that embedding approaches cannot cope with the sparse and low-quality extractions, performing more poorly than the baseline approaches and substantially trailing the probabilistic model. In the next two experiments, we analyze whether this failure can be attributed to sparsity or sensitivity to noise.

4.2 Sensitivity to Sparsity

One potential explanation for the lackluster performance of embedding approaches on extracted KGs is the sparsity of these datasets. To assess the impact of sparsity on the quality of learned embeddings, we remove triples from FB15K using two different techniques. The first technique, `sparse`, removes triples uniformly at random, with a constraint that such removal does not eliminate any entity or relation from the dataset. The second technique, `stable`, removes all triples for a particular relation, leaving other relations intact. `stable` is calibrated so that the training set size does not vary more than 2% between techniques.

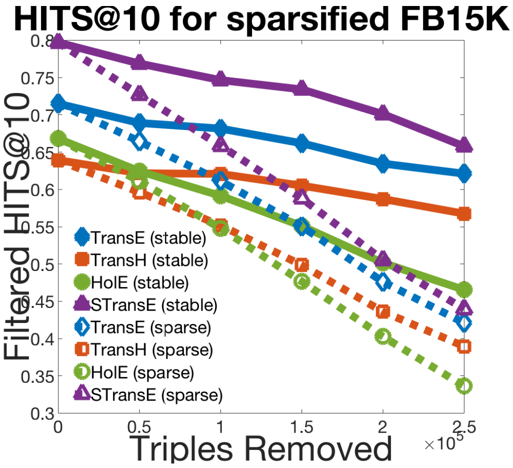


Figure 1: Triples are removed from FB15K to preserve relational density (`stable`, solid) or to increase sparsity (`sparse`, dotted). Sparse training sets have a pronounced impact on the learned embedding, as measured by HITS@10 on the test set.

Fig. 1 shows the filtered hits@10 metric (proportion of correct triples in top ten triples excluding training data) for both `sparse` and `stable` using the TransE, TransH, HolE, and STransE embeddings. Performance universally decreases as the training set diminishes. However, in the `sparse` treatment, performance deteriorates much more rapidly than in `stable`. Our experiments show that more complex representations such as TransH and HolE suffer more from sparsity, while TransE and the more sophisticated STransE have somewhat better performance. Ultimately, when half the triples have been randomly removed, corresponding to a (relatively high) RD value of 220, the `stable` outperforms `sparse` by as much as 60%. The contrast between a dense set of facts for each relation (`stable`) and a sparse set of relational training data is a vivid demonstration that embedding quality relies on dense training data.

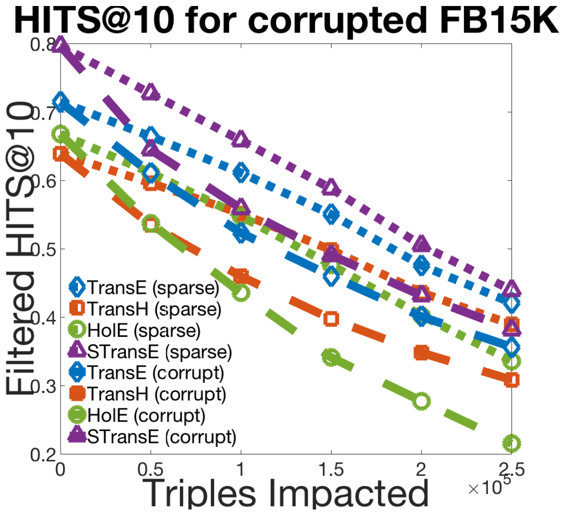


Figure 2: Randomly corrupting triples (`corrupt`, dashed) during training decreases embedding quality relative to randomly removing triples (`sparse`, dotted).

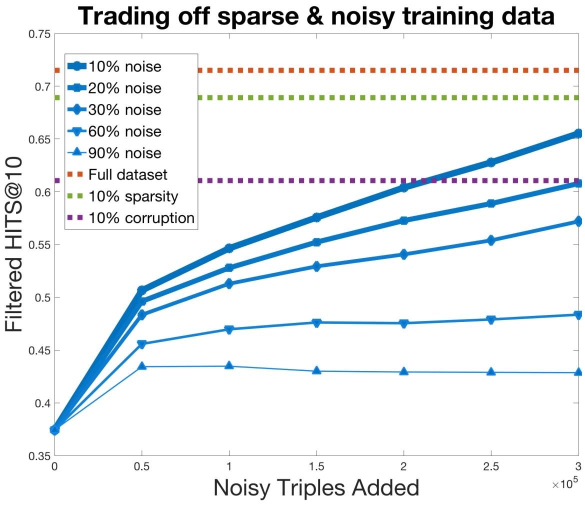


Figure 3: Starting with a sparse training set, adding unreliable triples can help embedding performance recover if the noise level is low.

4.3 Sensitivity to Unreliability

Beyond sparsity, candidate facts generated by knowledge extraction approaches can also be unreliable. To understand the sensitivity of embedding techniques to noise, we modified the FB15K dataset to include unreliable triples. Our approach to introducing noise, `corrupt` involved "corrupting" triples, substituting a replacement entity or relation for the true subject, predicate or object. The embedding approach is then trained with a corrupted version of the benchmark. Fig. 2 show how the Hits@10 metric suffers as increasing numbers of facts are either corrupted (`corrupt`) or removed (`sparse`). We find that across all methods, removing training data is better than providing incorrect training data to the learning algorithm, but surprisingly the deficit between `sparse` and `corrupt` remains relatively stable across all embeddings.

4.4 Trading off Sparsity and Noise

In many real-world scenarios, constructing a KG requires navigating a tradeoff between sparsity and noise. A sparse, high-quality set of extractions may be insufficient to learn meaningful embeddings. However, the benefit of incorporating additional, unreliable facts may also be questionable. We explore this tradeoff by randomly removing $300K$ triples from FB15K and incrementally adding unreliable triples at differing noise levels, where noise measures the probability a newly-added training triple is corrupted. We generate training sets for each noise level and size, train TransE, and compute the filtered Hits@10 metric on the test set. Fig. 2 shows all embeddings have an initial benefit from new training data, but noise level dictates the improvement as more data is introduced. For low noise settings, performance climbs steadily, while higher noise results in plateauing or diminishing performance. Surprisingly, even with 90% noise embeddings demonstrate a small net improvement, suggesting that for embedding methods a large, unreliable corpus may be better than an extremely sparse, high-quality one.

5 Conclusion

In this paper, we analyze several knowledge graphs and discuss key metrics for diversity, sparsity, and unreliability in realistic KGs. Our experimental evaluation concludes that KG embeddings are sensitive to sparse and unreliable data, and perform poorly on KGs extracted from text. These findings suggest a rich area of future research, determining new strategies to extend embeddings to cope with sparse and unreliable data. Three promising approaches include revising the closed-world assumption frequently used in training embeddings, combining embeddings and collective probabilistic models that perform well on extracted KGs, and devising an optimization approach for embeddings that exploits confidence from knowledge extraction systems.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *SIGMOD*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610. ACM.

Shangpu Jiang, Daniel Lowd, and Dejing Dou. 2012. Learning to Refine an Automatically Extracted Knowledge Base Using Markov Logic. In *ICDM*.

Douglas B Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI*.

Mausam, Michael D. Schmitz, Robert E. Bart, Stephen Soderland, and Oren Etzioni. 2012. Open Language Learning for Information Extraction. In *EMNLP*.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-Ending Learning. In *AAAI*.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. STransE: A Novel Embedding Model of Entities and Relationships in Knowledge Bases. In *NAACL-HLT*.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016a. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE*, 104(1).

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016b. Holographic Embeddings of Knowledge Graphs. In *AAAI*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. 2015. Using Semantics & Statistics to Turn Data into Knowledge. *AI Magazine*, 36(1):65–74.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL-HLT*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *JAIR*, 37(1).

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*.

Dual Tensor Model for Detecting Asymmetric Lexico-Semantic Relations

Goran Glavaš and **Simone Paolo Ponzetto**
Data and Web Science Group
University of Mannheim
B6, 26, DE-68159 Mannheim, Germany
{goran, simone}@informatik.uni-mannheim.de

Abstract

Detection of lexico-semantic relations is one of the central tasks of computational semantics. Although some fundamental relations (e.g., hypernymy) are asymmetric, most existing models account for asymmetry only implicitly and use the same concept representations to support detection of symmetric and asymmetric relations alike. In this work, we propose the Dual Tensor model, a neural architecture with which we explicitly model the asymmetry and capture the translation between unspecialized and specialized word embeddings via a pair of tensors. Although our Dual Tensor model needs only unspecialized embeddings as input, our experiments on hypernymy and meronymy detection suggest that it can outperform more complex and resource-intensive models. We further demonstrate that the model can account for polysemy and that it exhibits stable performance across languages.

1 Introduction

Detection of semantic relations that hold between words is the central task of lexical semantics, tightly coupled with obtaining representations that capture meaning of words (Mikolov et al., 2013; Wieting et al., 2015; Mrkšić et al., 2016, *inter alia*). As such, robust detection of lexico-semantic relations may benefit virtually any natural language processing application.

Because lexico-semantic knowledge bases (KBs) like WordNet (Fellbaum, 1998) are general and of limited coverage, numerous methods for detecting lexico-semantic relations rely on distributional word representations obtained from large corpora. Although distributional models have evolved over time, from count-based (Landauer et al., 1998) and generative (Blei et al., 2003) to prediction-based (Mikolov et al., 2013), the similarity between distributional vectors still indicates only the abstract semantic association and not a precise semantic relation (e.g., vectors of antonyms may be as similar as vectors of synonyms).

Consequently, a number of approaches have been proposed for specializing distributional spaces for specific lexico-semantic relations, either by (1) modifying the learning objective or regularization of the original embedding model by incorporating linguistic constraints (Yu and Dredze, 2014; Kiela et al., 2015) or (2) retroactively fitting the pre-trained unspecialized embeddings to linguistic constraints (Faruqui et al., 2015; Mrkšić et al., 2016). However, these methods specialize distributional vector spaces primarily for detecting the symmetric relation of *semantic similarity* (i.e., graded synonymy) and not for asymmetric lexico-semantic relations such as *hypernymy* and *meronymy*. On the other hand, models for embedding KBs (Bordes et al., 2013; Socher et al., 2013; Yang et al., 2015) uniformly model both symmetric and asymmetric relations. They learn a single vector representation (i.e., embedding) for each KB concept, assuming implicitly that the same concept representation is equally useful for predicting symmetric and asymmetric relations alike.

Relation-specific learning-based models have, to the largest extent, targeted hypernymy. Distributional models predict the hypernymy relations by combining raw distributional vectors of concepts in a pair (Baroni et al., 2012; Roller et al., 2014; Santus et al., 2014), whereas path-based models base predictions on lexico-syntactic paths from co-occurrence contexts obtained from a large corpus (Snow et al., 2004; Nakashole et al., 2012; Shwartz et al., 2016). Shwartz et al. (2016) combine the path-based and distributional models to

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1757–1767
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

reach state-of-the-art performance in hypernymy detection. Both distributional and path-based methods, however, model asymmetry only implicitly (e.g., via the order of embeddings in the concatenation). Besides, path-based models are language-dependent since they require syntactically preprocessed data as input.

In this work, we propose the Dual Tensor model, a neural architecture that (1) models asymmetry more explicitly than existing models and (2) explicitly captures the translation of unspecialized distributional vectors into specialized embeddings better suited to detect the asymmetric relation of interest. The Dual Tensor model can be considered distributional as it requires only distributional vectors of words as input. Consequently, in contrast to path-based methods, it is language-independent and more widely applicable. Experimental results on hypernymy and meronymy detection show that the Dual Tensor model outperforms both distributional and path-based models. We additionally demonstrate that our approach exhibits stable performance across languages and can, to some extent, diminish the negative effects of polysemy.

2 Related Work

Specializing Word Embeddings. Unspecialized word embeddings (Mikolov et al., 2013; Pennington et al., 2014) capture general semantic properties of words, but are unable to differentiate between different types of semantic relations (e.g., vectors of *car* and *driver* might be as similar as vectors of *car* and *vehicle*). However, we often need embeddings to be similar only if an exact lexico-semantic relation holds between the words. Numerous methods for specializing word embeddings for particular relations have been proposed (Yu and Dredze, 2014; Faruqui et al., 2015; Kiela et al., 2015; Mrkšić et al., 2016, *inter alia*), primarily aiming to differentiate synonymic similarity from other types of semantic relatedness.

Some methods modify the objective or regularization of general embedding algorithms like CBOW or skip-gram (Mikolov et al., 2013) in order to directly train relation-specific embeddings from large corpora. Yu and Dredze (2014) extend the CBOW objective with synonymy constraints from WordNet and Paraphrase Database (PPDB) (Ganitkevitch et al., 2013). Similarly, Kiela et al. (2015) add synonyms as additional contexts for the skip-gram objective.

Other models update the whole unspecialized embedding space by moving closer together vectors of words standing in a particular relation. Starting with unspecialized embeddings of concepts, Faruqui et al. (2015) run a belief propagation algorithm on a graph induced from WordNet or PPDB. Wieting et al. (2015) couple an objective maximizing the similarity of PPDB pairs with the smart selection of the negative examples. Mrkšić et al. (2016) take this idea further by using antonym pairs from WordNet as negative examples.

All aforementioned models either directly train specialized embeddings or derive them by updating the unspecialized embeddings. In contrast, via dual tensors, we explicitly capture the function that transforms unspecialized embeddings to specialized embeddings that are better suited to detect the asymmetric relation of interest.

Embedding Knowledge Graphs. Recently, various models for embedding KB concepts and relations have been proposed (Bordes et al., 2013; Socher et al., 2013; Yang et al., 2015; Nickel et al., 2016, *inter alia*). These models predict existence of relations between entities by arithmetically combining concept vectors and relation matrices or tensors. The scoring functions of KG embedding models combine the concept embeddings via linear product (i.e., relation tensor multiplies the concatenation of concept vectors of the two entities) (Bordes et al., 2011), bilinear product (i.e., relation tensor first multiplies the left concept embedding and the result multiplies the embedding of the second concept) (Yang et al., 2015), or the combination of the two (Socher et al., 2013). Both linear and bilinear scoring functions implicitly model asymmetry as they are not commutative with respect to concept embeddings. In this work, we choose to leverage the bilinear product in our model, following the findings of Yang et al. (2015) who report bilinear product outperforming other scoring combinations.

KG embedding models employ the same concept embeddings for predicting all relations, symmetric and asymmetric alike. By directly updating concept embeddings in training, they cannot make relation predictions for concepts outside of the training set.

Hypernymy and Meronymy Detection. Hypernymy and meronymy are arguably the two most prominent asymmetric lexico-semantic relations. Methods for their detection can roughly be classified as either distributional or path-based. Path-based methods consider lexico-syntactic paths connecting pairs of words in their co-occurrence contexts in large corpus. Early approaches, e.g., Hearst (1992) for hypernymy and Berland and Charniak (1999) for meronymy, exploited a small set of manually created lexico-syntactic patterns that imply a relation of interest (e.g., *a such as b*). Subsequent approaches looked at ways to eliminate the need for manual compilation of extraction patterns. Pantel and Pennacchiotti (2006) and Girju et al. (2006) proposed bootstrapping approaches to meronymy detection, starting from a seed set of part-whole pairs. Snow et al. (2004) provided all dependency paths connecting the concepts in corpus to a logistic regression classifier for hypernymy detection.

Distributional methods detect asymmetric relations using only distributional vectors of words as input. Distributional models come in both unsupervised and supervised flavors. Unsupervised metrics for hypernymy detection assume either that the hyponym's contexts are included in the hypernym's contexts (Weeds and Weir, 2003; Kotlerman et al., 2010) or that the linguistics contexts of a hyponym are more informative than the contexts of its hypernyms (Rimell, 2014; Santus et al., 2014). Supervised hypernymy classifiers represent the pair of words by combining their distributional vectors in different ways – concatenating them (Baroni et al., 2012) or subtracting them (Roller et al., 2014) – and feeding the resulting vector to a supervised classifier like logistic regression. Most recently, Shwartz et al. (2016) coupled path-based and distributional information with a recurrent neural network (RNN), yielding state-of-the-art hypernymy detection performance. Although our Dual Tensor model is purely distributional, we show that it may outperform such a hybrid model which additionally exploits syntactic information.

Distributional and path-based models have been used to discriminate between multiple lexico-semantic relations, including hypernymy and meronymy, at once (Santus et al., 2016; Shwartz and Dagan, 2016). However, as pointed out by (Chersoni et al., 2016), distributional vectors and scores based on their comparison fail to discriminate between multiple relation types at once. In this work, we focus on binary classification for a single relation (hypernymy and meronymy) at a time.

3 Dual Tensor Model

The following assumptions and desirable properties guided the design of the Dual Tensor model for detection of asymmetric lexico-semantic relations:

(1) Unspecialized distributional vectors are not good signals for detecting specific lexico-semantic relations. We thus need to derive specialized representations that are better suited for detecting the specific asymmetric relation of interest.

(2) The transformation from unspecialized distributional vectors of words to their relation-specialized embeddings should be captured explicitly, via a well-defined transformation function. Having an explicit embedding specialization function alleviates the need to specialize the entire unspecialized embedding space at once, like existing models do (Faruqui et al., 2015; Mrkšić et al., 2016).

(3) Each concept should have two different relation-specialized embeddings – one for each end of an asymmetric relation. For instance, for hypernymy, the concept's specialized embedding for pairs in which it is considered to be a hyponym (e.g., *dog* in *dog–animal*) should differ from its embedding in pairs in which it is tested as a hypernym (e.g., *dog* in *maltese–dog*).

(4) An unspecialized distributional vector of the word might – for each end of the asymmetric relation – be transformed into several specialized vectors instead of only one. This way the model may implicitly account for polysemy – i.e., different specialized vectors might capture asymmetric properties of different senses of polysemous words. E.g., the hyponym properties of *bank* in the pair *bank* vs. *building* may be different from those in the pair *bank* vs. *company*).

Figure 1 depicts the overall architecture of the Dual Tensor model, incorporating all four of above-mentioned design guidelines.

3.1 Dual Tensors

For a given pair of concepts (c_1, c_2), Dual Tensor model computes the score $s(c_1, c_2)$ indicating the likelihood that an asymmetric lexico-semantic relation holds between the concepts (e.g., for meronymy, how likely it is that c_1 is a *part of* c_2). The model takes as input the unspecialized embeddings of the two concepts, e_1 and e_2. For single-word concepts these are simply pre-trained word embeddings, whereas for multi-word concepts, similar to (Socher et al., 2013), we average the pre-trained embeddings of constituent words.

The unspecialized input embeddings are next translated into specialized embeddings, meant to

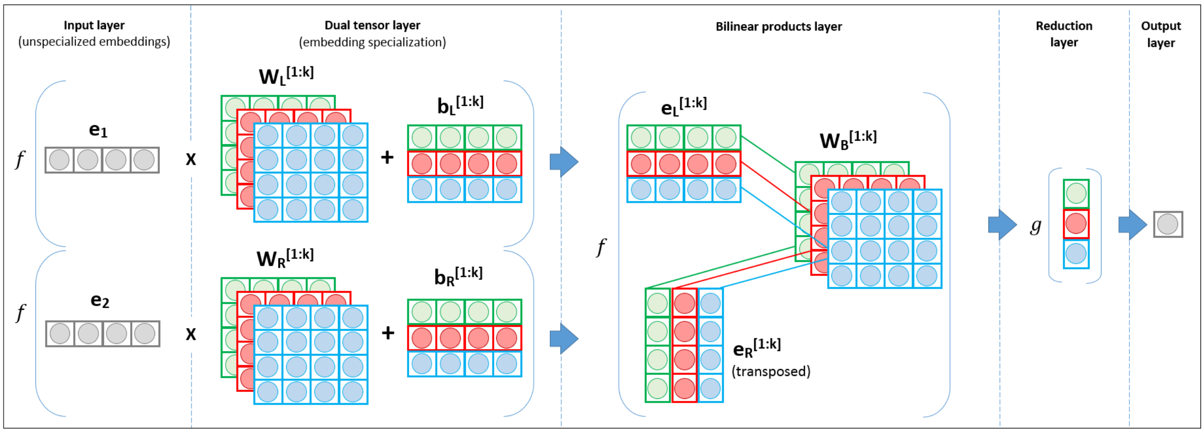


Figure 1: The architecture of the Dual Tensor model.

better capture the existence of the asymmetric relation between the concepts, via *specialization tensors*. By introducing dedicated tensors we – unlike existing models, which directly propagate updates to unspecialized embeddings (Faruqui et al., 2015; Mrkšić et al., 2016) – explicitly learn the specialization function. With an explicit specialization function, we do not have to specialize the whole embedding space at once. Also, unlike KG completion models (Bordes et al., 2013; Socher et al., 2013), we can make predictions for pairs involving concepts unseen in the training data.

We explicitly model asymmetry by introducing two specialization tensors (hence the model name) that differently specialize the unspecialized input embeddings of concepts. The left tensor, $\mathbf{W_L^{[1:k]}}$ (with the corresponding set of bias vectors $\mathbf{b_L^{[1:k]}}$), specializes the concept embedding if the concept is the first element of the pair, whereas the right tensor, $\mathbf{W_R^{[1:k]}}$ (with bias vectors $\mathbf{b_R^{[1:k]}}$), specializes the concept embedding when the concept is the second element of the pair:

$$\mathbf{e_L^{[1:k]}} = tanh\left(e_1\mathbf{W_L^{[1:k]}} + \mathbf{b_L^{[1:k]}}\right)$$
$$\mathbf{e_R^{[1:k]}} = tanh\left(e_2\mathbf{W_R^{[1:k]}} + \mathbf{b_R^{[1:k]}}\right)$$

When predicting hypernymy, for example, dual tensors ensure that the specialized representation for concept *cat* in pairs like *cat–animal* differs from its specialized representation in pairs like *birman–cat*.

Specialization tensors map an unspecialized embedding into a set of k specialized embeddings – each slice of the tensor, W_L^i (W_R^i), together with the corresponding bias vector b_L^i (b_R^i), produces one specialized vector e_L^i (e_R^i). By using specialization tensors with k slices instead of specialization matrices we make the model more general. The tensor-based model trivially degrades to the matrix-based model by setting $k = 1$. We obtain the final specialized representation of a concept by non-linearly transforming (hyperbolic tangent) the product of an unspecialized input embedding and the specialization tensor.[1]

3.2 Bilinear Product and Scoring

Using dual tensors, we transform unspecialized embeddings into asymmetrically specialized representations – sets of specialized vectors – which we next use to predict whether the asymmetric relation holds between the concepts. Our scoring function is based on bilinear products between (1) specialized vectors $\mathbf{e_L^{[1:k]}}$ of the first concept, (2) relation tensor $\mathbf{W_B^{[1:k]}}$, and (3) specialized vectors $\mathbf{e_R^{[1:k]}}$ of the second concept. For each pair of specialized vectors e_L^i and $e_R^i, i \in \{1, \dots, k\}$, we compute the bilinear product score, using the corresponding slice W_B^i of the relation tensor $\mathbf{W_B^{[1:k]}}$:

$$b^i = e_L^i W_B^i (e_R^i)^T.$$

The final relation score $s(c_1, c_2)$ for a given pair of concepts is computed by reducing the vector of bilinear product scores $\mathbf{b}$ to the mean value (function g in Figure 1)[2] and non-linearly bounding the resulting score to the $[-1, 1]$ range:

$$s(c_1, c_2) = tanh\left(\frac{1}{k}\sum_{i=1}^{k} b^i\right).$$

[1] Preliminary experiments without applying a non-linear transformation yielded consistently poorer performance.

[2] We also experimented with min- and max-reduction, but the reduction to the mean yielded best preliminary results.

3.3 Optimization

Dual Tensor model is parametrized by the specialization tensors, their corresponding bias vectors, and the relation tensor, namely, $\mathbf{\Omega} = \{\mathbf{W}_{\mathbf{L}}^{[1:k]}, \mathbf{W}_{\mathbf{R}}^{[1:k]}, \mathbf{b}_{\mathbf{L}}^{[1:k]}, \mathbf{b}_{\mathbf{R}}^{[1:k]}, \mathbf{W}_{\mathbf{B}}^{[1:k]}\}$. Let A be the set of concept pairs in the training set, $A = \{p^i = (c_1^i, c_2^i)\}_{i=1}^N$. We learn model's parameters by minimizing the margin-based objective:

$$J(\mathbf{\Omega}) = \lambda\|\mathbf{\Omega}\|_2 + \sum_{p^i \in A} max\left(0, 1 - s(p^i) \cdot y(p^i)\right)$$

where $s(p^i)$ is model's prediction for the pair (c_1^i, c_2^i), $y(p^i) \in \{-1, 1\}$ is the true label of that pair, and λ is the regularization coefficient. In all our experiments, we trained the model in mini-batches, optimizing the parameters with the RMSProp algorithm (Tieleman and Hinton, 2012).

The model has three hyperparameters: the length of the unspecialized input embeddings l, the number of tensor slices k, and the regularization factor λ. We optimize the hyperparameters (together with the starting learning rate value) via grid-search, by maximizing performance on the validation portion of each dataset. In all our experiments, except the multilingual comparison (Section 5.3), we evaluated variants of the Dual Tensor model using pre-trained English GloVe word embeddings (Pennington et al., 2014) with varying length, $l \in \{50, 100, 200, 300\}$ and tensors with $k \in \{1, \ldots, 5\}$ slices. In most experiments, the optimal configuration was $l = 300$ and $k = 3$.

4 Evaluation

We evaluate the Dual Tensor model on several datasets for detecting hypernymy and meronymy, two arguably most prominent asymmetric lexico-semantic relations. In all experiments, we compare the model's performance with state-of-the-art results on respective datasets. Additionally, aiming to quantify the effects that different components of the Dual Tensor model have on prediction performance, we evaluate two reduced models variants.

4.1 Datasets

We evaluate the Dual Tensor model on the following hypernymy and meronymy detection datasets:

HypeNet dataset. Arguing that existing datasets were too small for training their recurrent network, Shwartz et al. (2016) compiled this dataset for hypernymy detection from several external KBs, taking only pairs of concepts in direct relation (i.e., no transitive closure).

Other hypernymy detection datasets. We additionally evaluate the Dual Tensor model on four smaller datasets for hypernymy detection: (1) BLESS dataset (Baroni and Lenci, 2011) and EVALuation dataset (Santus et al., 2015) contain instances of hypernymy and four other relations. BLESS additionally contains random word pairs; (2) Weeds dataset (Weeds et al., 2014) contains hypernymy and co-hyponymy pairs; (3) Benotto dataset (Benotto, 2015) couples hypernymy pairs with synonymy and antonymy pairs. Because these datasets contain at most several thousand pairs, we only use them to evaluate the performance of models trained on larger datasets;

WN-Hy and WN-Me datasets. We create these datasets by taking concept pairs from WordNet. We take all instances from the transitive closure of hypernymy (all parts of speech) and meronymy (nouns) relations and couple them with all synonym and antonym relations (all parts of speech), as well as lexical entailment relations (verbs).

For the WN-Hy dataset we designate all hypernymy relations (i.e., both direct and indirect) as positive instances and their inverses (i.e., hyponymy relations) together with all other relations as negative instances. Finally, we balance the dataset by randomly sampling negative instances to match the number of positive instances. Analogously, we create the WN-Me dataset by taking meronymy relations as positive instances. We compile three different WN-Hy datasets: WN-Hy-EN using English WordNet (Fellbaum, 1998), WN-Hy-ES using Spanish WordNet (Gonzalez-Agirre et al., 2012), and WN-Hy-FR using French WordNet (Sagot and Fišer, 2008). To allow for fair comparison of model's performance across languages, we randomly sample two larger dataset (English and French) to match in size the smallest (Spanish).

Lexical and Random Splits. Levy et al. (2015) showed that supervised distributional models for classifying lexico-semantic relations suffer from overfitting in settings with significant lexical overlap between the training and test set. In such settings models tend to learn properties of individual words (e.g., that a word is a prototypical hypernym) instead of relations between words. The reported results on such datasets are thus overly optimistic estimates of models' true performance.

Dataset	Train	Val.	Test
HypeNet (rand)	49.5K (20%)	3.5K (19%)	17.7K (20%)
HypeNet (lex)	20.3K (20%)	1.4K (20%)	6.6K (20%)
BLESS	–	2.7K (5%)	23.9K (5%)
EVALuation	–	1.4K (24%)	12.3K (27%)
Weeds	–	293 (50%)	2.6K (50%)
Benotto	–	501 (41%)	4.5K (38%)
WN-Hy-EN	103K (50%)	15K (50%)	30K (50%)
WN-Hy-EN	103K (50%)	15K (50%)	30K (50%)
WN-Hy-FR	103K (50%)	15K (50%)	30K (50%)
WN-Me (rand)	13.9K (50%)	2K (50%)	4K (50%)
WN-Me (lex)	7.9K (50%)	208 (50%)	318 (50%)

Table 1: Datasets used in evaluation.

	Lex. split			Rand. split		
Model	P	R	F_1	P	R	F_1
HypeNet path-based	69.1	63.2	66.0	81.1	71.6	76.1
HypeNet hybrid	**80.9**	61.7	70.0	91.3	**89.0**	**90.1**
CONCAT-SVM	75.4	55.1	63.7	90.1	63.7	74.6
BILIN-PROD	53.1	53.3	53.2	74.0	79.4	76.6
SINGLE-T	68.4	70.0	69.2	84.8	86.7	85.7
DUAL-T	70.5	**78.5**	**74.3**	**93.3**	82.6	87.6

Table 2: Hypernymy classification performance.

To eliminate the effect of lexical memorization, Levy et al. (2015) propose dataset splits with no lexical overlap between the train and test portions. However, model's performance in a lexically-split setting is an overly pessimistic estimate of models' true performance – in a realistic scenario, the model will occasionally make predictions for pairs involving some of the concepts from the training set. Because the true model performance is likely between the performance on a randomly-split and performance on a lexically-split dataset, we report models' performance in both of these settings.

We show the sizes of all dataset variants used in our experiments in Table 1. We additionally report the proportion of positive instances (in brackets), as this percentage directly affects some evaluation metrics (precision, F_1-score, average precision).

4.2 Baselines

In addition to specific models yielding best performance on particular datasets, we compare the Dual Tensor model (DUAL-T) with these baselines:

Supervised distributional baseline (CONCAT-SVM). We train SVM model with RBF kernel on concatenation of unspecialized concept embeddings (Baroni et al., 2012), following Levy et al. (2015), who report this model outperforming other types of embedding composition;

Bilinear product (BILIN-PROD). This model is the simple bilinear product between the unspecialized concept embeddings, parametrized only by the relation matrix W_B. That is, the prediction score for a pair of concepts is given as $s(c_1, c_2) = e_1 W_B e_2^T$. The bilinear model implicitly captures asymmetry by learning a non-symmetric relation matrix W_B. By comparing the performances of BILIN-PROD and DUAL-T, we jointly quantify the effects of (1) explicit modeling of asymmetry and (2) relation-specific embedding specialization;

Single tensor model (SINGLE-T). This is the reduction of the Dual Tensor model in which we use only one specialization tensor, i.e., $\mathbf{W_L^{[1:k]}} = \mathbf{W_R^{[1:k]}}$. In other words, SINGLE-T model always specializes the unspecialized embedding of a concept the same way, regardless of the concept's position in a candidate pair. By comparing the performance of the DUAL-T model with that of SINGLE-T, we measure the effect of asymmetrically specializing unspecialized embeddings.

Same as for the DUAL-T model, we optimize the hyperparameters of the baselines on the validation portions of the datasets used for evaluation.

4.3 Classification Experiments

Binary classification is the most straightforward evaluation setting for relation detection models. For a pair of concepts, we make the binary asymmetric relation prediction $r_a(c_1, c_2)$ simply by thresholding the model's prediction scores, i.e., $r_a(c_1, c_2) = I\{s(c_1, c_2) > 0\}$, where I is the indicator function.

Hypernymy classification. We first evaluate the DUAL-T model and the baselines on the HypeNet dataset (Shwartz et al., 2016). We show the performance of the DUAL-T model in Table 2, together with the path-based and hybrid (combination of path-based and distributional signal) variants of the the state-of-the-art RNN model of Shwartz et al. (2016). On the more challenging, lexically-split dataset DUAL-T model significantly[3] outperforms the more complex hybrid HypeNet model (Shwartz et al., 2016), an RNN model coupling representations of syntactic paths from a large corpus with

[3] All performance differences were tested using the non-parametric stratified shuffling test (Yeh, 2000) with $\alpha = 0.05$.

	Lex. split			Rand. split		
Model	P	R	F_1	P	R	F_1
Concat-SVM	**78.6**	44.6	56.9	79.9	75.9	77.9
Bilin-Prod	73.3	50.0	59.4	81.0	79.8	80.5
Single-T	77.7	55.5	64.8	85.7	82.6	84.1
Dual-T	76.5	**61.1**	**67.9**	**87.7**	**85.3**	**86.5**

Table 3: Meronymy classification performance.

unspecialized concept embeddings. In both settings Dual-T outperforms Single-T which, in turn, outperforms Bilin-Prod. This empirically justifies both our explicit modeling of asymmetry and relation-specific embedding specialization.

Meronymy classification. We next evaluate the meronymy classification performance of the models on the WN-Me dataset. The results are shown in Table 3. Same as in the case of hypernymy classification, Dual-T significantly outperforms all three baselines, with Single-T outperforming Bilin-Prod. All distributional models we evaluate achieve poorer performance on meronymy than hypernymy detection, especially considering that WN-Me is a balanced dataset, whereas HypeNet is heavily skewed towards negative instances.

4.4 Ranking Experiments

Shwartz et al. (2017) propose ranking as an alternative evaluation setting for hypernymy detection. The goal is to rank positive relation pairs higher than negative ones. Our Dual-T model (and associated baselines) rank the concept pairs in decreasing order of assigned relations scores $s(c_1, c_2)$. Following Shwartz et al. (2017), we report performance in terms of overall average precision (AP) and average precision at rank 100 (AP@100).

Hypernymy ranking. We evaluate the ranking performance on four small hypernymy test sets: BLESS, EVALuation, Benotto, and Weeds (cf. Table 1). As these datasets are not big enough to train neural models, we train all models on the HypeNet dataset. For each test set we eliminate the lexical overlap by removing from the HypeNet dataset pairs containing any concept from that test set.

Table 4 displays ranking performance for Dual-T model, the supervised baselines, and the best-performing unsupervised hypernymy detection score (Best-unsup, performance taken from (Shwartz et al., 2017)). Hypernymy ranking results depict the effectiveness of the Dual-T model with respect to supervised baselines even more clearly than hypernymy classification results. All supervised models outperform the best unsupervised model in terms of AP, but only Dual-T is consistently better when considering only 100 top-ranked pairs (AP@100). This adds to the conclusion that explicit modeling of asymmetry using dual tensors yields crucial performance boost.

Meronymy ranking. We measure the ranking performance for meronymy detection on the WN-Me dataset, reporting the results for both randomly- and lexically-split variants of the dataset in Table 5. Meronymy ranking results are in line with performance figures for hypernymy ranking. Again, Dual-T consistently outperforms all three baselines. Absolute AP scores for meronymy are higher than those we report for hypernymy, but this is merely because WN-Me is a balanced dataset, whereas the hypernymy ranking test sets (with the exception of the Weeds dataset) are substantially skewed in favor of negative concept pairs.

5 Analysis

We perform additional analyses, providing further insights into Dual-T model's performance. We analyze how model's performance depends on concept distance in WordNet and on number of concept senses. We also examine the stability of Dual-T model's performance across different languages.

5.1 WordNet Distance

Unlike the HypeNet dataset (Shwartz et al., 2016), which contains only pairs of concepts that exist in a direct relation in some external knowledge base, our WN-Hy and WN-Me datasets (cf. Section 4.1) contain pairs of concepts of varying distance in WordNet, allowing for a more fine-grained analysis of the Dual Tensor model's performance.

We divide the test sets of WN-Hy-EN and WN-Me into five buckets according to the shortest path distance between concepts in WordNet.[4] We show hypernymy and meronymy prediction accuracies for all buckets in Figure 2. For hypernymy, we observe significantly lower accuracy for pairs of concepts appearing close in WordNet hierarchy. Close hyponym-hypernym pairs (e.g., *car–vehicle*) tend to occur in similar contexts and consequently have similar unspecialized embeddings. Such hypernymy instances are difficult to discern from syn-

[4] For any concept with multiple senses, we considered the WordNet synset of its dominant sense.

Dataset	BLESS		EVALuation		Benotto		Weeds	
Model	AP	AP@100	AP	AP@100	AP	AP@100	AP	AP@100
BEST-UNSUP (Shwartz et al., 2017)	.051	.540	.353	.661	.382	.617	.441	.911
CONCAT-SVM	.097	.235	.321	.329	.523	.586	.644	.793
BILIN-PROD	.277	.627	.355	.457	.477	.678	.712	.948
SINGLE-T	.463	.777	.433	.668	.501	.605	.771	.958
DUAL-T	**.487**	**.823**	**.446**	**.866**	**.557**	**.847**	**.774**	**.985**

Table 4: Hypernymy detection, ranking results.

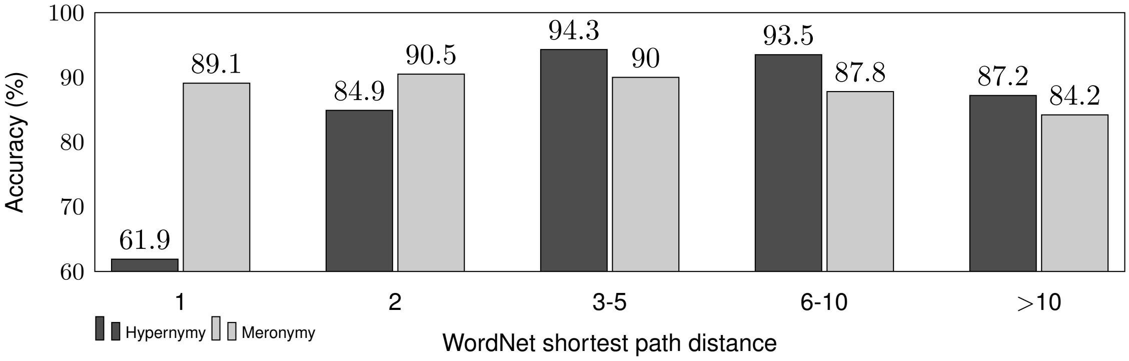


Figure 2: Hypernymy and meronymy performance with respect to WordNet shortest path distance.

	Lex. split		Rand. split	
Model	AP	AP@100	AP	AP@100
CONCAT-SVM	.686	.775	.796	.865
BILIN-PROD	.682	.832	.878	.947
SINGLE-T	.772	.900	.909	.979
DUAL-T	**.840**	**.967**	**.936**	**1.00**

Table 5: Meronymy detection, ranking results.

onymous pairs (e.g., *car–automobile*). The same effect is, however, not observed for meronymy – part-whole relations between close concepts are as detectable as between more distant concepts. This is probably because *part* concepts appear in different contexts than *whole* concepts (e.g., *wheel-car*), resulting in distinct unspecialized embeddings in the first place. For both relations we observe a drop in performance for pairs of very distant concepts. Such pairs typically contain one very abstract concept (e.g., *object*), but embeddings of abstract concepts are not superpositions of embeddings of their hyponyms (Rimell, 2014) nor their meronyms.

5.2 Effects of Polysemy

Given that our Dual Tensor model takes unspecialized concept embeddings as input and that unspecialized embeddings do not discern between different senses of words, our Dual Tensor model treats monosemous and polysemous concepts equally. Intuitively, predicting asymmetric relations for pairs involving polysemous concepts should be more difficult than for pairs of monosemous concepts, because the models in such cases additionally need to learn to discern between different concept senses.

While designing the Dual Tensor model, we hypothesized that different tensor slices might be able to accommodate for asymmetric relations involving different senses of polysemous words. In order to closer examine the effects of polysemy on the performance of the Dual Tensor model, we partitioned the test portions of the WN-Hy and WN-Me datasets according to number of senses of the concept pair (we average the number of senses of the two concepts in a candidate pair). We show the Dual Tensor model's performance ($k = 3, l = 300$) on different number-of-senses buckets, both for hypernymy and meronymy prediction, in Figure 3.

For hypernymy, the general trend is as expected: the larger the average number of senses of concepts in the candidate pair, the lower the prediction accuracy. The exception is the bucket $(3, 5]$ for which the performance is higher than for the previous bucket $(1, 3]$. The drop in performance is not drastic as long as the model is not dealing with highly

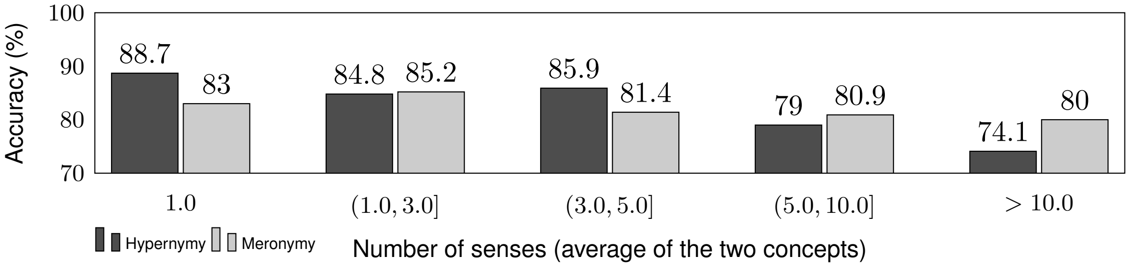


Figure 3: Hypernymy and meronymy performance with respect to concept polysemy.

polysemous concepts (with more than five senses). These performance figures suggest that, via the multiple tensor slices, the DUAL-T model can, to some extent, alleviate the effects that polysemy has on predicting asymmetric lexico-semantic relations.

Somewhat surprisingly, the polysemy seems not to have a clear negative effect for meronymy. Prediction accuracy on pairs of highly polysemous concepts seems to be similar to that on monosemous concept pairs. An instance-level inspection reveals that meronymy detection is more sensitive to the number of senses of the *part* candidate concept than of the *whole* concept. In other words, if we partition the test set only according to the number of senses of the *part* concept, then the trends are similar to those observed for hypernymy.

5.3 Multilingual Comparison

To examine how the Dual Tensor model performs across languages, we evaluate its performance on equally-sized hypernymy detection datasets in English, Spanish, and French (cf. Section 4.1 and Table 1). To increase the comparability of results, for each of the three languages we trained word embeddings using the CBOW algorithm (Mikolov et al., 2013) on the Wikipedia dump of respective language. Also, for all three models we select the hyperparameter configuration that turned out to be optimal most often in previous experiments – we set the length of unspecialized embeddings to $l = 300$ and number of tensor slices to $k = 3$. Hypernymy classification performance for different languages is shown in Table 6. The results suggest that Dual Tensor model exhibits stable performance across languages. The small performance differences between languages may be attributed to different sizes of respective Wikipedia dumps (on which we train unspecialized embeddings) as well as to inherent differences in language complexity (e.g., English being morpho-syntactically simpler).

Language	Dataset	P	R	F_1
English	WN-Hy-EN	89.9	86.1	87.9
Spanish	WN-Hy-ES	88.7	82.1	85.3
French	WN-Hy-FR	86.2	82.7	84.4

Table 6: Hypernymy classification performance for different languages.

6 Conclusion

We have presented a neural model for detecting asymmetric semantic relations. Unlike existing models, which uniformly treat asymmetric and symmetric relations, our Dual Tensor model captures asymmetry explicitly using a pair of specialization tensors that produce two different embedding specializations, depending on the concept's role in the relation. Instead of just updating unspecialized embeddings, with specialization tensors we also explicitly capture the mapping function.

The results from a battery of hypernymy and meronymy experiments show that via asymmetric specialization of concept embeddings the Dual Tensor model is able to outperform (1) the supervised model directly using unspecialized embeddings as well as (2) the more complex neural architecture that additionally exploits syntactic information. We have additionally shown that our model can diminish the negative effects of polysemy and that it exhibits stable performance across languages.

As future work, we plan to develop similar models based on explicit specialization tensors for detecting symmetric relations (e.g., synonymy, antonymy). We will also seek to exploit the Dual Tensor model in different downstream tasks, e.g., hypernymy detection for taxonomy induction (Faralli et al., 2017) or recognizing textual entailment.

Downloads. We make the code of the models and all datasets available at `https://bitbucket.org/gg42554/dual-tensors/`.

References

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. pages 23–32.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the 2011 Workshop on GEometrical Models of Natural Language Semantics*. pages 1–10.

Giulia Benotto. 2015. *Distributional models for semantic relations: A study on hyponymy and antonymy*. Ph.D. thesis, University of Pisa.

Matthew Berland and Eugene Charniak. 1999. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. pages 57–64.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3(Jan):993–1022.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 2013 Annual Conference on Neural Information Processing Systems*. pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*. pages 301–306.

Emmanuele Chersoni, Giulia Rambelli, and Enrico Santus. 2016. CogALex-V Shared Task: ROOT18. *CoRR* abs/1611.01101.

Stefano Faralli, Alexander Panchenko, Chris Biemann, and Simone Paolo Ponzetto. 2017. The ContrastMedium algorithm: Taxonomy induction from noisy knowledge graphs with just a few links. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. pages 590–600.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1606–1615.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 758–764.

Roxana Girju, Adriana Badulescu, and Dan Moldovan. 2006. Automatic discovery of part-whole relations. *Computational Linguistics* 32(1):83–135.

Aitor Gonzalez-Agirre, Egoitz Laparra, and German Rigau. 2012. Multilingual central repository version 3.0: upgrading a very large lexical knowledge base. In *Proceedings of the 6th Global WordNet Conference*.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics: Volume 2*. pages 539–545.

Douwe Kiela, Felix Hill, and Stephen Clark. 2015. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 2044–2048.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering* 16(04):359–389.

Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes* 25(2-3):259–284.

Omer Levy, Steffen Remus, Chris Biemann, Ido Dagan, and Israel Ramat-Gan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 970–976.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. pages 3111–3119.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 142–148.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: a taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. pages 1135–1145.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. pages 1955–1961.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*. pages 113–120.

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

Laura Rimell. 2014. Distributional lexical entailment by topic coherence. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. pages 511–519.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the 25th International Conference on Computational Linguistics*. pages 1025–1036.

Benoît Sagot and Darja Fišer. 2008. Building a free French WordNet from multilingual resources. In *Proceedings of the Ontolex 2008 Workshop*.

Enrico Santus, Anna Gladkova, Stefan Evert, and Alessandro Lenci. 2016. The CogALex-V shared task on the corpus-based identification of semantic relations. In *Proceedings of the 5th Workshop on Cognitive Aspects of the Lexicon*. pages 69–79.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. pages 38–42.

Enrico Santus, Frances Yung, Alessandro Lenci, and Chu-Ren Huang. 2015. EVALution 1.0: an evolving semantic dataset for training and evaluation of distributional semantic models. In *Proceedings of the 4th Workshop on Linked Data in Linguistics*. pages 64–69.

Vered Shwartz and Ido Dagan. 2016. Cogalex-V shared task: Lexnet-integrated path-based and distributional method for the identification of semantic relations. *arXiv preprint arXiv:1610.08694* .

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics: Volume 1, Long Papers*. pages 2389–2398.

Vered Shwartz, Enrico Santus, and Dominik Schlechtweg. 2017. Hypernyms under siege: Linguistically-motivated artillery for hypernymy detection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. pages 65–75.

Rion Snow, Daniel Jurafsky, Andrew Y Ng, et al. 2004. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of the 2004 Annual Conference on Neural Information Processing Systems*. volume 17, pages 1297–1304.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 2013 Annual Conference on Neural Information Processing Systems*. pages 926–934.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2).

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of the 25th International Conference on Computational Linguistics*. pages 2249–2259.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of the 2003 conference on Empirical methods in Natural Language Processing*. pages 81–88.

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

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the 2015 International Conference on Learning Representations*.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics*. pages 947–953.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. pages 545–550.

Incorporating Relation Paths in Neural Relation Extraction

Wenyuan Zeng[1], **Yankai Lin**[2], **Zhiyuan Liu**[2]*, **Maosong Sun**[2]
[1]Department of Physics, Tsinghua University, Beijing, China
[2]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China

Abstract

Distantly supervised relation extraction has been widely used to find novel relational facts from plain text. To predict the relation between a pair of two target entities, existing methods solely rely on those direct sentences containing both entities. In fact, there are also many sentences containing only one of the target entities, which also provide rich useful information but not yet employed by relation extraction. To address this issue, we build inference chains between two target entities via intermediate entities, and propose a path-based neural relation extraction model to encode the relational semantics from both direct sentences and inference chains. Experimental results on real-world datasets show that, our model can make full use of those sentences containing only one target entity, and achieves significant and consistent improvements on relation extraction as compared with strong baselines. The source code of this paper can be obtained from `https://github.com/thunlp/PathNRE`.

1 Introduction

Knowledge Bases (KBs) provide effective structured information for real world facts and have been used as crucial resources for several natural language processing (NLP) applications such as Web search and question answering. Typical KBs such as Freebase (Bollacker et al., 2008), DBpedia (Auer et al., 2007) and YAGO (Suchanek et al., 2007) usually describe knowledge as multi-relational data and represent them as triple facts. As the real-world facts are infinite and increasing every day, existing KBs are still far from complete. Recently, petabytes of natural-language text containing thousands of different structure types are readily available, which is an important resource for automatically finding unknown relational facts. Hence, relation extraction (RE), defined as the task of extracting structured information from plain text, has attracted much interest.

Most existing supervised RE systems usually suffer from the issue that lacks sufficient labelled relation-specific training data. Manual annotation is very time consuming and labor intensive. One promising approach to address this limitation is distant supervision. (Mintz et al., 2009) generates training data automatically by aligning a KB with plain text. They assume that if two target entities have a relation in KB, then all sentences that contain these two entities will express this relation and can be regarded as a positive training instance. Since neural models have been verified to be effective for classifying relations from plain text (Socher et al., 2012; Zeng et al., 2014; dos Santos et al., 2015), (Zeng et al., 2015; Lin et al., 2016) incorporate neural networks method with distant supervision relation extraction. Further, (Ye et al., 2016) considers finer-grained information, and achieves the state-of-the-art performance.

Although existing RE systems have achieved promising results with the help of distant supervision and neural models, they still suffer from a major drawback: the models only learn from those sentences contain both two target entities. However, those sentences containing only one of the entities could also provide useful information and help build inference chains. For example, if we know that "h is the father of e" and "e is the father of t", we can infer that h is the grandfather of t.

In this work, as illustrated in Fig. 1, we introduce a path-based neural relation extraction model

[*] Corresponding author: Z. Liu (liuzy@tsinghua.edu.cn).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1768–1777
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

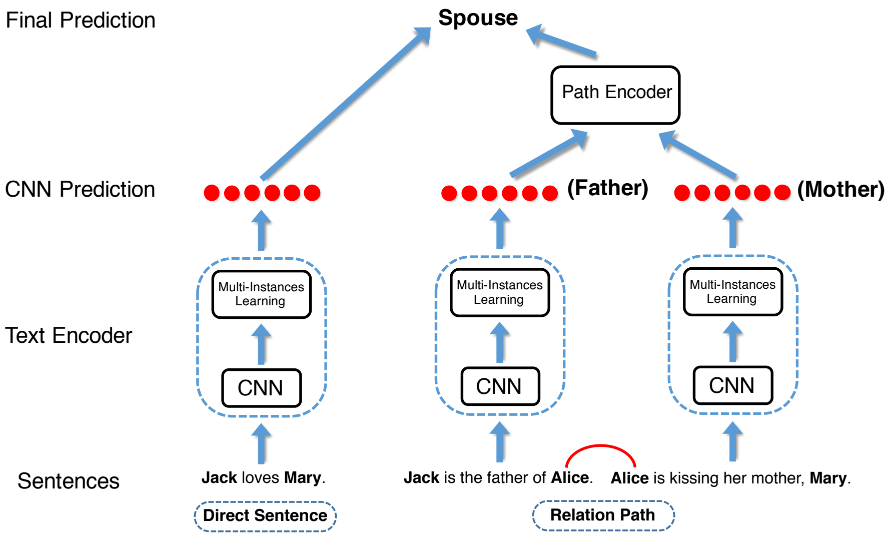


Figure 1: The architecture of our neural relation extraction model with relation paths.

with relation paths. First, we employ convolutional neural networks (CNN) to embed the semantics of sentences. Afterward, we build a relation path encoder, which measures the probability of relations given an inference chain in the text. Finally, we combine information from direct sentences and relation paths to predict the relation.

We evaluate our model on a real-world dataset for relation extraction. The experimental results show that our model achieves significant and consistent improvements as compared with baselines. Besides, with the help of those sentences containing one of the target entities, our model is more robust and performs well even when the number of noisy instances increases. To the best of our knowledge, this is the first effort to consider the information of relation path in plain text for neural relation extraction.

2 Related Work

2.1 Distant Supervision

Distant supervision for RE is originally proposed in (Craven et al., 1999). They focus on extracting binary relations between proteins using a protein KB as the source of distant supervision. Afterward, (Mintz et al., 2009) aligns plain text with Freebase, by using distant supervision . However, most of these methods heuristically transform distant supervision to traditional supervised learning, by regarding it as a single-instance single-label problem, while in reality, one instance could correspond with multiple labels in different scenarios and vice versa. To alleviate the issue, (Riedel et al., 2010) regards each sentence as a training instance and allows multiple instances to share the same label but disallows more than one label. Further, (Hoffmann et al., 2011; Surdeanu et al., 2012) adopt multi-instance multi-label learning in relation extraction. The main drawback of these methods is that they obtain most features directly from NLP tools with inevitable errors, and these errors will propagate to the relation extraction system and limit the performance.

2.2 Neural Relation Extraction

Recently, deep learning (Bengio, 2009) has been successfully applied in various areas, including computer vision, speech recognition and so on. Meanwhile, its effectiveness has also been verified in many NLP tasks such as sentiment analysis (dos Santos and Gatti, 2014), parsing (Socher et al., 2013), summarization (Rush et al., 2015) and machine translation (Sutskever et al., 2014). With the advances of deep learning, there are growing works that design neural networks for relation extraction. (Socher et al., 2012) uses a recursive neural network in relation extraction, and (Xu et al., 2015; Miwa and Bansal, 2016) further use LSTM. (Zeng et al., 2014; dos Santos et al., 2015) adopt CNN in this task, and (Zeng et al., 2015; Lin et al., 2016) combine attention-based multi-instance learning which shows promising results. However, these above models merely learn from those sentences which directly contain both two target entities. The important information of those relation paths hidden in the text is ignored. In this paper, we propose a novel path-based neural RE model to address this issue. Besides, although we choose CNN to test the effectiveness of our model, other neural models could also be easily adapted to our architecture.

2.3 Relation Path Modeling

Relation paths have been taken into consideration on large-scale KBs for relation inference. Path Ranking algorithm (PRA) (Lao and Cohen, 2010) has been adopted for expert finding (Lao and Cohen, 2010), information retrieval (Lao et al., 2012), and further for relation classification based on KB structure (Lao et al., 2011; Gardner et al., 2013). (Neelakantan et al., 2015; Lin et al., 2015; Das et al., 2016; Wu et al., 2016) use recurrent neural networks (RNN) to represent relation paths based on all involved relations in KBs.(Guu et al., 2015) proposes an embedding-based compositional training method to connect the triple knowledge for KB completion. Different from the above work of modeling relation paths in KBs, our model aims to utilize relation paths in text corpus, and help to extract knowledge directly from plain text.

3 Our Method

Given a pair of target entities, a set of corresponding direct sentences $S = \{s_1, s_2, \cdots, s_n\}$ which contains this entity pair, and a set of relation paths $P = \{p_1, p_2, \cdots, p_m\}$, our model aims to measure the confidence of each relation for this entity pair. In this section, we will introduce our model in three parts: (1) **Text Encoder.** Given the sentence with two corresponding target entities, we use a CNN to embed the sentence into a semantic space, and measure the probability of each relation given this sentence. (2) **Relation Path Encoder.** Given a relation path between the target entities, we measure the probability of each relation r, conditioned on the relation path. (3) **Joint Model.** We integrate the information from both direct sentences and relation paths, then predict the confidence of each relation.

3.1 Text Encoder

As shown in Fig. 2, we use a CNN to extract information from text. Given a set of sentences of an entity pair, we first transform each sentence s into its distributed representation $\mathbf{s}$, and then predict relation using the most representative sentence via a multi-instance learning mechanism.

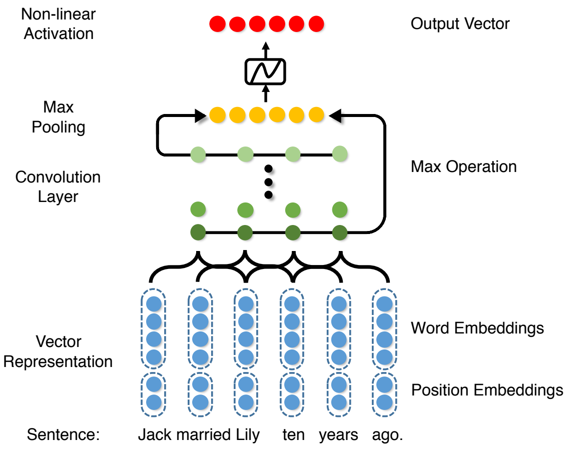


Figure 2: The architecture of CNN used for text encoder.

3.1.1 Input Vector

First, we transform the words $\{w_1, w_2, \cdots, w_l\}$ in sentence s into vectors of dimension d. For each word w_i, we use word embedding to encode its syntactic and semantic meanings, and use position embedding to encode its position information. We then concatenate both word embedding and position embedding to form the input vector of w_i for CNN. (See Figure 2.)

3.1.2 Convolution and Max-pooling Layers

When processing a sentence, it is a great challenge that important information could probably appear in all parts of that sentence. In addition, the length l of a sentence could also vary a lot. Therefore, we apply CNN to encode all local features regardless sentence length. We first apply a convolution layer to extract all possible local features, and then select the most important one via max-pooling layer.

To extract local features, the convolution layer first concatenates a sequence of word embeddings within a sliding window to be vector $\mathbf{q}_i$ of dimension $k \times d$:

$$\mathbf{q}_i = \mathbf{w}_{[i-k+1:i]} (1 \le i \le l + k - 1), \qquad (1)$$

where k is the size of the window, and we also set all out-of-index words to be zero vectors. It then multiplies $\mathbf{q}_i$ by a convolution matrix $\mathbf{W} \in \mathbb{R}^{d_c \times (k \times d)}$, where d_c is the dimension of sentence embeddings. Hence, the output of convolution layer could be expressed as $\mathbf{h} = \{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_{l+k-1}\}$:

$$\mathbf{h}_i = \mathbf{W}\mathbf{q}_i + \mathbf{b}, \qquad (2)$$

where $\mathbf{b}$ is a bias vector. Finally, the max-pooling layer takes a max operation, followed by a hyperbolic tangent activation, over the sequence of $\mathbf{h}_i$ to select the most important information, namely,

$$[\mathbf{s}]_j = \mathbf{tanh}(\max_i [\mathbf{h}_i]_j). \tag{3}$$

3.1.3 Multi-Instance Learning

Next, we apply a softmax classifier upon the sentence representation $\mathbf{s}$ to predict the corresponding relation. We define the condition probability of relation r as follows,

$$p(r|\theta, \mathbf{s}) = \frac{\exp(e_r)}{\sum_{i=1}^{n_r} \exp(e_i)}, \tag{4}$$

where e_i, a component of $\mathbf{e}$, measures how well this sentence matches relation r_i, and n_r is the number of relations. More specifically, $\mathbf{e}$ could be calculated from:

$$\mathbf{e} = \mathbf{U}\mathbf{s} + \mathbf{v}, \tag{5}$$

where $\mathbf{U} \in \mathbb{R}^{n_r \times d_c}$ is the coefficient matrix of relations and $\mathbf{v} \in \mathbb{R}^{n_r}$ is a bias vector.

We use multi-instance learning to alleviate the wrong-labeling issue in distant supervision, by choosing one sentence in the set of all direct sentences $S = \{s_1, s_2, \cdots, s_m\}$ which corresponds to the entity pair (h, t). Similar to (Zeng et al., 2015), we define the score function of this entity pair and its corresponding relation r as a max-one setting:

$$E(h, r, t|S) = \max_i p(r|\theta, \mathbf{s}_i). \tag{6}$$

where E reflects the direct information we derive from sentences. We can also set a random setting as a baseline:

$$E(h, r, t|S) = p(r|\theta, \mathbf{s}_i), \tag{7}$$

where s_i is randomly selected from S.

3.2 Relation Path Encoder

We use Relation Path Encoder to embed the inference information of relation paths. Relation Path Encoder measures the probability of each relation r given a relation path in the text. This will utilize the inference chain structure to help make predictions. More specifically, we define a path p_1 between (h, t) as $\{(h, e), (e, t)\}$, and the corresponding relations are r_A, r_B. Each of (h, e) and (e, t) corresponds to at least one sentence in the text. Our model calculates the probability of relation r conditioned on p_1 as follows,

$$p(r|r_A, r_B) = \frac{\exp(o_r)}{\sum_{i=1}^{n_r} \exp(o_i)}, \tag{8}$$

where o_i measures how well relation r matches with the relation path (r_A, r_B). Inspired by the work on relation path representation learning (Lin et al., 2015), our model first transforms relation r to its distributed representation, i.e. vector $\mathbf{r} \in \mathbb{R}^{d_R}$, and builds the path embeddings by composition of relation embeddings. Then, the similarity o_i is calculated as follows:

$$o_i = -\|\mathbf{r}_i - (\mathbf{r}_A + \mathbf{r}_B)\|_{L_1}. \tag{9}$$

Therefore, if $\mathbf{r}_i$ gets more similar to $(\mathbf{r}_A + \mathbf{r}_B)$, the conditioned predicting probability of r_i will become larger. Here, we make an implicit assumption that if r_i is semantically similar to relation path $p_i : h \xrightarrow{r_A} e \xrightarrow{r_B} t$, the embedding $\mathbf{r}_i$ will be closer to the relation path embedding $(\mathbf{r}_A + \mathbf{r}_B)$. Finally, for this relation path $p_i : h \xrightarrow{r_A} e \xrightarrow{r_B} t$, we define an relation-path score function,

$$G(h, r, t|p_i) = E(h, r_A, e)E(e, r_B, t)p(r|r_A, r_B), \tag{10}$$

where $E(h, r_A, e)$ and $E(e, r_B, t)$ measure the probabilities of relational facts (h, r_A, e) and (e, r_B, t) from text, and $p(r|r_A, r_B)$ measures the probability of relation r given relation path (r_A, r_B).

In reality, there are usually multiple relation paths between two entities. Hence, we define the inferring correlation between relation r and several sentence paths P as,

$$G(h, r, t|P) = \max_i G(h, r, t|p_i), \tag{11}$$

where we use max operation to filter out those noisy paths and select the most representative path.

3.3 Joint Model

Given any entity pair (h, t), those sentences S directly mentioning them and relation paths P between them, we define the global score function with respect to a candidate relation r as,

$$L(h, r, t) = E(h, r, t|S) + \alpha G(h, r, t|P), \tag{12}$$

where $E(h, r, t|S)$ models the correlation between r and (h, t) calculated from direct sentences,

$G(h, r, t|P)$ models the inferring correlation between relation r and several sentence paths P. α equals to $(1 - E(h, r, t|S))$ times a constant β. This term serves to depict the relative weight between direct sentences and relation paths, since we don't need to pay much attention on extra information when CNN has already given a confident prediction, namely $E(h, r, t|S)$ is large.

One of the advantages of this joint model is to alleviate the issue of error propagation. The uncertainty of information from Text Encoder and Relation Path encoder is characterized by its confidence, and could be integrated and corrected in this joint model step. Furthermore, since we treat relation paths in a probabilistic way, our model could fully utilize all relation paths, i.e. those always hold and those likely to hold.

3.4 Optimization and Implementation Details

The overall objective function is defined as:

$$J(\theta) = \sum_{(h,r,t)} \log(L(h, r, t)), \tag{13}$$

where the summing runs over the log loss of all entity pairs in text and θ represents the model parameters. To solve this optimization problem, we use mini-batch stochastic gradient descent (SGD) to maximize our objective function. We initialize $\mathbf{W}_E$ with the results from Skip-gram model, and initialize other parameters randomly. We also adopt dropout (Srivastava et al., 2014) upon the output layer of CNN.

We implement our model using C++. We train our model on Intel(R) Xeon(R) CPU E5-2620, and the training roughly takes half a day. The word embedding and other parameters are updated via back-propagation simultaneously, while the relation path structure is extracted before training and stored afterward.

4 Dataset

We build a novel dataset for evaluating relation extraction task. We first describe the most commonly used previous dataset and then explain the reason and how we construct the new dataset.

4.1 Previous Datasets & Reasons for New Dataset

A commonly used benchmark dataset for this task was developed by (Riedel et al., 2010). This dataset was built by aligning Freebase (Dec. 2009 Snapshot) with New York Times corpus (NYT). There are 53 possible relationships between two entities, including a special relation type NA, meaning that there is no relation between head and tail entities. For each relational fact in a filtered Freebase dataset, a sentence from NYT would be regarded as a mention of this relation if both the head and tail entity appear in that sentence.

While this previous dataset has been frequently used for evaluating relation extraction systems, we observe some limitations of it. First, the relational facts are extracted from a 2009 snapshot of Freebase. Therefore, this dataset is too old to contain many updated facts. This will underestimate the performance of a relation extraction system, since some real-world facts are missing from the dataset and labeled as NA. Second, the relational facts in this dataset are scattered, i.e. there are not sufficient relation paths in this dataset, while relational facts in real-world always have connections with each other. Third, Freebase will no longer update after 2016.These limitations mean that this dataset is somewhat improper for evaluating RE systems.

Although other relation extraction datasets exist, e.g. ACE[1] and (Hendrickx et al., 2009), they are too small to train an effective neural relation extraction model. Moreover, each relational fact in (Hendrickx et al., 2009) only corresponds with one sentence, which prevents it from evaluating multi-instance relation extraction systems. Hence, we constructed a novel relation extraction dataset to address these issues, and will make it available to the community.

4.2 Dataset Construction

Datasets	Sets	# sentences	# entity pairs	# facts
Riedel et.al.	Train	522,611	281,270	18,252
	Valid	-	-	-
	Test	172,448	96,678	1,950
Ours	Train	647,827	266,118	50,031
	Valid	234,350	121,160	5,609
	Test	235,609	121,837	5,756

Table 1: Statistics of datasets.

Our dataset contains more updated facts and richer structures of relations, e.g. more relations / relation paths, as compared to existing similar datasets. The dataset is expected to be more similar to real-world cases, and thus be more appropriate for evaluating RE systems' performances.

We build the dataset by aligning Wikidata[2] re-

[1] https://catalog.ldc.upenn.edu/LDC2006T06

[2] https://www.wikidata.org/

lations with the New York Times Corpus (NYT). Wikidata is a large, growing knowledge base, which contains more than 80 million triple facts and 20 million entities. Different from Freebase, Wikidata is still in maintenance and could be easily accessed by APIs. We first pick those entities simultaneously appeared in both Wikidata and Freebase, and relational facts associated with them. Then, we filtered out a subset S, reserving those facts associating with the 99 highest frequency relations. This results in $4,574,665$ triples with $1,045,385$ entities and 99 relations.

Next, we align those facts with NYT corpus, following the assumption of distant supervision. For each pair of entities appearing in our S, we traverse the corpus and pick those sentences where both entities appear. These sentences will be regarded as mentions of this fact, and labeled by this relation type. To simulate noise in the real world, we also add sentences corresponding to "No Relation" entity pairs into our dataset. To get those "No Relation" instances, we first create a fake knowledge base S^- by randomly replacing the head or tail entities in triples, i.e., $S^- = \{(h', r, t)\} \cup \{(h, r, t')\}$ and then align them with NYT corpus. Finally, we randomly split all those selected sentences into training, validation and testing set, assuring that a relational fact could be only mentioned by sentences in one set. The statistics of our dataset and (Riedel et al., 2010) are listed in Table 1.

5 Experiments

Following the previous work (Mintz et al., 2009), we evaluate our model by extracting relational facts from the sentences in test set, and compare them with those in Wikidata. We report Precision/Recall curves, Precision@N (P@N) and F1 scores for comparison in our experiments.

5.1 Initialization and Parameter Settings

In this paper, we use the word2vec tool [3] to pre-train word embeddings on NYT corpus. We keep the words which appear more than 100 times in the corpus as vocabulary. We tune our model on the validation set, using grid search to determine the optimal parameters, which are shown in boldface. We select learning rate for SGD $\lambda \in \{0.1, \mathbf{0.01}, 0.001\}$, the sentence embedding size $d_c \in \{50, 60, \cdots, \mathbf{230}, \cdots, 300\}$, the window size $k \in \{1, 2, \mathbf{3}, 4, 5\}$, and the mini-batch size $B \in \{40, \mathbf{160}, 640\}$. Besides, we select the relation embeddings size $d_R \in \{5, 10, \cdots, \mathbf{40}, \cdots, 60\}$, and the weight for information from relation paths $\beta \in \{0.01, 0.1, 0.2, \mathbf{0.5}, 1, \cdots, 5\}$. For other parameters which have little effect on the system performance, we follow the settings used in (Zeng et al., 2015): word embedding size d_w is 50, position embedding size d_p is 5 and dropout rate p is 0.5. For training, the iteration number over all training data is 25.

5.2 Effectiveness of Incorporating Relation Paths

5.2.1 Precision-Recall Curve Comparison

To demonstrate the effect of our approach, we empirically compare it with other neural relation extraction methods via held-out evaluation. (1) **CNN+rand** represents the CNN model reported in (Zeng et al., 2014). (2) **CNN+max** represents the CNN model with multi-instances learning used in (Zeng et al., 2015). (3) **Path+rand/max** is our model with those two multi-instance settings. We implement (1), (2) by ourselves which achieve comparable results as reported in those papers.

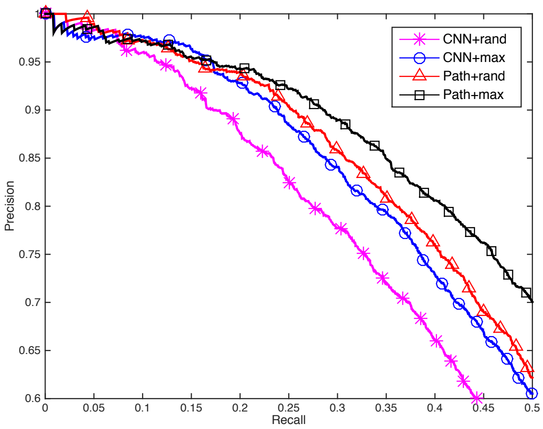


Figure 3: Aggregate precision/recall curve for CNN+rand, CNN+max, Path+rand, Path+max.

Fig. 3 shows the precision/recall curves of all methods. From the figure, we can observe that: (1) Our methods outperform their counterpart methods, achieving higher precision over almost entire range of recall. They also enhance recall by 20% without decrease of precision. These results prove the effectiveness of our approach. We notice that the improvements of our methods over

[3] https://code.google.com/p/word2vec/

Test Settings (Noise)	75%				85%				95%			
P@N (%)	10%	20%	50%	F1	10%	20%	50%	F1	10%	20%	50%	F1
CNN+rand	86.7	67.0	38.9	57.5	84.6	66.4	37.5	55.0	79.9	61.8	35.2	51.4
CNN+max	86.0	68.5	38.3	57.2	85.4	67.6	37.7	56.5	84.4	66.0	36.6	54.8
Path+rand	**89.4**	**71.7**	**39.9**	59.3	88.2	70.2	39.0	58.1	86.0	67.2	37.0	55.6
Path+max	89.0	71.5	39.8	**59.6**	**89.0**	**71.4**	**39.6**	**59.4**	**88.6**	**71.0**	**39.1**	**59.1**

Table 2: P@N and F1 for relation extraction in texts containing different percentage of no-relation facts.

baselines are relatively small at small recall value, which corresponds to high predicting confidence. This phenomenon is intuitive since our joint model could dynamically leverage the importance of direct sentence and relation paths, and tends to trust the Text Encoder when the confidence is high. (2) As the recall increases, our models exhibit larger improvements compared with CNN in terms of percentage. This is due to the fact that sometimes CNNs cannot extract reliable information from direct sentences, while our methods could alleviate this issue by considering more information from inference chains, and thus still maintain high precision. (3) Both CNN+max and Path+rand are variations of CNN+rand, aiming to alleviate the problem of noisy data. We see that Path+rand outperforms CNN+max over all range, which indicates that considering path information is a better way to solve this issue. Meanwhile, combining paths information and max operation, Path+max, gives the best performance. (4) Path+rand shows a larger improvement over CNN+rand, compared with those of Path+max and CNN+max. This furthermore proves the effectiveness of considering relation path information: CNN+rand has much more severe problem suffering from noise, so using our method to incorporate paths information to alleviate this issue could perform better.

5.2.2 Comparison on Long Tail Situation

	$N_s \leq 1$	$N_s \leq 2$	$N_s \leq 5$	All
CNN+rand	53.9	54.0	52.0	51.4
Path+rand	**58.4**	**58.1**	**56.0**	**55.6**
CNN+max	57.8	58.3	56.5	55.7
Path+max	**63.6 (+5.8)**	**62.5 (+4.2)**	**60.2 (+3.7)**	**59.1 (+3.4)**

Table 3: F1 score for long tail situation.

Real-world data follows long-tail distribution (power law). In the testing set, we also observe a fact that about 40% triple facts appear only in single sentence, and thus a multi-instance relation extraction system, e.g. CNN+max, could only rely on limited information and the multi-instance mechanism will not work well. Our system, on the contrary, can still utilize information from relation paths in this case, and is expected to perform much better in the long tail situation.

We evaluate the models on different parts of the long-tail distribution. To get testing instances from different parts of the distribution, we extract all the triple facts appearing less than N_s sentences in the testing set, and those sentences associating with them. All text related to 'No Relation' entity pair are also reserved in order to simulate noise. We then evaluate different models on those sampled testing set, and report the results of F1 score in Table 3.

From Table 3, we could observe that: (1) Incorporating relation paths is indeed effective in predicting relations, and our models have significant improvements compared with the baselines. (2) Path+max indeed has larger improvements over CNN+max when N_s is small, which is consistent with our previous expectation. Also notice that the gap between Path+rand and CNN+rand is relatively constant. This is due to the fact that both these methods only use one random sentence, regardless of how many sentences there are associating with an entity pair.

5.3 Model Robustness under Different Percentages of Noise

In the task of relation extraction, there are lots of noise in text which may hurt the model's performance. More specifically, “No Relation” entity pair is a kind of noise, since “No Relation” could actually contain many unknown relation types, and thus might confuse the relation extraction systems. Therefore, it is important to verify the robustness of our model in the presence of massive noise. Here, we evaluate those models in three settings, with the same relational facts and different percentages of “No Relation” sentences in the testing sets. In each experiment, we extract top 20,000 predicting relational facts according to the model's predicting scores, and report the precision @top 10%, @top 20%, @top 50% and F1 score in Table 2.

From the table, we can see that: (1) In terms of all evaluations, our models achieve the best perfor-

	Relation	Text
Path #1	mother	**Rebecca** gave birth to twin sons, **Esau** and Jacob, ...
Path #2	has_child	...**Isaac**'s marriage to Rebecca, by whom he has two sons, **Esau** and jacob, ...
Test	spouse	... **Isaac** and **Rebecca** and the female and male evil spirits ...
Path #1	shares_border_with	... in **Somalia**, ... soldiers and marines stationed in neighboring **Djibouti** ...
Path #2	shares_border_with	... **Ethiopia** have had the effect of making neighboring **Djibouti** ...
Test	shares_border_with	The next day, **Ethiopia** struck, its military pushing deep into **Somalia** ...

Table 4: Some representative examples of inference chians in NYT corpus. The bold is target entities.

mance as compared with other methods in all test settings. It demonstrates the effectiveness of our approach. (2) Even though the scores of all models drop as the increasing of noise, we find that Path+rand/max's scores decrease much less than their counterparts. This result proves the effectiveness of taking inference chains into consideration. Since we utilize more information to make predictions, our model is more robust to the presence of mass noise.

5.4 Effectiveness of Learned Features in Zero-Shot Scenario

It has been proved that CNN could automatically extract useful features, encoding syntactic and semantic meaning of sentences. These features are sometimes fed to subsequent models to solve other tasks. In this experiment, we demonstrate the effectiveness of the extracted features from our model. Since CNN-based models have already succeeded in extracting relations from single sentences, we set our experiment in a new scenario: predicting the relation between entities which have not appeared in the same sentence.

A natural approach is to build a relation path between this zero-shot entity pair. We assume that we can make a prediction about (h, t), once we know the information of (h, e) and (e, t). Therefore, we build the training set by extracting all such relation paths and their sentences from training text, and similar for testing set. To test the effectiveness of features, we encode sentences by CNN+rand/max, Path+rand/max respectively, and then feed the concatenation of sentence vectors to a logistic classifier.

Feature	Accuracy
CNN+rand	56.9
CNN+max	57.3
Path+rand	58.5
Path+max	**60.4**

Table 5: Accuracy of different models in zero-shot situation.

From Table 5, we could observe that: (1) The result using CNN+rand features is comparable to the result using CNN+max features. It shows that using max operation to train the features does not greatly improve the features' behavior in this task, even though it performs well in previous tasks. The reason is that, both CNN+rand and CNN+max only encode the information from a single sentence, and they are unable to capture the correlations between relations. (2) Feature from Path+rand/max shows its effectiveness over those from other methods. It indicates that our method is able to model the correlations between relations, while also keeps the syntactic and semantic meaning of a sentence. Therefore, the features extracted from Path+rand/max are useful for a wider range of applications, especially in those tasks which need the information from relations.

5.5 Case study

Table 4 shows some representative inference chains from the testing dataset. These examples can not be predicted correctly by the original CNN model, but are later corrected using our model. We show the test instances and their correct relations, as well as the inference chains the model uses. In the first example, the test sentence does not directly express the relation `spouse`, the proof of this relation appears in a further context in NYT. However, using path#1 and path#2, we could easily infer that *Rebecca* and *Issac* are `spouse`. The second example doesn't show the relation either. But with the help of intermediate entity, *Dijibouti*, our model predicts that *Somalia* shares the border with *Ethiopia*. Note that this inference chain doesn't always hold, but our model could capture this uncertainty well via a softmax operation. In general, our model can utilize common sense from inference chains. It helps make correct predictions even if the inference is not explicit.

6 Conclusion and Future Work

In this paper, we propose a neural relation extraction model which encodes the information of relation paths. As compared to existing neural relation extraction models, our model is able to utilize the sentences which contain both two target entities and only one target entity and is more robust for noisy data. Experimental results on real-world datasets show that our model achieves significant and consistent improvements on relation extraction as compared with baselines.

In the future, we will explore the following directions: **(1)** We will explore the combination of relation paths from both plain texts and KBs for relation extraction. **(2)** We may take advantages of probabilistic graphical model or recurrent neural network to encode more complicated correlations between relation paths, e.g. multi-step relation paths, for relation extraction.

Acknowledgments

This work is supported by the 973 Program (No. 2014CB340501), the National Natural Science Foundation of China (NSFC No. 61572273, 61661146007), China Association for Science and Technology (2016QNRC001), and Tsinghua University Initiative Scientific Research Program (20151080406).

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. *Dbpedia: A nucleus for a web of open data*. Springer.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of KDD*, pages 1247–1250.

Mark Craven, Johan Kumlien, et al. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of ISMB*, volume 1999, pages 77–86.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2016. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426*.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom M Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of EMNLP*, pages 833–838.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of EMNLP*, pages 318–327.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL-HLT*, pages 541–550.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of EMNLP*, pages 529–539.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of EMNLP-CoNLL*, pages 1017–1026.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2015. Modeling relation paths for representation learning of knowledge bases. *Proceedings of EMNLP*.

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

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

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of ACL*, pages 1105–1116.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base inference. In *2015 AAAI Spring Symposium Series*.

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

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *Proceedings of EMNLP*.

Cıcero Nogueira dos Santos and Maıra Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING*.

Cıcero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of ACL*, volume 1, pages 626–634.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of ACL*. Citeseer.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of WWW*, pages 697–706. ACM.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP*, pages 455–465.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, pages 3104–3112.

Jiawei Wu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Knowledge representation via joint learning of sequential text and knowledge graphs. *arXiv preprint arXiv:1609.07075*.

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

Hai Ye, Wenhan Chao, and Zhunchen Luo. 2016. Jointly extracting relations with class ties via effective deep ranking. *arXiv preprint arXiv:1612.07602*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*.

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

Adversarial Training for Relation Extraction

Yi Wu
Computer Science Division
UC Berkeley
jxwuyi@gmail.com

David Bamman
School of Information
UC Berkeley
dbamman@berkeley.edu

Stuart Russell
Computer Science Division
UC Berkeley
russell@berkeley.edu

Abstract

Adversarial training is a mean of regularizing classification algorithms by generating adversarial noise to the training data. We apply adversarial training in relation extraction within the multi-instance multi-label learning framework. We evaluate various neural network architectures on two different datasets. Experimental results demonstrate that adversarial training is generally effective for both CNN and RNN models and significantly improves the precision of predicted relations.

1 Introduction

Despite the recent successes of deep neural networks on various applications, neural network models tend to be overconfident about the noise in input signals. *Adversarial examples* (Szegedy et al., 2013) are examples generated by adding noise in the form of small perturbations to the original data, which are often indistinguishable for humans but drastically increase the loss incurred in a deep model. *Adversarial training* (Goodfellow et al., 2014) is a technique for regularizing deep models by encouraging the neural network to correctly classify both unmodified examples and perturbed ones, which in practice not only enhances the robustness of the neural network but also improves its generalizability. Previous work has largely applied adversarial training on straightforward classification tasks, including image classification (Goodfellow et al., 2014) and text classification (Miyato et al., 2016), where the goal is simply predicting a single label for every example and the training examples are able to provide strong supervision. It remains unclear whether adversarial training could be still effective for tasks with much weaker supervision, e.g., distant supervision (Mintz et al., 2009), or a different evaluation metric other than prediction accuracy (e.g., F1 score).

This paper focuses on the task of *relation extraction*, where the goal is to predict the relation that exists between a particular entity pair given several text mentions. One popular way to handle this problem is the multi-instance multi-label learning framework (MIML) (Hoffmann et al., 2011; Surdeanu et al., 2012) with distant supervision (Mintz et al., 2009), where the mentions for an entity pair are aligned with the relations in Freebase (Bollacker et al., 2008). In this setting, relation extraction is much harder than the canonical classification problem in two respects: (1) although distant supervision can provide a large amount of data, the training labels are very noisy, and due to the multi-instance framework, the supervision is much weaker; (2) the evaluation metric of relation extraction is often the precision-recall curve or F1 score, which cannot be represented (and thereby optimized) directly in the loss function.

In order to evaluate the effectiveness of adversarial training for relation extraction, we apply it to two different architectures (a convoluational neural network and a recurrent neural network) on two different datasets. Experimental results show that even on this harder task with much weaker supervision, adversarial training can still improve the performance on all of the cases we studied.

2 Related Work

Neural Relation Extraction: In recent years, neural network models have shown superior performance over approaches using hand-crafted features in various tasks. Convolutional neural networks (CNN) are among the first deep models that have been applied to relation extrac-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1778–1783
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

tion (Santos et al., 2015; Nguyen and Grishman, 2015). Variants of convolutional networks include piecewise-CNN (PCNN) (Zeng et al., 2014), split CNN (Adel et al., 2016), CNN with sentence-wise pooling (Jiang et al., 2016) and attention CNN (Wang et al., 2016). Recurrent neural networks (RNN) are another popular choice, and have been used in recent work in the form of recurrent CNNs (Cai et al., 2016) and attention RNNs (Zhou et al., 2016). An instance-level selective attention mechanism was introduced for MIML by Lin et al. (2016), and has significantly improved the prediction accuracy for several of these base deep models.

Adversarial Training: Adversarial training (AT) (Goodfellow et al., 2014) was originally introduced in the context of image classification tasks where the input data is continuous. Miyato et al. (2015, 2016) adapts AT to text classification by adding perturbations on word embeddings and also extends AT to a semi-supervised setting by minimizing the entropy of the predicted label distributions on unlabeled data.

AT introduces an end-to-end and deterministic way of data perturbation by utilizing the gradient information. There are also other works for regularizing classifiers by adding random noise to the data, such as dropout (Srivastava et al., 2014) and its variant for NLP tasks, word dropout (Iyyer et al., 2015). Xie et al. (2017) discusses various data noising techniques for language models. Søgaard (2013) and Li et al. (2017) focus on linguistic adversaries.

3 Methodology

We first introduce MIML and then describe the base neural network models we consider:[1] piecewise CNN (Zeng et al., 2015) (PCNN) and bidirectional GRU (Cho et al., 2014) (RNN). We also utilize the *selective attention* mechanism in Lin et al. (2016) for both PCNN and RNN models. Adversarial training is presented at the end of this section.

3.1 Preliminaries

In MIML, we consider the set of text sentences $X = \{x_1, x_2, \ldots, x_n\}$ for each entity pair. Supposing we have R predefined relations (including *NA*) to extract, we want to predict the probability of each of the R relations given the mentions. Formally, for each relation r, we want to predict $P(r \mid x_1, \ldots, x_n)$.

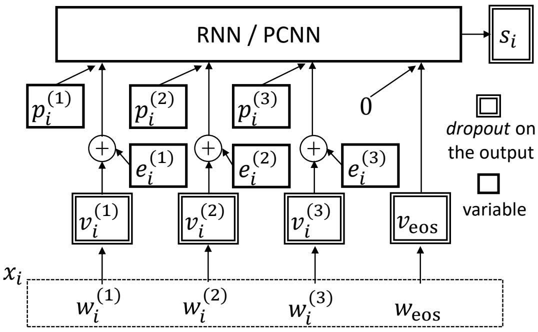


Figure 1: The computation graph of encoding a sentence x_i with adversarial training. e_i denotes the adversarial perturbation w.r.t. x_i. Dropout is placed on the output of the variables in the double-lined rectangles.

Note that since an entity pair may have no relations, we introduce a special relation *NA* to the label set. Hence, we simply assume there will be *at least* one relation existing for every entity pair. During evaluation, we ignore the probability predicted for the *NA* relation.

3.2 Neural Architectures

Input Representation: For each sentence x_i, we use pretrained word embeddings to project each word token into d_w-dimensional space. Note that we also need to include the entity position information in x_i. Here we introduce an extra feature vector $p_i^{(w)}$ for each word w to encode the entities' positions. One choice is the *position embedding* (Zeng et al., 2014): for each word w, we compute the relative distances to the two entities and embed the distances in two d_p-dimensional vectors, which are then concatenated as $p_i^{(w)}$. Position embedding introduces extra variables in the model and slows down the training time. We also investigate a simpler choice, *indicator encoding*: when a word w is exactly an entity, we generate a d_p-dimensional $\vec{1}$ vector and a $\vec{0}$ vector otherwise. In our experiments, position embedding is crucial for PCNN due to the spatial invariance of CNN. For RNN, position embedding helps little (likely because an RNN has the capacity of exploiting temporal dependencies) so we adopt indicator encoding instead.

Sentence Encoder: For a sentence x_i, we want to apply a non-linear transformation to the vector

[1] We primarily focus on effectiveness of AT. Other techniques in Sec. 2 are complementary to our focus.

representation of x_i to derive a feature vector $s_i = f(x_i;\theta)$ given a set of parameters θ. We consider both PCNN and RNN as $f(x_i;\theta)$.

For PCNN, inheriting the settings from (Zeng et al., 2014), we adopt a convolution kernel with window size 3 and d_s output channels and then apply piecewise pooling and ReLU (Nair and Hinton, 2010) as an activation function to eventually obtain a $3 \cdot d_s$-dimensional feature vector s_i.

For RNN, we adopt bidirectional GRU with d_s hidden units and concatenate the hidden states of the last timesteps from both the forward and the backward RNN as a $2 \cdot d_s$-dimensional feature vector s_i.

Selective Attention: Following Lin et al. (2016), for each relation r, we aim to softly select an attended sentence s_r by taking a weighted average of $s_1, s_2, \ldots, s_n$, namely $s_r = \sum_i \alpha_i^r s_i$. Here α^r denotes the attention weights w.r.t. relation r. For computing the weights, we define a *query vector* q_r for each relation r and compute $\alpha^r = \mathrm{softmax}(u^r)$ where $u_i^r = \tanh(s_i)^\top q_r$. The query vector q_r can be considered as the embedding vector for the relation r, which is jointly learned with other model parameters.

Loss Function: For an entity pair, we compute the probability of relation r by $P(r \mid X;\theta) = \mathrm{softmax}(As_r + b)$, where A is the projection matrix and b is the bias. For the multi-label setting, suppose K relations $r_1, \ldots, r_K$ exist for X. Simply taking the summation over the log probabilities of all those labels yields the final loss function

$$L(X;\theta) = -\sum_{i=1}^{K} \log P(r_i \mid X;\theta). \qquad (1)$$

Dropout: For regularizing the parameters, we apply dropout (Srivastava et al., 2014) to both the word embedding and the sentence feature vector s_i. Note that we do not perform dropout on the position embedding p_i.

3.3 Adversarial Training

Adversarial training (AT) is a way of regularizing the classifier to improve robustness to small *worst-case* perturbations by computing the gradient direction of a loss function w.r.t. the data. AT generates continuous perturbations, so we add the adversarial noise at the level of the word embeddings, similar to Miyato et al. (2016). Formally, consider the input data X and suppose the word embedding of all the words in X is V. AT adds a small adversarial perturbation e_{adv} to V and optimizes the following objective instead of Eq.(1).

Dataset	#Rel	#Ent-Pair	#Mention	Sent-Len
NYT-Train	58	290429	577434	145
UW-Train	5	132419	546731	120

Table 1: Dataset statistics (#Rel includes *NA*).

$$L_{\text{adv}}(X;\theta) = L(X + e_{\text{adv}};\theta), \text{ where} \qquad (2)$$

$$e_{\text{adv}} = \arg\max_{\|e\|\le\epsilon} L(X + e;\hat{\theta}) \qquad (3)$$

Here $\hat{\theta}$ denotes a fixed copy of the current value of θ. Since Eq.(3) is computationally intractable for neural nets, Goodfellow et al. (2014) proposes to approximate Eq.(3) by linearizing $L(X;\hat{\theta})$ near X:

$$e_{\text{adv}} = \epsilon g/\|g\|, \text{ where } g = \nabla_V L(X;\hat{\theta}). \qquad (4)$$

Here V denotes the word embedding of *all* the words in X. Accordingly, in Eq. 4, $\|g\|$ denotes the norm of gradients over *all* the words from *all* the sentences in X. In addition, we do not perturb the feature vector p for entity positions. A visualization of the process is demonstrated in Fig. 1.

4 Experiments

To measure the effectiveness of adversarial training on relation extraction, we evaluate both the CNN (PCNN) and RNN (bi-GRU) models on two different datasets, the NYT dataset (NYT) developed by Riedel et al. (2010) and the UW dataset (UW) by Liu et al. (2016). All code is implemented in Tensorflow (Abadi et al., 2016) and available at `https://github.com/jxwuyi/AtNRE`. We adopt Adam optimizer (Kingma and Ba, 2014) with learning rate 0.001, batch size 50 and dropout rate 0.5. For adversarial training, the only parameter is ϵ. In each of the following experiments, we fixed all the hyper-parameters of the base model, performed a binary search solely on ϵ and showed the most effective value of ϵ.

4.1 Datasets

The statistics of the two datasets are summarized in Table 1. We exclude sentences longer than *Sent-Len* during training and randomly split data for entity pairs with more than 500 mentions. Note that the number of target relations in these two datasets are significantly different, which helps

Recall	0.1	0.2	0.3	0.4	AUC
PCNN	0.667	0.572	0.476	0.392	0.329
PCNN-Adv	0.717	0.589	0.511	0.407	0.356
RNN	0.668	0.586	0.524	0.442	0.351
RNN-Adv	**0.728**	**0.646**	**0.553**	**0.481**	**0.382**

Table 2: Precisions of various models for different recalls on the NYT dataset, with best values in bold.

Recall	0.1	0.2	0.3	0.4	AUC
PCNN	0.765	0.717	0.713	0.677	0.576
PCNN-Adv	0.844	0.750	0.738	0.707	0.619
RNN	0.823	0.822	0.791	0.752	0.631
RNN-Adv	**0.929**	**0.878**	**0.850**	**0.779**	**0.671**

Table 3: Precisions of various models for different recalls on the UW dataset, with best values in bold.

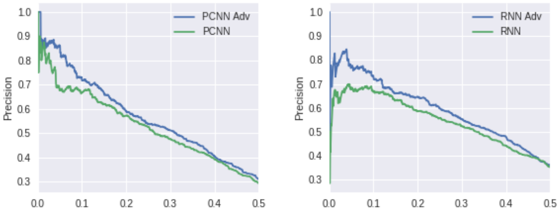


Figure 2: PR curves for PCNN (left) and RNN (right) on the NYT dataset with (blue) and without (green) adversarial training.

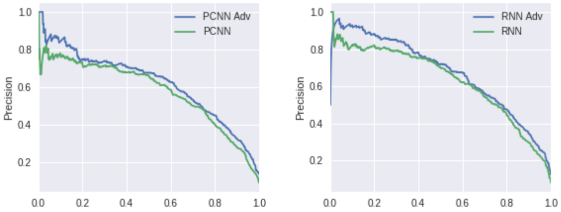


Figure 3: PR curves for PCNN (left) and RNN (right) on the UW dataset with (blue) and without (green) adversarial training.

demonstrate the applicability of adversarial training on various evaluation settings.

Since the test set of the UW dataset only contains 200 sentences, we adopt a subset of the test set from the NYT dataset: all the entity pairs with the corresponding 4 relations in UW and another 1500 randomly selected *NA* pairs.

4.2 Practical Performances

The NYT dataset:

We utilize the word embeddings released by Lin et al. (2016), which has $d_w = 50$ dimensions. For model parameters, we set $d_e = 5$ (dimension of the entity position feature vector) and $d_s = 230$ (dimension of sentence feature vector) for PCNN and $d_e = 3$ and $d_s = 150$ for RNN. For adversarial training, we choose $\epsilon = 0.01$ for PCNN and $\epsilon = 0.02$ for RNN. We empirically observed that when adding dropout to the word embeddings, PCNN performs significantly worse. Hence we only apply dropout to s_i for PCNN. However, even with a dropout rate of 0.5, RNN still performs well. We conjecture that it is due to PCNN being more sensitive to input signals and the dimensionality of the word embedding ($d_w = 50$) being very small.

The precision-recall curves for different models on the test set are shown in Fig. 2. Since the precision drops significantly with large recalls on the NYT dataset, we emphasize a part of the curve with recall number smaller than 0.5 in the figure. Adversarial training significantly improves the precision for both PCNN and RNN models. We also show the precision numbers for some particular recalls as well as the AUC (for the whole PR curve) in Table 2, where RNN generally leads to better precision.

The UW dataset:

We train a word embedding of $d_w = 200$ dimensions using Glove (Pennington et al., 2014) on the New York Times Corpus in this experiment. For model parameters, we set the entity feature dimension $d_e = 5$ and sentence feature dimension $d_s = 250$ for PCNN and $d_e = 3$ and $d_s = 200$ for RNN. For adversarial training, we choose $\epsilon = 0.05$ for PCNN and $\epsilon = 0.5$ for RNN. Since here word embedding dimension d_w is larger than that used for the NYT dataset, which implies that we now have word embeddings with larger norms, accordingly the optimal value of ϵ increases. The precision-recall curves on the test data are shown in Fig. 3, where adversarial training again significantly improves the precision for both models. The precision numbers for some particular recall values as well as the AUC numbers are demonstrated in Table 3. Similarly RNN yields superior performances on the UW dataset.

4.3 Discussion

CNN vs RNN: In the experiments, RNN generally produces more precise predictions than CNN due to its rich model capacity and also has high robustness to input embeddings. The CNN, in contrast, has far fewer parameters which leads to much faster training and testing, which suggests a practical trade-off.

Notably, although the improvement under AUC by adversarial training are roughly the same for both RNN and CNN, the optimal ϵ value for RNN is always much larger than CNN. This implies that empirically RNN is more robust under adversarial attacks than CNN, which also helps RNN maintain higher precision as recall increases.

Choice of ϵ: When $\epsilon = 0$, the AT loss (Eq.(2)) degenerates to the original loss (Eq.(1)); when ϵ becomes too large, the noise can change the semantics of a sentence[2] and make the model extremely hard to correctly classify the adversarial examples.

Notably, the optimal value of ϵ is much smaller than the norm of the word embedding, which implies adversarial training works most effectively when only producing tiny perturbations on word features while keeping the semantics of sentences unchanged[3].

Connection to other approaches: Li et al. (2017); Xie et al. (2017) proposes linguistic adversaries techniques to enhance the robustness of the model by randomly changing the word tokens in a sentence. This explicitly modifies the semantics of a sentence. By contrast, adversarial training focuses on smaller and continuous perturbations in the embedding space while preserving the semantics of sentences. Hence, adversarial training is complementary to linguistic adversaries.

Acknowledgments

We would like to thank our anonymous reviewers for valuable discussions. We also appreciate the generous help from Angli Liu and Yankai Lin. This work is partially supported by the DARPA PPAML program, contract FA8750-14-C-0011.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Heike Adel, Benjamin Roth, and Hinrich Schütze. 2016. Comparing convolutional neural networks to traditional models for slot filling. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, California, USA, June 12 - June 17, 2016*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Rui Cai, Xiaodong Zhang, and Houfeng Wang. 2016. Bidirectional recurrent convolutional neural network for relation classification. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Mohit Iyyer, Varun Manjunatha, Jordan L Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of ACL*.

Xiaotian Jiang, Quan Wang, Peng Li, and Bin Wang. 2016. Relation extraction with multi-instance multi-label convolutional neural networks. *Proceedings of COLING*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yitong Li, Trevor Cohn, and Timothy Baldwin. 2017. Robust training under linguistic adversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

[2] When ϵ is large enough, e.g., comparable to the norm of input embeddings, and if we add the perturbation to word w and consider its nearest unperturbed word embedding in the embedding space, the nearest word will be different from the original word w. This implicitly changes the content of a sentence.

[3] The nearest neighbor of word w remains unchanged after w being perturbed with the optimal ϵ.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*, volume 1, pages 2124–2133.

Angli Liu, Stephen Soderland, Jonathan Bragg, Christopher H Lin, Xiao Ling, and Daniel S Weld. 2016. Effective crowd annotation for relation extraction. In *Proceedings of the NAACL-HLT*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2015. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of NAACL-HLT*, pages 39–48.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, volume 14, pages 1532–1543.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. *Machine learning and knowledge discovery in databases*, pages 148–163.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks.

Anders Søgaard. 2013. Part-of-speech tagging with antagonistic adversaries. In *Proceedings of ACL*, pages 640–644.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465. Association for Computational Linguistics.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics*.

Ziang Xie, Sida I Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y Ng. 2017. Data noising as smoothing in neural network language models. *arXiv preprint arXiv:1703.02573*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*, pages 1753–1762.

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

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of The 54th Annual Meeting of the Association for Computational Linguistics*, page 207.

Context-Aware Representations for Knowledge Base Relation Extraction

Daniil Sorokin and **Iryna Gurevych**
Ubiquitous Knowledge Processing Lab (UKP)
Research Training Group AIPHES
Department of Computer Science
Technische Universität Darmstadt
www.ukp.tu-darmstadt.de

Abstract

We demonstrate that for sentence-level relation extraction it is beneficial to consider other relations in the sentential context while predicting the target relation. Our architecture uses an LSTM-based encoder to jointly learn representations for all relations in a single sentence. We combine the context representations with an attention mechanism to make the final prediction.

We use the Wikidata knowledge base to construct a dataset of multiple relations per sentence and to evaluate our approach. Compared to a baseline system, our method results in an average error reduction of 24% on a held-out set of relations.

The code and the dataset to replicate the experiments are made available at https://github.com/ukplab.

1 Introduction

The main goal of relation extraction is to determine a type of relation between two target entities that appear together in a text. In this paper, we consider the sentential relation extraction task: to each occurrence of the target entity pair $\langle e_1, e_2 \rangle$ in some sentence s one has to assign a relation type r from a given set R (Hoffmann et al., 2011). A triple $\langle e_1, r, e_2 \rangle$ is called a *relation instance* and we refer to the relation of the target entity pair as *target relation*. Relation extraction is a fundamental task that enables a wide range of semantic applications from question answering (Xu et al., 2016) to fact checking (Vlachos and Riedel, 2014).

For relation extraction, it is crucial to be able to extract relevant features from the sentential context (Riedel et al., 2010; Zeng et al., 2015). Modern approaches focus just on the relation between the target entities and disregard other relations that might be present in the same sentence (Zeng et al., 2015; Lin et al., 2016). For example, in order to correctly identify the relation type between the movie e_1 and the director e_2 in (1), it is important to separate out the INSTANCE_OF relation between the movie and its type e_3:

(1) [e_1 **Star Wars VII**] is an American [e_3 **space opera epic film**] directed by [e_2 **J. J. Abrams**].

We present a novel architecture that considers other relations in the sentence as a context for predicting the label of the target relation. We use the term *context relations* to refer to them throughout the paper. Our architecture uses an LSTM-based encoder to jointly learn representations for all relations in a single sentence. The representation of the target relation and representations of the context relations are combined to make the final prediction.

To facilitate the experiments we construct a dataset that contains multiple positive and negative relation instances per sentence. We employ a fast growing community managed knowledge base (KB) Wikidata (Vrandečić and Krötzsch, 2014) to build the dataset.

Our main contribution is the new neural network architecture for extracting relations between an entity pair that takes into account other relations in the sentence.

2 Related Work

We employ a neural network to automatically encode the target relation and the sentential context into a fixed-size feature vector. Mintz et al. (2009) and Riedel et al. (2010) have used manually engineered features based on part-of-speech tags and dependency parses to represent the target relations. Recently, Zeng et al. (2015) and Zhao et al. (2015) have shown that one can successfully apply convo-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1784–1789
Copenhagen, Denmark, September 7–11, 2017.

lutional neural networks to extract sentence-level features automatically.

Most of the methods (Riedel et al., 2010; Zeng et al., 2015; Lin et al., 2016) focus on predicting a single relation type based on the combined evidence from all of the occurrences of an entity pair. Hoffmann et al. (2011) and Surdeanu et al. (2012) assign multiple relation types to each entity pair, such that the predictions are tied to particular occurrences of the entity pair. We regard the relation extraction task similarly and predict relation types on the sentence level.

We use a distant supervision approach (Mintz et al., 2009) to construct the dataset. Mintz et al. (2009) and Riedel et al. (2010) have applied it to create relation extraction datasets for a large-scale KB. In contrast to our dataset, their data contains a single relation instance per sentence. That makes it incompatible with our method.

All of the aforementioned approaches consider just the relation between the target entities and disregard other relations that might be present in the same sentence. Our method uses context relations to predict the target relation. One can also use other types of structured information from the nearby context to improve relation extraction. Roth and Yih (2004) have combined named entity recognition and relation extraction in a structured prediction approach to improve both tasks. Later, Miwa and Bansal (2016) have implemented an end-to-end neural network to construct a context representation for joint entity and relation extraction. Finally, Li et al. (2013) have designed global features and constraints to extract multiple events and their arguments from the same sentence.

We don't implement global constraints in our approach, since unlike events and arguments, there are no restrictions as to what relations can appear together. Instead we encode all relations in the same context into fixed-size vectors and use an attention mechanism to combine them.

3 Data generation with Wikidata

Wikidata is a collaboratively constructed KB that encodes common world knowledge in a form of binary relation instances (e.g. CAPITAL:P36 (*Hawaii*:Q782, *Honolulu*:Q18094))[1]. It contains more than 28 million entities and 160 million relation instances.[2] A broad community oversight, similar to Wikipedia, ensures a higher data quality compared to other KBs (Färber et al., 2015).

	Train	Validation	Held-out
# of relation triples	284,295	113,852	287,902
# of relation inst.	578,199	190,160	600,804

Table 1: Statistics of the generated dataset.

We use the complete English Wikipedia corpus to generate training and evaluation data. Wikipedia and Wikidata are tightly integrated which enables us to employ manual wiki annotations to extract high quality data. From each sentence in a complete article we extract link annotations and retrieve Wikidata entity IDs corresponding to the linked articles. There is an unambiguous one-to-one mapping between Wikidata entities and Wikipedia articles. For example:

```
1: Input  Born in [[Honolulu|Honolulu,
   Hawaii]], Obama is a graduate of
   [[Columbia University]].
2: Links to Wikidata Ids Honolulu ↦ Q18094
          Columbia_University ↦ Q49088
```

For further processing, we filter out sentences that contain fewer than 3 annotated entities, since we need to have multiple relations per sentence for training (see Section 4).

We extract named entities and noun chunks from the input sentences with the Stanford CoreNLP toolkit (Manning et al., 2014) to identify entities that are not covered by the Wikipedia annotations (e.g. Obama in the sentence above). We retrieve IDs for those entities by searching through entity labels in Wikidata. We use HeidelTime (Strötgen and Gertz, 2013) to extract dates.

For each pair of entities, we query Wikidata for relation types that connect them. We discard an occurrence of an entity pair if the relation is ambiguous, i. e. multiple relation types were retrieved. For comparison, Surdeanu et al. (2012) report that only 7.5% of entity pairs have more than one corresponding relation type in the distantly supervised dataset of Riedel et al. (2010). The entity pairs that have no relation in the knowledge base are stored as negative instances.

The constructed dataset features 353 different relation types (out of approximately 1700 non-meta relation types in the Wikidata scheme). We split

[1] Unique IDs in Wikidata have a Q-prefix for entities and a P-prefix for relations.

[2] https://www.wikidata.org/wiki/Special:Statistics

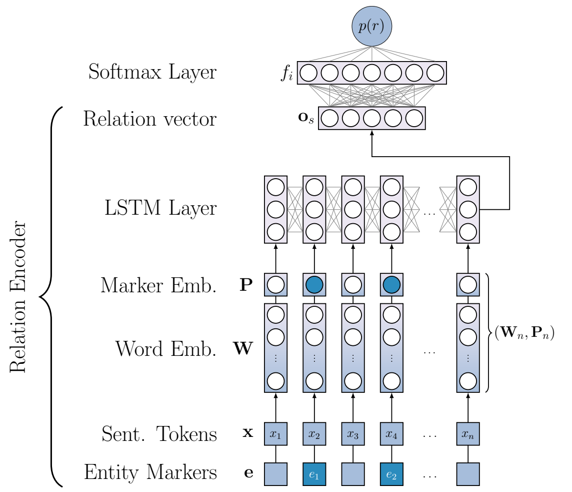


Figure 1: The architecture of the relation encoder

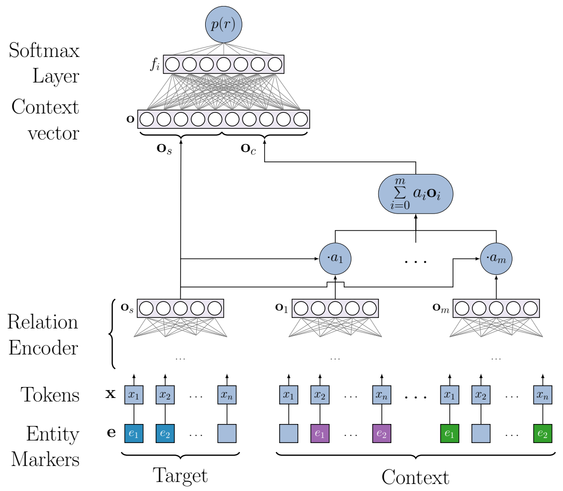


Figure 2: Incorporation of the context relations. For the ContextSum model variant $a_i = 1$.

it into train, validation and held-out sets, ensuring that there is no overlap in either sentences or relation triples between the three sets. Table 1 summarizes the statistics about the dataset. We assessed the quality of the distant supervision set-up on 200 manually verified sentences from the training set: 79.5% of relations in those sentences were correctly labeled with distant supervision (86.9 if one entity is linked, 74.7 if both are linked).

4 Model architecture

4.1 Relation encoder

The relation encoder produces a fixed-size vector representation $\mathbf{o}_s$ of a relation between two entities in a sentence (see Figure 1).

First, each token of the sentence $\mathbf{x} = \{x_1, x_2 \dots x_n\}$ is mapped to a k-dimensional embedding vector using a matrix $\mathbf{W} \in \mathbb{R}^{|V| \times k}$, where $|V|$ is the size of the vocabulary. Throughout the experiments in this paper, we use 50-dimensional GloVe embeddings pre-trained on a 6 billion corpus (Pennington et al., 2014).

Second, we mark each token in the sentence as either belonging to the first entity e_1, the second entity e_2 or to neither of those. A marker embedding matrix $\mathbf{P} \in \mathbb{R}^{3 \times d}$ is randomly initialized (d is the dimension of the position embedding and there are three marker types). For each token, we concatenate the marker embedding with the word embedding: $(\mathbf{W}_n, \mathbf{P}_n)$.

We apply a recurrent neural network (RNN) on the token embeddings. The length n naturally varies from sentence to sentence and an RNN provides a way to accommodate inputs of various sizes. It maps a sequence of n vectors to a fixed-size output vector $\mathbf{o}_s \in \mathbb{R}^o$. We take the output vector $\mathbf{o}_s$ as the representation of the relation between the target entities in the sentence. We use the Long Short-Term Memory (LSTM) variant of RNN (Hochreiter and Schmidhuber, 1997) that was successfully applied to information extraction before (Miwa and Bansal, 2016).

4.2 Model variants

LSTM baseline As the first model variant, we feed the output vector $\mathbf{o}_s$ of the relation encoder to a softmax layer to predict the final relation type for the target entity (see the upper part of Figure 1):

$$p(r|\langle e_1, e_2\rangle, \mathbf{x}; \theta) = \frac{\exp(f_r)}{\sum_{i=1}^{n_r} \exp(f_i)}, \qquad (1)$$
$$f_i = \mathbf{y}_i \cdot \mathbf{o}_s + b_i,$$

where $\mathbf{y}_i$ is a weight vector and b_i is a bias.

ContextSum We argue that for predicting a relation type for a target entity pair other context relations in the same sentence are relevant. Some relation types may tend to co-occur, such as DIRECTED_BY and PRODUCED_BY, whereas others may be restrictive (e. g. one can only have a single PLACE_OF_BIRTH).

Therefore, in addition to the target entity pair, we take other entities from the same sentence that were extracted at the data generation step. We construct a set of context relations by taking each possible pair of entities.[3] Example (2) shows a target entity pair $\langle e_1, e_2\rangle$ and context entities highlighted in bold.

[3] We limit the maximum number of relations in a sentence to 7 for computational reasons.

(2) [**Swag It Out**] is the official [**debut single**] by [**American singer**] [e_1**Zendaya**], known for starring in the series [e_2 **Shake It Up**].

We apply the same relation encoder on the target and context relations (see Figure 2). That ensures that representation for target and context relations are learned jointly. We sum the context relation representations: $\mathbf{o}_c = \sum_{i=0}^{m} \mathbf{o}_i$, where each element $\mathbf{o}_i$ is a vector representation of a single context relation. The resulting context representation $\mathbf{o}_c \in \mathbb{R}^o$ is concatenated with the vector representation of the target relation: $\mathbf{o} = [\mathbf{o}_s, \mathbf{o}_c]$. We feed the concatenated vector to the softmax layer in Eq. 1 to predict the final relation type for the target entity pair (see the upper part of Figure 2).

ContextAtt In this variant, we use a weighted sum of the context relation representation at the penultimate step:

$$\mathbf{o}_c = \sum_{i=0}^{m} a_i \mathbf{o}_i, \quad a_i = \frac{\exp(g(\mathbf{o}_i, \mathbf{o}_s))}{\sum_{j=0}^{m} \exp(g(\mathbf{o}_j, \mathbf{o}_s))}, \qquad (2)$$

where g_i computes an attention score for a context relation with respect to the target relation: $g(\mathbf{o}_i, \mathbf{o}_s) = \mathbf{o}_i \mathbf{A} \mathbf{o}_s$, and $\mathbf{A}$ is a weight matrix that is learned.

5 Experiments

5.1 Training the models

All models were trained using the Adam optimizer (Kingma and Ba, 2014) with categorical cross-entropy as the loss function. We use an early stopping criterion on the validation data to determine the number of training epochs. The learning rate is fixed to 0.01 and the rest of the optimization parameters are set as recommended in Kingma and Ba (2014): $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 1e-08$. The training is performed in batches of 128 instances.

We apply Dropout (Srivastava et al., 2014) on the penultimate layer as well as on the embeddings layer with a probability of 0.5. We choose the size of the layers (RNN layer size $o = 256$) and entity marker embeddings ($d = 3$) with a random search on the validation set.[4]

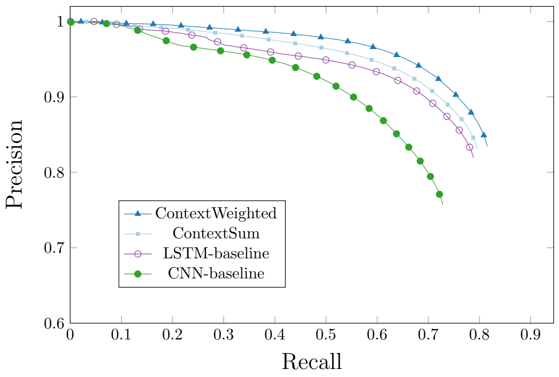


Figure 3: Aggregated precision-recall curves for the implemented models.

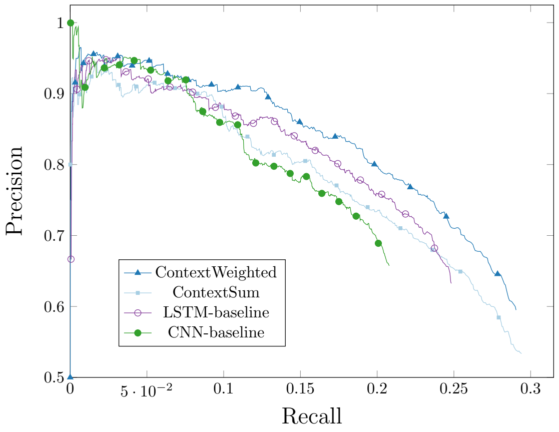


Figure 4: Aggregated macro precision-recall curves for the implemented models.

5.2 Held-out evaluation

As an additional baseline, we re-implement a sentence-level model based on convolutional neural networks (CNNs) described in Lin et al. (2016). This is a state-of-the-art model for fine-grained relation extraction that was previously tested on the single-relation dataset from Riedel et al. (2010). In addition to CNNs, their architecture uses a different position encoding scheme: position markers encode a relative position of each word with respect to the target entities.[5] We use the same GloVe word embeddings for this model and perform a hyper-parameter optimization on the validation set.

Our dataset lets us compare the baseline models and the models that use context relations on the same data. Following the previous work on rela-

[4] We test for the RNN layer size the values $\{64, 128, 256, 512\}$, for entity marker embeddings the values $\{1, 3, 5, 7\}$ and for the Dropout rate the values in the range 0.0–0.75.

[5] We have briefly experimented with such position markers for our models, but found no improvements.

Relation type	LSTM-Baseline P	 R	ContextAtt P	 R
COUNTRY	0.8899	0.9344	0.9130	0.9382
LOCATED IN	0.8329	0.8832	0.8655	0.8994
SHARES BORDER	0.7579	0.7078	0.7962	0.8075
INSTANCE OF	0.7864	0.8568	0.8478	0.8401
SPORT	0.9753	0.9828	0.9822	0.9823
CITIZENSHIP	0.9001	0.9448	0.9041	0.9417
PART OF	0.5623	0.4854	0.6269	0.5113
SUBCLASS OF	0.5230	0.4390	0.5272	0.5908

Table 2: Precision (P) and recall (R) for the top relations.

tion extraction, we report the aggregated precision-recall curves for each model on the held-out data (Figure 3).[6] To compute the curves, we rank the predictions of each model by their confidence and traverse this list top to bottom measuring the precision and recall at each step.

The models that take the context into account perform similar to the baselines at the smallest recall numbers, but start to positively deviate from them at higher recall rates. In particular, the ContextAtt model performs better than any other system in our study over the entire recall range. Compared to the competitive LSTM-baseline that uses the same relation encoder, the ContextAtt model achieves a 24% reduction of the average error: from 0.2096 ± 0.002 to 0.1590 ± 0.002. The difference between the models is statistically significant ($p = 0.009$).[7]

We also compute macro precision-recall curves that give equal weights to all relations in the dataset. Figure 4 shows that the ContextAtt model performs best over all relation types. One can also see that the ContextSum doesn't universally outperforms the LSTM-baseline. It demonstrates again that using attention is crucial to extract relevant information from the context relations.

On the relation-specific results (Table 2) we observe that the context-enabled model demonstrates the most improvement on precision and seems to be especially useful for taxonomy relations (see SUBCLASS OF, PART OF).

6 Conclusions

We have introduced a neural network architecture for relation extraction on the sentence level that takes into account other relations from the same context. We have shown by comparison with competitive baselines that these context relations are beneficial for relation extraction with a large set of relation types.

Our approach can be easily applied to other types of relation extraction models as well. For instance, Lin et al. (2016) extract sentence-level features and then combine features from multiple sentences with a selective attention mechanism. It would be possible to replace their sentence-level feature extractor with our model.

Acknowledgments

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1, and via the QA-EduInf project (grant GU 798/18-1 and grant RI 803/12-1). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

References

Michael Färber, Basil Ell, Carsten Menne, and Achim Rettinger. 2015. A Comparative Survey of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. *Semantic Web Journal*, 1:1–5.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1–32.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 541–550, Portland, Oregon, USA.

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint Event Extraction via Structured Prediction with Global Features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 73–82, Sofia, Bulgaria.

[6] We do not compare against the approach of Surdeanu et al. (2012) that also performs sentence-level relation extraction, since the provided implementation does not feature the complete pipeline and is only applicable on a particular Freebase dataset.

[7] The average error and the standard deviation are estimated on 5 training iterations for each model. The statistical significance is computed using the Wilcoxon rank-sum test on the error rates.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural Relation Extraction with Selective Attention over Instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2124–2133, Berlin, Germany.

Christopher D. Manning, John Bauer, Jenny Finkel, Steven J Bethard, Mihai Surdeanu, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*, pages 55–60, Baltimore, Maryland, USA.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Singapore, Singapore.

Makoto Miwa and Mohit Bansal. 2016. End-to-end Relation Extraction using LSTMs on Sequences and Tree Structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1105–1116, Berlin, Germany.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling Relations and Their Mentions without Labeled Text. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163, Barcelona, Spain.

Dan Roth and Wen-Tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8, Boston, Massachusetts, USA.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Jannik Strötgen and Michael Gertz. 2013. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance Multi-label Learning for Relation Extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 455–465, Jeju, Korea.

Andreas Vlachos and Sebastian Riedel. 2014. Fact Checking: Task definition and dataset construction. In *Proceedings of the ACL Workshop on Language Technologies and Computational Social Science*, pages 18–22, Baltimore, Maryland, USA.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM*, 57(10):78–85.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question Answering on Freebase via Relation Extraction and Textual Evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2326–2336, Berlin, Germany.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant Supervision for Relation Extraction via Piecewise Convolutional Neural Networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1753–1762, Lisbon, Portugal.

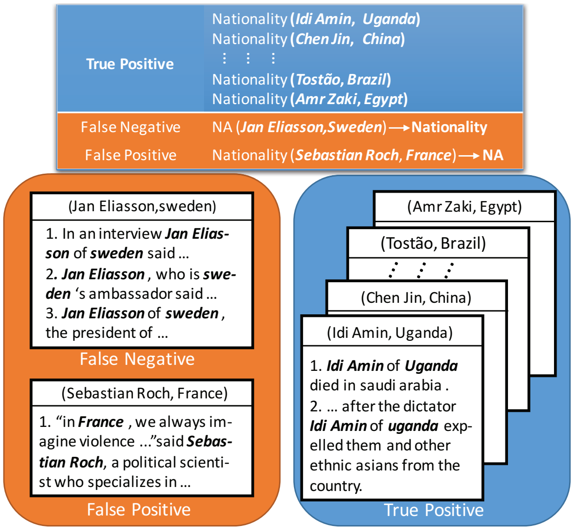
Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4069–4076, Buenos Aires, Argentina.

A Soft-label Method for Noise-tolerant Distantly Supervised Relation Extraction

Tianyu Liu, Kexiang Wang, Baobao Chang and **Zhifang Sui**
Key Laboratory of Computational Linguistics, Ministry of Education,
School of Electronics Engineering and Computer Science, Peking University, Beijing, China
{tianyu0421, wkx, chbb, szf}@pku.edu.cn

Abstract

Distant-supervised relation extraction inevitably suffers from wrong labeling problems because it heuristically labels relational facts with knowledge bases. Previous sentence level denoise models don't achieve satisfying performances because they use hard labels which are determined by distant supervision and immutable during training. To this end, we introduce an entity-pair level denoise method which exploits semantic information from correctly labeled entity pairs to correct wrong labels dynamically during training. We propose a joint score function which combines the relational scores based on the entity-pair representation and the confidence of the hard label to obtain a new label, namely a soft label, for certain entity pair. During training, soft labels instead of hard labels serve as gold labels. Experiments on the benchmark dataset show that our method dramatically reduces noisy instances and outperforms the state-of-the-art systems.


True Positive
Nationality (*Idi Amin, Uganda*)
Nationality (*Chen Jin, China*)
⋮ ⋮ ⋮
Nationality (*Tostão, Brazil*)
Nationality (*Amr Zaki, Egypt*)
False Negative
NA (*Jan Eliasson,Sweden*) ⟶ **Nationality**
False Positive
Nationality (*Sebastian Roch, France*) ⟶ **NA**
(Jan Eliasson,sweden)
False Negative
(Sebastian Roch, France)
False Positive
(Amr Zaki, Egypt)
(Tostão, Brazil)
(Chen Jin, China)
(Idi Amin, Uganda)
True Positive

1. In an interview ***Jan Eliasson*** of ***sweden*** said ...
2. ***Jan Eliasson*** , who is ***sweden*** 's ambassador said ...
3. ***Jan Eliasson*** of ***sweden*** , the president of ...

1. "in ***France*** , we always imagine violence ..."said ***Sebastian Roch***, a political scientist who specializes in ...

1. ***Idi Amin*** of ***Uganda*** died in saudi arabia .
2. ... after the dictator ***Idi Amin*** of ***uganda*** expelled them and other ethnic asians from the country.

Figure 1: An example of soft-label correction on *Nationality* relation. We intend to use syntactic/ semantic information of correctly labeled entity pairs (blue) to correct the false positive and false negative instances (orange) during training.

1 Introduction

Relation Extraction (RE) aims to obtain relational facts from plain text. Traditional supervised RE systems suffer from lack of manually labeled data. Mintz et al. (2009) proposes distant supervision, which exploits relational facts in knowledge bases (KBs). Distant supervision automatically generates training examples by aligning entity mentions in plain text with those in KB and labeling entity pairs with their relations in KB. If there's no relation link between certain entity pair in KB, it will be labeled as negative instance (NA). However, the automatic labeling inevitably accompanies with wrong labels because the relations of entity pairs might be missing from KBs or mislabeled.

Multi-instances learning (MIL) is proposed by Riedel et al. (2010) to combat the noise. The method divides the training set into multiple bags of entity pairs (shown in Fig 1) and labels the bags with the relations of entity pairs in the KB. Each bag consists of sentences mentioning both head and tail entities. Much effort has been made in reducing the influence of noisy sentences within the bag, including methods based on at-least-one assumption (Hoffmann et al., 2011; Ritter et al., 2013; Zeng et al., 2015) and attention mechanisms over instances (Lin et al., 2016; Ji et al., 2017).

However, the sentence level denoise methods can't fully address the wrong labeling problem largely because they use a hard-label method in which the labels of entity pairs are immutable dur-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1790–1795
Copenhagen, Denmark, September 7–11, 2017.

ing training, no matter whether they are correct or not. As shown in Fig 1, due to the absence of (*Jan Eliasson*[1] *, Sweden*) from *Nationality* relation in the KB, the entity pair is mislabeled as NA. However, we find the sentences in the bag of (*Jan Eliasson, Sweden*) share similar semantic pattern *"X of Y"* with correctly labeled instances (blue). In the false positive instance, *Sebastian Roch* is indeed from *France*, but the syntactic pattern of the sentence in the bag differs greatly from those of correctly labeled instances. Actually, the reliability of a distant-supervised (DS) label can be determined by the syntactic/semantic similarity between certain instance and the potential correctly labeled instances. Soft-label method intends to utilize corresponding similarities to correct wrong DS labels in the training stage dynamically, which means the same bag may have different soft labels in different epochs of training. The basis of soft-label method is the dominance of correctly labeled instances. Fortunately, Xu et al. (2013) proves that correctly labeled instances account for 94.4% (including true negatives) in the distant-supervised New York Times corpus (benchmark dataset).

To this end, we introduce a soft-label method to correct wrong labels at entity-pair level during training by exploiting semantic/syntactic information from correctly labeled instances. In our model, the representation of certain entity pair is a weighted combination of related sentences which are encoded by piecewise convolutional neural network (PCNN) (Zeng et al., 2015). Besides, we propose a joint score function to obtain soft labels during training by taking both the confidence of DS labels and the entity-pair representations into consideration. Our contributions are three-fold:

- To the best of our knowledge, we first propose an entity-pair level noise-tolerant method while previous works only focused on sentence level noise.

- We propose a simple but effective method called soft-label method to dynamically correct wrong labels during training. Case study shows our corrections are of high accuracy.

- We evaluate our model on the benchmark dataset and achieve substantial improvement compared with the state-of-the-art systems.

[1] Jan Eliasson is a Swedish diplomat.

2 Methodology

Multi-instances learning (MIL) framework splits the training set $\mathbf{M}$ into multiple entity-pair bags $\{\langle h_1, t_1\rangle, \langle h_2, t_2\rangle, \cdots, \langle h_n, t_n\rangle\}$. Each bag $\langle h_i, t_i\rangle$ contains sentences $\{x_1, x_2, \cdots, x_c\}$ which mention both head entity h_i and tail entity t_i. The representation $\mathbf{s}_i$ of bag $\langle h_i, t_i\rangle$ is a weighted combination of related sentence vectors $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_c\}$ which are encoded by CNN. Finally, we use soft-label score function to correct wrong labels of bags of entity pairs while computing probabilities for each relation type.

2.1 Sentence Encoder

We get the representation of certain sentence $\mathbf{x}_i = \{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_m\}$ by concatenating word embeddings $\{w_1, w_2, \cdots, w_m\}$ and position embeddings (Zeng et al., 2014) $\{p_1, p_2, \cdots, p_m\}$, where $\mathbf{w}_i \in \mathbb{R}^d, w_i \in \mathbb{R}^{d_w}, p_i \in \mathbb{R}^{d_p} (d = d_w + d_p)$.

Convolution layer utilizes a sliding window of size l. We define $\mathbf{q}_i \in \mathbb{R}^{l \times d}$ as the concatenation of words within the i-th window.

$$\mathbf{q}_i = \mathbf{w}_{i-l+1:i} (1 \leq i \leq m + l - 1) \quad (1)$$

The convolution matrix is denoted by $\mathbf{W}_c \in \mathbb{R}^{d_c \times (l \times d)}$, where d_c is the sentence embedding size. The i-th filter of the convolutional layer is computed as:

$$\mathbf{f}_i = [\mathbf{W}_c \mathbf{q} + \mathbf{b}]_i \quad (2)$$

Afterwards, Piecewise max-pooling (Zeng et al., 2015) is used to divide convolutional filter $\mathbf{f}_i$ into three parts $\{\mathbf{f}_i^1, \mathbf{f}_i^2, \mathbf{f}_i^3\}$ by head and tail entities. For example, the sentence "*Barack Obama* was born in *Honulul*u in 1961" are divided into '*Barack Obama*', 'was born in *Honululu*' and 'in 1961'. We perform max-pooling on these three parts separately, and the i-th element of sentence vector $\mathbf{x} \in \mathbb{R}^{d_c}$ is defined as the concatenation of them:

$$\mathbf{x}_i = [max(\mathbf{f}_i^1); max(\mathbf{f}_i^2); max(\mathbf{f}_i^3)] \quad (3)$$

2.2 Sentence Level Weight distribution

The representation of entity pair $\langle h_i, t_i\rangle$ is defined as a weighted combination of sentences in the bag. **At-least-one:** At-least-one assumption is a down sampling method which assumes at least one sentence in the bag will express the relation between two entities, and select the most likely sentence in the bag for training and prediction. To be more specific, the weight of the selected sentence is 1 while those of other sentences in the bag are all 0.

Selective Attention: Lin et al. (2016) proposes selective attention mechanism to reduce weights of noisy instances within the entity-pair bag.

$$\mathbf{s} = \sum_i \alpha_i \mathbf{x}_i; \alpha_i = \frac{\exp(\mathbf{x}_i \mathbf{A} \mathbf{r})}{\sum_k \exp(\mathbf{x}_k \mathbf{A} \mathbf{r})} \qquad (4)$$

where α_i is the weight of sentence vector $\mathbf{x}_i$, $\mathbf{A}$ and $\mathbf{r}$ are diagonal and relation query parameters.

2.3 Soft-label Adjustment

The key of our method is to derive a soft label as the gold label for each bag dynamically during training, which is not necessarily the same label as the distant supervised (DS) label. We still use DS labels while testing.

The soft label is determined dynamically, which means the same bag may have different soft labels in different training epochs. we propose following joint function to determine the soft label r_i for entity pair $\langle h_i, t_i \rangle$:

$$r_i = \arg\max(\mathbf{o} + \max(\mathbf{o})\mathbf{A} \odot L_i) \qquad (5)$$

where $\mathbf{o}, \mathbf{A}, L_i \in \mathbb{R}^{d_r}$ (d_r is the number of predefined relations). One-hot vector L_i indicates the DS label of $\langle h_i, t_i \rangle$. Relation Confidence vector $\mathbf{A}$ represents the reliability of DS labels. Each element in $\mathbf{A}$ is a decimal between 0 and 1, which indicates the confidence of corresponding DS labeled relation type. $\odot$ operation represents element-wise production. $\mathbf{o}$ is the vector of relational scores based on the entity-pair representation $\mathbf{s}_i$ of $\langle h_i, t_i \rangle$. $\max(\mathbf{o})$ is a scaling constant which restricts the effect of the DS label. The score of the t-th relation type $\mathbf{o}_t$ is calculated based on the trained relation matrice $\mathbf{M}$ and bias $\mathbf{b}$:

$$\mathbf{o}_t = \frac{\exp(\mathbf{M}\mathbf{s}_t + \mathbf{b})}{\sum_k \exp(\mathbf{M}\mathbf{s}_k + \mathbf{b})} \qquad (6)$$

We use entity-pair level cross-entropy loss function using soft labels as gold labels while training:

$$J(\theta) = \sum_{i=1}^{n} \log p(r_i | \mathbf{s}_i; \theta) \qquad (7)$$

In the testing stage, we still use the DS label l_i of certain entity pair $\langle h_i, t_i \rangle$ as the gold label:

$$G(\theta) = \sum_{i=1}^{n} \log p(l_i | \mathbf{s}_i; \theta) \qquad (8)$$

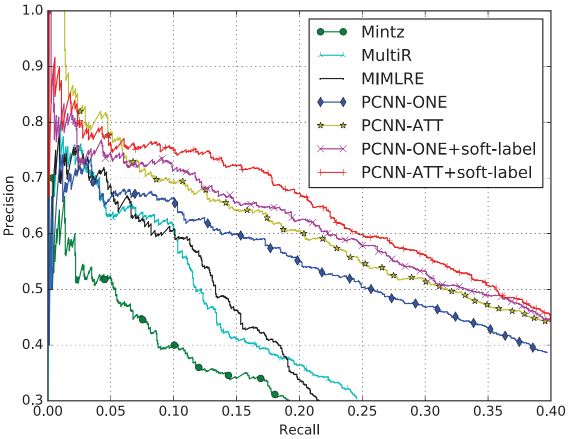


Figure 2: Precision/Recall curves of our model and previous state-of-the-art systems. Mintz (Mintz et al., 2009), MultiR (Hoffmann et al., 2011) and MIMLRE (Surdeanu et al., 2012) are feature-based models. ONE (Zeng et al., 2015) and ATT (Lin et al., 2016) are neural network models based on at-least-one assumption and selective attention, respectively.

3 Experiments

In this section, we first introduce the dataset and evaluation metrics in our experiments. Then, we demonstrate the parameter settings in our experiments. Besides, we compare the performance of our method with state-of-the-art feature-based and neural network baselines. Case study shows our soft-label corrections are of high accuracy.

3.1 Dataset and Evaluation Metrics

We evaluate our model on the benchmark dataset proposed by Mintz et al. (2009), which has also been used by Riedel et al. (2010), Hoffmann et al. (2011), Zeng et al. (2015) and Lin et al. (2016). The dataset uses Freebase (Bollacker et al., 2008) as distant-supervised knowledge base and New York Times (NYT) corpus as text resource. Sentences in NYT of the years 2005-2006 are used as training set while sentences in NYT of 2007 are used as testing set. There are 53 possible relations including NA which indicates no relation. The training data contains 522611 sentences, 281270 entity pairs and 18252 relational facts. The testing data contains 172448 sentences, 96678 entity pairs and 1950 relational facts.

Similar to the previous work, We report both aggregate precision/recall curves and top-N precision (P@N).

Window size	Word dimension	Position dimension	Filter dimension	Batch size	Learning rate	Dropout
$l = 3$	$d_w = 50$	$d_p = 5$	$d_c = 230$	$B = 160$	$\lambda = 0.001$	$p = 0.5$

Table 1: Parameter settings of our experiments.

Settings	One				Two				All			
P@N(%)	100	200	300	Avg	100	200	300	Avg	100	200	300	Avg
ONE	73.3	64.8	56.8	65.0	70.3	67.2	63.1	66.9	72.3	69.7	64.1	68.7
+soft-label	77.0	72.5	67.7	72.4	80.0	74.5	69.7	74.7	84.0	81.0	74.0	79.7
ATT	73.3	69.2	60.8	67.8	77.2	71.6	66.1	71.6	76.2	73.1	67.4	72.2
+soft-label	**84.0**	**75.5**	**68.3**	**75.9**	**86.0**	**77.0**	**73.3**	**78.8**	**87.0**	**84.5**	**77.0**	**82.8**

Table 2: Top-N precision (P@N) for relation extraction in the entity pairs with different number of sentences. Following (Lin et al., 2016), One, Two and All test settings random select one/two/all sentences on the bags of entity pairs from the testing set which have more than one sentence to predict relation.

3.2 Comparison with previous work

Mintz (Mintz et al., 2009), MultiR (Hoffmann et al., 2011) and MIMLRE (Surdeanu et al., 2012) are feature-based models. PCNN-ONE (Zeng et al., 2015) and PCNN-ATT (Lin et al., 2016) are piecewise convolutional neural network (PCNN) models based on at-least-one assumption and selective attention, which are introduced in Section 2.2, respectively. All the results of compared models come from the data reported in their papers.

3.3 Experimental Settings

We use cross-validation to determine the parameters in our model. Soft-label method uses PCNN-ONE/PCNN-ATT to represent the bags of entity pairs, and we don't tune on the parameters of PCNN-ONE/PCNN-ATT for fair comparsion. So we use the same pre-trained word embeddings and parameters of CNN encoder as those of Lin et al. (2016). Detailed parameter settings are shown in Table 1. Moreover, we use Adam optimizer. Besides, to avoid negative effects of dominant NA instances in the begining of training, soft-label method is adopted after 3000 steps of parameter updates. The confidence vector **A** is heuristically set as $[0.9, 0.7, \cdots, 0.7]$ (the confidence of NA is 0.9 while confidence of other relations are all 0.7).

3.4 Precision Recall Curve

We have following observations from Figure 2: (1) For both ATT and ONE configuration, soft-label method achieves higher precision than baselines when recall is greater than 0.05. After manual evaluation, we find that most wrong instances with less than 0.05 recall are wrong labeling entity pairs in test set. (2) Even weaker baseline PCNN-ONE using soft labels gets a slightly better performance than PCNN-ATT. (3) When recall is between 0.05 and 0.15, the curve of our model ATT+soft-label is relatively stable, which demonstrates soft-label can obtain relatively stable performance in extracting relational facts.

False positive: Place lived → **Place of death**
Fernand nault , one of canada 's foremost dance figures , died in ***montreal*** on tuesday .
False positive: Place lived → **NA**
Alexandra pelosi, a daughter of representative nancy pelosi $\cdots$, and paul pelosi of ***san francisco***, was married yesterday to $\cdots$.
False Negative: NA → **Nationality**
By spring the renowned chef ***Gordon Ramsay*** of ***England*** should be in hotels here.
False Negative: NA → **Work in**
$\cdots$, said ***Billy Ccox*** , a spokesman for the ***United States Department of Agriculture***.

Table 3: Some examples of soft-label corrections while training

3.5 Top N precision

Table 2 shows top-N precision (P@N) of the state-of-the-art systems and our model. We can see that (1) For both PCNN-ONE and PCNN-ATT model, soft-label method improves the precisions by over 10% in all test settings, which demonstrates the effects of our model. (2) Even a weaker baseline (PCNN-ONE) with soft-label method achieves higher precision than a strong model (PCNN-ATT). It shows that entity-pair level denoise model performs much better than the models which only focus on sentence level noise.

Case 1: **Place of Birth** → Nationality
Marcus Samuelsson began ··· when he was visiting his native ***Ethiopia***. ***Marcus Samuelsson*** chef born in ***Ethiopia*** and raised in Sweden ···.
Case 2: **Location Contains** → NA
···, he is from neighboring towns in ***Georgia*** (such as Blairsville and ***Young Harris***)

Table 4: Two typical wrong corrections of soft-label adjustment during training.

3.6 Case Study

Some examples of soft-label corrections during training are shown in Table 3. We can see that soft-label method can recognize both false positives and false negatives during training and correct wrong labels successfully. The two sentences above are mislabeled as *place lived* because triple facts *(Fernand nault, place lived, Montreal)* and *(Alexandra pelosi, place lived, San francisco)* exist in Freebase. However, the two sentences fail to express *place lived* relation. Our model can automatically correct them by soft-label adjustment. The two sentences below show that our model can also change false negative (NA) examples caused by missing facts in Freebase to correct ones.

Besides, our model has strong ability to distinguish different relational patterns, even for similar relations like *Place lived, Place of born, Place of Death*.

4 Error Analysis

We randomly select 200 instances of soft-label corrections during training for PCNN-ONE and PCNN-ATT respectively and check them manually. The accuracy of soft-label corrections for PCNN-ONE is 88.5% (177/200) while that for PCNN-ATT is 92% (184/200). We notice that the accuarcy of PCNN-ATT+soft-label is slightly higher than that of PCNN-ONE+soft-label. The condition is the same as our expectation. As explained in Sec 2.2, PCNN-ATT has better bag representations than PCNN-ONE because it can reduce the effect of noisy instances within the bag. The soft-label of certain bag is determined by its bag representation and the confidence of corresponding DS label. So the accuracy of soft-label corrections for PCNN-ATT can benefit from better bag representations.

Although most of soft-label corrections are of high accuracy (90.25%), there are still several wrong corrections. Table 4 lists two typical wrong corrections during training. Wrong corrections like Case 1 fail to distinguish similar relations (both *Nationality* and *place of birth* are relations between people and locations) between entities because of their similar sentence patterns. However, wrong corrections like Case 1 are rare (5/39) in our experiments. Soft-label method can still distinguish most similiar relations as shown in Sec 3.6. In Case 2, factual relation *location contains* is mistaken as NA partially because the relational pattern of this sentence is somewhat different from the regular *location contains* pattern. Additionally, soft-label method has a tendency to label ambiguous facts as NA because negative instances (NA) are dominated in the corpus. However, most bags which are soft-labeled as NA are still well-labeled in our experiments.

We argue that the minor wrong corrections of relational facts during training don't affect the overall performance much because distant supervision doesn't lack instances of relational facts due to its strong ability to automatically label large web text.

5 Conclusion and Future Work

This paper proposes a noise-tolerant method to combat wrong labels in distant-supervised relation extraction with soft labels. Our model focuses on entity-pair level noise while previous models only dealt with sentence level noise. Our model achieves significant improvement over baselines on the benchmark dataset. Case study shows that soft-label corrections are of high accuracy.

In the future, we plan to develop a new measurement for the reliability of certain distantly supervised label by evaluating the corresponding similarity between certain instance and the potential correctly labeled instances instead of using heuristically set confidence vector. In addition, we tend to find a more suitable sentence encoder rather than piece-wise CNN for our soft-label method.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work is supported by the National Key Basic Research Program of China (No. 2014CB340504) and the National Natural Science Foundation of China (No.61375074, 61273318). The contact authors are Zhifang Sui and Baobao Chang.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3060–3066.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of ACL*, volume 1, pages 2124–2133.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Alan Ritter, Luke Zettlemoyer, Oren Etzioni, et al. 2013. Modeling missing data in distant supervision for information extraction. *Transactions of the Association for Computational Linguistics*, 1:367–378.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *ACL (2)*, pages 665–670.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal*, pages 17–21.

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

A Sequential Model for Classifying Temporal Relations between Intra-Sentence Events

Prafulla Kumar Choubey and **Ruihong Huang**
Department of Computer Science and Engineering
Texas A&M University
(prafulla.choubey, huangrh)@tamu.edu

Abstract

We present a sequential model for temporal relation classification between intra-sentence events. The key observation is that the overall syntactic structure and compositional meanings of the multi-word context between events are important for distinguishing among fine-grained temporal relations. Specifically, our approach first extracts a sequence of context words that indicates the temporal relation between two events, which well align with the dependency path between two event mentions. The context word sequence, together with a parts-of-speech tag sequence and a dependency relation sequence that are generated corresponding to the word sequence, are then provided as input to bidirectional recurrent neural network (LSTM) models. The neural nets learn compositional syntactic and semantic representations of contexts surrounding the two events and predict the temporal relation between them. Evaluation of the proposed approach on TimeBank corpus shows that sequential modeling is capable of accurately recognizing temporal relations between events, which outperforms a neural net model using various discrete features as input that imitates previous feature based models.

1 Introduction

Identifying temporal relations between events is crucial to constructing events timeline. It has direct application in tasks such as question answering, event timeline generation and document summarization.

Bush said he saw little reason to be optimistic about a settlement of the **dispute**, which stems from Iraq's **invasion** of oil-wealthy Kuwait and its subsequent military **buildup** on the border of Saudi Arabia.
Relations: (dispute $after^{rel_1}$ invasion, invasion $ibefore^{rel_2}$ buildup, dispute $after^{rel_3}$ buildup)

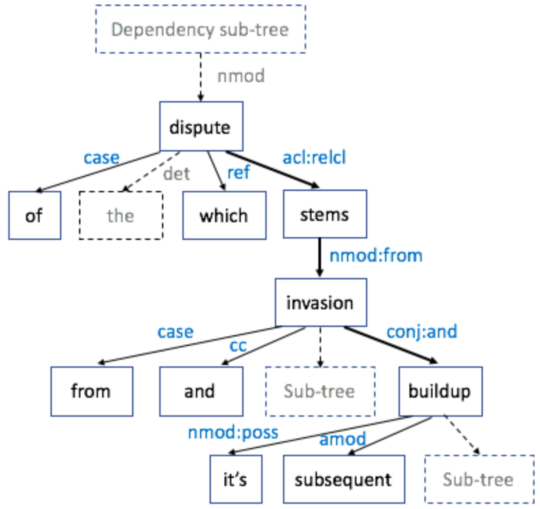


Figure 1: Example sentence to illustrate the temporal context for event pairs.

Previous works studied this task as the classification problem based on discrete features defined over lexico-syntactic, semantic and discourse features. However, these features are often derived from local contexts of two events and are only capable of capturing direct evidences indicating the temporal relation. Specifically, when two events are distantly located or are separated by other events in between, feature based approaches often fail to utilize compositional evidences, which are hard to encode using discrete features.

Consider the example sentence in Figure 1. Here, the first two temporal re-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1796–1802
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

lations, $dispute$ $\textbf{after}^{\textbf{rel}_1}$ $invasion$ and $invation$ $\textbf{ibefore}^{\textbf{rel}_2}$ $buildup$, involve events that are close by and discrete features, such as dependency relations and bag-of-words extracted from local contexts of two events, might be sufficient to correctly detect their relations. However, for the temporal relation $dispute$ $\textbf{after}^{\textbf{rel}_3}$ $buildup$, the context between the two events is long, complex and involves another event ($invasion$) as well, which makes it challenging for any individual feature or feature combinations to capture the temporal relation.

We propose that the overall syntactic structure of in-between contexts including the linear order of words as well as the compositional semantics of multi-word contexts are critical for predicting the temporal relation between two events. Furthermore, the most important syntactic and semantic structures are derived along dependency paths between two event mentions[1]. This aligns well with the observation that semantic composition relates to grammatical dependency relations (Monroe and Wang, 2014; Reddy et al., 2016).

Our approach defines rules on dependency parse trees to extract temporal relation indicating contexts. First, we extract the dependency path between two event mentions. Then we apply two heuristic rules to enrich extracted dependency paths and deal with complex syntactic structures such as punctuations. Empirically, we found that parts-of-speech tags (POS) and dependency sequences generated following the dependency path provide evidences to predict the temporal relation as well.

We use neural net sequence models to capture structural and semantic compositionality in describing temporal relations between events. Specifically, we generate three sequences for each dependency path, the word sequence, the POS tag sequence and the dependency relation sequence. Using the three types of sequences as input, we train bi-directional LSTM models that consume each of the three sequences and model compositional structural information, both syntactically and semantically.

The evaluation shows that each type of sequences is useful to temporal relation classification between events. Our complete neural net model taking all the three types of sequences performs the best, which clearly outperforms feature based models.

2 Related Works

Most of the previous works on temporal relation classification are based on feature-based classifiers. Mani et al. (2006) built MaxEnt classifier on hand-tagged features in the corpus, including tense, aspect, modality, polarity and event class for classifying temporal relations. Later Chambers et al. (2007) used a two-stage classifier which first learned imperfect event attributes and then combined them with other linguistic features in the second stage to perform the classification.

The following works mostly expanded the feature sets (Cheng et al., 2007; Bethard and Martin, 2007; UzZaman et al., 2012; Bethard, 2013; Kolomiyets et al., 2012; Chambers, 2013; Laokulrat et al., 2013). Specifically, Chambers (2013) used direct dependency path between event pairs to capture syntactic context. Laokulrat et al. (2013) used 3-grams of paths between two event mentions in a dependency tree as features instead of full paths as those are too sparse. We found that modeling the entire path as one sequence provides greater compositional evidence on the temporal relation. In addition, modifiers attached to the words in a path with specific dependency relations like *nmod:tmod* are also informative.

Ng (2013) proposed a hybrid system for temporal relation classification that combines the learned classifier with 437 hand-coded rules. Their system first applied high-accuracy rules and then used the learned classifier, trained on rich features including those high-accuracy rules as features, to classify the cases that were not handled by the rules. Ng et al. (2013) also showed the effectiveness of different discourse analysis frameworks for this task. Later Mirza and Tonelli (2014) showed that a simpler approach based on lexico-syntactic features achieved results comparable to Ng (2013). They also reported that dependency order between events, either governor-dependent or dependent-governor, was not useful in their experiments. However, we show that dependency relations, when modeled as a sequence, contribute significantly to this task.

3 Temporal Link Labeling

In this section, we describe the task of temporal relation classification, dataset, context words se-

[1] In this paper, we restrict ourselves to study temporal relation classification between event mentions that are within one sentence.

quence extraction model and the used recurrent neural net based classifier.

3.1 Task description

Early works on temporal relation classification Mani et al. (2006); Chambers et al. (2007) and the first two versions of TempEval (Verhagen et al., 2007, 2010) simplified the task by considering only six relation types. They combined the pair of relation types that are the inverse of each other and ignored the relations *during* and *during_inv*. Then TempEval-3 (Uzzaman et al., 2013) extended the task to complete 14 class classification problem and all later works have considered all 14 relations. Our model performs 14-class classification following the recent works, as this is arguably more challenging (Ng, 2013). Also, we consider gold annotated event pairs, mainly because the corpus is small and distribution of relations is very skewed. All previous works focusing on the problem of classifying temporal relation types assumed gold annotation.

3.2 Dataset

Relations	Train	Validate	Test
After	419	60	120
Before	337	48	97
Simultaneous	288	41	83
Identity	147	21	43
Includes	141	20	41
IS_included	93	13	27
Ended_by	66	9	19
During_inv	26	4	8
Begun_by	25	3	7
Begins	22	3	7
IBefore	16	2	5
IAfter	12	2	4
During	11	1	3
Ends	9	2	3
Total	1612	229	467

Table 1: Distribution of temporal relations in TimeBank v1.2.

We have used TimeBank corpus v1.2 for training and evaluating our model. The corpus consists of 14 temporal relations between 2308 event pairs, which are within the same sentence. These relations (Saurı et al., 2006) are *simultaneous, before, after, ibefore, iafter, begins, begun_by, ends, ended_by, includes, is_included, during, during_inv, identity*. Six pairs among them are inverse of each other and other two types are commutative ($e_1 \mathcal{R} e_2 \equiv e_2 \mathcal{R} e_1$, $\mathcal{R} \in \{$*identical, simultaneous*$\}$). Our sequential model requires that relation should always be between e_1 and e_2, where e_1 occurs before e_2 in the sentence. Therefore, before extracting the sequence, we inverted the relation types in cases where relation type was annotated in opposite order. Final distribution of dataset is given in Table 1.

3.3 Extracting Context Word Sequence

First, we extract words that are in the dependency path between two event mentions. However, event pairs can be very far in a sentence and are involved in complex syntactic structures. Therefore, we also apply two heuristic rules to deal with complex syntactic structures, e.g., two event mentions are in separate clauses and have a punctuation sign in their context. We describe our specific rules below. We used the Stanford parser (Chen and Manning, 2014) for generating dependency relations and parts-of-speech tags and all notations follow enhanced universal dependencies (De Marneffe and Manning, 2008).

Rule 1 (punctuation): Comma directly influences the meaning in text and omitting it may alter the meaning of phrase. Therefore, include comma if it precedes or follows e_1, e_2 or their modifiers.

Rule 2 (children): Modifiers like *now, then, will, yesterday, subsequent, when, was, etc.* contains information on the temporal order of events and help in grounding events to the timeline. These modifiers are often related to event mentions with a specific class of dependency relations. Include all such children of e_1, e_2 and other words in the path between them, which are connected with dependency relations nmod:tmod, mark, case, aux, conj, expl, cc, cop, amod, advmod, punct, ref.

3.4 Sequences and Classifier

We form three sequences on the extracted context words (with *t* words), which are based on (i) parts-of-speech tags: $\mathcal{P_T} = p_1, p_2, ..., p_t$ (ii) dependency relations: $\mathcal{D_T} = d_1, d_2, ..., d_n$[2] and (iii) word forms: $\mathcal{W_T} = w_1, w_2, ..., w_t$.

We transform each p_i and d_i to a one-hot vector and each w_i to a pre-trained embedding vector (Pennington et al., 2014). Then each sequence of vectors are encoded using their corresponding forward ($LSTM_f$) and backward ($LSTM_b$) LSTM layers.

Classifier: Figure 2 shows an overview of our model. It consists of six LSTM (Hochreiter and

[2] we only consider dependency relations for words in path connecting e_1 and e_2.

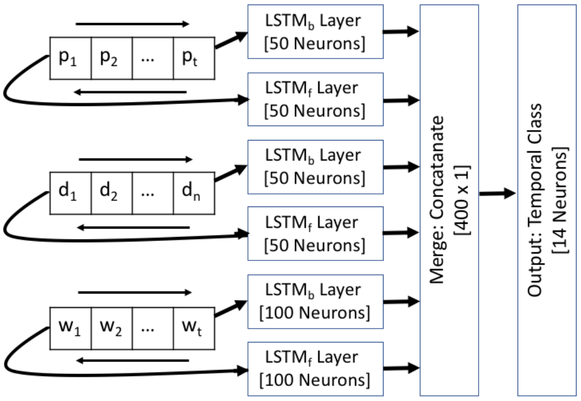


Figure 2: Bi-directional LSTM based classifier used for temporal relations classification.

Schmidhuber, 1997) layers, three of them encode feature sequences in forward order and remaining in reverse order. LSTM layers for POS tag and dependency relation have 50 neurons and have dropouts of 0.20. LSTM layers for word form have 100 neurons and have dropout of 0.25. All LSTM layers use 'tanh' activation function. Forward and backward embeddings of all sequences are concatenated and fed into another neural layer with 14 neurons corresponding to 14 fine-grained temporal relations. This neural layer uses softmax activation function. We train model for 100 iterations using rmsprop optimizer on batch size of 100 and error defined by categorical cross-entropy (Chollet, 2015) .

4 Evaluation

We evaluate our model using accuracy which has been used in previous research works for temporal relation classification. We also compare model performance using per-class F-score and macro F-score. We briefly describe all the systems we have used for evaluation.

Majority Class: assigns "after" relation to all event pairs.

Unidirectional LSTMs: use single LSTM layer to encode each sequence (POS tags, dependency relation and word forms) individually for extracted phrase in forward order.

Bidirectional LSTMs: use two LSTM layers to encode each sequence individually, taken from POS tags, dependency and word forms sequences. The first layer encodes sequence in forward and second in reverse order.

2 Sequences: bi-directional LSTM based models considering all combinations of two sequences taken from POS tags, dependency and word forms sequences.

Full model: our complete sequential model considering POS, dependency and word forms sequences.

Direct dependency path: the same as **Full model** except that the two heuristic rules were not applied in extracting sequences.

Baseline I: a neural network classifier using discrete features described in Mirza and Tonelli (2014); Ng (2013). The features used are: POS tag, dependency relation, token and lemma of $e_1(e_2)$; dependency relations between $e_1(e_2)$ and their children; binary features indicating if e_1 and e_2 are related with the 'happens-before' or the 'similar' relation according to VerbOcean (Chklovski and Pantel, 2004), if e_1 and e_2 have the same POS tag, or if $e_1(e_2)$ is the root and e_1 modifies (or governs) e_2; the dependency relation between e_1 and e_2 if they are directly connected in the dependency parse tree; prepositions that modify (or govern) $e_1(e_2)$; signal words (Derczynski and Gaizauskas, 2012) and entity distance between e_1 and e_2. These features are concatenated and fed into an output neural layer with 14 neurons.

Baseline II: a neural network classifier using POS tags and word forms of words in the *surface path* as input. The *surface path* consists of words that lie in between two event mentions based on the original sentence. The classifier uses four LSTM layers to encode both POS tag and word sequences in forward and backward order. The output neural layer and parameters for all LSTM layers are kept the same as the *Full model*.

Baseline III: a neural network classifier based on event embeddings for both event mentions that were learned using bidirectional LSTMs (Kiperwasser and Goldberg, 2016). The learning uses two LSTM layers, each with 150 neurons and dropout of 0.2, to embed the forward and backward representations for each event mention. The input to LSTM layers are sequences of concatenated word embeddings and POS tags; each sequence corresponding to 19 context words to the left or to the right side of an event mention for the forward or the backward LSTM layer respectively. Event embeddings are then concatenated and fed into an output neural layer with 14 neurons.

All baselines are trained using rmsprop optimizer on an objective function defined by categorical cross entropy and their output layer uses softmax activation function.

4.1 Results and Discussion

Models	*Accuracy*
Majority Class	25.69
Baseline I	41.97
Unidirectional LSTM: only POS	34.90
only Word	35.12
only Dependency	34.48
Bidirectional LSTMs: only POS	39.19
only Word	37.69
only Dependency	40.04
2 Sequences: POS + Word	44.54
Dependency + Word	45.18
Dependency + POS	47.75
Full Model	**53.32**
Direct dependency path	49.25
Baseline II	43.90
Baseline III	44.75

Table 2: Temporal relation classification result on TimeBank corpus.

Relations	*OurSystem*			*BaselineI*		
	P	R	F	P	R	F
After	**0.62**	**0.68**	**0.65**	0.56	0.48	0.45
Before	**0.56**	**0.52**	**0.53**	0.37	0.45	0.41
Simultan.	**0.44**	**0.51**	**0.47**	0.32	0.43	0.37
Identity	**0.47**	**0.56**	**0.51**	0.45	0.53	0.49
Includes	**0.59**	**0.39**	**0.47**	0.43	0.30	0.35
IS_includ.	0.5	**0.56**	0.53	**0.61**	0.51	**0.56**
Ended_by	**0.48**	**0.63**	**0.55**	0.41	0.47	0.44
During_in.	0	0	0	0	0	0
Begun_by	**0.75**	**0.43**	**0.55**	0	0	0
Begins	**1.0**	**0.29**	**0.44**	0	0	0
IBefore	**0.4**	**0.4**	**0.4**	0	0	0
IAfter	**0.33**	**0.25**	**0.29**	0	0	0
During	0	0	0	0	0	0
Ends	0	0	0	0	0	0
Macro Av.	**0.44**	**0.37**	**0.40**	0.23	0.22	0.22

Table 3: Per-class results of our best system and the baseline I.

Table 2 reports accuracy scores for all the systems. We see that simple sequential models outperform the strong feature based system, *Baseline I*, which used various discrete features. Note that dependency relation and POS tag sequences alone achieve reasonably high accuracies. This implies that an important aspect of temporal relation is contained in the syntactic context of event mentions. Moreover, Mirza and Tonelli (2014) observed that discrete features based on dependency parse tree did not contribute to improving their classifier's accuracy. On the contrary, using the sequence of dependency relations yields a high accuracy in our setting which signifies the advantages of using sequential representations for this task. Our *Full Model* achieves a performance gain of 11.35% over *Baseline I*.

We developed two more baselines (*Baseline II and III*) that do not require syntactic information as well as the *Direct dependency path* model that used no rules. The *Full Model* outperformed them by 9.42%, 8.57% and 4.07% respectively. This affirms that the most useful syntactic and semantic structures are derived along dependency paths and additional context words, including prepositions, signal words and punctuations that are indirectly attached to event words, entail evidence on temporal relations as well.

Table 3 compares precision, recall and F_1 scores of our *Full Model* with *Baseline I*. Our model performs reasonably well compared to the baseline system for most of the classes. In addition, it is able to identify relations present in small proportion like *begun_by, ibefore, iafter etc.*, which the baseline system couldn't identify. A similar observation was also reported by Mirza and Tonelli (2014) that relation types *begins, ibefore, ends* and *during* are difficult to identify using feature based systems, which often generate false positives for *before* and *after* relations.

5 Conclusion and Future work

In this paper, we have focused on modeling syntactic structural information and compositional semantics of contexts in predicting temporal relations between events in the same sentence. Our approach extracts lexical and syntactic sequences from contexts between two events and feed them to recurrent neural nets. The evaluation shows that our sequential models are promising in distinguishing among fine-grained temporal relations.

In the future, we will extend our sequential models to predict temporal relations for event pairs spanning across multiple sentences, for instance by incorporating discourse relations between sentences in a sequence.

Acknowledgments

We want to thank our anonymous reviewers for their insightful review comments and suggestions that helped making evaluations more extensive.

References

Steven Bethard. 2013. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 2, pages 10–14.

Steven Bethard and James H Martin. 2007. Cu-tmp: Temporal relation classification using syntactic and semantic features. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 129–132. Association for Computational Linguistics.

Nathanael Chambers. 2013. Navytime: Event and time ordering from raw text. Technical report, DTIC Document.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 173–176. Association for Computational Linguistics.

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

Yuchang Cheng, Masayuki Asahara, and Yuji Matsumoto. 2007. Naist. japan: Temporal relation identification using dependency parsed tree. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 245–248. Association for Computational Linguistics.

Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *EMNLP*, volume 4, pages 33–40.

François Chollet. 2015. Keras. https://github.com/fchollet/keras.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. Stanford typed dependencies manual. Technical report, Technical report, Stanford University.

Leon Derczynski and Robert Gaizauskas. 2012. Using signals to improve automatic classification of temporal relations. *arXiv preprint arXiv:1203.5055*.

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

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *arXiv preprint arXiv:1603.04351*.

Oleksandr Kolomiyets, Steven Bethard, and Marie-Francine Moens. 2012. Extracting narrative timelines as temporal dependency structures. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 88–97. Association for Computational Linguistics.

Natsuda Laokulrat, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Uttime: Temporal relation classification using deep syntactic features.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 753–760. Association for Computational Linguistics.

Paramita Mirza and Sara Tonelli. 2014. Classifying temporal relations with simple features. In *EACL*, volume 14, pages 308–317.

Will Monroe and Yushi Wang. 2014. Dependency parsing features for semantic parsing.

Jun-Ping Ng, Min-Yen Kan, Ziheng Lin, Vanessa Wei Feng, Bin Chen, Jian Su, and Chew Lim Tan. 2013. Exploiting discourse analysis for article-wide temporal classification. In *EMNLP*, pages 12–23.

Vincent Ng. 2013. Classifying temporal relations with rich linguistic knowledge.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Roser Saurı, Jessica Littman, Bob Knippen, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky. 2006. Timeml annotation guidelines version 1.2. 1.

Naushad UzZaman, Hector Llorens, James Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. 2012. Tempeval-3: Evaluating events, time expressions, and temporal relations. *arXiv preprint arXiv:1206.5333*.

Naushad Uzzaman, Hector Llorens, Leon Derczynski, Marc Verhagen, James Allen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 75–80. Association for Computational Linguistics.

Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 57–62. Association for Computational Linguistics.

Deep Residual Learning for Weakly-Supervised Relation Extraction

Yi Yao Huang
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan
b02901042@ntu.edu.tw

William Yang Wang
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
william@cs.ucsb.edu

Abstract

Deep residual learning (ResNet) (He et al., 2016) is a new method for training very deep neural networks using identity mapping for shortcut connections. ResNet has won the ImageNet ILSVRC 2015 classification task, and achieved state-of-the-art performances in many computer vision tasks. However, the effect of residual learning on noisy natural language processing tasks is still not well understood. In this paper, we design a novel convolutional neural network (CNN) with residual learning, and investigate its impacts on the task of distantly supervised noisy relation extraction. In contradictory to popular beliefs that ResNet only works well for very deep networks, we found that even with 9 layers of CNNs, using identity mapping could significantly improve the performance for distantly-supervised relation extraction.

1 Introduction

Relation extraction is the task of predicting attributes and relations for entities in a sentence (Zelenko et al., 2003; Bunescu and Mooney, 2005; GuoDong et al., 2005). For example, given a sentence *"**Barack Obama** was born in **Honolulu**, Hawaii."*, a relation classifier aims at predicting the relation of *"**bornInCity**"*. Relation extraction is the key component for building relation knowledge graphs, and it is of crucial significance to natural language processing applications such as structured search, sentiment analysis, question answering, and summarization.

A major issue for relation extraction is the lack of labeled training data. In recent years, distant supervision (Mintz et al., 2009; Hoffmann et al., 2011; Surdeanu et al., 2012) emerges as the most popular method for relation extraction— it uses knowledge base facts to select a set of noisy instances from unlabeled data. Among all the machine learning approaches for distant supervision, the recently proposed Convolutional Neural Networks (CNNs) model (Zeng et al., 2014) achieved the state-of-the-art performance. Following their success, Zeng et al. (2015) proposed a piece-wise max-pooling strategy to improve the CNNs. Various attention strategies (Lin et al., 2016; Shen and Huang, 2016) for CNNs are also proposed, obtaining impressive results. However, most of these neural relation extraction models are relatively shallow CNNs—typically only one convolutional layer and one fully connected layer were involved, and it was not clear whether deeper models could have benefits on distilling signals from noisy inputs in this task.

In this paper, we investigate the effects of training deeper CNNs for distantly-supervised relation extraction. More specifically, we designed a convolutional neural network based on residual learning (He et al., 2016)—we show how one can incorporate word embeddings and position embeddings into a deep residual network, while feeding identity feedback to convolutional layers for this noisy relation prediction task. Empirically, we evaluate on the NYT-Freebase dataset (Riedel et al., 2010), and demonstrate the state-of-the-art performance using deep CNNs with identify mapping and shortcuts. In contrast to popular beliefs in vision that deep residual network only works for very deep CNNs, we show that even with a moderately deep CNNs, there are substantial improvements over vanilla CNNs for relation extraction. Our contributions are three-fold:

- We are the first to consider deeper convolutional neural networks for weakly-supervised

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1803–1807
Copenhagen, Denmark, September 7–11, 2017.

relation extraction using residual learning;

- We show that our deep residual network model outperforms CNNs by a large margin empirically, obtaining state-of-the-art performances;
- Our identity mapping with shortcut feedback approach can be easily applicable to any variants of CNNs for relation extraction.

2 Deep Residual Networks for Relation Extraction

In this section, we describe a novel deep residual learning architecture for distantly supervised relation extraction. Figure 1 describes the architecture of our model.

2.1 Vector Representation

Let $\mathbf{x}_i$ be the *i*-th word in the sentence and *e1*, *e2* be the two corresponding entities. Each word will access two embedding look-up tables to get the word embedding $\mathbf{WF}_i$ and the position embedding $\mathbf{PF}_i$. Then, we concatenate the two embeddings and denote each word as a vector of $\mathbf{v}_i = [\mathbf{WF}_i, \mathbf{PF}_i]$.

2.1.1 Word Embeddings

Each representation $\mathbf{v}_i$ corresponding to $\mathbf{x}_i$ is a real-valued vector. All of the vectors are encoded in an embeddings matrix $\mathbf{V}_w \in \mathbb{R}^{d_w \times |V|}$ where V is a fixed-sized vocabulary.

2.1.2 Position Embeddings

In relation classification, we focus on finding a relation for entity pairs. Following (Zeng et al., 2014), a PF is the combination of the relative distances of the current word to the first entity e_1 and the second entity e_2. For instance, in the sentence ”*Steve_Jobs is the founder of Apple.*”, the relative distances from *founder* to e_1 (*Steve_Job*) and e_2 are 3 and -2, respectively. We then transform the relative distances into real valued vectors by looking up one randomly initialized position embedding matrices $\mathbf{V}_p \in \mathbb{R}^{d_p \times \|P\|}$ where P is fixed-sized distance set. It should be noted that if a word is too far from entities, it may be not related to the relation. Therefore, we choose maximum value e_{max} and minimum value e_{min} for the relative distance.

In the example shown in Figure 1, it is assumed that d_w is 4 and d_p is 1. There are two position embeddings: one for e_1, the other for e_2. Finally, we concatenate the word embeddings and position embeddings of all words and denote a sentence of length *n* (padded where necessary) as a vector

$$\mathbf{v} = \mathbf{v}_1 \oplus \mathbf{v}_2 \oplus \ldots \oplus \mathbf{v}_n$$

where $\oplus$ is the concatenation operator and $\mathbf{v}_i \in \mathbb{R}^d$ $(d = d_w + d_p \times 2)$.

2.2 Convolution

Let $\mathbf{v}_{i:i+j}$ refer to the concatenation of words $\mathbf{v}_i, \mathbf{v}_{i+1}, \ldots, \mathbf{v}_{i+j}$. A convolution operation involves a *filter* $\mathbf{w} \in \mathbb{R}^{hd}$, which is applied to a window of *h* words to produce a new feature. A feature c_i is generated from a window of word $\mathbf{v}_{i:i+h-1}$ by

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b)$$

Here $b \in \mathbb{R}$ is a bias term and f is a non-linear function. This filter is applied to each possible window of words from $\mathbf{v}_1$ to $\mathbf{v}_n$ to produce *feature* $\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}]$ with $\mathbf{c} \in \mathbb{R}^s (s = n - h + 1)$.

2.3 Residual Convolution Block

Residual learning connects low-level to high-level representations directly, and tackles the vanishing gradient problem in deep networks. In our model, we design the residual convolution block by applying shortcut connections. Each residual convolutional block is a sequence of two convolutional layers, each one followed by an ReLU activation. The kernel size of all convolutions is h, with padding such that the new feature will have the same size as the original one. Here are two convolutional *filter* $\mathbf{w}_1$, $\mathbf{w}_2 \in \mathbb{R}^{h \times 1}$. For the first convolutional layer:

$$\tilde{c}_i = f(\mathbf{w}_1 \cdot c_{i:i+h-1} + b_1)$$

For the second convolutional layer:

$$\acute{c}_i = f(\mathbf{w}_2 \cdot \tilde{c}_{i:i+h-1} + b_2)$$

Here b_1, b_2 are bias terms. For the residual learning operation:

$$\mathbf{c} = \mathbf{c} + \acute{\mathbf{c}}$$

Conveniently, the notation of $\mathbf{c}$ on the left is changed to define as the output vectors of the block. This operation is performed by a shortcut connection and element-wise addition. This block will be multiply concatenated in our architecture.

2.4 Max Pooling, Softmax Output

We then apply a max-pooling operation over the *feature* and take the maximum value $\hat{c} = max\{\mathbf{c}\}$.

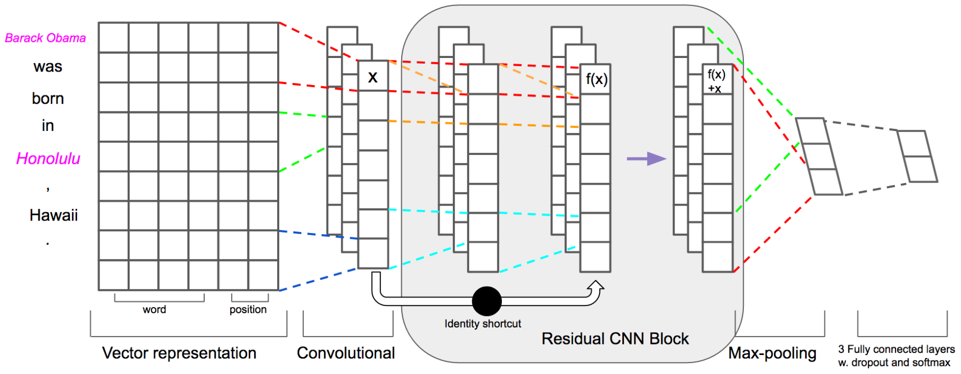


Figure 1: The architecture of ResCNN used for relation extraction.

We have described the process by which *one* feature is extracted from *one* filter. Take all features into one high level extracted feature $\mathbf{z} = [\hat{c}_1, \hat{c}_2, ..., \hat{c}_m]$(note that here we have m filters). Then, these features are passed to a fully connected softmax layer whose output is the probability distribution over relations. Instead of using $y = \mathbf{w} \cdot \mathbf{z} + b$ for output unit y in forward propagation, dropout uses $y = \mathbf{w} \cdot (\mathbf{z} \circ \mathbf{r}) + b$ where $\circ$ is the element-wise multiplication operation and $\mathbf{r} \in \mathbb{R}^m$ is a 'masking' vector of Bernoulli random variables with probability p of being 1. In the test procedure, the learned weight vectors are scaled by p such that $\hat{\mathbf{w}} = p\mathbf{w}$ and used (without dropout) to score unseen instances.

3 Experiments

3.1 Experimental Settings

In this paper, we use the word embeddings released by (Lin et al., 2016) which are trained on the NYT-Freebase corpus (Riedel et al., 2010). We fine tune our model using validation on the training data. The word embedding is of size 50. The input text is padded to a fixed size of 100. Training is performed with tensorflow adam optimizer, using a mini-batch of size 64, an initial learning rate of 0.001. We initialize our convolutional layers following (Glorot and Bengio, 2010). The implementation is done using Tensorflow 0.11. All experiments are performed on a single NVidia Titan X (Pascal) GPU. In Table 1 we show all parameters used in the experiments.
We experiment with several state-of-the-art baselines and variants of our model.

- **CNN-B**: Our implementation of the CNN baseline (Zeng et al., 2014) which contains one convolutional layer, and one fully connected layer.

Window size h	3
Word dimension d_w	50
Position dimension d_p	5
Position maximum distance e_{max}	30
Position minimum distance e_{min}	-30
Number of filters m	128
Batch size B	64
Learning rate λ	0.001
Dropout probability p	0.5

Table 1: Parameter settings

- **CNN+ATT**: CNN-B with attention over instance learning (Lin et al., 2016).
- **PCNN+ATT**: Piecewise CNN-B with attention over instance learning (Lin et al., 2016).
- **CNN**: Our CNN model which includes one convolutional layer and three fully connected layers.
- **CNN-x**: Deeper CNN model which has x convolutional layers. For example, CNN-9 is a model constructed with 9 convolutional layers (1 + 4 residual cnn block without identity shortcut) and three fully connected layers.
- **ResCNN-x**: Our proposed CNN-x model with residual identity shortcuts.

We evaluate our models on the widely used NYT freebase larger dataset (Riedel et al., 2010). Note that ImageNet dataset used by the original ResNet paper (He et al., 2016) has 1.28 million training instances. NYT freebase dataset includes 522K training sentences, which is the largest dataset in relation extraction, and it is the only suitable dataset to train deeper CNNs.

3.2 NYT-Freebase Dataset Performance

The advantage of this dataset is that there are 522,611 sentences in training data and 172,448 sentences in testing data and this size can support

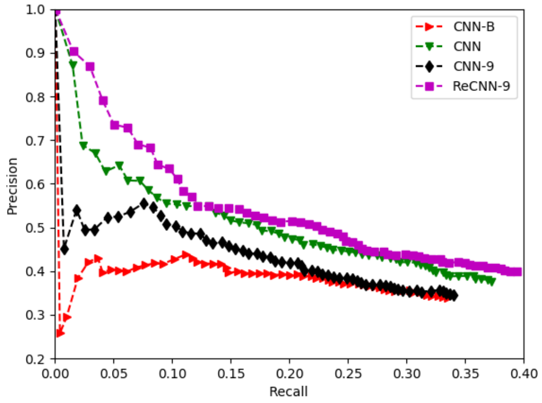


Figure 2: Comparing ResCNN to different CNNs.

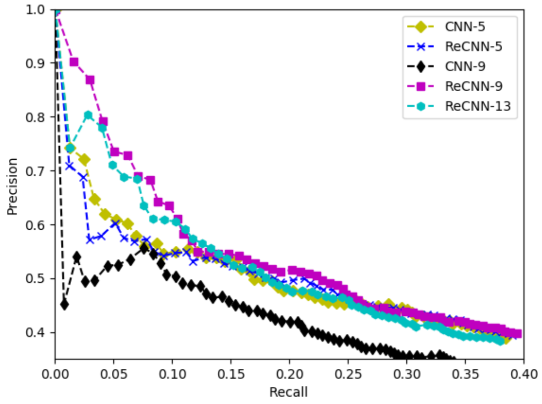


Figure 3: Varying the depths of our models.

us to train a deep network. Similar to previous work (Zeng et al., 2015; Lin et al., 2016), we evaluate our model using the held-out evaluation. We report both the aggregate curves precision/recall curves and Precision@N (P@N).

In Figure 2, we compare the proposed ResCNN model with various CNNs. First, CNNs with multiple fully-connected layers obtained very good results, which is a novel finding. Second, the results also suggest that deeper CNNs with residual learning help extracting signals from noisy distant supervision data. We observe that overfitting happened when we try to add more layers and the performance of CNN-9 is much worse than CNN. We find that ResNet can solve this problem and ResCNN-9 obtains better performance as compared to CNN-B and CNN and dominates the precision/recall curve overall.

We show the effect of depth in residual networks in Figure 3. We observe that ResCNN-5 is worse than CNN-5 because the ResNet does not work well for shallow CNNs, and this is consistent with the original ResNet paper. As we increase the network depth, we see that CNN-9 does overfit to the training data. With residual learning, both ResCNN-9 and ResCNN-13 provide significant improvements over CNN-5 and ResCNN-5 models. In contradictory to popular beliefs that ResNet only works well for very deep networks, we found that even with 9 layers of CNNs, using identity mapping could significantly improve the performance learning in a noisy input setting.

P@N(%)	100	200	300	Mean
CNN+ATT	76.2	68.6	59.8	68.2
PCNN+ATT	**76.2**	**73.1**	**67.4**	**72.2**
CNN-B	41.0	40.0	41.0	40.7
CNN	64.0	61.0	55.3	60.1
CNN-5	64.0	58.5	54.3	58.9
ResCNN-5	57.0	57.0	54.3	56.1
CNN-9	56.0	54.0	49.7	53.2
ResCNN-9	**79.0**	**69.0**	**61.0**	**69.7**
ResCNN-13	76.0	65.0	60.3	67.1

Table 2: P@N for relation extraction with different models. Top: models that select training data. Bottom: models without selective attention.

The intuition of ResNet help this task in two aspect. First, if the lower, middle, and higher levels learn hidden lexical, syntactic, and semantic representations respectively, sometimes it helps to bypass the syntax to connect lexical and semantic space directly. Second, ResNet tackles the vanishing gradient problem which will decrease the effect of noise in distant supervision data.

In Table 2, we compare the performance of our models to state-of-the-art baselines. We show that our ResCNN-9 outperforms all models that do not select training instances. And even without piecewise max-pooling and instance-based attention, our model is on par with the PCNN+ATT model.

For the more practical evaluation, we compare the results for precision@N where N is small (1, 5, 10, 20, 50) in Table 3. We observe that our ResCNN-9 model dominates the performance when we predict the relation in the range of higher probability. ResNet helps CNNs to focus on the highly possible candidate and mitigate the noise effect of distant supervision. We believe that residual connections actually can be seen as a form of renormalizing the gradients, which prevents the model from overfitting to the noisy distant supervision data.

In our distant-supervised relation extraction experience, we have two important observations: (1) We get significant improvements with CNNs adding multiple fully-connected layers. (2) Resid-

P@N(%)	1	5	10	20	50
PCNN+ATT	**1**	0.8	**0.9**	0.75	0.7
ResCNN-9	**1**	**1**	**0.9**	**0.9**	**0.88**

Table 3: P@N for relation extraction with different models where N is small. We get the result of PCNN+ATT using their public source code.

ual learning could significantly improve the performance for deeper CNNs.

4 Conclusion

In this paper, we introduce a deep residual learning method for distantly-supervised relation extraction. We show that deeper convolutional models help distill signals from noisy inputs. With shortcut connections and identify mapping, the performances are significantly improved. These results aligned with a recent study (Conneau et al., 2017), suggesting that deeper CNNs do have positive effects on noisy NLP problems.

References

Razvan Bunescu and Raymond J Mooney. 2005. Subsequence kernels for relation extraction. In *NIPS*, pages 171–178.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for natural language processing. *EACL 2017*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 427–434. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

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

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Yatian Shen and Xuanjing Huang. 2016. Attention-based convolutional neural network for semantic relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal*, pages 17–21.

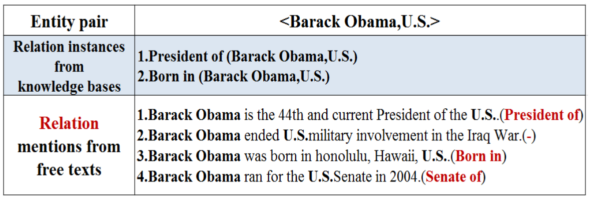
Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

Noise-Clustered Distant Supervision for Relation Extraction: A Nonparametric Bayesian Perspective

Qing Zhang and **Houfeng Wang**
Key Laboratory of Computational Linguistics (Peking University) Ministry of Education, China
{zqicl,wanghf}@pku.edu.cn

Abstract

For the task of relation extraction, distant supervision is an efficient approach to generate labeled data by aligning knowledge base with free texts. The essence of it is a challenging incomplete multi-label classification problem with sparse and noisy features. To address the challenge, this work presents a novel nonparametric Bayesian formulation for the task. Experiment results show substantially higher top-precision improvements over the traditional state-of-the-art approaches.



Entity pair	<Barack Obama,U.S.>
Relation instances from knowledge bases	1.President of (Barack Obama,U.S.) 2.Born in (Barack Obama,U.S.)
Relation mentions from free texts	1.**Barack Obama** is the 44th and current President of the **U.S.**.(**President of**) 2.**Barack Obama** ended **U.S.**military involvement in the Iraq War.(-) 3.**Barack Obama** was born in honolulu, Hawaii, **U.S.**.(**Born in**) 4.**Barack Obama** ran for the **U.S.**Senate in 2004.(**Senate of**)

Figure 1: Aligned Example (Fan et al., 2014): the relation instances related to the entity pair $\langle BarackObama, U.S. \rangle$ in the KB, and its mentions in the free text.

1 Introduction

To efficiently generate structured relation information from free texts, the research on distantly supervised Relation Extraction (RE) (Mintz et al., 2009; Riedel et al., 2013; Hoffmann et al., 2011) has been attracting much attention, because it can greatly reduce the manual annotation for training. It essentially based on the assumption that the relation between two entities in a Knowledge Base (KB), is also likely hold within a sentence that mentions the two entities in free texts. This assumption plays a crucial role in distant supervision, which is quite effective in real applications.

However, the assumption of distant alignment can also lead to the noisy training corpus problem (Fan et al., 2014), which is challenging for the task as follows: **i) Noisy features.** Not all relations existed in a KB keep the same meaning of that relation for the corresponding entities in a free text. For example, the second relation mention in Figure 1 does not explicitly describe any relation instance, so features extracted from this sentence can be noisy. Such analogous cases commonly exist in feature extraction. **ii) Incomplete labels.** Similar to noisy features, the generated label can be incomplete due to the incomplete knowledge base (Ritter et al., 2013). For example, the fourth relation mention in Figure 1 should be labeled by the relation Senate-of. However, the corresponding relation instance (Senate-of(Barack Obama, U.S.)) is missing in the knowledge base. Such analogous cases are also common in real applications. **iii) Sparse features.** Sophisticated features extracted from the mentions can result in a large number of sparse features (Riedel et al., 2013). The generalization ability of feature based prediction models will be badly hurt, when the features do not match between testing and training.

To tackle the problem, we develop a novel distant supervision approach from a nonparametric Bayesian perspective (Blei et al., 2016), along with the previously most effective research line (Petroni et al., 2015) of using matrix completion (Fan et al., 2014) for relation extraction. Our goal is to design a noise-tolerant relation extraction model for distantly supervised corpus with noise and sparsity problems. Different from (Fan et al., 2014) as one state-of-the-art method in this line, we model noisy data corpus using adaptive variance modeling approach (Chen et al., 2015), based on *Dirichlet Process* (Blei and Jordan, 2004) instead of a fixed way of controlling complex noise weighting. To the best of our knowledge, we are

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1808–1813
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

the first to apply this technique on relation extraction with distant supervision.

2 Approach

The essence of the task is a multi-label classification problem (Cabral et al., 2011) with noisy patterns (Han and Sun, 2014). One simple way, to solve the problem, is to learn separate classifiers for each of relation labels, using n samples with d features, by optimizing $b \in R^{1\times 1}$ and $\mathbf{w} \in R^{d\times 1}$,

$$\text{argmin}_{b,\mathbf{w}}\ l(\mathbf{y_{train}}, [\mathbf{1}\ X_{train}] \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix}), \tag{1}$$

where $\mathbf{1}$ is the all-one column vector; $X_{train} \in R^{n\times d}$ and $\mathbf{y_{train}} \in R^{n\times 1}$ are the corresponding feature matrix and label vector respectively. However, label correlations are not considered in the above formulation. To jointly consider feature correlations and label correlations, (Cabral et al., 2011) formulated the multi-label classification as a matrix completion problem. As a powerful framework, it has been successfully applied to relation extraction task with distant supervision.

2.1 Previous Formulation

The work in (Fan et al., 2014) first adopted the mentioned framework, as a general joint learning and inference framework (Cabral et al., 2011), to learn *noise-tolerant* distant supervision for relation extraction. It achieves the state-of-the-art performance. Suppose we have a training corpus, including n instances (entity pairs) including both training and test data, with d-dimensional features and t relation labels, which is built according to the basic alignment assumption. The task can be modeled with a sparse matrix $Z \in R^{nx(d+t)}$, defined as

$$Z = \begin{bmatrix} X_{train} & Y_{train} \\ X_{test} & Y_{test} \end{bmatrix}, \tag{2}$$

where each row in Z represents entity pair, and each column represents noisy textual feature in X or incomplete relation label in Y. In such a way, relation extraction is transformed into a problem of completing the unknown labels in Y_{test} for the test data X_{test} in Z. The rational of this modeling is that noisy features and incomplete labels are semantically correlated, which can be explained in an underlying low-rank structure (Riedel et al., 2013). Taking noise into consideration, Z is further defined as

$$Z = Z^* + E, \tag{3}$$

where Z^* is the underlying low-rank matrix

$$Z^* = \begin{bmatrix} X^*_{train} & Y^*_{train} \\ X^*_{test} & Y^*_{test} \end{bmatrix}, \tag{4}$$

and E is the error (noise) matrix

$$E = \begin{bmatrix} E_{X_{train}} & E_{Y_{train}} \\ E_{X_{test}} & 0 \end{bmatrix}. \tag{5}$$

This error (noise) modeling approach has been successfully applied to distantly supervised relation extraction. However, it still has clear limitations. The noise model is limited to a single source without considering the intrinsic clustering structures of data. In addition, the true rank is usually hard to determine, for adaptively modeling the correlations among features and labels.

2.2 Nonparametric Bayesian Modeling

The use of nonparametric Bayesian modeling has been widely adopted in Natural Language Processing (NLP) (Chen et al., 2014). Instead of imposing assumptions that might be wrong, it "lets the data speak for itself", without requiring optimizing parameters blindly by hands (Blei and Jordan, 2004). To take advantage of these merits, we here adopt it for the task, with the following motivations:

Motivation 1: Adaptive Noise-Clustered Attention. The goal is to find an adaptive *cluster specific* noise *parameterization* for the complex noisy corpus, *without* making overly strong assumptions about the noise distribution in real applications.

Motivation 2: Adaptive Latent Feature Space Selection. The goal is to automatically find better *dense representations* of latent entity-pair, feature and label *without* pre-specifying the rank values by laboriously retraining models.

2.2.1 Nonparametric Bayesian Formulation

We develop a novel formulation for distantly supervised relation extraction, using a nonparametric Bayesian approach, based on the Dirichlet Process, which can been seen as an infinite Dirichlet distribution, with clustering effect for modeling categorical variables adaptively.

Noise component modeling. Instead of using a single fixed noise model, we redefine $E = [\varepsilon_{i,j}] \in R^{n\times(d+t)}$ in Eq.(5). $\varepsilon_{i,j}$ is modeled by a summation of infinite noise models (Chen et al., 2015),

$$p(\varepsilon_{i,j}) = \sum_{k=1}^{\infty} \theta_k N(\varepsilon_{i,j}|0, \sigma_k), \tag{6}$$

where θ_k is the mixing proportion for the k-th gaussian component $N(\varepsilon_{i,j}|0, \sigma_k)$ with mean zero and variance σ_k. The θ is obtained from the stick-breaking process (Blei and Jordan, 2004), with $\sum_{k=1}^{\infty} \theta_k = 1$,

$$\theta_k = \beta_k \prod_{l=1}^{k-1} (1 - \beta_l), \tag{7}$$

where β_k are independent draws from the beta distribution $\beta(1, \alpha)$. As a result, the noise entries will cluster themselves into K groups without requiring a complicated model selection procedure. Since a mixture of Gaussians can approximate any continuous probability distribution (Zhao et al., 2014), this structural noise formulation can adapt much wider range of real noises than previous formulation (Fan et al., 2014) for relation extraction.

Low-rank component modeling. Different from (Fan et al., 2014), instead of directly minimizing the rank of Z^* in Eq.(3), we decompose Z^* into two low-rank matrices U and V, from probabilistic perspective (Salakhutdinov and Mnih, 2007). This modeling approach can lead to a more flexible way of estimating the optimal rank values for latent feature spaces. To determine the appropriate rank automatically, we adopt the *Automatic Relevance Determination* (ARD) method (Babacan et al., 2012) by imposing a prior on each dimmension (column) of U and V. Specifically, we impose the Gaussian priors with variance λ_r on the r-th columns of U and V, i.e., $u_{.r}$ and $v_{.r}$:

$$\begin{aligned} p(\mathbf{U}|\lambda) &= \prod_{r=1}^{R} N(u_{.r}|0, \lambda_r \mathbf{I_U}), \\ p(\mathbf{V}|\lambda) &= \prod_{r=1}^{R} N(v_{.r}|0, \lambda_r \mathbf{I_V}), \\ \lambda_r &\sim IG(a_1, b_1), \end{aligned} \tag{8}$$

where IG is an Inverse Gamma distribution for modeling the variance λ_r. Considering a column as latent factor in U or V with a zero mean in the prior, a very small variance indicates that this column will shrink to zero. Thus, the irrelevant columns hurting the performance will be eliminated adaptively, without pre-specifying the rank values by retraining models laboriously as in the previous modeling (Fan et al., 2014) for the task.

Prediction component modeling. We can leverage the above presented low-rank component for U, V and noise component for $\varepsilon_{i,j}$, to build Eq.(9) for prediction. Different from the state-of-the-art multi-label classification framework as adopted in (Fan et al., 2014), for simplicity, we design noise model for features and labels jointly,

$$p(y_{i,j}) = N(y_{i,j}| \underbrace{u_{i.} v_{j.}^T}_{low-rank\ component}, \underbrace{\varepsilon_{i,j}}_{noise\ component}), \tag{9}$$

where $u_{i.}$ and $v_{j.}$ are defined in Eq.(8) as rows of U and V respectively.

For each interaction between entity-pair and feature (or relation), $\varepsilon_{i,j}$ as defined in Eq.(6) can be injected into Eq.(9) (Chen et al., 2015) by

$$\begin{aligned} \varepsilon_{i,j} &= \sigma_{z_{ij}}, \\ \sigma_{z_{ij}} &\sim IG(a_0, b_0), \\ z_{ij} &= k \sim Mult(\theta_k), \end{aligned} \tag{10}$$

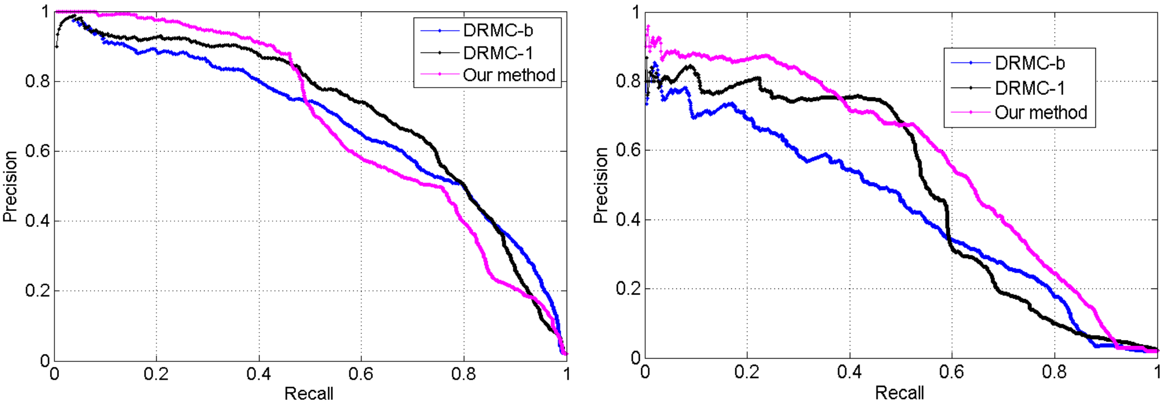
where θ_k is modeled in Eq.(7); Mult is a Multinomial distribution.

The mechanism of the introduced clustered noise component for relation extraction can be easily understood through considering its role in the Gaussian distribution. As shown in Eq.(9), $\varepsilon_{i,j}$ is used to control the variance. Large variance value means low confidence, and the small value means high confidence, for fitting $y_{i,j}$ with $u_{i.} v_{j.}^T$. The variance parameter $\varepsilon_{i,j}$, generated by noise component Eq.(7,10), serves as a confidence parameter for training instance. In the algebra view of likelihood, variance parameter is just the weight of training instance (i.e., the interaction between "entity pair and feature" or "entity pair and label"), measuring the importance for its contribution to the total likelihood. We can treat this mechanism as an importance weighting mechanism, for selecting noisy interactions y_{ij} with different clustering structures adaptively.

In this mechanism, for each $y_{i,j}$ in noisy corpus, it allows $1 \rightarrow 0$ (noisy feature) for features, and allow $1 \rightarrow 0$ (label with no supportive features) or $0 \rightarrow 1$ (incomplete label) for labels. In addition, for the task, we expect that our method can automatically adjust the importance weight for reducing the effect of common features, to differentiate two instances with different labels. To achieve the goal, in matrix Z, we fit both "1" (observed) and "0" for training labels as discriminative supervision, while we only fit "1" (observed) for features.

Dataset	#training	#testing	%more than one label	#features	#relation labels
NYT'10	4,700	1,950	7.5%	244,903	51
NYT'13	8,077	3,716	0%	1,957	51

Table 1: Statistics about the two widely used datasets.



(a) NYT'10 dataset

(b) NYT'13 dataset

Figure 2: Precision-Recall curve on NYT'10 and NYT'13 datasets. DRMC-b(1) (Fan et al., 2014).

Models	P	R	F1
Mintz	63.59%	61.20%	62.37%
Hoffmann	67.18%	36.41%	47.23%
Surdeanu	76.23%	53.18%	62.65%
DRMC-b	61.03%	66.82%	63.79%
DRMC-1	64.17%	**71.74%**	**67.75%**
Our	**87.94%**	46.00%	63.44%

Table 2: Results at the highest F1 point in the Precision-Recall (P-R) curve on NYT'10 dataset. Mintz (Mintz et al., 2009); Hoffmann (Hoffmann et al., 2011); Surdeanu (Surdeanu et al., 2012); DRMC-b(1) (Fan et al., 2014);

Learning. To combine Eqs. (7)-(10), we can construct the full Bayesian model. The goal turns to infer the posterior of all involved variables:

$$p(\mathbf{U}, \mathbf{V}, \lambda, \sigma, \mathbf{z}, \beta | \mathbf{X}_{observed}, \mathbf{Y}_{observed}), \quad (11)$$

where $\mathbf{X}_{observed}, \mathbf{Y}_{observed}$ are the observed binary features (fitting 1) and labels (fitting both 1 and 0). Variational inference is adopted as shown in (Chen et al., 2015).

Prediction. After learning[1], we use the expectation $E(P(y_{i,j}))$ in Eq.(9) to complete the entries in Y_{test}. Finally, we can acquire Top-N predicted relations via ranking the values $E(P(y_{i,j}))$, given entity pair i, for different relations j.

3 Experiments

We evaluate our method on two widely used datasets as shown in Table 1 with the same setting in (Fan et al., 2014).

Dataset. *NYT'10*, was developed by (Riedel et al., 2010). *NYT'13*, was also released by (Riedel et al., 2013), in which they only regarded the lexicalized dependency path between two entities as features. Both are automatically generated by aligning Freebase to New York Times corpus.

Parameter setting. For all the conducted experiments, the model hyperparameters are fixed without further tuning: $a_0 = b_0 = 10^{-4}$, $a_1 = b_1 = 0.1$ and $\alpha = 1$.

Model comparison. Since (Fan et al., 2014) achieves the state-of-the-art performance on the two datasets, we mainly compare our method with that in the same setting, to verify the effectiveness. *NYT'10 dataset*: Table 2 indicates that our model achieves the highest precision performance among all of the competitors. Although the recall performance is not competitive, the F1 score is also comparable to DRMC-b. Figure 2(a) further shows the strong precision performance when the recall is not large. *NYT'13 dataset*: Figure 2(b) illustrates that our approach outperforms the state-of-the-art methods, which shows that our approach can maintain a fairly high precision even when recall is larger. In addition, in practical applications, we also concern about the precision on

[1] We implement the system for relation extraction, based on the code at http://peixianc.me/amf_codes.zip.

Top-N	**NFE-13**	**DRMC-b**	**DRMC-1**	**Our**
Top-100	62.9%	82.0%	80.0%	**92.0%**
Top-200	57.1%	77.0%	80.0%	**88.2%**
Top-500	37.2%	70.2%	77.0%	**86.3%**
Average	52.4%	76.4%	79.0%	**88.8%**

Table 3: Precision of Top-N predicted instances on NYT'13 dataset. NFE-13 (Riedel et al., 2013); DRMC-b(1) (Fan et al., 2014).

Models	**P**	**R**	**F1**
DRMC-b	47.70%	49.58%	48.62%
DRMC-1	**67.99%**	50.42%	57.90%
Our	66.46%	**53.30%**	**59.16%**

Table 4: Results at the highest F1 point in the Precision-Recall (P-R) curve on NYT'13 dataset. DRMC-b(1) (Fan et al., 2014).

Top-N predicted instances. Table 3 shows that our model achieves much significant improvements on that. Moreover, Table 4 shows that our method can achieve the best F1, compared with the baselines.

NYT'10 and NYT'13 have different performance records, which could be explained as follows. From the dataset perspective, NYT'10 is a dataset with multi-label instances, which is more complex than NYT'13 only having single label instances. This is one reason of why the trends are quite different between them. More essentially, we further discuss the differences from the model mechanism perspective, to explain the reasons. In (Fan et al., 2014)'s work, it has no explicit noise modeling mechanism. The noise is modeled implicitly as the error of cost functions. From the probabilistic view, that error is sampled from single Gaussian with zero mean and fixed variance. In contrast, our method uses infinite Gaussian with automatically learnt variance. It may cause overfitting for complex dataset with sparse features. In addition, we guess the reason is that in (Fan et al., 2014)'s work, they use two separate cost functions for features and labels, while in our work we use one unified noise component for both of them, which shows the promising precision performance in NYT'10 when recall is less than 0.4.

In addition, in our experiments, we found that early stopping is crucial for achieving good results while model learning. This also verifies that the potential overfitting problem should be further considered while using the more flexible nonparametric method for NLP task.

4 Related Work

Our work is closest to (Fan et al., 2014), since we focus on the same noisy corpus problem. Although from different perspectives, we study it along with the same line of using matrix factorization (Petroni et al., 2015) for relation extraction. In this line, (Riedel et al., 2013) initially considered the task as a matrix factorization problem. Their method consists of several models, such as PCA (Collins et al., 2001) and collaborative filtering (Koren, 2008). However, the data noise brought by the assumption of distant supervision (Mintz et al., 2009), is not considered in the work. Another line addressing the problem uses deep neural networks (Zeng et al., 2015; Wang et al., 2015). The difference is that it is a supervised learning approach, while our focused one is a joint learning approach with transductive style, in which both training and test data are exploited simultaneously. In addition, (Han and Sun, 2016) explored Markov logic technique to enrich supervision knowledge, which can incorporate indirect supervision globally. Our method could be further augmented by that idea, using additional logical constraint to reduce the uncertainty for the clustered noise modeling.

5 Conclusion

In this paper, building on recent advances from the nonparametric Bayesian literature, we reformulate the task of relation extraction with distant supervision, based on the adaptive variance learning with intrinsic clustering structures. For the task, it can solve the sparsity problem via the learnt low-rank dense representations and can allow fitting noisy corpus through adaptive variance adjustment. Meanwhile, it can avoid turning a large number of parameters. Experiments suggest substantially higher top-precision than the competitors. In the future work, we plan to develop more sophisticated noise models for features and labels separately, and try to explore logical information, particularly in this context of nonparametric noise modeling, for further benefiting this task.

Acknowledgments

Our work is supported by National Natural Science Foundation of China (No.61370117 & No.61433015). The corresponding author of this paper is Houfeng Wang.

References

S. Derin Babacan, Martin Luessi, Rafael Molina, and Aggelos K. Katsaggelos. 2012. Sparse bayesian methods for low-rank matrix estimation. *IEEE Transactions on Signal Processing*, 60(8):3964–3977.

David M. Blei and Michael I. Jordan. 2004. Variational methods for the dirichlet process. In *Proceedings of the International Conference on Machine Learning, Banff, Alberta, Canada*.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. 2016. Variational inference: A review for statisticians. *CoRR*, abs/1601.00670.

Ricardo Silveira Cabral, Fernando De la Torre, João Paulo Costeira, and Alexandre Bernardino. 2011. Matrix completion for multi-label image classification. In *Proceedings of Advances in Neural Information Processing Systems, Granada, Spain.*, pages 190–198.

Miaohong Chen, Baobao Chang, and Wenzhe Pei. 2014. A joint model for unsupervised chinese word segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar*, pages 854–863.

Peixian Chen, Naiyan Wang, Nevin L. Zhang, and Dit-Yan Yeung. 2015. Bayesian adaptive matrix factorization with automatic model selection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, MA, USA*, pages 1284–1292.

Michael Collins, Sanjoy Dasgupta, and Robert E. Schapire. 2001. A generalization of principal components analysis to the exponential family. In *Proceedings of Advances in Neural Information Processing Systems, British Columbia, Canada*, pages 617–624.

Miao Fan, Deli Zhao, Qiang Zhou, Zhiyuan Liu, Thomas Fang Zheng, and Edward Y. Chang. 2014. Distant supervision for relation extraction with matrix completion. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA*, pages 839–849.

Xianpei Han and Le Sun. 2014. Semantic consistency: A local subspace based method for distant supervised relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA*, pages 718–724.

Xianpei Han and Le Sun. 2016. Global distant supervision for relation extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, Arizona, USA.*, pages 2950–2956.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of Annual Meeting of the Association for Computational Linguistics , Oregon, USA*, pages 541–550.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA*, pages 426–434.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics, Singapore*, pages 1003–1011.

Fabio Petroni, Luciano Del Corro, and Rainer Gemulla. 2015. CORE: context-aware open relation extraction with factorization machines. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal*, pages 1763–1773.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of Machine Learning and Knowledge Discovery in Databases, European Conference, Barcelona, Spain*, pages 148–163.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of Conference of the North American Chapter of the Association of Computational Linguistics, Atlanta, Georgia, USA*, pages 74–84.

Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. 2013. Modeling missing data in distant supervision for information extraction. *TACL*, 1:367–378.

Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic matrix factorization. In *Proceedings of Advances in Neural Information Processing Systems, British Columbia, Canada*, pages 1257–1264.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, Jeju Island, Korea*, pages 455–465.

Zhen Wang, Baobao Chang, and Zhifang Sui. 2015. Distantly supervised neural network model for relation extraction. In *Proceedings of Chinese Computational Linguistics and Natural Language Processing, Guangzhou, China*, pages 253–266.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal*, pages 1753–1762.

Qian Zhao, Deyu Meng, Zongben Xu, Wangmeng Zuo, and Lei Zhang. 2014. Robust principal component analysis with complex noise. In *Proceedings of the International Conference on Machine Learning, Beijing, China*, pages 55–63.

Exploring Vector Spaces for Semantic Relations

Kata Gábor[1], Haïfa Zargayouna[1], Isabelle Tellier[2], Davide Buscaldi[1], Thierry Charnois[1]
[1] LIPN, CNRS (UMR 7030), Université Paris 13
[2] LaTTiCe, CNRS (UMR 8094), ENS Paris, Université Sorbonne Nouvelle,
PSL Research University, SPC

Abstract

Word embeddings are used with success for a variety of tasks involving lexical semantic similarities between individual words. Using unsupervised methods and just cosine similarity, encouraging results were obtained for analogical similarities. In this paper, we explore the potential of pre-trained word embeddings to identify generic types of semantic relations in an unsupervised experiment. We propose a new relational similarity measure based on the combination of word2vec's CBOW input and output vectors which outperforms alternative vector representations, when used for unsupervised clustering on SemEval 2010 Relation Classification data.

1 Introduction

Vector space word representations or *word embeddings*, both 'count' models (Turney and Pantel, 2010) and learned vectors (Mikolov et al., 2013a; Pennington et al., 2014), were proven useful for a variety of semantic tasks (Mikolov et al., 2013b; Baroni et al., 2014). Word vectors are used with success because they capture a notion of semantics directly extracted from corpora. Distributional representations allow to compute a functional or topical semantic similarity between two words or, more recently, bigger text units (Le and Mikolov, 2014). The more similar two entities are semantically, the closer they are in the vector space (quantified usually, but not necessarily in terms of cosine similarity). Semantic similarity can be exploited for lexical substitution, synonym detection, subcategorization learning etc. Recent studies suggest that neural word embeddings show higher performance than count models (Baroni et al., 2014; Krebs and Paperno, 2016) for most semantic tasks, although Levy et al. (2015a) argue that this is only due to some specific hyperparameters that can be adapted to count vectors. In what follows, we will concentrate on exploring whether and how pre-trained, general-purpose word embeddings encode relational similarities.

1.1 Relational analogies as vector offsets

Relation extraction and classification deal with identifying the semantic relation linking two entities or concepts based on different kinds of information, such as their respective contexts, their co-occurrences in a corpus and their position in an ontology or other kind of semantic hierarchy. Whether the vector spaces of pre-trained word embeddings are appropriate for discovering or identifying *relational* similarities remains to be seen. Mikolov et al. (2013b) claimed that the embeddings created by a recursive neural network indeed encode a specific kind of relational similarities, i.e. *analogies* between pairs of words. He found that by using simple vector arithmetic, analogy questions in the form of "a_1 is to a_2 as b_1 is to b_2" (*man ~king :: woman ~queen*) could be solved. Relationships are assumed to be present as vector offsets, so that in the embedding space, all pairs of words sharing a particular relation are related by the same constant offset. Vector arithmetics give us the vector which fills the analogy, and we can search for the word b_2 whose embedding vector has the greatest similarity to it:

$$argmax_{b_2} = sim(b_2, (b_1 - a_1 + a_2)) \qquad (1)$$

Levy et al. (2015a) suggested that instead of a vector offset method, this calculation can also be considered as a combination of similarities. Using cosine similarity for sim, equation 1 can be written as a combination of similarities (Levy et al., 2015a)

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1814–1823
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

as

$$argmax_{b_2} = sim(b_2, b_1) - sim(b_2, a_1) + \\ + sim(b_2, a_2) \quad (2)$$

Analogy pairs, however, are a special case of relational similarity because not only a_1 (*man*) relates to a_2 (*king*) the same way that b_1 (*woman*) relates to b_2 (*queen*); the relation between a_1 (*man*) and b_1 (*woman*) is also parallel to the relation between a_2 (*king*) and b_2 (*queen*.) This is not always the case: when it comes to different types of semantic relations, their instances may or may not be analogical.

1.2 Criticism of the vector offset method

As precise as neural word embeddings combined with cosine similarity may be for calculating semantic proximity between individual words, recent results seem to suggest that their value in identifying relational analogies using vector arithmetics is limited. In fact, a big part of their merits is likely to come from the precise calculation of individual similarities instead of relational similarities. Hence, they can be approximated using relation-independent baselines. Linzen (2016) remarks that currently used analogy tasks evaluate not only the consistency of the offsets $a_1 - a_2$ and $b_1 - b_2$, but also the neighborhood structure of the words in the vector space. Concretely, "if a_1 and a_2 are very similar to each other (...) the nearest word to b_2 may simply be the nearest neighbor of b_1 (...) regardless of offset consistency" (Linzen, 2016). Moreover, some of the success obtained by the vector offset method on analogies can also be obtained by baselines that ignore a_2, or even both a_1 and a_2.
Levy et al. (2015b) point out similar limitations: word embedding combinations in supervised learning of taxonomical relations do not seem to learn the relations themselves, but individual properties of words. They tested previously suggested vector compositions for supervised learning of inference relations: concatenation, difference, comparing only the first or only the second element of the pairs. The study concludes that the classifiers only learn individual properties (e.g. a "category" type word is a good hypernym candidate), but not semantic relations between words. Altogether, these studies suggest that the semantic information obtained from word embeddings is correct for identifying similar or related units, but is already self-contained and difficult to enrich in order to retrieve more specific semantic contents such as relational similarities or specific relations.

In this paper, we aim to challenge this conclusion within a large scale semantic relation classification experiment, and show that it is possible to achieve improvements compared to baselines and current methods. We apply known vector composition methods, and propose a new one, to unsupervised large-scale clustering of entity pairs categorized according to their semantic relation. While large scale semantic relation classification is a very difficult task and the state of the art does not perform yet at human level, we expect that the experiment provides information to compare the potential of different vector/similarity combinations in a setting (i.e. clustering) that is more reliant on the global structure of the data instead of the close neighborhood structure of selected items.

2 Semantic Relations in Vector Spaces

2.1 Related work

Relation classification includes the task of finding the instances of the semantic relations, i.e. the entity tuples, and categorizing their relation according to an existing typology. In an unsupervised framework, relation types are inferred directly from the data. Supervised systems rely on a list of predefined relations and categorized examples, as described in the shared tasks of MUC, ACE or SemEval campaigns (Hobbs and Riloff, 2010; Jurgens et al., 2012; Hendrickx et al., 2010). Competing systems extract different kinds of features eventually combined with external knowledge sources, and build classifiers to categorize new relationship mentions (Zhou et al., 2005). A commonly used method, initiated by Turney (2005; 2006), is to represent entity pairs by a pair-pattern matrix and calculate similarities over the distribution of the pairs. Another way of constructing a distributional vector space to represent quantifiable context features for relation extraction is to combine the vectors of the two entities. Different combinations were proposed to represent compositional meaning (Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Baroni et al., 2012). Popular methods include addition (Mitchell and Lapata, 2010), concatenating the two vectors (Baroni et al., 2012) or taking their difference (Weeds et al., 2014; Roller et al., 2014). As of now, these vector combinations had two types of applications in semantic relation classification.

The first one aims to find specific types of semantic or functional analogies (Herdağdelen and Baroni, 2009; Makrai et al., 2013; Levy et al., 2015a). The second one tries to infer taxonomical relations, i.e. hypernymy, or lexical entailment, in supervised experiments (Weeds et al., 2014; Turney and Mohammad, 2014). For the hypernymy detection task, relation directionality can be captured by the inclusion of the hyponym's context in the broader term (Kotlerman et al., 2010), as well as by measuring the informativeness of their contexts (Santus et al., 2014).

A few experiments have been specifically targeted at combining different kinds of linguistic information for calculating relational similarities. Turney (2012) suggested a dual distributional feature space, composed of a *domain space* and a syntactic *function space*, for supervised classification. Herdağdelen and Baroni (2009) combine individual entity vectors with co-occurrence contexts in their vector space. These works either aim to identify very specific relation types (typically taxonomical relations) with a mixture of features and a supervised classifier, or target analogy pairs: a task in which, as we have seen, relation-unaware baselines approximate relation-aware representations. However, more recently, Shwartz et al. (2016) achieved promising results on the hypernymy detection task by combining dependency path-based context representations with distributional vectors; this finding can be relevant for a broader range of semantic relations as well.

2.2 Task definition

Whether we use the vector offset method or any pairwise similarity combination, finding the missing word in an analogy depends on two factors:

1. Vector quality (do semantically close elements have a higher cosine similarity?);
2. Density and structure of the vector space.

If we adapt 1) above to the more generic relational similarity task, the question can be formulated as follows:

3. How much information about the semantic relation is actually in the text and how fit is the vector combination method to encode this information?

In accordance with Linzen (2016) and Levy et al. (2015b), we also think that analogy test sets are not optimal to answer this question. It was already confirmed that word embeddings are precise in identifying closely related items, while it is an open question whether they are useful for inferring a global structure from potentially noisy data in a large scale experiment. We propose to study relational similarity using a more generic and large scale relation classification task (Hendrickx et al., 2010), and clustering pairs according to semantic relations, instead of finding the one missing word in an analogy. This way, we rely less on the neighborhood structure and more on actual "linguistic regularities".

We evaluate different vector combination methods, and propose a new one, for calculating relational similarities. The evaluation concentrates on the aspects above. We test whether cosine similarity over these vector spaces is adapted for discovering groups and classifying individual instances. We report clustering results and compare the vector combinations by their performance.

2.3 Motivation

The semantic relation classification task, supervised or not, is a difficult one with a strong upper bound: relations vary considerably with respect to the way they are defined and expressed in the text. Some relation types are more lexical by nature: relations such as *dog* is an *animal*; a *teacher* works at a *school*; a *car* is kept at a *parking lot*, can be identified out of context. On the other hand, many relations are contextual; they are time-anchored or tied to extra-linguistic, situational context. Contextual relations (e.g. "the *accident* was caused by the *woman*") tend to be expressed explicitly, but rarely, in a corpus. A similar distinction underlies the notion of *classical* vs *non-classical* lexical semantic relations coined by Morris and Hirst (2004); however, their distinction is made on the level of relation *types*, while different instances of the same relation type can also be different with respect to their lexical or contextual nature. Other types of relations are defined exclusively through examples or analogies (e.g. *badge* is to *policeman* as *crown* is to *king*).

Semantic relations can also be viewed as binary predicates, and such predicates have semantic constraints on their arguments, similarly to verb subcategorization. Indeed, the relations are often expressed by verbs, and we expect specific arguments of relations to belong to a specific semantic type, e.g. Message (*chapters* in this book investigate issues), Instrument (*telescope* assists the eye), Collection (essays collected in this *volume*), Container (image is hidden in a *carafe*). We expect vector space models (VSMs) to capture such semantic groups through pairwise similarity combinations.

Semantic relation types and instances differ with respect to the degree of semantic constraint on their arguments, the analogical nature of the relation, and the lexical/contextual aspect. We expect distributional VSMs to contribute to capture analogical similarities, as well as the semantic types of the arguments and lexical or prototypical relation instances. Since their ability to capture contextual relations is limited, they need to be complemented with e.g. pattern-based or dependency-based approaches when it comes to less typical examples.

In the scope of the current experiment, our primary goal is to argue that vector combinations may encode lexical relational similarities in themselves. If a representation is more capable than others to group together word pairs according to relational similarities, this potential can further be exploited in unsupervised as well as in supervised experiments.

The current task requires a change of perspective compared to the analogical task: when we look for missing elements in an analogy, we know the word exists and we presume to know where it will be in the vector space. In unsupervised clustering, our aim is to infer a global structure from the data.

2.4 Semantic relation data

The SemEval 2010 Task 8 data we used (Hendrickx et al., 2010) contains examples of relation instances for 9 relations with sufficiently broad coverage to be of general and practical interest (Table 1).

There is no overlap between classes, but there are two groups of strongly related relations to assess models' ability to make fine-grained distinctions (CONTENT-CONTAINER, COMPONENT-WHOLE, MEMBER-COLLECTION and ENTITY-ORIGIN, ENTITY-DESTINATION). Human agreement rates, when annotated in context, range from 58.2% to 98.5% depending on the relation type (Hendrickx et al., 2010). This data set is very challenging, not only because of the fine semantic distinctions, but also because semantic relations were annotated in context and contain many less typical relation instances. In the current experiment, the goal we set for ourselves is to explore models' abilities to capture the structure of the data, rather then in achieving a classification precision close to that of humans.

We used 6637 pairs of single word instances from the training data. Contexts in the training data were discarded. Class bias is present: the most frequent relation has 979 instances, the least frequent has 486.

3 Vector combination methods

If a_1, a_2, b_1, b_2 are entities (nouns or nominal compositions) from a corpus, each of them assigned a pre-trained word embedding, we would like to classify entity pairs $a = (a_1, a_2)$ and $b = (b_1, b_2)$ according to their semantic relation. This means that we are looking for an efficient combination of a_1, a_2 and b_1, b_2 vectors that encode their relational attributes. We aim to find effective methods to calculate a relational similarity $sim(a, b)$ by combining entity vectors a_1, a_2 and b_1, b_2.

Pairwise similarities build on the idea that if a_1 is semantically similar to b_1 and a_2 is similar to b_2, the relation between a_1 and a_2 is similar to the relation between b_1 and b_2. The recall of this approach is expected to be limited: the same relation can hold between different types of entities.

Analogical similarities presume that $b_1 - b_2$ shares the direction with $a_1 - a_2$, ignoring the pairwise similarities. We adapt this measure, while aware that analogy pairs are a specific case of relational similarity in that analogies work both ways (*man ~king* :: *woman ~queen* and also *man ~woman* :: *king ~queen*).

IN-OUT similarities: a new combination that builds on the integration of second order similarities.

Only a_1 : In this baseline solution, the similarity between two pairs is calculated as the similarity between the first entity of each pair, the other pair being ignored.

$$sim(a, b) = sim(a_1, b_1) \tag{3}$$

3.1 Pairwise similarities

Different combinations proposed in the literature were compared.

- **concatenative** : one vector for each entity pair is defined as the concatenation of the vec-

Relation	Instances in training data	Typical examples	Atypical examples
Cause-Effect	979	$suicide - death, injury - discomfort$	$women - accident$
Component-Whole	978	$claw - owl, walls - hospital$	$image - photos$
Entity-Destination	789	$solvent - flask, hay - barn$	$chair - corporation$
Product-Producer	775	$industry - models, artist - design$	$officer - oath$
Entity-Origin	762	$relics - culture, plane - runway$	$error - definition$
Member-Collection	729	$stable - hounds, ensemble - ladies$	$mission - monkeys$
Message-Topic	622	$pages - scene, speech - measures$	$exhibition - glamour$
Instrument-Agency	517	$user - console, eye - telescope$	$companies - governments$
Content-Container	486	$document - folder, pictures - box$	$message - paper$

Table 1: Semantic Relation Classification data

tors of the two entities.

$$sim(a, b) = sim((a_1 \oplus a_2), (b_1 \oplus b_2)) \quad (4)$$

- **pairwise addition** Pairwise similarities between respective entities are added up. If we use cosine similarity, this is only slightly different from the concatenative method. Vector addition proved to work well as a compositional representation (Mitchell and Lapata, 2010), despite the fact that word order is ignored.

$$sim(a, b) = sim(a_1, b_1) + sim(a_2, b_2) \quad (5)$$

A potential problem with this addition objective is that different properties of words are expressed on a different scale and, as a consequence, terms sharing these properties have a higher cosine similarity than terms that are similar with respect to a flatter property. It can be overcome by using multiplication instead of addition (Levy and Goldberg, 2014):

- **pairwise multiplication**

$$sim(a, b) = sim(a_1, b_1) \times sim(a_2, b_2) \quad (6)$$

3.2 Analogies

This is an adaptation of the measure proposed for *queen* = *king* - *man* + *woman* (Mikolov et al., 2013b). Vector arithmetics give us the vector which fills the analogy, and we can search for the word b_2 whose embedding vector has the greatest similarity to it:

$$argmax_{b_2} = sim(b_2, (b_1 - a_1 + a_2)) \quad (7)$$

which, using cosine similarity for sim, can be written as a combination of similarities (Levy et al., 2015a), as

$$argmax_{b_2} = sim(b_2, b_1) - sim(b_2, a_1) + sim(b_2, a_2) \quad (8)$$

Mikolov (2013b) notes that this measure is qualitatively similar to the relational similarity model in (Turney, 2012), which predicts similarity between members of the word pairs $(x_b, x_d), (x_c, x_d)$ and dissimilarity for (x_a, x_d).

In the current context, we do not look for the missing b_2 which maximizes the equation. Instead, we have different pairs a and b, and we aim to calculate $sim(a, b)$ to quantify how much the analogy *queen* - *woman* = *king* - *man* holds.

- **difference** Focuses on the similarity of b_1, b_2 and a_1, a_2, but does not take into account the pairwise distances between the individual entity vectors.

$$sim(a, b) = sim((a_1 - a_2), (b_1 - b_2)) \quad (9)$$

Levy et al. (2015a) propose a multiplicative version of the analogy formula. We tried to adapt it; however, this measure is not symmetrical (conceived to find b_2 which maximizes the form) and the adaptation produced bad results.

3.3 IN-OUT similarities

This metric is a combination of first order and second order similarities between the two entity pairs, adapted to relational similarity: a and b are similar if a_1 is similar to b_1 and also similar to the *contexts of* b_2, the opposite entity in b.

In the current experiment, second order similarities are estimated using both input and output vectors generated by word2vec's CBOW model. In this model, the IN vectors of words get closer to the OUT vectors of other words that they co-occur with.

Words with a high input-output similarity tend to appear in the context of each other. This similarity combination was recently included for a few, different tasks. It was shown to improve information retrieval (Nalisnick et al., 2016). Pennington et al. (2014) propose to use second-order similarity to improve similarity calculation between words. Their proposed formula combines first and second order similarity, normalized by the reflective second order similarity of the words with themselves. Finally, Melamud et al. (2015) used a combination of input-output similarities for lexical substitution.

In these contexts, the use of second-order similarities[1] is based on the observation that words are similar if they tend to appear in similar contexts, or if they tend to appear in the contexts of each other. In our experiment, second order similarities are used in a different way and with a different purpose. Second-order similarities are calculated between *opposite elements* of the entity pairs. We combine those similarities by taking the in-in similarity between a_1 and b_1, and the in-out similarities between a_1 and b_2, and between a_2 and b_1. Our motivation is to add relational information in a form which also preserves pairwise similarity information, as both are relevant for calculating relational similarities. We do it by using a co-occurrence component which gives higher score between more prototypical example pairs. A pairwise similarity can be high even if the entities are similar, but their relation is not: an obvious example is ambiguity (when they are similar with respect to a meaning, but co-occur with their pair in an other meaning). If an entity is similar to one argument of a relation and is also likely to appear in the context of the other argument, it indicates a higher likelihood of being an instance of the same relation.

- **additive in-out**

$$sim(a,b) = sim(a_1,b_1) + sim(a_2,b_2) + sim_2(a_1,b_2) + sim_2(a_2,b_1) \quad (10)$$

where sim_2 designates the second order similarity and is calculated as follows:

$$sim_2(x_1,y_2) = sim(x_1^{in}, y_2^{out}) + sim(x_1^{out}, y_2^{in}) \quad (11)$$

- **multiplicative in-out**: The same as above, but addition is replaced by multiplication in sim and sim_2.

$$sim(a,b) = sim(a_1,b_1) * sim(a_2,b_2) * sim_2(a_1,b_2) * sim_2(a_2,b_1) \quad (12)$$

where

$$sim_2(x_1,y_2) = sim(x_1^{in}, y_2^{out}) * sim(x_1^{out}, y_2^{in}) \quad (13)$$

4 Clustering Experiments

For supervised classification tasks, it is desirable to adapt word2vec's hyperparameters to the task and the data at hand (Levy et al., 2015a). The interaction between hyperparameters is also to be considered (Krebs and Paperno, 2016). However, our experiment is a clustering scenario aimed at exploratory analysis on a vector space created by pre-trained word embeddings; therefore, we set the parameters once and in advance.

We trained a word2vec CBOW model (Mikolov et al., 2013a) with negative sampling and a window size of 10 words on the ukWaC corpus (Baroni et al., 2009), and extracted both input and output vectors of size = 400 to build the vector combinations above. This size corresponds to the best performing model in the comparative paper by Baroni et al. (2012). An adjacency matrix was constructed for each vector/similarity combination using cosine similarity.

Clustering was implemented with Cluto's (Zhao et al., 2005) clustering function which takes the adjacency matrix as input. We used a hierarchical agglomerative clustering with the unweighted average distance (UPGMA) criterion function[2].

4.1 Evaluation as classification

At first, we ran the clustering with 9 clusters (the number of classes in the standard) and tried to make one-to-one correspondences between the standard and the output. Every cluster is mapped to the standard class that shares the more elements with. We

[1] Note that we use the term "second order similarity" in the sense of word-to-context similarity, unlike Pennington et al. (2014).

[2] We observed that these settings are sensitive to the chaining effect and there is probably room for improvement by experimenting with different task-specific clustering parameters.

then calculate precision and recall for each standard class (zero if the class doesn't show up as a majority class in any cluster). Average class-based precision and recall is reported, as well as the number of classes in the standard that could be assigned. These scores were published for the SemEval task participants, but ours are not comparable because we only consider one cluster for each class, and because we did the clustering on the training data.

INPUT	classes found	P	R	F
a_1(base)	5	0.1700	0.2086	0.1873
add	6	0.1918	0.1973	0.1945
conc	6	0.2031	0.2115	0.2072
in-out.add	8	0.2635	**0.2192**	**0.2393**
mult	7	0.1824	0.1493	0.1642
in-out.mult	5	0.1102	0.1232	0.1163
diff	5	**0.3762**	0.0918	0.1476

Table 2: Class-based results for 9 clusters

4.2 Evaluation as clustering

While the scores above can be indicative of the potential of different representations, they do not provide information on other aspects as cluster stability, purity, the amount of post-processing needed. Above all, in a completely unsupervised setting, the number of classes in the standard is not known and cluster quality (precision) plays an important role with respect to interpretability: it is easier to unify two homogeneous clusters than to separate a noisy one. We ran complementary experiments with different numbers of clusters. Table 3 indicates results for 20 and 30 clusters. The input-output combination method still has an advantage, and concatenation and multiplication also perform well. However, the advantages over the baseline are less significant than when the number of clusters was identical to the standard.

In the next runs, we measure how stable the different clustering solutions are with settings that are structurally very different from the standard, i.e. have significantly more clusters. Class-based precision and recall are less relevant measures in this setting, since they take the average over the nine standard classes and not over the produced clusters. We therefore decided to use *modified purity* (Korhonen et al., 2008), adapted for structurally different clustering solution. Modified purity gives the proportion of word pairs belonging to the majority class c in their cluster k:

$$PUR = \frac{\sum_{i=1}^{|K|} \max_j |w\ in\ k_i \cap w\ in\ c_j|}{\sum_{i=1}^{|K|} w\ in\ k_i} \quad (14)$$

INPUT	#clust	P	R	F
a_1(base)	20	**0.3429**	0.1642	0.2221
add	20	0.2434	0.1843	0.2098
conc	20	0.2718	**0.2116**	0.2380
in-out.add	20	0.2947	0.2076	**0.2436**
mult	20	0.3405	0.1886	0.2428
in-out.mult	20	0.2711	0.1432	0.1874
diff	20	0.2997	0.1161	0.1674
a_1(base)	30	0.3855	0.1712	0.2371
add	30	0.2714	0.1726	0.2110
conc	30	0.3331	0.1862	0.2389
in-out.add	30	0.3548	0.1947	**0.2514**
mult	30	0.3037	**0.1995**	0.2408
in-out.mult	30	**0.3916**	0.1304	0.1957
diff	30	0.3770	0.1318	0.1953

Table 3: Class-based results for 20 and 30 clusters

Modified purity is indicative of the quality and interpretability of the clusters. It favorizes small clusters, but singleton clusters were discarded. This measure corresponds to prediction accuracy in classification if we assign the majority label to clusters. Two series of runs were evaluated: for 10, 20... up to 50, and for 60, 70... up to 100 clusters. Average results are reported. These scores indicate the average purity of clusters over different runs.

INPUT	PUR
a_1(baseline)	0.2940
add	0.3059
conc	0.3107
in-out.add	**0.3235**
mult	0.2575
in-out.mult	0.2119
diff	0.2297

Table 4: Cluster-based results, 10-50 clusters

INPUT	PUR
a_1(baseline)	0.3291
add	0.3578
conc	**0.3737**
in-out.add	0.3674
mult	0.3235
in-out.mult	0.2587
diff	0.3058

Table 5: Cluster-based results, 60-100 clusters

5 Discussion

We note that the baseline and the simple pairwise combinations have a high performance because they already capture arguments' semantic types successfully. This also lies behind previous success on the analogy dataset. Moreover, the nature of semantic spaces and of semantic datasets is such that they typically contain close or quasi-identical variants for the same phenomenon, that the baselines identify easily.

The additive input-output combination shows promising results, especially when it comes to capturing the structure: in the clustering setting with 9 clusters, it identifies 8 classes out of 9 in the standard. This indicates a good potential in differentiating between relation types, especially because the standard is conceived in a way that it contains strongly related classes. It outperforms every other measure until the number of clusters grows significantly above those in the standard (Table 5), when the concatenative measure catches up. The baseline performs well, but additive methods all beat it, while difference is especially weak. Pairwise multiplication is good at recognizing the structure (7 classes out of 9), but not good at assigning elements.

Multiplicative methods show a fluctuating performance, especially the multiplicative input-output combination. This is due to the higher variance in similarities obtained by multiplication (in the case of input-output combination, 6 operands are multiplied), combined with the agglomerative clustering, which is sensitive to chaining.

The very high precision of the baseline method with a large number of clusters (Table 3) is noteworthy but not unexpected. Individual similarities have a strong precision for the easily identifiable clusters, while additional relational information is mostly expected to improve recall.

6 Conclusion and Future Work

We presented an experiment to identify relational similarities in word embedding compositions at a large scale, using an unsupervised approach. On the one hand, our results confirm the recent finding that many of the success attributed to vector arithmetics for analogies come from similarities between individual elements. On the other hand, taking second order similarity into account, we can improve relational similarities and take a step toward a meaningful representation for entity couples in a semantic relation.

The baseline performs well and is difficult to enrich with relation-aware information. The results indicate that the vector offset method for analogies, which replaces the pairwise similarity, is the least efficient in capturing generic semantic relations at a large scale. The vector difference representation does not conserve pairwise similarities and the offsets do not prove to be constant enough for unsupervised clustering. Multiplicative methods do not scale up either, although to a lesser extent: they capture some of the relational information, but this happens at the expense of losing precision from individual similarities. Pairwise similarities can be better exploited in an additive or concatenative setting. Moreover, they can be meaningfully complemented by including second order similarities without losing too much information for precise classification. The input-output combination measure coherently outperformed the other combinations in almost every setting, indicating a better potential for unsupervised experiments.

Unsupervised relation classification is a very challenging task for several reasons. Some relation instances are lexical by nature and, therefore, can be expected to show up in the same cluster based on distributional cues. On the other hand, contextual relation instances tend to have relation-specific indicators when they co-occur, but their individual vectors will not reveal this information (unless they co-occur very often). Moreover, semantic relations differ with respect to the semantic constraints they put on their arguments. For instance, the second argument of the Content-Container relation tend to belong to a specific semantic class in the standard (*bag, box, trunk, case, drawer...*), while both arguments of the Cause-Effect relation are much freer (*gas, prices, pain, acts, species* and *pyrolysis, collapse, contraction, society, noise*). Any future development towards an automated unsupervised classification needs to take these aspects into account and work towards a hybrid solution by separating relations with semantically constrained arguments from free ones, as well as adapting the clustering method to handle outliers.

Acknowledgments

This work was generously supported by the program "Investissements d'Avenir" overseen by the French National Research Agency, ANR-10-LABX-0083 (Labex EFL). We are grateful to the anonymous reviewers for their valuable comments that helped to improve the paper.

References

Marco Baroni, Raffaela Bernardi, Ngoc-Quynh Do, and Chung-Chieh. Shan. 2012. Entailment above the word level in distributional semantics. In *ACL '12*.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3).

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL '14*.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *EMNLP'10*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid O Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations*.

Amaç Herdağdelen and Marco Baroni. 2009. Bagpack: A general framework to represent semantic relations. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, GEMS '09.

Jerry Hobbs and Ellen Riloff. 2010. Information extraction. In *Handbook of Natural Language Processing, Second Edition*.

David Jurgens, Peter Turney, Saif M. Mohammad, and Keith Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the Workshop on Semantic Evaluations*.

Anna Korhonen, Yuval Krymolowski, and Nigel Collier. 2008. The choice of features for classification of verbs in biomedical texts. In *COLING'08*.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 21:5.

Alicia Krebs and Denis Paperno. 2016. When hyperparameters help: Beneficial parameter combinations in distributional semantic models. In *Joint Conference on Lexical and Computational Semantics (*SEM)*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML'14*.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *CONLL'14*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015a. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 3.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015b. Do supervised distributional methods really learn lexical inference relations? In *ACL '15*.

Tal Linzen. 2016. Issues in evaluating semantic spaces using word analogies. In *RepEval Workshop, ACL'16*.

Márton Makrai, Dávid Nemeskey, and András Kornai. 2013. Applicative structure in vector space models. In *Workshop on Continuous Vector Space Models and their Compositionality*.

Oren Melamud, Omer Levy, and Ido Dagan. 2015. A simple word embedding model for lexical substitution. In *Proceedings of the Vector Space Modeling for NLP Workshop, NAACL*.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *NAACL'13*.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34:8.

Jane Morris and Graeme Hirst. 2004. Non-classical lexical semantic relations. In *Workshop on Computational Lexical Semantics, HLT-NAACL'04*.

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving document ranking with dual word embeddings. In *Proceedings of the WWW Conference*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP'14*.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *COLING'14*.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte Im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *EACL'14*.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *ACL'16*.

Peter Turney. 2005. Measuring semantic similarity by latent relational analysis. In *IJCAI'05*.

Peter Turney. 2006. Similarity of semantic relations. *CoRR*, abs/cs/0608100.

Peter Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44.

Peter Turney and Saif M. Mohammad. 2014. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering*, 21:3.

Peter Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *CoRR*, abs/1003.1141.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *COLING '14*.

Ying Zhao, George Karypis, and Usama Fayyad. 2005. Hierarchical clustering algorithms for document datasets. *Data Mining for Knowledge Discovery*, 10.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *ACL'05*.

Temporal dynamics of semantic relations in word embeddings: an application to predicting armed conflict participants

Andrey Kutuzov
Department of Informatics
University of Oslo
andreku@ifi.uio.no

Erik Velldal
Department of Informatics
University of Oslo
erikve@ifi.uio.no

Lilja Øvrelid
Department of Informatics
University of Oslo
liljao@ifi.uio.no

Abstract

This paper deals with using word embedding models to trace the temporal dynamics of semantic relations between pairs of words. The set-up is similar to the well-known analogies task, but expanded with a time dimension. To this end, we apply incremental updating of the models with new training texts, including incremental vocabulary expansion, coupled with learned transformation matrices that let us map between members of the relation. The proposed approach is evaluated on the task of predicting insurgent armed groups based on geographical locations. The gold standard data for the time span 1994–2010 is extracted from the UCDP Armed Conflicts dataset. The results show that the method is feasible and outperforms the baselines, but also that important work still remains to be done.

1 Introduction and related work

In this research, we make an attempt to model the dynamics of worldwide armed conflicts on the basis of English news texts. To this end, we employ the well-known framework of Continuous Bag-of-Words modeling (Mikolov et al., 2013c) for training word embeddings on the English Gigaword news text corpus (Parker et al., 2011). We learn linear projections from the embeddings of geographical locations where violent armed groups were active to the embeddings of these groups. These projections are then applied to the embeddings and gold standard data from the subsequent year, thus predicting what entities act as violent groups in the next time slice. To evaluate our approach, we adapt the UCDP Armed Conflict Dataset (Gleditsch et al., 2002; Allansson et al., 2017) (see Section 2 for details).

Here is a simplified example of the task: given that in 2003, the *Kashmir Liberation Front* and *ULFA* were involved in armed conflicts in India, and *Lord's Resistance Army* in Uganda, predict entities playing the same role in 2004 in Iraq (the correct answers are *Ansar al-Islam*, *al-Mahdi Army* and *Islamic State*). The nature of the task is conceptually similar to that of analogical reasoning, but with the added complexity of temporal change.

Attempts to detect semantic change using unsupervised methods have a long history. Significant results have already been achieved in employing word embeddings to study diachronic language change. Among others, Eger and Mehler (2016) show that the embedding of a given word for a given time period to a large extent is a linear combination of its embeddings for the previous time periods. Hamilton et al. (2016) proposed an important distinction between cultural shifts and linguistic drifts. They proved that global embedding-based measures (comparing the similarities of words to all other words in the lexicon) are sensitive to regular processes of linguistic drift, while local measures (comparing nearest neighbors' lists) are a better fit for more irregular cultural shifts in word meaning.

Our focus here is on cultural shifts: it is not the dictionary meanings of the names denoting locations and armed groups that change, but rather their 'image' in the analyzed texts. Our measurement approach can also be defined as 'local' to some extent: the linear projections that we learn are mostly based and evaluated on the nearest neighborhood data. However, this method is different in that its scope is not single words but pairs of typed entities ('*location*' and '*armed group*' in our case) and the semantic relations between them.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1824–1829
Copenhagen, Denmark, September 7–11, 2017.

1.1 Contributions

The main contributions of this paper are:

1. We show that distributional semantic models, in particular word embeddings, can be used not only to trace diachronic semantic shifts in words, but also the temporal dynamics of semantic relations between pairs of words.

2. The necessary prerequisites for achieving decent performance in this task are incremental updating of the models with new textual data (instead of training from scratch each time new data is added) and some way of expanding the vocabulary of the models.

2 Gold standard data on armed conflicts

The UCDP/PRIO Armed Conflict Dataset maintained by the Uppsala Conflict Data Program and the Peace Research Institute Oslo is a manually annotated geographical and temporal dataset with information on armed conflicts, in the time period from 1946 to the present (Gleditsch et al., 2002; Allansson et al., 2017). It encodes conflicts, where at least one party is the government of a state. The Armed Conflict Dataset is widely used in statistical and macro-level conflict research; however, it was adapted and introduced to the NLP field only recently, starting with (Kutuzov et al., 2017). Whereas that work was focused on detecting the onset/endpoint of armed conflicts, the current paper further extends on this by using the dataset to evaluate the detection of changes in the semantic relation holding between participants of armed conflicts and their locations.

Two essential notions in the UCDP data are those of *event* and *armed conflict*. *Events* can evolve into full-scale *armed conflicts*, defined as contested incompatibilities that concern government and/or territory where the use of armed force between two parties, of which at least one is the government of a state, results in at least 25 battle-related deaths (Sundberg and Melander, 2013).

The subset of the data that we employ is the *UCDP Conflict Termination dataset* (Kreutz, 2010). It contains entries on starting and ending dates of about 2000 conflicts. We limit ourselves to the conflicts taking place between 1994 and 2010 (the Gigaword time span). Almost always, the first actor of the conflict (*sideA*) is the government of the corresponding location, and the second actor (*sideB*) is some insurgent armed group we are interested in. We omitted the conflicts where both sides were governments (about 2% of the entries) or where one of the sides was mentioned in the Gigaword less than 100 times (about 1% of the entries). In cases when the UCDP described the conflict as featuring several groups on the *sideB*, we created a separate entry for each.

This resulted in a test set of 673 conflicts, with 137 unique *Location–Insurgent* pairs throughout the whole time span (many pairs appear several times in different years). In total, it mentions 52 locations (with *India* being the most frequent) and 128 armed insurgent groups (with *ULFA* or *United Liberation Front of Assam* being the most frequent). This test set is available for subsequent reuse (http://ltr.uio.no/~andreku/armedconflicts/).

3 Predicting armed conflict participants

In this section, we provide a detailed description of our approach, starting with a synchronic example in 3.1 and then moving on to a toy diachronic example on one pair of years in 3.2. In the next section 4, we conduct evaluation on the full test set.

3.1 Synchronic modeling

We first conducted preliminary experiments to assess the hypothesis that the embeddings contain semantic relationships of the type ‘insurgent participant of an armed conflict in the location’. To this end, we trained a CBOW model on the full English Gigaword corpus (about 4.8 billion tokens in total), with a symmetric context window of 5 words, vector size 300, 10 negative samples and 5 iterations. Words with a frequency less than 100 were ignored during training. We used Gensim (Řehůřek and Sojka, 2010) for training, and in terms of corpus pre-processing we performed lemmatization, PoS-tagging and NER using Stanford CoreNLP (Manning et al., 2014). Named entities were concatenated to one token (for example, *United States* became *United::States_PROPN*).

Then, we used the 137 *Location–Insurgent* pairs derived in Section 2 to learn a projection matrix from the embeddings for locations to the embeddings for insurgents. The idea and the theory behind this approach are extensively described in (Mikolov et al., 2013b) and (Kutuzov et al., 2016), but essentially it involves training a linear regression which minimizes the error in transforming

	loc→group			group→loc		
λ	@1	@5	@10	@1	@5	@10
0.0	0.0	14.6	31.4	8.8	46.7	**70.8**
0.5	0.7	19.0	**35.0**	7.3	49.6	70.1
1.0	**2.2**	**19.7**	32.8	6.6	47.4	66.4

Table 1: Accuracies for synchronic projections from locations to armed groups, and vice versa

one set of vectors into another. Finding the optimal transformation matrix amounts to solving i normal equations (where i is the vector size in the embedding model being used), as shown in Equation 1:

$$\boldsymbol{\beta}_i = (\mathbf{X}^\intercal * \mathbf{X} + \lambda * L)^{-1} * \mathbf{X}^\intercal * y_i \qquad (1)$$

where $\mathbf{X}$ is the matrix of 137 location word vectors (input), y_i is the array of the ith components of 137 corresponding insurgent word vectors (correct predictions), L is the identity matrix of the size i, with 0 at the top left cell, and λ is a real number used to tune the influence of regularization term (if $\lambda = 0$, there is no regularization). $\boldsymbol{\beta}_i$ is the array of i optimal coefficients which transform an arbitrary location vector into the ith component of the corresponding insurgent vector. After learning such an array for each vector component, we have a linear projection matrix which can 'predict' an insurgent embedding from a location embedding.

To evaluate the resulting projections, we employed leave-one-out cross-validation, i.e., measuring the average accuracy of predictions on each pair from the test set, after training the matrix on all the pairs except the one used for the testing. The transformation matrix was dot-multiplied by the location vector from the test pair. Then, we found n nearest neighbors in the word embedding model for this predicted vector. If the real insurgent in the test pair was present in these n neighbors, the accuracy for this pair was 1, otherwise 0. In Table 1, the average accuracies with different values of λ and n are reported.

The relations of this kind are not symmetric: it is much easier to predict the location based on the insurgent than vice versa (see the right part of Table 1). Moreover, we find that the achieved results are roughly consistent with the performance of the same approach on the Google Analogies test set (Mikolov et al., 2013a). We converted the semantic sections in the Analogies test set containing only nouns (*capitals–common*, *capitals–world*, *cities in states*, *currency* and *family*) to sets of unique pairs. Then, linear projections with $\lambda = 1.0$ were learned and evaluated for each of them. The average accuracies over these sections were 13.0@1, 48.77@5 and 62.96@10.

The results on predicting armed groups are still worse than on the Google Analogies, because of 3 factors: 1) one-to-many relationships in the UCDP dataset (multiple armed groups can be active in the same location) make learning the transformation matrix more difficult; 2) the frequency of words denoting armed groups is lower than any of the words in the Google Analogies data set, thus, embeddings for them are of lower quality; 3) training the matrix on the whole Gigaword model is suboptimal, as the majority of armed groups were not active throughout all its time span.

All our experiments were also conducted using the very similar Continuous Skipgram models. However, as CBOW proved to consistently outperform Skipgram for our tasks, we only report results for CBOW, due to limited space.[1]

To sum up this section, many-to-one semantic relations between locations and insurgents do exist in the word embedding models. They are less expressed than one-to-one relations like those in the Google Analogies test set, but still can be found using linear projections. In the next section, we trace the dynamics of these relations as the models are updated with new data.

3.2 Diachronic modeling

Our approach to using learned transformation matrices to trace armed conflict dynamics through time consists of the following. We first train a CBOW model on the subsection of Gigaword texts belonging to the year 1994. Then, we incrementally update (train) this same model with new texts, saving a new model after each subsequent year. The size of the yearly subcorpora is about 250–320 million content words each. Importantly, we also use vocabulary expansion: new words are added to the vocabulary of the model if their frequency in the new yearly data satisfy our minimal threshold of 15.[2] Each yearly training session is performed in 5 iterations, with linearly decreasing learning rate. Note that we do not use any model alignment method (Procrustes, etc): our

[1] It seems CBOW is often better than Skipgram with linear projections; cf. the same claim in (Kutuzov et al., 2016).

[2] We did not experiment with different thresholds. It was initially set to the value which produced a reasonable vocabulary size of several hundred thousand words.

Pairs (size)	@1	@5	@10
All (38)	44.7	73.7	84.2
New (7)	14.3	28.6	42.9

Table 2: Projection accuracy for the isolated example experiment mapping from 2000 → 2001.

models are simply trained further with the new texts. A possible alternative to this can be incremental training of hierarchical softmax functions proposed in (Peng et al., 2017) or incremental negative sampling proposed in (Kaji and Kobayashi, 2017); we leave it for future work.

The experiment involves applying a learned transformation matrix across pairs of models. While in Section 4 we evaluate the approach across the entire Gigaword time period, this section reports a preliminary example experiment for the transition from 2000 to 2001 alone. This means we will have one model saved after sequential training for the years up to 2000, and one saved after year 2001. Our aim is to find out whether the *Location–Insurgent* projection learned on the first model is able to reveal conflicts that appear in 2001. Thus, we extract from the UCDP dataset all the pairs related to the conflicts which took place between 1994 and 2000 (91 pairs total). The projection is trained on their embeddings from the first model (actually, on 79 pairs, as 12 armed group names were not present in the 2000 model and subsequently skipped). Then, this projection is applied to the second model embeddings of the 47 locations, which are subject to armed conflicts in the year 2001 (38 after skipping pairs with out-of-vocabulary elements). Table 2 demonstrates the resulting performance (reflecting how close the predicted vectors are to the actual armed groups active in this or that location).

Note that out of 38 pairs from 2001, 31 were already present in the previous data set (ongoing conflicts). This explains why the evaluation on all the pairs gives high results. However, even for the new conflicts, the projection performance is encouraging. Among others, it managed to precisely spot the 2001 insurgency of the members of the *Kosovo Liberation Army* in Macedonia, notwithstanding the fact that the initial set of training pairs did not mention Macedonia at all. Thus, it seems that the models at least partially 'align' new data along the existing semantic axis trained before.

In the next section, we systematically evaluate our approach on the whole set of UCDP conflicts in the Gigaword years (1994–2010).

4 Evaluation of diachronic models

To evaluate our approach on all the UCDP data, we again tested how good it is in predicting the future conflicts based on the projection matrices learned from the previous years. We did this for all the years between 1994 and 2010. The evaluation metrics are the same as in the Section 3: we calculated the ratio of correctly predicted armed groups names from the conflict pairs, for which the UCDP datasets stated that these conflicts were active in this particular year. As before, the models employed in the experiment were incrementally trained on each successive year with vocabulary expansion. Words present in the gold standard but absent from the models under analysis were skipped. At the worst case, 25% of pairs were skipped from the test set; on average, 13% were skipped each year (but see the note below about the **incr. static** baseline). At test time, all the entities were lowercased.

We employ 3 baselines: 1) yearly models trained separately from scratch on the corpora containing texts from each year only (referred to as **separate** hereafter); 2) yearly models trained from scratch on all the texts from the particular year and the previous years (**cumulative** hereafter); 3) incrementally trained models without vocabulary expansion (**incr. static** hereafter).

Initially, the linear projections for all models were trained on all the conflict pairs from the past and present years, similar to Section 3.2 (dubbed **up-to-now** hereafter). However, the information about conflicts having ended several years before might not be strongly expressed in the model after it was incrementally updated with the data from all the subsequent years. For example, the 2005 model hardly contains much knowledge about the conflict relations between Mexico and the *Popular Revolutionary Army (EPR)* which stopped its activities after 1996. Thus, we additionally conducted a similar experiment, but this time the projections were learned only on the salient pairs (dubbed **previous**): that is, the pairs active in the last year up to which the model was trained.

Table 3 presents the results for these experiments, as well as baselines (averaged across 15 years). For the proposed **incr. dynamic** approach, the performance of the **previous** projections is

	Only in-vocabulary pairs						**All pairs, including OOV**					
	up-to-now			**previous**			**up-to-now**			**previous**		
	@1	@5	@10	@1	@5	@10	@1	@5	@10	@1	@5	@10
Separate	0.0	0.7	2.1	0.5	1.1	2.4	0.0	0.5	1.6	0.4	0.8	1.8
Cumulative	1.7	8.3	13.8	2.9	9.6	15.2	1.5	7.4	12.2	2.5	8.5	13.4
Incr. static	54.9	82.8	90.1	60.4	79.6	84.8	20.8	31.5	34.2	23.0	30.3	32.2
Incr. dynamic	32.5	64.5	72.2	42.6	64.8	71.5	28.1	56.1	**62.9**	**37.3**	**56.7**	62.6

Table 3: Average accuracies of predicting next-year insurgents on the basis of locations, using projections trained on the conflicts from all the preceding years (**up-to-now**) or the preceding year only (**previous**). Results for 3 baselines are shown along with the proposed **incremental dynamic** approach.

comparable to that of the **up-to-now** projections on the accuracies @5 and @10, and is even higher on the accuracy @1 (statistically significant with *t-test*, $p < 0.01$). Thus, the single-year projections are somewhat more 'focused', while taking much less time to learn, because of less training pairs.

The fact that our models were incrementally updated, not trained from scratch, is crucial. The results of the **separate** baseline look more like random jitter. The **cumulative** baseline results are slightly better, probably simply because they are trained on more data. However, they still perform much worse than the models trained using incremental updates. This is because the former models are not connected to each other, and thus are initialized with a different layout of words in the vector space. This gives rise to formally different directions of semantic relations in each yearly model (the relations themselves are still there, but they are rotated and scaled differently).

The results for the **incr. static** baseline, when tested only on the words present in the test model vocabulary (the left part of the table), seem better than those of the proposed **incr. dynamic** approach. This stems from the fact that incremental updating with static vocabulary means that we never add new words to the models; thus, they contain only the vocabulary learned from the 1994 texts. The result is that at test time we skip many more pairs than with the other approaches (about 62% in average). Subsequently, the projections are tested only on a minor part of the test sets.

Of course, skipping large parts of the data would be a major drawback for any realistic application, so the **incr. static** baseline is not really plausible. For comparison, the right part of Table 3 provides the accuracies for the setup in which all the pairs are evaluated (for pairs with OOV words the accuracy is always 0). Other tested approaches are not much affected by this change, but for **incr. static** the performance drops drastically. As a result, for the all pairs scenario, incremental updating with vocabulary expansion outperforms all the baselines (the differences are statistically significant with *t-test*, $p < 0.01$).

5 Conclusion

We have here shown how incrementally updated word embedding models with vocabulary expansion and linear projection matrices are able to trace the dynamics of subtle semantic relations over time. We applied this approach to the task of predicting armed groups active in particular geographical locations and showed that it significantly outperforms the baselines. However, it can be used for any kind of semantic relations. We believe that studying temporal shifts of such projections can lead to interesting findings far beyond the usual example of 'king is to queen as man is to woman'.

To our best knowledge, the behavior of semantic relations in updated word embedding models was not explored before. Our experiments show that the models do preserve these 'directions' and that the learned projections not only hold for the word pairs known to the initial model, but can also be used to predict relations for the new words.

In terms of future work, we plan to trace how quickly incremental updates to the model 'dilute' the projections, rendering them useless with time. We observed this performance drop in our experiments, and it would be interesting to know more about the regularities governing this deterioration. Also, for the particular task of analyzing armed conflicts, we plan to research ways of improving accuracy in predicting completely new armed groups not present in the training data, and the methods of filtering out locations not involved in armed conflicts.

References

Marie Allansson, Erik Melander, and Lotta Themnér. 2017. Organized violence, 1989–2016. *Journal of Peace Research*, 54(4).

Steffen Eger and Alexander Mehler. 2016. On the linearity of semantic change: Investigating meaning variation via dynamic graph models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 52–58, Berlin, Germany.

Nils Petter Gleditsch, Peter Wallensteen, Mikael Eriksson, Margareta Sollenberg, and Håvard Strand. 2002. Armed conflict 1946-2001: A new dataset. *Journal of Peace Research*, 39(5):615–637.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Cultural shift or linguistic drift? Comparing two computational measures of semantic change. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2116–2121, Austin, Texas, USA.

Nobuhiro Kaji and Hayato Kobayashi. 2017. Incremental skip-gram model with negative sampling. *arXiv preprint arXiv:1704.03956*.

Joakim Kreutz. 2010. How and when armed conflicts end: Introducing the UCDP conflict termination dataset. *Journal of Peace Research*, 47(2):243–250.

Andrey Kutuzov, Mikhail Kopotev, Tatyana Sviridenko, and Lyubov Ivanova. 2016. Clustering comparable corpora of Russian and Ukrainian academic texts: Word embeddings and semantic fingerprints. In *Proceedings of the Ninth Workshop on Building and Using Comparable Corpora*, pages 3–10.

Andrey Kutuzov, Erik Velldal, and Lilja Øvrelid. 2017. Tracing armed conflicts with diachronic word embedding models. In *Proceedings of the Events and Stories in the News workshop*, Vancouver, Canada. ACL.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Quoc Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013c. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition LDC2011T07. Technical report, Linguistic Data Consortium, Philadelphia.

Hao Peng, Jianxin Li, Yangqiu Song, and Yaopeng Liu. 2017. Incrementally learning the hierarchical softmax function for neural language models. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3267–327, San Francisco, California USA.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta.

Ralph Sundberg and Erik Melander. 2013. Introducing the UCDP georeferenced event dataset. *Journal of Peace Research*, 50(4):523–532.

Dynamic Entity Representations in Neural Language Models

Yangfeng Ji[*] **Chenhao Tan**[*] **Sebastian Martschat**[†] **Yejin Choi**[*] **Noah A. Smith**[*]
[*]Paul G. Allen School of Computer Science & Engineering, University of Washington
[†]Department of Computational Linguistics, Heidelberg University
{yangfeng,chenhao,yejin,nasmith}@cs.washington.edu
martschat@cl.uni-heidelberg.de

Abstract

Understanding a long document requires tracking how entities are introduced and evolve over time. We present a new type of language model, ENTITYNLM, that can explicitly model entities, dynamically update their representations, and contextually generate their mentions. Our model is generative and flexible; it can model an arbitrary number of entities in context while generating each entity mention at an arbitrary length. In addition, it can be used for several different tasks such as language modeling, coreference resolution, and entity prediction. Experimental results with all these tasks demonstrate that our model consistently outperforms strong baselines and prior work.

1 Introduction

Understanding a narrative requires keeping track of its participants over a long-term context. As a story unfolds, the information a reader associates with each character in a story increases, and expectations about what will happen next change accordingly. At present, models of natural language do not explicitly track entities; indeed, in today's language models, entities are no more than the words used to mention them.

In this paper, we endow a generative language model with the ability to build up a dynamic representation of each entity mentioned in the text. Our language model defines a probability distribution over the whole text, with a distinct generative story for entity mentions. It explicitly groups those mentions that corefer and associates with each entity a continuous representation that is updated by every contextualized mention of the entity, and that in turn affects the text that follows.

[*John*]$_1$ wanted to go to [*the coffee shop*]$_2$ in [*downtown Copenhagen*]$_3$. [*He*]$_1$ was told that [*it*]$_2$ sold [*the best beans*]$_4$.

Figure 1: ENTITYNLM explicitly tracks entities in a text, including coreferring relationships between entities like [*John*]$_1$ and [*He*]$_1$. As a language model, it is designed to predict that a coreferent of [*the coffee shop*]$_2$ is likely to follow "*told that*," that the referring expression will be "*it*", and that "*sold the best beans*" is likely to come next, by using entity information encoded in the dynamic distributed representation.

Our method builds on recent advances in representation learning, creating local probability distributions from neural networks. It can be understood as a recurrent neural network language model, augmented with random variables for entity mentions that capture coreference, and with dynamic representations of entities. We estimate the model's parameters from data that is annotated with entity mentions and coreference.

Because our model is generative, it can be queried in different ways. Marginalizing everything except the words, it can play the role of a language model. In §5.1, we find that it outperforms both a strong n-gram language model and a strong recurrent neural network language model on the English test set of the CoNLL 2012 shared task on coreference evaluation (Pradhan et al., 2012). The model can also identify entity mentions and coreference relationships among them. In §5.2, we show that it can easily be used to add a performance boost to a strong coreference resolution system, by reranking a list of k-best candidate outputs. On the CoNLL 2012 shared task test set, the reranked outputs are significantly better than the original top choices from the same system. Fi-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1830–1839
Copenhagen, Denmark, September 7–11, 2017.

nally, the model can perform entity cloze tasks. As presented in §5.3, it achieves state-of-the-art performance on the InScript corpus (Modi et al., 2017).

2 Model

A language model defines a distribution over sequences of word tokens; let X_t denote the random variable for the tth word in the sequence, x_t denote the value of X_t and $\mathbf{x}_t$ the distributed representation (embedding) of this word. Our starting point for language modeling is a recurrent neural network (Mikolov et al., 2010), which defines

$$p(X_t \mid \text{history}) = \operatorname{softmax}(\mathbf{W}_h \mathbf{h}_{t-1} + \mathbf{b}) \quad (1)$$

$$\mathbf{h}_{t-1} = \textsc{LSTM}(\mathbf{h}_{t-2}, \mathbf{x}_{t-1}) \quad (2)$$

where $\mathbf{W}_h$ and $\mathbf{b}$ are parameters of the model (along with word embeddings $\mathbf{x}_t$), LSTM is the widely used recurrent function known as "long short-term memory" (Hochreiter and Schmidhuber, 1997), and $\mathbf{h}_t$ is a LSTM hidden state encoding the history of the sequence up to the tth word.

Great success has been reported for this model (Zaremba et al., 2015), which posits nothing explicitly about the words appearing in the text sequence. Its generative story is simple: the value of each X_t is randomly chosen conditioned on the vector $\mathbf{h}_{t-1}$ encoding its history.

2.1 Additional random variables and representations for entities

To introduce our model, we associate with each word an additional set of random variables. At position t,

- R_t is a binary random variable that indicates whether x_t belongs to an entity mention ($R_t = 1$) or not ($R_t = 0$). Though not explored here, this is easily generalized to a categorial variable for the *type* of the entity (e.g., person, organization, etc.).

- $L_t \in \{1, \ldots, \ell_{max}\}$ is a categorical random variable if $R_t = 1$, which indicates the number of remaining words in this mention, including the current word (i.e., $L_t = 1$ for the last word in any mention). ℓ_{max} is a predefined maximum length fixed to be 25, which is an empirical value derived from the training corpora used in the experiments. If $R_t = 0$, then $L_t = 1$. We denote the value of L_t by ℓ_t.

- $E_t \in \mathcal{E}_t$ is the index of the entity referred to, if $R_t = 1$. The set $\mathcal{E}_t$ consists of $\{1, \ldots, 1 + \max_{t'<t} e_{t'}\}$, i.e., the indices of all previously mentioned entities plus an additional value for a new entity. Thus $\mathcal{E}_t$ starts as $\{1\}$ and grows monotonically with t, allowing for an arbitrary number of entities to be mentioned. We denote the value of E_t by e_t. If $R_t = 0$, then E_t is fixed to a special value $\varnothing$.

The values of these random variables for our running example are shown in Figure 2.

In addition to using symbolic variables to encode mentions and coreference relationships, we maintain a vector representation of each entity that evolves over time. For the ith entity, let $\mathbf{e}_{i,t}$ be its representation at time t. These vectors are different from word vectors ($\mathbf{x}_t$), in that they are not parameters of the model. They are similar to history representations ($\mathbf{h}_t$), in that they are derived through parameterized functions of the random variables' values, which we will describe in the next subsections.

2.2 Generative story

The generative story for the word (and other variables) at timestep t is as follows; forward-referenced equations are in the detailed discussion that follows.

1. If $\ell_{t-1} = 1$ (i.e., x_t is *not* continuing an already-started entity mention):
   - Choose r_t (Equation 3).
   - If $r_t = 0$, set $\ell_t = 1$ and $e_t = \varnothing$; then go to step 3. Otherwise:
     - If there is no embedding for the new candidate entity with index $1 + \max_{t'<t} e_{t'}$, create one following §2.4.
     - Select the entity e_t from $\{1, \ldots, 1 + \max_{t'<t} e_{t'}\}$ (Equation 4).
     - Set $\mathbf{e}_{current} = \mathbf{e}_{e_t, t-1}$, which is the entity embedding of e_t before timestep t.
     - Select the length of the mention, ℓ_t (Equation 5).
2. Otherwise,
   - Set $\ell_t = \ell_{t-1} - 1$, $r_t = r_{t-1}$, $e_t = e_{t-1}$.
3. Sample x_t from the word distribution given the LSTM hidden state $\mathbf{h}_{t-1}$ and the current

$X_{1:12}$:	John	wanted	to	go	to	the	coffee	shop	in	downtown	Copenhagen	.
$R_{1:12}$:	1	0	0	0	0	1	1	1	0	1	1	0
$E_{1:12}$:	1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	2	2	2	$\emptyset$	3	3	$\emptyset$
$L_{1:12}$:	1	1	1	1	1	3	2	1	1	2	1	1

$X_{13:22}$:	He	was	told	that	it	sold	the	best	beans	.
$R_{13:22}$:	1	0	0	0	1	0	1	1	1	.
$E_{13:22}$:	1	$\emptyset$	$\emptyset$	$\emptyset$	2	$\emptyset$	4	4	4	$\emptyset$
$L_{13:22}$:	1	1	1	1	1	1	3	2	1	0

Figure 2: The random variable values in ENTITYNLM for the running example in Figure 1.

(or most recent) entity embedding $\mathbf{e}_{current}$ (Equation 6). (If $r_t = 0$, then $\mathbf{e}_{current}$ still represents the most recently mentioned entity.)

4. Advance the RNN, i.e., feed it the word vector $\mathbf{x}_t$ to compute $\mathbf{h}_t$ (Equation 2).

5. If $r_t = 1$, update $\mathbf{e}_{e_t,t}$ using $\mathbf{e}_{e_t,t-1}$ and $\mathbf{h}_t$, then set $\mathbf{e}_{current} = \mathbf{e}_{e_t,t}$. Details of the entity updating are given in §2.4.

6. For every entity $e_\iota \in \mathcal{E}_t \setminus \{e_t\}$, set $\mathbf{e}_{\iota,t} = \mathbf{e}_{\iota,t-1}$ (i.e., no changes to other entities' representations).

Note that at any given time step t, $\mathbf{e}_{current}$ will always contain the most recent vector representation of the most recently mentioned entity.

A generative model with a similar hierarchical structure was used by Haghighi and Klein (2010) for coreference resolution. Our approach differs in two important ways. First, our model defines a joint distribution over all of the text, not just the entity mentions. Second, we use representation learning rather than Bayesian nonparametrics, allowing natural integration with the language model.

2.3 Probability distributions

The generative story above referenced several parametric distributions defined based on vector representations of histories and entities. These are defined as follows.

For $r \in \{0, 1\}$,

$$p(R_t = r \mid \mathbf{h}_{t-1}) \propto \exp(\mathbf{h}_{t-1}^\top \mathbf{W}_r \mathbf{r}), \tag{3}$$

where $\mathbf{r}$ is the parameterized embedding associated with r, which paves the way for exploring entity type representations in future work; $\mathbf{W}_r$ is a parameter matrix for the bilinear score for $\mathbf{h}_{t-1}$ and $\mathbf{r}$.

To give the possibility of predicting a new entity, we need an entity embedding beforehand with index $(1 + \max_{t'<t} e_{t'})$, which is randomly sampled from Equation 7. Then, for every $e \in \{1, \ldots, 1 + \max_{t'<t} e_{t'}\}$:

$$\begin{aligned} &p(E_t = e \mid R_t = 1, \mathbf{h}_{t-1}) \\ &\quad \propto \exp(\mathbf{h}_{t-1}^\top \mathbf{W}_{entity} \mathbf{e}_{e,t-1} + \mathbf{w}_{dist}^\top \mathbf{f}(e)), \end{aligned} \tag{4}$$

where $\mathbf{e}_{e,t-1}$ is the embedding of entity e at time step $t-1$ and $\mathbf{W}_{entity}$ is the weight matrix for predicting entities using their continuous representations. The score above is normalized over values $\{1, \ldots, 1 + \max_{t'<t} e_{t'}\}$. $\mathbf{f}(e)$ represents a vector of distance features associated with e and the mentions of the existing entities. Hence two information sources are used to predict the next entity: (i) contextual information $\mathbf{h}_{t-1}$, and (ii) distance features $\mathbf{f}(e)$ from the current mention to the closest mention from each previously mentioned entity. $\mathbf{f}(e) = \mathbf{0}$ if e is a new entity. This term can also be extended to include other surface-form features for coreference resolution (Martschat and Strube, 2015; Clark and Manning, 2016b).

For the chosen entity e_t from Equation 4, the distribution over its mention length is drawn according to

$$\begin{aligned} &p(L_t = \ell \mid \mathbf{h}_{t-1}, \mathbf{e}_{e_t,t-1}) \\ &\qquad \propto \exp(\mathbf{W}_{length,\ell}^\top [\mathbf{h}_{t-1}; \mathbf{e}_{e_t,t-1}]), \end{aligned} \tag{5}$$

where $\mathbf{e}_{e_t,t-1}$ is the most recent embedding of the entity e_t, not updated with $\mathbf{h}_t$. The intuition is that $\mathbf{e}_{e_t,t-1}$ will help contextual information $\mathbf{h}_{t-1}$ to select the residual length of entity e_t. $\mathbf{W}_{length}$ is the weight matrix for length prediction, with $\ell_{max} = 25$ rows.

Finally, the probability of a word x as the next token is jointly modeled by $\mathbf{h}_{t-1}$ and the vector representation of the most recently mentioned entity $\mathbf{e}_{current}$:

$$\begin{aligned} &p(X_t = x \mid \mathbf{h}_{t-1}, \mathbf{e}_{current}) \\ &\qquad \propto \text{CFSM}(\mathbf{h}_{t-1} + \mathbf{W}_e \mathbf{e}_{current}), \end{aligned} \tag{6}$$

where $\mathbf{W}_e$ is a transformation matrix to adjust the dimensionality of $\mathbf{e}_{current}$. CFSM is a class factorized softmax function (Goodman, 2001; Baltescu and Blunsom, 2015). It uses a two-step prediction with predefined word classes instead of direct prediction on the whole vocabulary, and reduces the time complexity to the log of vocabulary size.

2.4 Dynamic entity representations

Before predicting the entity at step t, we need an embedding for the new candidate entity with index $e' = 1 + \max_{t'<t} e_{t'}$ if it does not exist. The new embedding is generated randomly, according to a normal distribution, then projected onto the unit ball:

$$\begin{aligned} \mathbf{u} &\sim \mathcal{N}(\mathbf{r}_1, \sigma^2 \mathbf{I}); \\ \mathbf{e}_{e',t-1} &= \frac{\mathbf{u}}{\|\mathbf{u}\|_2}, \end{aligned} \tag{7}$$

where $\sigma = 0.01$. The time step $t-1$ in $\mathbf{e}_{e',t-1}$ means the current embedding contains no information from step t, although it will be updated once we have $\mathbf{h}_t$ and if $E_t = e'$. $\mathbf{r}_1$ is the parameterized embedding for $R_t = 1$, which will be jointly optimized with other parameters and is expected to encode some generic information about entities. All the initial entity embeddings are centered on the mean $\mathbf{r}_1$, which is used in Equation 3 to determine whether the next token belongs to an entity mention. Another choice would be to initialize with a zero vector, although our preliminary experiments showed this did not work as well as random initialization in Equation 7.

Assume $R_t = 1$ and $E_t = e_t$, which means x_t is part of a mention of entity e_t. Then, we need to update $\mathbf{e}_{e_t,t-1}$ based on the new information we have from $\mathbf{h}_t$. The new embedding $\mathbf{e}_{e_t,t}$ is a convex combination of the old embedding ($\mathbf{e}_{e_t,t-1}$) and current LSTM hidden state ($\mathbf{h}_t$) with the interpolation (δ_t) determined dynamically based on a bilinear function:

$$\begin{aligned} \delta_t &= \sigma(\mathbf{h}_t^\top \mathbf{W}_\delta \mathbf{e}_{e_t,t-1}); \\ \mathbf{u} &= \delta_t \mathbf{e}_{e_t,t-1} + (1-\delta_t)\mathbf{h}_t; \\ \mathbf{e}_{e_t,t} &= \frac{\mathbf{u}}{\|\mathbf{u}\|_2}, \end{aligned} \tag{8}$$

This updating scheme will be used to update e_t in *each* of all the following ℓ_t steps. The projection in the last step keeps the magnitude of the entity embedding fixed, avoiding numeric overflow. A similar updating scheme has been used by Henaff et al. (2016) for the "memory blocks" in their recurrent entity network models. The difference is that their model updates all memory blocks in each time step. Instead, our updating scheme in Equation 8 only applies to the selected entity e_t at time step t.

2.5 Training objective

The model is trained to maximize the log of the joint probability of $\mathbf{R}, \mathbf{E}, \mathbf{L}$, and $\mathbf{X}$:

$$\begin{aligned} \ell(\boldsymbol{\theta}) &= \log P(\mathbf{R}, \mathbf{E}, \mathbf{L}, \mathbf{X}; \boldsymbol{\theta}) \\ &= \sum_t \log P(R_t, E_t, L_t, X_t; \boldsymbol{\theta}), \end{aligned} \tag{9}$$

where $\boldsymbol{\theta}$ is the collection of all the parameters in this model. Based on the formulation in §2.3, Equation 9 can be decomposed as the sum of conditional log-probabilities of each random variable at each time step.

This objective requires the training data annotated as in Figure 2. We do not assume that these variables are observed at test time.

3 Implementation Details

Our model is implemented with DyNet (Neubig et al., 2017) and available at https://github.com/jiyfeng/entitynlm. We use AdaGrad (Duchi et al., 2011) with learning rate $\lambda = 0.1$ and ADAM (Kingma and Ba, 2014) with default learning rate $\lambda = 0.001$ as the candidate optimizers of our model. For all the parameters, we use the initialization tricks recommended by Glorot and Bengio (2010). To avoid overfitting, we also employ dropout (Srivastava et al., 2014) with the candidate rates as $\{0.2, 0.5\}$.

In addition, there are two tunable hyperparameters of ENTITYNLM: the size of word embeddings and the dimension of LSTM hidden states. For both of them, we consider the values $\{32, 48, 64, 128, 256\}$. We also experiment with the option to either use the pretrained GloVe word embeddings (Pennington et al., 2014) or randomly initialized word embeddings (then updated during training). For all experiments, the best configuration of hyperparameters and optimizers is selected based on the objective value on the development data.

4 Evaluation Tasks and Datasets

We evaluate our model in diverse use scenarios: (i) language modeling, (ii) coreference resolution, and (iii) entity prediction. The evaluation on language modeling shows how the internal entity representation, when marginalized out, can improve the perplexity of language models. The evaluation on coreference resolution experiment shows how our new language model can improve a competitive coreference resolution system. Finally, we employ an entity cloze task to demonstrate the generative performance of our model in predicting the next entity given the previous context.

We use two datasets for the three evaluation tasks. For language modeling and coreference resolution, we use the English benchmark data from the CoNLL 2012 shared task on coreference resolution (Pradhan et al., 2012). We employ the standard training/development/test split, which includes 2,802/343/348 documents with roughly 1M/150K/150K tokens, respectively. We follow the coreference annotation in the CoNLL dataset to extract entities and ignore the singleton mentions in texts.

For entity prediction, we employ the InScript corpus created by Modi et al. (2017). It consists of 10 scenarios, including grocery shopping, taking a flight, etc. It includes 910 crowdsourced simple narrative texts in total and 18 stories were ignored due to labeling problems (Modi et al., 2017). On average, each story has 12.4 sentences, 24.9 entities and 217.2 tokens. Each entity mention is labeled with its entity index. We use the same training/development/test split as in (Modi et al., 2017), which includes 619, 91, 182 texts, respectively.

Data preprocessing

For the CoNLL dataset, we lowercase all tokens and remove any token that only contains a punctuation symbol unless it is in an entity mention. We also replace numbers in the documents with the special token NUM and low-frequency word types with UNK. The vocabulary size of the CoNLL data after preprocessing is 10K. For entity mention extraction, in the CoNLL dataset, one entity mention could be embedded in another. For embedded mentions, only the enclosing entity mention is kept. We use the same preprocessed data for both language modeling and coreference resolution evaluation.

For the InScript corpus, we apply similar data preprocessing to lowercase all tokens, and we replace low-frequency word types with UNK. The vocabulary size after preprocessing is 1K.

5 Experiments

In this section, we present the experimental results on the three evaluation tasks.

5.1 Language modeling

Task description. The goal of language modeling is to compute the marginal probability:

$$P(\mathbf{X}) = \sum_{\mathbf{R},\mathbf{E},\mathbf{L}} P(\mathbf{X},\mathbf{R},\mathbf{E},\mathbf{L}). \tag{10}$$

However, due to the long-range dependency in recurrent neural networks, the search space of $\mathbf{R},\mathbf{E},\mathbf{L}$ during inference grows exponentially. We thus use importance sampling to approximate the marginal distribution of $\mathbf{X}$. Specifically, with the samples from a proposal distribution $Q(\mathbf{R},\mathbf{E},\mathbf{L}|\mathbf{X})$, the approximated marginal probability is defined as

$$\begin{aligned} P(\mathbf{X}) &= \sum_{\mathbf{R},\mathbf{E},\mathbf{L}} P(\mathbf{X},\mathbf{R},\mathbf{E},\mathbf{L}) \\ &= \sum_{\mathbf{R},\mathbf{E},\mathbf{L}} Q(\mathbf{R},\mathbf{E},\mathbf{L}\mid\mathbf{X}) \frac{P(\mathbf{X},\mathbf{R},\mathbf{E},\mathbf{L})}{Q(\mathbf{R},\mathbf{E},\mathbf{L}\mid\mathbf{X})} \\ &\approx \frac{1}{N} \sum_{\{\mathbf{r}^{(i)},\mathbf{e}^{(i)},\ell^{(i)}\}\sim Q} \frac{P(\mathbf{r}^{(i)},\mathbf{e}^{(i)},\ell^{(i)},\mathbf{x})}{Q(\mathbf{r}^{(i)},\mathbf{e}^{(i)},\ell^{(i)}\mid\mathbf{x})} \end{aligned} \tag{11}$$

A similar idea of using importance sampling for language modeling evaluation has been used by Dyer et al. (2016).

For language modeling evaluation, we train our model on the training set from the CoNLL 2012 dataset with coreference annotation. On the test data, we treat coreference structure as latent variables and use importance sampling to approximate the marginal distribution of $\mathbf{X}$. For each document, the model randomly draws $N = 100$ samples from the proposal distribution, discussed next.

Proposal distribution. For implementation of Q, we use a discriminative variant of ENTITYNLM by taking the current word x_t for predicting the entity-related variables in the same time step. Specifically, in the generative story described in §2.2, we delete step 3 (words are not generated, but rather conditioned upon), move step 4 before step 1, and replace $\mathbf{h}_{t-1}$ with $\mathbf{h}_t$ in the steps for predicting entity type R_t, entity E_t and mention length L_t. This model variant provides a

Model	Perplexity
1. 5-gram LM	138.37
2. RNNLM	134.79
3. ENTITYNLM	**131.64**

Table 1: Language modeling evaluation on the test sets of the English section in the CoNLL 2012 shared task. As mentioned in §4, the vocabulary size is 10K. ENTITYNLM does not require any coreference annotation on the test data.

conditional probability $Q(R_t, E_t, L_t \mid X_t)$ at each timestep.

Baselines. We compare the language modeling performance with two competitive baselines: 5-gram language model implemented in KenLM (Heafield et al., 2013) and RNNLM with LSTM units implemented in DyNet (Neubig et al., 2017). For RNNLM, we use the same hyperparameters described in §3 and grid search on the development data to find the best configuration.

Results. The results of ENTITYNLM and the baselines on both development and test data are reported in Table 1. For ENTITYNLM, we use the value of $2^{-\frac{1}{T}\sum_{t=1}^{T}\log P(X_t,R_t,E_t,L_t)}$ on the development set with coreference annotation to select the best model configuration and report the best number. On the test data, we are able to calculate perplexity by marginalizing all other random variables using Equation 11. To compute the perplexity numbers on the test data, our model only takes account of log probabilities on word prediction. The difference is that coreference information is only used for training ENTITYNLM and not for test. All three models reported in Table 1 share the same vocabulary, therefore the numbers on the test data are directly comparable. As shown in Table 1, ENTITYNLM outperforms both the 5-gram language model and the RNNLM on the test data. Better performance of ENTITYNLM on language modeling can be expected, if we also use the marginalization method defined in Equation 11 on the development data to select the best configuration. However, we plan to use the same experimental setup for all experiments, instead of customizing our model for each individual task.

5.2 Coreference reranking

Task description. We show how ENTITYLM, which allows an efficient computation of the probability $P(\mathbf{R}, \mathbf{E}, \mathbf{L}, \mathbf{X})$, can be used as a coreference reranker to improve a competitive coreference resolution system due to Martschat and Strube (2015). This task is analogous to the reranking approach used in machine translation (Shen et al., 2004). The specific formulation is as follows:

$$\arg \max_{\{\mathbf{r}^{(i)},\mathbf{e}^{(i)},\mathbf{l}^{(i)}\}\in\mathcal{K}} P(\mathbf{r}^{(i)}, \mathbf{e}^{(i)}, \mathbf{l}^{(i)}, \mathbf{x}) \qquad (12)$$

where $\mathcal{K}$ is the k-best list for a given document. In our experiments, $k = 100$. To the best of our knowledge, the problem of obtaining k-best outputs of a coreference resolution system has not been studied before.

Approximate k-best decoding. We rerank the output of a system that predicts an antecedent for each mention by relying on pairwise scores for mention pairs. This is the dominant approach for coreference resolution (Martschat and Strube, 2015; Clark and Manning, 2016a). The predictions induce an antecedent tree, which represents antecedent decisions for all mentions in the document. Coreference chains are obtained by transitive closure over the antecedent decisions encoded in the tree. A mention also can have an empty mention as antecedent, which denotes that the mention is non-anaphoric.

For extending Martschat and Strube's greedy decoding approach to k-best inference, we cannot simply take the k highest scoring trees according to the sum of edge scores, because different trees may represent the same coreference chain. Instead, we use an heuristic that creates an approximate k-best list on candidate antecedent trees. The idea is to generate trees from the original system output by considering suboptimal antecedent choices that lead to different coreference chains. For each mention pair (m_j, m_i), we compute the difference of its score to the score of the optimal antecedent choice for m_j. We then sort pairs in ascending order according to this difference and iterate through the list of pairs. For each pair (m_j, m_i), we create a tree $t_{j,i}$ by replacing the antecedent of m_j in the original system output with m_i. If this yields a tree that encodes different coreference chains from all chains encoded by trees in the k-best list, we add $t_{i,j}$ to the k-best list. In the case that we cannot generate a given number of trees (particularly for a short document with a large k), we pad the list with the last item added to the list.

Evaluation measures. For coreference resolution evaluation, we employ the CoNLL scorer (Pradhan et al., 2014). It computes three commonly used evaluation measures MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and CEAF$_e$ (Luo, 2005). We report the F_1 score of each evaluation measure and their average as the CoNLL score.

Competing systems. We employed CORT[1] (Martschat and Strube, 2015) as our baseline coreference resolution system. Here, we compare with the original (one best) outputs of CORT's latent ranking model, which is the best-performing model implemented in CORT. We consider two rerankers based on ENTITYNLM. The first reranking method only uses the log probability for ENTITYNLM to sort the candidate list (Equation 12). The second method uses a linear combination of both log probabilities from ENTITYNLM and the scores from CORT, where the coefficients were found via grid search with the CoNLL score on the development set.

Results. The reranked results on the CoNLL 2012 test set are reported in Table 2. The numbers of the baseline are higher than the results reported in Martschat and Strube (2015) since the feature set of CORT was subsequently extended. Lines 2 and 3 in Table 2 present the reranked best results. As shown in this table, both reranked results give more than 1% of CoNLL score improvement on the test set over CORT, which are significant based on an approximate randomization test[2].

Additional experiments also found that increasing k from 100 to 500 had a minor effect. That is because the diversity of each k-best list is limited by (i) the number of entity mentions in the document, (ii) the performance of the baseline coreference resolution system, and possibly (iii) the approximate nature of our k-best inference procedure. We suspect that a stronger baseline system (such as that of Clark and Manning, 2016a) could give greater improvements, if it can be adapted to provide k-best lists. Future work might incorporate the techniques embedded in such systems into ENTITYNLM.

[*I*]$_1$ was about to ride [*my*]$_1$ [*bicycle*]$_2$ to the [*park*]$_3$ one day when [*I*]$_1$ noticed that the front [*tire*]$_4$ was flat . [*I*]$_1$ realized that [*I*]$_1$ would have to repair [*it*]$_4$. [*I*]$_1$ went into [*my*]$_1$ [*garage*]$_5$ to get some [*tools*]$_5$. The first thing [*I*]$_1$ did was remove the xxxx

Figure 3: A short story on bicycles from the InScript corpus (Modi et al., 2017). The entity prediction task requires predicting xxxx given the preceding text either by choosing a previously mentioned entity or deciding that this is a "new entity". In this example, the ground-truth prediction is [*tire*]$_4$. For training, ENTITYNLM attempts to predict every entity. While, for testing, it predicts a maximum of 30 entities after the first three sentences, which is consistent with the experimental setup suggested by Modi et al. (2017).

5.3 Entity prediction

Task description. Based on Modi et al. (2017), we introduce a novel entity prediction task that tries to predict the next entity given the preceding text. For a given text as in Figure 3, this task makes a forward prediction based on only the left context. This is different from coreference resolution, where both left and right contexts from a given entity mention are used in decoding. It is also different from language modeling, since this task only requires predicting entities. Since ENTITYNLM is generative, it can be directly applied to this task. To predict entities in test data, R_t is always given and ENTITYNLM only needs to predict E_t when $R_t = 1$.

Baselines and human prediction. We introduce two baselines in this task: (i) the **always-new** baseline that always predicts "new entity"; (ii) a linear classification model using **shallow features** from Modi et al. (2017), including the recency of an entity's last mention and the frequency. We also compare with the model proposed by Modi et al. (2017). Their work assumes that the model has prior knowledge of all the participant types, which are specific to each scenario and fine-grained, e.g., rider in the bicycle narrative, and predicts participant types for new entities. This assumption is unrealistic for pure generative models like ours.

[1] https://github.com/smartschat/cort, we used version 0.2.4.5.

[2] https://github.com/smartschat/art

Model	CoNLL	MUC			B^3			CEAF$_e$		
		P	R	F_1	P	R	F_1	P	R	F_1
1. Baseline: CORT's one best	62.93	77.15	68.67	72.66	66.00	54.92	59.95	60.07	52.76	56.18
2. Rerank: EntityNLM	**64.00**	77.90	69.45	73.44	66.84	56.12	61.01	61.73	53.90	57.55
3. Rerank: EntityNLM + CORT	**64.04**	77.93	69.49	73.47	67.08	55.99	61.04	61.76	53.98	57.61

Table 2: Coreference resolution scores on the CoNLL 2012 test set. CORT is the best-performing model of Martschat and Strube (2015) with greedy decoding.

	Accuracy (%)
1. Baseline: always-new	31.08
2. Baseline: shallow features	45.34
3. Modi et al. (2017)	62.65
4. EntityNLM	**74.23**
5. *Human prediction*	77.35

Table 3: Entity prediction accuracy on the test set of the InScript corpus.

Therefore, we remove this assumption and adapt their prediction results to our formulation by mapping all the predicted entities that have not been mentioned to "new entity". We also compare to the adapted **human prediction** used in the InScript corpus. For each entity slot, Modi et al. (2017) acquired 20 human predictions, and the majority vote was selected. More details about human predictions are discussed in (Modi et al., 2017).

Results. Table 3 shows the prediction accuracies. EntityNLM (line 4) significantly outperforms both baselines (line 1 and 2) and prior work (line 3) ($p \ll 0.01$, paired t-test). The comparison between line 4 and 5 shows our model is even close to the human prediction performance.

6 Related Work

Rich-context language models. The originally proposed recurrent neural network language models only capture information within sentences. To extend the capacity of RNNLMs, various researchers have incorporated information beyond sentence boundaries. Previous work focuses on contextual information from previous sentences (Ji et al., 2016a) or discourse relations between adjacent sentences (Ji et al., 2016b), showing improvements to language modeling and related tasks like coherence evaluation and discourse relation prediction. In this work, EntityNLM adds explicit entity information to the language model, which is another way of adding a memory network for language modeling. Unlike the work by Tran et al. (2016), where memory blocks are used to store general contextual information for language modeling, EntityNLM assigns each memory block specifically to only one entity.

Entity-related models. Two recent approaches to modeling entities in text are closely related to our model. The first is the "reference-aware" language models proposed by Yang et al. (2016), where the referred entities are from either a predefined item list, an external database, or the context from the same document. Yang et al. (2016) present three models, one for each case. For modeling a document with entities, they use coreference links to recover entity clusters, though they only model entity mentions as containing a single word (an inappropriate assumption, in our view). Their entity updating method takes the latest hidden state (similar to $\mathbf{h}_t$ when $R_t = 1$ in our model) as the new representation of the current entity; no long-term history of the entity is maintained, just the current local context. In addition, their language model evaluation assumes that entity information is provided at test time (Yang, personal communication), which makes a direct comparison with our model impossible. Our entity updating scheme is similar to the "dynamic memory" method used by Henaff et al. (2016). Our entity representations are dynamically allocated and updated only when an entity appears up, while the EntNet from Henaff et al. (2016) does not model entities and their relationships explicitly. In their model, entity memory blocks are pre-allocated and updated simultaneously in each timestep. So there is no dedicated memory block for every entity and no distinction between entity mentions and non-mention words. As a consequence, it is not clear how to use their model for coreference reranking and entity prediction.

Coreference resolution. The hierarchical structure of our entity generation model is inspired by

Haghighi and Klein (2010). They implemented this idea as a probabillistic graphical model with the distance-dependent Chinese Restaurant Process (Pitman, 1995) for entity assignment, while our model is built on a recurrent neural network architecture. The reranking method considered in our coreference resolution evaluation could also be extended with samples from additional coreference resolution systems, to produce more variety (Ng, 2005). The benefit of such a system comes, we believe, from the explicit tracking of each entity throughout the text, providing entity-specific representations. In previous work, such information has been added as features (Luo et al., 2004; Björkelund and Kuhn, 2014) or by computing distributed entity representations (Wiseman et al., 2016; Clark and Manning, 2016b). Our approach complements these previous methods.

Entity prediction. The entity prediction task discussed in §5.3 is based on work by Modi et al. (2017). The main difference is that we do not assume that all entities belong to a previously known set of entity types specified for each narrative scenario. This task is also closely related to the "narrative cloze" task of Chambers and Jurafsky (2008) and the "story cloze test" of Mostafazadeh et al. (2016). Those studies aim to understand relationships between events, while our task focuses on predicting upcoming entity mentions.

7 Conclusion

We have presented a neural language model, EntityNLM, that defines a distribution over texts and the mentioned entities. It provides vector representations for the entities and updates them dynamically in context. The dynamic representations are further used to help generate specific entity mentions and the following text. This model outperforms strong baselines and prior work on three tasks: language modeling, coreference resolution, and entity prediction.

Acknowledgments

We thank anonymous reviewers for the helpful feedback on this work. We also thank the members of Noah's ARK and XLab at University of Washington for their valuable comments, particularly Eunsol Choi for pointing out the InScript corpus. This research was supported in part by a University of Washington Innovation Award, Samsung GRO, NSF grant IIS-1524371, the DARPA CwC program through ARO (W911NF-15-1-0543), and gifts by Google and Facebook.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *LREC Workshop on Linguistic Coreference*.

Paul Baltescu and Phil Blunsom. 2015. Pragmatic neural language modelling in machine translation. In *NAACL*.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *ACL*.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains. In *ACL*.

Kevin Clark and Christopher D. Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. In *EMNLP*.

Kevin Clark and Christopher D. Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. In *ACL*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *EMNLP*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256.

Joshua Goodman. 2001. Classes for fast maximum entropy training. In *ICASSP*.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *NAACL*.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *ACL*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. arXiv:1612.03969.

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

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2016a. Document context language models. In *ICLR (workshop track)*.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016b. A latent variable recurrent neural network for discourse-driven language models. In *NAACL-HLT*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *HLT-EMNLP*.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *ACL*.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:405–418.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

Ashutosh Modi, Ivan Titov, Vera Demberg, Asad Sayeed, and Manfred Pinkal. 2017. Modeling semantic expectation: Using script knowledge for referent prediction. *Transactions of the Association of Computational Linguistics*, 5:31–44.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. In *NAACL*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. arXiv:1701.03980.

Vincent Ng. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *ACL*.

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

Jim Pitman. 1995. Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields*, 102(2):145–158.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *ACL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *EMNLP-CoNLL*.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *NAACL*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. In *NAACL-HLT*.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *MUC*.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *NAACL*.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-aware language models. arXiv:1611.01628.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent neural network regularization. *ICLR*.

Towards Quantum Language Models

Ivano Basile
Scuola Normale Superiore, Pisa, Italy
ivano.basile@sns.it

Fabio Tamburini
FICLIT - University of Bologna, Italy
fabio.tamburini@unibo.it

Abstract

This paper presents a new approach for building Language Models using the Quantum Probability Theory, a Quantum Language Model (QLM). It mainly shows that relying on this probability calculus it is possible to build stochastic models able to benefit from quantum correlations due to interference and entanglement. We extensively tested our approach showing its superior performances, both in terms of model perplexity and inserting it into an automatic speech recognition evaluation setting, when compared with state-of-the-art language modelling techniques.

1 Introduction

Quantum Mechanics Theory (QMT) is one of the most successful theories in modern science. Despite its effectiveness in the physics realm, the attempts to apply it in other domains remain quite limited, excluding, of course, the large quantity of studies regarding Quantum Information Processing on quantum computers.

Only in recent years some scholars tried to embody principles derived from QMT into their specific fields, for example, by the Information Retrieval community (Zuccon et al., 2009; Melucci and van Rijsbergen, 2011; González and Caicedo, 2011; Melucci, 2015) and in the domain of cognitive sciences and decision making (Khrennikov, 2010; Busemeyer and Bruza, 2012; Aerts et al., 2013). In the machine learning field (Arjovsky et al., 2016; Wisdom et al., 2016; Jing et al., 2017) have used unitary evolution matrices for building deep neural networks obtaining interesting results, but we have to observe that their works do not adhere to QMT and use unitary evolution operators in a way not allowed by QMT. In recent years, also the Natural Language Processing (NLP) community started to look at QMT with interest and some studies using it have already been presented (Blacoe et al., 2013; Liu et al., 2013; Tamburini, 2014; Kartsaklis et al., 2016).

Language models (LM) are basic tools in NLP used in various applications, such as Automatic Speech Recognition (ASR), machine translation, part-of-speech tagging, etc., and were traditionally modeled by using N-grams and various smoothing techniques. Among the dozen of tools for computing N-gram LM, we will refer to CMU-SLM (with Good-Turing smoothing) (Clarkson and Rosenfeld, 1997) and IRSTLM (with Linear Witten-Bell smoothing) (Federico et al., 2008); the latter is the tool used in Kaldi (Povey et al., 2011b), one of the most powerful and used open-source ASR package that we will use for some of the experiments presented in the following sections.

In recent years new techniques from the Neural Networks (NN) domain have been introduced in order to enhance the performances of such models. Elman recurrent NN, as used in the RNNLM tool (Mikolov et al., 2010, 2011), or Long Short-Term Memory NN, as in the tool LSTMLM (Soutner and Müller, 2015), produce state-of-the-art performances for current language models.

This paper presents a different approach for building LM based on quantum probability theory. Actually, we present a QLM applicable only to problems defined on a small set of different tokens. This is a "proof-of-concept" study and our main aim is to show the potentialities of such approach rather than building a complete application for solving this problem for any setting.

The paper is organized as follows: we provide background on Quantum Probability Theory in Section 2 followed by the description of our proposed Quantum Language Model in Section 3. We then discuss some numerical issues mainly related

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1840–1849
Copenhagen, Denmark, September 7–11, 2017.

to the optimisation procedure in Section 4, and in Section 5 we present the experiments we did to validate our approach. In Section 6 we discuss our results and draw some provisional conclusions.

2 Quantum Probability Theory

In QMT the state of a system is usually described, in the most general case, by using density matrices over an Hilbert space $\mathcal{H}$. More specifically, a *density matrix* ρ is a positive semidefinite Hermitian matrix of unit trace, namely $\rho^\dagger = \rho, \quad \mathrm{Tr}(\rho) = 1$, and it is able to encode all the information about the state of a quantum system[1].

The measurable quantities, or *observables*, of the quantum system are associated to Hermitian matrices O defined on $\mathcal{H}$. The axioms of QMT specify how one can make predictions about the outcome of a measurement using a density matrix:

- the possible outcomes of a projective measurement of an observable O are its eigenvalues $\{\lambda_j\}$;
- the *probability* that the outcome of the measurement is λ_j is $P(\lambda_j) = \mathrm{Tr}(\rho\Pi_{\lambda_j}) = \mathrm{Tr}(\Pi_{\lambda_j}\rho)$, where Π_{λ_j} is the projector on the eigenspace of O associated to λ_j. Note that in the following we will use some properties of these kind of measurements, namely $\Pi^\dagger_{\lambda_j} = \Pi_{\lambda_j}$ and $\Pi^2_{\lambda_j} = \Pi_{\lambda_j}$;
- after the measurement the system state *collapses* in the following fashion: if the outcome of the measurement was λ_j, the collapse is
$$\rho' = \frac{\Pi_{\lambda_j}\rho\Pi_{\lambda_j}}{\mathrm{Tr}(\Pi_{\lambda_j}\rho\Pi_{\lambda_j})}$$
where the denominator is needed for trace normalization;
- time evolution of states using a fixed time step is described by a *unitary matrix* U over $\mathcal{H}$, i.e. $U^\dagger U = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Given a state ρ_t, at a specific time t, the system evolution without measurements modifies the state as:
$$\rho_{t+1} = U\rho_t U^\dagger.$$

See for example (Nielsen and Chuang, 2010) or (Vedral, 2007) for a complete introduction on QPT.

3 Quantum Language Models

In this section we describe our approach to build QLM that can compute probabilities for the occurrence of a sequence $\mathbf{w} = (w_1, w_2, ..., w_n)$ of length n, composed using N different symbols, the vocabulary containing all the words in the model, i.e. for every symbol w in the sequence $w \in \{0, ..., N-1\}$. We define a set of orthogonal N-dimensional vectors $\{\mathbf{e}_w : w \in \{0, ..., N-1\}\}$, spanning the complex space $\mathcal{H} = \mathbb{C}^N$; to measure the probability of a symbol w, collapsing the state over the space spanned by $\mathbf{e}_w$, we use the projector $\Pi_w = \mathbf{e}_w\mathbf{e}_w^\dagger$. Note that all the words in the vocabulary have been encoded as numbers corresponding to the N dimensions of the vector space $\mathcal{H}$.

Our method is sequential, from QMT point of view, in the sense that we use a quantum system that produces a single symbol upon measurement.

The basic idea is that the probabilistic information for a given sequence $\mathbf{w} = (w_1, w_2, ..., w_n)$ is encoded in the density matrix that results from the following process:

- **Inititalisation**

  Cond.Prob.: $P(w_1; \rho_0, U) = \mathrm{Tr}(\rho_0\Pi_{w_1})$

  Projection: $\rho'_1 = \frac{\Pi_{w_1}\rho_0\Pi_{w_1}}{\mathrm{Tr}(\Pi_{w_1}\rho_0\Pi_{w_1})}$

  Evolution: $\rho_1 = U\rho'_1 U^\dagger$
- **Recurrence** $(i = 2, .., n)$

  Cond.Prob.: $P(w_i|w_1, ..., w_{i-1}; \rho_0, U) = \mathrm{Tr}(\rho_{i-1}\Pi_{w_i})$

  Projection: $\rho'_i = \frac{\Pi_{w_i}\rho_{i-1}\Pi_{w_i}}{\mathrm{Tr}(\Pi_{w_i}\rho_{i-1}\Pi_{w_i})}$

  Evolution: $\rho_i = U\rho'_i U^\dagger$
- **Termination**
$$P(\mathbf{w}|\rho_0, U) = P(w_1; \rho_0, U) \cdot \prod_{i=2}^{n} P(w_i|w_1, ..., w_{i-1}; \rho_0, U)$$

The total probability $P(\mathbf{w}|\rho_0, U)$ for the given sequence is thus obtained, in the termination step, by multiplying the conditional probability $P(w_i|w_1, ..., w_{i-1}; \rho_0, U)$ for each word in the sequence.

We then use the initial density matrix ρ_0 and the time evolution unitary matrix U as parameters to optimise the *perplexity* Γ, evaluated on a training

[1] $\dagger$ marks the conjugate transpose of a vector/matrix and $\mathrm{Tr}(\cdot)$ is the trace of a matrix.

corpus of sequences S,

$$\Gamma(\rho_0, U) = \exp\Big(-\frac{1}{C} \sum_{\mathbf{w} \in S} \log P(\mathbf{w}|\rho_0, U) \Big)$$

which quantifies the uncertainty of the model. C is the number of tokens in the corpus.

Minimising Γ is equivalent of learning a model by fixing all the model parameters, a typical procedure in the machine learning domain.

3.1 Ancillary system

The problem with this setup is that the 'quantum effects' are completely washed out by the measurements on the system by using projectors. The resulting expression for the probability $P(\mathbf{w}|\rho_0, U)$ for a sequence $\mathbf{w}$ is identical to that obtained using a classical Markov model.

To solve this issue, our approach is to *avoid the complete collapse* of the state after each symbol measurement using a common technique in QMT: we introduce an *ancillary system* described by a fictitious D-dimensional Hilbert space, $\mathcal{H}_{ancilla} = \mathbb{C}^D$, and we couple the original system to the ancillary system. The resulting DN-dimensional Hilbert space is

$$\mathcal{H}_2 \equiv \mathcal{H}_{ancilla} \otimes \mathcal{H} = \mathbb{C}^{DN}$$

where $\otimes$ denotes the Kronecker product for matrices and D can be seen as a free hyper-parameter of the model. On this new space the projectors are now given by $\Pi_w^{(2)} = \mathbf{I}_D \otimes \Pi_w$, where $\mathbf{I}_D$ is the D-dimensional identity matrix.

The advantage of using this method is that the time evolution for the coupled system creates non-trivial correlations between the two entangled systems such that measuring and collapsing the symbol state keeps some information about the whole sequence stored in the ancillary part of the state. This information is then reshuffled into the symbol state via time evolution, resulting in a 'memory effect' that takes the whole sequence of symbols into account, thereby extending the idea behind the N-grams approach. Larger D values will results in more memory of this system and, of course, in a larger number of parameters to learn.

3.2 System evolution

We need to specify the system evolution for our coupled system. The simplest approach is to use a unitary $DN \times DN$ matrix U that acts on the entangled Hilbert space as shown before; it can be specified by $(DN)^2$ real parameters with a suitable parametrization (Spengler et al., 2010) that ensures the unitarity of U. However, in our preliminary experiments this approach resulted in an insufficient 'memory' capability for the QLM and in a very complex and slow minimisation procedure.

A different approach could be introduced by using a specific unitary matrix for each word, but this would lead to an enormous amount of parameters to learn with the optimization procedure.

There are a lot of techniques in NLP to represent single words with dense vectors (see for example (Mikolov et al., 2013) for the so called *word embeddings*). Following this idea, we can represent every symbol in our system with a specific p-dimensional vector trained using one of the available techniques $w \mapsto (\alpha_1(w), ..., \alpha_p(w))$ or fixed randomly.

We then work with a set of p $DN \times DN$ unitary matrices $\mathbf{U} = (U_1, ..., U_p)$, one for each component of the word vector, that are used to dynamically build a different system evolution matrix for each word in this way:

$$V(w) \equiv \prod_{i=1}^{p} U_i^{\alpha_i(w)}$$

This results in $p(DN)^2$ complex or $2p(DN)^2$ real parameters to be learned.

Essentially, we treat the words in our problem in different ways: the evolution operator for each word $V(w)$ is build by using a combination of the operators $\mathbf{U}$ defined for each word-vector component, while, considering the system projection, we treat each word as one basis vector for the space $\mathcal{H}$.

Note that the choice to use a set $\{V(w)\}$ of operators, one for each word w, does not violate the linearity of quantum mechanics: let K be the quantum operation

$$K(\rho) = \sum_{w} V(w) \Pi_w^{(2)} \rho \, \Pi_w^{(2)} V^\dagger(w)$$

defined using projectors and evolution matrices. Then K is a valid (i.e. a Completely Positive Trace-preserving) evolution map that exactly reproduces our results in the sequence of evolutions and collapses.

The number of evolutionary operators is a trade-off: as we said before, defining only one operator U resulted in a poor performance of the proposed method in all the relevant experiments, while defining an operator for each word would produce too many parameters to be learned. The trade-off that we chose is to use one operator for each word-vector component, and build the set $\{V(w)\}$ from them as described above while preserving unitarity.

With regard to the initial density matrix ρ_0, we have to define it combining the initial density matrix of our system, ρ_0^s, and the initial density matrix of the ancilla, ρ_0^a. We defined ρ_0^s as a diagonal $N \times N$ matrix containing the classical Maximum Likelihood probability Estimation to have a specific symbol at the first sequence position:

$$\rho_0^s = \frac{1}{|S|} \sum_{\mathbf{w} \in S} \Pi_{w_1}$$

where S is again the set of all sequences in the training set and w_1 is the first word in each sequence $\mathbf{w}$. With regard to the ancilla system we do not know anything about it and thus we have to define ρ_0^a as the $D \times D$ diagonal matrix

$$\rho_0^a = \frac{\mathbf{I}_D}{\mathrm{Tr}(\mathbf{I}_D)} \,.$$

Consequently we can define ρ_0 as

$$\rho_0 = \rho_0^a \otimes \rho_0^s \,.$$

3.3 The final model

Putting all the ingredients together, we can finally write down the formula for the probability $P(\mathbf{w}|\rho_0, \mathbf{U})$ for a sequence $\mathbf{w}$ in the QLM specified by ρ_0 and $\mathbf{U}$. The product of conditional probabilities simplifies because of the normalising denominators added at each collapse and time evolution step. The result is:

$$\begin{aligned} P(\mathbf{w}|\rho_0, \mathbf{U}) = \mathrm{Tr}(&\Pi_{w_n}^{(2)} ... V^\dagger(w_2) \Pi_{w_2}^{(2)} V^\dagger(w_1) \\ &\Pi_{w_1}^{(2)} \rho \Pi_{w_1}^{(2)} V(w_1) \Pi_{w_2}^{(2)} V(w_2) ... \Pi_{w_n}^{(2)}) \end{aligned} \tag{1}$$

Using the fact that projectors have many zero entries one can also re-express this trace of the product of $DN \times DN$ matrices in terms of the trace of the product of $D \times D$ matrices. The formula for $P(\mathbf{w}|\rho_0, \mathbf{U})$ then simplifies to our final result

$$P(\mathbf{w}|\rho_0, \mathbf{U}) = \mathrm{Tr}(T^\dagger R T) \tag{2}$$

where the matrices R and T are defined as follows:

- in terms of entries $R_{i,j}$ with indices $i, j = 0, ..., D-1$, the matrix R is given by

  $$R_{i,j} = [\rho_0]_{Ni+w_1, Nj+w_1} \,.$$

  Note that only the value of first symbol in the sequence, w_1, enters in the expression. This is to be expected since R derives from the initial density matrix ρ_0;

- analogously, the matrix T that encodes the chain of combined collapses and time evolutions is given by the product $T = T^{(2)} T^{(3)} ... T^{(n)}$, where the matrices $T^{(k)}$ are given in entries, with indices $i, j = 0, ..., D-1$, by

  $$T_{i,j}^{(k)} = [V(w_{k-1})]_{Ni+w_{k-1}, Nj+w_k} \,.$$

  These matrices can be pre-calculated for every pair of the involved symbols, so that the calculation of $P(\mathbf{w}|\rho_0, \mathbf{U})$ for all the sequences will be very fast.

The detailed calculation for obtaining the equation (2) can be found in the supplementary material.

4 Optimisation and Numerical Issues

In order to optimise the parameters $\mathbf{U}$ we numerically minimise the perplexity Γ computed on a given training corpus of sequences S. This requires that the matrices $\mathbf{U}$ remain strictly unitary at every step of the minimisation procedure and it can be accomplished in various ways.

The most straightforward way is to employ an explicit parametrization for unitary matrices, as was done in (Spengler et al., 2010). Due to the transcendental functions employed in this parametrisation, this approach resulted in a functional form for Γ that has proven to be very challenging to minimise efficiently in our experiments.

A more elegant and efficient approach is to consider the entries of $\mathbf{U}$ as parameters (thereby ensuring a polynomial functional form for Γ) and to employ techniques of differential geometry to keep the parameters from leaving the unitary subspace at each minimisation step. This can be done using a modification of the approach outlined in (Tagare, 2011) that considers the unitary matrices subspace as a manifold, the *Stiefel manifold* $\mathrm{U}(DN)$. It is then possible to project the gradient ∇f of a generic function $f(M)$ of the matrix variable M on the tangent space of the Stiefel manifold and build a line search algorithm that sweeps out curves on this manifold so that at each point the parameters are guaranteed to form a unitary matrix.

In our case we have multiple unitary matrices $\mathbf{U} = (U_1, ..., U_p)$. This simply results having curves defined on $\mathrm{U}(DN)^p$, parametrised by a p-dimensional vector of $DN \times DN$ unitary matrices.

4.1 Formula for the gradient

To implement the curvilinear search method described in (Tagare, 2011) one needs an expression for the gradient $\mathbf{G} = (G_1, ..., G_p)$ of the probability function. This gradient is organised in a p-dimensional vector of $DN \times DN$ matrices, such that the component G_j is obtained by computing the matrix derivative of $P(\mathbf{w}|\rho_0, \mathbf{U})$ with respect to U_j either analytically or by applying some numerical estimate of the gradients, for example by using finite differences. The latter method, when working with thousands or millions of variables can be very time consuming and, usually, an explicit analytic formula for the gradient accelerates considerably all the required processing.

A lengthy analytic computation results in an explicit result. Firstly, we introduce the following objects:

- The *spectral decomposition* of U_j, given by $U_j = S_j D_j S_j^\dagger$, guaranteed to exist by the spectral theorem. S_j is unitary and the diagonal matrix D_j contains the eigenvalues $(u_{j1}, ..., u_{jDN})$ of U_j, $j = 1, ..., p$.

- The $DN \times DN$ matrices $C_j(\alpha)$ defined, in entries, by

$$[C_j(\alpha)]_{ab} = \frac{\overline{u_{ja}}^\alpha - \overline{u_{jb}}^\alpha}{\overline{u_{ja}} - \overline{u_{jb}}} \quad \text{if} \quad u_{ja} \neq u_{jb}$$

$$[C_j(\alpha)]_{ab} = \alpha \overline{u_{ja}}^{\alpha-1} \quad \text{if} \quad u_{ja} = u_{jb}$$

  where $\overline{u}$ is the complex conjugate of u.

- The $D \times DN$ matrices Q_k given in entries by

$$(Q_k)_{jA} = \delta_{Nj+w_k,A}$$

  where $j = 0, ..., D-1, \quad A = 0, ..., DN-1$.

- The *lesser* and *greater products* associated to the construction of system evolution matrices

$$V^{<j}(w) = \prod_{i=1}^{j-1} U_i^{\alpha_i(w)}$$

$$V^{>j}(w) = \prod_{i=j+1}^{n} U_i^{\alpha_i(w)}.$$

With these ingredients, the resulting formula for the components G_j of the gradient is

$$G_j = 2S_j \sum_{k=2}^{n} \Big\{ \Big[S_j^\dagger \Big(V^{<j}(w_{k-1})^\dagger Q_{k-1}^T \Big(\prod_{l=2}^{k-1} T^{(l)} \Big)^\dagger RT \Big(\prod_{l=k+1}^{n} T^{(l)} \Big)^\dagger Q_k V^{>j}(w_{k-1})^\dagger \Big) S_j \Big] \cdot C_j(\alpha_j(w_{k-1})) \Big\} S_j^\dagger \tag{3}$$

where $\cdot$ denotes the element-wise matrix product. Again, all the detailed calculations for obtaining the analytic expression (3) for the gradient G_j can be found in the supplementary material.

Using Tagare's method we can project the gradient onto the Stiefel manifold and build a curvilinear search algorithm for the minimisation.

To achieve this aim, Tagare proposed an Armijo-Wolfe line search inserted into a simple gradient descent procedure. We developed an extension of this algorithm combining the minimization over the Steifel manifold technique with a Moré-Thuente (1994) line search and a Conjugate Gradient minimisation algorithm that uses the Polak-Ribière method for the combination of gradients and search directions (Nocedal and Wright, 2006). All the experiments presented in the next section were performed using these methods.

The minimisation uses random mini-batches that increase their size during the training: they start with approximately one tenth of the training set dimension and increase to include all the instances using a parametrised logistic function. As stopping criterion we used the minimum of the perplexity function over the validation set as suggested in (Bengio, 2012; Prechelt, 2012) for other machine learning techniques.

5 Experiments and Results

5.1 Data

The TIMIT corpus is a read speech corpus designed to provide speech data for acoustic-phonetic studies and for the development and evaluation of automatic speech recognition systems (Garofolo et al., 1990). It contains broadband recordings of 630 speakers of eight major dialects of American English and includes time-aligned orthographic, phonetic and word transcriptions as well as a 16-bit, 16 kHz speech waveform file for each utterance.

In the speech community, the TIMIT corpus is the base for a standard phone-recognition task with specific evaluation procedures described in detail in (Lopes and Perdigao, 2011). We stick completely to this evaluation to test the effectiveness of our proposed model adopting, among the other procedures, the same splitting between the different data sets: the training set contains 3696 utterances (140225 phones), the validation set 400 utterances (15057 phones) and the test set 192 utterances (7215 phones).

5.2 Evaluation Results

We tested the proposed model by setting up two different evaluations: the first is an intrinsic evaluation of LM performances in terms of global perplexity on the TIMIT testset; the second is an extrinsic evaluation in which we replace the LM tools provided with the Kaldi ASR toolkit (Povey et al., 2011b) with our model in order to check the final system performances in a phone-recognition task and comparing them with the other state-of-the-art LM techniques briefly introduced in Section 1.

5.2.1 Intrinsic evaluation

The first experiment consisted in an evaluation of models perplexity (PPL) on the TIMIT testset. We compared the QLM model with two N-gram implementations, namely CMU-SLM (Clarkson and Rosenfeld, 1997) and IRSTLM (Federico et al., 2008), and two recurrent NN models able to produce state-of-the-art results in language modelling, the RNNLM (Mikolov et al., 2010, 2011) and the LSTMLM (Soutner and Müller, 2015) packages.

Table 1 shows the results of the intrinsic evaluation. With regard to RNNLM and LSTMLM results, only the best hyper-parameters combination after a lot of experiments, optimizing them on the validation set, has been inserted into the Table.

With regard to QLM, all the presented experiments are based on artificial word vectors produced randomly using values from the set $\{-1, 0, 1\}$ instead of real word embeddings. Every word vector is different from the others and we decided not to use real embeddings in order to test the core QMT method without adding the contextual information, contained in word embeddings, that could have helped our approach to obtain better performances, at least in principle.

Model	Parameters	PPL
CMU-SLM	2-gram	15.49
(Good-Turing	3-gram	14.28
smoothing)	4-gram	15.62
	5-gram	17.33
IRSTLM	2-gram	15.47
(linear Witten-	3-gram	14.07
Bell smoothing)	4-gram	15.55
	5-gram	17.53
RNNLM	280 neurons	13.32
LSTMLM	25 neurons, 1 layer	13.17
QLM	N=48, p=4, D=10	13.44
	N=48, p=4, D=20	**13.15**
	N=48, p=4, D=30	**13.10**
	N=48, p=4, D=40	**12.99**

Table 1: Perplexity (PPL) of the tested language-modelling techniques on the TIMIT testset. All the QLM results in bold face are better than the other systems we tested.

5.2.2 Extrinsic evaluation

The "TIMIT recipe" contained in the Kaldi distribution[2] reproduces exactly the same evaluation settings described in (Lopes and Perdigao, 2011) for a phone recognition task based on this corpus. Moreover, Kaldi provides some n-best rescoring scripts that apply RNNLM hypothesis rescoring and interpolate the results with the standard N-gram model results used in the evaluation. We slightly modified these scripts to work with LSTMLM and QLM in order to test different models using the same setting. This allowed us to replace the LM used in Kaldi and experiment with all the systems evaluated in the previous section.

Table 2 outlines the results we obtained replacing the LM technique into Kaldi ASR package w.r.t. the different ASR systems that the TIMIT recipe implements. These systems are built on top of MFCC, LDA, MLLT, fMLLR with CMN[3] features (see (Povey et al., 2011b; Rath et al., 2013) for all acronyms references and a complete feature

[2] https://github.com/kaldi-asr/kaldi

[3] MFCC: Mel-Frequency Cepstral Coefficients; LDA: Linear Discriminant Analysis; MLTT: Maximum Likelihood Linear Transform; fMLLR: feature space Maximum Likelihood Linear Regression; SAT: Speaker Adapted Training, i.e. train on fMLLR-adapted features; CMN: Cepstral Mean Normalization.

or recipe descriptions).

For this extrinsic evaluation we used the best models we obtained in the previous experiments interpolating their log-probability results for each utterance with the original bigram (or trigram) log-probability using a linear model with a ratio 0.25/0.75 between the original N-gram LM and the tested one as suggested in the standard Kaldi rescoring script. For this test we rescored the 10,000-best hypothesis.

We have to say that in this experiment we were not trying to build the best possible phone recogniser, but simply to compare the relative performances of the analysed LM techniques showing the effectiveness of QLM when used in a real application. Thus absolute Phone Error Rate is not so important here and it can be certainly possible to devise recognisers with better performances by applying more sophisticated techniques. For example (Peddinti et al., 2015) presented a method for lattice rescoring in Kaldi that exhibits better performances than the n-best rescoring we used to interpolate between n-grams and the tested models, but modifying it in order to test LSTMLM and QLM presented a lot of problems and thus we decided to use the simpler n-best approach. For completeness, the last column of Table 2 outlines the results obtained using this lattice rescoring method with RNNLM as described in (Peddinti et al., 2015).

6 Discussion and conclusions

We presented a new technique for building LM based on QMT, and its probability calculus, testing it extensively both with intrinsic and extrinsic evaluation methods.

The PPL results for the intrinsic evaluation, outlined in Table 1, show a clear superiority of the proposed method when compared with state-of-the-art techniques such as RNNLM and LSTMLM. It is interesting to note that even using $D = 20$, that means a system containing a quarter of parameters, therefore much less 'memory', w.r.t. the system with $D = 40$, we obtain a PPL performance better than the other methods.

With regard to the second experiment we made, an extrinsic evaluation where we replaced the LM of an ASR system with the LM produced by all the tested methods (see Table 2), QLM consistently exhibits the best performances for all the tested ASR systems from the Kaldi "TIMIT recipe". Despite using a n-best technique in this evaluation for hypothesis rescoring, that is known to perform worse than the lattice rescoring method proposed in (Peddinti et al., 2015), the QLM performances are even better than this method.

The approach we have presented in this paper is not without problems: the number of different word types in the considered language has to be small in order to keep the model computationally tractable. Even if the code we used in the evaluations is analytically highly optimised, the training of this model is rather slow and requires relevant computational resources even for small problems. On the contrary, inference is very quick, faster than the RNNLM and LSTMLM packages we tested.

The main research question that drove this work was to verify if the distinguishing properties of quantum probability theory, namely interference and system entaglement that could allow the ancilla to have a "potentially infinite" memory, were enough to build stochastic systems more powerful than those built using classical probabilities or those built using recurrent NN. Our main aim was not to build a complete model to handle all possible LM scenarios, but to present a "proof-of-concept" study to test the potentialities of this approach. For this reason we tried to keep the model as simple as possible using orthogonal projectors: for measuring probabilities, projecting the system state, each word is mapped onto a single basis vector and the dimension of the system Hilbert space, N, is equal to the number of different words. Given the matrix dimensions that we have to manage when we add the ancilla, $DN \times DN$, this setting does not scale to real LM problems (e.g. the Brown corpus), even though the calculations are performed using $D \times D$ submatrices, but allowed us to successfully verify the research question. For the same reason out-of-vocabulary words cannot be handled in this model because there are no basis vectors assigned to them.

In order to overcome these limitations, this work can be extended by using generalized quantum measurements projectors (POVM) and by using a different structure for the system Hilbert space: instead of mapping each word onto a single basis vector we can span this space using as basis the same p-basis vectors used to define the V matrices. In this way we will project the system state on a generic word vector built as a superposi-

Kaldi	**IRSTLM**	**N-Best rescoring**			**Lattice**
ASR	**2-gram**	**IRSTLM 2-gram LM interp. with:**			**rescoring**
System		**RNNLM**	**LSTMLM**	**QLM**	**RNNLM**
tri1	26.32	25.74	*25.09*	**24.59**	25.70
tri2	24.14	23.34	23.23	**23.05**	*23.17*
tri3	21.55	21.07	21.22	**20.35**	*20.85*
SGMM2	19.15	18.99	*18.52*	**18.23**	18.75
Dan NN	22.27	22.20	22.26	**21.80**	*22.05*
Kaldi	**IRSTLM**	**N-Best rescoring**			**Lattice**
ASR	**3-gram**	**IRSTLM 3-gram LM interp. with:**			**rescoring**
System		**RNNLM**	**LSTMLM**	**QLM**	**RNNLM**
tri1	25.64	25.39	*24.86*	**24.59**	25.42
tri2	23.16	23.13	*22.90*	**22.65**	22.97
tri3	20.80	20.57	*20.68*	**20.04**	*20.68*
SGMM2	18.64	18.41	18.48	**18.23**	*18.27*
Dan NN	21.72	21.90	21.95	**21.34**	*21.48*

Table 2: Phone-recognition performances, in terms of Phone Error Rate, for the TIMIT dataset and the different Kaldi ASR models, rescoring the 10,000-best solutions with the tested LM techniques interpolated with the IRSTLM bigrams and trigrams LM (the standard LM used in Kaldi). In boldface the best performing system and in italics the second best. Kaldi ASR systems descriptions: tri1 = a triphone model using 13 dim. MFCC+Δ+$\Delta\Delta$; tri2 = tri1+LDA+MLLT; tri3 = tri2+SAT; SGMM2 = Semi-supervised Gaussian Mixture Model (Huang and Hasegawa-Johnson, 2010; Povey et al., 2011a); Dan NN = DNN model by (Zhang et al., 2014; Povey et al., 2015).

tion on the p-basis. Such improvement would reduce dramatically the dimensions of the matrices to $Dp \times Dp$ potentially mitigating the computational issue. Moreover, this would solve also the problem of out-of-vocabulary words allowing for a proper management of the large set of different words typical of real applications.

We are still working on these improvements and we will hope to get a complete model soon.

With this contribution we would like to raise also some interest in the community to analyse and develop more effective techniques, both on the modelling and minimisation/learning sides, to allow to build real world application based on this framework. QMT and its probability calculus seem to be promising methodologies to enhance the performances of our systems in NLP and certainly deserve further investigations.

Acknowledgments

We acknowledge the CINECA[4] award no. HP10C7XVUO under the ISCRA initiative, for the availability of HPC resources and support.

References

Diederik Aerts, Jan Broekaert, Liane Gabora, and Sandro Sozzo. 2013. Quantum structure and human thought. *Behavioral and Brain Sciences*, 36(3):274276.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. 2016. Unitary evolution recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - ICML'16*, pages 1120–1128.

Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade: Second Edition*, pages 437–478. Springer Berlin Heidelberg, Berlin, Heidelberg.

William Blacoe, Elham Kashefi, and Mirella Lapata. 2013. A quantum-theoretic approach to distributional semantics. In *Proceedings of Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Atlanta, Georgia*, pages 847–857.

Jerome R. Busemeyer and Peter D. Bruza. 2012. *Quantum Models of Cognition and Decision*. Cambridge University Press, New York, NY.

Philip Clarkson and Ronald Rosenfeld. 1997. Statistical language modeling using the cmu-cambridge

[4] https://www.cineca.it/en

toolkit. In *Proceedings of EUROSPEECH '97*, pages 2707–2710. ISCA.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. IRSTLM: an open source toolkit for handling large scale language models. In *INTERSPEECH 2008, 9th Annual Conference of the International Speech Communication Association, Brisbane, Australia*, pages 1618–1621.

John Garofolo, Lori Lamel, William Fisher, Jonathan Fiscus, David Pallett, Nancy Dahlgren, and Victor Zue. 1990. Darpa timit acoustic-phonetic continuous speech corpus cd-rom. *DARPA, TIMIT Acoustic-Phonetic Continuous Speech Corpus CD-ROM*.

Fabio A. González and Juan C. Caicedo. 2011. Quantum latent semantic analysis. In *Advances in Information Retrieval Theory, LNCS, 6931*, pages 52–63.

Jui Ting Huang and Mark Hasegawa-Johnson. 2010. Semi-supervised training of gaussian mixture models by conditional entropy minimization. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010*, pages 1353–1356.

Li Jing, Yichen Shen, Tena Dubcek, John Peurifoy, Scott A. Skirlo, Max Tegmark, and Marin Soljacic. 2017. Tunable efficient unitary neural networks (EUNN) and their application to RNN. In *Thirty-fourth International Conference on Machine Learning - ICML2017*.

Dimitrios Kartsaklis, Martha Lewis, and Laura Rimell. 2016. *Proceedings of the 2016 Workshop on Semantic Spaces at the Intersection of NLP, Physics and Cognitive Science*, volume 221. Electronic Proceedings in Theoretical Computer Science.

Andrei Y. Khrennikov. 2010. *Ubiquitous Quantum Structure: From Psychology to Finance*. Springer-Verlag Berlin Heidelberg.

Ding Liu, Xiaofang Yang, and Minghu Jiang. 2013. A novel classifier based on quantum computation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Sofia, Bulgaria*, pages 484–488.

Carla Lopes and Fernando Perdigao. 2011. Phoneme recognition on the timit database. In Ivo Ipsic, editor, *Speech Technologies*. InTech, Rijeka.

Massimo Melucci. 2015. *Introduction to Information Retrieval and Quantum Mechanics*. The Information Retrieval Series 35. Springer-Verlag Berlin Heidelberg.

Massimo Melucci and Keith van Rijsbergen. 2011. Quantum mechanics and information retrieval. In Massimo Melucci and Ricardo Baeza-Yates, editors, *Advanced Topics in Information Retrieval*, pages 125–155. Springer Berlin Heidelberg, Berlin, Heidelberg.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proc. of Workshop at ICLR*.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan*, pages 1045–1048.

Tomáš Mikolov, Stefan Kombrink, Anoop Deoras, Lukáš Burget, and Jan Černocký. 2011. Rnnlm - recurrent neural network language modeling toolkit. In *Proceedings of ASRU 2011*, pages 1–4.

Jorge J. Moré and David J. Thuente. 1994. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Softw.*, 20(3):286–307.

Michael A. Nielsen and Isaac L. Chuang. 2010. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press.

J. Nocedal and S. J. Wright. 2006. *Numerical Optimization*, 2nd edition. Springer, New York.

Vijayaditya Peddinti, Guoguo Chen, Vimal Manohar, Tom Ko, Daniel Povey, and Sanjeev Khudanpur. 2015. JHU aspire system: Robust LVCSR with tdnns, ivector adaptation and RNN-LMS. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2015, Scottsdale, AZ, USA*, pages 539–546.

Daniel Povey, Lukáš Burget, Mohit Agarwal, Pinar Akyazi, Feng Kai, Arnab Ghoshal, Ondřej Glembek, Nagendra Goel, Martin Karafiát, Ariya Rastrow, Richard C. Rose, Petr Schwarz, and Samuel Thomas. 2011a. The subspace gaussian mixture model-a structured model for speech recognition. *Comput. Speech Lang.*, 25(2):404–439.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011b. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society.

Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. 2015. Parallel training of dnns with natural gradient and parameter averaging. In *International Conference on Learning Representations - ICLR2015*.

Lutz Prechelt. 2012. Early stopping — but when? In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade: Second Edition*, pages 53–67. Springer Berlin Heidelberg, Berlin, Heidelberg.

Shakti P. Rath, Daniel Povey, Karel Veselý, and Jan Cernocký. 2013. Improved feature processing for deep neural networks. In *Proceedings of the 14th Annual Conference of the International Speech Communication Association - INTERSPEECH2013, Lyon, France*, pages 109–113.

Daniel Soutner and Luděk Müller. 2015. In Adrian-Horia Dediu, Carlos Martín-Vide, and Klára Vicsi, editors, *Statistical Language and Speech Processing: Third International Conference, SLSP 2015, Budapest, Hungary, November 24-26, 2015, Proceedings*, pages 267–274. Springer International Publishing.

Christoph Spengler, Marcus Huber, and Beatrix C Hiesmayr. 2010. A composite parameterization of unitary groups, density matrices and subspaces. *Journal of Physics A: Mathematical and Theoretical*, 43(38):385306.

H.D. Tagare. 2011. Notes on optimization on Stiefel manifolds. Technical report, Technical report, Yale University.

Fabio Tamburini. 2014. Are quantum classifiers promising? In *Proceedings of the First Italian Conference on Computational Linguistics CLiC-it 2014, Pisa, Pisa University Press*, pages 360–364.

Vlatko Vedral. 2007. *Introduction to Quantum Information Science*. Oxford University Press, USA.

Scott Wisdom, Thomas Powers, John Hershey, Jonathan Le Roux, and Les Atlas. 2016. Full-capacity unitary recurrent neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4880–4888. Curran Associates, Inc.

Xiaohui Zhang, Jan Trmal, Daniel Povey, and Sanjeev Khudanpur. 2014. Improving deep neural network acoustic models using generalized maxout networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 215–219. IEEE.

Guido Zuccon, Leif A. Azzopardi, and Keith van Rijsbergen. 2009. The quantum probability ranking principle for information retrieval. In Leif et al. Azzopardi, editor, *Advances in Information Retrieval Theory, LNCS, 5766*, pages 232–240.

Reference-Aware Language Models

Zichao Yang[1*], **Phil Blunsom**[2,3], **Chris Dyer**[1,2], **and Wang Ling**[2]
[1]Carnegie Mellon University, [2]DeepMind, and [3]University of Oxford
zichaoy@cs.cmu.edu, {pblunsom,cdyer,lingwang}@google.com

Abstract

We propose a general class of language models that treat reference as discrete stochastic latent variables. This decision allows for the creation of entity mentions by accessing external databases of referents (required by, e.g., dialogue generation) or past internal state (required to explicitly model coreferentiality). Beyond simple copying, our coreference model can additionally refer to a referent using varied mention forms (e.g., a reference to "Jane" can be realized as "she"), a characteristic feature of reference in natural languages. Experiments on three representative applications show our model variants outperform models based on deterministic attention and standard language modeling baselines.

1 Introduction

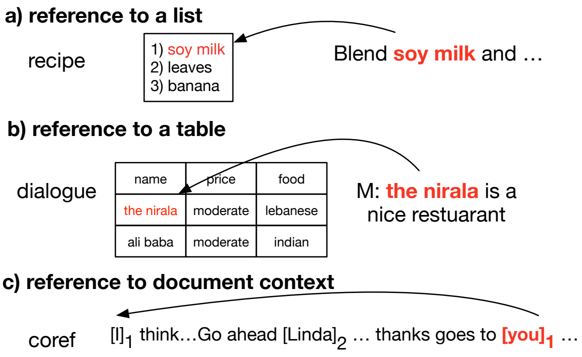


Figure 1: Reference-aware language models.

Referring expressions (REs) in natural language are noun phrases (proper nouns, common nouns, and pronouns) that identify objects, entities, and events in an environment. REs occur frequently and they play a key role in communicating information efficiently. While REs are common in natural language, most previous work does not model them explicitly, either treating REs as ordinary words in the model or replacing them with special tokens that are filled in with a post processing step (Wen et al., 2015; Luong et al., 2015). Here we propose a language modeling framework that explicitly incorporates reference decisions. In part, this is based on the principle of pointer networks in which copies are made from another source (Gülçehre et al., 2016; Gu et al., 2016; Ling et al., 2016; Vinyals et al., 2015; Ahn et al., 2016; Merity et al., 2016). However, in the full version of our model, we go beyond simple copying and enable coreferent mentions to have different forms, a key characteristic of natural language reference.

Figure 1 depicts examples of REs in the context of the three tasks that we consider in this work. First, many models need to refer to a list of items (Kiddon et al., 2016; Wen et al., 2015). In the task of recipe generation from a list of ingredients (Kiddon et al., 2016), the generation of the recipe will frequently refer to these items. As shown in Figure 1, in the recipe "Blend `soy milk` and ...", `soy milk` refers to the ingredient summaries. Second, reference to a database is crucial in many applications. One example is in task oriented dialogue where access to a database is necessary to answer a user's query (Young et al., 2013; Li et al., 2016; Vinyals and Le, 2015; Wen et al., 2015; Sordoni et al., 2015; Serban et al., 2016; Bordes and Weston, 2016; Williams and Zweig, 2016; Shang et al., 2015; Wen et al., 2016). Here we consider the domain of restaurant recommendation where a system refers to restaurants (name) and their attributes (address, phone number etc) in its responses. When the system

[*] Work completed at DeepMind.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1850–1859
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

says “`the nirala` is a nice restaurant”, it refers to the restaurant name `the nirala` from the database. Finally, we address references within a document (Mikolov et al., 2010; Ji et al., 2015; Wang and Cho, 2015), as the generation of words will often refer to previously generated words. For instance the same entity will often be referred to throughout a document. In Figure 1, the entity `you` refers to `I` in a previous utterance. In this case, copying is insufficient– although the referent is the same, the form of the mention is different.

In this work we develop a language model that has a specific module for generating REs. A series of decisions (should I generate a RE? If yes, which entity in the context should I refer to? How should the RE be rendered?) augment a traditional recurrent neural network language model and the two components are combined as a mixture model. Selecting an entity in context is similar to familiar models of attention (Bahdanau et al., 2014), but rather than being a soft decision that reweights representations of elements in the context, it is treated as a hard decision over contextual elements which are stochastically selected and then copied or, if the task warrants it, transformed (e.g., a pronoun rather than a proper name is produced as output). In cases when the stochastic decision is not available in training, we treat it as a latent variable and marginalize it out. For each of the three tasks, we pick one representative application and demonstrate our reference aware model's efficacy in evaluations against models that do not explicitly include a reference operation.

Our contributions are as follows:

- We propose a general framework to model reference in language. We consider reference to entries in lists, tables, and document context. We instantiate these tasks into three specific applications: recipe generation, dialogue modeling, and coreference based language models.
- We develop the first neural model of reference that goes being copying and can model (conditional on context) how to form the mention.
- We perform comprehensive evaluation of our models on the three data sets and verify our proposed models perform better than strong baselines.

2 Reference-aware language models

Here we propose a general framework for reference-aware language models.

We denote each document as a series of tokens $x_1, \ldots, x_L$, where L is the number of tokens. Our goal is to maximize $p(x_i \mid c_i)$, the probability of each word x_i given its previous context $c_i = x_1, \ldots, x_{i-1}$. In contrast to traditional neural language models, we introduce a variable z_i at each position, which controls the decision on which source x_i is generated from. Then the conditional probability is given by:

$$p(x_i, z_i \mid c_i) = p(x_i \mid z_i, c_i)p(z_i \mid c_i), \quad (1)$$

where z_i has different meanings in different contexts. If x_i is from a reference list or a database, then z_i is one dimensional and $z_i = 1$ denotes x_i is generated as a reference. z_i can also model more complex decisions. In coreference based language model, z_i denotes a series of sequential decisions, such as whether x_i is an entity, if yes, which entity x_i refers to. When z_i is not observed, we will train our model to maximize the marginal probability over z_i, i.e., $p(x_i|c_i) = \sum_{z_i} p(x_i|z_i, c_i)p(z_i|c_i)$.

2.1 Reference to lists

We begin to instantiate the framework by considering reference to a list of items. Referring to a list of items has broad applications, such as generating documents based on summaries etc. Here we specifically consider the application of recipe generation conditioning on the ingredient lists. Table. 1 illustrates the ingredient list and recipe for *Spinach and Banana Power Smoothie*. We can see that the ingredients `soy milk, spinach leaves, and banana` occur in the recipe.

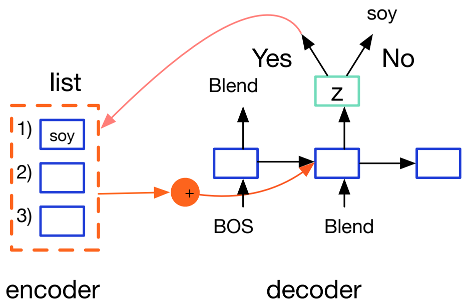


Figure 2: Recipe pointer

Let the ingredients of a recipe be $X = \{x_i\}_{i=1}^{T}$ and each ingredient contains L tokens $x_i =$

Ingredients	Recipe
1 cup plain `soy milk` 3/4 cup packed fresh `spinach leaves` 1 large `banana`, sliced	Blend `soy milk` and `spinach leaves` together in a blender until smooth. Add `banana` and pulse until thoroughly blended.

Table 1: Ingredients and recipe for *Spinach and Banana Power Smoothie*.

$\{x_{ij}\}_{j=1}^{L}$. The corresponding recipe is $y = \{y_v\}_{v=1}^{K}$. We would like to model $p(y|X) = \Pi_v p(y_v|X, y_{<v})$.

We first use a LSTM (Hochreiter and Schmidhuber, 1997) to encode each ingredient: $h_{i,j} = \text{LSTM}_\text{E}(W_E x_{ij}, h_{i,j-1}) \quad \forall i$. Then, we sum the resulting final state of each ingredient to obtain the starting LSTM state of the decoder. We use an attention based decoder:

$$\begin{aligned} s_v &= \text{LSTM}_\text{D}([W_\text{E} y_{v-1}, d_{v-1}], s_{v-1}), \\ p_v^{\text{copy}} &= \text{ATTN}(\{\{h_{i,j}\}_{i=1}^{T}\}_{j=1}^{L}, s_v), \\ d_v &= \sum_{ij} p_{v,i,j} h_{i,j}, \\ p(z_v|s_v) &= \text{sigmoid}(W[s_v, d_v]), \\ p_v^{\text{vocab}} &= \text{softmax}(W[s_v, d_v]), \end{aligned}$$

where $\text{ATTN}(h, q)$ is the attention function that returns the probability distribution over the set of vectors h, conditioned on any input representation q. A full description of this operation is described in (Bahdanau et al., 2014). The decision to copy from the ingredient list or generate a new word from the softmax is performed using a switch, denoted as $p(z_v|s_v)$. We can obtain a probability distribution of copying each of the words in the ingredients by computing $p_v^{\text{copy}} = \text{ATTN}(\{\{h_{i,j}\}_{i=1}^{T}\}_{j=1}^{L}, s_v)$ in the attention mechanism.

Objective: We can obtain the value of z_v through a string match of tokens in recipes with tokens in ingredients. If a token appears in the ingredients, we set $z_v = 1$ and $z_v = 0$ otherwise. We can train the model in a fully supervised fashion, i.e., we can obtain the probability of y_v as $p(y_v, z_v|s_v) = p_v^{\text{copy}}(y_v)p(1|s_v)$ if $z_v = 1$ and $p_v^{\text{vocab}}(y_v)(1 - p(1|s_{i,v}))$ otherwise.

However, it may be not be accurate. In many cases, the tokens that appear in the ingredients do not specifically refer to ingredients tokens. For examples, the recipe may start with “Prepare a cup of water”. The token “cup” does not refer to the “cup” in the ingredient list “1 cup plain soy milk”. To solve this problem, we treat z_i as a latent variable, we wish to maximize the marginal probability of y_v over all possible values of z_v. In this way, the model can automatically learn when to refer to tokens in the ingredients. Thus, the probability of generating token y_v is defined as:

$$\begin{aligned} p(y_v|s_v) &= p_v^{\text{vocab}}(y_v)p(0|s_v) + p_v^{\text{copy}}(y_v)p(1|s_v) \\ &= p_v^{\text{vocab}}(y_v)(1 - p(1|s_v)) + p_v^{\text{copy}}(y_v)p(1|s_v). \end{aligned}$$

If no string match is found for y_v, we simply set $p_v^{\text{copy}}(y_v) = 0$ in the above objective.

2.2 Reference to databases

We then consider the more complicated task of reference to database entries. Referring to databases is quite common in question answering and dialogue systems, in which databases are external knowledge and they are resorted to reply users’ query. In our paper, we consider the application of task-oriented dialogue systems in the domain of restaurant recommendations. Different from lists that are one dimensional, databases are two-dimensional and referring to table entries requires sophisticated model design.

To better understand the model, we first make a brief introduction of the data set. We use dialogues from the second Dialogue State Tracking Challenge (DSTC2) (Henderson et al., 2014). Table. 3 is one example dialogue from this dataset.

We can observe from this example, users get recommendations of restaurants based on queries that specify the area, price and food type of the restaurant. We can support the system’s decisions by incorporating a mechanism that allows the model to query the database to find restaurants that satisfy the users’ queries. A sample of our database (refer to data preparation part on how we construct the database) is shown in Table 2. We can observe that each restaurant contains 6 attributes that are generally referred in the dialogue dataset. As such, if the user requests a restaurant that serves “indian” food, we wish to train a model that can search for entries whose “food”

NAME	PRICE RANGE	FOOD	AREA	ADDRESS	POST CODE	PHONE
ali baba	moderate	lebanese	centre	59 Hills Road City Centre	CB 2, 1 NT	01462 432565
the nirala	moderate	indian	north	7 Milton Road Chesterton	CB 4, 1 UY	01223 360966

Table 2: Fragment of database for dialogue system.

M:	Hello, welcome to the Cambridge restaurant system? You can ask for restaurants by `area`, `price range` or `food type` . How may I help you?
U:	i want a `moderately priced` restaurant in the `north` town
M:	`the nirala` is a nice restaurant in the north of town in the moderate price range
U:	what is the `address`
M:	Sure , `the nirala` is on `7 Milton Road Chesterton`

Table 3: Example dialogue, M stands for Machine and U stands for User

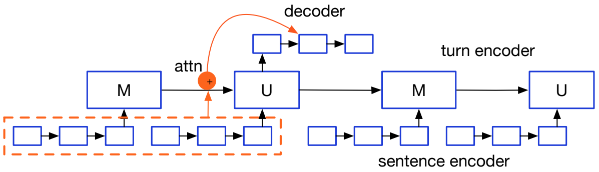


Figure 3: Hierarchical RNN Seq2Seq model. The red box denotes attention mechanism over the utterances in the previous turn.

column contains "indian". Now, we describe how we deploy a model that fulfills these requirements. We first introduce the basic dialogue framework in which we incorporates the table reference module.

Basic Dialogue Framework: We build a basic dialogue model based on the hierarchical RNN model described in (Serban et al., 2016), as in dialogues, the generation of the response is not only dependent on the previous sentence, but on all sentences leading to the response. We assume that a dialogue is alternated between a machine and a user. An illustration of the model is shown in Figure 3.

Consider a dialogue with T turns, the utterances from a user and a machines are denoted as $X = \{x_i\}_{i=1}^{T}$ and $Y = \{y_i\}_{i=1}^{T}$ respectively, where i is the i-th utterance. We define $x_i = \{x_{ij}\}_{j=1}^{|x_i|}$, $y_i = \{y_{iv}\}_{v=1}^{|y_i|}$, where x_{ij} (y_{iv}) denotes the j-th (v-th) token in the i-th utterance from the user (the machines). The dialogue sequence starts with a machine utterance and is given by $\{y_1, x_1, y_2, x_2, \ldots, y_T, x_T\}$. We would like to model the utterances from the machine

$$p(y_1, y_2, \ldots, y_T | x_1, x_2, \ldots, x_T) =$$
$$\prod_i p(y_i | y_{<i}, x_{<i}) = \prod_{i,v} p(y_{i,v} | y_{i,<v}, y_{<i}, x_{<i}).$$

We encode $y_{<i}$ and $x_{<i}$ into continuous space in a hierarchical way with LSTM: **Sentence Encoder**: For a given utterance x_i, We encode it as $h^x_{i,j} = \text{LSTM}_\text{E}(W_E x_{i,j}, h^x_{i,j-1})$. The representation of x_i is given by the $h^x_i = h^x_{i,|x_i|}$. The same process is applied to obtain the machine utterance representation $h^y_i = h^y_{i,|y_i|}$. **Turn Encoder**: We further encode the sequence $\{h^y_1, h^x_1, ..., h^y_i, h^x_i\}$ with another LSTM encoder. We shall refer the last hidden state as u_i, which can be seen as the hierarchical encoding of the previous i utterances. **Decoder**: We use u_{i-1} as the initial state of decoder LSTM and decode each token in y_i. We can express the decoder as:

$$s^y_{i,v} = \text{LSTM}_\text{D}(W_E y_{i,v-1}, s_{i,v-1}),$$
$$p^y_{i,v} = \text{softmax}(W s^y_{i,v}).$$

We can also incoroprate the attetionn mechanism in the decoder. As shown in Figure. 3, we use the attention mechanism over the utterance in the previous turn. Due to space limit, we don't present the attention based decoder mathmatically and readers can refer to (Bahdanau et al., 2014) for details.

2.2.1 Incorporating Table Reference

We now extend the decoder in order to allow the model to condition the generation on a database.

Pointer Switch: We use $z_{i,v} \in \{0, 1\}$ to denote the decision of whether to copy one cell from the table. We compute this probability as follows:

$$p(z_{i,v} | s_{i,v}) = \text{sigmoid}(W s_{i,v}).$$

Thus, if $z_{i,v} = 1$, the next token $y_{i,v}$ is generated from the database, whereas if $z_{i,v} = 0$, then the following token is generated from a softmax. We now describe how we generate tokens from the database.

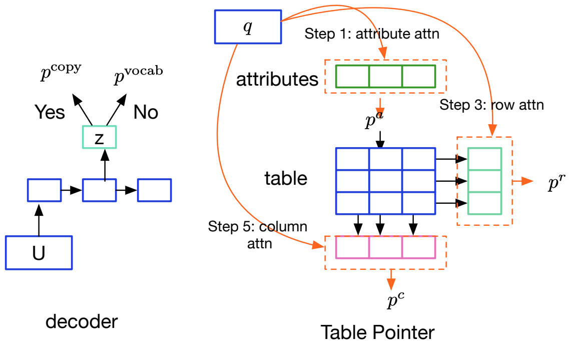


Figure 4: Decoder with table pointer.

We denote a table with R rows and C columns as $\{t_{r,c}\}, r \in [1, R], c \in [1, C]$, where $t_{r,c}$ is the cell in row r and column c. The attribute of each column is denoted as s_c, where c is the c-th attribute. $t_{r,c}$ and s_c are one-hot vector.

Table Encoding: To encode the table, we first build an attribute vector and then an encoding vector for each cell. The attribute vector is simply an embedding lookup $g_c = W_E s_c$. For the encoding of each cell, we first concatenate embedding lookup of the cell with the corresponding attribute vector g_c and then feed it through a one-layer MLP as follows: then $e_{r,c} = \tanh(W[W_E t_{r,c}, g_c])$.

Table Pointer: The detailed process of calculating the probability distribution over the table is shown in Figure 4. The attention over cells in the table is conditioned on a given vector q, similarly to the attention model for sequences. However, rather than a sequence of vectors, we now operate over a table.

Step 1: Attention over the attributes to find out the attributes that a user asks about, $p^a = \text{ATTN}(\{g_c\}, q)$. Suppose a user says `cheap`, then we should focus on the `price` attribute.

Step 2: Conditional row representation calculation, $e_r = \sum_c p^a_c e_{r,c} \quad \forall r$. So that e_r contains the price information of the restaurant in row r.

Step 3: Attention over e_r to find out the restaurants that satisfy users' query, $p^r = \text{ATTN}(\{e_r\}, q)$. Restaurants with `cheap` price will be picked.

Step 4: Using the probabilities p^r, we compute the weighted average over the all rows $e_c = \sum_r p^r_r e_{r,c}$. $\{e_r\}$ contains the information of `cheap` restaurant.

Step 5: Attention over columns $\{e_r\}$ to compute the probabilities of copying each column $p^c = \text{ATTN}(\{e_c\}, q)$.

Step 6: To get the probability matrix of copying each cell, we simply compute the outer product $p^{\text{copy}} = p^r \otimes p^c$.
The overall process is as follows:

$$\begin{aligned}
p^a &= \text{ATTN}(\{g_c\}, q), \\
e_r &= \sum_c p^a_c e_{r,c} \quad \forall r, \\
p^r &= \text{ATTN}(\{e_r\}, q), \\
e_c &= \sum_r p^r_r e_{r,c} \quad \forall c, \\
p^c &= \text{ATTN}(\{e_c\}, q), \\
p^{\text{copy}} &= p^r \otimes p^c.
\end{aligned}$$

If $z_{i,v} = 1$, we embed the above attention process in the decoder by replacing the conditioned state q with the current decoder state $s^y_{i,v}$.
Objective: As in previous task, we can train the model in a fully supervised fashion, or we can treat the decision as a latent variable. We can get $p(y_{i,v}|s_{i,v})$ in a similar way.

2.3 Reference to document context

Finally, we address the references that happen in a document itself and build a language model that uses coreference links to point to previous entities. Before generating a word, we first make the decision on whether it is an entity mention. If so, we decide which entity this mention belongs to, then we generate the word based on that entity. Denote the document as $X = \{x_i\}_{i=1}^L$, and the entities are $E = \{e_i\}_{i=1}^N$, each entity has M_i mentions, $e_i = \{m_{ij}\}_{j=1}^{M_i}$, such that $\{x_{m_{ij}}\}_{j=1}^{M_i}$ refer to the same entity. We use a LSTM to model the document, the hidden state of each token is $h_i = \text{LSTM}(W_E x_i, h_{i-1})$. We use a set $h^e = \{h^e_0, h^e_1, ..., h^e_M\}$ to keep track of the entity states, where h^e_j is the state of entity j.
Word generation: At each time step before generating the next word, we predict whether the word is an entity mention:

$$\begin{aligned}
p^{\text{coref}}(v_i|h_{i-1}, h^e) &= \text{ATTN}(h^e, h_{i-1}), \\
d_i &= \sum_{v_i} p(v_i) h^e_{v_i}, \\
p(z_i|h_{i-1}) &= \text{sigmoid}(W[h_{i-1}, d_i]),
\end{aligned}$$

where z_i denotes whether the next word is an entity and if yes v_i denotes which entity the next word corefers to. If the next word is an entity mention, then $p(x_i|v_i, h_{i-1}, h^e) =$

um and [I]$_1$ think that is whats - Go ahead [Linda]$_2$. Well and thanks goes to [you]$_1$ and to [the media]$_3$ to help [us]$_4$...So [our]$_4$ hat is off to all of [you]$_5$...

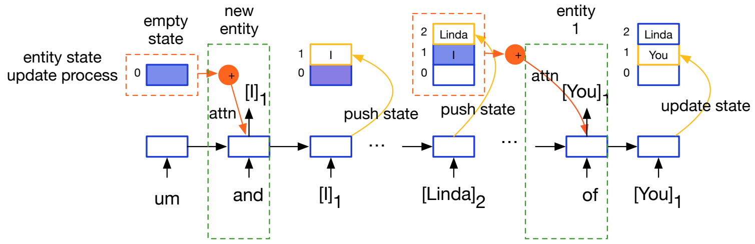


Figure 5: Coreference based language model, example taken from Wiseman et al. (2016).

softmax$(W_1 \tanh(W_2[h_{i-1}, h^e_{v_i}]))$ else $p(x_i|h_{i-1}) = \text{softmax}(W_1 h_{i-1})$. Hence,

$$p(x_i|x_{<i}) = \begin{cases} p(x_i|h_{i-1})p(z_i|h_{i-1}, h^e) & \text{if } z_i = 0. \\ p(x_i|v_i, h_{i-1}, h^e)\times & \\ \quad p^{\text{coref}}(v_i|h_{i-1}, h^e)p(z_i|h_{i-1}, h^e) & \text{if } z_i = 1. \end{cases}$$

Entity state update: Since there are multiple mentions for each entity and the mentions appear dynamically, we need to keep track of the entity state in order to use coreference information in entity mention prediction. We update the entity state h^e at each time step. In the beginning, $h^e = \{h^e_0\}$, h^e_0 denotes the state of an virtual empty entity and is a learnable variable. If $z_i = 1$ and $v_i = 0$, then it indicates the next word is a new entity mention, then in the next step, we append h_i to h^e, i.e., $h^e = \{h^e, h_i\}$, if $z_i = 1$ and $v_i > 0$, then we update the corresponding entity state with the new hidden state, $h^e[v_i] = h_i$. Another way to update the entity state is to use one LSTM to encode the mention states and get the new entity state. Here we use the latest entity mention state as the new entity state for simplicity. The detailed update process is shown in Figure 5.

Note that the stochastic decisions in this task are more complicated than previous two tasks. We need to make two sequential decisions: whether the next word is an entity mention, and if yes, which entity the mention corefers to. It is intractable to marginalize these decisions, so we train this model in a supervised fashion (refer to data preparation part on how we get coreference annotations).

3 Experiments

3.1 Data sets and preprocessing

Recipes: We crawled all recipes from `www.allrecipes.com`. There are about 31,000 recipes in total, and every recipe has an ingredient list and a corresponding recipe. We exclude the recipes that have less than 10 tokens or more than 500 tokens, those recipes take about 0.1% of all data set. On average each recipe has 118 tokens and 9 ingredients. We random shuffle the whole data set and take 80% as training and 10% for validation and test. We use a vocabulary size of 10,000 in the model.

Dialogue: We use the DSTC2 data set. We only use the dialogue transcripts from the data set. There are about 3,200 dialogues in total. The table is not available from DSTC2. To reconstruct the table, we crawled TripAdvisor for restaurants in the Cambridge area, where the dialog dataset was collected. Then, we remove restaurants that do not appear in the data set and create a database with 109 restaurants and their attributes (e.g. food type). Since this is a small data set, we use 5-fold cross validation and report the average result over the 5 partitions. There may be multiple tokens in each table cell, for example in Table. 2, the name, address, post code and phone number have multiple tokens, we replace them with one special token. For the name, address, post code and phone number of the j-th row, we replace the tokens in each cell with _NAME_j, _ADDR_j, _POSTCODE_j, _PHONE_j. If a table cell is empty, we replace it with an empty token _EMPTY. We do a string match in the transcript and replace the corresponding tokens in transcripts from the table with the special tokens. Each dialogue on average has 8 turns (16 sentences). We use a vocabulary size of 900, including about 400 table tokens and 500 words.

Coref LM: We use the Xinhua News data set from Gigaword Fifth Edition and sample 100,000 documents that has length in range from 100 to 500. Each document has on average 234 tokens, so there are 23 million tokens in total. We process the documents to get coreference annotations and use the annotations, i.e., z_i, v_i, in training. We take 80% as training and 10% as validation and test respectively. We ignore the entities that have only one mention and for the mentions that have multiple tokens, we take the token that is most frequent in the all the mentions for this entity. After preprocessing, tokens that are entity mentions take about 10% of all tokens. We use a vocabulary size of 50,000 in the model.

3.2 Baselines, model training and evaluation

We compare our model with baselines that do not model reference explicitly. For recipe generation and dialogue modeling, we compare our model with basic seq2seq and attention model. We also apply attention mechanism over the table for dialogue modeling as a baseline. For coreference based language model, we compare our model with simple RNN language model.

We train all models with simple stochastic gradient descent with gradient clipping. We use a one-layer LSTM for all RNN components. Hyperparameters are selected using grid search based on the validation set.

Evaluation of our model is challenging since it involves three rather different applications. We focus on evaluating the accuracy of predicting the reference tokens, which is the goal of our model. Specifically, we report the perplexity of all words, words that can be generated from reference and non-reference words. The perplexity is calculated by multiplying the probability of decision at each step all together. Note that for non-reference words, they also appear in the vocabulary. So it is a fair comparison to models that do not model reference explicitly. For the recipe task, we also generate the recipes using beam size of 10 and evaluate the generated recipes with BLEU. We didn't use BLEU for dialogue generation since the database entries take only a very small part of all tokens in utterances.

3.3 Results and analysis

The results for recipe generation, dialogue and coref based language model are shown in Table 4, 5, and 6 respectively. The recipe results in Table 4 verifies that modeling reference explicitly improves performance. Latent and Pointer perform better than Seq2Seq and Attn model. The Latent model performs better than the Pointer model since tokens in ingredients that match with recipes do not necessarily come from the ingredients. Imposing a supervised signal gives wrong information to the model and hence makes the result worse. With latent decision, the model learns to when to copy and when to generate it from the vocabulary.

The findings for dialogue basically follow that of recipe generation, as shown in Table 5. Conditioning table performs better in predicting table tokens in general. Table Pointer has the lowest perplexity for tokens in the table. Since the table tokens appear rarely in the dialogue transcripts, the overall perplexity does not differ much and the non-table token perplexity are similar. With attention mechanism over the table, the perplexity of table token improves over basic Seq2Seq model, but still not as good as directly pointing to cells in the table, which shows the advantage of modeling reference explicitly. As expected, using sentence attention improves significantly over models without sentence attention. Surprisingly, Table Latent performs much worse than Table Pointer. We also measure the perplexity of table tokens that appear only in test set. For models other than Table Pointer, because the tokens never appear in the training set, the perplexity is quite high, while Table Pointer can predict these tokens much more accurately. This verifies our conjecture that our model can learn reasoning over databases.

The coref based LM results are shown in Table 6. We find that coref based LM performs much better on the entity perplexity, but is a little bit worse for non-entity words. We found it was an optimization problem and the model was stuck in a local optimum. So we initialize the Pointer model with the weights learned from LM, the Pointer model performs better than LM both for entity perplexity and non-entity words perplexity.

In Appendix A, we also visualize the heat map of table reference and list items reference. The visualization shows that our model can correctly predict when to refer to which entries according to context.

4 Related Work

In terms of methodology, our work is closely related to previous works that incorporate copying mechanism with neural models (Gülçehre et al., 2016; Gu et al., 2016; Ling et al., 2016; Vinyals et al., 2015). Our models are similar to models proposed in (Ahn et al., 2016; Merity et al., 2016),

Model	val				test			
	PPL			BLEU	PPL			BLEU
	All	Ing	Word		All	Ing	Word	
Seq2Seq	5.60	11.26	**5.00**	14.07	5.52	11.26	**4.91**	14.39
Attn	5.25	6.86	5.03	14.84	5.19	6.92	4.95	15.15
Pointer	5.15	5.86	5.04	**15.06**	5.11	6.04	4.98	15.29
Latent	**5.02**	**5.10**	5.01	14.87	**4.97**	**5.19**	4.94	**15.41**

Table 4: Recipe results, evaluated in perplexity and BLEU score. All means all tokens, Ing denotes tokens from recipes that appear in ingredients. Word means non-table tokens. Pointer and Latent differs in that for Pointer, we provide supervised signal on when to generate a reference token, while in Latent it is a latent decision.

Model	All	Table	Table OOV	Word
Seq2Seq	1.35±0.01	4.98±0.38	1.99E7±7.75E6	1.23±0.01
Table Attn	1.37±0.01	5.09±0.64	7.91E7±1.39E8	1.24±0.01
Table Pointer	**1.33±0.01**	**3.99±0.36**	**1360 ± 2600**	**1.23±0.01**
Table Latent	1.36±0.01	4.99±0.20	3.78E7±6.08E7	1.24±0.01
+ Sentence Attn				
Seq2Seq	1.28±0.01	3.31±0.21	2.83E9 ± 4.69E9	**1.19±0.01**
Table Attn	1.28±0.01	3.17±0.21	1.67E7±9.5E6	1.20±0.01
Table Pointer	**1.27±0.01**	**2.99±0.19**	**82.86±110**	1.20±0.01
Table Latent	1.28±0.01	3.26±0.25	1.27E7±1.41E7	1.20±0.01

Table 5: Dialogue perplexity results. Table means tokens from table, Table OOV denotes table tokens that do not appear in the training set. **Sentence Attn** denotes we use attention mechanism over tokens in utterances from the previous turn.

Model	val			test		
	All	Entity	Word	All	Entity	Word
LM	33.08	44.52	32.04	33.08	43.86	32.10
Pointer	32.57	32.07	32.62	32.62	32.07	32.69
Pointer + init	**30.43**	**28.56**	**30.63**	**30.42**	**28.56**	**30.66**

Table 6: Coreference based LM. Pointer + init means we initialize the model with the LM weights.

where the generation of each word can be conditioned on a particular entry in knowledge lists and previous words. In our work, we describe a model with broader applications, allowing us to condition, on databases, lists and dynamic lists.

In terms of applications, our work is related to chit-chat dialogue (Li et al., 2016; Vinyals and Le, 2015; Sordoni et al., 2015; Serban et al., 2016; Shang et al., 2015) and task oriented dialogue (Wen et al., 2015; Bordes and Weston, 2016; Williams and Zweig, 2016; Wen et al., 2016). Most of previous works on task oriented dialogues embed the seq2seq model in traditional dialogue systems, in which the table query part is not differentiable, while our model queries the database directly. Recipe generation was proposed in (Kiddon et al., 2016). They use attention mechanism over the checklists, whereas our work models explicit references to them. Context dependent language models (Mikolov et al., 2010; Jozefowicz et al., 2016; Mikolov et al., 2010; Ji et al., 2015; Wang and Cho, 2015) are proposed to capture long term dependency of text. There are also lots of works on coreference resolution (Haghighi and Klein, 2010; Wiseman et al., 2016). We are the first to combine coreference with language modeling, to the best of our knowledge.

5 Conclusion

We introduce reference-aware language models which explicitly model the decision of from where to generate the token at each step. Our model can also learns the decision by treating it as a latent variable. We demonstrate on three applications, table based dialogue modeling, recipe generation and coref based LM, that our model performs better than attention based model, which does not incorporate this decision explicitly. There are several directions to explore further based on our framework. The current evaluation method is based on perplexity and BLEU. In task oriented dialogues, we can also try human evaluation to see if the model can reply users' query accurately. It is also interesting to use reinforcement learning to learn the actions in each step in coref based LM.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *CoRR*, abs/1608.00318.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *CoRR*, abs/1603.06393.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *CoRR*, abs/1603.08148.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 385–393. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Jason Williams. 2014. Dialog state tracking challenge 2 & 3.

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

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2015. Document context language models. *arXiv preprint arXiv:1511.03962*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proc. EMNLP*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. In *Proc. EMNLP*.

Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Andrew Senior, Fumin Wang, and Phil Blunsom. 2016. Latent predictor networks for code generation. In *Proc. ACL*.

Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 11–19.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Meg Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proc. NAACL*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proc. NIPS*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *Proc. ICML Deep Learning Workshop*.

Tian Wang and Kyunghyun Cho. 2015. Larger-context language modelling. *arXiv preprint arXiv:1511.03729*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-hao Su, David Vandyke, and Steve J. Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proc. EMNLP*.

Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269*.

Sam Wiseman, Alexander M Rush, and Stuart M Shieber. 2016. Learning global features for coreference resolution. *arXiv preprint arXiv:1604.03035*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

A Simple Language Model based on PMI Matrix Approximations

Oren Melamud
IBM Research
Yorktown Heights, NY, USA
oren.melamud@ibm.com

Ido Dagan
Computer Science Dept.
Bar-Ilan University, Israel
dagan@cs.biu.ac.il

Jacob Goldberger
Faculty of Engineering
Bar-Ilan University, Israel
jacob.goldberger@biu.ac.il

Abstract

In this study, we introduce a new approach for learning language models by training them to estimate word-context pointwise mutual information (PMI), and then deriving the desired conditional probabilities from PMI at test time. Specifically, we show that with minor modifications to *word2vec*'s algorithm, we get principled language models that are closely related to the well-established Noise Contrastive Estimation (NCE) based language models. A compelling aspect of our approach is that our models are trained with the same simple negative sampling objective function that is commonly used in *word2vec* to learn word embeddings.

1 Introduction

Language models (LMs) learn to estimate the probability of a word given a context of preceding words. Recurrent Neural Network (RNN) language models recently outperformed traditional n-gram LMs across a range of tasks (Jozefowicz et al., 2016). However, an important practical issue associated with such neural-network LMs is the high computational cost incurred. The key factor that limits the scalability of traditional neural LMs is the computation of the normalization term in the softmax output layer, whose cost is linearly proportional to the size of the word vocabulary.

Several methods have been proposed to cope with this scaling issue by replacing the softmax with a more computationally efficient component at train time.[1] These include importance sampling (Bengio and et al, 2003), hierarchical softmax (Minh and Hinton, 2008), BlackOut (Ji et al., 2016) and Noise Contrastive Estimation (NCE) (Gutmann and Hyvarinen, 2012). NCE has been applied to train neural LMs with large vocabularies (Mnih and Teh, 2012) and more recently was also successfully used to train LSTM-RNN LMs (Vaswani et al., 2013; Chen et al., 2015; Zoph et al., 2016). NCE-based language models achieved near state-of-the-art performance on language modeling tasks (Jozefowicz et al., 2016; Chen et al., 2016), and as we later show, are closely related to the method presented in this paper.

Continuous word embeddings were initially introduced as a 'by-product' of learning neural language models (Bengio and et al, 2003). However, they were later adopted in many other NLP tasks, and the most popular recent word embedding learning models are no longer proper language models. In particular, the skip-gram with negative sampling (NEG) embedding algorithm (Mikolov et al., 2013) as implemented in the *word2vec* toolkit, has become one of the most popular such models today. This is largely attributed to its scalability to huge volumes of data, which is critical for learning high-quality embeddings. Recently, Levy and Goldberg (2014) offered a motivation for the NEG objective function, showing that by maximizing this function, the skip-gram algorithm implicitly attempts to factorize a word-context pointwise mutual information (PMI) matrix. Melamud and Goldberger (2017) rederived this result by offering an information-theory interpretation of NEG.

The NEG objective function is considered a simplification of the NCE's objective, unsuitable for learning language models (Dyer, 2014). However, in this study, we show that despite its simplicity, it can be used in a principled way to effectively train a language model, based on PMI matrix factorization. More specifically, we use NEG to train

[1] An alternative recent approach for coping with large word vocabularies is to represent words as compositions of sub-word units, such as individual characters. This approach has notable merits (Jozefowicz et al., 2016; Sennrich et al., 2016), but is out of the scope of this paper.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1860–1865
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

a model for estimating the PMI between words and their preceding contexts, and then derive conditional probabilities from PMI at test time. The obtained *PMI-LM* can be viewed as a simple variant of *word2vec*'s algorithm, where the context of a predicted word is the preceding sequence of words, rather than a single word within a context window (skip-gram), or a bag-of-context-words (CBOW).

Our analysis shows that the proposed PMI-LM is very closely related to NCE language models (*NCE-LMs*). Similar to NCE-LMs, PMI-LM avoids the dependency of train run-time on the size of the word vocabulary by sampling from a negative (noise) distribution. Furthermore, conveniently, it also has a notably more simplified objective function formulation inherited from *word2vec*, which allows it to avoid the heuristic components and initialization procedures used in various implementations of NCE language models (Vaswani et al., 2013; Chen et al., 2015; Zoph et al., 2016).

Finally, we report on a perplexity evaluation of PMI and NCE language models on two standard language modeling datasets. The evaluation yielded comparable results, supporting our theoretical analysis.

2 NCE-based Language Modeling

Noise Contrastive Estimation (NCE) has recently been used to learn language models efficiently. NCE transforms the parameter learning problem into a binary classifier training problem. Let $p(w|c)$ be the probability of a word w given a context c that represents its entire preceding context, and let $p(w)$ be a 'noise' word distribution (e.g. a unigram distribution). The NCE approach assumes that the word w is sampled from a mixture distribution $\frac{1}{k+1}(p(w|c) + kp(w))$ such that the noise samples are k times more frequent than samples from the 'true' distribution $p(w|c)$. Let y be a binary random variable such that $y = 0$ and $y = 1$ correspond to a noise sample and a true sample, respectively, i.e. $p(w|c, y = 0) = p(w)$ and $p(w|c, y = 1) = p(w|c)$. Assume the distribution $p(w|c)$ has the following parametric form:

$$p_{nce}(w|c) = \frac{1}{Z_c} \exp(\vec{w} \cdot \vec{c} + b_w) \qquad (1)$$

such that $\vec{w}$ and $\vec{c}$ are vector representations of the word w and its context c. Applying Bayes rule, it can be easily verified that:

$$p_{nce}(y = 1|w, c) = \qquad (2)$$

$$\sigma(\vec{w} \cdot \vec{c} + b_w - \log Z_c - \log(p(w)k))$$

where $\sigma()$ is the sigmoid function.

NCE uses Eq. (2) and the following objective function to train a binary classifier that decides which distribution was used to sample w:

$$S_{nce} = \sum_{w,c \in D} \left[\log p(1|w, c) + \sum_{i=1}^{k} \log p(0|u_i, c) \right] \qquad (3)$$

such that w, c go over all the word-context co-occurrences in the learning corpus D and $u_1, ..., u_k$ are 'noise' samples drawn from the word unigram distribution.

Note that the normalization factor Z_c is not a free parameter and to obtain its value, one needs to compute $Z_c = \sum_{w \in V} \exp(\vec{w} \cdot \vec{c} + b_w)$ for each context c, where V is the word vocabulary. This computation is typically not feasible due to the large vocabulary size and the exponentially large number of possible contexts and therefore it was heuristically circumvented by prior work. Mnih and Teh (2012) found empirically that setting $Z_c = 1$ didn't hurt the performance (see also discussion in (Andreas and Klein, 2015)). Chen et al. (2015) reported that setting $\log(Z_c) = 9$ gave them the best results. Recent works (Vaswani et al., 2013; Zoph et al., 2016) used $Z_c = 1$ and also initialized NCE's bias term from Eq. (2) to $b_w = -\log |V|$. They reported that without these heuristics the training procedure did not converge to a meaningful model.

In the following section, we describe our proposed language model, which is derived from *word2vec*'s interpretation as a low-rank PMI matrix approximation. Interestingly, this model turns out to be a close variant of NCE language models, but with a simplified objective function that avoids the need for the normalization factor Z_c and the bias terms.

3 PMI-based Language Modeling

The skip-gram negative sampling word embedding algorithm represents each word w and each context word c as d-dimensional vectors, with the purpose that words that are "similar" to each other will have similar vector representations. The algorithm optimizes the following NEG objective function (Mikolov et al., 2013):

$$S_{neg} = \sum_{w,c \in D} \left[\log \sigma(\vec{w} \cdot \vec{c}) + \sum_{i=1}^{k} \log \sigma(-\vec{u}_i \cdot \vec{c}) \right] \tag{4}$$

such that w, c go over all the word-context co-occurrences in the learning corpus D, $u_1, ..., u_k$ are words independently sampled from the word unigram distribution, $\vec{x}$ is the embedding of x and $\sigma()$ is the sigmoid function. The objective function S_{neg} can be viewed as a log-likelihood function of a binary logistic regression classifier that treats a sample from a joint word-context distribution as a positive instance, and two independent samples from the word and context unigram distributions as a negative instance, while k is the proportion between negative and positive instances. Levy and Goldberg (2014) showed that this objective function achieves its maximal value when for every word-context pair w, c:

$$\vec{w} \cdot \vec{c} = \text{pmi}_k(w, c) = \log \frac{p(w|c)}{kp(w)} \tag{5}$$

where $\text{pmi}_k(w, c)$ is the word-context *PMI matrix*. Actually achieving this maximal value is typically infeasible, since the embedding dimensionality is intentionally limited. Therefore, learning word and context embeddings that optimize skip-gram's NEG objective function (4) can be viewed as finding a low-rank approximation of the word-context PMI matrix. An explicit expression of the approximation criterion optimized by the skip-gram algorithm can be found in (Melamud and Goldberger, 2017).

Our study is based on two simple observations regarding this finding of Levy and Goldberg (2014). First, Equation (5) can be reformulated as follows to derive an estimate of the conditional distribution $p(w|c)$:

$$\hat{p}(w|c) \propto \exp(\vec{w} \cdot \vec{c}) p(w) \tag{6}$$

where the constant k is dropped since $p(w|c)$ is a distribution. Second, while the above analysis had been originally applied to the case of word-context joint distributions $p(w, c)$, it is easy to see that the PMI matrix approximation analysis also holds for every Euclidean embedding of a joint distribution $p(x, y)$ of any two given random variables X and Y. In particular, we note that it holds for word-context joint distributions $p(w, c)$, where w is a single word, but c represents its entire preceding context, rather than just a single context word, and $\vec{c}$ is a vector representation of this entire context. Altogether, this allows us to use *word2vec*'s NEG objective function (4) to approximate the language modeling conditional probability $\hat{p}(w|c)$ (6), with c being the entire preceding context of the predicted word w.

We next describe the design details of the proposed PMI-based language modeling. We use a simple lookup table for the word representation $\vec{w}$, and an LSTM recurrent neural network to obtain a low dimensional representation of the entire preceding context $\vec{c}$. These representations are trained to maximize the NEG objective in Eq. (4), where this time w goes over every word token in the corpus, and c is its preceding context. We showed above that optimizing this objective seeks to obtain the best low-dimensional approximation of the PMI matrix associated with the joint distribution of the word and its preceding context (Eq. (5)). Hence, based on Eq. (6), for a reasonable embedding dimensionality and a good model for representing the preceding context, we expect $\hat{p}(w|c)$ to be a good estimate of the language modeling conditional distribution.

At test time, to obtain a proper distribution, we perform a normalization operation as done by all other comparable models. The train and test steps of the proposed language modeling algorithm are shown in algorithm box 1.

Note that while the NCE approach (1) learns to explicitly estimate normalized conditional distributions, our model learns to approximate the PMI matrix. Hence, we have no real motivation to include additional learned normalization parameters, as considered in comparable NCE language models (Mnih and Teh, 2012; Zoph et al., 2016).

The NEG and NCE objective functions share a similar form:

$$S = \sum_{w,c} \left[\log s(w, c) + \sum_{i=1}^{k} \log(1 - s(u_i, c)) \right] \tag{7}$$

with the differences summarized in Table 1. The comparison shows that PMI-LM's NEG objective function is much simpler. Furthermore, due to the component $\log(p(w)k))$ in NCE's objective function, its input to the sigmoid function is sensitive to the variable values in the unigram distribution, and

	Training objective function	Test probability estimate
NCE-LM	$s(w, c) = \sigma(\vec{w} \cdot \vec{c} + b_w - \log Z_c - \log(kp(w)))$	$\hat{p}(w\|c) \propto \exp(\vec{w} \cdot \vec{c} + b_w)$
PMI-LM	$s(w, c) = \sigma(\vec{w} \cdot \vec{c})$	$\hat{p}(w\|c) \propto \exp(\vec{w} \cdot \vec{c})p(w)$

Table 1: Comparison of the training objective functions (see Eq. (7)) and the respective test-time conditional word probability functions for NCE-LM and PMI-LM algorithms.

Algorithm 1 PMI Language Modeling

Training phase:

- Use a simple lookup table for the word representation and an LSTM recurrent neural network to obtain the preceding context representation.

- Train the word and preceding context embeddings to maximize the objective:

$$S_{neg} = \sum_{w,c \in D} \left[\log \sigma(\vec{w} \cdot \vec{c}) + \sum_{i=1}^{k} \log \sigma(-\vec{u}_i \cdot \vec{c}) \right]$$

such that w and c go over every word and it preceding context in the corpus D, and $u_1, ..., u_k$ are words independently sampled from the unigram distribution $p(w)$.

Test phase:

The conditional probability estimate for a word w given a preceding context c is:

$$\hat{p}(w|c) = \frac{\exp(\vec{w} \cdot \vec{c})p(w)}{\sum_{v \in V} \exp(\vec{v} \cdot \vec{c})p(v)}$$

where V is the word vocabulary.

therefore potentially more difficult to concentrate around zero with low variance to facilitate effective back-propagation. This may explain heuristics used by prior work for initializing the values of b_w (Vaswani et al., 2013; Zoph et al., 2016).

4 Experiments

The goal of the evaluation described in this section is to empirically establish PMI-LM as a sound language model. We do so by comparing its performance with the well-established NCE-LM, using the popular perplexity measure on two standard datasets, under the same terms. We describe our hyperparameter choices below and stress that for a fair comparison, we followed prior best practices and avoided hyperparameter optimization in favor of PMI-LM. All of the models described hereafter were implemented using the Chainer toolkit (Tokui et al., 2015).

For our NCE baseline, we used the heuristics that worked well in (Vaswani et al., 2013; Zoph et al., 2016), initializing NCE's bias term from Eq. (2) to $b_w = -\log |V|$, where V is the word vocabulary, and using $Z_c = 1$.

The first dataset we used is a version of the Penn Tree Bank (PTB), commonly used to evaluate language models.[2] It consists of 929K training words, 73K validation words and 82K test words with a 10K word vocabulary. To build and train the compared models in this setting, we followed the work of Zaremba et al. (2014), who achieved excellent results on this dataset. Specifically, we used a 2-layer 300-hidden-units LSTM with a 50% dropout ratio to represent the preceding (left-side) context of a predicted word.[3] We represented end-of-sentence as a special <eos> token and predicted this token like any other word. During training, we performed truncated back-propagation-through-time, unrolling the LSTM for 20 steps at a time without ever resetting the LSTM state. We trained our model for 39 epochs using Stochastic Gradient Descent (SGD) with a learning rate of 1, which is decreased by a factor of 1.2 after every epoch starting after epoch 6. We clipped the norms of the gradient to 5 and used a mini-batch size of 20. We set the negative sampling parameter to $k = 100$ following Zoph et al. (2016), who showed highly competitive performance with NCE LMs trained with this number of samples.

As the second dataset, we used the much larger WMT 1B-word benchmark introduced by Chelba et al. (2013). This dataset comprises about 0.8B training words and has a held-out set partitioned into 50 subsets. The test set is the first subset in the held-out, comprising 159K words, including the <eos> tokens. We used the second subset as the validation set with 165K words. The original vocabulary size of this dataset is 0.8M words after converting all

[2] Available from Tomas Mikolov at: `http://www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz`

[3] Zaremba et al. (2014) used larger models with more units and also applied dropout to the output of the top LSTM layer, which we did not.

	PMI-LM	NCE-LM
PTB	**98.35**	104.33
WMT	**65.84**	69.28

Table 2: Perplexity results on test sets.

words that occur less than 3 times in the corpus to an <unk> token. However, we followed previous works (Williams et al., 2015; Ji et al., 2016) and trimmed the vocabulary further down to the top 64K most frequent words in order to successfully fit a neural model to this data using reasonably modest compute resources. To build and train our models, we used a similar method to the one used with PTB, with the following differences. We used a single-layer 512-hidden-unit LSTM to represent the preceding context. We followed Jozefowicz et al. (2016), who found a 10% dropout rate to be sufficient for relatively small models fitted to this large training corpus. We trained our model for only one epoch using the Adam optimizer (Kingma and Ba, 2014) with default parameters, which we found to converge more quickly and effectively than SGD. We used a mini-batch size of 1000.

The perplexity results achieved by the compared models appear in Table 2. As can be seen, the performance of our PMI-LM is competitive, slightly outperforming the NCE-LM on both test sets. To put these numbers in a broader context, we note that state-of-the-art results on these datasets are notably better. For example, on the small PTB test set, Zaremba et al. (2014) achieved 78.4 perplexity with a larger LSTM model and using the more costly softmax component. On the larger WMT dataset, Jozefowicz et al. (2016) achieved 46.1 and 43.7 perplexity numbers using NCE and importance sampling respectively, and with much larger LSTM models trained over the full vocabulary, rather than our trimmed one. They also achieved 23.7 with an ensemble method, which is the best result on this dataset to date. Yet, as intended, we argue that our experimental results affirm the claim that PMI-LM is a sound language model on par with NCE-LM.

5 Conclusions

In this work, we have shown that *word2vec*'s negative sampling objective function, popularized in the context of learning word representations, can also be used to effectively learn parametric language models. These language models are closely related to NCE language models, but utilize a simpler, potentially more robust objective function. More generally, our theoretical analysis shows that any *word2vec* model trained with negative sampling can be used in a principled way to estimate the conditional distribution $p(w|c)$, by following our proposed procedure at test time.

Acknowledgments

This work is supported by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI).

References

J. Andreas and D. Klein. 2015. When and why are log-linear models self-normalizing? In *NAACL*.

Y. Bengio and J. Senecal et al. 2003. Quick training of probabilistic neural nets by importance sampling. In *AISTATS*.

C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, and T. Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

W. Chen, D. Grangier, and M. Auli. 2016. Strategies for training large vocabulary neural language models. *CoRR*, abs/1512.04906.

X. Chen, X. Liu, M. Gales, and P. C. Woodland. 2015. Recurrent neural network language model training with noise contrastive estimation for speech recognition. In *ICASSP*.

C. Dyer. 2014. Notes on noise contrastive estimation and negative sampling. *arXiv preprint arXiv:1410.8251*.

M. U. Gutmann and A. Hyvarinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361.

S. Ji, S. Vishwanathan, N. Satish, A. Nadathur, J. Michael, and P. Dubey. 2016. Blackout: Speeding up recurrent neural network language models with very large vocabularies. ICLR.

R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

O. Levy and Y. Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*.

O. Melamud and J. Goldberger. 2017. Information-theory interpretation of the skip-gram negative-sampling objective function. In *Proceedings of ACL*.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*.

A. Minh and G. E. Hinton. 2008. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*.

A. Mnih and Y. W. Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *ICML*.

R. Sennrich, B. Haddow, and A. Birch. 2016. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

S. Tokui, K. Oono, S. Hido, and J. Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Workshop on Machine Learning Systems (LearningSys) in The 29th Annual Conference on Neural Information Processing Systems*.

A. Vaswani, Y. Zhao, V. Fossum, and D. Chiang. 2013. Decoding with large-scale neural language models improves translation. In *EMNLP*.

W. Williams, N. Prasad, D. Mrva, T. Ash, and T. Robinson. 2015. Scaling recurrent neural network language models. ICASSP.

W. Zaremba, I. Sutskever, and O. Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

B. Zoph, A. Vaswani, J. May, and K. Knight. 2016. Simple, fast noise-contrastive estimation for large RNN vocabularies. In *NAACL*.

Syllable-aware Neural Language Models: A Failure to Beat Character-aware Ones

Zhenisbek Assylbekov
School of Science and Technology
Nazarbayev University
zhassylbekov@nu.edu.kz

Rustem Takhanov
School of Science and Technology
Nazarbayev University
rustem.takhanov@nu.edu.kz

Bagdat Myrzakhmetov
National Laboratory Astana
Nazarbayev University
bagdat.myrzakhmetov@nu.edu.kz

Jonathan N. Washington
Linguistics Department
Swarthmore College
jonathan.washington@swarthmore.edu

Abstract

Syllabification does not seem to improve word-level RNN language modeling quality when compared to character-based segmentation. However, our best syllable-aware language model, achieving performance comparable to the competitive character-aware model, has 18%–33% fewer parameters and is trained 1.2–2.2 times faster.

1 Introduction

Recent advances in neural language modeling (NLM) are connected with character-aware models (Kim et al., 2016; Ling et al., 2015b; Verwimp et al., 2017). This is a promising approach, and we propose the following direction related to it: We would like to make sure that in the pursuit of the most fine-grained representations one has not missed possible intermediate ways of segmentation, e.g., by syllables. Syllables, in our opinion, are better supported as linguistic units of language than single characters. In most languages, words can be naturally split into syllables:

ES: el par-la-men-to a-po-yó la en-mien-da
RU: пар-ла-мент под-дер-жал по-прав-ку
(EN: the parliament supported the amendment)

Based on this observation, we attempted to determine whether syllable-aware NLM has any advantages over character-aware NLM. We experimented with a variety of models but could not find any evidence to support this hypothesis: splitting words into syllables does not seem to improve the language modeling quality when compared to splitting into characters. However, there are some positive findings: while our best syllable-aware language model achieves performance comparable to the competitive character-aware model, it has 18%–33% fewer parameters and is 1.2–2.2 times faster to train.

2 Related Work

Much research has been done on subword-level and subword-aware[1] neural language modeling when subwords are characters (Ling et al., 2015b; Kim et al., 2016; Verwimp et al., 2017) or morphemes (Botha and Blunsom, 2014; Qiu et al., 2014; Cotterell and Schütze, 2015). However, not much work has been done on syllable-level or syllable-aware NLM. Mikolov et al. (2012) show that subword-level language models outperform character-level ones.[2] They keep the most frequent words untouched and split all other words into syllable-like units. Our approach differs mainly in the following aspects: we make predictions at the word level, use a more linguistically sound syllabification algorithm, and consider a variety of more advanced neural architectures.

We have recently come across a concurrent paper (Vania and Lopez, 2017) where the authors systematically compare different subword units (characters, character trigrams, BPE (Sennrich et al., 2016), morphemes) and different representation models (CNN, Bi-LSTM, summation) on languages with various morphological typology. However, they do not consider syllables, and they experiment with relatively small models on small data sets (0.6M–1.4M tokens).

3 Syllable-aware word embeddings

Let $\mathcal{W}$ and $\mathcal{S}$ be finite vocabularies of words and syllables respectively. We assume that both words

[1] *Subword-level* LMs rely on subword-level inputs and make predictions at the level of subwords; *subword-aware* LMs also rely on subword-level inputs but make predictions at the level of words.

[2] Not to be confused with character-aware ones, see the previous footnote.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1866–1872
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

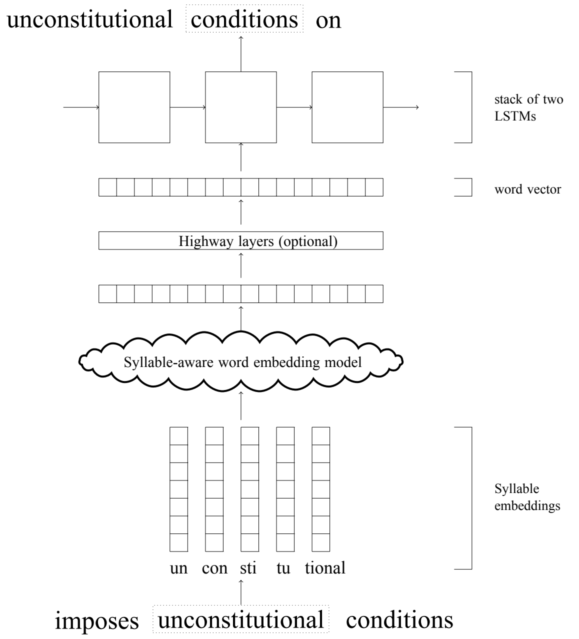


Figure 1: Syllable-aware language model.

and syllables have already been converted into indices. Let $\mathbf{E}_{\mathcal{S}} \in \mathbb{R}^{|\mathcal{S}| \times d_{\mathcal{S}}}$ be an embedding matrix for syllables — i.e., it is a matrix in which the sth row (denoted as $\mathbf{s}$) corresponds to an embedding of the syllable $s \in \mathcal{S}$. Any word $w \in \mathcal{W}$ is a sequence of its syllables $(s_1, s_2, \ldots, s_{n_w})$, and hence can be represented as a sequence of the corresponding syllable vectors:

$$[\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_{n_w}}]. \tag{1}$$

The question is: How shall we pack the sequence (1) into a single vector $\mathbf{x} \in \mathbb{R}^{d_W}$ to produce a better embedding of the word w?[3] In our case "better" means "better than a character-aware embedding of w via the Char-CNN model of Kim et al. (2016)". Below we present several viable approaches.

3.1 Recurrent sequential model (Syl-LSTM)

Since the syllables are coming in a sequence it is natural to try a recurrent sequential model:

$$\mathbf{h_t} = f(\mathbf{s_t}, \mathbf{h_{t-1}}), \qquad \mathbf{h_0} = \mathbf{0}, \tag{2}$$

which converts the sequence of syllable vectors (1) into a sequence of state vectors $\mathbf{h_{1:n_w}}$. The last state vector $\mathbf{h_{n_w}}$ is assumed to contain the information on the whole sequence (1), and is therefore used as a word embedding for w. There is a big variety of transformations from which one can choose f in (2); however, a recent thorough evaluation (Jozefowicz et al., 2015) shows that the LSTM (Hochreiter and Schmidhuber, 1997) with its forget bias initialized to 1 outperforms other popular architectures on almost all tasks, and we decided to use it for our experiments. We will refer to this model as *Syl-LSTM*.

3.2 Convolutional model (Syl-CNN)

Inspired by recent work on character-aware neural language models (Kim et al., 2016) we decided to try this approach (Char-CNN) on syllables. Our case differs mainly in the following two aspects:

1. The set of syllables $\mathcal{S}$ is usually bigger than the set of characters $\mathcal{C}$,[4] and also the dimensionality $d_{\mathcal{S}}$ of syllable vectors is expected to be greater than the dimensionality $d_{\mathcal{C}}$ of character vectors. Both of these factors result in allocating more parameters on syllable embeddings compared to character embeddings.
2. On average a word contains fewer syllables than characters, and therefore we need narrower convolutional filters for syllables. This results in spending fewer parameters per convolution.

This means that by varying $d_{\mathcal{S}}$ and the maximum width of convolutional filters L we can still fit the parameter budget of Kim et al. (2016) to allow fair comparison of the models.

Like in Char-CNN, our syllable-aware model, which is referred to as *Syl-CNN-[L]*, utilizes max-pooling and highway layers (Srivastava et al., 2015) to model interactions between the syllables. The dimensionality of a highway layer is denoted by d_{HW}.

3.3 Linear combinations

We also considered using linear combinations of syllable-vectors to represent the word embedding:

$$\mathbf{x} = \textstyle\sum_{t=1}^{n_w} \alpha_t(s_t) \cdot \mathbf{s_t}. \tag{3}$$

The choice for α_t is motivated mainly by the existing approaches (discussed below) which proved to be successful for other tasks.

Syl-Sum: Summing up syllable vectors to get a word vector can be obtained by setting $\alpha_t(s_t) = 1$. This approach was used by Botha and Blunsom (2014) to combine a word and its morpheme embeddings into a single word vector.

[3] The same question applies to any model that segments words into a sequence of characters or other subword units.

[4] In languages with alphabetic writing systems.

Syl-Avg: A simple average of syllable vectors can be obtained by setting $\alpha_t(s_t) = 1/n_w$. This can be also called a "continuous bag of syllables" in an analogy to a CBOW model (Mikolov et al., 2013), where vectors of neighboring words are averaged to get a word embedding of the current word.

Syl-Avg-A: We let the weights α_t in (3) be a function of parameters $(a_1, \dots, a_n)$ of the model, which are jointly trained together with other parameters. Here $n = \max_w\{n_w\}$ is a maximum word length in syllables. In order to have a weighted average in (3) we apply a softmax normalization:

$$\alpha_t = \text{softmax}(\mathbf{a})_t = \frac{\exp(a_t)}{\sum_{\tau=1}^{n}\exp(a_\tau)} \qquad (4)$$

Syl-Avg-B: We can let α_t depend on syllables and their positions:

$$\alpha_t = \alpha_t(s_t) = \text{softmax}(\mathbf{a}_{s_t} + \mathbf{b})_t$$

where $\mathbf{A} \in \mathbb{R}^{d_S \times n}$ (with elements $a_{s,t}$) is a set of parameters that determine the importance of each syllable type in each (relative) position, $\mathbf{b} \in \mathbb{R}^n$ is a bias, which is conditioned only on the relative position. This approach is motivated by recent work on using an attention mechanism in the CBOW model (Ling et al., 2015a).

We feed the resulting $\mathbf{x}$ from (3) into a stack of highway layers to allow interactions between the syllables.

3.4 Concatenation (Syl-Concat)

In this model we simply concatenate syllable vectors (1) into a single word vector:

$$\mathbf{x} = [\mathbf{s_1}; \mathbf{s_2}; \dots; \mathbf{s_{n_w}}; \underbrace{\mathbf{0}; \mathbf{0}; \dots; \mathbf{0}}_{n-n_w}]$$

We zero-pad $\mathbf{x}$ so that all word vectors have the same length $n \cdot d_S$ to allow batch processing, and then we feed $\mathbf{x}$ into a stack of highway layers.

4 Word-level language model

Once we have word embeddings $\mathbf{x_{1:k}}$ for a sequence of words $w_{1:k}$ we can use a word-level RNN language model to produce a sequence of states $\mathbf{h_{1:k}}$ and then predict the next word according to the probability distribution

$$\Pr(w_{k+1}|w_{1:k}) = \text{softmax}(\mathbf{h_k W} + \mathbf{b}),$$

where $\mathbf{W} \in \mathbb{R}^{d_{\text{LM}} \times |\mathcal{W}|}$, $\mathbf{b} \in \mathbb{R}^{|\mathcal{W}|}$, and d_{LM} is the hidden layer size of the RNN. Training the model involves minimizing the negative log-likelihood over the corpus $w_{1:K}$:

$$-\sum_{k=1}^{K} \log \Pr(w_k|w_{1:k-1}) \longrightarrow \min \qquad (5)$$

As was mentioned in Section 3.1 there is a huge variety of RNN architectures to choose from. The most advanced recurrent neural architectures, at the time of this writing, are recurrent highway networks (Zilly et al., 2017) and a novel model which was obtained through a neural architecture search with reinforcement learning (Zoph and Le, 2017). These models can be spiced up with the most recent regularization techniques for RNNs (Gal and Ghahramani, 2016) to reach state-of-the-art. However, to make our results directly comparable to those of Kim et al. (2016) we select a two-layer LSTM and regularize it as in Zaremba et al. (2014).

5 Experimental Setup

We search for the best model in two steps: first, we block the word-level LSTM's architecture and pre-select the three best models under a small parameter budget (5M), and then we tune these three best models' hyperparameters under a larger budget (20M).

Pre-selection: We fix d_{LM} (hidden layer size of the word-level LSTM) at 300 units per layer and run each syllable-aware word embedding method from Section 3 on the English PTB data set (Marcus et al., 1993), keeping the total parameter budget at 5M. The architectural choices are specified in Appendix A.

Hyperparameter tuning: The hyperparameters of the three best-performing models from the pre-selection step are then thoroughly tuned on the same English PTB data through a random search according to the marginal distributions:

- $d_S \sim U(20, 650)$,[5]
- $\log(d_{\text{HW}}) \sim U(\log(160), \log(2000))$,
- $\log(d_{\text{LM}}) \sim U(\log(300), \log(2000))$,

with the restriction $d_S < d_{\text{LM}}$. The total parameter budget is kept at 20M to allow for easy comparison to the results of Kim et al. (2016). Then these three best models (with their hyperparameters tuned on PTB) are trained and evaluated on small- (DATA-S) and medium-sized (DATA-L) data sets in six languages.

Optimizaton is performed in almost the same way as in the work of Zaremba et al. (2014). See Appendix B for details.

[5] $U(a, b)$ stands for a uniform distribution over (a, b).

Model	PPL	Model	PPL
LSTM-Word	88.0	Char-CNN	92.3
Syl-LSTM	88.7	Syl-Avg	88.5
Syl-CNN-2	86.6	Syl-Avg-A	91.4
Syl-CNN-3	**84.6**	Syl-Avg-B	88.5
Syl-CNN-4	86.8	Syl-Concat	**83.7**
Syl-Sum	**84.6**		

Table 1: Pre-selection results. PPL stands for test set perplexity, all models have $\approx$ 5M parameters.

Model	$d_{\mathcal{S}}$	d_{HW}	d_{LM}	Size	PPL
Syl-CNN	242	1170	380	15M	80.5
Syl-Sum	438	1256	435	18M	80.3
Syl-Concat	228	781	439	13M	**79.4**

Table 2: Hyperparameters tuning. In Syl-CNN, d_{HW} is a function of the primary hyperparameter $c = 195$ (see Appendix A).

Syllabification: The true syllabification of a word requires its grapheme-to-phoneme conversion and then splitting it into syllables based on some rules. Since these are not always available for less-resourced languages, we decided to utilize Liang's widely-used hyphenation algorithm (Liang, 1983).

6 Results

The results of the pre-selection are reported in Table 1. All syllable-aware models comfortably outperform the Char-CNN when the budget is limited to 5M parameters. Surprisingly, a pure word-level model,[6] LSTM-Word, also beats the character-aware one under such budget. The three best configurations are Syl-Concat, Syl-Sum, and Syl-CNN-3 (hereinafter referred to as Syl-CNN), and tuning their hyperparameters under 20M parameter budget gives the architectures in Table 2. The results of evaluating these three models on small (1M tokens) and medium-sized (17M–57M tokens) data sets against Char-CNN for different languages are provided in Table 3. The models demonstrate similar performance on small data, but Char-CNN scales significantly better on medium-sized data. From the three syllable-aware models, Syl-Concat looks the most advantageous as it demonstrates stable results and has the least number of parameters. Therefore in what follows we will make a more detailed comparison of Syl-Concat with Char-CNN.

Model	EN	FR	ES	DE	CS	RU	
Char-CNN	**78.9**	**184**	**165**	**239**	**371**	**261**	DATA-S
Syl-CNN	80.5	191	172	239	374	269	
Syl-Sum	80.3	193	170	243	389	273	
Syl-Concat	79.4	188	168	244	383	265	
Char-CNN	**160**	**124**	**118**	**198**	**392**	**190**	DATA-L
Syl-CNN[7]	–	–	–	–	–	–	
Syl-Sum	170	141	129	212	451	233	
Syl-Concat	176	139	129	225	449	225	

Table 3: Evaluation of the syllable-aware models against Char-CNN. In each case the smallest model, Syl-Concat, has 18%–33% less parameters than Char-CNN and is trained 1.2–2.2 times faster (Appendix C).

Shared errors: It is interesting to see whether Char-CNN and Syl-Concat are making similar errors. We say that a model gives an error if it assigns a probability less than p^* to a correct word from the test set. Figure 2 shows the percentage of errors which are shared by Syl-Concat and Char-CNN depending on the value of p^*. We see that the vast majority of errors are shared by both models even when p^* is small (0.01).

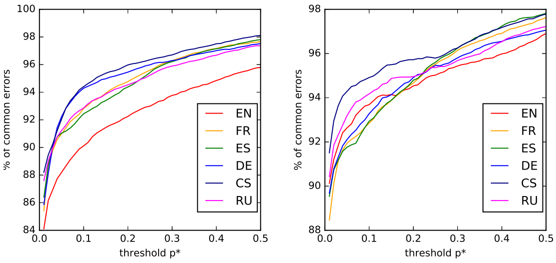


Figure 2: Percentage of errors shared by both Syl-Concat and Char-CNN on DATA-S (left) and DATA-L (right).

PPL breakdown by token frequency: To find out *how* Char-CNN outperforms Syl-Concat, we partition the test sets on token frequency, as computed on the training data. We can observe in Figure 3 that, on average, the more frequent the word is, the bigger the advantage of Char-CNN over Syl-Concat. The more Char-CNN sees a word in different contexts, the more it can learn about this word (due to its powerful CNN filters). Syl-Concat, on the other hand, has limitations – it cannot see below syllables, which prevents it from extracting the same amount of knowledge about the word.

PCA of word embeddings: The intrinsic advantage of Char-CNN over Syl-Concat is also sup-

[6] When words are directly embedded into $\mathbb{R}^{d_{\mathcal{W}}}$ through an embedding matrix $\mathbf{E}_{\mathcal{W}} \in \mathbb{R}^{|\mathcal{W}| \times d_{\mathcal{W}}}$.

[7] Syl-CNN results on DATA-L are not reported since computational resources were insufficient to run these configurations.

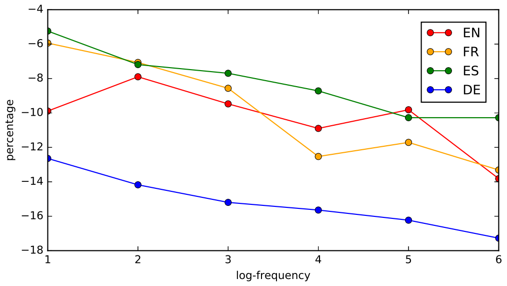


Figure 3: PPL reduction by token frequency, Char-CNN relative to Syl-Concat on DATA-L.

Model	80%	90%	95%	99%
Char-CNN	568	762	893	1038
Syl-Concat	515	729	875	1035

Table 4: Number of principle components when PCA is applied to word embeddings produced by each model, depending on % of variance to retain.

ported by the following experiment: We took word embeddings produced by both models on the English PTB, and applied PCA to them.[8] Regardless of the threshold percentage of variance to retain, the embeddings from Char-CNN always have more principal components than the embeddings from Syl-Concat (see Table 4). This means that Char-CNN embeds words into higher dimensional space than Syl-Concat, and thus can better distinguish them in different contexts.

LSTM limitations: During the hyperparameters tuning we noticed that increasing $d_{\mathcal{S}}$, d_{HW} and d_{LM} from the optimal values (in Table 2) did not result in better performance for Syl-Concat. Could it be due to the limitations of the word-level LSTM (the topmost layer in Fig. 1)? To find out whether this was the case we replaced the LSTM by a Variational RHN (Zilly et al., 2017), and that resulted in a significant reduction of perplexities on PTB for both Char-CNN and Syl-Concat (Table 5). Moreover, increasing d_{LM} from 439 to 650 did result in better performance for Syl-Concat. Optimization details are given in Appendix B.

Comparing syllable and morpheme embeddings: It is interesting to compare morphemes and syllables. We trained Morfessor 2.0 (Creutz and Lagus, 2007) in its default configuration on the PTB training data and used it instead of the syllabifier in our models. Interestingly, we got ≈3K unique morphemes, whereas the number of unique syllables was ≈6K. We then trained all our models on PTB under 5M parameter budget, keeping the state size of the word-level LSTM at 300 (as in our pre-selection step for syllable-aware models). The reduction in number of subword types allowed us to give them higher dimensionality $d_{\mathcal{M}} = 100$ (cf. $d_{\mathcal{S}} = 50$).[9]

Model	depth	d_{LM}	Size	PPL
RHN-Char-CNN	8	650	20M	67.6
RHN-Syl-Concat	8	439	13M	72.0
RHN-Syl-Concat	8	650	20M	69.4

Table 5: Replacing LSTM with Variational RHN.

Convolutional (Morph-CNN-3) and additive (Morph-Sum) models performed better than others with test set PPLs 83.0 and 83.9 respectively. Due to limited amount of time, we did not perform a thorough hyperparameter search under 20M budget. Instead, we ran two configurations for Morph-CNN-3 and two configurations for Morph-Sum with hyperparameters close to those, which were optimal for Syl-CNN-3 and Syl-Sum correspondingly. All told, our best morpheme-aware model is Morph-Sum with $d_{\mathcal{M}} = 550$, $d_{\text{HW}} = 1100$, $d_{\text{LM}} = 550$, and test set PPL 79.5, which is practically the same as the result of our best syllable-aware model Syl-Concat (79.4). This makes Morph-Sum a notable alternative to Char-CNN and Syl-Concat, and we defer its thorough study to future work.

Source code: The source code for the models discussed in this paper is available at `https://github.com/zh3nis/lstm-syl`.

7 Conclusion

It seems that syllable-aware language models fail to outperform competitive character-aware ones. However, usage of syllabification can reduce the total number of parameters and increase the training speed, albeit at the expense of language-dependent preprocessing. Morphological segmentation is a noteworthy alternative to syllabification: a simple morpheme-aware model which sums morpheme embeddings looks promising, and its study is deferred to future work.

[8] We equalized highway layer sizes d_{HW} in both models to have same dimensions for embeddings. In both cases, word vectors were standardized using the z-score transformation.

[9] $\mathcal{M}$ stands for morphemes.

A Pre-selection

In all models with highway layers there are two of them and the non-linear activation of any highway layer is a ReLU.

LSTM-Word: $d_{\mathcal{W}} = 108$, $d_{\text{LM}} = 300$.

Syl-LSTM: $d_{\mathcal{S}} = 50$, $d_{\text{LM}} = 300$.

Syl-CNN-[L]: $d_{\mathcal{S}} = 50$, convolutional filter widths are $[1, \ldots, L]$, the corresponding convolutional filter depths are $[c \cdot l]_{l=1}^{L}$, $d_{\text{HW}} = c \cdot (1 + \ldots + L)$. We experimented with $L = 2, 3, 4$. The corresponding values of c are chosen to be $120, 60, 35$ to fit the total parameter budget. CNN activation is tanh.

Linear combinations: We give higher dimensionality to syllable vectors here (compared to other models) since the resulting word vector will have the same size as syllable vectors (see (3)). $d_{\mathcal{S}} = 175$, $d_{\text{HW}} = 175$ in all models except the Syl-Avg-B, where we have $d_{\mathcal{S}} = 160$, $d_{\text{HW}} = 160$.

Syl-Concat: $d_{\mathcal{S}} = 50$, $d_{\text{HW}} = 300$.

B Optimization

LSTM-based models: We perform the training (5) by truncated BPTT (Werbos, 1990; Graves, 2013). We backpropagate for 70 time steps on DATA-S and for 35 time steps on DATA-L using stochastic gradient descent where the learning rate is initially set to 1.0 and halved if the perplexity does not decrease on the validation set after an epoch. We use batch sizes of 20 for DATA-S and 100 for DATA-L. We train for 50 epochs on DATA-S and for 25 epochs on DATA-L, picking the best-performing model on the validation set. Parameters of the models are randomly initialized uniformly in $[-0.05, 0.05]$, except the forget bias of the word-level LSTM, which is initialized to 1. For regularization we use dropout (Srivastava et al., 2014) with probability 0.5 between word-level LSTM layers and on the hidden-to-output softmax layer. We clip the norm of the gradients (normalized by minibatch size) at 5. These choices were guided by previous work on word-level language modeling with LSTMs (Zaremba et al., 2014).

To speed up training on DATA-L we use a sampled softmax (Jean et al., 2015) with the number of samples equal to 20% of the vocabulary size (Chen et al., 2016). Although Kim et al. (2016) used a hierarchical softmax (Morin and Bengio, 2005) for the same purpose, a recent study (Grave et al., 2016) shows that it is outperformed by sampled softmax on the Europarl corpus, from which DATA-L was derived (Botha and Blunsom, 2014).

RHN-based models are optimized as in Zilly et al. (2017), except that we unrolled the networks for 70 time steps in truncated BPTT, and dropout rates were chosen to be as follows: 0.2 for the embedding layer, 0.7 for the input to the gates, 0.7 for the hidden units and 0.2 for the output activations.

C Sizes and speeds

On DATA-S, Syl-Concat has 28%–33% fewer parameters than Char-CNN, and on DATA-L the reduction is 18%–27% (see Fig. 4).

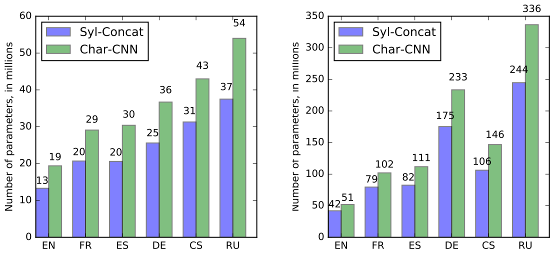


Figure 4: Model sizes on DATA-S (left) and DATA-L, in millions of trainable variables.

Training speeds are provided in the Table 6. Models were implemented in TensorFlow, and were run on NVIDIA Titan X (Pascal).

Model	EN	FR	ES	DE	CS	RU	
Char-CNN	9	8	8	7	6	6	S
Syl-Concat	14	12	12	11	10	9	
Char-CNN	10	8	7	5	7	4	L
Syl-Concat	22	13	13	6	10	5	

Table 6: Training speeds, in thousands of tokens per second.

Acknowledgements

We gratefully acknowledge the NVIDIA Corporation for their donation of the Titan X Pascal GPU used for this research. The work of Bagdat Myrzakhmetov has been funded by the Committee of Science of the Ministry of Education and Science of the Republic of Kazakhstan under the targeted program O.0743 (0115PK02473). The authors would like to thank anonymous reviewers and Aibek Makazhanov for valuable feedback, Makat Tlebaliyev and Dmitriy Polynin for IT support, and Yoon Kim for providing the preprocessed datasets.

References

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of ICML*.

Wenlin Chen, David Grangier, and Michael Auli. 2016. Strategies for training large vocabulary neural language models. In *Proceedings of ACL*.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proceedings of HLT-NAACL*.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1):3.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of NIPS*.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2016. Efficient softmax approximation for gpus. *arXiv preprint arXiv:1609.04309*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

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

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of ACL-IJCNLP*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of ICML*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of AAAI*.

Franklin Mark Liang. 1983. *Word Hy-phen-a-tion by Com-put-er*. Citeseer.

Wang Ling, Lin Chu-Cheng, Yulia Tsvetkov, and Silvio Amir. 2015a. Not all contexts are created equal: Better word representations with variable attention. In *Proceedings of EMNLP*.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015b. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of EMNLP*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocky. 2012. Subword language modeling with neural networks. *preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf)*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of AISTATS*.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of COLING*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proceedings of NIPS*.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of ACL*.

Lyan Verwimp, Joris Pelemans, Patrick Wambacq, et al. 2017. Character-word lstm language models. In *Proceedings of EACL*.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10).

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutník, and Jürgen Schmidhuber. 2017. Recurrent highway networks. In *Proceedings of ICML*.

Barret Zoph and Quoc V Le. 2017. Neural architecture search with reinforcement learning. In *Proceedings of ICLR*.

Inducing Semantic Micro-Clusters from Deep Multi-View Representations of Novels

Lea Frermann
University of Edinburgh [*]
l.frermann@ed.ac.uk

György Szarvas
Amazon Development Center Germany GmbH
szarvasg@amazon.de

Abstract

Automatically understanding the plot of novels is important both for informing literary scholarship and applications such as summarization or recommendation. Various models have addressed this task, but their evaluation has remained largely intrinsic and qualitative. Here, we propose a principled and scalable framework leveraging expert-provided semantic tags (e.g., *mystery*, *pirates*) to evaluate plot representations in an extrinsic fashion, assessing their ability to produce locally coherent groupings of novels (*micro-clusters*) in model space. We present a deep recurrent autoencoder model that learns richly structured *multi-view* plot representations, and show that they i) yield better micro-clusters than less structured representations; and ii) are interpretable, and thus useful for further literary analysis or labelling of the emerging micro-clusters.

1 Introduction

For the literature aficionado, the quest for the next novel to read can be daunting: the sheer number of novels of different styles, topics and genres is difficult to navigate. It is intuitively clear that readers select novels based on specific but potentially diverse and structured preferences (e.g., they might prefer novels of a particular theme (*small-town romance*), mood (*dark*) or based on character types (*grumpy boss*), character relations (*love, enmity*) and their development). These preferences also manifest in the organization of online book stores or recommendation platforms.[1] For example, the Amazon book catalog contains semantic tags provided by experts (publishers), including labels of character types (*pirates*) or theme (*secret baby romance*) to aid focused search for novels of interest.

Although these tags are already fairly granular, many cover large sets of novels (e.g., the tag *secret baby romance* covers almost $4,000$ novels), limiting their utility for exhaustive exploration and call for even finer grained micro-groupings. Can we instead *automatically* induce fine-grained novel clusters in an unsupervised, data-driven way?

We propose a framework to learn structured, interpretable book representations that capture different aspects of the plot, and verify that such representations are rich enough to support downstream tasks like generating interpretable book groupings. A real-world application of this work is content-based book recommendation based on diverse and interpretable book characteristics. Content-based recommendation has been criticized by the limited complexity of typically employed features (*limited content analysis*; Lops et al. (2011); Adomavicius and Tuzhilin (2005)). This work addresses this problem by inducing complex, structured and interpretable representations. Our contributions are two-fold.

First, assuming that richly *structured* book tags call for rich content representations (which expert taggers arguably possess), we describe a deep unsupervised model for learning *multi-view* representations of novel plots. We use the term *view* to refer to specific types of plot characteristics (e.g., pertaining to events, characters or mood), and *multi-view* to refer to combinations of these views. We use multi-view book representations to construct meaningful and locally coherent neighbourhoods in model space, which we will refer to as *micro-clusters*. To this end, we extend a recent autoencoder model (Iyyer et al., 2016) to learn multi-view representations of books. Our model

[*] Work done while the first author was an intern at Amazon (ADC Germany GmbH, Berlin).

[1] E.g., www.amazon.com or www.goodreads.com

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1873–1883
Copenhagen, Denmark, September 7–11, 2017.

encodes properties of characters (view v_1), relations between characters (view v_2), and their respective trajectories over the plot.[2] View-specific encodings are learnt in an unsupervised way from raw text as separate sets of word clusters which are jointly optimized to encode `relevant` and `distinct` information. These properties are crucial for applications such as book recommendation, because they allow to i) explain why particular books are similar based on the inferred latent structure and ii) find similarities based on important and distinct aspects of a novel (character types or interactions). Our framework of unsupervised multi-view learning is very flexible and can straightforwardly be applied to learn arbitrary kinds and numbers of views from raw text.

Secondly, we propose an empirical evaluation framework. Before we can use models to *extend* existing categories as discussed above, it must be shown that the representations capture *existing* associations. To this end, we investigate whether micro-clusters derived from induced representations resemble reference clusters defined as groups of novels sharing tags in the Amazon catalog. While automatic induction of plot representations has attracted considerable attention (see Jockers (2013)), evaluation has remained largely qualitative and intrinsic. To the best of our knowledge, we are the first to investigate the utility of automatically induced plot representations on an extrinsic task at scale. We evaluate micro-clusters as local neighbourhoods in model space containing $10,000$ novels under 50 reference tags.

We show that rich multi-view representations produce better micro-clusters compared to competitive but simpler models, and that interpretability of the learnt representations is not compromised despite the more complex objective. We also qualitatively demonstrate that high-quality micro-clusters emerge from a smaller, more homogeneous data set of Gutenberg[3] novels.

2 Related Work

Automatically learning representations of book plots, as structured summaries of their content, has attracted much attention (cf, Jockers (2013) for a review). Unsupervised models have been proposed which, given raw text, extract prototypical event structure (McIntyre and Lapata, 2010; Chambers and Jurafsky, 2009), prototypical characters (Bamman et al., 2013, 2014; Elsner, 2012) and their social networks (Elson et al., 2010).

Other work focused on the *dynamics* of a plot, learning trajectories of relations between two characters (Iyyer et al., 2016; Chaturvedi et al., 2017). Iyyer et al. (2016) combine dictionary learning (Olshausen and Field, 1997) with deep recurrent autoencoders to learn interpretable character relationship descriptors. They show that their deep model learns better representations than conceptually similar topic models (Gruber et al., 2007; Chang et al., 2009). Here, we extend the model of Iyyer et al. (2016) to simultaneously induce multiple views on the plot.

Methodologically, our work falls into the class of multi-view learning, and we propose a novel formulation of the model objective which encourages encoding of *distinct* information in the views. Our objective function is inspired by prior work in multi-task learning and deep domain adaptation for classification (Ganin and Lempitsky, 2015; Ganin et al., 2016). They train neural networks to simultaneously learn classifiers which are accurate on their target task and are agnostic about feature fluctuation pertaining to domain shift. We adapt this idea to unsupervised models with a reconstruction objective and learn multi-view representations which efficiently encode the input data and, at the same time, learn to *only* encode information relevant for the particular view.

Evaluating induced plot representations is notoriously difficult. Most evaluation has resorted to manual inspection, or crowd-sourced human judgments of the coherence and interpretability of the representations (Iyyer et al., 2016; Chaturvedi et al., 2017). While such evaluations demonstrated that the induced representations are qualitatively valuable, it is not clear whether they are rich and general enough to be used for downstream tasks and applications. Others have used automatically created gold-standards of re-occurring character names across scripts ('gang member') (Bamman et al., 2013), prototypical plot templates (tropes, e.g., 'corrupt corporate executive') or manually created gold-standards of character types (Vala et al., 2016) or their relations (Massey et al., 2015; Chaturvedi et al., 2017) to automatically measure the *intrinsic* value of learnt representations. Here,

[2] We argue that both characters, and their relations evolve throughout the plot: Heroes pick up new attitudes or skills, and utilize those to different extents; relations change and develop over time (hate to love, friendship to enmity and back).

[3] `https://www.gutenberg.org/`

we investigate how these results extend to *extrinsic* tasks, and use structured plot representations for the task of inducing micro-clusters of novels.

Elsner (2012) depart from the above pattern, suggesting an extrinsic, albeit artificial, evaluation paradigm. Approaching plot understanding from the angle of its utility for summarization, they use kernel methods to learn character-centric plot representations. They evaluate their trained models on their ability to differentiate between real and artificially distorted novels (e.g., with shuffled chapters). While this evaluation is extrinsic and quantitative, it leverages artificial data and it is not clear how the results extend to real-world summaries.

Language features were previously used in content-based book recommendation e.g., as bags-of-words (Mooney and Roy, 1999) or semantic frames (Clercq et al., 2014). Both works use structured databases and plot summaries rather than the raw book text. Other work used topic models to augment a recommender system of scientific articles (Wang and Blei, 2011). Similar to our work, these works emphasize the added value of *interpretable* representations and recommendations, however, they do not leverage the raw content of entire novels and the richness of information encoded in those.

3 Multi-View Novel Representations

We first provide an intuitive description of Relationship Modeling Networks (RMN; Iyyer et al. 2016), and our extension (henceforth MVPlot), which *jointly* induces temporally aware *multi-view* representations of novel plots. Afterwards we describe the MVPlot model in technical detail.

3.1 Intuition

Iyyer et al. (2016) introduce the RMN, an unsupervised model which learns interpretable plot representations in terms of types of relations between pairs of book characters, and their development over time. Given a book and a character pair, the model learns relation types as word clusters (not unlike topics in a topic model (Blei et al., 2003)) from local contexts mentioning both characters. In addition the RMN learns for each character pair how these relations vary over time as a trajectory of relations. Methodologically, the RMN combines a deep recurrent autoencoder with dictionary learning, where terms in the dictionary are relationship descriptors. The RMN learns to efficiently encode local text spans as a linear combination of these relation descriptors.

View	Descriptor
v_1	laugh scream laughing yell joke cringe disgrace embarrassment hate cursing
v_1	snug fleece warm comfortable wet blanket flannel cozy comfort roomy
v_1	excellency mademoiselle monsieur majesty duchess empress madame countess madam
v_2	love loving lovely dear sweetest dearest thank darling congratulation hello
v_2	associate assistant senior chairman executive leadership vice director liaison vice-president

Table 1: Example property (v_1) and relation (v_2) descriptors induced by MVPlot on the Gutenberg corpus, as their nearest neighbours in GloVe space.

We extend RMNs to induce temporally aware multi-view representations of novel plots. Multiple interpretable views are induced jointly within one process in an unsupervised way. The core of our model closely corresponds to the structure of the RMN (as technically described in Section 3.2). However, we provide the model with distinct types of informative input for each view, and, reformulate the objective in a way that jointly optimizes parameterizations of all views to encode *distinct* information (cf., Section 3.3).

Our MVPlot model learns two views: properties associated with individual characters (v_1), relations between character pairs (v_2, as in the RMN) and their respective development over the course of the plot (examples of descriptors learnt by MVPlot for both views are shown in Table 1). Our modeling framework, however, is very general in the sense that any number and type of views can be learnt jointly as long as input with relevant signals can be provided for each view. For example, we could naturally extend the model described here with a 'plot' view to capture properties of the story which are not related to any character.

3.2 The MVPlot Model

We now formally describe the MVPlot model for learning multi-view plot representations encoding individual character properties (v_1), character pair relationships (v_2), and their respective trajectories. The full model is shown in Figure 1.

Input to our model are two corpora of text spans, one for each view, S_{v1} and S_{v2}. The corpora consist of different sets of relevant view-specific local contexts as described in Section 5. Given a book b and a character c, $S_{v1}^{c,b}$ contains

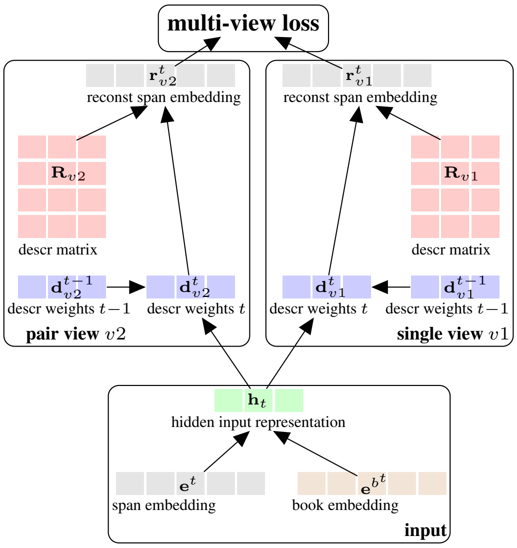


Figure 1: Visualization of the MVPlot model.

linearly ordered[4] sequences of text spans s^t at time t=$\{1...T\}$ in which character c is mentioned, but no other character,

$$S^{c,b}_{v1}=\{s^1,s^2,...,s^T\}\ s.th.\ \forall_t : c \in s^t$$

Similarly, $S^{c_1,c_2,b}_{v2}$, given a book b and a pair of characters c_1 and c_2, contains linearly ordered text spans which mention both c_1 and c_2, but no other character,

$$S_2^{c_1,c_2,b}=\{s^1,s^2,...,s^T\}\ s.th.\ \forall_t :\ c_1 \in s^t, c_2 \in s^t.$$

The rest of the input preparation follows Iyyer et al. (2016) as follows. We map text spans into word embedding space, by mapping each word w to its 300-dimensional GloVe embedding e_w (Pennington et al., 2014) pre-trained on Common-Crawl, and averaging the word embeddings,

$$\mathbf{e}^t = \frac{1}{|s^t|}\sum_{w \in s^t} e_w. \qquad (1)$$

We provide MVPlot with a trainable matrix $\mathbf{B}$ of dimensions $b \times n$, where b is the number of books in our data set, and each row $\mathbf{e}^b$ is an n-dimensional book embedding, encoding background information (e.g, about its general setting or style) which is relevant to neither view of MVPlot.[5] Finally the span embedding and the corresponding book embedding are concatenated, and passed through a ReLu non-linearity (cf., Figure 1, bottom),

$$\mathbf{h}^t = ReLu(\mathbf{W_h}[\mathbf{e}^t;\mathbf{e}^{b^t}]). \qquad (2)$$

Model architecture MVPlot uses the architecture of the RMN autoencoder, but replicates it for each input view, v_1 and v_2 (cf., Figure 1, center). Each part will induce an encoding of view-specific information. The feed-forward pass, described below, is identical for both parts, however, the loss and backpropagation will differ (cf., Section 3.3).

We describe the feed-forward pass for v_2, noting that it works analogously for v_1. The latent input representation $\mathbf{h}^t$ (eqn (2)) is passed through a softmax layer which returns a weight vector over descriptors, $\mathbf{d}^t_{v2} = softmax(\mathbf{W}^d_{v2}[\mathbf{h}^t])$. Descriptors are rows in the $k \times d$-dimensional descriptor matrix $\mathbf{R}_{v2}$, with each row k corresponding to one d-dimensional descriptor (similar to a topic in a topic model). The input $\mathbf{e}^t$ is reconstructed through the dot product of $\mathbf{d}^t_{v2}$ and the descriptor matrix $\mathbf{R}_{v2}$,

$$\mathbf{r}^t = \mathbf{d}^t_{v2}\mathbf{R}_{v2}. \qquad (3)$$

Like in the original RMN, we want to capture the *temporal* development of character relations or properties. Intuitively, we assume that the relations between (or properties of) characters at time t depend on their relations (or properties) at time $t-1$. As in the RMN, we factor the descriptor weights of the *previous* time step $\mathbf{d}^{t-1}$ into the representation at time t, such that

$$\mathbf{d}^t_{v2} = \alpha\ softmax(\mathbf{W}^d_{v2}[\mathbf{h}_t;\mathbf{d}^{t-1}_{v2}])+(1-\alpha)\mathbf{d}^{t-1}_{v2} \qquad (4)$$

Output First, the model induces property descriptors (rows in $\mathbf{R}_{v1}$) and the relationship descriptors (rows in $\mathbf{R}_{v2}$). Both sets of descriptors are optimized to reconstruct model input in GloVe embedding space (cf., Section 3.3 for details). They consequently themselves live in GloVe word embedding space, and can be visualized through their nearest neighbours in this space. In addition, for each book b, character c^b and character pair $\{c_1, c_2\}$, sequences of weight vectors over relations

$$\mathcal{T}^{c_1,c_2,b}_{v2} = \mathbf{d}^1_{v2}...\mathbf{d}^T_{v2},$$

and over properties

$$\mathcal{T}^{c,b}_{v1} = \mathbf{d}^1_{v1}...\mathbf{d}^T_{v1}$$

are induced, which encode their trajectory of relations and properties, respectively. We will utilize these trajectories for inducing micro-clusters of novels (Section 6.1).

[4] with respect to their occurrence in the novel

[5] The RMN learns background encodings for characters in addition to the book embeddings. We omit this for MVPlot as this information is explicitly learned in the views.

3.3 The Multi-View Loss

We formulate our loss as a Hinge loss within the contrastive max-margin framework. Our objective is to learn parameters for each view $\in \{v_1, v_2\}$ which efficiently encode view-specific input in a low-dimensional space from which the original input can be re-constructed with high accuracy. In addition, we want to learn view-specific parameters which encode *distinct* information such that when utilized together, they provide an improved embedding of the data. Intuitively, we achieve this by *discouraging* parameters of view v_1 from accurately reconstructing input spans from view v_2, and vice versa.

Our loss combines these two objectives as follows. The first part of the loss corresponds to the loss of the RMN. We use negative sampling to induce parameters for each view which reconstruct their respective view-specific input well. Formally, assuming model input from view v_1, $\mathbf{e}^t_{v1}$, we construct a set of 10 'negative inputs'$\{\mathbf{e}^{n_1}_{v1}, ...\mathbf{e}^{n_I}_{v1}\}$ which are sampled at random from the $v1$ input corpus. We want to learn parameters encoding view v_1 to reconstruct the input such that the inner product between the true input $\mathbf{e}^t_{v_1}$ and its reconstruction $\mathbf{r}^t_{v1}$ is higher than the inner product between $\mathbf{r}^t_{v1}$ and any of the negative samples $\mathbf{e}^{n_i}_{v1}$ by a margin of at least 1,

$$J(\theta) = \sum_t \sum_i max(0, 1 - \mathbf{r}^t_{v1}\mathbf{e}^t_{v1} + \mathbf{r}^t_{v1}\mathbf{e}^{n_i}_{v1}), \quad (5)$$

where θ refers to the set of all model parameters. We add an orthogonality-encouraging regularizing term to this objective in order to obtain view-specific descriptors which are distant from each other (Hyvärinen and Oja, 2000),

$$\begin{aligned} J(\theta) =& \sum_t \sum_i max(0, 1 - \mathbf{r}^t_{v1}\mathbf{e^t}_{v1} + \mathbf{r}^t_{v1}\mathbf{e}^{n_i}_{v1}) \\ &+ \lambda||\mathbf{R}_{v1}\mathbf{R}^T_{v1} - \mathbf{I}||. \end{aligned} \quad (6)$$

The loss is defined analogously for input of view v_2. Note that so far, the loss is defined in an entirely view-specific way, independent of the $v2$ parameters (e.g., the $v1$ loss in equation (6) is independent of the $v2$ parameters).

We break this independence by adding a second term to our loss function, which ensures that view-specific parameters encode *only relevant* information. That is, we want v_2-specific parameters to *only* encode v_2-specific information, and vice versa. Assuming model input from view v_1, $\mathbf{e}^t_{v_1}$, we learn parameters for to view v_2 that reconstruct the input poorly. Again, we use the max-margin framework, maximizing the margin between the (high) quality reconstruction of $\mathbf{e}^t_{v_1}$ from v_1 parameters, $\mathbf{r}^t_{v1}$, and the (poor) quality of the reconstruction from v_2 parameters, $\mathbf{r}^t_{v2}$,

$$K(\theta) = max(0, 1 - \mathbf{e}^t_{v_1}\mathbf{r}^t_{v1} + \mathbf{e}^t_{v_1}\mathbf{r}^t_{v2}). \quad (7)$$

The update is defined analogously, swapping $v1$ and $v2$ subscripts, when the true input stems from $v2$. The full loss is defined as a weighted linear combination of its terms (eqns (6) and (7)),

$$L(\theta) = \beta J(\theta) + (1 - \beta)K(\theta). \quad (8)$$

Genre	Example Tags
Mystery	British Detectives; FBI Agents; Female Protagonists; Private Investigators
Romance	Cowboys; Criminals & Outlaws; Doctors; Royalty & Aristocrats; Spies; Wealthy
SciFi	AIs; Aliens; Clones; Corporations; Mutants; Pirates; Psychics; Robots & Androids

Table 2: Example tags from the Amazon book catalog for the refinement `character type`.

4 Semantic Micro-Cluster Evaluation

MVPlot induces structured representations of a novel b as relation trajectories between characters pairs in b, and property trajectories of individual characters in b. Are those representations rich and informative enough to produce meaningful and interpretable micro-clusters of novels? In Section 6.1 we evaluate the quality of such micro-clusters, i.e., local novel neighbourhoods in model space. We propose an objective and empirical evaluation employing expert-provided semantic novel tags in the Amazon catalog.

Novels listed in the Amazon catalog are tagged with respect to their `genre` (e.g., *mystery, romance*). They are further labelled with *refinements* pertaining to diverse information like `character types` or `mood`, which take different sets of values, depending on the genre, and are as such predestined as an objective reference for evaluating the diverse information captured by our model. Table 2 lists example tags for the refinement `character type`.

All tags are provided by publishers and can consequently be taken as a reliable source of information. In our evaluation we assume that novels which share a tag are related to each other. We use this tag-overlap metric to evaluate local neighbourhoods of book representations in model space.

	# novels	# v_1 sequences	# v_2 sequences
Gutenberg	3,500	45,182	60,493
Amazon	10,000	91,511	70,156

Table 3: The number of novels and property (v_1) relation (v_2) input sequences for the Gutenberg and the Amazon corpus.

We selected a set of 50 representative tags from the Amazon catalog and did not tune this set for our evaluation. The full tag set is included in the supplementary material.

Note that while this scheme provides an empirical way of evaluating plot representations, it may not capture their full potential: our models are not explicitly tuned towards producing micro-clusters which are coherent with respect to our gold-standard tags, and may encode additional structure which is not probed in this evaluation. That said, we consider this evaluation as a good procedure to evaluate the *relative* quality of different models in the sense that a better model should produce micro-clusters that better correspond to reference clusters derived from gold-standard tags.

5 Data

We evaluate our model on two data sets. First, we create a diverse data set by sampling 10,000 digital novels under our 50 gold-standard tags (cf., Section 4) of the Amazon catalog (Amazon). Our second data set consists of 3,500 novels from Project Gutenberg, a large digital collection of freely available novels consisting primarily of classic literature (Gutenberg). The Amazon novels are already labelled with genre and refinement tags, such that evaluation using our gold-standard is straightforward. While Gutenberg novels come with the advantage of being freely available, they are unlabelled, and not fully covered by our 50 gold-standard tags. We therefore restrict our quantitative analysis to the Amazon data set. However, we also report qualitative results on the Gutenberg corpus, demonstrating that our model induces meaningful novel representations for corpora of varying size and diversity.

Both data sets were pre-processed with the BookNLP pipeline (Bamman et al., 2014) for coreference resolution of character mentions. We filtered stop-words and low-frequency words by discarding the 500 most frequent words and those which occur in less than 100 novels, and discarded novels less than 100 sentences long or containing fewer than 5 characters from our data set.

We created view-specific input corpora as follows: (1) a relation corpus of chronologically ordered sequences of text spans of 20 words for character pairs $\{c_1, c_2\}$ in a book b, $S_{v2}^{c_1,c_2,b}$, which mention *only* c_1 and c_2 with a margin of 10 words for the Amazon corpus (1 word for the smaller Gutenberg corpus) but no other character; and (2) a property corpus of chronologically ordered sequences of 20 word text spans for individual characters c in book b, $S_{v2}^{c_1,c_2,b}$, which mention *only* c, using the same margins as above.

We keep only sequences of length n time steps s.th., $5 \leq n \leq 250$. We only keep pair sequences if we also obtain sequences for each individual character confirming to the above criteria. Table 3 summarizes statistics on our input corpora.

6 Evaluation

Section 6.1 quantitatively evaluates the quality of local neighbourhoods in model space induced from the Amazon corpus against our proposed gold-standard. Section 6.2 evaluates the quality of the induced descriptors from both the Amazon and Gutenberg corpus both through crowd sourcing and illustrative examples.

Models We set the MVPlot performance into perspective comparing it against the RMN.[6] MVPlot induces both character properties and relations, and is trained on both the relation-view and property-view input, while the RMN only induces pair relationships and can only utilize relation-view input. In addition, we report a frequency baseline, which is trained on both property and relation-view input. We concatenate all input spans of a given view for a particular novel; construct its term frequency vector and use cosine similarity to compute the nearest neighbours to each novel.

Parameter settings Across all experiments and corpus-specific models, we set β=0.99 for MVPlot, and for both MVPlot and RMN we set α=0.5, λ=10^{-5}, k=50.[7] We train both RMN and MVPlot for 15 epochs using SGD and ADAM (Kingma and Ba, 2014).[8]

[6] We do not compare against topic model baselines because they were outperformed by RMN (Iyyer et al., 2016).

[7] Parameters were tuned on a small subset of books used in the nearest neighbourhood evaluation (Section 6.1).

[8] Our implementation builds on the available RMN code `https://github.com/miyyer/rmn`.

6.1 Nearest Neighbours Evaluation

We evaluate local neighbourhoods in model space using the 500 most popular novels by their number of Amazon reviews as reference novels from the Amazon corpus. For each reference novel we compute the 10 nearest neighbours as described below. We measure neighbourhood quality using the gold-standard tags from Section 4, regarding neighbours as *relevant* if at least one tag is shared with the reference novel. We report precision at rank 10 (P@10) and mean average precision (MAP).

Method MVPlot represents a book b in terms of trajectories of weight vectors over relation descriptors $\mathcal{T}_{v2}^{b}$ and property descriptors $\mathcal{T}_{v1}^{b}$. RMN only learns relation descriptors and their trajectories. For both models, we map each induced trajectory for book b to a fixed-sized k-dimensional vector representation by averaging the time-specific weight vectors, for example for a v_2 trajectory,

$$\mathcal{T}_{v2}^{c_1,c_2,b} = \frac{1}{\left|\mathcal{T}_{v2}^{c_1,c_2,b}\right|} \sum_{t\in\left|\mathcal{T}_{v2}^{c_1,c_2,b}\right|} \mathbf{d}_{v2}^{t}, \tag{9}$$

and equivalently for v_1 trajectories, $\mathcal{T}_{v1}^{c,b}$.

We compute the similarity between two books $\{b_r, b_c\}$ as follows. We align the v_2 trajectory for each character pair $\{c_1, c_2\}$ in b_r, $\mathcal{T}^{c_1,c_2,b_r}$, to its closest neighbouring character pair vector in b_c, $\mathcal{T}^{c'_1,c'_2,b_c}$, by Euclidean distance, and compute the overall book similarity in terms of character relations between b_r and b_c as the average over all distances.

$$sim_{v2}^{b_r,b_c} = \frac{1}{|\mathcal{T}_{v2}^{b_r}|} \sum_{\mathcal{T}\in\mathcal{T}_{v2}^{b_r}} \operatorname*{arg\,min}_{\mathcal{T}'\in\mathcal{T}_{v2}^{b_c}} dist(\mathcal{T},\mathcal{T}'). \tag{10}$$

We obtain $sim_{v1}^{b_r,b_c}$ in an analogous process. For $cosine$ and MVPlot we obtain a final, *multi-view* similarity by averaging similarity scores obtained in each view's space,

$$sim_{both}^{b_r,b_c} = \frac{1}{2}\left(sim_{v1}^{b_r,b_c} + sim_{v2}^{b_r,b_c}\right). \tag{11}$$

For RMN we compute similarity only in character relation space.

Model	**View**	P@10	MAP
cosine	v_1	0.516 ‡	0.392 †
	v_2	0.468 ‡	0.339 ‡
	$both$	0.512 ‡	0.390 ‡
RMN	v_2	0.479 ‡	0.347 ‡
MVPlot	v_1	0.529 †	0.401 †
	v_2	0.496 ‡	0.367 ‡
	$both$	**0.546**	**0.421**

Table 4: Micro-cluster quality results (Amazon corpus). Differences of $cosine$ and RMN compared to the best MVPlot result are significant with $p < 0.05$ (†) or $p < 0.01$ (‡) (paired t-test).

Results Table 4 presents micro-cluster quality in terms of $precision$@10 and MAP. The full MVPlot model statistically significantly outperfoms all other models.[9] The same pattern emerges when comparing models with the same underlying views: MVPlot $v2$ outperforms both cosine $v2$ and RMN $v2$ (similarly for MVPlot $v1$ and cosine $v1$), indicating that the MVPlot character relation representations are most informative for micro-cluster induction.

In order to shed light on the contribution of individual model components, we compare the full MVPlot model (*both*) to model versions with access to only $v1$ or $v2$ (Table 4 bottom). Combining information from both views boosts performance compared to the single-view versions. This confirms that MVPlot indeed encodes distinct and relevant information in the respective views.

While cosine is a strong baseline, its representations are not structured or interpretable. It consequently does not provide sufficient information for applications like book tagging or recommendation with respect to specific aspects or criteria. Similarly, RMN cannot learn representations of multiple, distinct views of the plot.

Advancing our understanding of the information encoded in the individual views of MVPlot, we took a closer look at the refinement tags for which the single view MVPlot model ($v1$) has the clearest advantage over the pair view MVPlot model ($v2$), and vice-versa. We computed tagwise F1-scores for the two MVPlot variants. Table 5 lists the book tags for which the scores of the two views diverge the most.

In terms of types of refinements, view $v2$ suffers most for predicting book categories, or topical tags ('sports', 'second changes'), while view $v1$ is particularly deficient for predicting character types. While this seems counterintuitive we hypothesize that character types are to a large extent defined by their interactions with, or relations to, other char-

[9] Also, intra-view comparisons except for MVPlot v_1 and cosine v_1 are statistically significant.

$F1\_v2 >> F1\_v1$		$F1\_v1 >> F1\_v2$	
Tag	**RefType**	**Tag**	**RefType**
Robots & Androids	Character	Hard SciFi	Category
Corporations	Character	Sports	Category
International	Theme	Horror	Theme
Aliens	Character	Second Chances	Theme
Cowboys	Character	Crime	Category

Table 5: The tags (Tag) and their refinement types (RefType) for which MVPlot $v1$ most clearly outperforms MVPlot $v2$ (left) and vice versa (right) in terms of tag-specific F1-measure.

acters. Topical information, however, is encoded robustly in the properties of individual characters.

6.2 Evaluating Induced Descriptors

This evaluation investigates whether induced relation descriptors indeed capture relational information. We evaluate the interpretability of the induced descriptors, comparing the v_2 (relation) descriptors induced by RMN and MVPlot. We apply both models to both the Amazon and the Gutenberg corpus, and report results on both data sets.

Method We collect crowdsourced judgments on Amazon Mechanical Turkto qualitatively evaluate the learnt descriptors, following Chaturvedi et al. (2017). In each task a worker is shown one induced descriptor as a set of its 10 closest words in GloVe space (like in Table 1), and is asked to indicate whether "the words in the group describe a relation, event or interaction between people". We compare the proportion of positive answers, i.e., the number of descriptors considered *relevant*, for RMN descriptors and MVPlot pair descriptors. We collect 30 judgments for each of k=50 descriptors induced by the respective models.

Results Figure 2 displays our results. We observe a similar pattern of ratings across models and corpora, e.g., around 50% of the descriptors are labelled as relevant by at least 50% of the annotators. None of the differences are statistically significant which lets us conclude that interpretability of induced descriptors is comparable for the RMN and MVPlot. This is encouraging because we confirm that representation interpretability is not compromised by MVPlot's more complex objective.

Table 1 displays examples of property and relation descriptors induced by MVPlot from the Gutenberg corpus. We can see that the different views indeed capture differing information (e.g., a v_1 descriptor refers to individuals' *titles*, while a v_2 descriptor refers to a *love* relation). Despite its smaller size and more homogeneous nature, we show that MVPlot learns meaningful representations from the Gutenberg corpus, demonstrating the flexibility of our model.

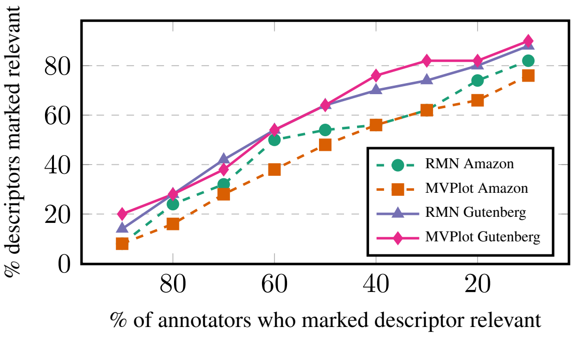


Figure 2: Results of descriptor interpretability. (% of descriptors marked as 'relevant descriptors of relations' by various proportions of annotators).

Figure 3 further illustrates this, displaying example local neighbourhoods of four reference novels (left) with their eight nearest neighbours ordered by proximity (left to right). The neighbourhoods are intuitively meaningful, and particularly impressive bearing in mind that the full model space contains $3,500$ novels. While most neighbourhoods are dominated by novels of the same author, some exceptions emerge. Row two, for example, contains novels by Thomas Hardy and Charles Dickens who both are known for biographical 17th century novels focusing on class and social changes.

7 Conclusions

Content-based micro-clustering of novels is a complex but interesting task. In order to eventually augment the diverse associations humans have, models must be able to pick up rich and structured signals from raw text. This paper presented a deep recurrent autoencoder which learns multi-view representations of plots, and introduced a principled evaluation framework using clusters based on expert-provided book tags.

Our evaluation showed that rich multi-view representations are better suited to recover such reference clusters compared to each individual view, as well as compared to simpler, but competitive models which induce less structured representations. Our view-specific representations are interpretable which allows to analyse and explain the emerging

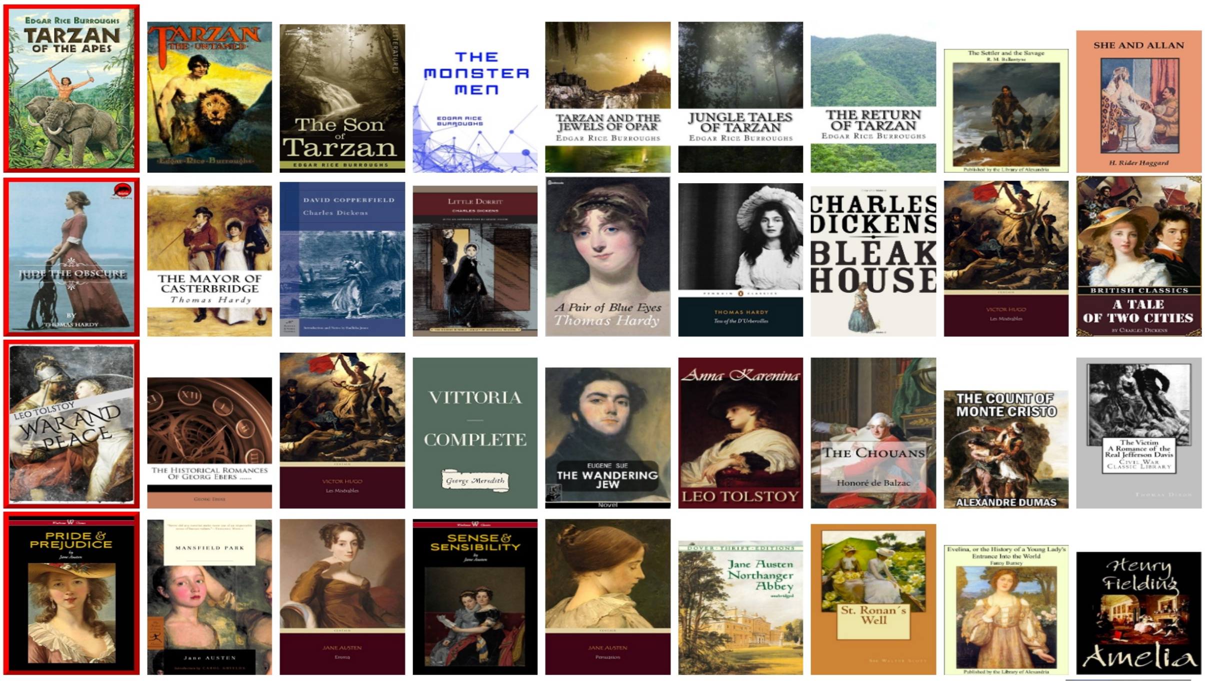


Figure 3: Nearest neighbours for four classic stories from the Gutenberg Corpus. Target novels on the left (with red border), and NNs are presented in the same row, ordered by their distance to the target novel.

micro-clusters, and might reveal previously unnoticed parallels between novels and may be useful for literary analysis or content-based recommendation. This is an exciting avenue for future work.

Our method is general and scalable both in terms of its input, utilizing raw text with only automatic pre-processing, and in terms of the number of distinct views it can learn. We described an objective function which triggers views to encode *distinct* information. In future work we plan to explore joint learning of more and different views.

Our approach relies strongly on the assumption that text spans mentioning two characters contain information about character relations, and that text spans mentioning one character contain information about the character's properties. While our results suggest that these assumptions are valid, they are arguably crude. In the future we plan to define more targeted input, e.g., by using semantic and syntactic information from dependency parses.

In this work we induced dual-view representations of book content, however, we emphasize that the proposed method is very general. The number and kinds of views, as well as underlying data are in no way constrained, as long as relevant view-specific input can be defined. In the context of novel representation it would be interesting to induce additional views, for example one that captures the mood of a novel. Another interesting avenue for future work would be to apply the framework to questions arising in the digital humanities, e.g., to extract different views from news articles.

The presented model and evaluation are designed with the objective to detect a different kinds of similarity between novels, with the ultimate goal to *enrich* human-provided genres and tags. We described a first step in this direction, verifying that the learnt information is meaningful and can *reproduce* human-created semantic book tags. Expert book tags exist for a wide variety of information (mood, theme, characters), and provide a rich evaluation environment for learnt representations. We invite the community to join us in exploring the full space of opportunities and evaluating induced representations *holistically* in the future.

Acknowledgements

We would like to thank Alex Klementiev, Kevin Small, Joon Hao Chuah and Mohammad Kanso for their valuable insights, feedback and technical help on the work presented in this paper. We also thank the anonymous reviewers for their valuable feedback and suggestions.

References

Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749.

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 352–361, Sofia, Bulgaria. Association for Computational Linguistics.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 370–379, Baltimore, Maryland. Association for Computational Linguistics.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610, Suntec, Singapore. Association for Computational Linguistics.

Jonathan Chang, Jordan L. Boyd-Graber, and David M. Blei. 2009. Connections between the lines: augmenting social networks with text. In *KDD*, pages 169–178. ACM.

Snigdha Chaturvedi, Mohit Iyyer, and Hal Daumé III. 2017. Unsupervised learning of evolving relationships between literary characters. In *Association for the Advancement of Artificial Intelligence*.

Orphée De Clercq, Michael Schuhmacher, Simone Paolo Ponzetto, and Véronique Hoste. 2014. Exploiting framenet for content-based book recommendation. In *Proceedings of the 1st Workshop on New Trends in Content-based Recommender Systems co-located with the 8th ACM Conference on Recommender Systems, CBRecSys@RecSys 2014, Foster City, Silicon Valley, California, USA, October 6, 2014.*, pages 14–21.

Micha Elsner. 2012. Character-based kernels for novelistic plot structure. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 634–644, Avignon, France. Association for Computational Linguistics.

David Elson, Nicholas Dames, and Kathleen McKeown. 2010. Extracting social networks from literary fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 138–147, Uppsala, Sweden. Association for Computational Linguistics.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1180–1189. JMLR Workshop and Conference Proceedings.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030.

Amit Gruber, Yair Weiss, and Michal Rosen-Zvi. 2007. Hidden topic markov models. In *AISTATS*, volume 2 of *JMLR Proceedings*, pages 163–170. JMLR.org.

A. Hyvärinen and E. Oja. 2000. Independent component analysis: Algorithms and applications. *Neural Netw.*, (4-5):411–430.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *North American Association for Computational Linguistics*.

Matthew L. Jockers. 2013. *Macroanalysis: Digital Methods and Literary History*. University of Illinois Press.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2011. *Content-based Recommender Systems: State of the Art and Trends*. Springer US, Boston, MA.

Philip Massey, Patrick Xia, David Bamman, and Noah A. Smith. 2015. Annotating character relationships in literary texts. *CoRR*, abs/1512.00728.

Neil McIntyre and Mirella Lapata. 2010. Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1562–1572, Uppsala, Sweden. Association for Computational Linguistics.

Raymond J. Mooney and Loriene Roy. 1999. Content-based book recommending using learning for text categorization. In *Proceedings of the SIGIR-99 Workshop on Recommender Systems: Algorithms and Evaluation*, Berkeley, CA.

Bruno A. Olshausen and David J. Field. 1997. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311 – 3325.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Hardik Vala, Stefan Dimitrov, David Jurgens, Andrew Piper, and Derek Ruths. 2016. Annotating Characters in Literary Corpora: A Scheme, the CHARLES Tool, and an Annotated Novel. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 448–456, New York, NY, USA. ACM.

Initializing Convolutional Filters with Semantic Features for Text Classification

Shen Li[1,2]
shen@mail.bnu.edu.cn

Zhe Zhao[4,5]
helloworld@ruc.edu.cn

Tao Liu[4,5]
tliu@ruc.edu.cn

Renfen Hu[3,†]
irishere@mail.bnu.edu.cn

Xiaoyong Du[4,5]
duyong@ruc.edu.cn

[1] Institute of Chinese Information Processing, Beijing Normal University
[2] UltraPower-BNU Joint Laboratory for Artificial Intelligence, Beijing Normal University
[3] College of Chinese Language and Culture, Beijing Normal University
[4] School of Information, Renmin University of China
[5] Key Laboratory of Data Engineering and Knowledge Engineering, MOE

Abstract

Convolutional Neural Networks (CNNs) are widely used in NLP tasks. This paper presents a novel weight initialization method to improve the CNNs for text classification. Instead of randomly initializing the convolutional filters, we encode semantic features into them, which helps the model focus on learning useful features at the beginning of the training. Experiments demonstrate the effectiveness of the initialization technique on seven text classification tasks, including sentiment analysis and topic classification.

1 Introduction

Recently, neural networks (NNs) dominate the state-of-the-art results on a wide range of natural language processing (NLP) tasks. The commonly used neural networks in NLP include Recurrent NNs, Convolutional NNs, Recursive NNs and their combinations. NNs are known for their strong abilities to learn features automatically. However, the lack of data or inappropriate parameter settings might greatly limit the generalization abilities of the models (Bengio et al., 2009; LeCun et al., 2015; Krizhevsky et al., 2012; Srivastava et al., 2014). To enhance the performance, a lot of improved methods have been proposed, e.g. developing advanced structures (Zhao et al., 2015; Zhang et al., 2016a), introducing prior knowledge (Hu et al., 2016) and utilizing external resources (Xie et al., 2016; Qian et al., 2016).

It is also noteworthy that the neural networks' performance is sensitive to weight initialization because their objectives are non-convex (Glorot and Bengio, 2010; Saxe et al., 2013; Mishkin and Matas, 2015). In fact, initialization techniques even play a role of catalyst for the revival of neural networks (Hinton et al., 2006; LeCun et al., 2015). Most improvements on initializing weights are based on mathematical methods, e.g. xavier initialization (Glorot and Bengio, 2010) and orthogonal initialization (Saxe et al., 2013). For NLP tasks, an influential technique is to use pre-trained word vectors to initialize embedding layers (Kim, 2014; Chen and Manning, 2014). Consider the embedding layers could be initialized by pre-trained word vectors, how about weights in other layers that are still randomly initialized?

Inspired by this question, we propose a simple yet effective method to improve CNNs by initializing convolutional layers (filters). Unlike the previous weight initialization based on mathematical methods, we encode semantic features into the filters instead of initializing them randomly. As CNNs exploit 1-D convolutional filters to extract n-gram features, our method aims at helping the filters focus on learning useful n-grams, e.g. "not bad" which is more useful than "watch a movie" for determining reviews' polarities. Specifically, we select n-grams from training data via a novel Naive Bayes (NB) weighting technique, and then cluster the n-gram embeddings with K-means algorithm. After that, we use the centroid vectors of the clusters to initialize the filters.

With this initialization method, CNN filters tend to extract important n-gram features at the beginning of the training process. By integrating our method into a classic CNN model for text classification (Kim, 2014), we observe significant im-

[†] Corresponding author.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1884–1889
Copenhagen, Denmark, September 7–11, 2017.

provements in sentiment analysis and topic classification tasks. The advantages of our approach are as follows:

- Features are directly extracted from training data without involving any external resources;
- The computation brought by our method is relatively small, resulting in small additional training costs;
- The filter initialization is task independent. It could be easily applied to other NLP tasks.

Also, we further analyze the filters, shedding some light on the mechanism how our method influences the training process. The source code is released at https://github.com/shenshen-hungry/Semantic-CNN.

2 Related Work

Most recently, CNNs are becoming increasingly popular in a variety of NLP tasks. An influential one is the work of (Kim, 2014), where a simple CNN with a single layer of convolution is used for feature extraction. Despite its simple structure, the model achieves strong baselines on many sentence classification datasets. Following this work, several improved models are proposed. Zhang and Wallace (2015) improve the model by optimizing hyper-parameters and provide a detailed analysis of the CNN (Kim, 2014). Yin and Schütze (2016) and Zhang et al. (2016b) exploit different pre-trained word embeddings (e.g. word2vec and GloVe) to enhance the model.

In addition to initializing embedding layers with pre-trained word vectors, other pre-designed features also prove to be very effective in assisting the training of neural models. For example, in (Hu et al., 2016), neural models are harnessed by logic rules. Li et al. (2016) propose to use pre-calculated words' weights to guide Paragraph Vector model. Dai and Le (2015) combine the hidden layers of RNNs with linearly increasing weights. Xie et al. (2016) use entity descriptions from knowledge bases (e.g. Freebase) to learn knowledge representations for entity classification and knowledge graph completion. Qian et al. (2016) propose linguistically regularized LSTMs for sentiment analysis with sentiment lexicons, negation words, and intensity words. In this work, we encode semantic features into convolutional layers by initializing them with important n-grams. Being aware of which n-grams are important, CNN is able to extract more discriminative features for text classification.

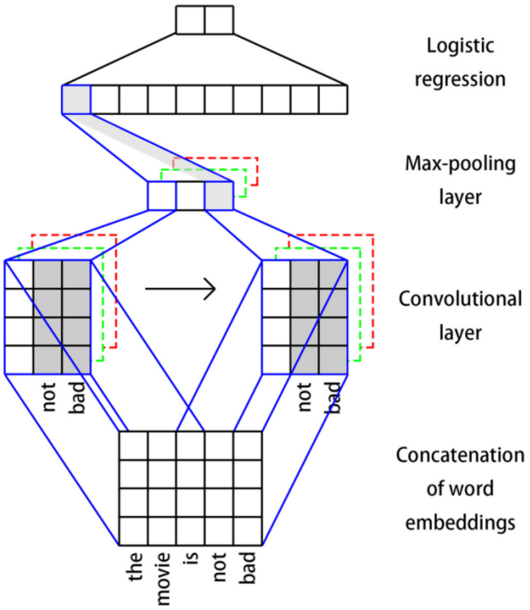


Figure 1: The framework of CNN with one layer of convolution and pooling.

3 Our Method

The intuition behind our method is simple: Since CNNs essentially capture semantic features of n-grams, we can use important n-grams to initialize the filters. As a result, the filters are able to focus on extracting those important n-gram features at the beginning of the training. As shown in Figure 1, we use embeddings of "not" and "bad" to initialize the filter. A larger score will be obtained when the "not bad" filter matches the bigram "not bad" in the text, otherwise a relatively smaller score will be returned.

3.1 N-gram Selection

Firstly, we extract important n-grams from the training data. Intuitively, n-gram "not bad" is much more important than "watch a movie" for determining reviews' polarities. Naive Bayes (NB) weighting is an effective technique for determining the words' importance (Martineau and Finin, 2009; Wang and Manning, 2012). NB weight $\mathbf{r}$ of a n-gram w in class c is calculated as follows:

$$\mathbf{r} = \frac{(\mathbf{p}_c^w + \alpha)/||\mathbf{p}_c||_1}{(\mathbf{p}_{\tilde{c}}^w + \alpha)/||\mathbf{p}_{\tilde{c}}||_1}$$

where $\mathbf{p}_c^w$ is the number of texts that contain

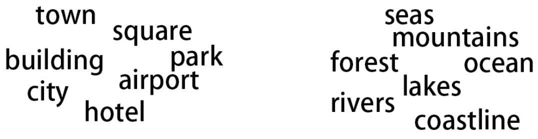


Figure 2: Uni-gram cluster examples.

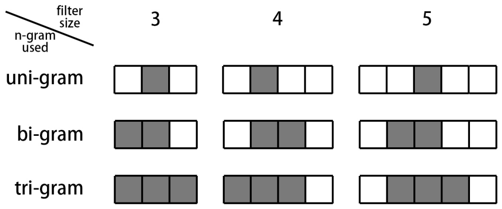


Figure 3: Filter initialization.

n-gram w in class c, $\mathbf{p}_{\bar{c}}^{w}$ is the number of texts that contain n-gram w in other classes, $||\mathbf{p}_c||_1$is the number of texts in class c, $||\mathbf{p}_{\bar{c}}||_1$ is the number of texts in other classes, α is a smoothing parameter. For *positive* class in movie review dataset, the ratios of n-grams like "amazing" and "not bad" should be large since they appear much more frequently in positive texts than in negative texts. For neutral n-grams like "of the" and "movie", their ratios should be around 1. For each class, we select the n-grams whose ratios are much higher than 1 for filter initialization. We give examples of n-grams selected by our method in Appendix.

3.2 Filter Initialization

We concatenate word embeddings to construct n-gram embeddings. For example, a tri-gram embedding has 3*100 dimensions when word embedding has 100 dimensions. This concatenation follows the mechanism of convolutional filters,where a filter with n*d dimensions is able to capture n-gram features (d is the dimension of word embedding). Because the number of filters in CNNs is much smaller than the number of n-grams, a filter tends to extract the features of a class of n-grams rather than an individual n-gram. Based on this observation, we don't use n-gram embeddings to initialize the filters directly. Instead, we firstly use K-means to cluster features of the selected n-grams, and then use the clusters' centroid vectors to initialize the filters. In this work, we consider clustering uni-gram (word), bi-gram and tri-gram features. Figure 2 shows two uni-gram cluster examples extracted from the location questions in TREC dataset (Li and Roth, 2002).

After obtaining the n-gram clusters, we feed their centroid vectors into the center of the filters. The remaining positions are still initialized randomly. Taking filters with size 3, 4, 5 as examples, Figure 3 shows how we fill uni, bi, and tri-gram features into the filters. By doing this, we encode semantic features into the filters. For example, in the TREC question classification task, the initialization will result in six types of filters which are sensitive to abbreviation, entity, description, human, location and number questions respectively.

4 Experiments

4.1 Datasets and Hyper-parameter Settings

CNN-non-static[1] (short for CNN) proposed by Kim (2014) is used as our baseline, which consists of one embedding layer, one convolutional layer, one max pooling layer, and one fully connected layer. The model proposed by Kim (2014) is a strong baseline in sentence classification. For details of the model, one can see (Kim, 2014; Zhang and Wallace, 2015). Pre-trained word embeddings on Google News via word2vec toolkit[2] are used for initializing the convolutional filters, besides initializing the embedding layer of CNN as in (Kim, 2014). For a fair comparison, we use the same seven datasets [3] and hyper-parameter setting with Kim (2014)'s work for training and testing. Uni, bi, and tri-gram features are used to initialize the filters. For a K-way classification problem, we select top *10%* n-grams in each class according to NB weighting. Since *300* filters are used in Kim (2014)'s work, we follow this setting and aggregate n-grams into *300/K* clusters for each class. Centroid vectors are used for filling the filters. Taking binary classification dataset MR as an example, *150* "positive" filters and *150* "negative" filters are obtained after initialization.

4.2 Effectiveness of Filter Initialization

In this section, we demonstrate the effectiveness of our initialization technique. We respectively use uni, bi and tri-gram centroid vectors to fill the filters. Table 1 lists the results. The CNN has provided very strong baselines. Our method

[1] The embedding layer in CNN-non-static is initialized with pre-trained vectors from word2vec toolkit and fine-tuned for each task.

[2] https://code.google.com/p/word2vec/

[3] (MR (Pang and Lee, 2005), SST-1/2 (Socher et al., 2013), Subj (Pang and Lee, 2004), TREC (Li and Roth, 2002), CR (Hu and Liu, 2004), and MPQA (Wiebe et al., 2005))

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-non-static	81.5	48.0	87.2	93.4	93.6	84.3	**89.5**
+UNI	82.1	**50.8**	**89.0**	93.7	94.4	**86.0**	89.3
+BI	**82.2**	50.7	88.3	93.7	**94.6**	85.8	**89.5**
+TRI	82.1	49.8	88.2	**93.8**	94.2	85.9	89.2

Table 1: Effectiveness of filter initialization.

Model	MR	SST-1	SST-2	Subj	TREC	CR	MPQA
CNN-non-static (Kim, 2014)	81.5	48.0	87.2	93.4	93.6	84.3	**89.5**
MV-CNN (Yin and Schütze, 2016)	-	49.6	**89.4**	93.9	-	-	-
MGNC-CNN (Zhang et al., 2016b)	-	48.7	88.3	**94.1**	**95.5**	-	-
CNN-Rule (Hu et al., 2016)	81.7	-	89.3	-	-	85.3	-
Our Model (CNN-non-static+UNI)	**82.1**	**50.8**	89.0	93.7	94.4	**86.0**	89.3
combine-skip (Kiros et al., 2015)	76.5	-	-	93.6	92.2	80.1	87.1
Adasent (Zhao et al., 2015)	**83.1**	-	-	**95.5**	92.4	**86.3**	**93.3**
DSCNN (Zhang et al., 2016a)	82.2	50.6	**88.7**	93.9	**95.6**	-	-
PV (Le and Mikolov, 2014)	74.8	48.7	87.8	90.5	91.8	78.1	74.2
NBSVM (Wang and Manning, 2012)	79.4	-	-	93.2	-	81.8	86.3
Tree LSTM (Tai et al., 2015)	-	**51.0**	88.0	-	-	-	-

Table 2: Comparisons of state-of-the-arts.

further improves the accuracies significantly on all datasets except MPQA. The results are consistent with (Wang and Manning, 2012), where NB weighting produces little improvement over MPQA. We can also observe that the performance of uni, bi and tri-grams are comparable. None of them outperforms the others on all datasets.

4.3 Comparisons with State-of-the-arts

Table 2 lists the results of our model and other state-of-the-arts. Models in the first group are improved CNNs based on (Kim, 2014). Among them, MV-CNN and MGNC-CNN utilize multiple pre-trained embeddings as inputs, and CNN-Rule integrates logic rules. Our model achieves the best performance on three tasks without requiring any extra training costs and resources. With this simple initialization method, our model also gives competitive results against more sophisticated deep learning models in the second group, e.g. Adasent (Zhao et al., 2015) and DSCNN (Zhang et al., 2016a) that have complex structures or use the combinations of NNs.

Experiments show that our n-gram features make great contribution to both two-class and multi-class classification. Essentially, our method enables CNNs to obtain better generalization abilities. Furthermore, as the initialization does not rely on any external prior knowledge or resources, it could be easily applied to other NLP tasks or other languages.

	positive filters		negative filters	
	+	-	+	-
UNI	29.5	20.5	21.3	28.7
BI	31.3	18.7	18.1	31.9
TRI	31.4	18.6	17.2	32.8

Table 3: "+" and "-" are used to denote the number of positive and negative weights respectively. The data in the table are obtained from MR by the average of 10 times training. Every time we select 100 filters. 50 of them are initialized with positive n-grams and the rest are with negative n-grams.

4.4 Further Analysis of Filters

We further analyze the filter initialization with an example of binary sentiment classification. Through the initialization we have determined which filters are positive or negative in advance. The corresponding neurons of positive filters upon max-pooling layer are supposed to be activated by positive samples. Since positive (negative) samples have labels of 1 (0), the corresponding weights (in logistic regression) of those "positive" neurons tend to be positive. For the same reason, the negative filters tend to have negative weights. The results shown in Table 3 confirm our hypothesis: Positive/negative filters respectively tend to have positive(+)/negative(-) weights. The difference between positive and negative filters are more obvious in bi-gram and tri-gram cases. It is because bi and tri-gram centroid vectors could initialize more parameters of filters than uni-gram.

In Table 1, experiments show that different choices of uni, bi, and tri-grams have little influence on the results. The following is our assumption: Compared to uni-grams (words), bi and trigrams can cover more spaces of filters and introduce more NB information to filters. Filters initialized by them thus pay more attention to NB information than filters initialized by uni-grams according to Table 3. However, bi and tri-grams are also sparser in data than uni-grams. Their NB weights are not as accurate as those of uni-grams, even applied smoothing. As NB weight of a n-gram denotes its contribution to the classification, model initialized with tri-grams does not always perform the best.

5 Conclusion

This paper proposes a novel weight initialization technique for CNNs. We discover that convolutional filters that encode semantic features at the beginning of the training tend to produce better results than being randomly initialized. This has a similar effect with embedding layer initialization via pre-trained word vectors. Experimental results demonstrate the effectiveness of the initialization technique on multiple text classification tasks. In addition, our method requires few external resources and relatively small calculation, making it attractive for scenarios where training costs may be an issue.

In textual data, the features extracted by CNNs are n-grams. However, in fields like computer vision, features extracted by filters are more difficult to interpret. It still requires further exploration to apply our method to fields beyond NLP.

Acknowledgments

This work is supported by the National High Technology Research and Development Program of China (No. 2012AA011104), National Language Committee Research Program of China (No. YB125-124), and National Natural Science Foundation of China (No. 61472428).

References

Yoshua Bengio et al. 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning* 2(1):1–127.

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

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Proceedings of NIPS 2015*. pages 3079–3087.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*. volume 9, pages 249–256.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computation* 18(7):1527–1554.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of ACM SIGKDD 2004*. ACM, pages 168–177.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard H. Hovy, and Eric P. Xing. 2016. Harnessing deep neural networks with logic rules. In *Proceedings of ACL 2016*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP 2014*. pages 1746–1751.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of NIPS 2015*. pages 3294–3302.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. pages 1097–1105.

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

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521(7553):436–444.

Bofang Li, Zhe Zhao, Tao Liu, Puwei Wang, and Xiaoyong Du. 2016. Weighted neural bag-of-n-grams model: New baselines for text classification. In *Proceedings of COLING 2016*. pages 1591–1600.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of ACL 2002*. Association for Computational Linguistics, pages 1–7.

Justin Martineau and Tim Finin. 2009. Delta TFIDF: an improved feature space for sentiment analysis. In *Proceedings of ICWSM 2009*.

Dmytro Mishkin and Jiri Matas. 2015. All you need is a good init. *CoRR* abs/1511.06422.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL 2004*. Association for Computational Linguistics, page 271.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL 2005*. Association for Computational Linguistics, pages 115–124.

Qiao Qian, Minlie Huang, and Xiaoyan Zhu. 2016. Linguistically regularized lstms for sentiment classification. *arXiv preprint arXiv:1611.03949* .

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *CoRR* abs/1312.6120.

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP 2013*. Citeseer, volume 1631, page 1642.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Sida I. Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of ACL 2012, Volume 2: Short Papers*. pages 90–94.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation* 39(2):165–210.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*. pages 2659–2665.

Wenpeng Yin and Hinrich Schütze. 2016. Multichannel variable-size convolution for sentence classification. *CoRR* abs/1603.04513.

Rui Zhang, Honglak Lee, and Dragomir R. Radev. 2016a. Dependency sensitive convolutional neural networks for modeling sentences and documents. In *Proceedings of NAACL 2016*. pages 1512–1521.

Ye Zhang, Stephen Roller, and Byron C. Wallace. 2016b. MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification. *CoRR* abs/1603.00968.

Ye Zhang and Byron C. Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR* abs/1510.03820.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *Proceedings of IJCAI 2015*. pages 4069–4076.

Shortest-Path Graph Kernels for Document Similarity

Giannis Nikolentzos
École Polytechnique and AUEB
nikolentzos@aueb.gr

Polykarpos Meladianos
École Polytechnique and AUEB
pmeladianos@aueb.gr

François Rousseau
École Polytechnique
rousseau@lix.polytechnique.fr

Michalis Vazirgiannis
École Polytechnique and AUEB
mvazirg@aueb.gr

Yannis Stavrakas
IMIS / RC ATHENA
yannis@imis.athena-innovation.gr

Abstract

In this paper, we present a novel document similarity measure based on the definition of a graph kernel between pairs of documents. The proposed measure takes into account both the terms contained in the documents and the relationships between them. By representing each document as a graph-of-words, we are able to model these relationships and then determine how similar two documents are by using a modified shortest-path graph kernel. We evaluate our approach on two tasks and compare it against several baseline approaches using various performance metrics such as DET curves and macro-average F1-score. Experimental results on a range of datasets showed that our proposed approach outperforms traditional techniques and is capable of measuring more accurately the similarity between two documents.

1 Introduction

In recent years, we have witnessed a tremendous growth in the volume of textual documents available on the Web. With this rapid increase in the number of available content, new opportunities for knowledge extraction have arisen. Many text mining tasks such as information retrieval, text categorization and document clustering involve the direct comparison of two documents. It is thus crucial to be able to determine accurately how similar two documents are by defining a document similarity measure.

Generally speaking, a similarity measure is a real-valued function that quantifies the common information shared by two objects (in our case documents). Determining the similarity between two documents is not a trivial task. Whether two documents are similar or different is not always clear and may vary from application to application.

Similarity measures that make use of the vector-space model (Salton et al., 1975) treat words in a document as if they were independent of one another, which is not realistic. In fact, words relate to one another to form meaningful phrases and to develop ideas. It is known that the human brain utilizes these relations between words to facilitate understanding (Altmann and Steedman, 1988). In general, we assume that two terms are related if they co-occur together in a small context, typically a phrase or a window of specific size, which resulted in n-gram features in many text mining tasks (an n-gram is a sequence of n terms in this paper). But n-grams correspond to sequences of words and thus fail to capture word inversion and subset matching (e. g., "article about news" vs. "news article"). To take into account these statistical relations, we propose to represent each document as a graph-of-words instead. And then, in order to measure the similarity between two documents, we capitalize on recent advances in graph kernels. Kernels can be thought of as measures of similarity between pairs of objects (Schölkopf and Smola, 2002). A graph kernel is a kernel function that measures the similarity between pairs of graphs.

Our aim in this paper is neither to define a similarity measure for only a certain category of documents based on background knowledge and features specific to that field nor to improve similarity estimation by using external knowledge. In-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1890–1900
Copenhagen, Denmark, September 7–11, 2017.

stead, we propose to define a similarity measure that does not incorporate any background or external knowledge. Hence it is, without changes, applicable to all types of textual documents even if they come from different areas. The method takes as input a pair of documents and automatically computes how similar they are to each other based solely on their content.

The rest of this paper is organized as follows. Section 2 provides an overview of the related work and elaborates our contribution. Section 3 provides a detailed description of our proposed graph-of-words kernel. Section 4 evaluates the proposed approach on a wide range of tasks. Finally, Section 5 summarizes the work and presents potential future work.

2 Related Work

In this section, we review the related work published in the areas of *document similarity*, *graph kernels*, *kernel-based text categorization* and *graph-based text categorization*.

2.1 Document Similarity

There has been a variety of similarity measures defined to assess how close two objects are to each other, including documents. Let $< d_1, d_2 >$ be a pair of documents and D_1 (resp. D_2) the set of terms in d_1 (resp. d_2). Common similarity measures discussed by Manning (1999) are defined as follows:

$$Matching(d_1, d_2) = |D_1 \cap D_2|$$

$$Dice(d_1, d_2) = 2\frac{|D_1 \cap D_2|}{|D_1| + |D_2|}$$

$$Jaccard(d_1, d_2) = \frac{|D_1 \cap D_2|}{|D_1 \cup D_2|}$$

$$Overlap(d_1, d_2) = \frac{|D_1 \cap D_2|}{min(|D_1|, |D_2|)}$$

$$Cosine(d_1, d_2) = \frac{|D_1 \cap D_2|}{\sqrt{|D_1| \times |D_2|}}$$

The terms might be processed unigrams as well as processed n-grams present in the text. The set of operations described above are equivalent to vector operations when representing d_1 and d_2 as binary vectors.

2.2 Graph Kernels

Graph kernels are instances of the R-convolution kernels introduced by Haussler (1999). Convolution kernels have been proposed as a principled way of designing kernels on structured objects, such as sequences, trees and graphs. Graph kernels compute the similarity between pairs of graphs, based on common substructures they share. A wide variety of substructures has been proposed, such as random walks (Gärtner et al., 2003; Vishwanathan et al., 2010), shortest paths (Borgwardt and Kriegel, 2005), subtrees (Ramon and Gärtner, 2003), cycles (Horváth et al., 2004), and graphlets (Shervashidze et al., 2009).

2.3 Kernel-based Text Categorization

In recent years, there has been a great deal of work in using kernel methods, such as SVMs for text classification (Joachims, 1998; Dumais et al., 1998). Such work concentrates on building specialized kernels aimed at measuring similarity between documents. We outline some of these approaches below.

The works closest to ours are the ones reported by Lodhi et al. (2002) and by Cancedda et al. (2003). Lodhi et al. propose the use of string kernels as an alternative to the vector-space model. The feature space is generated by any ordered subsequence of characters found in the text not necessarily contiguously. Each subsequence consists of a specific number of characters and is weighted by an exponentially decaying factor of its full length in the text. Due to the enormous amount of computation needed to compute this feature vector, the authors present a dynamic programming technique, which allows the efficient calculation of the kernel values. Our work differs from theirs in that we use graph kernels instead of sequence kernels, and we concentrate on the word level instead of the character level. Cancedda et al. modified their string kernel to work with sequences of words rather than characters. Two sequences of words are considered similar if they have many common words in a given order. The similarity between two documents is assessed by the number of matching word sequences. Non-contiguous occurrences are penalized according to the number of gaps they contain. The proposed kernel is more appealing as it is more computationally efficient and it takes advantage of the standard linguistic preprocessing techniques. This approach differs in fundamental respects from our work since we represented documents as graphs-of-words in order to model word co-occurrence rather than sequences of words and we used a graph kernel instead of a sequence kernel to measure the similarity between pairs of documents. Other text categorization works use kernels that measure the semantic similarity between concepts extracted from the text (Bleik et al., 2013; Wang and Domeniconi, 2008).

2.4 Graph-based Text Categorization

Our work is also related to methods that represent documents as graphs and perform graph mining tasks to achieve improved classification performance. These methods either extract frequent subgraphs which are then used to produce feature vectors for the documents (Jiang et al., 2010; Rousseau et al., 2015) or they determine term weights to be used in the vector-space model based on centrality criteria or random walks (Hassan et al., 2007; Malliaros and Skianis, 2015).

3 A Graph Kernel for Document Similarity

In this section, we first discuss the essential definitions from graph theory. We then present our graph-of-words model for representing textual documents. And finally, we define our custom Shortest-Path Graph Kernel (SPGK) capable of measuring the similarity between pairs of documents.

3.1 Graph Concepts

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected and unweighted graph consisting of a set $\mathcal{V}$ of vertices and a set $\mathcal{E}$ of edges between them. In this paper, we will denote by n the number of vertices and by m the number of edges.

A labeled graph is a graph with labels on vertices and/or edges. Given a set of labels $\mathcal{L}$, $\ell : V \rightarrow \mathcal{L}$ is a function that assigns labels to the and/or edges of the graph. In our case, we deal with fully-labeled graphs as labels are assigned both to vertices and to edges.

A graph $\mathcal{G}$ can be represented by its adjacency matrix $\mathbf{A}$. The $(i,j)^{th}$ entry of $\mathbf{A}$ is 1 if the edge (v_i, v_j) between vertices v_i and v_j exists, and 0 otherwise.

A walk in a graph $\mathcal{G}$ is a sequence of vertices $v_1, v_2, \ldots, v_{k+1}$ where $v_i \in \mathcal{V}$ and $(v_i, v_{i+1}) \in \mathcal{E}$ for $1 \leq i \leq k$. The length of the walk is equal to the number of edges in the sequence, i. e. k in the above case. A walk in which $v_i \neq v_j \Leftrightarrow i \neq j$ is called a path. In other words, a path is a walk without repetition of nodes.

3.2 Graph-of-words

We chose to represent each textual document as a statistical graph-of-words, following earlier approaches in keyword extraction (Ohsawa et al., 1998; Mihalcea and Tarau, 2004) and more recent ones in ad hoc IR (Blanco and Lioma, 2012; Rousseau and Vazirgiannis, 2013) and in summarization (Meladianos et al., 2015).

The construction of each graph is preceded by a preprocessing phase where standard text processing tasks such as tokenization, stopword, punctuation and special character removal, and stemming are performed. The processed document is then transformed into an unweighted, undirected graph whose vertices represent unique terms and whose edges represent co-occurrences between the connected terms within a fixed-size window (hence the statistical denomination). The graph-of-words representation of text provides enhanced modeling capabilities compared to the bag-of-words representation. Besides the terms (vertices), it also models the relationships between them (edges). All the words present in a document have some relationships with one another, modulo a window size outside of which the relationship is not taken into consideration, and graphs are able to capture these dependencies. The extended modeling capabilities, however, come with an increase in complexity.

An example of a document represented as an unweighted undirected graph is given in Figure 1. The source text comes from Shakespeare's play "Hamlet": "to be or not to be: that is the question". For illustration purposes, only the colon is removed and no other text processing tasks are performed. The size of the window is set to 2, i. e. it captures bigram relationships. Hence, each word (vertex) is connected with an edge with its previous and its next word, if any.

3.3 Shortest-Path Graph-of-words Kernel (SPGK)

Our proposed approach measures the similarity between two textual documents by representing them as graphs-of-words, transforming these graphs into other graphs, and using graph kernels to calculate the similarity of the new graphs. Specifically, we capitalize on the shortest-path graph kernel (Borgwardt and Kriegel, 2005) and

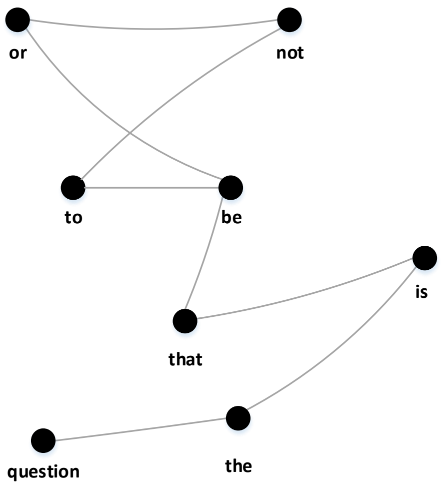


Figure 1: Example of the graph representation of a textual document.

we modify it to compare the graph representations of pairs of documents.

The first step of our proposed approach is to transform the graph-of-words representation of each document $\mathcal{G}$ into another graph $\mathcal{C}$ whose vertices are connected with an edge only if the shortest distance between them is not greater than a variable d. The emerging graph contains the same set of vertices as the graph-of-words from which it was generated. However, there exist edges only between vertices that are connected by a path of length at most d. Every node in $\mathcal{C}$ is labeled by the term that it represents, while every edge between two vertices is labeled by the shortest distance between these vertices given that it is no greater than d. Specifically, the label of an edge e that links two vertices whose shortest path is p is set equal to $label(e) = {}^{1}\!/_{p}$. For $d = 1$, the emerging network is equivalent in a structural sense to its corresponding graph-of-words. For greater values of d, it is very likely that the number of edges of the graph will have increased compared to its predecessor.

The commonly-used unigram bag-of-words representation assumes that words in a document are independent of one another. Although similarity measures based on this assumption have shown to work well in practice in many fields, it is not rational to completely ignore word order and word dependence. Hence, the distance between two terms in a document determines their relationship. This led us to explore alternative document similarity metrics that take into account the co-occurrence of words in the documents. More specifically, we assume that two terms are related given that they appear together inside a window. The underlying assumption is that each word present in a document has some relationship with the other words that are close to it. We set the size of the window over the processed text equal to 2. Therefore, in our graph-of-words representation of a document, each term is linked with its preceding and its following term with an edge. In our transformed graphs, terms are not only connected with terms that are next to them, but also with terms that are close to their neighbors ($d = 2$), with lower label values, and close to neighbors of their neighbors ($d = 3$), with even lower label values. Parameter d determines how far from the initial terms we allow the paths to go. Our intuition is that given an initial term, terms that are close to terms that are close to the initial term or beyond, may have also some relation with the initial term, and the strength of this relation decreases as the shortest path length increases. Therefore, although the proposed kernel does not incorporate any knowledge of the language being used, it does capture some statistical information and is thus capable of outperforming metrics based on the unigram and even n-gram vector-space model.

To determine the edge labels in the new graph $\mathcal{C}$, we can perform depth-first search (DFS) or breadth-first search (BFS) traversals from each vertex in the graph, limiting the depth to d. The complexity for calculating paths of length up to d from a source vertex to all other vertices using either DFS or BFS is at most $\mathcal{O}(b^d)$, where b is the average branching factor. The branching factor depends on the average degree of the vertices of the graphs-of-words $\mathcal{G}$ which, in its turn, depends on the selected size of the sliding window. For $W = 2$, the average degree of the vertices will be typically only slightly above 2 and the branching factor will be only slightly above 1. Calculating paths of length up to d for all vertices takes thus $\mathcal{O}(nb^d)$ time. This still yields reasonable time complexity estimates for small values of d.

After our original graphs have been transformed into the graphs described above, we can measure their similarity using the following kernel:

Definition 1 (Custom shortest-path graph kernel)**.** *Let $\mathcal{G}_1$, $\mathcal{G}_2$ denote two graph-of-words representations of two textual documents d_1, d_2 that are* *transformed into graphs $\mathcal{C}_1$, $\mathcal{C}_2$ through the process described above. The proposed Shortest-Path Graph Kernel (SPGK) on $\mathcal{C}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{C}_2 = (\mathcal{V}_2, \mathcal{E}_2)$ is defined as follows:*

$$k(d_1, d_2) = \frac{\begin{pmatrix} \sum_{v_1 \in \mathcal{V}_1, v_2 \in \mathcal{V}_2} k_{node}(v_1, v_2) \\ + \sum_{e_1 \in \mathcal{E}_1, e_2 \in \mathcal{E}_2} k_{walk}^{(1)}(e_1, e_2) \end{pmatrix}}{norm} \tag{1}$$

where k_{node} is a positive definite kernel for comparing two vertices, $k_{walk}^{(1)}$ a positive definite kernel for comparing two edge walks of length 1 in $\mathcal{C}$ (i. e. up to d in $\mathcal{G}$) and $norm$ a normalization factor described next.

The similarity value generated by our custom shortest-path graph kernel is equal to the sum over the kernel values of all pairs of vertices on the transformed graphs plus the sum over the kernel values of all pairs of edge walks of length 1 over a positive normalization factor. The k_{node} kernel is a function for comparing two vertices. In practice, we use a delta kernel defined as:

$$k_{node}(v_1, v_2) = \begin{cases} 1 & \text{if } \ell(v_1) = \ell(v_2), \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

but other works have considered distances in word embeddings for instance to account for word similarity at the cost of having to compare every node of a graph to every other nodes of the other graph (Srivastava et al., 2013).

The normalization factor is introduced because the nominator of the proposed kernel depends on the length of the compared documents. Specifically, given the adjacency matrices of the transformed graph representations of two documents $\mathbf{A}_1, \mathbf{A}_2$ where the value of each entry in the adjacency matrix is set equal to the label of the corresponding edge, and the diagonal matrices $\mathbf{D}_1, \mathbf{D}_2$ with diagonal entries set to 1 if the corresponding term exists in the corresponding document, we first compute the matrices M_1, M_2 as shown below:

$$\mathbf{M}_1 = \mathbf{A}_1 + \mathbf{D}_1$$

$$\mathbf{M}_2 = \mathbf{A}_2 + \mathbf{D}_2$$

and we then compute the normalization factor using the following formula:

$$norm = \|\mathbf{M}_1\|_F \times \|\mathbf{M}_2\|_F$$

where $\|\cdot\|_F$ is the Frobenius norm for matrices.

The $k_{walk}^{(1)}$ kernel can be expressed as the product of kernels on vertices and edges along the walk. Only walks of length 1 in $\mathcal{C}$ are considered, therefore, $k_{walk}^{(1)}$ can be calculated in terms of the original vertex, the destination vertex, and the edge connecting them.

Definition 2 (Custom edge walk kernel)**.** *Let u_1, v_1 be two vertices of graph $\mathcal{C}_1$ ($u_1, v_1 \in \mathcal{V}_1$) and e_1 the edge connecting them. Let also u_2, v_2 be two vertices of graph $\mathcal{C}_2$ ($u_2, v_2 \in \mathcal{V}_2$) and e_2 the edge connecting them. The edge walk kernel is defined as follows:*

$$k_{walk}^{(1)}(e_1, e_2) = k_{node}(u_1, u_2) \times k_{edge}(e_1, e_2) \\ \times k_{node}(v_1, v_2) \tag{3}$$

where k_{node} is the kernel function defined above and k_{edge} is a kernel function for comparing two edges defined as follows:

$$k_{edge}(e_1, e_2) = \begin{cases} \ell(e_1) \times \ell(e_2) & \text{if } e_1 \in \mathcal{E}_1 \wedge \\ & e_2 \in \mathcal{E}_2, \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The measure of similarity between two graphs depends on the kernel values corresponding to the vertices and edges that compose each walk, while the matching between two vertices or two edges is determined by comparing their labels. The values of our kernel function lie in the interval $[0, 1]$. It takes a value equal to 0 for documents with no common terms and a value equal to 1 for identical documents.

Lemma 1. *SPGK is a valid kernel.*

Proof. Based on the proofs presented in (Borgwardt and Kriegel, 2005) and (Borgwardt et al., 2005), we show that our custom shortest-path graph kernel is positive definite. The k_{node} kernel is a delta kernel, which is known to be positive definite (Schölkopf and Smola, 2002) and therefore a valid kernel. The k_{edge} kernel is also a delta kernel multiplied by a positive real number. Since the multiplication of a kernel by a positive constant preserves positive definiteness, this kernel is also valid. Regarding the $k_{walk}^{(1)}$ kernel, it is positive definite as the point-wise multiplication of positive definite kernels (k_{node}, k_{edge}) preserves positive definiteness. The $\sum_{v_1 \in \mathcal{V}_1, v_2 \in \mathcal{V}_2} k_{node}(v_1, v_2)$ function is the sum of valid kernel functions, hence, it is also positive definite. Regarding the $\sum_{e_1 \in \mathcal{E}_1, e_2 \in \mathcal{E}_2} k^{(1)}_{walk}(e_1, e_2)$ function, it is a walk kernel that takes into account only walks of length 1 on the transformed graphs and is zero-extended to the whole set of pairs of walks that do not satisfy the above constraint. Therefore, kernel values for walks with length greater than 1 are set to zero. This zero-extension is known to preserve positive definiteness (Haussler, 1999). This function is a convolution kernel, which is proven to be positive definite (Haussler, 1999). Finally, the kernel is divided by a positive constant and its positive definiteness is preserved. □

3.4 Run Time Complexity

We now determine the time complexity of our proposed kernel for measuring the similarity between two documents. Let us assume that the graph-of-words representations of the two documents consist of n vertices each. To determine the shortest paths of length at most d from a root vertex to all other vertices, we need $\mathcal{O}(b^d)$ time when using a graph traversal algorithm (depth-first or breadth-first search). There are also n vertices in the transformed graph, hence, the transformation will require $\mathcal{O}(nb^d)$ time for each graph. In order to determine the kernel value, it is necessary to compute the value of $k^{(1)}_{walk}$ for all pairs of edges between the two transformed graphs. The number of edges in the transformed graph can be at most n^2 in the case all the shortest paths in the original graph are no longer than d. Thus, there are at most $n^2 \cdot n^2 = n^4$ pairs of edges. However, due to the label enrichment that has been applied to the vertices of the transformed graphs, the number of matching nodes in the two graphs has been radically reduced and the number of pairs of edges that have to be considered is also reduced. Specifically, we have to consider n^2 pairs of edges as only paths between vertices whose label is the same in the two graphs are considered. The kernel value can thus be computed in $\mathcal{O}(n^2 + nb^d)$ time.

3.5 Alternative Computation Method for $d = 1$

In the case we consider only the common paths of length 1, there is a more efficient algorithm to compute the kernel values. The common paths of length 1 correspond to common edges between the graph representations of the documents. The emerging kernel takes into account the number of common vertices (terms) between the two graphs and the number of common edges (terms co-occurring in the same window) as well. More specifically, given two documents d_1 and d_2, the adjacency matrices of their graph representations $\mathbf{A}_1, \mathbf{A}_2$ where each entry in the adjacency matrix is set to 1 if the corresponding edge exists in the graph and the diagonal matrices $\mathbf{D}_1, \mathbf{D}_2$ with diagonal entries set to 1 if the corresponding term exists in the document, we first compute the matrices $\mathbf{M}_1, \mathbf{M}_2$ as described previously and then we compute the kernel value using the following formula:

$$k(d_1, d_2) = \frac{\sum \mathbf{M}_1 \circ \mathbf{M}_2}{\|\mathbf{M}_1\|_F \times \|\mathbf{M}_2\|_F} \qquad (5)$$

where $(\cdot \circ \cdot)$ is the Hadamard or element-wise product between matrices.

If n is the number of unique node labels, i. e. the length of the vocabulary, and m the number of edges, the computation of the kernel values requires $\mathcal{O}(n + m)$ time in the worst case scenario. For the baseline similarity measures, with unigram features, the computational cost is $\mathcal{O}(n)$ time but it goes up as we consider higher order n-grams.

4 Experiments and Evaluation

In this section, we present the experiments we conducted to evaluate and validate our proposed kernel between documents.

4.1 Evaluation Metrics

To assess the effectiveness of the different approaches, we employed a set of well-known evaluation metrics inherited from Information Retrieval: *accuracy*, *macro-average F1-score* and for the story link detection task *DET curves* (Martin et al., 1997).

The DET curve is a variant of the ROC curve that plots the *missed detection probability* ($P_{miss} = {}^{fn}/_{(tp+fn)}$) versus the *false alarm probability* ($P_{fa} = {}^{fp}/_{(tn+fp)}$) for various system operating points, which allows someone to get a greater insight into the effectiveness of the evaluated approaches. A method is considered to perform best at thresholds that correspond to points that are close to the lower-left of the graph (i. e. lower error probabilities) and the area under the curve should be minimal.

For the story link detection experiments, we also computed the normalized C_{Det} costs, the

Dataset	# training examples	# test examples	# classes	vocabulary size	avg. terms per document	avg. degree
WebKB	$2,803$	$1,396$	4	$7,772$	77.93	2.54
News	$32,604$	CV	7	$34,131$	25.57	2.08
Subjectivity	$10,000$	CV	2	$21,335$	20.74	2.08
Amazon	$6,400$	$1,600$	4	$39,133$	86.96	2.67
Polarity	$10,662$	CV	2	$18,777$	18.44	2.00

Table 1: Summary of the 5 datasets that were used in our text categorization experiments.

standard performance measure of TDT as described in (Fiscus and Wheatley, 2004).

4.2 Datasets

We evaluate the SPGK and the baselines on 5 standard datasets for text categorization: (1) `WebKB`: Web pages collected from Computer Science departments of various Universities manually classified into 7 categories (we removed Web pages that belong to the classes "staff", "department" and "other") (Craven et al., 1998). (2) `News`: News extracted from RSS feeds of popular newspaper websites classified into 7 categories based on the taxonomies of their publishing websites (Vitale et al., 2012). (3) `Subjectivity`: Subjective and objective sentences corresponding to movie reviews from Rotten Tomatoes and to plot summaries gathered from the Internet Movie Database respectively (Pang and Lee, 2004). (4) `Amazon`: Product reviews over four different sub-collections (Blitzer et al., 2007). (5) `Polarity`: Positive and negative snippets acquired from Rotten Tomatoes (Pang and Lee, 2005). Table 1 shows statistics of the datasets that were used for the evaluation. For the Story Link Detection task, we employed the TDT-5 corpus that contains stories from various newswire sources (Glenn et al., 2006; Graff and Kong, 2006). We only used the English part of the dataset for our experiments consisting of $221,306$ documents.

4.3 Baselines

The similarity measure presented in this paper is best suited for settings where the concept of a predefined corpus does not exist. For example, it could find applications in plagiarism detection and in cases where independent pairs of documents must be compared to each other. In such settings, due to the absense of a corpus, we cannot learn mappings of terms to a vector space (i. e. word embeddings) or use methods that take advantage of the corpus to increase their performance. Hence, our set of baselines includes methods that take as input two documents and output their similarity.

More specifically, the performance of our proposed kernel was compared to the performances of three baseline kernels based on similarity measures between pairs of documents $< d_1, d_2 >$ in the n-gram feature space (up to 4-grams):

1. The linear kernel, which uses the *dot product* as similarity measure: $k_{dp}(\vec{d_1}, \vec{d_2}) = \vec{d_1} \cdot \vec{d_2}$ where $\vec{d}$ is the n-gram feature vector associated with the document d;

2. *Cosine*, which measures the cosine of the angle between the two vectors: $k_c(\vec{d_1}, \vec{d_2}) = \frac{\vec{d_1} \cdot \vec{d_2}}{\|\vec{d_1}\| \times \|\vec{d_2}\|}$ where $\| \cdot \|$ is the L_2-norm.

3. *Tanimoto coefficient* (also known as Jaccard coefficient), which measures the intersection of features divided by their union: $k_{tc}(\vec{d_1}, \vec{d_2}) = \frac{\vec{d_1} \cdot \vec{d_2}}{\|\vec{d_1}\|^2 + \|\vec{d_2}\|^2 - \vec{d_1} \cdot \vec{d_2}}$

In the task of text categorization, we also compared the proposed kernel against the so-called Dynamic Convolutional Neural Network (*DCNN*) which is capable of generating representations for larger pieces of text such as sentences and documents (Kalchbrenner et al., 2014) and a convolutional neural network (CNN) architecture that has recently showed state-of-the-art results on many NLP sentence classification tasks (Kim, 2014). We used two variants of the CNN: (1) a model where all words are initialized to random vectors and are kept static during training (*CNN static,rand*), and (2) a model where again all words are initialized to random vectors, but are modified during training (*CNN non-static,rand*). The second model as well as DCNN have access to the whole corpus to generate word/document embeddings. Hence, it is not fair in a sense to compare the proposed kernel against these methods.

Dataset		WebKB		News		Subjectivity		Amazon		Polarity	
Method		Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
Dot product	$n = 1$	0.9026	0.8923	0.8110	0.7764	0.8992	0.8992	0.9188	0.9188	0.7627	0.7626
	$n = 2$	0.9047	0.8950	0.8091	0.7732	0.9101	0.9101	0.9200	0.9202	0.7746	0.7745
	$n = 3$	0.9026	0.8917	0.8072	0.7710	0.9090	0.9090	0.9181	0.9185	0.7741	0.7740
	$n = 4$	0.8940	0.8813	0.8031	0.7651	0.9039	0.9039	0.9131	0.9133	0.7719	0.7718
Cosine	$n = 1$	0.9248	0.9188	0.8117	0.7766	0.9003	0.9002	0.9400	0.9400	0.7670	0.7669
	$n = 2$	0.9305	0.9275	**0.8149**	**0.7797**	0.9094	0.9094	0.9413	0.9413	0.7756	0.7756
	$n = 3$	0.9298	0.9259	0.8097	0.7738	0.9099	0.9099	0.9419	0.9418	0.7765	0.7765
	$n = 4$	0.9248	0.9208	0.8076	0.7709	0.9076	0.9075	0.9413	0.9413	0.7753	0.7753
Tanimoto	$n = 1$	0.9062	0.8983	0.8155	0.7815	0.9094	0.9093	0.9225	0.9226	0.7749	0.7748
	$n = 2$	0.9040	0.8945	0.8075	0.7700	0.9061	0.9060	0.9181	0.9185	0.7735	0.7735
	$n = 3$	0.9241	0.9180	0.7980	0.7575	0.9021	0.9020	0.9344	0.9347	0.7648	0.7648
	$n = 4$	0.9176	0.9084	0.7899	0.7483	0.8953	0.8952	0.9300	0.9300	0.7586	0.7586
DCNN		0.8918	0.8799	0.7991	0.7615	0.9026	0.9026	0.9181	0.9181	0.7326	0.7326
CNN	static,rand	> 1 day		0.7757	0.7337	0.8716	0.8715	0.8881	0.8882	0.7150	0.7150
	non-static,rand	> 1 day		0.8113	0.7749	0.8961	0.8960	0.9356	0.9356	0.7654	0.7653
SPGK	$d = 1$	0.9327	0.9278	0.8104	0.7749	**0.9148***	**0.9148**	0.9400	0.9401	0.7776	0.7775
	$d = 2$	**0.9370***	**0.9336**	0.8089	0.7729	0.9146*	0.9146	0.9413	0.9413	**0.7789***	**0.7788**
	$d = 3$	0.9291	0.9233	0.8078	0.7703	0.9137*	0.9137	0.9444	0.9444	0.7761	0.7760
	$d = 4$	0.9291	0.9223	0.8097	0.7730	0.9118	0.9118	**0.9463**	**0.9463**	0.7780	0.7780

Table 2: Performance of the 6 approaches in text categorization. * indicates statistical significance in accuracy improvement at $p < 0.05$ using the micro sign test against the Cosine ($n = 2$) baseline of the same column. > 1 day indicates that the computation did not finish after 1 day.

4.4 Text Categorization

To perform text categorization, for all methods except the DCNN and the two CNNs, we employed a Support Vector Machine (SVM) classifier (Boser et al., 1992). It is interesting to note that all we need to train an SVM classifier is the kernel matrix of the training examples. We optimized the parameter C of the SVM by performing 10-fold cross-validation on the training set. We then made predictions on the test set using the optimal value of C. For DCNN the dimensionality of the generated embeddings was set to 100, while for the two CNNs it was set to 300. For DCNN and the two CNNs, the number of training epochs was set to 25. All similarity measures were coded in Python[1].

For each value of the parameter d, we obtain a new kernel and in turn the resultant kernel matrix contains different values. To study the effect of parameter d on the classification performance, we performed tests for values of d ranging from 1 to 4. We did not further increase the value of d since in most cases, for values greater than 4, the performance of the classifier stayed the same.

Table 2 shows the performance of the baseline methods and the proposed shortest-path graph kernel (SPGK), on the five datasets. Bold font marks the best performance in a column, while $*$ indicates statistical significance in accuracy improvement at $p < 0.05$ using the micro sign test (Yang and Liu, 1999) against the Cosine ($n = 2$) baseline of the same column. We chose to test for significance against that measure, as it corresponds to the best-performing baseline. On all datasets except one (News), SPGK outperforms the other three similarity measures and the neural network architectures. In addition, the results show a statistically significant improvement of at least one of our kernels over the Cosine ($n = 2$) approach on all datasets except two (News, Amazon). In general, our kernel is followed in performance by Cosine, Tanimoto, Dot Product in that order. The three neural network architectures fail to outperform the proposed kernel even on a single dataset. Furthermore, the approaches that make use of the whole corpus to generate embeddings (DCNN and CNN non-static,rand) do not seem to gain any advantage from having access to the whole dataset. This may be due to the fact that the size of the datasets is not large enough for learning high-quality representations.

4.5 Story Link Detection

Story link detection, as defined by the Topic Detection and Tracking (TDT) research program (Allan, 2002), is the task of determining whether two stories, such as news articles and radio broadcasts, are "linked" by the same event. According to TDT,

[1] Code available at: `http://www.db-net.aueb.gr/nikolentzos/code/spgk.zip`

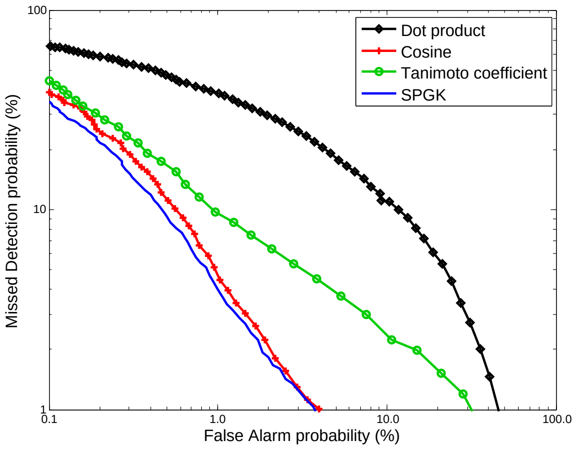


Figure 2: DET curves for all similarity measures on story link detection track.

Similarity measure		$(\mathrm{C_{det}})_{\mathrm{norm}}$
Dot product		0.3908
Cosine		0.0953
Tanimoto coefficient		0.1453
SPGK	$d = 1$	**0.0883**
	$d = 2$	0.0884
	$d = 3$	0.0888
	$d = 4$	0.0888

Table 3: Performance of all similarity measures in story link detection.

an event is something that happens at some specific time and place and two stories are "linked" if they discuss the same event.

In Figure 2, we plot the DET curves comparing the proposed approaches. For clarity, we only plot one curve for our SPGK approach ($d = 1$) since the plots overlapped, and the best performing curve for each of the baseline approaches. It is clear that our approach outperforms the baselines over the whole set of operating points. We also searched for the threshold values for which each approach maximizes its performance. Our next step was to compare the four systems in terms of detection effectiveness at that optimal threshold. Table 3 illustrates the normalized C_{det} of the proposed methods and the baselines. We can see that the proposed methods are better than baseline methods in terms of the normalized C_{det} metric.

5 Conclusion

In this paper, we presented a graph kernel for measuring the similarity between pairs of documents. The graph-of-words representation of textual documents allows us to model relationships between terms in documents and, hence, to go beyond the limits of the vector-space model. At the same time, it allows us to measure the similarity between two documents by comparing their graph representations using kernel functions. The effectiveness of the proposed kernel was empirically tested on two different tasks, namely text categorization and story link detection. The proposed measure showed improved performance on both tasks compared to the baselines.

References

J. Allan. 2002. Introduction to Topic Detection and Tracking. In *Topic Detection and Tracking*, pages 1–16.

G. Altmann and M. Steedman. 1988. Interaction with context during human sentence processing. *Cognition*, 30(3):191–238.

R. Blanco and C. Lioma. 2012. Graph-based term weighting for information retrieval. *Information retrieval*, 15(1):54–92.

S. Bleik, M. Mishra, J. Huan, and M. Song. 2013. Text Categorization of Biomedical Data Sets using Graph Kernels and a Controlled Vocabulary. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(5):1211–1217.

J. Blitzer, M. Dredze, and F. Pereira. 2007. Biographies, Bollywood, Boomboxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447.

K. M. Borgwardt and H. Kriegel. 2005. Shortest-path kernels on graphs. In *Proceedings of the 5th IEEE International Conference on Data Mining*, pages 74—81.

K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H. Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl 1):i47–i56.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. 1992. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 144–152.

N. Cancedda, E. Gaussier, C. Goutte, and J. M. Renders. 2003. Word-Sequence Kernels. *The Journal of Machine Learning Research*, 3:1059–1082.

M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. 1998. Learning to Extract Symbolic Knowledge from the World Wide Web. In *Proceedings of the 10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pages 509–516.

S. Dumais, J. Platt, D. Heckerman, and M. Sahami. 1998. Inductive Learning Algorithms and Representations for Text Categorization. In *Proceedings of the 7th ACM International Conference on Information and Knowledge Management*, pages 148–155.

J. Fiscus and B. Wheatley. 2004. Overview of the TDT 2004 Evaluation and Results. In *TDT Workshop*.

T. Gärtner, P. Flach, and S. Wrobel. 2003. On Graph Kernels: Hardness Results and Efficient Alternatives. In *Learning Theory and Kernel Machines*, pages 129–143.

M. Glenn, S. Strassel, J. Kong, and K. Maeda. 2006. TDT5 Topics and Annotations. *Linguistic Data Consortium (LDC)*.

D. Graff and J. Kong. 2006. TDT5 Multilingual Text. *Linguistic Data Consortium (LDC)*.

S. Hassan, R. Mihalcea, and C. Banea. 2007. Random-Walk Term Weighting for Improved Text Classification. *International Journal of Semantic Computing*, 1(04):421–439.

D. Haussler. 1999. Convolution Kernels on Discrete Structures. Technical report, UCSC-CRL-99-10, Department of Computer Science, University of California, Santa Cruz.

T. Horváth, T. Gärtner, and S. Wrobel. 2004. Cyclic Pattern Kernels for Predictive Graph Mining. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 158–167.

C. Jiang, F. Coenen, R. Sanderson, and M. Zito. 2010. Text classification using graph mining-based feature extraction. *Knowledge-Based Systems*, 23(4):302–308.

T. Joachims. 1998. *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Y. Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. 2002. Text Classification using String Kernels. *The Journal of Machine Learning Research*, 2:419–444.

F. Malliaros and K. Skianis. 2015. Graph-Based Term Weighting for Text Categorization. In *Proceedings of the 2015 International Conference on Advances in Social Networks Analysis and Mining*, pages 1473–1479.

C. D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT press.

A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. 1997. The DET Curve in Assessment of Detection Task Performance. Technical report, DTIC Document.

P. Meladianos, G. Nikolentzos, F. Rousseau, Y. Stavrakas, and M. Vazirgiannis. 2015. Degeneracy-based Real-Time Sub-Event Detection in Twitter Stream. In *Proceedings of the 9th AAAI Conference on Web and Social Media*, pages 248–257.

R. Mihalcea and P. Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 404–411.

Y. Ohsawa, N. E. Benson, and M. Yachida. 1998. KeyGraph: Automatic Indexing by Co-occurrence Graph based on Building Construction Metaphor. In *Proceedings of the Advances in Digital Libraries Conference*, pages 12–18.

B. Pang and L. Lee. 2004. A Sentimental Education: Sentiment Analysis using Subjectivity Summarization based on Minimum Cuts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 271–278.

B. Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124.

J. Ramon and T. Gärtner. 2003. Expressivity versus Efficiency of Graph Kernels. In *1st International Workshop on Mining Graphs, Trees and Sequences*, pages 65–74.

F. Rousseau, E. Kiagias, and M. Vazirgiannis. 2015. Text Categorization as a Graph Classification Problem. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1702–1712.

F. Rousseau and M. Vazirgiannis. 2013. Graph-of-word and TW-IDF: New Approach to Ad Hoc IR. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 59–68.

G. Salton, A. Wong, and C. Yang. 1975. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620.

B. Schölkopf and A. J. Smola. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press.

N. Shervashidze, T. Petri, K. Mehlhorn, K. M. Borgwardt, and S. Vishwanathan. 2009. Efficient Graphlet Kernels for Large Graph Comparison. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, pages 488–495.

S. Srivastava, D. Hovy, and E. H. Hovy. 2013. A Walk-Based Semantically Enriched Tree Kernel Over Distributed Word Representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1411–1416.

S. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. 2010. Graph Kernels. *The Journal of Machine Learning Research*, 11:1201–1242.

D. Vitale, P. Ferragina, and U. Scaiella. 2012. Classification of Short Texts by Deploying Topical Annotations. In *Advances in Information Retrieval*, pages 376–387.

P. Wang and C. Domeniconi. 2008. Building Semantic Kernels for Text Classification using Wikipedia. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 713–721.

Y. Yang and X. Liu. 1999. A Re-examination of Text Categorization Methods. In *Proceedings of the 22nd International SIGIR Conference on Research and Development in Information Retrieval*, pages 42–49.

Adapting Topic Models using Lexical Associations with Tree Priors

Weiwei Yang
Computer Science (CS)
University of Maryland
College Park, MD
wwyang@cs.umd.edu

Jordan Boyd-Graber
CS, iSchool, LSC, and UMIACS
University of Maryland
College Park, MD
jbg@umiacs.umd.edu

Philip Resnik
Linguistics and UMIACS
University of Maryland
College Park, MD
resnik@umd.edu

Abstract

Models work best when they are optimized taking into account the evaluation criteria that people care about. For topic models, people often care about interpretability, which can be approximated using measures of lexical association. We integrate lexical association into topic optimization using *tree priors*, which provide a flexible framework that can take advantage of both first order word associations and the higher-order associations captured by word embeddings. Tree priors improve topic interpretability without hurting extrinsic performance.

1 Introduction

Goodman (1996) introduces a key insight for machine learning models in natural language processing: if you know how performance on a problem is evaluated, it makes more sense to optimize using *that* evaluation metric, rather than others. Goodman applies his insight to parsing algorithms, but this insight has had an even larger impact in machine translation, where the introduction of the fully automatic BLEU metric makes it possible to tune systems using a score correlated with human rankings of MT system performance (Papineni et al., 2002).

Chang et al. (2009) provide a similar insight for topic models (Blei et al., 2003, LDA): if what you care about is the interpretability of topics, the standard objective function for parameter inference (likelihood) is not only poorly correlated with a human-centered measurement of topic coherence, but *inversely* correlated. Nonetheless, most topic models are still trained using methods that optimize likelihood (McAuliffe and Blei, 2008; Nguyen et al., 2013).

We take the logical next step suggested when you bring together the insights of Goodman (1996) and Chang et al. (2009), namely incorporating an approximation of human topic interpretability into the topic model optimization process in a way that is effective and more straightforward than previous methods (Newman et al., 2011). We take advantage of the human-centered evaluation of Chang et al. (2009), which can be reasonably approximated using an automatic metric based on word associations derived from a large, more general corpus (Lau et al., 2014). We exploit LDA and its Bayesian formulation by bringing word associations into the picture using a prior—specifically, we use external lexical association to create a tree structure and then use *tree* LDA (Boyd-Graber et al., 2007, tLDA), which derives topics using a given tree prior.

We construct tree priors with combinations of two types of word association scores (skip-gram probability (Mikolov et al., 2013) and G2 likelihood ratio (Dunning, 1993)) and three construction algorithms (two-level, hierarchical clustering with and without leaf duplication). Then tLDA identifies topics with these tree priors in Amazon reviews and the 20NewsGroups datasets. tLDA topics are more coherent compared with "vanilla" LDA topics, while retaining and often slightly improving topics' extrinsic performance as features for supervised classification. Our approach can be viewed as a form of adaptation, and the flexibility of the tree prior approach—amenable to *any* kind of association score—suggests that there are many directions to pursue beyond the two flavors of association explored here.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1901–1906
Copenhagen, Denmark, September 7–11, 2017.

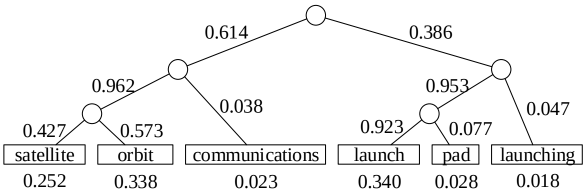


Figure 1: An example of a tree prior (the tree structure) and gold posterior edge and word probabilities learned by tLDA. Numbers beside the edges denote the probability of moving from the parent node to the child node. A word's probability, i.e., the number below the word, is the product of probabilities moving from the root to the leaf.

2 Tree LDA: LDA with Tree Priors

Tree priors organize the vocabulary of a dataset in a tree structure, contrasting with introducing topic correlations (Blei and Lafferty, 2007; He et al., 2017). Words are located at the leaf level and share ancestor internal nodes. In our use of tree priors, if two words have a lower association score, their common ancestor node will be closer to the root node, e.g., contrast (orbit, satellite) with (orbit, launch) in Figure 1.

Tree LDA (Boyd-Graber et al., 2007, tLDA) is an LDA extension that creates topics from a tree prior. A topic in tLDA is a multinomial distribution over the paths from the root to leaves. An internal node, i.e., the circles in Figure 1, is a multinomial distribution over its child nodes. The probability of a path is the product of probabilities of picking the nodes in the path, e.g., $\Pr(\text{satellite}) = 0.614 \times 0.962 \times 0.427 \approx 0.252$. Thus two paths with shared nodes have correlated weights in a topic. The generative process of tLDA is:

1. For topics $k \in \{1, \ldots, K\}$ and internal nodes n_i
   (a) Draw child distribution[1] $\boldsymbol{\pi_{k,i}} \sim \text{Dir}(\beta)$
2. For each document $d \in \{1, \ldots, D\}$
   (a) Draw topic distribution $\boldsymbol{\theta_d} \sim \text{Dir}(\alpha)$
   (b) For each token $t_{d,n}$ in document d
      i. Draw topic assignment $z_{d,n} \sim \text{Mult}(\boldsymbol{\theta_d})$
      ii. Draw path $y_{d,n}$ to word $w_{d,n}$ with probability $\prod_{(i,j)\in y_{d,n}} \pi_{z_{d,n},i,j}$

tLDA can perform different tasks using different tree priors. If we encode synonyms in the tree prior, tLDA disambiguates word senses (Boyd-Graber et al., 2007). With word translation priors, it is a multilingual topic model (Hu et al., 2014).

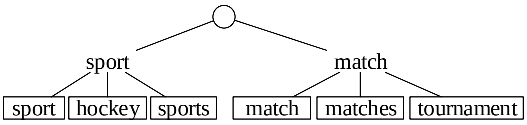


Figure 2: A two-level tree example with $N = 2$. The words in the internal nodes denote *concepts* and have no effect in tLDA.

3 Tree Prior Construction from Word Association Scores

A two-level tree is the most straightforward construction.[2] Each internal node, n_i, is a *concept* associated with a word v_i in the vocabulary. Then we sort all other words in descending order of their association scores with v_i and select the top N words (we use $N = 10$) as n_i's child leaf nodes. n_i has an additional child node which represents v_i, to ensure that every word appears at the leaf level at least once (Figure 2).[3] Thus, if the vocabulary size is V, there will be a total of $(N + 1)V$ leaf nodes.

3.1 Hierarchical Clustering (HAC)

While a two-level tree is bushy (high branching factor) and flat, hierarchical agglomerative clustering (Lukasová, 1979, HAC) reduces the number of leaf nodes and encodes levels of word association information in its hierarchy (Figure 1).

The HAC process starts from V clusters representing the V words in the vocabulary. It then repeatedly merges the two clusters with the highest association score until there is only one cluster left. If at least one of the two clusters, c_i and c_j, has multiple words, their association score is the average association score of the pairwise words from the two clusters:

$$S(c_i, c_j) = \frac{1}{|c_i||c_j|} \sum_{w_{i'} \in c_i} \sum_{w_{j'} \in c_j} S(w_{i'}, w_{j'}). \quad (1)$$

3.2 HAC with Leaf Duplication (HAC-LD)

HAC might merge words with multiple senses. For example, the word "spring" could mean either a season (similar to "summer") or a place with water (similar to "lake"). Assigning "spring" to either side will cause information loss on the other side.

To alleviate this problem, we first pair every word with its most similar word and create a cluster with the pair. Thus "spring" is paired with "summer" and "lake" simultaneously (Figure 3).

[1] Unlike other tree-based topic models such as Andrzejewski et al. (2009), all Dirichlet hyperparameters are the same for all internal nodes. Regardless of cardinality, all Dirichlet parameters are the same scalar β.

[2] The root node is not considered a level.

[3] All tree prior examples are real sub-trees of the priors built on Gigaword.

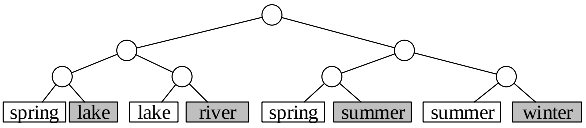


Figure 3: An example of HAC-LD for the words "spring", "summer", and "lake", whose paired words are shaded in gray. HAC-LD alleviates the problem in HAC that a word with multiple senses can only be assigned to a single cluster close to one of its senses.

Corpus	#Vocabulary	#Docs	#Tokens	#Classes
20NG	9,194	18,769	1.75M	20
Amazon	9,410	39,392	1.51M	2

Table 1: Corpus Statistics

4 Experiments

We compute two versions of word association scores from Gigaword, using word2vec (Mikolov et al., 2013) and G2 likelihood ratio (Dunning, 1993).[4] Given the word vectors $\boldsymbol{v_i}$ and $\boldsymbol{v_j}$, which represent words w_i and w_j, their word2vec association score is

$$S(w_i, w_j) = \frac{\exp\left(\boldsymbol{v_i} \cdot \boldsymbol{v_j}\right)}{\sum_k \exp\left(\boldsymbol{v_i} \cdot \boldsymbol{v_k}\right)}. \tag{2}$$

Then we apply the three tree construction algorithms to construct six tree priors. In the two-level trees, the value of N, i.e., the number of child nodes per internal node, is ten.

We use Amazon reviews (Jindal and Liu, 2008) and 20NewsGroups (Lang, 1995, 20NG). We apply the same tokenization and stopword removal methods. We then sort the words by their document frequencies and return the top words, while also removing words that appear in more than 30% of the documents (Table 1).

Both corpora are split into five folds. For classification tasks, each fold is further equally split into a development set and a test set. All the results are averaged across five-fold cross-validation using 20 topics with hyper-parameters $\alpha = \beta = 0.01$. For 20NewsGroups classification, a post's newsgroup is its label. For Amazon reviews, 4–5 star reviews have positive labels, 1–2 stars negative, and reviews with 3 stars are discarded.

Model	Tree	20NG	Amazon
LDA	–	2158.74	999.98
tLDA	G2-2LV	2214.99	1018.72
	G2-HAC	2234.34	1017.17
	G2-HAC-LD	2251.65	1015.06
tLDA	W2V-2LV	2204.94	1016.31
	W2V-HAC	2222.53	1013.07
	W2V-HAC-LD	2234.08	1017.77

Table 2: The average perplexity results on the test sets by various models. LDA gives the lowest perplexity, because tLDA models have constraint from the tree priors and sacrifice the perplexity.

4.1 Perplexity

Before evaluating topic quality, we conduct a sanity check of the models' average perplexity on the test sets (Table 2).

LDA achieves the lowest perplexity among all models on both corpora while tLDA models yield suboptimal perplexity results owing to the constraints given by tree priors. As shown in the following sections, the sacrifice in perplexity brings improvement in topic coherence, while not hurting or slightly improving extrinsic performance using topics as features in supervised classification.

Tree priors built from word2vec generally outperform the ones built using the G2 likelihood ratio. Among the three tree prior construction algorithms, the two-level is the best on the 20NewsGroups corpus. However, there is no such consistent pattern on Amazon reviews.

4.2 Topic Coherence

Instead of manually evaluating topic quality using word intrusion (Chang et al., 2009), we use an automatic alternative to compute topic coherence (Lau et al., 2014). For every topic, we extract its top ten words and compute average pairwise PMI on a reference corpus (Wikipedia as of October 8, 2014).

We include LDA and the latent concept topic model (Hu and Tsujii, 2016, LCTM) as baselines. LCTM also incorporates prior knowledge from word embeddings. It assumes that latent concepts exist in the embedding space and are Gaussian distributions over word embeddings, and a topic is a multinomial distribution over these concepts. We marginalize over concepts and obtain the probability mass of every word in every topic and compare against LDA and tLDA topics.

[4] `https://catalog.ldc.upenn.edu/ldc2011t07`.

Topic	KLD	Model	Words
Christian	0.709	LDA	god, jesus, church, christ, christian, bible, man, christians, lord, sin
		tLDA	god, jesus, bible, christian, christ, church, christians, faith, people, lord
Security	0.720	LDA	key, encryption, chip, clipper, keys, government, public, security, system, law
		tLDA	key, encryption, chip, clipper, government, keys, privacy, security, system, public
Middle East	0.765	LDA	israel, jews, war, israeli, jewish, arab, people, world, peace, muslims
		tLDA	israel, jews, israeli, war, jewish, arab, muslims, people, peace, world
Sports	1.212	LDA	hockey, team, game, play, la, nhl, ca, period, pit, cup
		tLDA	game, team, year, games, play, players, hockey, season, win, baseball
University Research	1.647	LDA	university, information, national, april, states, year, research, number, united, american
		tLDA	university, research, information, april, national, **center**, **science**, year, number, **institute**
Health	1.914	LDA	medical, people, disease, health, cancer, *food*, *sex*, *cramer*, *men*, drug
		tLDA	health, medical, disease, drug, cancer, **patients**, **insurance**, **drugs**, **aids**, **treatment**
Images	1.995	LDA	image, ftp, software, graphics, *mail*, *data*, version, file, pub, images
		tLDA	file, image, **jpeg**, graphics, images, files, format, **bit**, **color**, program
Hardware	2.127	LDA	drive, card, mb, scsi, disk, *mac*, system, *pc*, *apple*, bit
		tLDA	drive, scsi, disk, mb, hard, **drives**, **dos**, **controller**, **ide**, system
People	2.512	LDA	armenian, people, turkish, armenians, armenia, turkey, turks, *didn*, soviet, *time*
		tLDA	armenian, turkish, armenians, armenia, turkey, turks, soviet, people, **russian**, genocide

Table 3: We sort topics into thirds by Kullback-Leibler divergence (KLD): low, medium, and high divergence between vanilla LDA and tLDA. Unique coherent words are in **black and bold**. Unique incoherent words are in *red and italic*. tLDA brings in more topic-relevant words.

Most tLDA models yield more coherent topics (Figure 4). Among all tLDA models, the two-level tree built on word2vec improves the most. LCTM performs poorly: after marginalizing out the concepts on 20NewsGroups, all its topics consist of words like "don", "dodgers", "au", "alot", "people", "alicea", "uw", "arabia", "sps", and "entry" with slight differences in ordering.

To show how subjective topic quality improves over LDA, we extract the topics given by LDA and tLDA (with two-level tree built on word2vec scores) on 20NewsGroups, pair them, and sort the pairs based on KL divergence (KLD). In Table 3, we select and present three topics from each of the top, middle, and bottom third of the sorted topics.

Topics with low KLD (Christian, Security, and Middle East) do not differ significantly. Although the topics of Sports have medium KLD and quite different words, they are generally coherent. As the KLD increases, tLDA topics have more coherent words. In University Research topics, tLDA includes more research-related words, e.g., "center", "science", and "institute". In Health topics, the tLDA topic has more coherent words like "patients", "insurance", "aids", and "treatment", while LDA includes less relevant words, e.g., "food", "sex", and "cramer".

In the topics with large KLD, tLDA topics are also more coherent. For instance, in the Images topics, the LDA topic contains less relevant words like "mail" and "data", while the tLDA topic mostly consists of words related to images, and even includes words like "jpeg", "color", and "bit" that are not among the top words in the LDA topic. In the topics for Hardware, there are more words closer to the hardware level for tLDA, e.g., "drives", "dos", "controller", and "ide", in contrast to LDA, e.g., "mac", "pc", and "apple". tLDA also ranks hardware-related words higher. For instance, "scsi" and "disk" come before "mb". The words in the topics for People are generally coherent, except "didn" and "time" in the LDA topic.

4.3 Extrinsic Classification

To extrinsically evaluate topic quality, we use binary and multi-class classification on Amazon reviews and 20NewsGroups corpora using SVM-light (Joachims, 1998) and SVM-multiclass.[5] We tune the parameter C, the trade-off between training error and margin, on the development set and apply the trained model with the best performance on the development set to the test set. The classification accuracies are given in Table 4.

We compare the accuracies of features of bag-of-words (BOW) and LDA/LCTM/tLDA topics. For the tLDA models with two-level and HAC-LD tree priors, the path assignment is an additional feature.[6] We also include the features of BOW and the average word vector for the document (BOW+VEC).

[5] SVM-light: http://svmlight.joachims.org/. SVM-multiclass: https://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html.

[6] tLDA models with HAC prior do not have this feature, because the paths have a 1-to-1 mapping with the vocabulary.

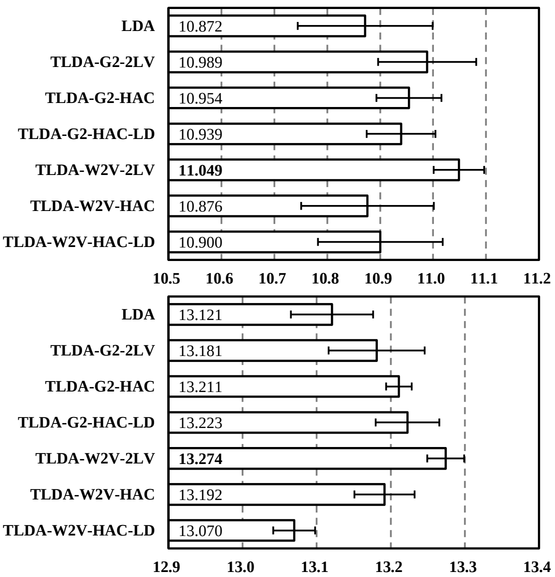


Figure 4: Average PMI of top 10 words in topics given by models on 20NewsGroups (upper) and Amazon (lower). Most tLDA topics are more coherent than LDA topics. The PMI of LCTM are too low to be included: 8.862±0.657 on 20NewsGroups and 6.340±1.208 on Amazon reviews.

Features based on most tLDA topics perform at least as well as LDA-based topic features; with no statistically significant differences, our tree priors do not sacrifice extrinsic performance for improving topic coherence. In addition, the path assignment feature improves topical classification but not sentiment classification. Although the word2vec feature (BOW+VEC) performs the best on Amazon reviews, it lacks the interpretability of topic models.

4.4 Learned Trees

Tree-based topics distinguish polysemous words. In Figure 5, the upper sub-tree comes from the Politics topic (“president”, “people”, “clinton”, “myers”, “money”, etc.) where “pounds” is more likely to be reached in the sense of British currency. In the Health topic (Table 3), “pounds” is more associated with weights (lower tree).

5 Conclusions and Future Work

Combining topic models and vector space models is an emerging area. We introduce a method that is simpler and more flexible than previous work (Hu and Tsujii, 2016), and although we extract prior knowledge from word vectors, our model is not restricted to this and can use *any* word association scores. Our model yields more coherent topics and maintains extrinsic performance, and in addition it is less computationally costly.[7]

Model	Tree	Path	20NG	Amazon
BOW	–	–	86.64	86.73
BOW+VEC	–	–	86.59	**87.30**
LDA	–	–	86.67	86.99
LCTM	–	–	86.52	86.83
tLDA	W2V-2LV	N	86.75	87.07
		Y	86.73	87.13
	W2V-HAC	–	86.79	87.19
	W2V-HAC-LD	N	86.73	87.02
		Y	86.94	86.88
tLDA	G2-2LV	N	86.82	87.15
		Y	**86.96**	87.05
	G2-HAC	–	86.63	87.11
	G2-HAC-LD	N	86.73	87.07
		Y	86.91	86.94

Table 4: Accuracies of topical classification on 20NewsGroups and sentiment analysis on Amazon reviews. Although not significantly improving the performance, tLDA topics at least do not hurt.

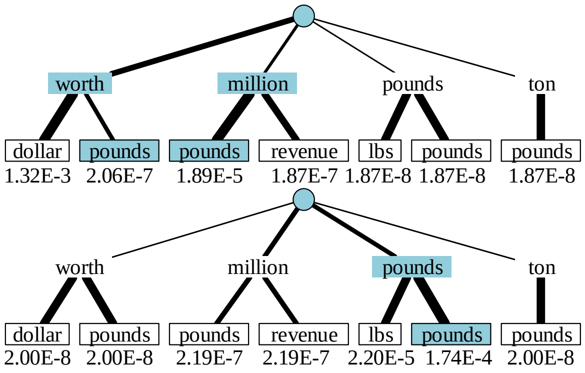


Figure 5: Sub-trees for “pounds” in two topics, from 20NewsGroups corpus using two-level tree prior from word2vec. “Pounds” is more associated with British currency in Politics (upper), while closer to weight in Health (lower). High probability *paths* are shaded; high probability *edges* have thicker lines.

We plan to merge tree prior construction and the topic modeling into a unified framework (Teh et al., 2007; Görür and Teh, 2009; Hu et al., 2013). This will allow tree priors to change along with the topics they produce instead of using a static one constructed *a priori*.

[7] tLDA Java implementation converges in twelve hours; LCTM needs sixty hours (2.8GHz Intel Xeon and 110G RAM).

Acknowledgement

We thank the anonymous reviewers for their insightful and constructive comments. This research has been supported in part, under subcontract to Raytheon BBN Technologies, by DARPA award HR0011-15-C-0113. Boyd-Graber is also supported by NSF grants IIS-1320538, IIS-1409287, IIS-1564275, IIS-1652666, and NCSE-1422492. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsors.

References

David Andrzejewski, Xiaojin Zhu, and Mark Craven. 2009. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In *Proceedings of the International Conference of Machine Learning*.

David M. Blei and John D. Lafferty. 2007. A correlated topic model of science. *The Annals of Applied Statistics*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*.

Jordan Boyd-Graber, David M. Blei, and Xiaojin Zhu. 2007. A topic model for word sense disambiguation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-Graber, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*.

Joshua Goodman. 1996. Parsing algorithms and metrics. In *Proceedings of the Association for Computational Linguistics*.

Dilan Görür and Yee Whye Teh. 2009. An efficient sequential Monte Carlo algorithm for coalescent clustering. In *Proceedings of Advances in Neural Information Processing Systems*.

Junxian He, Zhiting Hu, Taylor Berg-Kirkpatrick, Ying Huang, and Eric P. Xing. 2017. Efficient correlated topic modeling with topic embedding. In *Knowledge Discovery and Data Mining*.

Weihua Hu and Jun'ichi Tsujii. 2016. A latent concept topic model for robust topic inference using word embeddings. In *Proceedings of the Association for Computational Linguistics*.

Yuening Hu, Jordan Boyd-Graber, Hal Daumé III, and Z. Irene Ying. 2013. Binary to bushy: Bayesian hierarchical clustering with the Beta coalescent. In *Proceedings of Advances in Neural Information Processing Systems*.

Yuening Hu, Ke Zhai, Vlad Eidelman, and Jordan Boyd-Graber. 2014. Polylingual tree-based topic models for translation domain adaptation. In *Proceedings of the Association for Computational Linguistics*.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of ACM International Conference on Web Search and Data Mining*.

Thorsten Joachims. 1998. Making large-scale SVM learning practical. Technical report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the International Conference of Machine Learning*.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the Association for Computational Linguistics*.

Alena Lukasová. 1979. Hierarchical agglomerative clustering procedure. *Pattern Recognition*.

Jon D. McAuliffe and David M. Blei. 2008. Supervised topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems*.

David Newman, Edwin V. Bonilla, and Wray Buntine. 2011. Improving topic coherence with regularized topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Viet-An Nguyen, Jordan Boyd-Graber, and Philip Resnik. 2013. Lexical and hierarchical topic regression. In *Proceedings of Advances in Neural Information Processing Systems*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics*.

Yee Whye Teh, Hal Daumé III, and Daniel M. Roy. 2007. Bayesian agglomerative clustering with coalescents. In *Proceedings of Advances in Neural Information Processing Systems*.

Finding Patterns in Noisy Crowds: Regression-based Annotation Aggregation for Crowdsourced Data

Natalie Parde and **Rodney D. Nielsen**
Department of Computer Science and Engineering
University of North Texas
{natalie.parde,rodney.nielsen}@unt.edu

Abstract

Crowdsourcing offers a convenient means of obtaining labeled data quickly and inexpensively. However, crowdsourced labels are often noisier than expert-annotated data, making it difficult to aggregate them meaningfully. We present an aggregation approach that learns a regression model from crowdsourced annotations to predict aggregated labels for instances that have no expert adjudications. The predicted labels achieve a correlation of 0.594 with expert labels on our data, outperforming the best alternative aggregation method by 11.9%. Our approach also outperforms the alternatives on third-party datasets.

1 Introduction

Publicly-available labeled datasets are scarce for many NLP tasks, and crowdsourcing services such as Amazon Mechanical Turk[1] (AMT) offer researchers a quick, inexpensive means of labeling their data. However, workers employed by these services are typically unfamiliar with the annotation tasks, and they may have little motivation to perform high-quality work due to factors such as low pay and anonymity. To further complicate matters, some workers may produce spam or malicious responses. Thus, it is not uncommon for workers to correlate poorly with one another.

Researchers using crowdsourcing services commonly aggregate the labels they receive via simple strategies such as using the majority or average label. These methods are best suited for simple, straightforward tasks; with noisier data such as that which may be obtained for more difficult or subjective tasks, these strategies may produce skewed labels that misrepresent the instance. Thus, it is desirable to devise more effective aggregation strategies that consider factors such as label distribution and worker quality, while still avoiding manual adjudication of all instances.

In this work, our contributions are as follows: (1) we develop a regression-based method for automatically aggregating crowdsourced annotations of varying quality, with poor agreement and minimal expert-adjudicated data, that addresses multiple potential flaws or biases in non-expert human annotation. To do so, we (2) crowdsource annotations for a difficult NLP task, metaphor novelty scoring, and (3) describe a process by which we automatically detect untrustworthy workers. We then (4) introduce a feature set that captures label distribution and trustworthiness, and extract the features from our crowdsourced annotations. Finally, (5) we train a regression model that predicts aggregated labels for unseen instances and compare the predictions to expert annotations, finding that our method outperforms the best alternative approach. We evaluate our approach both on our data and on existing crowdsourcing datasets. All datasets and source code are available for the research community to improve on our results.[2]

2 Related Work

Several methods have been proposed to identify low-quality workers in crowdsourced data. Jagabathula et al. (2016) filtered adversarial workers in binary labeling tasks by identifying those with outlier labeling patterns, and Lin et al. (2014) identified when additional labels for binary tasks should be crowdsourced to optimize classifier accuracy. Unlike these approaches, our filtering algorithm is suitable for multi-class annotation tasks.

[1] www.mturk.com

[2] Our data can be downloaded at http://hilt.cse.unt.edu/resources.html, and our source code is available at https://github.com/natalieparde/label-aggregation.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1907–1912
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Various methods have also been explored as intelligent modes of label aggregation. Most (Snow et al., 2008; Raykar et al., 2010; Karger et al., 2011; Liu et al., 2012; Hovy et al., 2013; Felt et al., 2014; Huang et al., 2015) have built upon the probabilistic item-response model first proposed by Dawid and Skene (1979), which simultaneously estimates annotator quality and aggregated labels using an expectation-maximization algorithm. MACE (Hovy et al., 2013) is a popular implementation inspired by this that aggregates labels as a function of the annotation and a learned binary variable indicating whether the annotator is a spammer. We posit that although annotator quality is an important factor in predicting accurate aggregations, the interplay between it and other factors is more nuanced. Thus, rather than adapting the item-response method, our learning approach incorporates features that address multiple potential flaws or biases in crowdsourced annotations.

Some researchers have also used data-aware approaches to predict aggregations (Raykar et al., 2010; Felt et al., 2014, 2015, 2016). We do not use the data itself in this work, to avoid skewing labels in a way that makes it trivial to learn classifiers based on the same data. To the best of our knowledge, our work is the first to frame label aggregation as a regression task, with features based solely on workers and their labels, that learns entirely from a small amount of expert-adjudicated crowdsourced annotations.

3 Methods

3.1 Data Collection

We evaluated our approach on our new metaphor novelty dataset, as well as on third-party datasets. To build our dataset, we crowdsourced annotations for 3112 potentially metaphoric word pairs, and randomly divided the instances into training (1036), validation (1038), and test (1038) subsets. We developed features and selected our regression algorithm using the training and validation sets only; the test set was withheld until the evaluation.

3.1.1 Annotation Task

Instances were comprised of pairs of words from 1840 sentences in the VU Amsterdam Metaphor Corpus (VUAMC) (Steen et al., 2010). The VUAMC consists of documents for which individual words are labeled as metaphors. The novelty of those metaphors varies widely, from highly conventional to quite novel. Each sentence for which we collected annotations contained a content word (noun, verb, adjective, or adverb) labeled as being metaphoric, and one or more other content words or personal pronouns that were syntactically related to the metaphoric word. Word pairs containing a metaphoric word and a syntactically-related content word or personal pronoun were considered instances. AMT workers ("Turkers") were asked to score each instance on a discrete scale from non-metaphoric (0) to highly novel metaphor (3). Some examples are shown in Table 1.[3]

Example	**Score**
Alice looked up, and there stood the Queen in front of them, with her arms folded, *frowning* like a **thunderstorm**.	Novel Metaphor (3)
'Once,' said the Mock Turtle at last, with a **deep** *sigh*, 'I was a real Turtle.'	Conventional Metaphor (1)
A large rose-tree **stood** near the *entrance* of the garden: the roses growing on it were white, but there were three gardeners at it, busily painting them red.	Non-Metaphor (0)

Table 1: Sample word pairs provided to Turkers.

Instances were grouped into Human Intelligence Tasks (HITs) containing all instances associated with 10 sentences each. Five worker assignments were requested per HIT, and Turkers were paid $0.20 per HIT. Overall, 237 Turkers annotated 942 assignments, with an average correlation of 0.269 per HIT (the poor agreement suggests this is a very difficult annotation task). An expert adjudicated all 3112 instances; those labels were considered the gold standard.

3.1.2 Data Filtering

Spam and malicious workers were identified during data collection using a filtering algorithm that compared annotations with those completed by "potentially good annotators" (PGA). Alg. 1 describes this process. Letting H_i be a set of HITs collected, A_i be the set of annotators who annotated H_i, and A=$\cup(A_1, \ldots, A_j)$ be the set of all annotators, the algorithm computes three sets of annotators: good annotators (GA), spammers or malicious annotators (Bad Robots, or BR), and annotators of currently unknown quality UQA.

$\text{R}(a_j, a_k)$ computes the correlation coefficient between two annotators a_j and a_k, where a_k is a potentially good annotator whose annotations overlap with a_j's, and $\text{AVG\_R}(a_j)$ computes the average correlation between a_j and all a_k. HITs

[3] Sentences are from Lewis Carroll's *Alice in Wonderland*.

Algorithm 1 Worker Filtering for Annotation Set i

```
  $PGA \leftarrow A \setminus BR$
  **repeat**
      **for** $a_j$ in $A$ **do**
          $A^j \leftarrow \{a \in PGA\}$ who annotated $\geq 1$ unfiltered
                  HIT in common with $a_j$
          **for** $a_k$ in $A^j$ **do**
              $r_{j,k} \leftarrow \mathrm{R}(a_j, a_k)$
          $r_j \leftarrow \mathrm{AVG\_R}(a_j)$
      $B^- \leftarrow \{a_j \in A | r_j < 0.0\}$
      $B^0 \leftarrow \{a_j \in A | r_j == 0.0$ or $r_j == \infty\}$[4]
      $B^+ \leftarrow \{a_j \in A\}$, of size $|B^-|$, with the lowest $r_j >$
  0.0
      $B^{<.1} \leftarrow \{a_j \in A | r_j < 0.1\}$
      $PGA = A - (B^- + B^0 + B^+ + B^{<.1})$
  **until** convergence or iterations = max
  $GA \leftarrow \{a_j \in A | r_j > 0.35\}$
  $BR \leftarrow B^- + B^0 + \mathrm{BOTTOM}(\mathrm{ROUND}(\frac{2}{3}|B^-|), B^+)$
```

completed under a minimum time threshold were also filtered. Following algorithm completion, filtered HITs and unpaid HITs from members of BR were rejected, and annotators in BR were disqualified from accepting future HITs. 116 total assignments were rejected by the filtering algorithm. Annotators in UQA (UQA=A−GA−BR) who had completed $\geq$ 2 HITs and had an $r_j < 0.1$ were also disqualified. All other HITs were accepted.

3.2 Features

We designed features to capture the distribution and trustworthiness of crowdsourced labels for each instance. The features are described in Table 2. Annotations are designed to provide the regression algorithm with label distributions based on label value and worker trustworthiness. Avg. R features are intended to further clarify worker quality, and Avg. R (Good) is meant to provide a more selective view of the same characteristic. Avg., Weighted Avg., and Weighted Avg. (Good) allow the regressor to consider three different versions of a popular aggregation strategy, and finally, HIT R supplies the algorithm with an estimate of agreement on the current instance to consider when making its prediction.

Feature	Description
Annotations	From highest to lowest label, the five annotations for the instance.[5]
Avg. R	For each annotator, in order of label value, his/her avg. correlation with other workers across all instances he/she annotated.[5]
Avg. R (Good)	Avg. R in which each annotator is compared only to annotators with r_j>0.35. If the annotator has no overlapping annotations with those, Avg. R is repeated.
Avg.	Average of the five Annotations.
Weighted Avg.	Let l_i be the i^{th} Annotation, and r_i be its annotator's Avg. R. Then, Weighted Avg. $= \frac{\sum_{i=1}^{5}(l_i \times r_i)}{\sum_{i=1}^{5} r_i}$.
Weighted Avg. (Good)	Similar to Weighted Avg., with weights (r_i) taken from Avg. R. (Good) instead of Avg. R.
HIT R	The average weighted correlation among annotators for the HIT containing the instance. Letting $w_{i,j}$ be the weight for a pair of annotators equal to $\frac{r_i + r_j}{2}$, where r_i and r_j are the Avg. R associated with annotators a_i and a_j, $r_{i,j}$ be the correlation between annotators a_i and a_j for the HIT, and P contain all annotator pairs (a_i, a_j) for the HIT, HIT R $= \frac{\sum_{p \in P} r_{i,j} \times w_{i,j}}{\sum_{p \in P} w_{i,j}}$

Table 2: Features used.

	Affect (Emo.)	Affect (Val.)	WebRel	Ours
Instances	600	100	2439	3112
Annotators	38	38	722	237
Annotators / Instance	10	10	5	5
Label Range	0-100	-100-100	0-2	0-3

Table 3: Dataset Details

3.3 Regression Algorithm

The approach utilizes a random subspace regressor, which was selected based on its performance on the training and validation data relative to a large variety of other regression algorithms. Random subspace is similar in nature to bagging and random forests, using multiple decision trees constructed from subsets of features selected randomly without replacement to make its predictions (Ho, 1998). We used the implementation from the Weka library (Frank et al., 2016), with Weka's REPTree classifier as the base decision tree model.

4 Evaluation

4.1 Other Datasets

In addition to evaluating our approach on our data, we evaluate it on three existing crowdsourcing datasets that differ in terms of their size, noise level, and number of annotators. Details about each dataset are shown in Table 3, with additional information below. Each third-party dataset was randomly divided into 66% training and 34% test.

[4] Turkers who assigned the same label to every instance, or whose assignments had already been filtered for some other reason (e.g., violating the minimum time threshold).

[5] We also include a second copy of these features ordered by the annotators' average r values.

Affect (Emotion and Valence). Affect (Emotion) and Affect (Valence) were created for Snow et al.'s (2008) work, and contain emotion (*anger*, *fear*, *disgust*, *joy*, *sadness*, and *surprise*) and valence ratings for 100 headlines from the SemEval affective text annotation task (Strapparava and Mihalcea, 2007) test set. Annotations indicate the degree of emotion in an emotion-headline pair (Affect (Emotion)) and the overall positive or negative valence of a headline (Affect (Valence)). Snow et al. report an average correlation among annotators of 0.669 (emotion) and 0.844 (valence).

WebRel. WebRel was originally created for the TREC 2010 Relevance Feedback Track (Buckley et al., 2010), and its annotations indicate the relevance of web documents retrieved for queries. The full dataset contains crowdsourced annotations for 20,232 topic-document pairs; 3277 of those pairs additionally have gold-standard labels. The number of annotations collected per instance varied. We used the subset of instances with gold standard labels and at least five annotations, and reconstructed their HIT groupings based on the workers that annotated each instance (we assumed all instances annotated by the exact same set of workers were originally from the same HIT). Average correlation per HIT was 0.102 (quite noisy).

4.2 Experimental Setup

We compare our approach to a number of alternative methods, detailed with justifications in Table 4. The alternatives are popular aggregation techniques that address different potential flaws in non-expert annotation. We train our approach on the training (and validation, for our dataset) data, and test on the test set. Since MACE (used for *Item-Response*) learns from and outputs predictions for the same data, we provide it with the entire dataset (training, validation if available, and test), but report its results for the test instances only. We provide input to MACE in an n-dimensional sparse matrix (1 row per instance and 1 column per each of n distinct annotators in the dataset, with filled values only for the annotators who provided annotations for that instance), since the approach requires knowledge of which annotator provided each annotation to function properly.[6]

Approach	Description
Majority Vote	The most frequent label given by annotators for the instance. Ties were broken by taking the highest of the tied labels—assumes the most popular opinion should be trusted.
Highest	The highest label for the instance—assumes those who see a metaphor should be trusted.
Item-Response	The prediction expected from an item-response model. We use MACE (Hovy et al., 2013) to generate predictions since it is a well-documented item-response approach that is publicly available online.
Mode Average	The real-valued average of the mode(s) of the instance's labels (if only one mode, this feature is that mode)—assumes popular opinions should be trusted, and equally popular opinions are equally trustworthy.
Average	The average of all five labels—assumes each annotator's opinion is equally valid.
Rule-Based	Assigns a value of 0 if 4+ annotators labeled the instance as such; otherwise, takes the avg. non-zero label—assumes annotators frequently miss tricky or subtle instances.

Table 4: Alternative Approaches.

We also evaluate the performance of different feature subsets on our data. *All–Averages* contains all features except for AVG., WEIGHTED AVG., and WEIGHTED AVG. (GOOD). Each other subset contains all features except for the respective feature type noted from Table 2. The correlation coefficient (r) and root mean squared error (RMSE) were recorded for each test condition since our estimator produced continuous-valued scores. Since *Mode Average*, *Average*, and *Rule-Based* result in continuous values and *Majority Vote*, *Highest*, and *Item-Response* result in discrete values, we present two versions of our results; in one, predictions were rounded to the nearest integer (forcing a 0, 1, 2, or 3) and in the other, they were left as-is. For the discrete approaches on our data, we also report accuracy.

4.3 Results

The results are presented in Tables 5, 6, and 7. Table 5 compares our method with each alternative approach on our data, and Table 6 compares our method with the alternatives on each third-party dataset. Table 7 shows the results of the feature ablation. On our dataset, our approach outperformed all other approaches, with r = 0.594 with the gold standard and RMSE (0-3) = 0.605. This represented correlation improvements of 18.6%, 11.9%, and 69.2% relative to the continuous alternative approaches (*Mode Average*, *Average*, and *Rule-Based*, respectively). The

[6] Note: Item-response approaches are better-suited to scenarios in which fewer workers annotate more instances each, but our results would also improve under such circumstances where a worker's trustworthiness, as measured by average r value, is more reliable.

Method	*r*	**RMSE**	**Acc.**
Majority Vote	0.443	1.011	0.536
Highest	0.295	1.701	0.183
Item-Response	0.362	1.083	0.483
Ours (Rounded)	**0.490**	**0.690**	**0.600**
Mode Average	0.501	0.836	—
Average	0.531	0.743	—
Rule-Based	0.351	1.126	—
Ours (Continuous)	**0.594**	**0.605**	**—**

Table 5: Comparison with alternative methods.

	Method	*r*	**RMSE**
Affect (Emotion)	Majority Vote	0.510	23.2
	Highest	0.416	52.4
	Item-Response	0.526	21.8
	Ours (R)	**0.578**	**16.6**
	Mode Average	0.506	21.9
	Average	**0.613**	16.7
	Rule-Based	0.462	26.5
	Ours (C)	0.578	**16.6**
Affect (Valence)	Majority Vote	0.423	50.1
	Highest	0.573	75.3
	Item-Response	0.483	46.0
	Ours (R)	**0.938**	**18.4**
	Mode Average	0.644	37.4
	Average	0.926	22.4
	Rule-Based	0.913	19.7
	Ours (C)	**0.938**	**18.4**
WebRel	Majority Vote	0.325	1.0
	Highest	0.219	1.2
	Item-Response	0.385	0.9
	Ours (R)	**0.412**	**0.8**
	Mode Average	0.350	0.9
	Average	0.372	0.8
	Rule-Based	0.282	0.9
	Ours (C)	**0.523**	**0.7**

Table 6: Comparison on third-party datasets.

rounded predictions also outperformed all discrete alternatives (*Majority Vote*, *Highest* and *Item-Response*) with relative correlation improvements of 10.6%, 66.1%, and 35.4%, respectively. All approaches had strong positive statistically significant (p<<0.0001) correlations and the improvement of our results over the alternatives was statistically significant (p<<0.0001).

On WebRel and Affect (Valence), our approach outperformed all other approaches for both the discrete and continuous conditions. On Affect (Emotion), our approach outperformed all alternatives for the discrete condition and had a lower RMSE than all other approaches for the continuous condition (relative reductions in error to Rule-Based, Average, and Mode Average were 37.4%, 0.6%, and 24.2%, respectively), but the predictions from Average correlated better with the gold standard than did those of our approach.

	Rounded		**Continuous**	
Feature Set	*r*	**RMSE**	*r*	**RMSE**
All	0.490	**0.690**	**0.594**	**0.605**
All−Annotations	0.440	0.716	0.557	0.627
All−Avg. R	0.480	0.701	0.581	0.611
All−Avg. R (G.)	**0.494**	0.692	0.582	0.611
All−Averages	0.465	0.703	**0.594**	0.607
All−HIT R	0.486	0.693	0.587	0.608

Table 7: Feature subset performance comparison.

Interestingly, Table 7 shows that the discrete version of our approach performed slightly better when the features indicating annotators' correlations with good annotators were removed; this was not the case for the continuous-labeled version. The raw annotations themselves were the most valuable features for both cases. Their removal led to a correlation reduction of 10.2% (rounded) and 6.2% (continuous) relative to using all features.

The results suggest that our approach is a suitable means of automatically aggregating noisy crowdsourced labels, and that reasonable results can be obtained even when training on only a small amount of expert-adjudicated instances. Further, the performance of the alternative approaches suggests that typical aggregation techniques may be less suitable for tasks with many workers who completed relatively few annotations.

5 Conclusion

In this work, we present a regression-based aggregation method that addresses multiple potential flaws or biases in non-expert human annotation. We show that the predictions from our approach correlate at r=0.594 with expert adjudications for a noisy, difficult task, outperforming the best alternative approach by 11.9% on our data and by up to 63.7% on third-party crowdsourcing datasets. This improvement shows that a learning approach can overcome some of the challenges faced by simple label aggregation techniques for these types of tasks. Our data and source code is publicly available for further research by others.

Acknowledgments

This material is based upon work supported by the NSF Graduate Research Fellowship Program under Grant 1144248, and the NSF under Grant 1262860. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Chris Buckley, Matthew Lease, Mark D Smucker, Hyun Joon Jung, Catherine Grady, et al. 2010. Overview of the trec 2010 relevance feedback track (notebook). In *The Nineteenth Text Retrieval Conference (TREC) Notebook*, pages 88–90. Association for Computational Linguistics.

A. P. Dawid and A. M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.

Paul Felt, Robbie Haertel, Eric Ringger, and Kevin Seppi. 2014. Momresp: A bayesian model for multi-annotator document labeling. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Paul Felt, Eric Ringger, Jordan Boyd-Graber, and Kevin Seppi. 2015. Making the most of crowdsourced document annotations: Confused supervised lda. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 194–203, Beijing, China. Association for Computational Linguistics.

Paul Felt, Eric Ringger, and Kevin Seppi. 2016. Semantic annotation aggregation with conditional crowdsourcing models and word embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1787–1796, Osaka, Japan. The COLING 2016 Organizing Committee.

Eibe Frank, Mark A. Hall, and Ian H. Witten. 2016. The weka workbench. In *Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques"*, fourth edition. edition. Morgan Kaufmann.

Tin Kam Ho. 1998. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with mace. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Ziheng Huang, Jialu Zhong, and Rebecca J. Passonneau. 2015. Estimation of discourse segmentation labels from crowd data. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2190–2200, Lisbon, Portugal. Association for Computational Linguistics.

Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. 2016. Identifying unreliable and adversarial workers in crowdsourced labeling tasks. *Journal of Machine Learning Research*, 17(1).

David R. Karger, Sewoong Oh, and Devavrat Shah. 2011. Iterative learning for reliable crowdsourcing systems. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1953–1961. Curran Associates, Inc.

Christopher H Lin, Daniel S Weld, et al. 2014. To re (label), or not to re (label). In *Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing (HCOMP 2014)*.

Qiang Liu, Jian Peng, and Alexander T Ihler. 2012. Variational inference for crowdsourcing. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 692–700. Curran Associates, Inc.

Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *Journal of Machine Learning Research*, 11(Apr):1297–1322.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 254–263, Stroudsburg, PA, USA. Association for Computational Linguistics.

Gerard J Steen, Aletta G Dorst, J Berenike Herrmann, Anna Kaal, Tina Krennmayr, and Trijntje Pasma. 2010. *A method for linguistic metaphor identification: From MIP to MIPVU*, volume 14. John Benjamins Publishing.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 70–74. Association for Computational Linguistics.

CROWD-IN-THE-LOOP: A Hybrid Approach for Annotating Semantic Roles

Chenguang Wang[†] **, Alan Akbik**[‡*]**, Laura Chiticariu**[†] **, Yunyao Li**[†] **, Fei Xia**[§] **, Anbang Xu**[†]
[†]IBM Research - Almaden
[‡]Zalando Research, Berlin
[§]Department of Linguistics, University of Washington
chenguang.wang@ibm.com, alan.akbik@zalando.de
{chiti, yunyaoli, anbangxu}@us.ibm.com, fxia@uw.edu

Abstract

Crowdsourcing has proven to be an effective method for generating labeled data for a range of NLP tasks. However, multiple recent attempts of using crowdsourcing to generate gold-labeled training data for semantic role labeling (SRL) reported only modest results, indicating that SRL is perhaps too difficult a task to be effectively crowdsourced. In this paper, we postulate that while producing SRL annotation does require expert involvement in general, a large subset of SRL labeling tasks is in fact appropriate for the crowd. We present a novel workflow in which we employ a classifier to identify difficult annotation tasks and route each task either to experts or crowd workers according to their difficulties. Our experimental evaluation shows that the proposed approach reduces the workload for experts by over two-thirds, and thus significantly reduces the cost of producing SRL annotation at little loss in quality.

1 Introduction

Semantic role labeling (SRL) is the task of labeling the predicate-argument structures of sentences with semantic *frames* and their *roles* (Baker et al., 1998; Palmer et al., 2005). It has been found useful for a wide variety of NLP tasks such as question-answering (Shen and Lapata, 2007), information extraction (Fader et al., 2011) and machine translation (Lo et al., 2013). A major bottleneck impeding the wide adoption of SRL is the need for large amounts of labeled training data to capture broad-coverage semantics. Such data requires trained experts and is highly costly to produce (Hovy et al., 2006).

Crowdsourcing SRL *Crowdsourcing* has shown its effectiveness to generate labeled data for a range of NLP tasks (Snow et al., 2008; Hong and Baker, 2011; Franklin et al., 2011). A core advantage of crowdsourcing is that it allows the annotation workload to be scaled out among large numbers of inexpensive *crowd workers*. Not surprisingly, a number of recent SRL works have also attempted to leverage crowdsourcing to generate labeled training data for SRL and investigated a variety of ways of formulating crowdsourcing tasks (Fossati et al., 2013; Pavlick et al., 2015; Akbik et al., 2016). All have found that crowd feedback generally suffers from low inter-annotator agreement scores and often produces incorrect labels. These results seem to indicate that, regardless of the design of the task, SRL is simply too difficult to be effectively crowdsourced.

Proposed Approach We observe that there are significant differences in difficulties among SRL annotation tasks, depending on factors such as the complexity of a specific sentence or the difficulty of a specific semantic role. We therefore postulate that a subset of annotation tasks is in fact suitable for crowd workers, while others require expert involvement. We also postulate that it is possible to use a classifier to *predict* whether a specific task is easy enough for crowd workers.

Based on these intuitions, we propose CROWD-IN-THE-LOOP, a hybrid annotation approach that involves both crowd workers and experts: All annotation tasks are passed through a decision function (referred to as TASKROUTER) that classifies them as either *crowd-appropriate* or *expert-required*, and sent to crowd or expert annotators accordingly. Refer to Figure 1 for an illustration of this workflow.

[*]The work was done while the author was at IBM Research - Almaden.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1913–1922
Copenhagen, Denmark, September 7–11, 2017.

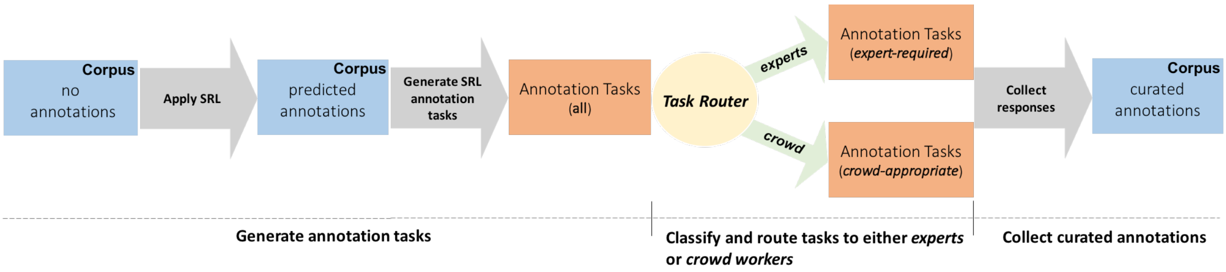

Corpus
no annotations
Apply SRL
Corpus
predicted annotations
Generate SRL annotation tasks
Annotation Tasks (all)
Task Router
experts
crowd
Annotation Tasks (expert-required)
Annotation Tasks (crowd-appropriate)
Collect responses
Corpus
curated annotations
Generate annotation tasks
Classify and route tasks to either experts or crowd workers
Collect curated annotations

Figure 1: Overview of proposed CROWD-IN-THE-LOOP approach for curating SRL annotations.

We conduct an experimental evaluation that shows (1) that we are able to design a classifier that can distinguish between crowd-appropriate and expert-required tasks at very high accuracy (96%), and (2) that our proposed workflow allows us to pass over two-thirds of the annotation workload to crowd workers, thereby significantly reducing the need for costly expert involvement.

Contributions In detail, our contributions are:

- We propose CROWD-IN-THE-LOOP, a novel approach for creating annotated SRL data with both crowd workers and experts. It reduces overall labeling costs by leveraging the crowd whenever possible, and maintains annotation quality by involving experts whenever necessary.
- We propose TASKROUTER, an *annotation task decision function* (or *classifier*), that classifies each annotation task into one of two categories: *expert-required* or *crowd-appropriate*. We carefully define the classification task, discuss features and evaluate different classification models.
- We conduct a detailed experimental evaluation of the proposed workflow against several baselines including standard crowdsourcing and other hybrid annotation approaches. We analyze the strengths and weaknesses of each approach and illustrate how expert involvement is required to address errors made by crowd workers.

Outline This paper is organized as follows: We first conduct a baseline study of crowdsourcing SRL annotation, and analyze the difficulties of relying solely on crowd workers (Section 2). Based on this analysis, we define the classification problem for CROWD-IN-THE-LOOP, discuss the design of our classifier, and evaluate its accuracy (Section 3). We then employ this classifier in the proposed CROWD-IN-THE-LOOP approach and comparatively evaluate it against a number of crowdsourcing and hybrid workflows (Section 4). We discuss related work (Section 5) and conclude the study in Section 6.

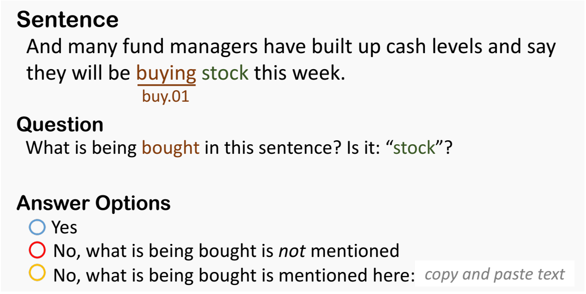
2 Crowdsourcing SRL

We first conduct a baseline study of crowdsourcing SRL. We illustrate how we design and create annotation tasks, how we gather and interpret crowd feedback, and analyze the results of the study to determine the applicability of crowdsourcing for producing SRL annotation.

SRL formalism. In this study, and throughout the paper, we use the PROPBANK formalism of SRL (Palmer et al., 2005), which defines verb-specific frames (BUY.01, BUY.02), frame-specific core roles (**A0** to **A5**), and frame-independent non-core roles (for temporal, location and other contexts).

2.1 Annotation Task Design

To design the annotation task, we replicate a setup proposed in previous work (Akbik et al., 2016) in which crowd workers are employed to *curate* the output of a statistical SRL system. This setup generates annotation tasks as following:



Sentence

And many fund managers have built up cash levels and say they will be buying stock this week.
buy.01

Question

What is being bought in this sentence? Is it: "stock"?

Answer Options

○ Yes
○ No, what is being bought is *not* mentioned
○ No, what is being bought is mentioned here: copy and paste text

Figure 2: Example annotation task, consisting of a *sentence* with predicted role labels, a human readable *question* regarding to one label, and a set of answer *options*. By answering, crowd workers curate a prediction made by the SRL.

Step 1. We use a statistical SRL system to predict SRL labels for a set of sentences (see Figure 1). While state-of-the-art SRL will predict many correct labels, some predicted labels will be incorrect, and some labels will be missing. Annotation tasks are therefore designed to detect and correct precision and recall errors.

Step 2. We generate two types of annotation tasks for the study, namely ConfirmPrediction and AddMissing tasks: (1) The first, ConfirmPrediction tasks, ask users to confirm, reject or correct each predicted frame or role. This type of task addresses *precision* issues in the SRL. We present to workers a human-readable question-answer pair (He et al., 2015) for each predicted label, an example of which is illustrated in Figure 2. (2) The second, AddMissing tasks, address potentially missing annotation, i.e. *recall* issues in the SRL. We generate a question without a suggested answer and ask workers to either confirm that this role does not appear in the sentence, or supply the correct span. We identify potentially missing annotation using PropBank frame definitions; any unseen core role in a sentence is considered potentially missing.

We use a manually created mapping of frame-roles to questions to generate these tasks. See Table 1 for a mapping of the roles of the buy.01 frame to questions.

Step 3. Each question is presented to crowd workers together with the sentence and a set of answer options. Example annotation tasks are illustrated in Figures 2 and 3. A task thus is defined as follows:

Definition 1 Annotation Task*: A task consists of a sentence, a human readable question regarding a predicted label, and a set of answer options.*

We collect worker responses to these tasks. If the majority of crowd workers agrees on a correction, we remove or correct incorrectly predicted labels for ConfirmPrediction tasks and add new labels for AddMissing tasks.

Frame: buy.01 (*purchase*)		
Role	Description	Question
A0	buyer	Who is buying something?
A1	thing bought	What is being bought?
A2	seller	From whom is something bought?
A3	price paid	What is the price paid?
A4	benefactive	For whom is something bought?

Table 1: Examples of mapping between semantic labels and question phrases of frame buy.01. The description column lists the textual role descriptions from PropBank frame files.

Agreement	#Tasks	#Correct	#Incorrect	Precision
all 5 agree on answer	1,801	1,679	122	**0.93**
4 out of 5 agree	436	376	60	0.86
3 out of 5 agree	278	187	91	0.67
no majority answer	34	0	34	0.0
total	2,549	2,242	307	0.88

Table 2: Tasks in our crowdsourcing study by ratio of how many workers agreed on an answer. If all five workers agree, the majority answer is correct in 93% of cases. If fewer workers agree, the precision of the majority answer decreases.

2.2 Crowdsourcing Study

We conduct a crowdsourcing study consisting of 2,549 annotation tasks, generated by running a state-of-the-art SRL system (Akbik and Li, 2016) over 250 randomly selected gold-labeled sentences from the English training dataset in the CoNLL-2009 shared task (Hajič et al., 2009). We generated tasks using our question mappings from the predicted labels. This setup allows us to compare crowd feedback to gold labels and determine how often the crowd provides incorrect answers.

Human Annotators For *crowd annotators*, we employ five native speakers of English from Upwork[1], selected using the following procedure: We required workers to complete a short tutorial[2], followed by 20 annotation tasks, which we evaluated against the gold data. We used the results to select the best-scoring 5 of 7 applicants. We then asked them to complete the remaining labeling tasks. The study was conducted in a span of three weeks. Crowd workers were paid a fixed sum for the completion of the study, which resulted in a cost of 2 cents per worker per task. In total, workers estimated an average of 9 hours to complete the full task.

2.3 Analysis

We gather crowd feedback and compare the majority answer for each task with the gold label. Refer to Table 2 for an overview of results. We make several observations:

The more workers agree, the better the answer. Generally, we note that majority answers tend to be more often correct if more workers agree. Specifically, as Table 2 shows, all 5 annota-

[1] https://www.upwork.com/

[2] The tutorial is available upon request.

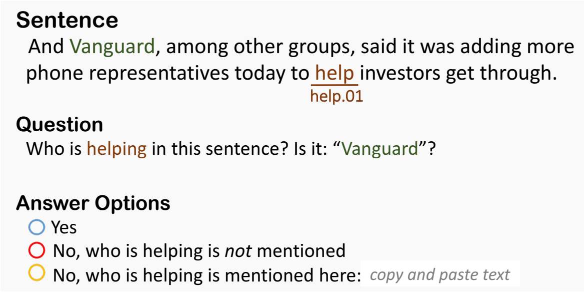
Type		Frame	A0	A1	A2	A3	A4	LOC	TMP
CONFIRM-PREDICTION	Expert-required	**134 (38%)**	280 (32%)	382 (32%)	73 (33%)	**6 (43%)**	**8 (50%)**	36 (35%)	120 (36%)
	Crowd-appropriate	222 (62%)	608 (68%)	797 (68%)	146 (67%)	8 (57%)	8 (50%)	67 (65%)	211 (64%)
ADD-MISSING	Expert-required	0	82 (31%)	54 (33%)	**240 (37%)**	99 (34%)	**72 (38%)**	0	0
	Crowd-appropriate	0	186 (69%)	111 (67%)	405 (63%)	190 (66%)	120 (62%)	0	0

Table 3: Breakdown of annotation tasks by question types and semantic labels, and proportion of expert-required tasks (formally defined in Section 3). Percentages in each cell add up to 100%. On average, 34% of tasks are expert required. Task types that lie above this average are highlighted bold. For instance, 38% of all frame confirmation questions are expert-required, indicating that this question type is of above-average difficulty.

tors agreed in 1,801 out of all 2,549 tasks (71%). Of these tasks, the majority answer was correct in 1,679 cases, and incorrect in 122, yielding a precision of 93% for tasks in full agreement. If only 4 out of 5 agree (i.e. one annotator provided a different answer), the precision drops to 86%. If only three annotators agree on an answer, the precision is even lower, at 67%. Furthermore, we note 34 cases in which there was no majority answer (no agreement by at least 3 workers). We therefore see a direct correlation between agreement scores and the validity of majority answers.

Even if all workers agree, errors are made. We also note that all 5 crowd workers sometimes unanimously agree an *incorrect annotation*, in a total of 122 cases. To illustrate such a case, consider the example in Figure 3: In our study, all 5 workers incorrectly selected **yes** as answer. However, (perhaps somewhat counterintuitively to non-experts) under the PropBank paradigm it is the "phone representative" that provide explicit help in this sentence, not "Vanguard."

Characteristics of difficult annotation tasks. As illustrated in Table 3, we break down annotation tasks by question types and semantic labels to gain a better understanding of which tasks are difficult for the crowd. The first row in the table lists results for CONFIRMPREDICTION tasks. We note that some tasks of this type require above-average expert involvement, such as confirmation questions that pertain to the frame label or higher numbered roles (A3 and A4). The second row lists results for ADDMISSING tasks. Here, we note that again higher order roles tend to be above average expert-required[3]. However, while the breakdown in Table 3 indicates some general trends for the difficulty of annotation tasks, the question type itself does not suffice to determine whether an individual instance requires expert involvement or not.


Sentence
And Vanguard, among other groups, said it was adding more phone representatives today to help investors get through.
help.01

Question
Who is helping in this sentence? Is it: "Vanguard"?

Answer Options
○ Yes
○ No, who is helping is *not* mentioned
○ No, who is helping is mentioned here: *copy and paste text*

Figure 3: Example of an annotation task where crowd workers unanimously provided an incorrect answer in our study (see 2.3). This task is classified as *expert-required.*

Summary. Our crowdsourcing study supports the initial hypothesis that a portion of SRL tasks is in fact appropriate for crowd workers, but also shows that identifying such tasks is not straightforward since neither crowd agreement scores nor the annotation task type is sufficient indicators of difficult tasks. We investigate this problem further in the next section.

3 TASKROUTER: Annotation Task Classification

Our study shows that some annotation tasks are appropriate for crowd workers, while others are not. In this section, we define a classification problem in which we wish to classify each task into one of the two following classes:

Definition 2 Crowd-appropriate*: A task for which: (1) All crowd workers agree on the answer. (2) The agreed-upon answer is correct.*

Definition 3 Expert-required*: A task that is not crowd-appropriate.*

According to these definitions, our crowdsourcing study found that the task in Figure 2 is *crowd-appropriate*, i.e. easy enough for the crowd to provide correct and consistent answers, while the task in Figure 3 is considered *expert-required*.

[3] Note that there are no ADDMISSING questions for frames since our SRL predicts a label for each verb in a sentence. Also there are no missing optional arguments since we ask missing argument questions only for core roles.

3.1 Features

To solve the task classification problem, we note two groups of distinct features (see Table 4):

Task-level features $\mathbf{X}^g$ capture the general difficulty of a labeling task, as defined by a frame or role type. The intuition here is that certain frames/roles are inherently difficult for non-experts, and that annotation tasks related to such frames/roles should be handled by experts. In the BUY.01 frame for instance, *buyer* (**A0**) is a simple crowd-appropriate semantic concept, while *benefactive* (**A4**) generally produces lower agreement scores. Task-level features therefore include the frame and role labels themselves, as well as the complexity of each question, measured in features such as the question word (*what*, *how*, *when* etc.), its length measured in number of tokens, and all tokens, lemmas and POS-tags in the question.

Sentence-level features $\mathbf{X}^l$ capture complexity associated with the specific *task instance*. The intuition is that some sentences are more complex and more difficult to understand than others. In such sentences, even roles with generally crowd-appropriate definitions might be incorrectly answered by non-experts. We capture the complexity of a sentence with features such as its length (number of tokens in sentence), the numbers of frames, roles, verbs, and nouns in the sentence, as well as all tokens, lemmas and POS-tags.

3.2 Classification Model

In addition to task- and sentence-level features, we present a classifier that also models the interplay between multiple annotation tasks generated from the same sentence. The intuition here is that there is an interdependence between labeling decisions in the same sentence. For instance, the presence of a difficult role may alter the interpretation of a sentence and make other labeling decisions more complicated. We thus propose a fuzzy classification model with two layers (Ishibuchi et al., 1995) of SVM classifiers (Wang et al., 2016), which introduces the context of the task using fuzzy indicators to model the interplay between the two groups of features.

Type	Features
Task-level features	Frame label; role label; question type; length of question in # tokens; Wh-word; tokens, lemmas, POS tags of all words in question.
Sentence-level features	# of questions for sentence, # of question types for sentence; # of verbs, # of nouns, # of frames, # of roles in sentence; length of sentence in # tokens; tokens, lemmas and POS tags of all words in sentence; head word and dependency relation to head word.

Table 4: Features for annotation task classification.

Specifically, we train a *local-layer SVM classifier* $\mathcal{L}^l$ using the sentence-level features $\mathbf{X}^l$ (computed from sentences). We also train a *global-layer SVM classifier* $\mathcal{L}^g$ using the task-level features $\mathbf{X}^g$ (computed from tasks). We refer to the predictions of the local and global classifiers as *fuzzy indicators* and we incorporate them as additional features to the fuzzy two-layer SVM classifier $\mathcal{L}^f$ as follows. Given task a_i among all tasks a_1 to a_n for a sentence s, the first layer of the fuzzy classifier, consists of applying the local-layer classifier using the sentence-level features of s. The second layer of the fuzzy classifier consists in applying the global-layer classifier n times, each time using task-level features for task $a_j, 1 \leq j \leq n$, resulting in $n+1$ values: one *local-layer indicator*, and n *global-layer indicators*. Our final fuzzy classifier model uses the $n+1$ local and global indicators as features, in addition to the sentence- and task-level features of a_i.

Note that the classification of task a_i considers features from other tasks a_j from the same sentence, but more efficiently than placing all task-level features of all tasks into a single feature vector. Formally, the objective function of the fuzzy two-layer SVM classification model $\mathcal{L}^f$ is:

$$\max_{\alpha} \mathbf{1}^T \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{Y} \mathbf{K}(\mathbf{X}^{f^T} \mathbf{X}^f) \mathbf{Y} \boldsymbol{\alpha} \qquad (1)$$
$$s.t. \quad \boldsymbol{y}^T \boldsymbol{\alpha} = 0, 0 \leq \boldsymbol{\alpha} \leq C\mathbf{1}.$$

where $\mathbf{K}(\mathbf{X}^{f^T} \mathbf{X}^f)$ is the fuzzy two-layer RBF kernel function, $\mathbf{X}^f = [\mathbf{X}^{gT}, \mathbf{X}^{l^T}, \mathbf{Y_1}^{gT}, \cdots, \mathbf{Y}_j{}^{gT}, \cdots, \mathbf{Y}_n{}^{gT}, \mathbf{Y}^{l^T}]$ is the fuzzy two-layer feature matrix, n is the number of annotation tasks generated from a sentence, $\mathbf{Y}_j{}^g$ represents the j-th fuzzy indicator generated by the j-th global classifier $\mathcal{L}^g{}_j$, $\mathbf{Y}^l$ is the fuzzy indicator generated by the local classifier $\mathcal{L}^l$, $\mathbf{Y}$ is the label matrix, $\mathbf{1}$ is a vector of all ones and C is a positive trade-off parameter.

3.3 Evaluation

To evaluate the accuracy of TASKROUTER we use the standard measure of *accuracy* for binary classifiers. As Table 5 shows, we evaluate four setups

Approach	Accuracy
SVM$_{task}$: Task features only	0.91
SVM$_{sentence}$: Sentence features only	0.87
SVM$_{task+sentence}$: All features	0.94
TaskRouter: Fuzzy two-layer	**0.96***

Table 5: Performance of classifiers trained with five-fold cross validation on training set. The improvements of TaskRouter over other classifiers are significant at the * 0.05 level, paired t-test.

in which we train an SVM with (1) task-level features, (2) sentence-level features, (3) all features, and (4) our proposed fuzzy two-layer classifier.

Data. We use the dataset created in our crowdsourcing study (see Section 2.2), which consists of 2,549 annotation tasks labeled as either *expert-required* or *crowd-appropriate* according to our definitions and the results of the study. We leverage five-fold cross validation to train the classifiers over a training split (80%).

Results. The cross validation results are listed in Table 5. Our proposed classifier outperforms all baselines and reaches a classification accuracy of **0.96**. Interestingly, we also note that task-level features are significantly more important than sentence-level features, as the setup SVM$_{task}$ outperforms SVM$_{sentence}$ by 6 accuracy points. Furthermore, our proposed approach outperforms SVM$_{task+sentence}$, indicating a positive impact of modeling the global interplay of annotation tasks.

These experiments confirm our initial postulation that it is possible to train a high quality classifier to detect expert-required tasks. We refer to the best performing setup as TaskRouter.

4 Crowd-in-the-Loop Study

Having created TaskRouter, we now execute our proposed crowd-in-the-loop workflow and comparatively evaluate it against a number of crowdsourcing and hybrid approaches. We wish to determine (1) to what degree does having the crowd in the loop reduce the workload of experts? (2) How does the quality of the produced annotated data compare to purely crowdsourced or expert annotations?

4.1 Approaches

We evaluate the following approaches:

1. Baseline without curation The first is a simple baseline in which we use the output of SRL as-is, i.e. with no additional curation either by the crowd or experts. We list this method to show the quality of the starting point for the curation workload.

2. Crowd (Crowdsourcing) The second baseline is a standard crowdsourcing approach as described in Section 2, i.e. without experts. We send all annotation tasks (100%) to the crowd and gather crowd feedback to correct labels in three different settings. We correct all labels based on majority vote, i.e., if at least 3 (Crowd$_{min3}$), 4 (Crowd$_{min4}$) or all 5 (Crowd$_{all5}$) out of 5 annotators agree on an answer.

3. Hybrid (Crowdsourcing + Expert curation) In this setting, we replicate the approach proposed by (Akbik et al., 2016): After first executing crowdsourcing (i.e. sending 100% of the tasks to the crowd), we identify all tasks in which crowd workers provided conflicting answers. These tasks are sent to experts for additional curation (expert answers are used for curation instead of the crowd response). We use three definitions of what constitutes a conflicting answer: (1) We consider all answers in which at least a majority (3 out of 5) agreed as crowd-appropriate and send the rest (2.2%) to experts. We refer to this setup as Hybrid$_{min3}$. (2) Only tasks where 4 out of 5 agreed are crowd-appropriate, the remaining 9.9% go to experts (Hybrid$_{min4}$). (3) Any task in which there is no unanimous agreement (27.3%) is deemed expert-required (Hybrid$_{all5}$).

4. Crowd-in-the-Loop This setup is *the proposed approach* in which we use TaskRouter trained over a holdout training set to split annotation tasks into crowd and expert groups. In our experiments, TaskRouter determines the following partitions: 66.4% of tasks to the crowd, the remaining 33.6% to experts. To give an indication of the lower bound of the approach given these partitions, we list results for two settings: (1) Crowd-in-the-Loop$_{Random}$, a lower bound setting in which we randomly split into these partitions. (2) Crowd-in-the-Loop$_{TaskRouter}$, the proposed setting in which we use TaskRouter to perform this split.

Refer to Table 6 for an overview of these experiments. The workload columns indicate what percentage of tasks is sent to crowd and experts.

4.2 Experimental Setup

Data We use the dataset created in the crowdsourcing study in Section 2, consisting of 2,549 annotation tasks labeled either as expert-required

Approach	Annotation Quality			Workload		Correctness	
	P	R	F1	crowd	expert	crowd-only	hybrid
Baseline without curation	0.86	0.83	0.85	0%	0%	-	-
Crowd$_{min3}$	0.92	0.88	0.90	100.0%	0%	0.84	0.84
Crowd$_{min4}$	0.89	0.85	0.87	100.0%	0%	0.84	0.84
Crowd$_{all5}$	0.87	0.84	0.85	100.0%	0%	0.84	0.84
Hybrid$_{min3}$	0.90	0.86	0.88	100.0%	2.2%	0.84	0.84
Hybrid$_{min4}$	0.91	0.87	0.89	100.0%	9.9%	0.84	0.86
Hybrid$_{all5}$	0.93	0.89	0.91	100.0%	27.3%	0.84	0.88
Crowd-in-the-Loop$_{Random}$	0.92	0.88	0.90	**66.4%**	**33.6%**	0.83	0.89
Crowd-in-the-Loop$_{TaskRouter}$	**0.96***	**0.92***	**0.94***	**66.4%**	**33.6%**	**0.92***	**0.95***

Table 6: Comparative evaluation of different approaches for generating gold-standard SRL annotation. The improvements of Crowd-in-the-Loop$_{TaskRouter}$ over other approaches are significant at the * 0.05 level, paired t-test.

or crowd-appropriate [4]. As shown in Section 3.3, we use 80% of the dataset to train TaskRouter under cross validation, and conduct the comparative evaluation using the remaining 20%.

Human annotators & curation We simulate an *expert annotator* using the CoNLL-2009 gold SRL labels and reuse the crowd answers from the study for *crowd annotators*. For each setting, we gather crowd and expert answers to the annotation tasks, and interpret the answers to curate the SRL labels that were produced by the statistical SRL system. After curation, we evaluate the resulting labeled sentences against gold-labeled data to determine the annotation quality in terms of precision, recall and F$_1$-score.

Evaluation Metrics Next to the quality of resulting annotations, we are interested to evaluate how effectively we integrate the crowd. We measure this in two metrics. (1) One is the percentage of tasks that go to the crowd and to experts respectively. Note that in the Hybrid setup, some tasks go to both crowd workers and experts, so that the percentages can add up to over a hundred percent. This information is illustrated in column Workload in Table 6. (2) The second is the overall validity of crowd feedback, referred to as *correctness*, measured as the ratio of correct answers among all answers retrieved from the crowd. We provide two values for correctness in Table 6, under column Correctness: The first is the correctness only over crowd feedback. Note that this value is the same for all Crowd and Hybrid setups since in these approaches 100% of annotation tasks are passed to the crowd. The second named *hybrid* is the overall correctness of the resolved answers with both expert and crowd feedback.

4.3 Experimental Results

The results of our experiments are summarized in Table 6. We make the following observations:

Crowd-in-the-Loop significantly increases annotation quality. Our evaluation shows that crowd-in-the-loop produces SRL annotation with significantly higher quality compared to crowdsourcing or hybrid scenarios. With a resulting F$_1$-score of 0.94, it outperforms the best performing crowdsourcing setup (0.90) by 4 points. More importantly, our proposed approach also outperforms other hybrid approaches that partially leverage experts. It outperforms the best hybrid approach (0.91) by 3 points, indicating that TaskRouter is better to select expert-required tasks than a method with only crowd agreement.

Significantly less expert involvement required. In our experiments, more than two-thirds of all tasks were determined to be crowd-appropriate by TaskRouter. This considerably reduces the need for expert involvement compared to expert labeling, while still maintaining relatively high annotation quality. In particular, our approach compares favorably to other hybrid setups in which a similar partition of tasks is completed by experts. Since TaskRouter is more capable to choose expert-required tasks than previous approaches, we achieve higher overall quality at similar levels of expert involvement.

Crowd workers more effective. As the correctness column in Table 6 shows, the selection of tasks by TaskRouter is more appropriate for the crowd in general. Their average correctness increases to 0.92, compared to 0.84 if the crowd completes 100% of the tasks.

[4] We will release the dataset soon.

4.4 Discussion and Outlook

The proposed approach far outperforms crowdsourcing and hybrid approaches in terms of annotation quality. In particular, even at similar levels of expert involvement, it outperforms the HYBRID$_{all5}$ approach. However, we also note that with an F$_1$-score of 0.94, our approach does not yet reach the quality of gold annotated data.

Insights for further improving quality. To further improve the quality of generated SRL training data, future work may (1) investigate additional features (Wang et al., 2015) and classification models to improve the TASKROUTER to better distinguish between crowd-appropriate and expert-required tasks, and (2) experiment with other SRL crowdsourcing designs to make more tasks crowd-appropriate. Nevertheless, we suspect that a small decrease in quality cannot be fully avoided if large amounts of non-experts are involved in a labeling task such as SRL. Given such involvement of non-experts, we believe that our proposed approach is a compelling way for increasing crowdsourcing quality while keeping expert costs relatively low.

Flexible trade-off of costs vs quality. Another avenue for research is to experiment with classifier parameters that allow us to more flexibly control the trade-off between how many experts we wish to involve and what annotation quality we desire (e.g., active learning (Wang et al., 2017)). This may be helpful to scenarios in which costs are fixed, or where one aims to compute the costs for generating annotated data of specific quality.

Use for SRL domain adaptation. One intended avenue for study is to apply our approach to generate training data for a specific textual domain for which little or no SRL training data currently exists. We believe that due to its relatively lower costs, our approach may be an ideal candidate for practical domain adaptation of SRL.

Applicability to other NLP crowdsourcing tasks. Finally, while in this paper we focused on the task of generating labeled training data for SRL, we believe that our proposed approach may be applicable to other NLP tasks that have only reported moderate results to-date. To study this applicability, one would first need to conduct a similar study as in Section 2 to identify crowd-appropriate and expert-required tasks and attempt the training of a classifier.

5 Related Work

Crowdsourcing SRL Annotation Different approaches have been adapted to formulate SRL tasks for non-expert crowd workers (Hong and Baker, 2011). Typical tasks include selecting answers from a set of candidates (Fossati et al., 2013), marking text passages that contain specific semantic roles (Feizabadi and Padó, 2014), and constructing question-answer pairs (He et al., 2015, 2016). However, a particular challenge is that SRL annotation tasks are often complex and crowdsourcing inevitably leads to low-quality annotations (Pavlick et al., 2015).

Instead of attempting to design a better annotation task, our proposed approach addresses this problem by accepting that a certain portion of annotation tasks is too difficult for the crowd. We create a classifier to identify such tasks and involve experts whenever necessary.

Routing Tasks Recent approaches have been developed to address the task routing problem in crowdsourcing (Bragg et al., 2014; Bozzon et al., 2013; Hassan and Curry, 2013). As workers vary in skill and tasks vary in difficulty, prior recommended approaches often consider the match between the task content and workers' profiles. However, these approaches are difficult to apply to the particular context of SRL annotation since we only distinguish between either experts familiar with PropBank, or non-expert crowd workers.

Rather than routing tasks to the most appropriate workers, our proposed approach determines which SRL tasks are appropriate for crowdsourcing, and sends the remaining ones to experts.

Human-in-the-loop Methods Our method is similar in the spirit of human-in-the-loop learning (Fung et al., 1992; Li et al., 2016). Human-in-the-loop learning generally aims to leverage humans to complete easy commonsense tasks, such as the recognition of objects in images (Patterson et al., 2013). Recent work also proposed human-in-the-loop parsing (He et al., 2016) to include human feedback into parsing. However, unlike these approaches, we aim to combine both experts and non-experts to address the difficulty of some SRL annotation tasks, while leveraging the crowd for the majority of tasks.

6 Conclusion

In this paper, we proposed CROWD-IN-THE-LOOP an approach for creating high-quality annotated data for SRL that leverages both crowd and expert workers. We conducted a crowdsourcing study and analyzed its results to design a classifier to distinguish between crowd-appropriate and expert-required tasks. Our experimental evaluation showed that our proposed approach significantly outperforms baseline crowdsourcing and hybrid approaches, and successfully limits the need for expert involvement while achieving high annotation quality.

References

Alan Akbik and Yunyao Li. 2016. K-srl: Instance-based learning for semantic role labeling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*, pages 599–608.

Alan Akbik, Kumar Vishwajeet, and Yunyao Li. 2016. Towards semi-automatic generation of proposition banks for low-resource languages. In *Conference on Empirical Methods on Natural Language Processing*, pages 993–998.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Annual Meeting of the Association for Computational Linguistics*, pages 86–90.

Alessandro Bozzon, Marco Brambilla, Stefano Ceri, Matteo Silvestri, and Giuliano Vesci. 2013. Choosing the right crowd: expert finding in social networks. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 637–648.

Jonathan Bragg, Andrey Kolobov, Mausam Mausam, and Daniel S Weld. 2014. Parallel task routing for crowdsourcing. In *Second AAAI Conference on Human Computation and Crowdsourcing*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Conference on Empirical Methods on Natural Language Processing*, pages 1535–1545.

Parvin Sadat Feizabadi and Sebastian Padó. 2014. Crowdsourcing annotation of non-local semantic roles. In *European Chapter of the Association for Computational Linguistics*, pages 226–230.

Marco Fossati, Claudio Giuliano, and Sara Tonelli. 2013. Outsourcing framenet to the crowd. In *Annual Meeting of the Association for Computational Linguistics(2)*, pages 742–747.

Michael J Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. 2011. Crowddb: answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 61–72.

Patrick T Fung, Graham Norgate, Timothy A Dilts, Andrew S Jones, and Rangaswamy Ravindran. 1992. Human-in-the-loop machine control loop. US Patent 5,116,180.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18.

Umairul Hassan and Edward Curry. 2013. A capability requirements approach for predicting worker performance in crowdsourcing. In *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on*, pages 429–437.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Conference on Empirical Methods on Natural Language Processing*, pages 643–653.

Luheng He, Julian Michael, Mike Lewis, and Luke Zettlemoyer. 2016. Human-in-the-loop parsing. In *Conference on Empirical Methods in Natural Language Processing*.

Jisup Hong and Collin F Baker. 2011. How good is the crowd at real wsd? In *Proceedings of the 5th linguistic annotation workshop*, pages 30–37.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Human Language Technology Conference of the NAACL*, pages 57–60.

Hisao Ishibuchi, Ken Nozaki, Naohisa Yamamoto, and Hideo Tanaka. 1995. Selecting fuzzy if-then rules for classification problems using genetic algorithms. *IEEE Transactions on fuzzy systems*, 3(3):260–270.

Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2016. Dialogue learning with human-in-the-loop. *arXiv preprint arXiv:1611.09823*.

Chi-kiu Lo, Meriem Beloucif, and Dekai Wu. 2013. Improving machine translation into chinese by tuning against chinese meant. In *IWSLT 2013, 10th International Workshop on Spoken Language Translation*.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Genevieve Patterson, Grant Van, Horn Serge, Belongie Pietro, and Perona James Hays. 2013. Bootstrapping fine-grained classifiers: Active learning with a crowd in the loop. In *Neural Information Processing Systems (Workshop)*.

Ellie Pavlick, Travis Wolfe, Pushpendre Rastogi, Chris Callison-Burch, Mark Dredze, and Benjamin Van Durme. 2015. Framenet+: Fast paraphrastic tripling of framenet. pages 408–413.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Conference on Empirical Methods on Natural Language Processing-CoNLL*, pages 12–21.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263.

Chenguang Wang, Laura Chiticariu, and Yunyao Li. 2017. Active learning for black-box semantic role labeling with neural factors. In *IJCAI*, page to appear.

Chenguang Wang, Yangqiu Song, Ahmed El-Kishky, Dan Roth, Ming Zhang, and Jiawei Han. 2015. Incorporating world knowledge to document clustering via heterogeneous information networks. In *KDD*, pages 1215–1224.

Chenguang Wang, Yangqiu Song, Haoran Li, Ming Zhang, and Jiawei Han. 2016. Text classification with heterogeneous information network kernels. In *AAAI*, pages 2130–2136.

A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks

Kazuma Hashimoto[*], Caiming Xiong[†], Yoshimasa Tsuruoka, and Richard Socher
The University of Tokyo
`{hassy, tsuruoka}@logos.t.u-tokyo.ac.jp`
Salesforce Research
`{cxiong, rsocher}@salesforce.com`

Abstract

Transfer and multi-task learning have traditionally focused on either a single source-target pair or very few, similar tasks. Ideally, the linguistic levels of morphology, syntax and semantics would benefit each other by being trained in a single model. We introduce a joint many-task model together with a strategy for successively growing its depth to solve increasingly complex tasks. Higher layers include shortcut connections to lower-level task predictions to reflect linguistic hierarchies. We use a simple regularization term to allow for optimizing all model weights to improve one task's loss without exhibiting catastrophic interference of the other tasks. Our single end-to-end model obtains state-of-the-art or competitive results on five different tasks from tagging, parsing, relatedness, and entailment tasks.

1 Introduction

The potential for leveraging multiple levels of representation has been demonstrated in various ways in the field of Natural Language Processing (NLP). For example, Part-Of-Speech (POS) tags are used for syntactic parsers. The parsers are used to improve higher-level tasks, such as natural language inference (Chen et al., 2016) and machine translation (Eriguchi et al., 2016). These systems are often pipelines and not trained end-to-end.

Deep NLP models have yet shown benefits from predicting many increasingly complex tasks each at a successively deeper layer. Existing models often ignore linguistic hierarchies by predicting different tasks either entirely separately or at the same depth (Collobert et al., 2011).

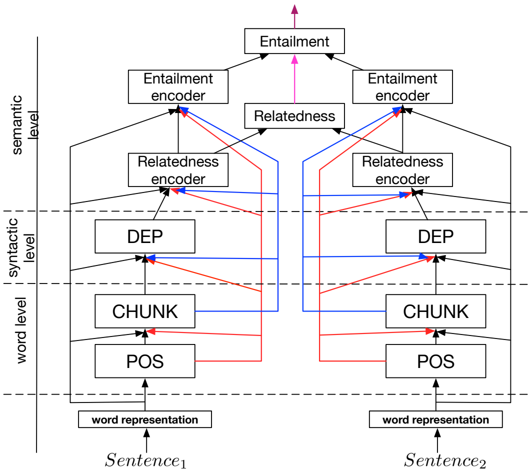


Figure 1: Overview of the joint many-task model predicting different linguistic outputs at successively deeper layers.

We introduce a Joint Many-Task (JMT) model, outlined in Figure 1, which predicts increasingly complex NLP tasks at successively deeper layers. Unlike traditional pipeline systems, our single JMT model can be trained end-to-end for POS tagging, chunking, dependency parsing, semantic relatedness, and textual entailment, by considering linguistic hierarchies. We propose an adaptive training and regularization strategy to grow this model in its depth. With the help of this strategy we avoid catastrophic interference between the tasks. Our model is motivated by Søgaard and Goldberg (2016) who showed that predicting two different tasks is more accurate when performed in different layers than in the same layer (Collobert et al., 2011). Experimental results show that our single model achieves competitive results for all of the five different tasks, demonstrating that us-

[*] Work was done while the first author was an intern at Salesforce Research.

[†] Corresponding author.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1923–1933
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

ing linguistic hierarchies is more important than handling different tasks in the same layer.

2 The Joint Many-Task Model

This section describes the inference procedure of our model, beginning at the lowest level and working our way to higher layers and more complex tasks; our model handles the five different tasks in the order of POS tagging, chunking, dependency parsing, semantic relatedness, and textual entailment, by considering linguistic hierarchies. The POS tags are used for chunking, and the chunking tags are used for dependency parsing (Attardi and DellOrletta, 2008). Tai et al. (2015) have shown that dependencies improve the relatedness task. The relatedness and entailment tasks are closely related to each other. If the semantic relatedness between two sentences is very low, they are unlikely to entail each other. Based on this observation, we make use of the information from the relatedness task for improving the entailment task.

2.1 Word Representations

For each word w_t in the input sentence s of length L, we use two types of embeddings.

Word embeddings: We use Skip-gram (Mikolov et al., 2013) to train word embeddings.

Character embeddings: Character n-gram embeddings are trained by the same Skip-gram objective. We construct the character n-gram vocabulary in the training data and assign an embedding for each entry. The final character embedding is the average of the *unique* character n-gram embeddings of w_t. For example, the character n-grams ($n = 1, 2, 3$) of the word "Cat" are {C, a, t, #B#C, Ca, at, t#E#, #B#Ca, Cat, at#E#}, where "#B#" and "#E#" represent the beginning and the end of each word, respectively. Using the character embeddings efficiently provides morphological features. Each word is subsequently represented as x_t, the concatenation of its corresponding word and character embeddings shared across the tasks.[1]

2.2 Word-Level Task: POS Tagging

The first layer of the model is a bi-directional LSTM (Graves and Schmidhuber, 2005; Hochreiter and Schmidhuber, 1997) whose hidden states are used to predict POS tags. We use the following Long Short-Term Memory (LSTM) units for the forward direction:

$$\begin{aligned} i_t &= \sigma\left(W_i g_t + b_i\right), \; f_t = \sigma\left(W_f g_t + b_f\right), \\ u_t &= \tanh\left(W_u g_t + b_u\right), \\ c_t &= i_t \odot u_t + f_t \odot c_{t-1}, \qquad (1) \\ o_t &= \sigma\left(W_o g_t + b_o\right), \; h_t = o_t \odot \tanh\left(c_t\right), \end{aligned}$$

where we define the input g_t as $g_t = [\overrightarrow{h}_{t-1}; x_t]$, i.e. the concatenation of the previous hidden state and the word representation of w_t. The backward pass is expanded in the same way, but a different set of weights are used.

For predicting the POS tag of w_t, we use the concatenation of the forward and backward states in a one-layer bi-LSTM layer corresponding to the t-th word: $h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t]$. Then each h_t $(1 \le t \le L)$ is fed into a standard softmax classifier with a single ReLU layer which outputs the probability vector $y^{(1)}$ for each of the POS tags.

2.3 Word-Level Task: Chunking

Chunking is also a word-level classification task which assigns a chunking tag (B-NP, I-VP, etc.) for each word. The tag specifies the region of major phrases (e.g., noun phrases) in the sentence.

Chunking is performed in the second bi-LSTM layer on top of the POS layer. When stacking the bi-LSTM layers, we use Eq. (1) with input $g_t^{(2)} = [h_{t-1}^{(2)}; h_t^{(1)}; x_t; y_t^{(pos)}]$, where $h_t^{(1)}$ is the hidden state of the first (POS) layer. We define the weighted label embedding $y_t^{(pos)}$ as follows:

$$y_t^{(pos)} = \sum_{j=1}^{C} p(y_t^{(1)} = j | h_t^{(1)}) \ell(j), \qquad (2)$$

where C is the number of the POS tags, $p(y_t^{(1)} = j | h_t^{(1)})$ is the probability value that the j-th POS tag is assigned to w_t, and $\ell(j)$ is the corresponding label embedding. The probability values are predicted by the POS layer, and thus no gold POS tags are needed. This output embedding is similar to the K-best POS tag feature which has been shown to be effective in syntactic tasks (Andor et al., 2016; Alberti et al., 2015). For predicting the chunking tags, we employ the same strategy as POS tagging by using the concatenated bi-directional hidden states $h_t^{(2)} = [\overrightarrow{h}_t^{(2)}; \overleftarrow{h}_t^{(2)}]$ in the chunking layer. We also use a single ReLU hidden layer before the softmax classifier.

[1] Bojanowski et al. (2017) previously proposed to train the character n-gram embeddings by the Skip-gram objective.

2.4 Syntactic Task: Dependency Parsing

Dependency parsing identifies syntactic relations (such as an adjective modifying a noun) between word pairs in a sentence. We use the third bi-LSTM layer to classify relations between all pairs of words. The input vector for the LSTM includes hidden states, word representations, and the label embeddings for the two previous tasks: $g_t^{(3)} = [h_{t-1}^{(3)}; h_t^{(2)}; x_t; (y_t^{(pos)} + y_t^{(chk)})]$, where we computed the chunking vector in a similar fashion as the POS vector in Eq. (2).

We predict the parent node (*head*) for each word. Then a dependency label is predicted for each child-parent pair. This approach is related to Dozat and Manning (2017) and Zhang et al. (2017), where the main difference is that our model works on a multi-task framework. To predict the parent node of w_t, we define a matching function between w_t and the candidates of the parent node as $m\left(t, j\right) = h_t^{(3)} \cdot (W_d h_j^{(3)})$, where W_d is a parameter matrix. For the root, we define $h_{L+1}^{(3)} = r$ as a parameterized vector. To compute the probability that w_j (or the root node) is the parent of w_t, the scores are normalized:

$$p(j|h_t^{(3)}) = \frac{\exp\left(m\left(t, j\right)\right)}{\sum_{k=1, k\neq t}^{L+1} \exp\left(m\left(t, k\right)\right)}. \quad (3)$$

The dependency labels are predicted using $[h_t^{(3)}; h_j^{(3)}]$ as input to a $\mathrm{softmax}$ classifier with a single ReLU layer. We greedily select the parent node and the dependency label for each word. When the parsing result is not a well-formed tree, we apply the first-order Eisner's algorithm (Eisner, 1996) to obtain a well-formed tree from it.

2.5 Semantic Task: Semantic relatedness

The next two tasks model the semantic relationships between two input sentences. The first task measures the semantic relatedness between two sentences. The output is a real-valued relatedness score for the input sentence pair. The second task is textual entailment, which requires one to determine whether a premise sentence entails a hypothesis sentence. There are typically three classes: entailment, contradiction, and neutral. We use the fourth and fifth bi-LSTM layer for the relatedness and entailment task, respectively.

Now it is required to obtain the sentence-level representation rather than the word-level representation $h_t^{(4)}$ used in the first three tasks. We compute the sentence-level representation $h_{\mathbf{s}}^{(4)}$ as the element-wise maximum values across all of the word-level representations in the fourth layer:

$$h_{\mathbf{s}}^{(4)} = \max\left(h_1^{(4)}, h_2^{(4)}, \ldots, h_L^{(4)}\right). \quad (4)$$

This max-pooling technique has proven effective in text classification tasks (Lai et al., 2015).

To model the semantic relatedness between s and s', we follow Tai et al. (2015). The feature vector for representing the semantic relatedness is computed as follows:

$$d_1(s, s') = \left[\left|h_{\mathbf{s}}^{(4)} - h_{\mathbf{s}'}^{(4)}\right|; h_{\mathbf{s}}^{(4)} \odot h_{\mathbf{s}'}^{(4)}\right], \quad (5)$$

where $\left|h_{\mathbf{s}}^{(4)} - h_{\mathbf{s}'}^{(4)}\right|$ is the absolute values of the element-wise subtraction, and $h_{\mathbf{s}}^{(4)} \odot h_{\mathbf{s}'}^{(4)}$ is the element-wise multiplication. Then $d_1(s, s')$ is fed into a $\mathrm{softmax}$ classifier with a single Maxout hidden layer (Goodfellow et al., 2013) to output a relatedness score (from 1 to 5 in our case).

2.6 Semantic Task: Textual entailment

For entailment classification, we also use the max-pooling technique as in the semantic relatedness task. To classify the premise-hypothesis pair (s, s') into one of the three classes, we compute the feature vector $d_2(s, s')$ as in Eq. (5) except that we do not use the absolute values of the element-wise subtraction, because we need to identify which is the premise (or hypothesis). Then $d_2(s, s')$ is fed into a $\mathrm{softmax}$ classifier.

To use the output from the relatedness layer directly, we use the label embeddings for the relatedness task. More concretely, we compute the class label embeddings for the semantic relatedness task similar to Eq. (2). The final feature vectors that are concatenated and fed into the entailment classifier are the weighted relatedness label embedding and the feature vector $d_2(s, s')$. We use three Maxout hidden layers before the classifier.

3 Training the JMT Model

The model is trained jointly over all datasets. During each epoch, the optimization iterates over each full training dataset in the same order as the corresponding tasks described in the modeling section.

3.1 Pre-Training Word Representations

We pre-train word embeddings using the Skip-gram model with negative sampling (Mikolov et al., 2013). We also pre-train the character n-gram embeddings using Skip-gram.[2] The only difference is that each input word embedding is replaced with its corresponding average character n-gram embedding described in Section 2.1. These embeddings are fine-tuned during the model training. We denote the embedding parameters as θ_e.

3.2 Training the POS Layer

Let $\theta_{\text{POS}} = (W_{\text{POS}}, b_{\text{POS}}, \theta_e)$ denote the set of model parameters associated with the POS layer, where W_{POS} is the set of the weight matrices in the first bi-LSTM and the classifier, and b_{POS} is the set of the bias vectors. The objective function to optimize θ_{POS} is defined as follows:

$$J_1(\theta_{\text{POS}}) = -\sum_s \sum_t \log p(y_t^{(1)} = \alpha | h_t^{(1)}) + \lambda \|W_{\text{POS}}\|^2 + \delta \|\theta_e - \theta'_e\|^2, \tag{6}$$

where $p(y_t^{(1)} = \alpha_{w_t} | h_t^{(1)})$ is the probability value that the correct label α is assigned to w_t in the sentence s, $\lambda \|W_{\text{POS}}\|^2$ is the L2-norm regularization term, and λ is a hyperparameter.

We call the second regularization term $\delta \|\theta_e - \theta'_e\|^2$ a *successive* regularization term. The successive regularization is based on the idea that we do not want the model to forget the information learned for the other tasks. In the case of POS tagging, the regularization is applied to θ_e, and θ'_e is the embedding parameter after training the final task in the top-most layer at the previous training epoch. δ is a hyperparameter.

3.3 Training the Chunking Layer

The objective function is defined as follows:

$$J_2(\theta_{\text{chk}}) = -\sum_s \sum_t \log p(y_t^{(2)} = \alpha | h_t^{(2)}) + \lambda \|W_{\text{chk}}\|^2 + \delta \|\theta_{\text{POS}} - \theta'_{\text{POS}}\|^2, \tag{7}$$

which is similar to that of POS tagging, and θ_{chk} is $(W_{\text{chk}}, b_{\text{chk}}, E_{\text{POS}}, \theta_e)$, where W_{chk} and b_{chk} are the weight and bias parameters including those in θ_{POS}, and E_{POS} is the set of the POS label embeddings. θ'_{POS} is the one after training the POS layer at the current training epoch.

3.4 Training the Dependency Layer

The objective function is defined as follows:

$$J_3(\theta_{\text{dep}}) = -\sum_s \sum_t \log p(\alpha | h_t^{(3)}) p(\beta | h_t^{(3)}, h_\alpha^{(3)}) + \lambda (\|W_{\text{dep}}\|^2 + \|W_d\|^2) + \delta \|\theta_{\text{chk}} - \theta'_{\text{chk}}\|^2, \tag{8}$$

where $p(\alpha | h_t^{(3)})$ is the probability value assigned to the correct parent node α for w_t, and $p(\beta | h_t^{(3)}, h_\alpha^{(3)})$ is the probability value assigned to the correct dependency label β for the child-parent pair (w_t, α). θ_{dep} is defined as $(W_{\text{dep}}, b_{\text{dep}}, W_d, r, E_{\text{POS}}, E_{\text{chk}}, \theta_e)$, where W_{dep} and b_{dep} are the weight and bias parameters including those in θ_{chk}, and E_{chk} is the set of the chunking label embeddings.

3.5 Training the Relatedness Layer

Following Tai et al. (2015), the objective function is defined as follows:

$$J_4(\theta_{\text{rel}}) = \sum_{(s,s')} \text{KL}\left(\hat{p}(s, s') \middle\| p(h_s^{(4)}, h_{s'}^{(4)})\right) + \lambda \|W_{\text{rel}}\|^2 + \delta \|\theta_{\text{dep}} - \theta'_{\text{dep}}\|^2, \tag{9}$$

where $\hat{p}(s, s')$ is the gold distribution over the defined relatedness scores, $p(h_s^{(4)}, h_{s'}^{(4)})$ is the predicted distribution given the the sentence representations, and $\text{KL}\left(\hat{p}(s, s') \middle\| p(h_s^{(4)}, h_{s'}^{(4)})\right)$ is the KL-divergence between the two distributions. θ_{rel} is defined as $(W_{\text{rel}}, b_{\text{rel}}, E_{\text{POS}}, E_{\text{chk}}, \theta_e)$.

3.6 Training the Entailment Layer

The objective function is defined as follows:

$$J_5(\theta_{\text{ent}}) = -\sum_{(s,s')} \log p(y_{(s,s')}^{(5)} = \alpha | h_s^{(5)}, h_{s'}^{(5)}) + \lambda \|W_{\text{ent}}\|^2 + \delta \|\theta_{\text{rel}} - \theta'_{\text{rel}}\|^2, \tag{10}$$

where $p(y_{(s,s')}^{(5)} = \alpha | h_s^{(5)}, h_{s'}^{(5)})$ is the probability value that the correct label α is assigned to the premise-hypothesis pair (s, s'). θ_{ent} is defined as $(W_{\text{ent}}, b_{\text{ent}}, E_{\text{POS}}, E_{\text{chk}}, E_{\text{rel}}, \theta_e)$, where E_{rel} is the set of the relatedness label embeddings.

4 Related Work

Many deep learning approaches have proven to be effective in a variety of NLP tasks and are becoming more and more complex. They are typically

[2] The training code and the pre-trained embeddings are available at https://github.com/hassyGo/charNgram2vec.

designed to handle single tasks, or some of them are designed as general-purpose models (Kumar et al., 2016; Sutskever et al., 2014) but applied to different tasks independently.

For handling multiple NLP tasks, multi-task learning models with deep neural networks have been proposed (Collobert et al., 2011; Luong et al., 2016), and more recently Søgaard and Goldberg (2016) have suggested that using different layers for different tasks is more effective than using the same layer in jointly learning closely-related tasks, such as POS tagging and chunking. However, the number of tasks was limited or they have very similar task settings like word-level tagging, and it was not clear how lower-level tasks could be also improved by combining higher-level tasks.

More related to our work, Godwin et al. (2016) also followed Søgaard and Goldberg (2016) to jointly learn POS tagging, chunking, and language modeling, and Zhang and Weiss (2016) have shown that it is effective to jointly learn POS tagging and dependency parsing by sharing internal representations. In the field of relation extraction, Miwa and Bansal (2016) proposed a joint learning model for entity detection and relation extraction. All of them suggest the importance of multi-task learning, and we investigate the potential of handling different types of NLP tasks rather than closely-related ones in a single hierarchical deep model.

In the field of computer vision, some transfer and multi-task learning approaches have also been proposed (Li and Hoiem, 2016; Misra et al., 2016). For example, Misra et al. (2016) proposed a multi-task learning model to handle different tasks. However, they assume that each data sample has annotations for the different tasks, and do not explicitly consider task hierarchies.

Recently, Rusu et al. (2016) have proposed a progressive neural network model to handle multiple reinforcement learning tasks, such as Atari games. Like our JMT model, their model is also successively trained according to different tasks using different layers called columns in their paper. In their model, once the first task is completed, the model parameters for the first task are fixed, and then the second task is handled with new model parameters. Therefore, accuracy of the previously trained tasks is never improved. In NLP tasks, multi-task learning has the potential to improve not only higher-level tasks, but also lower-level tasks. Rather than fixing the pre-trained model parameters, our successive regularization allows our model to continuously train the lower-level tasks without significant accuracy drops.

5 Experimental Settings

5.1 Datasets

POS tagging: To train the POS tagging layer, we used the Wall Street Journal (WSJ) portion of Penn Treebank, and followed the standard split for the training (Section 0-18), development (Section 19-21), and test (Section 22-24) sets. The evaluation metric is the word-level accuracy.

Chunking: For chunking, we also used the WSJ corpus, and followed the standard split for the training (Section 15-18) and test (Section 20) sets as in the CoNLL 2000 shared task. We used Section 19 as the development set and employed the IOBES tagging scheme. The evaluation metric is the F1 score defined in the shared task.

Dependency parsing: We also used the WSJ corpus for dependency parsing, and followed the standard split for the training (Section 2-21), development (Section 22), and test (Section 23) sets. We obtained Stanford style dependencies using the version 3.3.0 of the Stanford converter. The evaluation metrics are the Unlabeled Attachment Score (UAS) and the Labeled Attachment Score (LAS), and punctuations are excluded for the evaluation.

Semantic relatedness: For the semantic relatedness task, we used the SICK dataset (Marelli et al., 2014), and followed the standard split for the training, development, and test sets. The evaluation metric is the Mean Squared Error (MSE) between the gold and predicted scores.

Textual entailment: For textual entailment, we also used the SICK dataset and exactly the same data split as the semantic relatedness dataset. The evaluation metric is the accuracy.

5.2 Training Details

We set the dimensionality of the embeddings and the hidden states in the bi-LSTMs to 100. At each training epoch, we trained our model in the order of POS tagging, chunking, dependency parsing, semantic relatedness, and textual entailment. We used mini-batch stochastic gradient decent and empirically found it effective to use a gradient clipping method with growing clipping values for the different tasks; concretely, we employed the simple function: $\min(3.0, depth)$, where $depth$ is the number of bi-LSTM layers involved in each task, and 3.0 is the maximum value. We applied our successive regularization to our model, along with L2-norm regularization and dropout (Srivastava et al., 2014). More details are summarized in the supplemental material.

6 Results and Discussion

Table 1 shows our results on the test sets of the five tasks.[3] The column "Single" shows the results of handling each task separately using single-layer bi-LSTMs, and the column "JMT$_{\text{all}}$" shows the results of our JMT model. The single task settings only use the annotations of their own tasks. For example, when handling dependency parsing as a single task, the POS and chunking tags are *not* used. We can see that all results of the five tasks are improved in our JMT model, which shows that our JMT model can handle the five different tasks in a single model. Our JMT model allows us to access arbitrary information learned from the different tasks. If we want to use the model just as a POS tagger, we can use only first bi-LSTM layer.

Table 1 also shows the results of five subsets of the different tasks. For example, in the case of "JMT$_{\text{ABC}}$", only the first three layers of the bi-LSTMs are used to handle the three tasks. In the case of "JMT$_{\text{DE}}$", only the top two layers are used as a two-layer bi-LSTM by omitting all information from the first three layers. The results of the closely-related tasks ("AB", "ABC", and "DE") show that our JMT model improves both of the high-level and low-level tasks. The results of "JMT$_{\text{CD}}$" and "JMT$_{\text{CE}}$" show that the parsing task can be improved by the semantic tasks.

It should be noted that in our analysis on the greedy parsing results of the "JMT$_{\text{ABC}}$" setting, we have found that more than 95% are well-formed dependency trees on the development set. In the 1,700 sentences of the development data, 11 results have multiple root notes, 11 results have no root nodes, and 61 results have cycles. These 83 parsing results are converted into well-formed trees by Eisner's algorithm, and the accuracy does not significantly change (UAS: 94.52%→94.53%, LAS: 92.61%→92.62%).

6.1 Comparison with Published Results

POS tagging Table 2 shows the results of POS tagging, and our JMT model achieves the score close to the state-of-the-art results. The best result to date has been achieved by Ling et al. (2015), which uses character-based LSTMs. Incorporating the character-based encoders into our JMT model would be an interesting direction, but we have shown that the simple pre-trained character n-gram embeddings lead to the promising result.

Chunking Table 3 shows the results of chunking, and our JMT model achieves the state-of-the-art result. Søgaard and Goldberg (2016) proposed to jointly learn POS tagging and chunking in different layers, but they only showed improvement for chunking. By contrast, our results show that the low-level tasks are also improved.

Dependency parsing Table 4 shows the results of dependency parsing by using only the WSJ corpus in terms of the dependency annotations.[4] It is notable that our simple greedy dependency parser outperforms the model in Andor et al. (2016) which is based on beam search with global information. The result suggests that the bi-LSTMs efficiently capture global information necessary for dependency parsing. Moreover, our single task result already achieves high accuracy without the POS and chunking information. The best result to date has been achieved by the model propsoed in Dozat and Manning (2017), which uses higher dimensional representations than ours and proposes a more sophisticated attention mechanism called *biaffine attention*. It should be promising to incorporate their attention mechanism into our parsing component.

Semantic relatedness Table 5 shows the results of the semantic relatedness task, and our JMT model achieves the state-of-the-art result. The result of "JMT$_{\text{DE}}$" is already better than the previous state-of-the-art results. Both of Zhou et al. (2016) and Tai et al. (2015) explicitly used syntactic trees, and Zhou et al. (2016) relied on attention mechanisms. However, our method uses the simple max-pooling strategy, which suggests that it is worth

[3] In chunking evaluation, we only show the results of "Single" and "JMT$_{\text{AB}}$" because the sentences for chunking evaluation overlap the training data for dependency parsing.

[4] Choe and Charniak (2016) employed a tri-training method to expand the training data with 400,000 trees in addition to the WSJ data, and they reported 95.9 UAS and 94.1 LAS by converting their constituency trees into dependency trees. Kuncoro et al. (2017) also reported high accuracy (95.8 UAS and 94.6 LAS) by using a converter.

		Single	JMT$_{\mathrm{all}}$	JMT$_{\mathrm{AB}}$	JMT$_{\mathrm{ABC}}$	JMT$_{\mathrm{DE}}$	JMT$_{\mathrm{CD}}$	JMT$_{\mathrm{CE}}$
A ↑	POS	97.45	97.55	97.52	97.54	n/a	n/a	n/a
B ↑	Chunking	95.02	n/a	95.77	n/a	n/a	n/a	n/a
C ↑	Dependency UAS	93.35	94.67	n/a	94.71	n/a	93.53	93.57
	Dependency LAS	91.42	92.90	n/a	92.92	n/a	91.62	91.69
D ↓	Relatedness	0.247	0.233	n/a	n/a	0.238	0.251	n/a
E ↑	Entailment	81.8	86.2	n/a	n/a	86.8	n/a	82.4

Table 1: Test set results for the five tasks. In the relatedness task, the lower scores are better.

Method	Acc. ↑
JMT$_{\mathrm{all}}$	97.55
Ling et al. (2015)	**97.78**
Kumar et al. (2016)	97.56
Ma and Hovy (2016)	97.55
Søgaard (2011)	97.50
Collobert et al. (2011)	97.29
Tsuruoka et al. (2011)	97.28
Toutanova et al. (2003)	97.27

Table 2: POS tagging results.

Method	F1 ↑
JMT$_{\mathrm{AB}}$	**95.77**
Single	95.02
Søgaard and Goldberg (2016)	95.56
Suzuki and Isozaki (2008)	95.15
Collobert et al. (2011)	94.32
Kudo and Matsumoto (2001)	93.91
Tsuruoka et al. (2011)	93.81

Table 3: Chunking results.

Method	UAS ↑	LAS ↑
JMT$_{\mathrm{all}}$	94.67	92.90
Single	93.35	91.42
Dozat and Manning (2017)	**95.74**	**94.08**
Andor et al. (2016)	94.61	92.79
Alberti et al. (2015)	94.23	92.36
Zhang et al. (2017)	94.10	91.90
Weiss et al. (2015)	93.99	92.05
Dyer et al. (2015)	93.10	90.90
Bohnet (2010)	92.88	90.71

Table 4: Dependency results.

Method	MSE ↓
JMT$_{\mathrm{all}}$	**0.233**
JMT$_{\mathrm{DE}}$	0.238
Zhou et al. (2016)	0.243
Tai et al. (2015)	0.253

Table 5: Semantic relatedness results.

Method	Acc. ↑
JMT$_{\mathrm{all}}$	86.2
JMT$_{\mathrm{DE}}$	**86.8**
Yin et al. (2016)	86.2
Lai and Hockenmaier (2014)	84.6

Table 6: Textual entailment results.

	JMT$_{\mathrm{all}}$	w/o SC	w/o LE	w/o SC&LE
POS	97.88	97.79	97.85	97.87
Chunking	97.59	97.08	97.40	97.33
Dependency UAS	94.51	94.52	94.09	94.04
Dependency LAS	92.60	92.62	92.14	92.03
Relatedness	0.236	0.698	0.261	0.765
Entailment	84.6	75.0	81.6	71.2

Table 7: Effectiveness of the Shortcut Connections (SC) and the Label Embeddings (LE).

	JMT$_{\mathrm{ABC}}$	w/o SC&LE	All-3
POS	97.90	97.87	97.62
Chunking	97.80	97.41	96.52
Dependency UAS	94.52	94.13	93.59
Dependency LAS	92.61	92.16	91.47

Table 8: Effectiveness of using different layers for different tasks.

investigating such simple methods before developing complex methods for simple tasks. Currently, our JMT model does not explicitly use the learned dependency structures, and thus the explicit use of the output from the dependency layer should be an interesting direction of future work.

Textual entailment Table 6 shows the results of textual entailment, and our JMT model achieves the state-of-the-art result. The previous state-of-the-art result in Yin et al. (2016) relied on attention mechanisms and dataset-specific data pre-processing and features. Again, our simple max-pooling strategy achieves the state-of-the-art result boosted by the joint training. These results show the importance of jointly handling related tasks.

6.2 Analysis on the Model Architectures

We investigate the effectiveness of our model in detail. All of the results shown in this section are the development set results.

Shortcut connections Our JMT model feeds the word representations into all of the bi-LSTM layers, which is called the shortcut connection. Table 7 shows the results of "JMT$_{\mathrm{all}}$" with and without the shortcut connections. The results without the shortcut connections are shown in the column of "w/o SC". These results clearly show that the importance of the shortcut connections, and in particular, the semantic tasks in the higher layers strongly rely on the shortcut connections. That is, simply stacking the LSTM layers is not sufficient to handle a variety of NLP tasks in a single model. In the supplementary material, it is qualitatively shown how the shortcut connections work in our model.

Output label embeddings Table 7 also shows the results without using the output labels of the POS, chunking, and relatedness layers, in the column of "w/o LE". These results show that the explicit use of the output information from the classifiers of the lower layers is important in our JMT

	JMT$_{\text{all}}$	w/o SR	w/o VC
POS	97.88	97.85	97.82
Chunking	97.59	97.13	97.45
Dependency UAS	94.51	94.46	94.38
Dependency LAS	92.60	92.57	92.48
Relatedness	0.236	0.239	0.241
Entailment	84.6	84.2	84.8

Table 9: Effectiveness of the Successive Regularization (SR) and the Vertical Connections (VC).

	JMT$_{\text{all}}$	Random
POS	97.88	97.83
Chunking	97.59	97.71
Dependency UAS	94.51	94.66
Dependency LAS	92.60	92.80
Relatedness	0.236	0.298
Entailment	84.6	83.2

Table 10: Effects of the order of training.

model. The results in the column of “w/o SC&LE” are the ones without both of the shortcut connections and the label embeddings.

Different layers for different tasks Table 8 shows the results of our “JMT$_{\text{ABC}}$” setting and that of not using the shortcut connections and the label embeddings (“w/o SC&LE”) as in Table 7. In addition, in the column of “All-3”, we show the results of using the highest (i.e., the third) layer for all of the three tasks without any shortcut connections and label embeddings, and thus the two settings “w/o SC&LE” and “All-3” require exactly the same number of the model parameters. The “All-3” setting is similar to the multi-task model of Collobert et al. (2011) in that task-specific output layers are used but most of the model parameters are shared. The results show that using the same layers for the three different tasks hampers the effectiveness of our JMT model, and the design of the model is much more important than the number of the model parameters.

Successive regularization In Table 9, the column of “w/o SR” shows the results of omitting the successive regularization terms described in Section 3. We can see that the accuracy of chunking is improved by the successive regularization, while other results are not affected so much. The chunking dataset used here is relatively small compared with other low-level tasks, POS tagging and dependency parsing. Thus, these results suggest that the successive regularization is effective when dataset sizes are imbalanced.

Vertical connections We investigated our JMT results without using the vertical connections in the five-layer bi-LSTMs. More concretely, when constructing the input vectors g_t, we do not use the bi-LSTM hidden states of the previous layers. Table 9 also shows the JMT$_{\text{all}}$ results with and without the vertical connections. As shown in the column of “w/o VC”, we observed the competitive results. Therefore, in the target tasks used in our model, sharing the word representations and the output label embeddings is more effective than just stacking the bi-LSTM layers.

	Single	Single+
POS	97.52	
Chunking	95.65	96.08
Dependency UAS	93.38	93.88
Dependency LAS	91.37	91.83
Relatedness	0.239	0.665
Entailment	83.8	66.4

Table 11: Effects of depth for the *single* tasks.

Single	W&C	Only W
POS	97.52	96.26
Chunking	95.65	94.92
Dependency UAS	93.38	92.90
Dependency LAS	91.37	90.44

Table 12: Effects of the character embeddings.

Order of training Our JMT model iterates the training process in the order described in Section 3. Our hypothesis is that it is important to start from the lower-level tasks and gradually move to the higher-level tasks. Table 10 shows the results of training our model by randomly shuffling the order of the tasks for each epoch in the column of “Random”. We see that the scores of the semantic tasks drop by the random strategy. In our preliminary experiments, we have found that constructing the mini-batch samples from different tasks also hampers the effectiveness of our model, which also supports our hypothesis.

Depth The single task settings shown in Table 1 are obtained by using single layer bi-LSTMs, but in our JMT model, the higher-level tasks use successively deeper layers. To investigate the gap between the different number of the layers for each task, we also show the results of using multi-layer bi-LSTMs for the single task settings, in the column of “Single+” in Table 11. More concretely, we use the same number of the layers with our JMT model; for example, three layers are used for dependency parsing, and five layers are used for textual entailment. As shown in these results, deeper layers do not always lead to better results, and the joint learning is more important than making the models complex only for single tasks.

Character n-gram embeddings Finally, Table 12 shows the results for the three single tasks with and without the pre-trained character n-gram embeddings. The column of "W&C" corresponds to using both of the word and character n-gram embeddings, and that of "Only W" corresponds to using only the word embeddings. These results clearly show that jointly using the pre-trained word and character n-gram embeddings is helpful in improving the results. The pre-training of the character n-gram embeddings is also effective; for example, without the pre-training, the POS accuracy drops from 97.52% to 97.38% and the chunking accuracy drops from 95.65% to 95.14%.

6.3 Discussion

Training strategies In our JMT model, it is not obvious when to stop the training while trying to maximize the scores of all the five tasks. We focused on maximizing the accuracy of dependency parsing on the development data in our experiments. However, the sizes of the training data are different across the different tasks; for example, the semantic tasks include only 4,500 sentence pairs, and the dependency parsing dataset includes 39,832 sentences with word-level annotations. Thus, in general, dependency parsing requires more training epochs than the semantic tasks, but currently, our model trains all of the tasks for the same training epochs. The same strategy for decreasing the learning rate is also shared across all the different tasks, although our growing gradient clipping method described in Section 5.2 helps improve the results. Indeed, we observed that better scores of the semantic tasks can be achieved before the accuracy of dependency parsing reaches the best score. Developing a method for achieving the best scores for all of the tasks at the same time is important future work.

More tasks Our JMT model has the potential of handling more tasks than the five tasks used in our experiments; examples include entity detection and relation extraction as in Miwa and Bansal (2016) as well as language modeling (Godwin et al., 2016). It is also a promising direction to train each task for multiple domains by focusing on domain adaptation (Søgaard and Goldberg, 2016). In particular, incorporating language modeling tasks provides an opportunity to use large text data. Such large text data was used in our experiments to pre-train the word and character n-gram embeddings. However, it would be preferable to efficiently use it for improving the entire model.

Task-oriented learning of low-level tasks Each task in our JMT model is supervised by its corresponding dataset. However, it would be possible to learn low-level tasks by optimizing high-level tasks, because the model parameters of the low-level tasks can be directly modified by learning the high-level tasks. One example has already been presented in Hashimoto and Tsuruoka (2017), where our JMT model is extended to learning task-oriented latent graph structures of sentences by training our dependency parsing component according to a neural machine translation objective.

7 Conclusion

We presented a joint many-task model to handle multiple NLP tasks with growing depth in a single end-to-end model. Our model is successively trained by considering linguistic hierarchies, directly feeding word representations into all layers, explicitly using low-level predictions, and applying successive regularization. In experiments on five NLP tasks, our single model achieves the state-of-the-art or competitive results on chunking, dependency parsing, semantic relatedness, and textual entailment.

Acknowledgments

We thank the anonymous reviewers and the Salesforce Research team members for their fruitful comments and discussions.

References

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved Transition-Based Parsing and Tagging with Neural Networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1354–1359.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452.

Giuseppe Attardi and Felice DellOrletta. 2008. Chunking and Dependency Parsing. In *Proceedings of LREC 2008 Workshop on Partial Parsing*.

Bernd Bohnet. 2010. Top Accuracy and Fast Dependency Parsing is not a Contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 89–97.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Enhancing and Combining Sequential and Tree LSTM for Natural Language Inference. *arXiv*, cs.CL 1609.06038.

Do Kook Choe and Eugene Charniak. 2016. Parsing as Language Modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen nad Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of the 5th International Conference on Learning Representations*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343.

Jason Eisner. 1996. Efficient Normal-Form Parsing for Combinatory Categorial Grammar. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 79–86.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-Sequence Attentional Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833.

Jonathan Godwin, Pontus Stenetorp, and Sebastian Riedel. 2016. Deep Semi-Supervised Learning with Linguistically Motivated Sequence Labeling Task Hierarchies. *arXiv*, cs.CL 1612.09113.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout Networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1319–1327.

Alex Graves and Jurgen Schmidhuber. 2005. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks*, 18(5):602–610.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2017. Neural Machine Translation with Source-Side Latent Graph Parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. To appear.

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

Taku Kudo and Yuji Matsumoto. 2001. Chunking with Support Vector Machines. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1378–1387.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What Do Recurrent Neural Network Grammars Learn About Syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1249–1258.

Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: A Denotational and Distributional Approach to Semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 329–334.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2267–2273.

Zhizhong Li and Derek Hoiem. 2016. Learning without Forgetting. *CoRR*, abs/1606.09282.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task Sequence to Sequence Learning. In *Proceedings of the 4th International Conference on Learning Representations*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 1–8.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch Networks for Multi-task Learning. *CoRR*, abs/1604.03539.

Makoto Miwa and Mohit Bansal. 2016. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive Neural Networks. *CoRR*, abs/1606.04671.

Anders Søgaard. 2011. Semi-supervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 48–52.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Jun Suzuki and Hideki Isozaki. 2008. Semi-Supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 665–673.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 173–180.

Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Kazama. 2011. Learning with Lookahead: Can History-Based Models Rival Globally Optimized Models? In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 238–246.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured Training for Neural Network Transition-Based Parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333.

Wenpeng Yin, Hinrich Schtze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

Xingxing Zhang, Jianpeng Cheng, and Mirella Lapata. 2017. Dependency Parsing as Head Selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 665–676.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved Representation Learning for Syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1557–1566.

Yao Zhou, Cong Liu, and Yan Pan. 2016. Modelling Sentence Pairs with Tree-structured Attentive Encoder. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 2912–2922.

Earth Mover's Distance Minimization for Unsupervised Bilingual Lexicon Induction

Meng Zhang[†‡] **Yang Liu**[†‡] **Huanbo Luan**[†] **Maosong Sun**[†‡]
[†]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[‡]Jiangsu Collaborative Innovation Center for Language Competence, Jiangsu, China
zmlarry@foxmail.com, liuyang2011@tsinghua.edu.cn
luanhuanbo@gmail.com, sms@tsinghua.edu.cn

Abstract

Cross-lingual natural language processing hinges on the premise that there exists invariance across languages. At the word level, researchers have identified such invariance in the word embedding semantic spaces of different languages. However, in order to connect the separate spaces, cross-lingual supervision encoded in parallel data is typically required. In this paper, we attempt to establish the cross-lingual connection without relying on any cross-lingual supervision. By viewing word embedding spaces as distributions, we propose to minimize their earth mover's distance, a measure of divergence between distributions. We demonstrate the success on the unsupervised bilingual lexicon induction task. In addition, we reveal an interesting finding that the earth mover's distance shows potential as a measure of language difference.

1 Introduction

Despite tremendous variation and diversity, languages are believed to share something in common. Indeed, this belief forms the underlying basis of computational approaches to cross-lingual transfer (Täckström et al., 2013, *inter alia*), otherwise it would be inconceivable for the transfer to successfully generalize.

Linguistic universals manifest themselves at various levels of linguistic units. At the word level, there is evidence that different languages represent concepts with similar structure (Youn et al., 2016). Interestingly, as computational models of word semantics, monolingual word embeddings also exhibit isomorphism across languages (Mikolov et al., 2013a). This finding opens up the possibility to use a simple transformation, e.g. a linear map, to connect separately trained word embeddings cross-lingually. Learning such a transformation typically calls for cross-lingual supervision from parallel data (Faruqui and Dyer, 2014; Lu et al., 2015; Dinu et al., 2015; Lazaridou et al., 2015; Smith et al., 2017).

In this paper, we ask the question: Can we uncover the transformation without any cross-lingual supervision? At first sight, this task appears formidable, as it would imply that a bilingual semantic space can be constructed by using monolingual corpora only. On the other hand, the existence of structural isomorphism across monolingual embedding spaces points to the feasibility of this task: The transformation exists right there only to be discovered by the right tool.

We propose such a tool to answer the above question in the affirmative. The key insight is to view embedding spaces as distributions, and the desired transformation should make the two distributions close. This naturally calls for a measure of distribution closeness, for which we introduce the earth mover's distance. Therefore, our task can be formulated as the minimization of the earth mover's distance between the transformed source embedding distribution and the target one with respect to the transformation. Importantly, the minimization is performed at the distribution level, and hence no word-level supervision is required.

We demonstrate that the earth mover's distance minimization successfully uncovers the transformation for cross-lingual connection, as evidenced by experiments on the bilingual lexicon induction task. In fact, as an *unsupervised* approach, its performance turns out to be highly competitive with *supervised* methods. Moreover, as an interesting byproduct, the earth mover's distance provides a distance measure that may quantify a facet of language difference.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1934–1945
Copenhagen, Denmark, September 7–11, 2017.

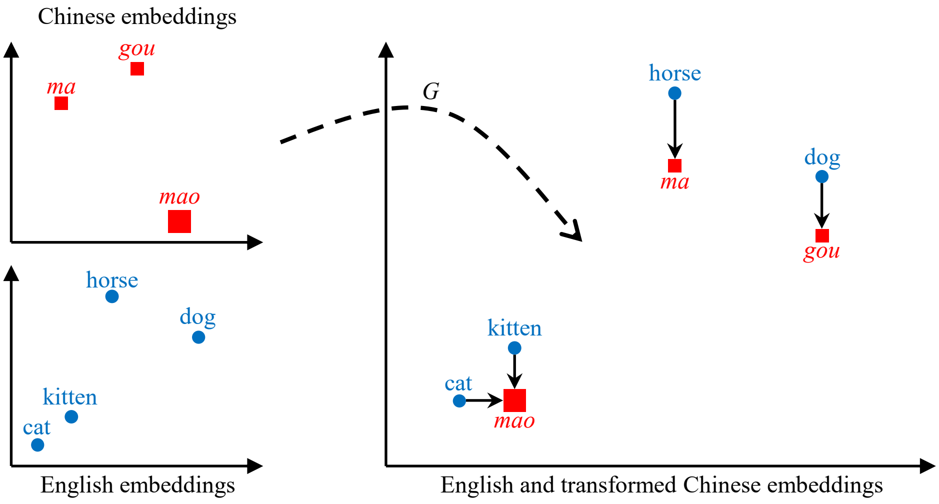


Figure 1: An illustration of our earth mover's distance minimization formulation. The subplots on the left schematically visualize Chinese and English embeddings. Due to isomorphism, there exists a simple transformation G that aligns the two embedding spaces well, as shown on the right. We expect to find the transformation G by minimizing the earth mover's distance without the need for cross-lingual word-level supervision, because the earth mover's distance holistically measures the closeness between two sets of weighted points. It computes the minimal cost of transporting one set of points to the other, whose weights are indicated by the sizes of squares and dots. We show the transport scheme in the right subplot with arrows, which can be interpreted as word translations.

2 Background

2.1 Aligning Isomorphic Embeddings

As discovered by previous work (Mikolov et al., 2013a), monolingual word embeddings exhibit isomorphism across languages, i.e., they appear similar in structure. However, as they are trained independently, the specific "orientation" of each embedding space is arbitrary, as illustrated in the left part of Figure 1. In order to connect the separate embedding spaces, we can try to transform the source embeddings so that they align well with target ones. Naturally, we need a measure for the quality of the alignment to guide our search for the transformation.

As we aim to eliminate the need for cross-lingual supervision from word translation pairs, the measure cannot be defined at the word level as in previous work (Mikolov et al., 2013a). Rather, it should quantify the difference between the entire distributions of embeddings. With this in mind, we find the earth mover's distance to be a suitable choice (Zhang et al., 2016b). Its workings are illustrated in the right part of Figure 1. We can think of target embeddings as piles of earth, and transformed source embeddings as holes to be filled. Then the earth mover's distance computes the minimal cost of moving the earth to fill the holes. Clearly, if the two sets of embeddings align well, the earth mover's distance will be small. Therefore, we can try to find the transformation that minimizes the earth mover's distance.

Another desirable feature of the earth mover's distance is that the computed transport scheme can be readily interpreted as translations. Moreover, this interpretation naturally handles multiple alternative translations. For example, the Chinese word "*mao*" can be translated to "cat" or "kitten", as shown in Figure 1.

2.2 The Form of the Transformation

The approximate isomorphism across embedding spaces inspires researchers to use a simple form of transformation. For example, Mikolov et al. (2013a) chose to use a linear transformation, i.e. the transformation G parametrized by a matrix. Later, proposals for using an orthogonal transformation are supported empirically (Xing et al., 2015; Zhang et al., 2016c; Artetxe et al., 2016) and theoretically (Smith et al., 2017). Indeed, an orthogonal transformation has desirable properties in this setting. If G is an orthogonal matrix that transforms the source embeddings into the target space, then its transpose (also its inverse) $G^\top$

performs transformation in the reverse direction. In that case, any word embedding a can be recovered by transforming back and forth because $G^{\top}Ga = a$. Moreover, computing the cosine similarity between a source embedding a and a target embedding b will be independent of the semantic space in which the similarity is measured, because $b^{\top}Ga / \|Ga\| \|b\| = a^{\top}G^{\top}b / \|a\| \left\|G^{\top}b\right\|$. Therefore we are inclined to use an orthogonal transformation for our task.

2.3 The Earth Mover's Distance

The earth mover's distance (EMD) is a powerful tool widely used in computer vision and natural language processing (Rubner et al., 1998; Kusner et al., 2015; Huang et al., 2016; Zhang et al., 2016b,a). Mathematically speaking, the EMD defines a distance between probability distributions. In the discrete case, a probability distribution can be represented by a sum of Dirac delta functions. For a pair of discrete distributions $\mathbb{P}_1 = \sum_i u_i \delta_{x_i}$ and $\mathbb{P}_2 = \sum_j v_j \delta_{y_j}$, the EMD is defined as

$$\mathrm{EMD}\left(\mathbb{P}_1, \mathbb{P}_2\right) = \min_{T \in \mathcal{U}(u,v)} \sum_i \sum_j T_{ij} c\left(x_i, y_j\right), \quad (1)$$

where $c\left(x_i, y_j\right)$ gives the ground distance between x_i and y_j, and $\mathcal{U}\left(u, v\right)$ is known as the transport polytope, defined as

$$\left\{ T | T_{ij} \geq 0, \sum_j T_{ij} = u_i, \sum_i T_{ij} = v_j, \forall i, j \right\}. \quad (2)$$

After solving the minimization program (1), the transport matrix T stores information of the transport scheme: A non-zero T_{ij} indicates the amount of probability mass transported from y_j to x_i. For our task, this can be interpreted as evidence for word translation (Zhang et al., 2016b), as indicated by arrows in the right part of Figure 1.

The EMD is closely related to the Wasserstein distance in mathematics, defined as

$$\mathrm{W}\left(\mathbb{P}_1, \mathbb{P}_2\right) = \inf_{\gamma \in \Gamma(\mathbb{P}_1, \mathbb{P}_2)} \mathbb{E}_{(x,y)\sim\gamma}\left[c\left(x, y\right)\right], \quad (3)$$

where $\Gamma\left(\mathbb{P}_1, \mathbb{P}_2\right)$ denotes the set of all joint distributions $\gamma\left(x, y\right)$ with marginals $\mathbb{P}_1$ and $\mathbb{P}_2$ on the first and second factors respectively. As we can see, the Wasserstein distance generalizes the EMD to allow continuous distributions. In our context, we will use both terms interchangeably.

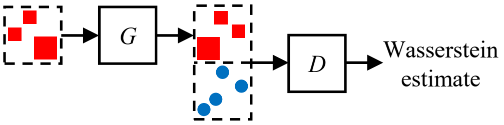


Figure 2: The Wasserstein GAN for unsupervised bilingual lexicon induction. The generator G transforms the source word embeddings into the target space. The critic D takes both sets of embeddings and tries to estimate their Wasserstein distance, and this information will be passed to the generator G during training to guide it towards minimizing the Wasserstein estimate.

3 Approaches

In our task, we are interested in a pair of distributions of word embeddings, one for the source language and the other for the target language. A source word embedding w_s^{S} is a d-dimensional column vector that represents the s-th source word in the V^{S}-sized source language vocabulary. Its distribution is characterized by a positive vector of frequencies f^{S} satisfying $\sum_{s=1}^{V^{\mathrm{S}}} f_s^{\mathrm{S}} = 1$, i.e. $P\left(w_s^{\mathrm{S}}\right) = f_s^{\mathrm{S}}$. Notations are similar for the target side. We assume the embeddings are normalized to have unit L_2 norm, which makes no difference to the result as we use cosine to measure semantic similarity.

Under this setting, we develop two approaches to our EMD minimization idea, called WGAN (Section 3.1) and EMDOT (Section 3.2) respectively.

3.1 Wasserstein GAN (WGAN)

Generative adversarial nets (GANs) are originally proposed to generate natural images (Goodfellow et al., 2014). They can generate sharp images if trained well, but they are notoriously difficult to train. Therefore, a lot of research efforts have been dedicated to the investigation into stabler training (Radford et al., 2015; Salimans et al., 2016; Nowozin et al., 2016; Metz et al., 2016; Poole et al., 2016; Arjovsky and Bottou, 2017), and the recently proposed Wasserstein GAN (Arjovsky et al., 2017) is a promising technique along this line of research.

While the original GAN is formulated as an adversarial game (hence its name), the Wasserstein GAN can be directly understood as minimizing the Wasserstein distance (3). Figure 2 illustrates the concept in the context of our unsupervised bilingual lexicon induction task. The generator G takes source word embeddings and transforms them, with the goal that the transformed source distribution $\mathbb{P}^{G(\mathrm{S})}$ and the target distribution $\mathbb{P}^{\mathrm{T}}$ should be close as measured by the Wasserstein distance. The critic D takes both transformed source word embeddings and target word embeddings and attempts to accurately estimate their Wasserstein distance, which will guide the generator during training. The overall objective is

$$\min_{G \in \mathbb{R}^{d \times d}} \mathrm{W}\left(\mathbb{P}^{G(\mathrm{S})}, \mathbb{P}^{\mathrm{T}}\right), \tag{4}$$

where $\mathbb{P}^{G(\mathrm{S})} = \sum_{s=1}^{V^{\mathrm{S}}} f_s^{\mathrm{S}} \delta_{G w_s^{\mathrm{S}}}$ and $\mathbb{P}^{\mathrm{T}} = \sum_{t=1}^{V^{\mathrm{T}}} f_t^{\mathrm{T}} \delta_{w_t^{\mathrm{T}}}$ are the distributions of transformed source word embeddings and target word embeddings. Here we do not impose the orthogonal constraint on G to facilitate the use of a gradient-based optimizer. With the ground distance c being Euclidean distance L_2, the Kantorovich-Rubinstein duality (Villani, 2009) gives

$$\begin{aligned} & \mathrm{W}\left(\mathbb{P}^{G(\mathrm{S})}, \mathbb{P}^{\mathrm{T}}\right) \\ = & \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{y \sim \mathbb{P}^{\mathrm{T}}}\left[f(y)\right] - \mathbb{E}_{y \sim \mathbb{P}^{G(\mathrm{S})}}\left[f(y)\right], \end{aligned} \tag{5}$$

where the supremum is over all K-Lipschitz functions f. As neural networks are universal function approximators (Hornik, 1991), we can attempt to approximate f with a neural network, called the critic D, with weight clipping to ensure the function family is K-Lipschitz. Therefore the objective of the critic is

$$\max_{D} \mathbb{E}_{y \sim \mathbb{P}^{\mathrm{T}}}\left[f_D(y)\right] - \mathbb{E}_{x \sim \mathbb{P}^{\mathrm{S}}}\left[f_D(Gx)\right]. \tag{6}$$

Conceptually, the critic D assigns scores f_D to real target embeddings and fake ones generated by the generator G. When the objective (6) is trained until optimality, the difference of the scores will approximate the Wasserstein distance up to a multiplicative constant. The generator G then aims to minimize the approximate distance, which leads to

$$\min_{G \in \mathbb{R}^{d \times d}} -\mathbb{E}_{x \sim \mathbb{P}^{\mathrm{S}}}\left[f_D(Gx)\right]. \tag{7}$$

3.2 EMD Minimization Under Orthogonal Transformation (EMDOT)

Alternative to minimizing the Wasserstein distance by duality, the primal program with the orthogonal constraint can be formalized as

$$\min_{G \in \mathcal{O}(d)} \mathrm{EMD}\left(\mathbb{P}^{G(\mathrm{S})}, \mathbb{P}^{\mathrm{T}}\right), \tag{8}$$

where $\mathcal{O}(d)$ is the orthogonal group in dimension d. The exact solution to this minimization program is NP-hard (Ding and Xu, 2016). Fortunately, an alternating minimization procedure is guaranteed to converge to a local minimum (Cohen and Guibas, 1999). Starting from an initial matrix $G^{(0)}$, we alternate between the following subprograms repeatedly:

$$T^{(k)} = \operatorname*{arg\,min}_{T \in \mathcal{U}(f^{\mathrm{S}}, f^{\mathrm{T}})} \sum_{s=1}^{V^{\mathrm{S}}} \sum_{t=1}^{V^{\mathrm{T}}} T_{st} c\left(G^{(k)} w_s^{\mathrm{S}}, w_t^{\mathrm{T}}\right), \tag{9}$$

$$G^{(k+1)} = \operatorname*{arg\,min}_{G \in \mathcal{O}(d)} \sum_{s=1}^{V^{\mathrm{S}}} \sum_{t=1}^{V^{\mathrm{T}}} T_{st}^{(k)} c\left(G w_s^{\mathrm{S}}, w_t^{\mathrm{T}}\right). \tag{10}$$

The minimization in (9) is the EMD program (1), with existing solvers available. For better scalability, we choose an approximate solver (Cuturi, 2013).

The minimization in (10) aims to find the transformation $G^{(k+1)}$ with cross-lingual connection provided in $T^{(k)}$. This is exactly the supervised scenario, and previous works typically resort to gradient-based solvers (Mikolov et al., 2013a). But they can be cumbersome especially as we impose the orthogonal constraint on G. Fortunately, if we choose the ground distance c to be the squared Euclidean distance L_2^2, the program (10) is an extension of the orthogonal Procrustes problem (Schönemann, 1966), which admits a closed-form solution:

$$G^{(k+1)} = UV^{\top}, \tag{11}$$

where U and V are obtained from a singular value decomposition (SVD):

$$\sum_{s=1}^{V^{\mathrm{S}}} \sum_{t=1}^{V^{\mathrm{T}}} T_{st}^{(k)} w_t^{\mathrm{T}} w_s^{\mathrm{S}\top} = USV^{\top}. \tag{12}$$

Note that the SVD is efficient because it is performed on a $d \times d$ matrix, which is typically low-dimensional. Choosing $c = L_2^2$ is also motivated by its equivalence to the cosine dissimilarity, as proved in Appendix A.

3.3 Discussion

Starting from the idea of earth mover's distance minimization, we have developed two approaches towards the goal. They employ different optimization techniques, which in turn lead to different practical choices. For example, we choose $c = L_2^2$ for the EMDOT approach to obtain a closed-form solution to the subprogram (10), otherwise we would have to use gradient-based solvers. In contrast, the WGAN approach calls for $c = L_2$ because the Kantorovich-Rubinstein duality takes a simple form only in this case.

The EMDOT approach is attractive for several reasons: It is consistent for training and testing (the equivalence between the ground distance $c = L_2^2$ and cosine dissimilarity), compatible with the orthogonal constraint, mathematically sound (without much assumption and approximation), guaranteed to converge, almost hyperparameter free, and fast in speed (the alternating subprograms have either effective approximate solvers or closed-form solutions). However, it suffers from a serious limitation: The alternating minimization procedure only converges to local minima, and they often turn out to be rather poor in practice.

Although the WGAN approach employs a stochastic-gradient-based optimizer (RMSProp) and does not guarantee global optima either, it works reasonably well in practice. It seems better at exploring the parameter space and finally landing in a neighborhood of a good optimum. Like other success stories of using stochastic-gradient-based optimizers to train neural networks, theoretical understanding of the behavior remains elusive.

We can enjoy the best of both worlds by incorporating the merits of both approaches: First the WGAN approach locates a good neighborhood of the parameter space, and then, starting from a reasonable initialization, the EMDOT approach efficiently explores the neighborhood to achieve enhanced performance.

4 Experiments

We first investigate the learning behavior of our WGAN approach, and then present experiments on the bilingual lexicon induction task, followed by a showcase of the earth mover's distance as a language distance measure. Details of the data sets and hyperparameters are described in Appendices B and C.

4.1 Learning Behavior of WGAN

We analyze the learning behavior of WGAN by looking at a typical training trajectory on Chinese-English. During training, we save 100 models, translate based on the nearest neighbor, and record their accuracy as the bilingual lexicon induction performance indicator at these training checkpoints. In theory, the critic objective (6) provides an estimate of the Wasserstein distance up to a multiplicative constant, and a smaller Wasserstein distance should mean the transformed source embedding space and the target embedding space align better, which should in turn result in a better bilingual lexicon. This is validated in Figure 3 by the correlation between Wasserstein estimate and accuracy. Therefore, the Wasserstein estimate can serve as an indicator for the bilingual lexicon induction performance, and we can save the model with the lowest value during training as the final model.

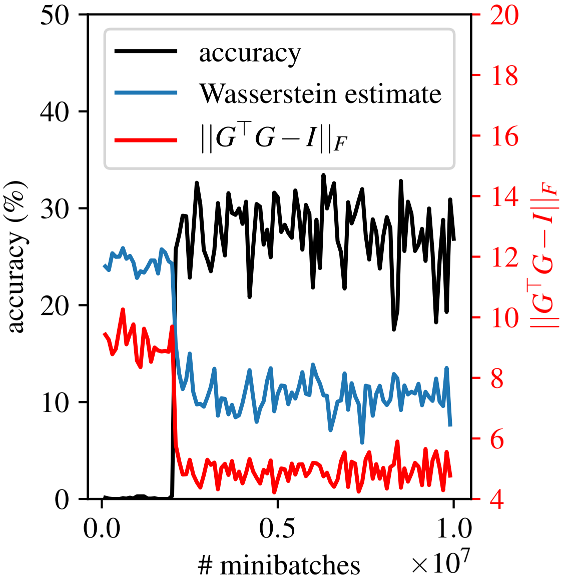


Figure 3: A typical training trajectory of WGAN. The three curves all correlate well. The Wasserstein estimate is rescaled because its magnitude is irrelevant.

In Figure 3, we also plot the value of $\left\|G^\top G - I\right\|_F$, which indicates the degree of orthogonality of the transformation matrix G. Interestingly, this also correlates nicely with the other curves, even though our WGAN formulation does not encourage G towards orthogonality. This finding confirms that a good transformation matrix is indeed close to orthogonality, and empirically justifies the orthogonal constraint for the EMDOT formulation.

Finally, we observe that the curves in Figure 3 are not very smooth. This means that although WGAN does well in exploring the parameter space and locating a reasonable transforma-

method	# seeds	zh-en	es-en	it-en	ja-zh	tr-en
TM	50	1.71	1.80	1.31	1.40	0.41
	100	17.27	24.93	23.22	22.91	15.02
	200	24.87	30.19	30.09	31.30	25.50
	500	28.24	32.11	31.69	35.79	32.63
IA	50	14.02	20.48	16.88	17.71	9.06
	100	22.14	28.73	25.99	28.24	18.37
	200	25.63	30.59	30.24	32.66	25.15
	500	27.21	31.94	31.54	35.33	31.50
WGAN	0	21.36	29.91	27.23	27.14	9.76
EMDOT	0	27.78	32.26	31.37	34.83	21.95

Table 1: F_1 scores for bilingual lexicon induction on Chinese-English, Spanish-English, Italian-English, Japanese-Chinese, and Turkish-English. The supervised methods TM and IA require seeds to train, and are listed for reference. Our EMDOT approach is initialized with the transformation found by WGAN, and consistently improves on it, reaching competitive performance with supervised methods.

tion matrix, it cannot stably refine the transformation. Fortunately, this is where EMDOT thrives, and hence combining them enjoys the benefits of both approaches.

4.2 Bilingual Lexicon Induction Performance

We test the quality of the cross-lingual transformation by evaluating on the bilingual lexicon induction task for five language pairs: Chinese-English, Spanish-English, Italian-English, Japanese-Chinese, and Turkish-English.

As the EMD automatically handles multiple alternative translations, we follow (Zhang et al., 2016b,a) to use F_1 score as the preferred evaluation metric.

Baselines

Our formulation is based on the isomorphism found across monolingual word embeddings. This idea has led to previous supervised methods:

- Translation matrix (TM) (Mikolov et al., 2013a): the pioneer of this type of methods, using linear transformation. We use a publicly available implementation.[1]
- Isometric alignment (IA) (Zhang et al., 2016c): an extension of TM by augmenting its learning objective with the isometric (orthogonal) constraint. Although Zhang et al. (2016c) had subsequent steps for their POS tagging task, it could be used for bilingual lexicon induction as well.

Although they need seed word translation pairs to train and thus not directly comparable to our system, we nonetheless report their results using {50, 100, 200, 500} seeds for a ballpark range of expected performance on this task, and skip the set of 500 seeds when testing all systems. We ensure the same input embeddings for these methods and ours. Their seeds are obtained through Google Translate (details in Appendix B.2). We apply the EMD as a postprocessing step (Zhang et al., 2016b) to allow them to handle multiple alternative translations. This is also done for our WGAN approach, as it does not produce the transport scheme to interpret as translation due to its duality formulation.

Results

Table 1 shows the F_1 scores on the five language pairs. As we can see, WGAN successfully finds a transformation that produces reasonable word translations. On top of that, EMDOT considerably improves the performance, which indicates that EMDOT refines the transformation found by WGAN.

Similar behavior across language pairs proves the generality of our approaches, as they build on embeddings learned from monolingual corpora without language-specific engineering. The quality of the embeddings, thus, will have an important effect on the performance, which may explain the lower scores on Turkish-English, as this low-resource setting may lack sufficient data to produce reliable embeddings. Higher noise levels in the preprocessing and ground truth for this lan-

[1] http://clic.cimec.unitn.it/~georgiana.dinu/down

	zh-en	es-en	it-en	ja-zh	tr-en
EMD	0.650	0.445	0.559	0.599	0.788
typology dissimilarity	0.467	0.342	0.259	0.433	0.541
geographical distance (km)	8161	1246	1464	2095	2854

Table 2: The earth mover's distance (EMD), typology dissimilarity, and geographical distance for Chinese-English, Spanish-English, Italian-English, Japanese-Chinese, and Turkish-English. The EMD shows correlation with both factors of linguistic difference.

guage pair (cf. the supplemental material), as well as the morphological richness of Turkish, may also be contributing factors to the relatively low scores.

Concerning the supervised methods TM and IA, they attain better performance with more supervision from seeds, as expected. For TM in particular, hundreds of seeds are needed for generalization, in line with the finding in (Vulić and Korhonen, 2016). Below that threshold, its performance drops dramatically, and this is when IA fares better with the orthogonal constraint. This indicates the importance of orthogonality when the seeds are few, or even zero as faced by our system. As the number of seeds increases, the performance of the supervised methods converges to a level comparable to our system.

4.3 The EMD as Language Distance

As our system minimizes the earth mover's distance between embeddings of two languages, we show here the final EMD can indicate the degree of difference between languages, serving as a proxy for language distance. Table 2 lists the EMD for the five language pairs considered in this paper, as well as their typology dissimilarity and geographical distance. The typology dissimilarity is computed from features in the WALS database (Dryer and Haspelmath, 2013). It is defined as one minus relative Hamming similarity, which is in turn defined as the number of agreeing features divided by the number of total features available for the language pair (Albu, 2006; Cysouw, 2013b). As a rough approximation, the geographical distance is measured by the distance between the capital cities of the countries where the considered languages are spoken (Eger et al., 2016).

The typology dissimilarity reflects genealogical influence on the divergence between languages, while the geographical distance indicates the effect of language contact. Both play important roles in shaping the languages we perceive today, and they also correlate with each other (Cysouw, 2013a). As we analyze Table 2, we find the EMD may be explained by both factors. Spanish-English and Italian-English are close both genealogically and geographically, and their EMD values are the lowest. English, Chinese, and Japanese belong to different language families, but the geographical proximity of the latter two enables intensive language contact, especially for the vocabularies, causing relatively smaller EMD. Finally, Turkish and English are distant in both aspects, and the EMD between them is large. Note that, however, the large EMD may also be caused by the relatively poor quality of monolingual embeddings due to low resource, and this should be a caveat of using the EMD to measure language distance.

5 Related Work

5.1 Bilingual Lexicon Induction

Bilingual lexicon induction is a long-standing research task in cross-lingual natural language processing. Traditional methods build statistical models for monolingual word co-occurrence, and combine cross-lingual supervision to solve the task. As word alignment for parallel sentences can produce fairly good bilingual lexica (Och and Ney, 2003), these methods focus on non-parallel data with a seed lexicon as cross-lingual supervision (Rapp, 1999; Gaussier et al., 2004).

An exception that does not rely on cross-lingual supervision is the decipherment approach (Dou and Knight, 2012, 2013; Dou et al., 2015). It views the source language as a cipher for the target language, and solves a statistical model that attempts to decipher the source language.

Following the popularity of monolingual word embeddings, cross-lingual word representation learning has also attracted significant attention in recent years. Building bilingual lexica from the learned cross-lingual embeddings is often considered an evaluative tool. Most methods rely on supervision encoded in parallel data, at the document level (Vulić and Moens, 2015), the sentence level (Zou et al., 2013; Chandar A P et al., 2014; Hermann and Blunsom, 2014; Kočiský et al., 2014; Gouws et al., 2015; Luong et al., 2015; Coulmance et al., 2015; Oshikiri et al., 2016), or the word level (i.e. in the form of seed lexicon) (Gouws and Søgaard, 2015; Wick et al., 2016; Duong et al., 2016; Shi et al., 2015; Mikolov et al., 2013a; Faruqui and Dyer, 2014; Lu et al., 2015; Dinu et al., 2015; Lazaridou et al., 2015; Ammar et al., 2016; Zhang et al., 2016a, 2017; Smith et al., 2017).

There is a recent work that aims to remove the need for cross-lingual supervision (Cao et al., 2016). Similar to ours, the underlying idea is to match cross-lingually at the level of distribution rather than word. However, the distributions considered in that work are the hidden states of neural embedding models during the course of training. They are assumed to be Gaussian, so that the matching of distributions reduces to matching their means and variances, but this assumption is hard to justify and interpret. In contrast, our proposal does not make any assumption on the distributions, and directly matches the transformed source embedding distribution with the target distribution by minimizing their earth mover's distance.

Another attempt to learn cross-lingual embedding transformation without supervision is (Barone, 2016). Architectures of generative adversarial nets and adversarial autoencoders (Makhzani et al., 2015) are experimented, but the reported results are not positive. We tried the publicly available code on our data and obtained negative results as well. This outcome is likely caused by the training difficulty pointed out by (Arjovsky and Bottou, 2017), as traditional GAN training minimizes Jensen-Shannon divergence between distributions, which can provide pathological gradient to the generator and hamper its learning. The use of Wasserstein GAN addresses this problem and allows our simple architecture to be trained successfully.

5.2 Language Distance

Quantifying language difference is an open question with on-going efforts that put forward better measures based on manually compiled data (Albu, 2006; Hammarström and O'Connor, 2013). Researchers in computational linguistics also try to contribute corpus-based approaches to this question. Parallel data is typically exploited, and ideas range from information-theoretic (Juola, 1998), statistical (Mayer and Cysouw, 2012), to graph-based (Eger et al., 2016; Asgari and Mofrad, 2016). To our knowledge, the earth mover's distance is proposed for language distance for the first time, with the distinctive feature of relying on non-parallel data only.

5.3 The Earth Mover's Distance

First introduced into computer vision (Rubner et al., 1998), the earth mover's distance also finds application in natural language processing (Kusner et al., 2015; Huang et al., 2016), including bilingual lexicon induction (Zhang et al., 2016b,a). Zhang et al. (2016b) build upon bilingual word embeddings and apply the EMD program as a postprocessing step to automatically produce multiple alternative translations. Later, Zhang et al. (2016a) introduce the EMD into the training objective of bilingual word embeddings as a regularizer. These previous works rely on cross-lingual supervision, and do not approach the task from the view of embedding transformation, while our work formulates the task as EMD minimization to allow zero supervision.

Apart from the usage as a regularizer (Zhang et al., 2016a), the EMD can also play other roles in optimization programs designed for various applications (Cuturi and Doucet, 2014; Frogner et al., 2015; Montavon et al., 2016).

6 Conclusion and Future Work

In this work, we attack the problem of finding cross-lingual transformation between monolingual word embeddings in a purely unsupervised setting. We introduce earth mover's distance minimization to tackle this task by exploiting its distribution-level matching to sidestep the requirement for word-level cross-lingual supervision. Even though zero supervision poses a clear challenge, our system attains competitive performance with supervised methods for bilingual lexicon induction. In addition, the earth mover's distance provides a natural measure that may prove helpful for quantifying language difference.

We have implemented the earth mover's distance minimization framework from two paths, and their combination has worked well, but both can be potentially improved by recent advances in optimization techniques (Gulrajani et al., 2017; Ding and Xu, 2016). Future work should also evaluate the earth mover's distance between more languages to assess its quality as language distance.

A Proof

The following proof shows that using squared Euclidean distance as the ground distance ($c = L_2^2$) is equivalent to using cosine dissimilarity when minimizing Equation (10).

$$\begin{aligned}
&\min_{G \in \mathcal{O}(d)} \sum_{s=1}^{V^{\mathrm{S}}} \sum_{t=1}^{V^{\mathrm{T}}} T_{st}^{(k)} \left\| G w_s^{\mathrm{S}} - w_t^{\mathrm{T}} \right\|^2 \\
=& \min_{G \in \mathcal{O}(d)} \sum_{s=1}^{V^{\mathrm{S}}} \sum_{t=1}^{V^{\mathrm{T}}} T_{st}^{(k)} \\
&\left(\left\| w_s^{\mathrm{S}} \right\|^2 + \left\| w_t^{\mathrm{T}} \right\|^2 - 2 w_t^{\mathrm{T}\top} G w_s^{\mathrm{S}} \right) \qquad (13) \\
=& \min_{G \in \mathcal{O}(d)} -2 \sum_{s=1}^{V^{\mathrm{S}}} \sum_{t=1}^{V^{\mathrm{T}}} T_{st}^{(k)} \cos\left(G w_s^{\mathrm{S}}, w_t^{\mathrm{T}} \right) \\
&+ \mathrm{const} \\
=& \min_{G \in \mathcal{O}(d)} - \sum_{s=1}^{V^{\mathrm{S}}} \sum_{t=1}^{V^{\mathrm{T}}} T_{st}^{(k)} \cos\left(G w_s^{\mathrm{S}}, w_t^{\mathrm{T}} \right).
\end{aligned}$$

B Data Preparation

B.1 Non-Parallel Corpora for Training Embeddings

The data for training monolingual word embeddings comes from Wikipedia comparable corpora.[2] Following (Vulić and Moens, 2013), we retain only nouns with at least 1,000 occurrences except for Turkish-English, whose frequency cutoff threshold is 100, as the amount of data is relatively small in this low-resource setting. For the Chinese side, we first use OpenCC[3] to normalize characters to be simplified, and then perform Chinese word segmentation and POS tagging with THULAC.[4] The preprocessing of the English side involves tokenization, POS tagging, lemmatization, and lowercasing, which we carry out with the NLTK toolkit[5] for the Chinese-English pair. For Spanish-English and Italian-English, we choose to use TreeTagger[6] for preprocessing, as in (Vulić and Moens, 2013). For the Japanese corpus, we use MeCab[7] for word segmentation and POS tagging. For Turkish, we utilize the preprocessing tools (tokenization and POS tagging) provided in LORELEI Language Packs (Strassel and Tracey, 2016), and its English side is preprocessed by NLTK. The statistics of the preprocessed corpora is given in Table 3.

		# tokens	vocabulary size
zh-en	zh	21m	3,349
	en	53m	5,154
es-en	es	61m	4,774
	en	95m	6,637
it-en	it	73m	8,490
	en	93m	6,597
ja-zh	ja	38m	6,043
	zh	16m	2,814
tr-en	tr	6m	7,482
	en	28m	13,220

Table 3: Statistics of the non-parallel corpora for training monolingual word embeddings. Language codes: zh = Chinese, en = English, es = Spanish, it = Italian, ja = Japanese, tr = Turkish.

B.2 Seed Word Translation Pairs

The seed word translation pairs for the translation matrix (`TM`) and isometric alignment (`IA`) approaches are obtained as follows. First, we ask Google Translate[8] to translate the source language vocabulary. Then the target translations are queried again and translated back to the source language, and those that do not match the original source words are discarded. This helps to ensure the translation quality. Finally, the translations are discarded if they fall out of our target language vocabulary.

B.3 Ground Truth

As the ground truth bilingual lexicon for evaluation, we use Chinese-English Translation Lexicon Version 3.0 (LDC2002L27) for the Chinese-English pair. For Spanish-English and Italian-English, we access Open Multilingual WordNet[9] through NLTK. For Japanese-Chinese, we use an in-house lexicon. For Turkish-English, we build a set of ground truth translation pairs in the same

[2] http://linguatools.org/tools/corpora/wikipedia-comparable-corpora

[3] https://github.com/BYVoid/OpenCC

[4] http://thulac.thunlp.org

[5] http://www.nltk.org

[6] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger

[7] http://taku910.github.io/mecab

[8] https://translate.google.com

[9] http://compling.hss.ntu.edu.sg/omw

way as how we obtain seed word translation pairs from Google Translate, described above.

C Hyperparameters

C.1 WGAN

We parametrize the critic D as a feed-forward neural network with one hidden layer of 500 neurons. The generator G is initialized with a random orthogonal matrix. The expectations in critic and generator objectives (6)(7) are approximated by minibatches of 1024 samples. We train for 10^7 minibatches. Most other hyperparameters follow from (Arjovsky et al., 2017) except the learning rates, for which larger values of 0.05 and 0.0005 are used for the generator and the critic respectively for faster convergence.

C.2 EMDOT

The approximate EMD solver (Cuturi, 2013) gives fairly accurate approximation with orders of magnitude speedup. However, it makes the transport matrix T no longer sparse. This is problematic, as we rely on interpreting a non-zero T_{st} as evidence to translate the s-th source word to the t-th target word (Zhang et al., 2016b). We therefore retain the largest pV^{S} elements of T, where p encodes our belief of the expected number of translations a source word can have. We set $p = 1.3$.

The alternating minimization procedure converges very fast. We run 10 iterations.

C.3 Monolingual Word Embeddings

As input monolingual word embeddings to the tested systems, we train the CBOW model (Mikolov et al., 2013b) with default hyperparameters in `word2vec`[10]. The embedding dimension d is 50.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. This work is supported by the National Natural Science Foundation of China (No. 61522204), the 973 Program (2014CB340501), and the National Natural Science Foundation of China (No. 61331013). This research is also supported by the Singapore National Research Foundation under its International Research Centre@Singapore Funding Initiative and administered by the IDM Programme.

[10] https://code.google.com/archive/p/word2vec

References

Mihai Albu. 2006. *Quantitative analyses of typological data*. Ph.D. thesis, Univ. Leipzig.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively Multilingual Word Embeddings. *arXiv:1602.01925 [cs]*.

Martin Arjovsky and Léon Bottou. 2017. Towards Principled Methods For Training Generative Adversarial Networks. In *ICLR*.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. *arXiv:1701.07875 [cs, stat]*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *EMNLP*.

Ehsaneddin Asgari and Mohammad R. K. Mofrad. 2016. Comparing Fifty Natural Languages and Twelve Genetic Languages Using Word Embedding Language Divergence (WELD) as a Quantitative Measure of Language Distance. In *Proceedings of the Workshop on Multilingual and Cross-lingual Methods in NLP*.

Antonio Valerio Miceli Barone. 2016. Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders. In *Proceedings of the 1st Workshop on Representation Learning for NLP*.

Hailong Cao, Tiejun Zhao, Shu Zhang, and Yao Meng. 2016. A Distribution-based Model to Learn Bilingual Word Embeddings. In *COLING*.

Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An Autoencoder Approach to Learning Bilingual Word Representations. In *NIPS*.

Scott Cohen and Leonidas Guibas. 1999. The Earth Mover's Distance Under Transformation Sets. In *ICCV*.

Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. 2015. Trans-gram, Fast Cross-lingual Word-embeddings. In *EMNLP*.

Marco Cuturi. 2013. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *NIPS*.

Marco Cuturi and Arnaud Doucet. 2014. Fast Computation of Wasserstein Barycenters. In *ICML*.

Michael Cysouw. 2013a. Disentangling geography from genealogy. *Space in language and linguistics: Geographical, interactional, and cognitive perspectives*.

Michael Cysouw. 2013b. Predicting language learning difficulty. *Approaches to measuring linguistic differences*.

Hu Ding and Jinhui Xu. 2016. FPTAS for Minimizing the Earth Mover's Distance Under Rigid Transformations and Related Problems. *Algorithmica*.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving Zero-Shot Learning by Mitigating the Hubness Problem. In *ICLR Workshop*.

Qing Dou and Kevin Knight. 2012. Large scale decipherment for out-of-domain machine translation. In *EMNLP-CoNLL*.

Qing Dou and Kevin Knight. 2013. Dependency-Based Decipherment for Resource-Limited Machine Translation. In *EMNLP*.

Qing Dou, Ashish Vaswani, Kevin Knight, and Chris Dyer. 2015. Unifying Bayesian Inference and Vector Space Models for Improved Decipherment. In *ACL-IJCNLP*.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning Crosslingual Word Embeddings without Bilingual Corpora. In *EMNLP*.

Steffen Eger, Armin Hoenen, and Alexander Mehler. 2016. Language classification from bilingual word embedding graphs. In *COLING*.

Manaal Faruqui and Chris Dyer. 2014. Improving Vector Space Word Representations Using Multilingual Correlation. In *EACL*.

Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya, and Tomaso A Poggio. 2015. Learning with a Wasserstein Loss. In *NIPS*.

Eric Gaussier, J.M. Renders, I. Matveeva, C. Goutte, and H. Dejean. 2004. A Geometric View on Bilingual Lexicon Extraction from Comparable Corpora. In *ACL*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS*.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast Bilingual Distributed Representations without Word Alignments. In *ICML*.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *NAACL-HLT*.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved Training of Wasserstein GANs. *arXiv:1704.00028 [cs, stat]*.

Harald Hammarström and Loretta O'Connor. 2013. Dependency-sensitive typological distance. *Approaches to measuring linguistic differences*.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual Distributed Representations without Word Alignment. In *ICLR*.

Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks*.

Gao Huang, Chuan Guo, Matt J Kusner, Yu Sun, Fei Sha, and Kilian Q Weinberger. 2016. Supervised Word Mover's Distance. In *NIPS*.

Patrick Juola. 1998. Cross-Entropy and Linguistic Typology. In *CoNLL*.

Tomáš Kočiský, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning Bilingual Word Representations by Marginalizing Alignments. In *ACL*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From Word Embeddings To Document Distances. In *ICML*.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and Pollution: Delving into Cross-Space Mapping for Zero-Shot Learning. In *ACL-IJCNLP*.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep Multilingual Correlation for Improved Word Embeddings. In *NAACL-HLT*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual Word Representations with Monolingual Quality in Mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial Autoencoders. *arXiv:1511.05644 [cs]*.

Thomas Mayer and Michael Cysouw. 2012. Language comparison through sparse multilingual word alignment. In *Proceedings of the EACL 2012 Joint Workshop of LINGVIS & UNCLH*.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. 2016. Unrolled Generative Adversarial Networks. *arXiv:1611.02163 [cs, stat]*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting Similarities among Languages for Machine Translation. *arXiv:1309.4168 [cs]*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*.

Grégoire Montavon, Klaus-Robert Müller, and Marco Cuturi. 2016. Wasserstein Training of Restricted Boltzmann Machines. In *NIPS*.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. *arXiv:1606.00709 [cs, stat]*.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *CL*.

Takamasa Oshikiri, Kazuki Fukui, and Hidetoshi Shimodaira. 2016. Cross-Lingual Word Representations via Spectral Graph Embeddings. In *ACL*.

Ben Poole, Alexander A. Alemi, Jascha Sohl-Dickstein, and Anelia Angelova. 2016. Improved generator objectives for GANs. *arXiv:1612.02780 [cs, stat]*. ArXiv: 1612.02780.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434 [cs]*.

Reinhard Rapp. 1999. Automatic Identification of Word Translations from Unrelated English and German Corpora. In *ACL*.

Y. Rubner, C. Tomasi, and L.J. Guibas. 1998. A Metric for Distributions with Applications to Image Databases. In *ICCV*.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved Techniques for Training GANs. In *NIPS*.

Peter H. Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*.

Tianze Shi, Zhiyuan Liu, Yang Liu, and Maosong Sun. 2015. Learning Cross-lingual Word Embeddings via Matrix Co-factorization. In *ACL-IJCNLP*.

Samuel Smith, David Turban, Steven Hamblin, and Nils Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *ICLR*.

Stephanie Strassel and Jennifer Tracey. 2016. LORELEI Language Packs: Data, Tools, and Resources for Technology Development in Low Resource Languages. In *LREC*.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and Type Constraints for Cross-Lingual Part-of-Speech Tagging. *TACL*.

Cédric Villani. 2009. *Optimal Transport: Old and New*.

Ivan Vulić and Anna Korhonen. 2016. On the Role of Seed Lexicons in Learning Bilingual Word Embeddings. In *ACL*.

Ivan Vulić and Marie-Francine Moens. 2013. Cross-Lingual Semantic Similarity of Words as the Similarity of Their Semantic Word Responses. In *NAACL-HLT*.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual Word Embeddings from Non-Parallel Document-Aligned Data Applied to Bilingual Lexicon Induction. In *ACL-IJCNLP*.

Michael Wick, Pallika Kanani, and Adam Pocock. 2016. Minimally-Constrained Multilingual Embeddings via Artificial Code-Switching. In *AAAI*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation. In *NAACL-HLT*.

Hyejin Youn, Logan Sutton, Eric Smith, Cristopher Moore, Jon F. Wilkins, Ian Maddieson, William Croft, and Tanmoy Bhattacharya. 2016. On the universal structure of human lexical semantics. *Proceedings of the National Academy of Sciences*.

Meng Zhang, Yang Liu, Huanbo Luan, Yiqun Liu, and Maosong Sun. 2016a. Inducing Bilingual Lexica From Non-Parallel Data With Earth Mover's Distance Regularization. In *COLING*.

Meng Zhang, Yang Liu, Huanbo Luan, Maosong Sun, Tatsuya Izuha, and Jie Hao. 2016b. Building Earth Mover's Distance on Bilingual Word Embeddings for Machine Translation. In *AAAI*.

Meng Zhang, Haoruo Peng, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Bilingual Lexicon Induction From Non-Parallel Data With Minimal Supervision. In *AAAI*.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016c. Ten Pairs to Tag – Multilingual POS Tagging via Coarse Mapping between Embeddings. In *NAACL-HLT*.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual Word Embeddings for Phrase-Based Machine Translation. In *EMNLP*.

Unfolding and Shrinking Neural Machine Translation Ensembles

Felix Stahlberg and **Bill Byrne**
{fs439,wjb31}@cam.ac.uk
Department of Engineering
University of Cambridge, UK

Abstract

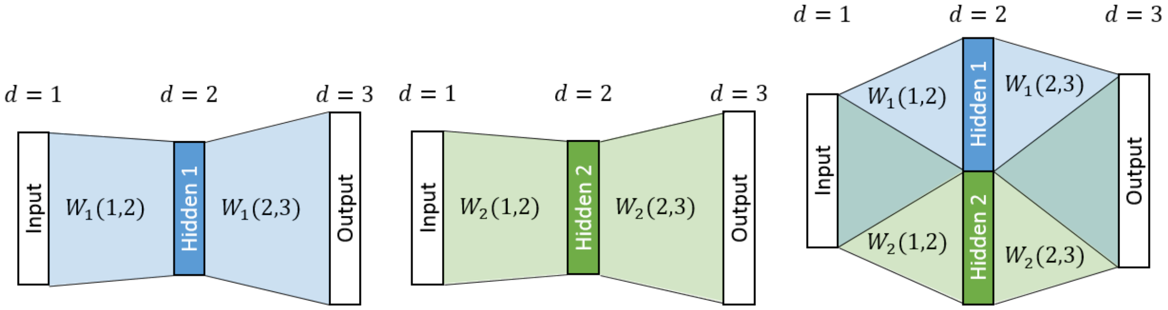
Ensembling is a well-known technique in neural machine translation (NMT) to improve system performance. Instead of a single neural net, multiple neural nets with the same topology are trained separately, and the decoder generates predictions by averaging over the individual models. Ensembling often improves the quality of the generated translations drastically. However, it is not suitable for production systems because it is cumbersome and slow. This work aims to reduce the runtime to be on par with a single system without compromising the translation quality. First, we show that the ensemble can be unfolded into a single large neural network which imitates the output of the ensemble system. We show that unfolding can already improve the runtime in practice since more work can be done on the GPU. We proceed by describing a set of techniques to shrink the unfolded network by reducing the dimensionality of layers. On Japanese-English we report that the resulting network has the size and decoding speed of a single NMT network but performs on the level of a 3-ensemble system.

1 Introduction

The top systems in recent machine translation evaluation campaigns on various language pairs use ensembles of a number of NMT systems (Bojar et al., 2016; Sennrich et al., 2016a; Chung et al., 2016; Neubig, 2016; Wu et al., 2016; Cromieres et al., 2016; Durrani et al., 2017). Ensembling (Dietterich, 2000; Hansen and Salamon, 1990) of neural networks is a simple yet very effective technique to improve the accuracy of NMT. The decoder makes use of K NMT networks which are either trained independently (Sutskever et al., 2014; Chung et al., 2016; Neubig, 2016; Wu et al., 2016) or share some amount of training iterations (Sennrich et al., 2016b,a; Cromieres et al., 2016; Durrani et al., 2017). The ensemble decoder computes predictions from each of the individual models which are then combined using the arithmetic average (Sutskever et al., 2014) or the geometric average (Cromieres et al., 2016).

Ensembling consistently outperforms single NMT by a large margin. However, the decoding speed is significantly worse since the decoder needs to apply K NMT models rather than only one. Therefore, a recent line of research transfers the idea of *knowledge distillation* (Bucilu et al., 2006; Hinton et al., 2014) to NMT and trains a smaller network (the student) by minimizing the cross-entropy to the output of the ensemble system (the teacher) (Kim and Rush, 2016; Freitag et al., 2017). This paper presents an alternative to knowledge distillation as we aim to speed up decoding to be comparable to single NMT while retaining the boost in translation accuracy from the ensemble. In a first step, we describe how to construct a single large neural network which imitates the output of an ensemble of multiple networks with the same topology. We will refer to this process as *unfolding*. GPU-based decoding with the unfolded network is often much faster than ensemble decoding since more work can be done on the GPU. In a second step, we explore methods to reduce the size of the unfolded network. This idea is justified by the fact that ensembled neural networks are often over-parameterized and have a large degree of redundancy (LeCun et al., 1989; Hassibi et al., 1993; Srinivas and Babu, 2015). Shrinking the unfolded network leads to a smaller model which consumes less space on the disk and in the memory; a crucial factor on mobile devices. More importantly, the

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1946–1956
Copenhagen, Denmark, September 7–11, 2017.



(a) Single network 1.

(b) Single network 2.

(c) Unfolded network.

Figure 1: Unfolding mimics the output of the ensemble of two single layer feedforward networks.

decoding speed on all platforms benefits greatly from the reduced number of neurons. We find that the dimensionality of linear embedding layers in the NMT network can be reduced heavily by low-rank matrix approximation based on singular value decomposition (SVD). This suggest that high dimensional embedding layers may be needed for training, but do not play an important role for decoding. The NMT network, however, also consists of complex layers like gated recurrent units (Cho et al., 2014, GRUs) and attention (Bahdanau et al., 2015). Therefore, we introduce a novel algorithm based on linear combinations of neurons which can be applied either during training (*data-bound*) or directly on the weight matrices without using training data (*data-free*). We report that with a mix of the presented shrinking methods we are able to reduce the size of the unfolded network to the size of the single NMT network while keeping the boost in BLEU score from the ensemble. Depending on the aggressiveness of shrinking, we report either a gain of 2.2 BLEU at the same decoding speed, or a 3.4× CPU decoding speed up with only a minor drop in BLEU compared to the original single NMT system. Furthermore, it is often much easier to stage a single NMT system than an ensemble in a commercial MT workflow, and it is crucial to be able to optimize quality at specific speed and memory constraints. Unfolding and shrinking address these problems directly.

2 Unfolding K Networks into a Single Large Neural Network

The first concept of our approach is called *unfolding*. Unfolding is an alternative to ensembling of multiple neural networks with the same topology. Rather than averaging their predictions, unfolding constructs a single large neural net out of the individual models which has the same number of input and output neurons but larger inner layers. Our main motivation for unfolding is to obtain a single network with ensemble level performance which can be shrunk with the techniques in Sec. 3.

Suppose we ensemble two single layer feedforward neural nets as shown in Fig. 1. Normally, ensembling is implemented by performing an isolated forward pass through the first network (Fig. 1(a)), another isolated forward pass through the second network (Fig. 1(b)), and averaging the activities in the output layers of both networks. This can be simulated by merging both networks into a single large network as shown in Fig. 1(c). The first neurons in the hidden layer of the combined network correspond to the hidden layer in the first single network, and the others to the hidden layer of the second network. A single pass through the combined network yields the same output as the ensemble if the output layer is linear (up to a factor 2). The weight matrices in the unfolded network can be constructed by stacking the corresponding weight matrices (either horizontally or vertically) in network 1 and 2. This kind of aggregation of multiple networks with the same topology is not only possible for single-layer feedforward architectures but also for complex networks consisting of multiple GRU layers and attention.

For a formal description of unfolding we address layers with indices $d = 0, 1, \ldots, D$. The special layer 0 has a single neuron for modelling bias vectors. Layer 1 holds the input neurons and layer D is the output layer. We denote the size of a layer in the individual models as $s(d)$. When combining K networks, the layer size $s'(d)$ in the unfolded network is increased by factor K if d is an inner layer, and equal to $s(d)$ if d is the in-

$$W'(d_1,d_2) = \begin{cases} \begin{pmatrix} W_1(d_1,d_2) & 0 & \cdots & 0 \\ 0 & W_2(d_1,d_2) & \vdots & \vdots \\ \vdots & \cdots & \ddots & 0 \\ 0 & \cdots & & W_K(d_1,d_2) \end{pmatrix} & \text{if } d_1 \in \text{InnerLayers and } d_2 \in \text{InnerLayers} \\ \frac{1}{K}\begin{pmatrix} W_1(d_1,d_2) \\ \vdots \\ W_K(d_1,d_2) \end{pmatrix} & \text{if } d_1 \in \text{InnerLayers and } d_2 \notin \text{InnerLayers} \\ \begin{pmatrix} W_1(d_1,d_2) & \cdots & W_K(d_1,d_2) \end{pmatrix} & \text{if } d_1 \notin \text{InnerLayers and } d_2 \in \text{InnerLayers} \end{cases}$$

Figure 2: General formula for unfolding weight matrices. The set InnerLayers $:= [2, D-1]$ includes all layers except the input, output, and bias layer.

put or output layer. We denote the weight matrix between two layers $d_1, d_2 \in [0, D]$ in the k-th individual model ($k \in [1, K]$) as $W_k(d_1, d_2) \in \mathbb{R}^{s(d_1)\times s(d_2)}$, and the corresponding weight matrix in the unfolded network as $W'(d_1, d_2) \in \mathbb{R}^{s'(d_1)\times s'(d_2)}$. We explicitly allow d_1 and d_2 to be non-consecutive or reversed to be able to model recurrent networks. We use the zero-matrix if layers d_1 and d_2 are not connected. The construction of the unfolded weight matrix $W'(d_1, d_2)$ from the individual matrices $W_k(d_1, d_2)$ depends on whether the connected layers are inner layers or not. The complete formula is listed in Fig. 2.

Unfolded NMT networks approximate but do not exactly match the output of the ensemble due to two reasons. First, the unfolded network synchronizes the attentions of the individual models. Each decoding step in the unfolded network computes a single attention weight vector. In contrast, ensemble decoding would compute one attention weight vector for each of the K input models. A second difference is that the ensemble decoder first applies the softmax at the output layer, and then averages the prediction probabilities. The unfolded network averages the neuron activities (i.e. the logits) first, and then applies the softmax function. Interestingly, as shown in Sec. 4, these differences do not have any impact on the BLEU score but yield potential speed advantages of unfolding since the computationally expensive softmax layer is only applied once.

3 Shrinking the Unfolded Network

After constructing the weight matrices of the unfolded network we reduce the size of it by iteratively shrinking layer sizes. In this section we denote the incoming weight matrix of the layer to shrink as $U \in \mathbb{R}^{m_{in}\times m}$ and the outgoing weight matrix as $V \in \mathbb{R}^{m\times m_{out}}$. Our procedure is inspired by the method of Srinivas and Babu (2015). They propose a criterion for removing neurons in inner layers of the network based on two intuitions. First, similarly to Hebb's learning rule, they detect redundancy by the principle *neurons which fire together, wire together*. If the incoming weight vectors $U_{:,i}$ and $U_{:,j}$ are exactly the same for two neurons i and j, we can remove the neuron j and add its outgoing connections to neuron i ($V_{i,:} \leftarrow V_{i,:} + V_{j,:}$) without changing the output.[1] This holds since the activity in neuron j will always be equal to the activity in neuron i. In practice, Srinivas and Babu use a distance measure based on the difference of the incoming weight vectors to search for similar neurons as exact matches are very rare.

The second intuition of the criterion used by Srinivas and Babu (2015) is that neurons with small outgoing weights contribute very little overall. Therefore, they search for a pair of neurons $i, j \in [1, m]$ according the following term and remove the j-th neuron.[2]

$$\arg\min_{i,j\in[1,m]} ||U_{:,i} - U_{:,j}||_2^2 ||V_{j,:}||_2^2 \tag{1}$$

Neuron j is selected for removal if (1) there is another neuron i which has a very similar set of incoming weights and if (2) j has a small outgoing weight vector. Their criterion is *data-free* since

[1] We denote the i-th row vector of a matrix A with $A_{i,:}$ and the i-th column vector as $A_{:,i}$.

[2] Note that the criterion in Eq. 1 generalizes the criterion of Srinivas and Babu (2015) to multiple outgoing weights. Also note that Srinivas and Babu (2015) propose some heuristic improvements to this criterion. However, these heuristics did not work well in our NMT experiments.

it does not require any training data. For further details we refer to Srinivas and Babu (2015).

3.1 Data-Free Neuron Removal

Srinivas and Babu (2015) propose to add the outgoing weights of j to the weights of a similar neuron i to compensate for the removal of j. However, we have found that this approach does not work well on NMT networks. We propose instead to compensate for the removal of a neuron by a linear combination of the remaining neurons in the layer. Data-free shrinking assumes for the sake of deriving the update rule that the neuron activation function is linear. We now ask the following question: How can we compensate as well as possible for the loss of neuron j such that the impact on the output of the whole network is minimized? Data-free shrinking represents the incoming weight vector of neuron j ($U_{:,j}$) as linear combination of the incoming weight vectors of the other neurons. The linear factors can be found by satisfying the following linear system:

$$U_{:,\neg j}\lambda = U_{:,j} \tag{2}$$

where $U_{:,\neg j}$ is matrix U without the j-th column. In practice, we use the method of ordinary least squares to find λ because the system may be overdetermined. The idea is that if we mix the outputs of all neurons in the layer by the λ-weights, we get the output of the j-th neuron. The row vector $V_{j,:}$ contains the contributions of the j-th neuron to each of the neurons in the next layer. Rather than using these connections, we approximate their effect by adding some weight to the outgoing connections of the other neurons. How much weight depends on λ and the outgoing weights $V_{j,:}$. The factor $D_{k,l}$ which we need to add to the outgoing connection of the k-th neuron to compensate for the loss of the j-th neuron on the l-th neuron in the next layer is:

$$D_{k,l} = \lambda_k V_{j,l} \tag{3}$$

Therefore, the update rule for V is:

$$V \leftarrow V + D \tag{4}$$

In the remainder we will refer to this method as *data-free* shrinking. Note that we recover the update rule of Srinivas and Babu (2015) by setting λ to the i-th unit vector. Also note that the error introduced by our shrinking method is due to the fact that we ignore the non-linearity, and that the solution for λ may not be exact. The method is error-free on linear layers as long as the residuals of the least-squares analysis in Eq. 2 are zero.

GRU layers The terminology of *neurons* needs some further elaboration for GRU layers which rather consist of update and reset gates and states (Cho et al., 2014). On GRU layers, we treat the states as neurons, i.e. the j-th neuron refers to the j-th entry in the GRU state vector. Input connections to the gates are included in the incoming weight matrix U for estimating λ in Eq. 2. Removing neuron j in a GRU layer means deleting the j-th entry in the states and both gate vectors.

3.2 Data-Bound Neuron Removal

Although we find our data-free approach to be a substantial improvement over the methods of Srinivas and Babu (2015) on NMT networks, it still leads to a non-negligible decline in BLEU score when applied to recurrent GRU layers. Our data-free method uses the incoming weights to identify similar neurons, i.e. neurons expected to have similar activities. This works well enough for simple layers, but the interdependencies between the states and the gates inside gated layers like GRUs or LSTMs are complex enough that redundancies cannot be found simply by looking for similar weights. In the spirit of Babaeizadeh et al. (2016), our *data-bound* version records neuron activities during training to estimate λ. We compensate for the removal of the j-th neuron by using a linear combination of the output of remaining neurons with similar activity patterns. In each layer, we prune 40 neurons each 450 training iterations until the target layer size is reached. Let A be the matrix which holds the records of neuron activities in the layer since the last removal. For example, for the decoder GRU layer, a batch size of 80, and target sentence lengths of 20, A has $20 \cdot 80 \cdot 450 = 720K$ rows and m (the number of neurons in the layer) columns.[3] Similarly to Eq. 2 we find interpolation weights λ using the method of least squares on the following linear system.

$$A_{:,\neg j}\lambda = A_{:,j} \tag{5}$$

The update rule for the outgoing weight matrix is the same as for our data-free method (Eq. 4).

[3] In practice, we use a random sample of 50K rows rather than the full matrix to keep the complexity of the least-squares analysis under control.

The key difference between data-free and data-bound shrinking is the way λ is estimated. Data-free shrinking uses the similarities between incoming weights, and data-bound shrinking uses neuron activities recorded during training. Once we select a neuron to remove, we estimate λ, compensate for the removal, and proceed with the shrunk network. Both methods are prior to any decoding and result in shrunk parameter files which are then loaded to the decoder. Both methods remove neurons rather than single weights.

The data-bound algorithm runs gradient-based optimization on the unfolded network. We use the AdaGrad (Duchi et al., 2011) step rule, a small learning rate of 0.0001, and aggressive step clipping at 0.05 to avoid destroying useful weights which were learned in the individual networks prior to the construction of the unfolded network.

Our data-bound algorithm uses a data-bound version of the neuron selection criterion in Eq. 1 which operates on the activity matrix A. We search for the pair $i, j \in [1, m]$ according the following term and remove neuron j.

$$\arg\min_{i,j\in[1,m]} ||A_{:,i} - A_{:,j}||_2^2 ||A_{:,j}||_2^2 \tag{6}$$

3.3 Shrinking Embedding Layers with SVD

The standard attention-based NMT network architecture (Bahdanau et al., 2015) includes three linear layers: the embedding layer in the encoder, and the output and feedback embedding layers in the decoder. We have found that linear layers are particularly easy to shrink using low-rank matrix approximation. As before we denote the incoming weight matrix as $U \in \mathbb{R}^{m_{in}\times m}$ and the outgoing weight matrix as $V \in \mathbb{R}^{m\times m_{out}}$. Since the layer is linear, we could directly connect the previous layer with the next layer using the product of both weight matrices $X = U \cdot V$. However, X may be very large. Therefore, we approximate X as a product of two low rank matrices $Y \in \mathbb{R}^{m_{in}\times m'}$ and $Z \in \mathbb{R}^{m'\times m_{out}}$ ($X \approx YZ$) where $m' \ll m$ is the desired layer size. A very common way to find such a matrix factorization is using truncated singular value decomposition (SVD). The layer is eventually shrunk by replacing U with Y and V with Z.

4 Results

The individual NMT systems we use as source for constructing the unfolded networks are trained using AdaDelta (Zeiler, 2012) on the Blocks/Theano implementation (van Merriënboer et al., 2015; Bastien et al., 2012) of the standard attention-based NMT model (Bahdanau et al., 2015) with: 1000 dimensional GRU layers (Cho et al., 2014) in both the decoder and bidrectional encoder; a single maxout output layer (Goodfellow et al., 2013); and 620 dimensional embedding layers. We follow Sennrich et al. (2016b) and use subword units based on byte pair encoding rather than words as modelling units. Our SGNMT decoder (Stahlberg et al., 2017)[4] with a beam size of 12 is used in all experiments. Our primary corpus is the Japanese-English (Ja-En) ASPEC data set (Nakazawa et al., 2016). We select a subset of 500K sentence pairs to train our models as suggested by Neubig et al. (2015). We report cased BLEU scores calculated with Moses' `multi-bleu.pl` to be strictly comparable to the evaluation done in the Workshop of Asian Translation (WAT). We also apply our method to the WMT data set for English-German (En-De), using the *news-test2014* as a development set, and keeping *news-test2015* and *news-test2016* as test sets. En-De BLEU scores are computed using `mteval-v13a.pl` as in the WMT evaluation. We set the vocabulary sizes to 30K for Ja-En and 50K for En-De. We also report the *size factor* for each model which is the total number of model parameters (sum of all weight matrix sizes) divided by the number of parameters in the original NMT network (86M for Ja-En and 120M for En-De). We choose a widely used, simple ensembling method (prediction averaging) as our baseline. We feel that the prevalence of this method makes it a reasonable baseline for our experiments.

Shrinking the Unfolded Network First, we investigate which shrinking methods are effective for which layers. Tab. 1 summarizes our results on a 2-unfold network for Ja-En, i.e. two separate NMT networks are combined in a single large network as described in Sec. 2. The layers in the combined network are shrunk to the size of the original networks using the methods discussed in Sec. 3.

Shrinking the linear embedding layers with SVD (Sec. 3.3) is very effective. The unfolded model with shrunk embedding layers performs at the same level as the ensemble (compare rows (b) and (c)). In our initial experiments, we applied the method of Srinivas and Babu (2015) to

[4] 'vanilla' decoding strategy

		Shrinking Methods								
	Base	**Encoder**		**Attention**	**Decoder**			**Size**	**BLEU**	
		Embed.	**GRUs**	**Match**	**GRU**	**Maxout**	**Embeds.**	**Factor**	**dev**	**test**
(a)	Single	-	-	-	-	-	-	1.00	20.8	23.5
(b)	2-Ens.	-	-	-	-	-	-	2×1.00	22.7	25.2
(c)	2-Unfold	SVD	-	-	-	-	SVD	1.85	22.7	25.1
(d)	2-Unfold	SVD	-	Data-Free	-	-	SVD	1.77	22.7	25.1
(e)	2-Unfold	SVD	Data-Free	Data-Free	Data-Free	-	SVD	1.05	21.6	24.2
(f)	2-Unfold	SVD	Data-Bound	Data-Free	Data-Bound	-	SVD	1.05	22.4	**25.3**
(g)	2-Unfold	SVD	Data-Bound	Data-Free	Data-Bound	Data-Free	SVD	1.00	16.9	19.3
(h)	2-Unfold	SVD	Data-Bound	Data-Free	Data-Bound	Data-Bound	SVD	1.00	21.9	24.6

Table 1: Shrinking layers of the unfolded network on Ja-En to their original size.

shrink the other layers, but their approach performed very poorly on this kind of network: the BLEU score dropped down to 15.5 on the development set when shrinking all layers except the decoder maxout and embedding layers, and to 9.9 BLEU when applying their method only to embedding layers.[5] Row (e) in Tab. 1 shows that our data-free algorithm from Sec. 3.1 is better suited for shrinking the GRU and attention layers, leading to a drop of only 1 BLEU point compared to the ensemble (b) (i.e. 0.8 BLEU better than the single system (a)). However, using the data-bound version of our shrinking algorithm (Sec. 3.2) for the GRU layers performs best.[6] The shrunk model yields about the same BLEU score as the ensemble on the test set (25.2 in (b) and 25.3 in (f)). Shrinking the maxout layer remains more of a challenge (rows (g) and (h)), but the number of parameters in this layer is small. Therefore, shrinking all layers except the maxout layer leads to almost the same number of parameters (factor 1.05 in row (f)) as the original NMT network (a), and thus to a similar storage size, memory consumption, and decoding speed, but with a 1.8 BLEU gain. Based on these results we fix the shrinking method used for each layer for all remaining experiments as follows: We shrink linear embedding layers with our SVD-based method, GRU layers with our data-bound method, the attention layer with our data-free method, and do not shrink the maxout layer.

Our data-bound algorithm from Sec. 3.2 has two mechanisms to compensate for the removal of a neuron. First, we use a linear combination of the remaining neurons to update the outgoing weight matrix by imitating its activations (Eq. 4). Second, stochastic gradient descent (SGD) fine-tunes all weights during this process. Tab. 2 demonstrates that both mechanisms are crucial for minimizing the effect of shrinking on the BLEU score.

	Compensation Method		**BLEU**	
	Linear Combination	**SGD**	**dev**	**test**
(a)			16.3	18.0
(b)	✓		22.1	24.3
(c)		✓	21.7	24.4
(d)	✓	✓	22.4	25.3

Table 2: Compensating for neuron removal in the data-bound algorithm. Row (d) corresponds to row (f) in Tab. 1.

Decoding Speed Our testing environment is an Ubuntu 16.04 with Linux 4.4.0 kernel, 32 GB RAM, an Intel® Core i7-6700 CPU at 3.40 GHz and an Nvidia GeForce GTX Titan X GPU. CPU decoding uses a single thread. We used the first 500 sentences of the Ja-En WAT development set for the time measurements.

Our results in Tab. 3 show that decoding with ensembles (rows (b) and (e)) is slow: combining the predictions of the individual models on the CPU is computationally expensive, and ensemble decoding requires K passes through the softmax layer which is also computationally expensive. Unfolding the ensemble into a single network and shrinking the embedding and attention layers improves the runtimes on the GPU significantly without noticeable impact on BLEU (rows (c) and (f)). This can be attributed to the fact that unfolding can reduce the communication overhead between CPU and GPU. Comparing rows (d) and (g) with row (a) reveals that shrinking the unfolded networks even further speeds up CPU and GPU decoding almost to the level of single system decoding. However, more aggressive shrinking yields a BLEU score of 25.3 when combining three systems (row (g)) – 1.8 BLEU better than the single system, but 0.6 BLEU worse than the 3-

[5] Results with the original method of Srinivas and Babu (2015) are not included in Tab. 1.

[6] If we apply different methods to different layers of the same network, we first apply SVD-based shrinking, then the data-free method, and finally the data-bound method.

	System	Words/Min.		Size	BLEU	
		CPU	GPU	Factor	dev	test
(a)	Single	323.4	2993.6	1.00	20.8	23.5
(b)	2-Ensemble	163.7	1641.1	2 × 1.00	22.7	25.2
(c)	2-Unfold, shrunk embed.& attention	157.2	2592.2	1.77	22.7	25.1
(d)	2-Unfold, shrunk all except maxout	308.3	2961.4	1.05	22.4	25.3
(e)	3-Ensemble	110.9	1158.2	3 × 1.00	23.4	25.9
(f)	3-Unfold, shrunk embed.& attention	95.4	2182.1	2.99	23.2	25.9
(g)	3-Unfold, shrunk all except maxout	301.6	3024.4	1.09	22.2	25.3

Table 3: Time measurements on Ja-En. Layers are shrunk to their size in the original NMT model.

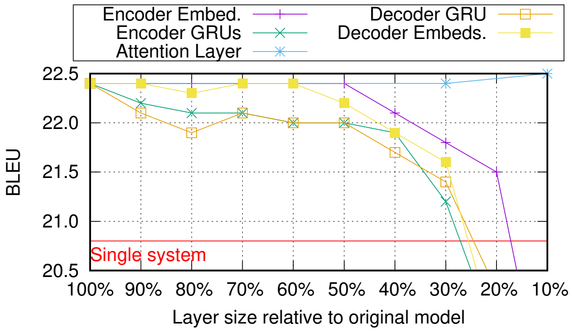


Figure 3: Impact of shrinking on the BLEU score.

	Single	3-Unfold		
		Normal	Small	Tiny
Enc. Embed.	620	410	310	170
Enc. GRUs	1000	1300	580	580
Attention	1000	100	100	100
Dec. GRU	1000	1350	590	590
Dec. Maxout	500	1500	1500	1500
Dec. Embeds.	620	430	320	170
Size Factor	1.00	1.00	0.50	0.33

Table 4: Layer sizes of our setups for Ja-En.

ensemble. Therefore, we will investigate the impact of shrinking on the different layers in the next sections more thoroughly.

Degrees of Redundancy in Different Layers We applied our shrinking methods to isolated layers in the 2-Unfold network of Tab. 1 (f). Fig. 3 plots the BLEU score when isolated layers are shrunk even below their size in the original NMT network. The attention layer is very robust against shrinking and can be reduced to 100 neurons (10% of the original size) without impacting the BLEU score. The embedding layers can be reduced to 60% but are sensitive to more aggressive pruning. Shrinking the GRU layers affects the BLEU score the most but still outperforms the single system when the GRU layers are shrunk to 30%.

Adjusting the Target Sizes of Layers Based on our previous experiments we revise our approach to shrink the 3-Unfold system in Tab. 3. Instead of shrinking all layers except the maxout layer to the same degree, we adjust the aggressiveness of shrinking for each layer. We suggest three different setups (*Normal*, *Small*, and *Tiny*) with the layer sizes specified in Tab. 4. *3-Unfold-Normal* has the same number of parameters as the original NMT networks (size factor: 1.0), *3-Unfold-Small* is only half their size (size factor: 0.5), and *3-Unfold-Tiny* reduces the size by two thirds (size factor: 0.33). When comparing rows (a) and (c) in Tab. 5 we observe that *3-Unfold-Normal* yields a gain of 2.2 BLEU with respect to the original single system and a slight improvement in decoding speed at the same time.[7] Networks with the size factor 1.0 like *3-Unfold-Normal* are very likely to yield about the same decoding speed as the *Single* network regardless of the decoder implementation, machine learning framework, and hardware. Therefore, we think that similar results are possible on other platforms as well.

	System	Words/Min.		BLEU	
		CPU	GPU	dev	test
(a)	Single	323.4	2993.6	20.8	23.5
(b)	3-Ensemble	110.9	1158.2	23.4	25.9
(c)	3-Unfold-Normal	445.2	3071.1	22.9	25.7
(d)	3-Unfold-Small	946.1	3572.0	21.7	23.9
(e)	3-Unfold-Tiny	1102.5	3483.7	20.6	23.2

Table 5: Our best models on Ja-En.

CPU decoding speed directly benefits even more from smaller setups – *3-Unfold-Tiny* is only 0.3 BLEU worse than *Single* but decoding on a single CPU is 3.4 times faster (row (a) vs. row (e) in Tab. 5). This is of great practical use: batch decoding with only two CPU threads surpasses production speed which is often set to 2000 words per minute (Beck et al., 2016). Our initial experiments in Tab. 6 suggest that the *Normal* setup is applicable to En-De as well, with substantial improve-

[7] To validate that the gains come from ensembling and unfolding and not from the layer sizes in *3-Unfold-Normal* we trained a network from scratch with the same dimensions. This network performed similarly to our *Single* system.

System	Wrds/Min.	BLEU on news-test*		
	(GPU)	2014	2015	2016
Single	2128.7	19.6	21.9	24.6
2-Ensemble	1135.3	20.5	22.9	26.1
2-Unfold-Norm.	2099.1	20.7	23.1	25.8

Table 6: Our best models on En-De.

ments in BLEU compared to *Single* with about the same decoding speed.

5 Related Work

The idea of pruning neural networks to improve the compactness of the models dates back more than 25 years (LeCun et al., 1989). The literature is therefore vast (Augasta and Kathirvalavakumar, 2013). One line of research aims to remove unimportant network connections. The connections can be selected for deletion based on the second-derivative of the training error with respect to the weight (LeCun et al., 1989; Hassibi et al., 1993), or by a threshold criterion on its magnitude (Han et al., 2015). See et al. (2016) confirmed a high degree of weight redundancy in NMT networks.

In this work we are interested in removing neurons rather than single connections since we strive to shrink the unfolded network such that it resembles the layout of an individual model. We argued in Sec. 4 that removing neurons rather than connections does not only improve the model size but also the memory footprint and decoding speed. As explained in Sec. 3.1, our data-free method is an extension of the approach by Srinivas and Babu (2015); our extension performs significantly better on NMT networks. Our data-bound method (Sec. 3.2) is inspired by Babaeizadeh et al. (2016) as we combine neurons with similar activities during training, but we use linear combinations of multiple neurons to compensate for the loss of a neuron rather than merging pairs of neurons.

Using low rank matrices for neural network compression, particularly approximations via SVD, has been studied widely in the literature (Denil et al., 2013; Denton et al., 2014; Xue et al., 2013; Prabhavalkar et al., 2016; Lu et al., 2016). These approaches often use low rank matrices to approximate a full rank weight matrix in the original network. In contrast, we shrink an entire linear layer by applying SVD on the product of the incoming and outgoing weight matrices (Sec. 3.3).

In this paper we mimicked the output of the high performing but cumbersome ensemble by constructing a large unfolded network, and shrank this network afterwards. Another approach, known as *knowledge distillation*, uses the large model (the teacher) to generate soft training labels for the smaller student network (Bucilu et al., 2006; Hinton et al., 2014). The student network is trained by minimizing the cross-entropy to the teacher. This idea has been applied to sequence modelling tasks such as machine translation and speech recognition (Wong and Gales, 2016; Kim and Rush, 2016; Freitag et al., 2017). Our approach can be computationally more efficient as the training set does not have to be decoded by the large teacher network.

Junczys-Dowmunt et al. (2016a; 2016b) reported gains from averaging the weight matrices of multiple checkpoints of the same training run. However, our attempts to replicate their approach were not successful. Averaging might work well when the behaviour of corresponding units is similar across networks, but that cannot be guaranteed when networks are trained independently.

6 Conclusion

We have described a generic method for improving the decoding speed and BLEU score of single system NMT. Our approach involves unfolding an ensemble of multiple systems into a single large neural network and shrinking this network by removing redundant neurons. Our best results on Japanese-English either yield a gain of 2.2 BLEU compared to the original single NMT network at about the same decoding speed, or a $3.4\times$ CPU decoding speed up with only a minor drop in BLEU.

The current formulation of unfolding works for networks of the same topology as the concatenation of layers is only possible for analogous layers in different networks. Unfolding and shrinking diverse networks could be possible, for example by applying the technique only to the input and output layers or by some other scheme of finding associations between units in different models, but we leave this investigation to future work as models in NMT ensembles in current research usually have the same topology (Bojar et al., 2016; Sennrich et al., 2016a; Chung et al., 2016; Neubig, 2016; Wu et al., 2016; Durrani et al., 2017).

Acknowledgments

This work was supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC grant EP/L027623/1).

Appendix: Probabilistic Interpretation of Data-Free and Data-Bound Shrinking

Data-free and data-bound shrinking can be interpreted as setting the expected difference between network outputs before and after a removal operation to zero under different assumptions.

For simplicity, we focus our probabilistic treatment of shrinking on single layer feedforward networks. Such a network maps an input $\mathbf{x} \in \mathbb{R}^{m_{in}}$ to an output $\mathbf{y} \in \mathbb{R}^{m_{out}}$. The l-th output y_l is computed according the following equation

$$y_l = \sum_{k \in [1,m]} \sigma(\mathbf{x}\mathbf{u}_k^T) V_{k,l} \tag{7}$$

where $\mathbf{u}_k \in \mathbb{R}^{m_{in}}$ is the incoming weight vector of the k-th hidden neuron (denoted as $U_{:,k}$ in the main paper) and $V \in \mathbb{R}^{m \times m_{out}}$ the outgoing weight matrix of the m-dimensional hidden layer. We now remove the j-th neuron in the hidden layer and modify the outgoing weights to compensate for the removal:

$$y'_l = \sum_{k \in [1,m] \backslash \{j\}} \sigma(\mathbf{x}\mathbf{u}_k^T) V'_{k,l} \tag{8}$$

where y'_l is the output after the removal operation and $V' \in \mathbb{R}^{m \times m_{out}}$ are the modified outgoing weights. Our goal is to choose V' such that the expected error introduced by removing neuron j is zero:

$$\mathbb{E}_{\mathbf{x}}(y_l - y'_l) = 0 \tag{9}$$

Data-free shrinking Data-free shrinking makes two assumptions to satisfy Eq. 9. First, we assume that the incoming weight vector $\mathbf{u}_j$ can be represented as linear combination of the other weight vectors.

$$\mathbf{u}_j = \sum_{k \in [1,m] \backslash \{j\}} \lambda_k \mathbf{u}_k \tag{10}$$

Second, it assumes that the neuron activation function $\sigma(\cdot)$ is linear. Starting with Eqs. 7 and 8 we can write $\mathbb{E}_{\mathbf{x}}(y_l - y'_l)$ as

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x}}\Big(\sigma(\mathbf{x}\mathbf{u}_j^T)V_{j,l} + \underbrace{\sum_{k \in [1,m] \backslash \{j\}} \sigma(\mathbf{x}\mathbf{u}_k^T)(V_{k,l} - V'_{k,l})\Big)}_{:=R} \\
\overset{\text{Eq. 10}}{=} \quad & \mathbb{E}_{\mathbf{x}}\Big(\sigma(\mathbf{x}(\sum_{k \in [1,m] \backslash \{j\}} \lambda_k \mathbf{u}_k)^T)V_{j,l} + R\Big) \\
\overset{\sigma(\cdot)\text{ lin.}}{=} \quad & \mathbb{E}_{\mathbf{x}}\Big(\sum_{k \in [1,m] \backslash \{j\}} \sigma(\mathbf{x}\mathbf{u}_k^T)\lambda_k V_{j,l} + R\Big) \\
= \quad & \sum_{k \in [1,m] \backslash \{j\}} \mathbb{E}_{\mathbf{x}}\Big(\sigma(\mathbf{x}\mathbf{u}_k^T)\Big)(V_{k,l} - V'_{k,l} + \lambda_k V_{j,l})
\end{aligned}$$

We set this term to zero (and thus satisfy Eq. 9) by setting each component of the sum to zero.

$$\forall k \in [1,m] \setminus \{j\} : V'_{k,l} = V_{k,l} + \lambda_k V_{j,l} \tag{11}$$

This condition is directly implemented by the update rule in our shrinking algorithm (Eq. 3 and 4).

Data-bound shrinking Data-bound shrinking does not require linearity in $\sigma(\cdot)$. It rather assumes that the expected value of the neuron activity j is a linear combination of the expected values of the other activities:

$$\mathbb{E}_{\mathbf{x}}(\sigma(\mathbf{x}\mathbf{u}_j^T)) = \sum_{k \in [1,m] \backslash \{j\}} \lambda_k \mathbb{E}_{\mathbf{x}}(\sigma(\mathbf{x}\mathbf{u}_k^T)) \tag{12}$$

$\mathbb{E}_{\mathbf{x}}(\cdot)$ is estimated using importance sampling:

$$\hat{\mathbb{E}}_{\mathbf{x}}(\sigma(\mathbf{x}\mathbf{u}_k^T); \mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x}' \in \mathcal{X}} \sigma(\mathbf{x}'\mathbf{u}_k^T) \tag{13}$$

In practice, the samples in $\mathcal{X}$ are collected in the activity matrix A from Sec. 3.2. We can satisfy Eq. 9 by using the λ-values from Eq. 12, so that $\mathbb{E}_{\mathbf{x}}(y_l - y'_l)$ becomes

$$\begin{aligned}
\overset{\text{Eqs. 7,8}}{=} \quad & \mathbb{E}_{\mathbf{x}}\Big(\sigma(\mathbf{x}\mathbf{u}_j^T)V_{j,l} \\
& + \sum_{k \in [1,m] \backslash \{j\}} \sigma(\mathbf{x}\mathbf{u}_k^T)(V_{k,l} - V'_{k,l})\Big) \\
= \quad & \mathbb{E}_{\mathbf{x}}(\sigma(\mathbf{x}\mathbf{u}_j^T)V_{j,l}) \\
& + \sum_{k \in [1,m] \backslash \{j\}} \mathbb{E}_{\mathbf{x}}(\sigma(\mathbf{x}\mathbf{u}_k^T))(V_{k,l} - V'_{k,l}) \\
\overset{\text{Eq. 12}}{=} \quad & \sum_{k \in [1,m] \backslash \{j\}} \mathbb{E}_{\mathbf{x}}(\sigma(\mathbf{x}\mathbf{u}_k^T))(V_{k,l} - V'_{k,l} + \lambda_k V_{j,l})
\end{aligned}$$

Again, we set this to zero using Eq. 11.

References

M. Gethsiyal Augasta and Thangairulappan Kathirvalavakumar. 2013. Pruning algorithms of neural networks – a comparative study. *Central European Journal of Computer Science*, 3(3):105–115.

Mohammad Babaeizadeh, Paris Smaragdis, and Roy H. Campbell. 2016. NoiseOut: A simple way to prune neural networks. In *Proceedings of the 1st International Workshop on Efficient Methods for Deep Neural Networks (EMDNN)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*, Toulon, France.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. 2012. Theano: new features and speed improvements. In *NIPS*, South Lake Tahoe, Nevada, USA.

Daniel Beck, Adrià de Gispert, Gonzalo Iglesias, Aurelien Waite, and Bill Byrne. 2016. Speed-constrained tuning for statistical machine translation using Bayesian optimization. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 856–865, San Diego, California. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1693–1703, Berlin, Germany. Association for Computational Linguistics.

Fabien Cromieres, Chenhui Chu, Toshiaki Nakazawa, and Sadao Kurohashi. 2016. Kyoto university participation to WAT 2016. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, pages 166–174, Osaka, Japan. The COLING 2016 Organizing Committee.

Misha Denil, Babak Shakibi, Laurent Dinh, Nando de Freitas, et al. 2013. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems*, pages 2148–2156.

Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277.

Thomas G. Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, pages 2121–2159.

Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and Stephan Vogel. 2017. QCRI machine translation systems for IWSLT 16. *arXiv preprint arXiv:1701.03924*.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. 2017. Ensemble distillation for neural machine translation. *arXiv preprint arXiv:1702.01802*.

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *ICML*, pages 1319–1327.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143.

Lars Kai Hansen and Peter Salamon. 1990. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001.

Babak Hassibi, David G. Stork, et al. 1993. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, pages 164–164.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016a. Is neural machine translation ready for deployment? A case study on 30 translation directions. In *International Workshop on Spoken Language Translation IWSLT*.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Rico Sennrich. 2016b. The AMU-UEDIN submission to the WMT16 news translation task: Attention-based NMT models as feature functions in phrase-based SMT. In *Proceedings of the First Conference on Machine Translation*, pages 319–325, Berlin, Germany. Association for Computational Linguistics.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Yann LeCun, John S. Denker, Sara A. Solla, Richard E. Howard, and Lawrence D. Jackel. 1989. Optimal brain damage. In *NIPS*, volume 2, pages 598–605.

Zhiyun Lu, Vikas Sindhwani, and Tara N. Sainath. 2016. Learning compact recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5960–5964. IEEE.

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. 2015. Blocks and Fuel: Frameworks for deep learning. *CoRR*.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. ASPEC: Asian scientific paper excerpt corpus. In *LREC*, pages 2204–2208, Portoroz, Slovenia.

Graham Neubig. 2016. Lexicons and minimum risk training for neural machine translation: NAIST-CMU at WAT2016. In *Proceedings of the 3rd Workshop on Asian Translation (WAT2016)*, pages 119–125, Osaka, Japan. The COLING 2016 Organizing Committee.

Graham Neubig, Makoto Morishita, and Satoshi Nakamura. 2015. Neural reranking improves subjective quality of machine translation: NAIST at WAT2015. In *Workshop on Asian Translation*, pages 35–41.

Rohit Prabhavalkar, Ouais Alsharif, Antoine Bruguier, and Lan McGraw. 2016. On the compression of recurrent neural networks with an application to LVCSR acoustic modeling for embedded speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5970–5974. IEEE.

Abigail See, Minh-Thang Luong, and D. Christopher Manning. 2016. Compression of neural machine translation models via pruning. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 291–301. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Suraj Srinivas and R. Venkatesh Babu. 2015. Data-free parameter pruning for deep neural networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 31.1–31.12. BMVA Press.

Felix Stahlberg, Eva Hasler, Danielle Saunders, and Bill Byrne. 2017. SGNMT – A flexible NMT decoding platform for quick prototyping of new models and search strategies. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pages 3104–3112. MIT Press.

Jeremy HM. Wong and Mark JF. Gales. 2016. Sequence student-teacher training of deep neural networks. *Interspeech 2016*, pages 2761–2765.

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

Jian Xue, Jinyu Li, and Yifan Gong. 2013. Restructuring of deep neural network acoustic models with singular value decomposition. In *Interspeech*, pages 2365–2369.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Graph Convolutional Encoders for Syntax-aware Neural Machine Translation

Joost Bastings[1] Ivan Titov[1,2] Wilker Aziz[1]
Diego Marcheggiani[1] Khalil Sima'an[1]
[1]ILLC, University of Amsterdam [2]ILCC, University of Edinburgh
{bastings,titov,w.aziz,marcheggiani,k.simaan}@uva.nl

Abstract

We present a simple and effective approach to incorporating syntactic structure into neural attention-based encoder-decoder models for machine translation. We rely on graph-convolutional networks (GCNs), a recent class of neural networks developed for modeling graph-structured data. Our GCNs use predicted syntactic dependency trees of source sentences to produce representations of words (i.e. hidden states of the encoder) that are sensitive to their syntactic neighborhoods. GCNs take word representations as input and produce word representations as output, so they can easily be incorporated as layers into standard encoders (e.g., on top of bidirectional RNNs or convolutional neural networks). We evaluate their effectiveness with English-German and English-Czech translation experiments for different types of encoders and observe substantial improvements over their syntax-agnostic versions in all the considered setups.

1 Introduction

Neural machine translation (NMT) is one of success stories of deep learning in natural language processing, with recent NMT systems outperforming traditional phrase-based approaches on many language pairs (Sennrich et al., 2016a). State-of-the-art NMT systems rely on sequential encoder-decoders (Sutskever et al., 2014; Bahdanau et al., 2015) and lack any explicit modeling of syntax or any hierarchical structure of language. One potential reason for why we have not seen much benefit from using syntactic information in NMT is the lack of simple and effective methods for incorporating structured information in neural encoders, including RNNs. Despite some successes, techniques explored so far either incorporate syntactic information in NMT models in a relatively indirect way (e.g., multi-task learning (Luong et al., 2015a; Nadejde et al., 2017; Eriguchi et al., 2017; Hashimoto and Tsuruoka, 2017)) or may be too restrictive in modeling the interface between syntax and the translation task (e.g., learning representations of linguistic phrases (Eriguchi et al., 2016)). Our goal is to provide the encoder with access to rich syntactic information but let it decide which aspects of syntax are beneficial for MT, without placing rigid constraints on the interaction between syntax and the translation task. This goal is in line with claims that rigid syntactic constraints typically hurt MT (Zollmann and Venugopal, 2006; Smith and Eisner, 2006; Chiang, 2010), and, though these claims have been made in the context of traditional MT systems, we believe they are no less valid for NMT.

Attention-based NMT systems (Bahdanau et al., 2015; Luong et al., 2015b) represent source sentence words as latent-feature vectors in the encoder and use these vectors when generating a translation. Our goal is to automatically incorporate information about syntactic neighborhoods of source words into these feature vectors, and, thus, potentially improve quality of the translation output. Since vectors correspond to words, it is natural for us to use dependency syntax. Dependency trees (see Figure 1) represent syntactic relations between words: for example, *monkey* is a subject of the predicate *eats*, and *banana* is its object.

In order to produce syntax-aware feature representations of words, we exploit graph-convolutional networks (GCNs) (Duvenaud et al., 2015; Defferrard et al., 2016; Kearnes et al., 2016; Kipf and Welling, 2016). GCNs can be regarded as computing a latent-feature representation of a node (i.e. a real-valued vector) based on its k-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1957–1967
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

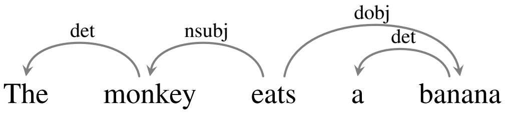


Figure 1: A dependency tree for the example sentence: "*The monkey eats a banana.*"

th order neighborhood (i.e. nodes at most k hops aways from the node) (Gilmer et al., 2017). They are generally simple and computationally inexpensive. We use Syntactic GCNs, a version of GCN operating on top of syntactic dependency trees, recently shown effective in the context of semantic role labeling (Marcheggiani and Titov, 2017).

Since syntactic GCNs produce representations at word level, it is straightforward to use them as encoders within the attention-based encoder-decoder framework. As NMT systems are trained end-to-end, GCNs end up capturing syntactic properties specifically relevant to the translation task. Though GCNs can take word embeddings as input, we will see that they are more effective when used as layers on top of recurrent neural network (RNN) or convolutional neural network (CNN) encoders (Gehring et al., 2016), enriching their states with syntactic information. A comparison to RNNs is the most challenging test for GCNs, as it has been shown that RNNs (e.g., LSTMs) are able to capture certain syntactic phenomena (e.g., subject-verb agreement) reasonably well on their own, without explicit treebank supervision (Linzen et al., 2016; Shi et al., 2016). Nevertheless, GCNs appear beneficial even in this challenging set-up: we obtain +1.2 and +0.7 BLEU point improvements from using syntactic GCNs on top of bidirectional RNNs for English-German and English-Czech, respectively.

In principle, GCNs are flexible enough to incorporate any linguistic structure as long as they can be represented as graphs (e.g., dependency-based semantic-role labeling representations (Surdeanu et al., 2008), AMR semantic graphs (Banarescu et al., 2012) and co-reference chains). For example, unlike recursive neural networks (Socher et al., 2013), GCNs do not require the graphs to be trees. However, in this work we solely focus on dependency syntax and leave more general investigation for future work.

Our main contributions can be summarized as follows:

- we introduce a method for incorporating structure into NMT using syntactic GCNs;
- we show that GCNs can be used along with RNN and CNN encoders;
- we show that incorporating structure is beneficial for machine translation on English-Czech and English-German.

2 Background

Notation. We use $\mathbf{x}$ for vectors, $\mathbf{x}_{1:t}$ for a sequence of t vectors, and X for matrices. The i-th *value* of vector $\mathbf{x}$ is denoted by x_i. We use $\circ$ for vector concatenation.

2.1 Neural Machine Translation

In NMT (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014b), given example translation pairs from a parallel corpus, a neural network is trained to directly estimate the conditional distribution $p(y_{1:T_y}|x_{1:T_x})$ of translating a source sentence $x_{1:T_x}$ (a sequence of T_x words) into a target sentence $y_{1:T_y}$. NMT models typically consist of an encoder, a decoder and some method for conditioning the decoder on the encoder, for example, an attention mechanism. We will now briefly describe the components that we use in this paper.

2.1.1 Encoders

An encoder is a function that takes as input the source sentence and produces a representation encoding its semantic content. We describe recurrent, convolutional and bag-of-words encoders.

Recurrent. Recurrent neural networks (RNNs) (Elman, 1990) model sequential data. They receive one input vector at each time step and update their hidden state to summarize all inputs up to that point. Given an input sequence $\mathbf{x}_{1:T_x} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{T_x}$ of word embeddings an RNN is defined recursively as follows:

$$\mathrm{RNN}(\mathbf{x}_{1:t}) = f(\mathbf{x}_t, \mathrm{RNN}(\mathbf{x}_{1:t-1}))$$

where f is a nonlinear function such as an LSTM (Hochreiter and Schmidhuber, 1997) or a GRU (Cho et al., 2014b). We will use the function RNN as an abstract mapping from an input sequence $\mathbf{x}_{1:T}$ to final hidden state $\mathrm{RNN}(\mathbf{x}_{1:T_x})$, regardless of the used nonlinearity. To not only summarize the past of a word, but also its future, a bidirectional RNN (Schuster and Paliwal, 1997; Irsoy and

Cardie, 2014) is often used. A bidirectional RNN reads the input sentence in two directions and then concatenates the states for each time step:

$$\text{BiRNN}(\mathbf{x}_{1:T_x}, t) = \text{RNN}_F(\mathbf{x}_{1:t}) \circ \text{RNN}_B(\mathbf{x}_{T_x:t})$$

where RNN_F and RNN_B are the forward and backward RNNs, respectively. For further details we refer to the encoder of Bahdanau et al. (2015).

Convolutional. Convolutional Neural Networks (CNNs) apply a fixed-size window over the input sequence to capture the local context of each word (Gehring et al., 2016). One advantage of this approach over RNNs is that it allows for fast parallel computation, while sacrificing non-local context. To remedy the loss of context, multiple CNN layers can be stacked. Formally, given an input sequence $\mathbf{x}_{1:T_x}$, we define a CNN as follows:

$$\text{CNN}(\mathbf{x}_{1:T_x}, t) = f(\mathbf{x}_{t-\lfloor w/2 \rfloor}, .., \mathbf{x}_t, .., \mathbf{x}_{t+\lfloor w/2 \rfloor})$$

where f is a nonlinear function, typically a linear transformation followed by ReLU, and w is the size of the window.

Bag-of-Words. In a bag-of-words (BoW) encoder every word is simply represented by its word embedding. To give the decoder some sense of word position, position embeddings (PE) may be added. There are different strategies for defining position embeddings, and in this paper we choose to learn a vector for each absolute word position up to a certain maximum length. We then represent the t-th word in a sequence as follows:

$$\text{BoW}(\mathbf{x}_{1:T_x}, t) = \mathbf{x}_t + \mathbf{p}_t$$

where $\mathbf{x}_t$ is the word embedding and $\mathbf{p}_t$ is the t-th position embedding.

2.1.2 Decoder

A decoder produces the target sentence conditioned on the representation of the source sentence induced by the encoder. In Bahdanau et al. (2015) the decoder is implemented as an RNN conditioned on an additional input $\mathbf{c}_i$, the context vector, which is dynamically computed at each time step using an attention mechanism.

The probability of a target word y_i is now a function of the decoder RNN state, the previous target word embedding, and the context vector. The model is trained end-to-end for maximum log likelihood of the next target word given its context.

2.2 Graph Convolutional Networks

We will now describe the Graph Convolutional Networks (GCNs) of Kipf and Welling (2016). For a comprehensive overview of alternative GCN architectures see Gilmer et al. (2017).

A GCN is a multilayer neural network that operates directly on a graph, encoding information about the neighborhood of a node as a real-valued vector. In each GCN layer, information flows along edges of the graph; in other words, each node receives messages from all its immediate neighbors. When multiple GCN layers are stacked, information about larger neighborhoods gets integrated. For example, in the second layer, a node will receive information from its immediate neighbors, but this information already includes information from their respective neighbors. By choosing the number of GCN layers, we regulate the distance the information travels: with k layers a node receives information from neighbors at most k hops away.

Formally, consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of n nodes, and $\mathcal{E}$ is a set of edges. Every node is assumed to be connected to itself, i.e. $\forall v \in \mathcal{V} : (v, v) \in \mathcal{E}$. Now, let $X \in \mathbb{R}^{d \times n}$ be a matrix containing all n nodes with their features, where d is the dimensionality of the feature vectors. In our case, X will contain word embeddings, but in general it can contain any kind of features. For a 1-layer GCN, the new node representations are computed as follows:

$$\mathbf{h}_v = \rho\left(\sum_{u \in \mathcal{N}(v)} W\mathbf{x}_u + \mathbf{b}\right)$$

where $W \in \mathbb{R}^{d \times d}$ is a weight matrix and $\mathbf{b} \in \mathbb{R}^d$ a bias vector.[1] ρ is an activation function, e.g. a ReLU. $\mathcal{N}(v)$ is the set of neighbors of v, which we assume here to always include v itself. As stated before, to allow information to flow over multiple hops, we need to stack GCN layers. The recursive computation is as follows:

$$\mathbf{h}_v^{(j+1)} = \rho\left(\sum_{u \in \mathcal{N}(v)} W^{(j)}\mathbf{h}_u^{(j)} + \mathbf{b}^{(j)}\right)$$

where j indexes the layer, and $\mathbf{h}_v^{(0)} = \mathbf{x}_v$.

[1] We dropped the normalization factor used by Kipf and Welling (2016), as it is not used in syntactic GCNs of Marcheggiani and Titov (2017).

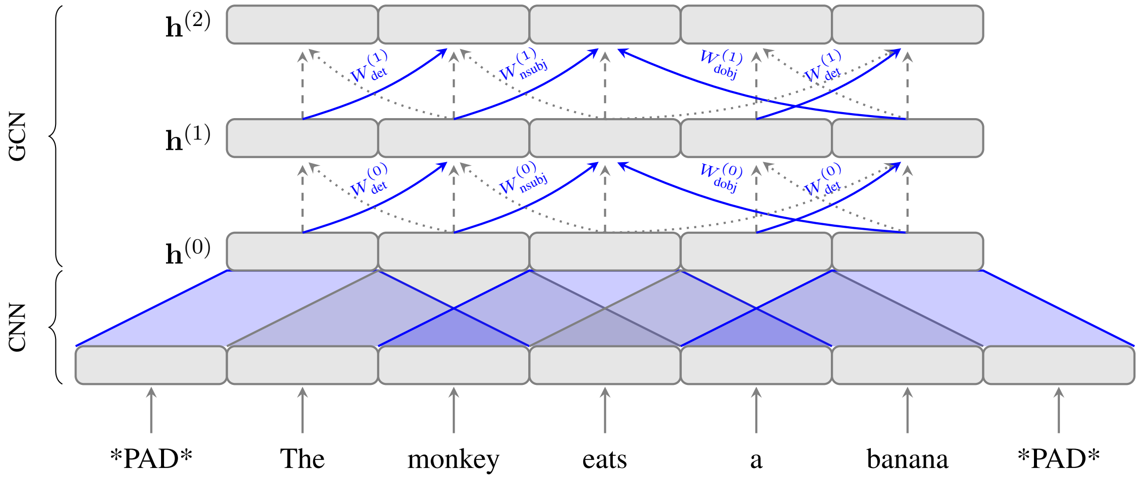


Figure 2: A 2-layer syntactic GCN on top of a convolutional encoder. Loop connections are depicted with dashed edges, syntactic ones with solid (dependents to heads) and dotted (heads to dependents) edges. Gates and some labels are omitted for clarity.

2.3 Syntactic GCNs

Marcheggiani and Titov (2017) generalize GCNs to operate on directed and labeled graphs.[2] This makes it possible to use linguistic structures such as dependency trees, where directionality and edge labels play an important role. They also integrate edge-wise gates which let the model regulate contributions of individual dependency edges. We will briefly describe these modifications.

Directionality. In order to deal with directionality of edges, separate weight matrices are used for incoming and outgoing edges. We follow the convention that in dependency trees heads point to their dependents, and thus *outgoing* edges are used for head-to-dependent connections, and *incoming* edges are used for dependent-to-head connections. Modifying the recursive computation for directionality, we arrive at:

$$\mathbf{h}_v^{(j+1)} = \rho\left(\sum_{u\in\mathcal{N}(v)} W^{(j)}_{\text{dir}(u,v)}\,\mathbf{h}_u^{(j)} + \mathbf{b}^{(j)}_{\text{dir}(u,v)}\right)$$

where $\text{dir}(u, v)$ selects the weight matrix associated with the directionality of the edge connecting u and v (i.e. W_{IN} for u-to-v, W_{OUT} for v-to-u, and W_{LOOP} for v-to-v). Note that self loops are modeled separately,

so there are now three times as many parameters as in a non-directional GCN.

Labels. Making the GCN sensitive to labels is straightforward given the above modifications for directionality. Instead of using separate matrices for each direction, separate matrices are now defined for each direction and label combination:

$$\mathbf{h}_v^{(j+1)} = \rho\left(\sum_{u\in\mathcal{N}(v)} W^{(j)}_{\text{lab}(u,v)}\,\mathbf{h}_u^{(j)} + \mathbf{b}^{(j)}_{\text{lab}(u,v)}\right)$$

where we incorporate the directionality of an edge directly in its label.

Importantly, to prevent over-parametrization, only bias terms are made label-specific, in other words: $W_{\text{lab}(u,v)} = W_{\text{dir}(u,v)}$. The resulting syntactic GCN is illustrated in Figure 2 (shown on top of a CNN, as we will explain in the subsequent section).

Edge-wise gating. Syntactic GCNs also include gates, which can down-weight the contribution of individual edges. They also allow the model to deal with noisy predicted structure, i.e. to ignore potentially erroneous syntactic edges. For each edge, a scalar gate is calculated as follows:

$$g^{(j)}_{u,v} = \sigma\left(\mathbf{h}_u^{(j)} \cdot \hat{\mathbf{w}}^{(j)}_{\text{dir}(u,v)} + \hat{b}^{(j)}_{\text{lab}(u,v)}\right)$$

where σ is the logistic sigmoid function, and $\hat{\mathbf{w}}^{(j)}_{\text{dir}(u,v)} \in \mathbb{R}^d$ and $\hat{b}^{(j)}_{\text{lab}(u,v)} \in \mathbb{R}$ are learned parameters for the gate. The computation becomes:

$$\mathbf{h}_v^{(j+1)}{=}\rho\Big(\sum_{u\in\mathcal{N}(v)} g^{(j)}_{u,v}\big(W^{(j)}_{\text{dir}(u,v)}\,\mathbf{h}_u^{(j)} + \mathbf{b}^{(j)}_{\text{lab}(u,v)}\big)\Big)$$

[2] For an alternative approach to integrating labels and directions, see applications of GCNs to statistical relation learning (Schlichtkrull et al., 2017).

3 Graph Convolutional Encoders

In this work we focus on exploiting structural information on the source side, i.e. in the encoder. We hypothesize that using an encoder that incorporates syntax will lead to more informative representations of words, and that these representations, when used as context vectors by the decoder, will lead to an improvement in translation quality. Consequently, in all our models, we use the decoder of Bahdanau et al. (2015) and keep this part of the model constant. As is now common practice, we do not use a maxout layer in the decoder, but apart from this we do not deviate from the original definition. In all models we make use of GRUs (Cho et al., 2014b) as our RNN units.

Our models vary in the encoder part, where we exploit the power of GCNs to induce syntactically-aware representations. We now define a series of encoders of increasing complexity.

BoW + GCN. In our first and simplest model, we propose a bag-of-words encoder (with position embeddings, see §2.1.1), with a GCN on top. In other words, inputs $\mathbf{h}^{(0)}$ are a sum of embeddings of a word and its position in a sentence. Since the original BoW encoder captures the linear ordering information only in a very crude way (through the position embeddings), the structural information provided by GCN should be highly beneficial.

Convolutional + GCN. In our second model, we use convolutional neural networks to learn word representations. CNNs are fast, but by definition only use a limited window of context. Instead of the approach used by Gehring et al. (2016) (i.e. stacking mulitple CNN layers on top of each other), we use a GCN to enrich the one-layer CNN representations. Figure 2 shows this model. Note that, while the figure shows a CNN with a window size of 3, we will use a larger window size of 5 in our experiments. We expect this model to perform better than BoW + GCN, because of the additional local context captured by the CNN.

BiRNN + GCN. In our third and most powerful model, we employ bidirectional recurrent neural networks. In this model, we start by encoding the source sentence using a BiRNN (i.e. BiGRU), and use the resulting hidden states as input to a GCN. Instead of relying on linear order only, the GCN will allow the encoder to 'teleport' over parts of the input sentence, along dependency edges, connecting words that otherwise might be far apart. The model might not only benefit from this teleporting capability however; also the nature of the relations between words (i.e. dependency relation types) may be useful, and the GCN exploits this information (see §2.3 for details).

This is the most challenging setup for GCNs, as RNNs have been shown capable of capturing at least some degree of syntactic information without explicit supervision (Linzen et al., 2016), and hence they should be hard to improve on by incorporating treebank syntax.

Marcheggiani and Titov (2017) did not observe improvements from using multiple GCN layers in semantic role labeling. However, we do expect that propagating information from further in the tree should be beneficial in principle. We hypothesize that the first layer is the most influential one, capturing most of the syntactic context, and that additional layers only modestly modify the representations. To ease optimization, we add a residual connection (He et al., 2016) between the GCN layers, when using more than one layer.

4 Experiments

Experiments are performed using the Neural Monkey toolkit[3] (Helcl and Libovický, 2017), which implements the model of Bahdanau et al. (2015) in TensorFlow. We use the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001 (0.0002 for CNN models).[4] The batch size is set to 80. Between layers we apply dropout with a probability of 0.2, and in experiments with GCNs[5] we use the same value for edge dropout. We train for 45 epochs, evaluating the BLEU performance of the model every epoch on the validation set. For testing, we select the model with the highest validation BLEU. L2 regularization is used with a value of 10^{-8}. All the model selection (incl. hyperparameter selections) was performed on the validation set. In all experiments we obtain translations using a greedy decoder, i.e. we select the output token with the highest probability at each time step.

We will describe an artificial experiment in §4.1 and MT experiments in §4.2.

[3] https://github.com/ufal/neuralmonkey

[4] Like Gehring et al. (2016) we note that Adam is too aggressive for CNN models, hence we use a lower learning rate.

[5] GCN code at https://github.com/bastings/neuralmonkey

4.1 Reordering artificial sequences

Our goal here is to provide an intuition for the capabilities of GCNs. We define a reordering task where randomly permuted sequences need to be put back into the original order. We encode the original order using edges, and test if GCNs can successfully exploit them. Note that this task is not meant to provide a fair comparison to RNNs. The input (besides the edges) simply does not carry any information about the original ordering, so RNNs cannot possibly solve this task.

Data. From a vocabulary of 26 types, we generate random sequences of 3-10 tokens. We then randomly permute them, pointing every token to its original predecessor with a label sampled from a set of 5 labels. Additionally, we point every token to an *arbitrary* position in the sequence with a label from a distinct set of 5 'fake' labels. We sample 25000 training and 1000 validation sequences.

Model. We use the BiRNN + GCN model, i.e. a bidirectional GRU with a 1-layer GCN on top. We use 32, 64 and 128 units for embeddings, GRU units and GCN layers, respectively.

Results. After 6 epochs of training, the model learns to put permuted sequences back into order, reaching a validation BLEU of 99.2. Figure 3 shows that the mean values of the bias terms of gates (i.e. $\hat{b}$) for real and fake edges are far apart, suggesting that the GCN learns to distinguish them. Interestingly, this illustrates why edge-wise gating is beneficial. A gate-less model would not understand which of the two outgoing arcs is fake and which is genuine, because only biases b would then be label-dependent. Consequently, it would only do a mediocre job in reordering. Although using label-specific matrices W would also help, this would not scale to the real scenario (see §2.3).

4.2 Machine Translation

Data. For our experiments we use the En-De and En-Cs News Commentary v11 data from the WMT16 translation task.[6] For En-De we also train on the full WMT16 data set. As our validation set and test set we use `newstest2015` and `newstest2016`, respectively.

Pre-processing. The English sides of the corpora are tokenized and parsed into dependency trees by SyntaxNet,[7] using the pre-trained Parsey McParseface model.[8] The Czech and German sides are tokenized using the Moses tokenizer.[9] Sentence pairs where either side is longer than 50 words are filtered out after tokenization.

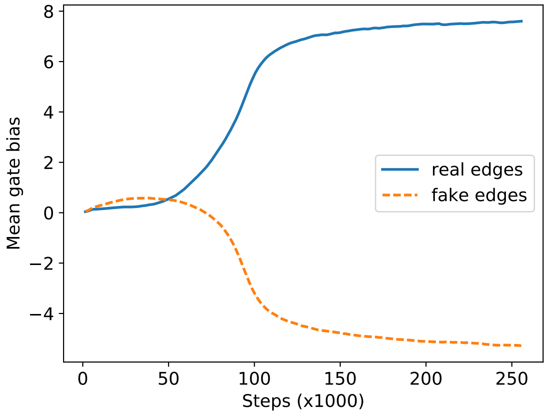


Figure 3: Mean values of gate bias terms for real (useful) labels and for fake (non useful) labels suggest the GCN learns to distinguish them.

Vocabularies. For the English sides, we construct vocabularies from all words except those with a training set frequency smaller than three. For Czech and German, to deal with rare words and phenomena such as inflection and compounding, we learn byte-pair encodings (BPE) as described by Sennrich et al. (2016b). Given the size of our data set, and following Wu et al. (2016), we use 8000 BPE merges to obtain robust frequencies for our subword units (16000 merges for full data experiment). Data set statistics are summarized in Table 1 and vocabulary sizes in Table 2.

	Train	Val.	Test
English-German	226822	2169	2999
English-German (full)	4500966	2169	2999
English-Czech	181112	2656	2999

Table 1: The number of sentences in our data sets.

Hyperparameters. We use 256 units for word embeddings, 512 units for GRUs (800 for En-De full data set experiment), and 512 units for convolutional layers (or equivalently, 512 'channels'). The dimensionality of the GCN layers is equiva-

[6] http://www.statmt.org/wmt16/translation-task.html

[7] https://github.com/tensorflow/models/tree/master/syntaxnet

[8] The used dependency parses can be reproduced by using the `syntaxnet/demo.sh` shell script.

[9] https://github.com/moses-smt/mosesdecoder

	Source	Target
English-German	37824	8099 (BPE)
English-German (full)	50000	16000 (BPE)
English-Czech	33786	8116 (BPE)

Table 2: Vocabulary sizes.

lent to the dimensionality of their input. We report results for 2-layer GCNs, as we find them most effective (see ablation studies below).

Baselines. We provide three baselines, each with a different encoder: a bag-of-words encoder, a convolutional encoder with window size $w = 5$, and a BiRNN. See §2.1.1 for details.

Evaluation. We report (cased) BLEU results (Papineni et al., 2002) using `multi-bleu`, as well as Kendall τ reordering scores.[10]

4.2.1 Results

English-German. Table 3 shows test results on English-German. Unsurprisingly, the bag-of-words baseline performs the worst. We expected the BoW+GCN model to make easy gains over this baseline, which is indeed what happens. The CNN baseline reaches a higher BLEU_4 score than the BoW models, but interestingly its BLEU_1 score is lower than the BoW+GCN model. The CNN+GCN model improves over the CNN baseline by +1.9 and +1.1 for BLEU_1 and BLEU_4, respectively. The BiRNN, the strongest baseline, reaches a BLEU_4 of 14.9. Interestingly, GCNs still manage to improve the result by +2.3 BLEU_1 and +1.2 BLEU_4 points. Finally, we observe a big jump in BLEU_4 by using the full data set and beam search (beam 12). The BiRNN now reaches 23.3, while adding a GCN achieves a score of 23.9.

	Kendall	BLEU_1	BLEU_4
BoW	0.3352	40.6	9.5
+ GCN	0.3520	44.9	12.2
CNN	0.3601	42.8	12.6
+ GCN	0.3777	44.7	13.7
BiRNN	0.3984	45.2	14.9
+ GCN	0.4089	47.5	16.1
BiRNN (full)	0.5440	53.0	23.3
+ GCN	0.5555	54.6	23.9

Table 3: Test results for English-German.

English-Czech. Table 4 shows test results on English-Czech. While it is difficult to obtain high absolute BLEU scores on this dataset, we can still see similar relative improvements. Again the BoW baseline scores worst, with the BoW+GCN easily beating that result. The CNN baseline scores BLEU_4 of 8.1, but the CNN+GCN improves on that, this time by +1.0 and +0.6 for BLEU_1 and BLEU_4, respectively. Interestingly, BLEU_1 scores for the BoW+GCN and CNN+GCN models are higher than both baselines so far. Finally, the BiRNN baseline scores a BLEU_4 of 8.9, but it is again beaten by the BiRNN+GCN model with +1.9 BLEU_1 and +0.7 BLEU_4.

	Kendall	BLEU_1	BLEU_4
BoW	0.2498	32.9	6.0
+ GCN	0.2561	35.4	7.5
CNN	0.2756	35.1	8.1
+ GCN	0.2850	36.1	8.7
BiRNN	0.2961	36.9	8.9
+ GCN	0.3046	38.8	9.6

Table 4: Test results for English-Czech.

Effect of GCN layers. How many GCN layers do we need? Every layer gives us an extra hop in the graph and expands the syntactic neighborhood of a word. Table 5 shows validation BLEU performance as a function of the number of GCN layers. For English-German, using a 1-layer GCN improves BLEU-1, but surprisingly has little effect on BLEU_4. Adding an additional layer gives improvements on both BLEU_1 and BLEU_4 of +1.3 and +0.73, respectively. For English-Czech, performance increases with each added GCN layer.

	En-De		En-Cs	
	BLEU_1	BLEU_4	BLEU_1	BLEU_4
BiRNN	44.2	14.1	37.8	8.9
+ GCN (1L)	45.0	14.1	38.3	9.6
+ GCN (2L)	46.3	14.8	39.6	9.9

Table 5: Validation BLEU for English-German and English-Czech for 1- and 2-layer GCNs.

[10] See Stanojević and Simaan (2015). TER (Snover et al., 2006) and BEER (Stanojević and Sima'an, 2014) metrics, even though omitted due to space considerations, are consistent with the reported results.

Effect of sentence length. We hypothesize that GCNs should be more beneficial for longer sentences: these are likely to contain long-distance syntactic dependencies which may not be adequately captured by RNNs but directly encoded in GCNs. To test this, we partition the validation data into five buckets and calculate BLEU for each of them. Figure 4 shows that GCN-based models outperform their respective baselines rather uniformly across all buckets. This is a surprising result. One explanation may be that syntactic parses are noisier for longer sentences, and this prevents us from obtaining extra improvements with GCNs.

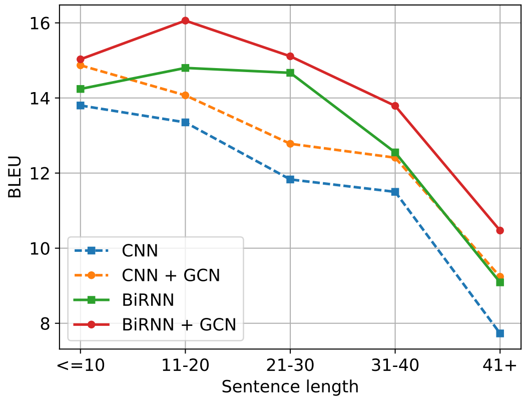


Figure 4: Validation BLEU per sentence length.

Discussion. Results suggest that the syntax-aware representations provided by GCNs consistently lead to improved translation performance as measured by BLEU$_4$ (as well as TER and BEER). Consistent gains in terms of Kendall tau and BLEU$_1$ indicate that improvements correlate with better word order and lexical/BPE selection, two phenomena that depend crucially on syntax.

5 Related Work

We review various accounts to syntax in NMT as well as other convolutional encoders.

Syntactic features and/or constraints. Sennrich and Haddow (2016) embed features such as POS-tags, lemmas and dependency labels and feed these into the network along with word embeddings. Eriguchi et al. (2016) parse English sentences with an HPSG parser and use a Tree-LSTM to encode the internal nodes of the tree. In the decoder, word and node representations compete under the same attention mechanism. Stahlberg et al. (2016) use a pruned lattice from a hierarchical phrase-based model (hiero) to constrain NMT. Hiero trees are not syntactically-aware, but instead constrained by symmetrized word alignments. Aharoni and Goldberg (2017) propose neural string-to-tree by predicting linearized parse trees.

Multi-task Learning. Sharing NMT parameters with a syntactic parser is a popular approach to obtaining syntactically-aware representations. Luong et al. (2015a) predict linearized constituency parses as an additional task. Eriguchi et al. (2017) multi-task with a target-side RNNG parser (Dyer et al., 2016) and improve on various language pairs with English on the target side. Nadejde et al. (2017) multi-task with CCG tagging, and also integrate syntax on the target side by predicting a sequence of words interleaved with CCG supertags.

Latent structure. Hashimoto and Tsuruoka (2017) add a syntax-inspired encoder on top of a BiLSTM layer. They encode source words as a learned average of potential parents emulating a relaxed dependency tree. While their model is trained purely on translation data, they also experiment with pre-training the encoder using treebank annotation and report modest improvements on English-Japanese. Yogatama et al. (2016) introduce a model for language understanding and generation that composes words into sentences by inducing unlabeled binary bracketing trees.

Convolutional encoders. Gehring et al. (2016) show that CNNs can be competitive to BiRNNs when used as encoders. To increase the receptive field of a word's context they stack multiple CNN layers. Kalchbrenner et al. (2016) use convolution in both the encoder and the decoder; they make use of dilation to increase the receptive field. In contrast to both approaches, we use a GCN informed by dependency structure to increase it. Finally, Cho et al. (2014a) propose a recursive convolutional neural network which builds a tree out of the word leaf nodes, but which ends up compressing the source sentence in a single vector.

6 Conclusions

We have presented a simple and effective approach to integrating syntax into neural machine translation models and have shown consistent BLEU$_4$ improvements for two challenging language pairs: English-German and English-Czech. Since GCNs are capable of encoding any kind of graph-based structure, in future work we would like to go beyond syntax, by using semantic annotations such as SRL and AMR, and co-reference chains.

Acknowledgments

We would like to thank Michael Schlichtkrull and Thomas Kipf for their suggestions and comments. This work was supported by the European Research Council (ERC StG BroadSem 678254) and the Dutch National Science Foundation (NWO VIDI 639.022.518, NWO VICI 277-89-002).

References

Roee Aharoni and Yoav Goldberg. 2017. Towards String-to-Tree Neural Machine Translation. *ArXiv e-prints*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2012. Abstract meaning representation (amr) 1.0 specification. In *Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452, Uppsala, Sweden. Association for Computational Linguistics.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, volume abs/1409.1259, pages 103–111.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3837–3845.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833, Berlin, Germany. Association for Computational Linguistics.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to Parse and Translate Improves Neural Machine Translation. *ArXiv e-prints*.

Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. 2016. A convolutional encoder model for neural machine translation. *CoRR*, abs/1611.02344.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. *ArXiv e-prints*.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2017. Neural machine translation with source-side latent graph parsing. *CoRR*, abs/1702.02265.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Jindřich Helcl and Jindřich Libovický. 2017. Neural monkey: An open-source tool for sequence learning. *The Prague Bulletin of Mathematical Linguistics*, (107):5–17.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Ozan Irsoy and Claire Cardie. 2014. Opinion Mining with Deep Recurrent Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 720–728, Doha, Qatar. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *CoRR*, abs/1610.10099.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. 2016. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task Sequence to Sequence Learning. *CoRR*, abs/1511.06114.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015b. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2017. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark. Association for Computational Linguistics.

Maria Nadejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch. 2017. Syntax-aware Neural Machine Translation Using CCG. *ArXiv e-prints*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. pages 311–318.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling Relational Data with Graph Convolutional Networks. *ArXiv e-prints*.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Rico Sennrich and Barry Haddow. 2016. Linguistic Input Features Improve Neural Machine Translation. In *Proceedings of the First Conference on Machine Translation (WMT16)*, volume abs/1606.02892.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for wmt 16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

David Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings on the Workshop on Statistical Machine Translation*, pages 23–30, New York City. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

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

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016. Syntactically guided neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 299–305, Berlin, Germany. Association for Computational Linguistics.

Miloš Stanojević and Khalil Simaan. 2015. Evaluating mt systems with beer. *The Prague Bulletin of Mathematical Linguistics*, 104(1):17–26.

Miloš Stanojević and Khalil Sima'an. 2014. Fitting sentence level translation evaluation with many dense features. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 202–206, Doha, Qatar. Association for Computational Linguistics.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Dani Yogatama, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Wang Ling. 2016. Learning to compose words into sentences with reinforcement learning. *CoRR*, abs/1611.09100.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, StatMT '06, pages 138–141, Stroudsburg, PA, USA. Association for Computational Linguistics.

Trainable Greedy Decoding for Neural Machine Translation

Jiatao Gu[†], **Kyunghyun Cho**[‡] **and Victor O.K. Li**[†]

[†]The University of Hong Kong
[‡]New York University
[†]{jiataogu, vli}@eee.hku.hk
[‡]kyunghyun.cho@nyu.edu

Abstract

Recent research in neural machine translation has largely focused on two aspects; neural network architectures and end-to-end learning algorithms. The problem of decoding, however, has received relatively little attention from the research community. In this paper, we solely focus on the problem of decoding given a trained neural machine translation model. Instead of trying to build a new decoding algorithm for any specific decoding objective, we propose the idea of *trainable decoding algorithm* in which we train a decoding algorithm to find a translation that maximizes an arbitrary decoding objective. More specifically, we design an actor that observes and manipulates the hidden state of the neural machine translation decoder and propose to train it using a variant of deterministic policy gradient. We extensively evaluate the proposed algorithm using four language pairs and two decoding objectives, and show that we can indeed train a trainable greedy decoder that generates a better translation (in terms of a target decoding objective) with minimal computational overhead.

1 Introduction

Neural machine translation has recently become a method of choice in machine translation research. Besides its success in traditional settings of machine translation, that is one-to-one translation between two languages, (Sennrich et al., 2016; Chung et al., 2016), neural machine translation has ventured into more sophisticated settings of machine translation. For instance, neural machine translation has successfully proven itself to be capable of handling subword-level representation of sentences (Lee et al., 2016; Luong and Manning, 2016; Sennrich et al., 2015; Costa-Jussa and Fonollosa, 2016; Ling et al., 2015). Furthermore, several research groups have shown its potential in seamlessly handling multiple languages (Dong et al., 2015; Luong et al., 2015a; Firat et al., 2016a,b; Lee et al., 2016; Ha et al., 2016; Viégas et al., 2016).

A typical scenario of neural machine translation starts with training a model to maximize its log-likelihood. That is, we often train a model to maximize the conditional probability of a reference translation given a source sentence over a large parallel corpus. Once the model is trained in this way, it defines the conditional distribution over all possible translations given a source sentence, and the task of translation becomes equivalent to finding a translation to which the model assigns the highest conditional probability. Since it is computationally intractable to do so exactly, it is a usual practice to resort to approximate search/decoding algorithms such as greedy decoding or beam search. In this scenario, we have identified two points where improvements could be made. They are (1) training (including the selection of a model architecture) and (2) decoding.

Much of the research on neural machine translation has focused solely on the former, that is, on improving the model architecture. Neural machine translation started with with a simple encoder-decoder architecture in which a source sentence is encoded into a single, fixed-size vector (Cho et al., 2014; Sutskever et al., 2014; Kalchbrenner and Blunsom, 2013). It soon evolved with the attention mechanism (Bahdanau et al., 2014). A few variants of the attention mechanism, or its regularization, have been proposed recently to improve both the translation quality as well as the computational efficiency (Luong et al., 2015b; Cohn et al., 2016; Tu et al., 2016b). More recently, convolutional net-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1968–1978
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

works have been adopted either as a replacement of or a complement to a recurrent network in order to efficiently utilize parallel computing (Kalchbrenner et al., 2016; Lee et al., 2016; Gehring et al., 2016).

On the aspect of decoding, only a few research groups have tackled this problem by incorporating a target decoding algorithm into training. Wiseman and Rush (2016) and Shen et al. (2015) proposed a learning algorithm tailored for beam search. Ranzato et al. (2015) and (Bahdanau et al., 2016) suggested to use a reinforcement learning algorithm by viewing a neural machine translation model as a policy function. Investigation on decoding alone has, however, been limited. Cho (2016) showed the limitation of greedy decoding by simply injecting unstructured noise into the hidden state of the neural machine translation system. Tu et al. (2016a) similarly showed that the exactness of beam search does not correlate well with actual translation quality, and proposed to augment the learning cost function with reconstruction to alleviate this problem. Li et al. (2016) proposed a modification to the existing beam search algorithm to improve its exploration of the translation space.

In this paper, we tackle the problem of decoding in neural machine translation by introducing a concept of *trainable greedy decoding*. Instead of manually designing a new decoding algorithm suitable for neural machine translation, we propose to learn a decoding algorithm with an arbitrary decoding objective. More specifically, we introduce a neural-network-based decoding algorithm that works on an already-trained neural machine translation system by observing and manipulating its hidden state. We treat such a neural network as an agent with a deterministic, continuous action and train it with a variant of the deterministic policy gradient algorithm (Silver et al., 2014).

We extensively evaluate the proposed trainable greedy decoding on four language pairs (En-Cs, En-De, En-Ru and En-Fi; in both directions) with two different decoding objectives; sentence-level BLEU and negative perplexity. By training such trainable greedy decoding using deterministic policy gradient with the proposed critic-aware actor learning, we observe that we can improve decoding performance with minimal computational overhead. Furthermore, the trained actors are found to improve beam search as well, suggesting a future research direction in extending the proposed idea of trainable decoding for more sophisticated underlying decoding algorithms.

2 Background

2.1 Neural Machine Translation

Neural machine translation is a special case of conditional recurrent language modeling, where the source and target are natural language sentences. Let us use $X = \{x_1, \ldots, x_{T_s}\}$ and $Y = \{y_1, \ldots, y_T\}$ to denote source and target sentences, respectively. Neural machine translation then models the target sentence given the source sentence as: $p(Y|X) = \prod_{t=1}^{T} p(y_t|y_{<t}, X)$. Each term on the r.h.s. of the equation above is modelled as a composite of two parametric functions:

$$p(y_t|y_{<t}, X) \propto \exp\left(g\left(y_t, z_t; \theta_g\right)\right),$$

where $z_t = f(z_{t-1}, y_{t-1}, e_t(X; \theta_e); \theta_f)$. g is a read-out function that transforms the hidden state z_t into the distribution over all possible symbols, and f is a recurrent function that compresses all the previous target words $y_{<t}$ and the time-dependent representation $e_t(X; \theta_e)$ of the source sentence X. This time-dependent representation e_t is often implemented as a recurrent network encoder of the source sentence coupled with an attention mechanism (Bahdanau et al., 2014).

Maximum Likelihood Learning We train a neural machine translation model, or equivalently estimate the parameters θ_g, θ_f and θ_e, by maximizing the log-probability of a reference translation $\hat{Y} = \{\hat{y}_1, ..., \hat{y}_T\}$ given a source sentence. That is, we maximize the log-likelihood function:

$$J^{\mathrm{ML}}(\theta_g, \theta_f, \theta_e) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p_\theta(\hat{y}_t^n | \hat{y}_{<t}^n, X^n),$$

given a training set consisting of N source-target sentence pairs. It is important to note that this maximum likelihood learning does not take into account how a trained model would be used. Rather, it is only concerned with learning a distribution over all possible translations.

2.2 Decoding

Once the model is trained, either by maximum likelihood learning or by any other recently proposed algorithms (Wiseman and Rush, 2016; Shen et al., 2015; Bahdanau et al., 2016; Ranzato et al., 2015), we can let the model translate a given sentence by finding a translation that maximizes

$$\hat{Y} = \arg\max_{Y} \log p_\theta(Y|X),$$

where $\theta = (\theta_g, \theta_f, \theta_e)$. This is, however, computationally intractable, and it is a usual practice to resort to approximate decoding algorithms.

Greedy Decoding One such approximate decoding algorithm is greedy decoding. In greedy decoding, we follow the conditional dependency path and pick the symbol with the highest conditional probability so far at each node. This is equivalent to picking the best symbol one at a time from left to right in conditional language modelling. A decoded translation of greedy decoding is $\hat{Y} = (\hat{y}_1, \dots, \hat{y}_T)$, where

$$\hat{y}_t = \arg\max_{y \in V} \log p_\theta(y|\hat{y}_{<t}, X). \quad (1)$$

Despite its preferable computational complexity $O(|V| \times T)$, greedy decoding has been over time found to be undesirably sub-optimal.

Beam Search Beam search keeps $K > 1$ hypotheses, unlike greedy decoding which keeps only a single hypothesis during decoding. At each time step t, beam search picks K hypotheses with the highest scores ($\prod_{t'=1}^{t} p(y_t|y_{<t}, X)$). When all the hypotheses terminate (outputting the end-of-the-sentence symbol), it returns the hypothesis with the highest log-probability. Despite its superior performance compared to greedy decoding, the computational complexity grows linearly w.r.t. the size of beam K, which makes it less preferable especially in the production environment.

3 Trainable Greedy Decoding

3.1 Many Decoding Objectives

Although we have described decoding in neural machine translation as a maximum-a-posteriori estimation in $\log p(Y|X)$, this is not necessarily the only nor the desirable decoding objective.

First, each potential scenario in which neural machine translation is used calls for a unique decoding objective. In simultaneous translation/interpretation, which has recently been studied in the context of neural machine translation (Gu et al., 2016), the decoding objective is formulated as a trade-off between the translation quality and delay. On the other hand, when a machine translation system is used as a part of a larger information extraction system, it is more important to correctly translate named entities and events than to translate syntactic function words. The decoding objective in this case must account for how the translation is used in subsequent modules in a larger system.

Second, the conditional probability assigned by a trained neural machine translation model does not necessarily reflect our perception of translation quality. Although Cho (2016) provided empirical evidence of high correlation between the log-probability and BLEU, a *de facto* standard metric in machine translation, there have also been reports on large mismatch between the log-probability and BLEU. For instance, Tu et al. (2016a) showed that beam search with a very large beam, which is supposed to find translations with better log-probabilities, suffers from pathological translations of very short length, resulting in low translation quality. This calls for a way to design or *learn* a decoding algorithm with an objective that is more directly correlated to translation quality.

In short, there is a significant need for designing multiple decoding algorithms for neural machine translation, regardless of how it was trained. It is however non-trivial to manually design a new decoding algorithm with an arbitrary objective. This is especially true with neural machine translation, as the underlying structure of the decoding/search process – the high-dimensional hidden state of a recurrent network – is accessible but not interpretable. Instead, in the remainder of this section, we propose our approach of *trainable greedy decoding*.

3.2 Trainable Greedy Decoding

We start from the noisy, parallel approximate decoding (NPAD) algorithm proposed in (Cho, 2016). The main idea behind NPAD algorithm is that a better translation with a higher log-probability may be found by injecting unstructured noise in the transition function of a recurrent network. That is,

$$z_t = f(z_{t-1} + \epsilon_t, y_{t-1}, e_t(X; \theta_e); \theta_f),$$

where $\epsilon_t \sim \mathcal{N}(0, (\sigma_0/t)^2)$. NPAD avoids potential degradation of translation quality by running such a noisy greedy decoding process multiple times in parallel. An important lesson of NPAD algorithm is that there exists a decoding strategy with the asymptotically same computational complexity that results in a better translation quality, and that such a better translation can be found by manipulating the hidden state of the recurrent network.

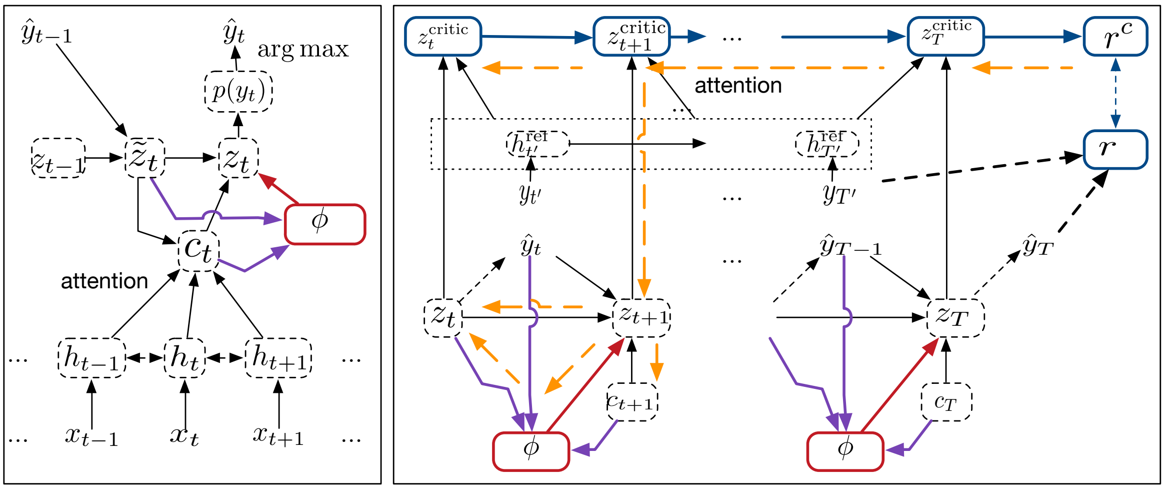


Figure 1: Graphical illustrations of the trainable greedy decoding. The left panel shows a single step of the actor interacting with the underlying neural translation model, and The right panel the interaction among the underlying neural translation system (dashed-border boxes), actor (red-border boxes), and critic (blue-border boxes). The solid arrows indicate the forward pass, and the dashed yellow arrows the actor's backward pass. The dotted-border box shows the use of a reference translation.

In this work, we propose to significantly extend NPAD by replacing the unstructured noise ϵ_t with a parametric function approximator, or an agent, π_ϕ. This agent takes as input the previous hidden state z_{t-1}, previously decoded word $\hat{y}_{t-1}$ and the time-dependent context vector $e_t(X; \theta_e)$ and outputs a real-valued vectorial action $a_t \in \mathbb{R}^{\dim(z_t)}$. Such an agent is trained such that greedy decoding with the agent finds a translation that maximizes any predefined, arbitrary decoding objective, while the underlying neural machine translation model is pretrained and fixed. Once the agent is trained, we generate a translation given a source sentence by greedy decoding however augmented with this agent. We call this decoding strategy *trainable greedy decoding*.

Related Work: Soothsayer prediction function Independently from and concurrently with our work here, Li et al. (2017) proposed, just two weeks earlier, to train a neural network that predicts an arbitrary decoding objective given a source sentence and a partial hypothesis, or a prefix of translation, and to use it as an auxiliary score in beam search. For training such a network, referred to as a Q network in their paper, they generate each training example by either running beam search or using a ground-truth translation (when appropriate) for each source sentence. This approach allows one to use an arbitrary decoding objective, but it still relies heavily on the log-probability of the underlying neural translation system in actual decoding. We expect a combination of these and our approaches may further improve decoding for neural machine translation in the future.

3.3 Learning and Challenges

While all the parameters—θ_g, θ_f and θ_e— of the underlying neural translation model are fixed, we only update the parameters ϕ of the agent π. This ensures the generality of the pretrained translation model, and allows us to train multiple trainable greedy decoding agents with different decoding objectives, maximizing the utility of a single trained translation model.

Let us denote by R our arbitrary decoding objective as a function that scores a translation generated from trainable greedy decoding. Then, our learning objective for trainable greedy decoding is

$$J^{\mathrm{A}}(\phi) = \mathbb{E}_{X \sim D}^{\hat{Y} = G_\pi(X)} \left[R(\hat{Y}) \right],$$

where we used $G_\pi(X)$ as a shorthand for trainable greedy decoding with an agent π.

There are two major challenges in learning an agent with such an objective. First, the decoding objective R may not be differentiable with respect to the agent. Especially because our goal is to accommodate an arbitrary decoding objective, this becomes a problem. For instance, BLEU, a standard quality metric in machine translation, is a piecewise linear function with zero derivatives almost everywhere. Second, the agent here is a real-valued, deterministic policy with a very high-dimensional action space (1000s of dimensions), which is well known to be difficult. In order to alleviate these difficulties, we propose to use a variant of the deterministic policy gradient algorithm (Silver et al., 2014; Lillicrap et al., 2015).

4 Deterministic Policy Gradient with Critic-Aware Actor Learning

4.1 Deterministic Policy Gradient for Trainable Greedy Decoding

It is highly unlikely for us to have access to the gradient of an arbitrary decoding objective R with respect to the agent π, or its parameters ϕ. Furthermore, we cannot estimate it stochastically because our policy π is defined to be deterministic without a predefined nor learned distribution over the action. Instead, following (Silver et al., 2014; Lillicrap et al., 2015), we use a parametric, differentiable approximator, called a critic R^c, for the non-differentiable objective R. We train the critic by minimizing

$$J^{\mathrm{C}}(\psi) = \mathbb{E}_{X \sim D}^{\hat{Y}=G_\pi(X)} \left[R^c_\psi(z_{1:T}) - R(\hat{Y}) \right]^2 .$$

The critic observes the state-action sequence of the agent π via the modified hidden states $(z_1, \ldots, z_T)$ of the recurrent network, and predicts the associated decoding objective. By minimizing the mean squared error above, we effectively encourage the critic to approximate the non-differentiable objective as closely as possible in the vicinity of the state-action sequence visited by the agent.

We implement the critic R^c as a recurrent network, similarly to the underlying neural machine translation system. This implies that we can compute the derivative of the predicted decoding objective with respect to the input, that is, the state-action sequence $z_{1:T}$, which allows us to update the actor π, or equivalently its parameters ϕ, to maximize the predicted decoding objective. Effectively we avoid the issue of non-differentiability of the original decoding objective by working with its proxy.

With the critic, the learning objective of the actor is now to maximize not the original decoding objective R but its proxy R^{C} such that

$$\hat{J}^{\mathrm{A}}(\phi) = \mathbb{E}_{X \sim D}^{\hat{Y}=G_\pi(X)} \left[R^{\mathrm{C}}(\hat{Y}) \right] .$$

Algorithm 1 Trainable Greedy Decoding

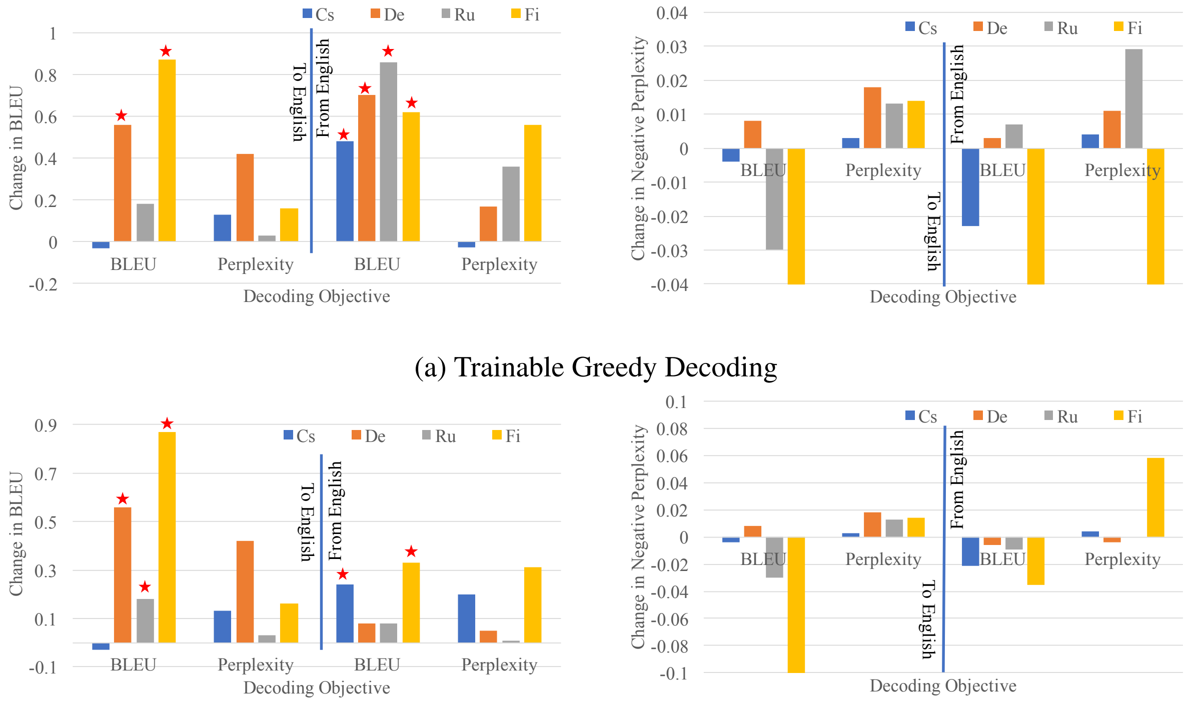
```
Require: NMT $\theta$, actor $\phi$, critic $\psi$, $N_c$, $N_a$, $S_c$, $S_a$, $\tau$
Train $\theta$ using MLE on training set $D$;
Initialize $\phi$ and $\psi$;
Shuffle $D$ twice into $D_\phi$ and $D_\psi$
while stopping criterion is not met do
    for $t = 1 : N_c$ do
        Draw a translation pair: $(X, Y) \sim D_\psi$;
        $r, r^c = $ DECODE$(S_c, X, Y, 1)$
        Update $\psi$ using $\nabla_\psi \sum_k (r^c_k - r_k)^2 / (S_c + 1)$
    for $t = 1 : N_a$ do
        Draw a translation pair: $(X, Y) \sim D_\phi$;
        $r, r^c = $ DECODE$(S_a, X, Y, 0)$
        Compute $w_k = \exp\left(-(r^c_k - r_k)^2 / \tau\right)$
        Compute $\tilde{w}_k = w_k / \sum_k w_k$
        Update $\phi$ using $-\sum_k (\tilde{w}_k \cdot \nabla_\phi r^c_k)$

Function: DECODE$(S, X, Y, c)$
$Y_s = \{\}, Z_s = \{\}, r = \{\}, r^c = \{\}$;
for $k = 1 : S$ do
    Sample noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ for each action;
    Greedy decoding $\hat{Y}^k = G_{\theta,\phi}(X)$ with $\epsilon$;
    Collect hidden states $z^k_{1:T}$ given $X$, $\hat{Y}$, $\theta$, $\phi$
    $Y_s \leftarrow Y_s \cup \{Y^k\}$
    $Z_s \leftarrow Z_s \cup \{z^k_{1:T}\}$
if $c = 1$ then
    Collect hidden states $z_{1:T}$ given $X$, $Y$, $\theta$
    $Y_s \leftarrow Y_s \cup \{Y\}$
    $Z_s \leftarrow Z_s \cup \{z_{1:T}\}$
for $\hat{Y}, Z \in Y_s, Z_s$ do
    Compute the critic output $r^c \leftarrow R^c_\psi(Z, \hat{Y})$
    Compute true reward $r \leftarrow R(Y, \hat{Y})$
return $r, r^c$
```

Unlike the original objective, this objective function is fully differentiable with respect to the agent π. We thus use a usual stochastic gradient descent algorithm to train the agent, while simultaneously training the critic. We do so by alternating between training the actor and critic. Note that we maximize the return of a full episode rather than the Q value, unlike usual approaches in reinforcement learning.

4.2 Critic-Aware Actor Learning

Challenges The most apparent challenge for training such a deterministic actor with a large action space is that most of action configurations will lead to zero return. It is also not trivial to devise an efficient exploration strategy with a deterministic actor with real-valued actions. This issue has however turned out to be less of a problem than in a usual reinforcement learning setting, as the state and action spaces are well structured thanks to pretraining by maximum likelihood learning. As observed by Cho (2016), any reasonable perturbation to the hidden state of the recurrent network generates a reasonable translation which would re-



(a) Trainable Greedy Decoding

(b) Beam Search + Trainable Greedy Decoding

Figure 2: The plots draw the improvements by the trainable greedy decoding on the test set. The x-axes correspond to the objectives used to train trainable greedy decoding, and the y-axes to the changes in the achieved objectives (BLEU for the figures on the left, and negative perplexity on the right.) The top row (a) shows the cases when the trainable greedy decoder is used on its own, and the bottom row (b) when it is used together with beam search. When training and evaluation are both done with BLEU, we test the statistical significance (Koehn, 2004), and we mark significant cases with red stars ($p < 0.05$.) The underlying neural machine translation models achieved the BLEU scores of 14.49/16.20 for En-Cs, 18.90/21.20 for Cs-En, 18.97/21.33 for En-De, 21.63/24.46 for De-En, 16.97/19.68 for En-Ru, 21.06/23.34 for Ru-En, 7.53/8.82 for En-Fi and 9.79/11.03 for Fi-En (greedy/beam).

ceive again a reasonable return.

Although this property of dense reward makes the problem of trainable greedy decoding more manageable, we have observed other issues during our preliminary experiment with the vanilla deterministic policy gradient. In order to avoid these issues that caused instability, we propose the following modifications to the vanilla algorithm.

Critic-Aware Actor Learning A major goal of the critic is not to estimate the return of a given episode, but to estimate the gradient of the return evaluated given an episode. In order to do so, the critic must be trained, or presented, with state-action sequences $z_{1:T'}$ similar though not identical to the state-action sequence generated by the current actor π. This is achieved, in our case, by injecting unstructured noise to the action at each time step, similar to (Heess et al., 2015):

$$\tilde{a}_t = \phi(z_t, a_{t-1}) + \sigma \cdot \epsilon, \tag{2}$$

where ϵ is a zero-mean, unit-variance normal variable. This noise injection procedure is mainly used when training the critic.

We have however observed that the quality of the reward and its gradient estimate of the critic is very noisy even when the critic was trained with this kind of noisy actor. This imperfection of the critic often led to the instability in training the actor in our preliminary experiments. In order to avoid this, we describe here a technique which we refer to as *critic-aware actor gradient estimation*.

Instead of using the point estimate $\frac{\partial R^c}{\partial \phi}$ of the gradient of the predicted objective with respect to the actor's parameters ϕ, we propose to use the expected gradient of the predicted objective with respect to the critic-aware distribution Q. That is,

$$\mathbb{E}_Q\left[\frac{\partial R^c_\psi}{\partial \phi}\right], \tag{3}$$

where we define the critic-aware distribution Q as

$$Q(\epsilon) \propto \exp(\underbrace{-(R^c_\psi - R)^2/\tau}_{\text{Critic-awareness}}) \exp(\underbrace{-\frac{\epsilon^2}{2\sigma^2}}_{\text{Locality}}). \tag{4}$$

This expectation allows us to incorporate the noisy, non-uniform nature of the critic's approximation of the objective by up-weighting the gradient computed at a point with a higher critic quality and down-weighting the gradient computed at a point with a lower critic quality. The first term in Q reflects this, while the second term ensures that our estimation is based on a small region around the state-action sequence generated by the current, noise-free actor π.

Since it is intractable to compute Eq. (3) exactly, we resort to importance sampling with the proposed distribution equal to the second term in Eq. (4). Then, our gradient estimate for the actor becomes the sum of the gradients from multiple realizations of the noisy actor in Eq. (2), where each gradient is weighted by the quality of the critic $\exp(-(R^c_\phi - R)^2/\tau)$. τ is a hyperparameter that controls the smoothness of the weights. We observed in our preliminary experiment that the use of this critic-aware actor learning significantly stabilizes general learning of both the actor and critic.

Reference Translations for Training the Critic In our setting of neural machine translation, we have access to a reference translation for each source sentence X, unlike in a usual setting of reinforcement learning. By force-feeding the reference translation into the underlying neural machine translation system (rather than feeding the decoded symbols), we can generate the reference state-action sequence. This sequence is much less correlated with those sequences generated by the actor, and facilitates computing a better estimate of the gradient w.r.t. the critic.

In Alg. 1, we present the complete algorithm. To make the description less cluttered, we only show the version of minibatch size = 1 which can be naturally extended. We also illustrate the proposed trainable greedy decoding and the proposed learning strategy in Fig. 1.

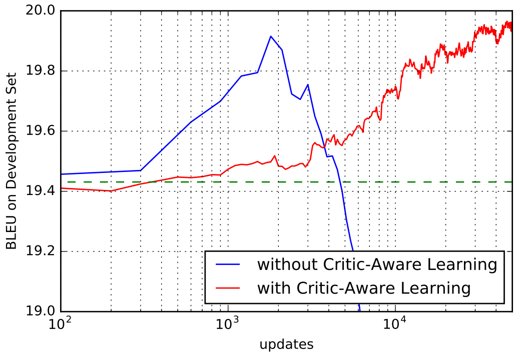


Figure 3: Comparison of greedy BLEU scores whether using the critic-aware exploration or not on Ru-En Dataset. The green line means the BLEU score achieved by greedy decoding from the underlying NMT model.

5 Experimental Settings

We empirically evaluate the proposed trainable greedy decoding on four language pairs – En-De, En-Ru, En-Cs and En-Fi – using a standard attention-based neural machine translation system (Bahdanau et al., 2014). We train underlying neural translation systems using the parallel corpora made available from WMT'15.[1] The same set of corpora are used for trainable greedy decoding as well. All the corpora are tokenized and segmented into subword symbols using byte-pair encoding (BPE) (Sennrich et al., 2015). We use sentences of length up to 50 subword symbols for MLE training and 200 symbols for trainable decoding. For validation and testing, we use newstest-2013 and newstest-2015, respectively.

5.1 Model Architectures and Learning

Underlying NMT Model For each language pair, we implement an attention-based neural machine translation model whose encoder and decoder recurrent networks have 1,028 gated recurrent units (GRU, Cho et al., 2014) each. Source and target symbols are projected into 512-dimensional embedding vectors. We trained each model for approximately 1.5 weeks using Adadelta (Zeiler, 2012).

Actor π We use a feedforward network with a single hidden layer as the actor. The input is a 2,056-dimensional vector which is the concatenation of the decoder hidden state and the time-dependent context vector from the attention mech-

[1] http://www.statmt.org/wmt15/

```
(a) S:  Главное зеркало инфракрасного космического телескопа имеет диаметр 6,5 метров
    T:  The primary mirror of the infrared space telescope has a diameter of 6.5 metres .
    G:  The main mirror of the infrared spaceboard has a diameter 6.5 m .
    A:  The main mirror of the infrared space-type telescope has a diameter of 6.5 meters .

(b) S:  Еще один пункт - это дать им понять , что они должны вести себя онлайн так же , как делают это оффлайн .
    T:  Another point is to make them see that they must behave online as they do offline .
    G:  Another option is to give them a chance to behave online as well as do this offline .
    A:  Another option is to give them to know that they must behave online as well as offline .

(c) S:  Возможен ли долговременный мир между арабами и израильтянами на Ближнем Востоке ?
    T:  Can there ever be a lasting peace between Arabs and Jews in the Middle East ?
    G:  Can the Long-term Peace be Out of the Middle East ?
    A:  Can the Long-term Peace be between Arabs and Israelis in the Middle East ?
```

Figure 4: Three Ru-En examples in which the difference between the trainable greedy decoding (A) and the conventional greedy decoding (G) is large. Each step is marked with magenta, when the actor significantly influenced the output distribution.

anism, and it outputs a 1,028-dimensional action vector for the decoder. We use 32 units for the hidden layer with $\tanh$ activations.

Critic R^c The critic is implemented as a variant of an attention-based neural machine translation model that takes a reference translation as a source sentence and a state-action sequence from the actor as a target sentence. Both the size of GRU units and embedding vectors are the same with the underlying model. Unlike a usual neural machine translation system, the critic does not language-model the target sentence but simply outputs a scalar value to predict the true return. When we predict a bounded return, such as sentence BLEU, we use a sigmoid activation at the output. For other unbounded return like perplexity, we use a linear activation.

Learning We train the actor and critic simultaneously by alternating between updating the actor and critic. As the quality of the critic's approximation of the decoding objective has direct influence on the actor's learning, we make ten updates to the critic before each time we update the actor once. We use RMSProp (Tieleman and Hinton, 2012) with the initial learning rates of 2×10^{-6} and 2×10^{-4}, respectively, for the actor and critic.

We monitor the progress of learning by measuring the decoding objective on the validation set. After training, we pick the actor that results in the best decoding objective on the validation set, and test it on the test set.

Decoding Objectives For each neural machine translation model, pretrained using maximum likelihood criterion, we train two trainable greedy decoding actors. One actor is trained to maximize BLEU (or its smoothed version for sentence-level scoring (Lin and Och, 2004)) as its decoding objective, and the other to minimize perplexity (or equivalently the negative log-probability normalized by the length.)

We have chosen the first two decoding objectives for two purposes. First, we demonstrate that it is possible to build multiple trainable decoders with a single underlying model trained using maximum likelihood learning. Second, the comparison between these two objectives provides a glimpse into the relationship between BLEU (the most widely used automatic metric for evaluating translation systems) and log-likelihood (the most widely used learning criterion for neural machine translation).

Evaluation We test the trainable greedy decoder with both greedy decoding and beam search. Although our decoder is always trained with greedy decoding, beam search in practice can be used together with the actor of the trainable greedy decoder. Beam search is expected to work better especially when our training of the trainable greedy decoder is unlikely to be optimal. In both cases, we report both the perplexity and BLEU.

5.2 Results and Analysis

We present the improvements of BLEU and perplexity (or its negation) in Fig. 2 for all the language pair-directions. It is clear from these plots that the best result is achieved when the trainable greedy decoder was trained to maximize the target decoding objective. When the decoder was trained to maximize sentence-level BLEU, we see the improvement in BLEU but often the degradation in the perplexity (see the left plots in Fig. 2.) On the other hand, when the actor was trained to minimize the perplexity, we only see the improvement in perplexity (see the right plots in Fig. 2.) This confirms our earlier claim that it is necessary and desirable to tune for the target decoding objective regardless of what the underlying translation system was trained for, and strongly supports the proposed idea of trainable decoding.

The improvement from using the proposed trainable greedy decoding is smaller when used together with beam search, as seen in Fig. 2 (b). However, we still observe statistically significant improvement in terms of BLEU (marked with red stars.) This suggests a future direction in which we extend the proposed trainable greedy decoding to directly incorporate beam search into its training procedure to further improve the translation quality.

It is worthwhile to note that we achieved all of these improvements with negligible computational overhead. This is due to the fact that our actor is a very small, shallow neural network, and that the more complicated critic is thrown away after training. We suspect the effectiveness of such a small actor is due to the well-structured hidden state space of the underlying neural machine translation model which was trained with a large amount of parallel corpus. We believe this favourable computational complexity makes the proposed method suitable for production-grade neural machine translation (Wu et al., 2016; Crego et al., 2016).

Importance of Critic-Aware Actor Learning In Fig. 3, we show sample learning curves with and without the proposed critic-aware actor learning. Both curves were from the models trained under the same condition. Despite a slower start in the early stage of learning, we see that the critic-aware actor learning has greatly stabilized the learning progress. We emphasize that we would not have been able to train all these 16 actors without the proposed critic-aware actor learning.

Examples In Fig. 4, we present three examples from Ru-En. We defined the influence as the KL divergence between the conditional distributions without the trainable greedy decoding and with the trainable greedy decoding, assuming the fixed previous hidden state and target symbol. We colored a target word with magenta, when the influence of the trainable greedy decoding is large (> 0.001). Manual inspection of these examples as well as others has revealed that the trainable greedy decoder focuses on fixing prepositions and removing any unnecessary symbol generation. More in-depth analysis is however left as future work.

6 Conclusion

We proposed trainable greedy decoding as a way to learn a decoding algorithm for neural machine translation with an arbitrary decoding objective. The proposed trainable greedy decoder observes and manipulates the hidden state of a trained neural translation system, and is trained by a novel variant of deterministic policy gradient, called critic-aware actor learning. Our extensive experiments on eight language pair-directions and two objectives confirmed its validity and usefulness. The proposed trainable greedy decoding is a generic idea that can be applied to any recurrent language modeling, and we anticipate future research both on the fundamentals of the trainable decoding as well as on the applications to more diverse tasks such as image caption generating and dialogue modeling.

Acknowledgement

KC thanks the support by TenCent, eBay, Facebook, Google (Google Faculty Award 2016) and NVidia. This work was partly supported by Samsung Advanced Institute of Technology (Next Generation Deep Learning: from pattern recognition to AI). We sincerely thank Martin Arjovsky, Zihang Dai, Graham Neubig, Pengcheng Yin and Chunting Zhou for helpful discussions and insightful feedbacks.

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086* .

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

Kyunghyun Cho. 2016. Noisy parallel approximate decoding for conditional recurrent language model. *arXiv preprint arXiv:1605.03835* .

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

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. Nyu-mila neural machine translation systems for wmt16. In *Proceedings of the First Conference on Machine Translation, Berlin, Germany. Association for Computational Linguistics*.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. *arXiv preprint arXiv:1601.01085* .

Marta R Costa-Jussa and José AR Fonollosa. 2016. Character-based neural machine translation. *arXiv preprint arXiv:1603.00810* .

Josep Crego, Jungi Kim, Guillaume Klein, Anabel Rebollo, Kathy Yang, Jean Senellart, Egor Akhanov, Patrice Brunelle, Aurelien Coquard, Yongchao Deng, et al. 2016. Systran's pure neural machine translation systems. *arXiv preprint arXiv:1610.05540* .

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. ACL.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016a. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *NAACL*.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T Yarman Vural, and Kyunghyun Cho. 2016b. Zero-resource translation with multi-lingual neural machine translation. In *EMNLP*.

Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2016. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344* .

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor OK Li. 2016. Learning to translate in real-time with neural machine translation. *arXiv preprint arXiv:1610.00388* .

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798* .

Nicolas Heess, Gregory Wayne, David Silver, Tim Lillicrap, Tom Erez, and Yuval Tassa. 2015. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*. pages 2944–2952.

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

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* .

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*. pages 388–395.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017* .

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562* .

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Learning to decode for future success. *arXiv preprint arXiv:1701.06549* .

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* .

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 605.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586* .

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015a. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114* .

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *arXiv preprint arXiv:1604.00788* .

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015b. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. *arXiv preprint arXiv:1606.02891* .

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

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *ICML*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *NIPS* .

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4(2).

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2016a. Neural machine translation with reconstruction. *arXiv preprint arXiv:1611.01874* .

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016b. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811* .

Fernanda Viégas, Greg Corrado, Jeffrey Dean, Macduff Hughes, Martin Wattenberg, Maxim Krikun, Melvin Johnson, Mike Schuster, Nikhil Thorat, Quoc V Le, et al. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation .

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960* .

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv e-prints* .

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Satirical News Detection and Analysis using Attention Mechanism and Linguistic Features

Fan Yang and **Arjun Mukherjee**
Department of Computer Science
University of Houston
{fyang11,arjun}@uh.edu

Eduard Gragut
Computer and Information Sciences
Temple University
edragut@temple.edu

Abstract

Satirical news is considered to be entertainment, but it is potentially deceptive and harmful. Despite the embedded genre in the article, not everyone can recognize the satirical cues and therefore believe the news as true news. We observe that satirical cues are often reflected in certain paragraphs rather than the whole document. Existing works only consider document-level features to detect the satire, which could be limited. We consider paragraph-level linguistic features to unveil the satire by incorporating neural network and attention mechanism. We investigate the difference between paragraph-level features and document-level features, and analyze them on a large satirical news dataset. The evaluation shows that the proposed model detects satirical news effectively and reveals what features are important at which level.

1 Introduction

> *"When information is cheap, attention becomes expensive."* — James Gleick

Satirical news is considered to be entertainment. However, it is not easy to recognize the satire if the satirical cues are too subtle to be unmasked and the reader lacks the contextual or cultural background. The example illustrated in Table 1 is a piece of satirical news with subtle satirical cues.

Assuming readers interpret satirical news as true news, there is not much difference between satirical news and fake news in terms of the consequence, which may hurt the credibility of the media and the trust in the society. In fact, it is reported in *the Guardian* that people may believe satirical news and spread them to the public regardless of the ridiculous content[1]. It is also concluded that fake news is similar to satirical news via a thorough comparison among true news, fake news, and satirical news (Horne and Adali, 2017). This paper focuses on *satirical news detection* to ensure the trustworthiness of online news and prevent the spreading of potential misleading information.

> ...
> "Kids these days are done with stories where things happen," said CBC consultant and world's oldest child psychologist Obadiah Sugarman. "We'll finally be giving them the stiff Victorian morality that I assume is in vogue. Not to mention, doing a period piece is a great way to make sure white people are adequately represented on television."
> ...

Table 1: A paragraph of satirical news

Some works tackling fake news and misleading information favor to discover the truth (Xiao et al., 2016; Wan et al., 2016) through knowledge base (Dong et al., 2015) and truthfulness estimation (Ge et al., 2013). These approaches may not be feasible for satirical news because there is no ground-truth in the stories. Another track of works analyze social network activities (Zhao et al., 2015) to evaluate the spreading information (Gupta et al., 2012; Castillo et al., 2011). This could be ineffective for both fake news and satirical news because once they are distributed on the social network, the damage has been done. Finally, works evaluating culture difference (Pérez-Rosas and Mihalcea, 2014), psycholinguistic features (Ott et al., 2011), and writing styles (Feng et al., 2012) for deception detection are suitable for satirical news detection. These works consider features at document level, while we observe that satirical cues are usually located in certain para-

[1] https://www.theguardian.com/media/2016/nov/17/facebook-fake-news-satire

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1979–1989
Copenhagen, Denmark, September 7–11, 2017.

graphs rather than the whole document. This indicates that many document level features may be superfluous and less effective.

To understand how paragraph-level features and document-level features are varied towards detection decision when only document level labels are available, we propose a 4-level neural network in a character-word-paragraph-document hierarchy and utilize attention mechanism (Bahdanau et al., 2014) to reveal their relative difference. We apply psycholinguistic features, writing stylistic features, structural features, and readability features to understand satire. The paragraph-level features are embedded into attention mechanism for selecting highly attended paragraphs, and the document-level features are incorporated for the final classification. This is the first work that unveils satirical cues between paragraph-level and document-level through neural networks to our knowledge.

We make the following contributions in our paper:

- We propose a 4-level hierarchical network for satirical news detection. The model detects satirical news effectively and incorporates attention mechanism to reveal paragraph-level satirical cues.

- We show that paragraph-level features are more important than document-level features in terms of the psycholinguistic feature, writing stylistic feature, and structural feature, while the readability feature is more important at the document level.

- We collect satirical news (16,000+) and true news (160,000+) from various sources and conduct extensive experiments on this corpus[2].

2 Related Work

We categorize related works into four categories: content-based detection for news genre, truth verification and truthfulness evaluation, deception detection, and identification of highly attended component using attention mechanism.

Content-based detection for news genre.Content-based methods are considerably effective to prevent satirical news from being recognized as true news and spreading through social media. Burfoot and Baldwin (2009) introduce headline features, profanity, and slang to embody satirical news. They consider absurdity as the major device in satirical news and model this feature by comparing entity combination in a given document with Google query results. Rubin et al. (2016) also consider absurdity but model it through unexpected new name entities. They introduce additional features including humor, grammar, negative affect, and punctuation to empower the detection. Besides satirical news, Chen et al. (2015) aim to detect click-baits, whose content exaggerates fact. Potthast et al. (2017) report a writing style analysis of hyperpartisan news. Barbieri et al. (2015) focus on multilingual tweets that advertise satirical news.

It is noteworthy that satirical news used for evaluation in above works are of limited quantity (around 200 articles). Diverse examples of satire may not be included as discussed by Rubin et al. (2016). This issue inspires us to collect more than 16,000 satirical news for our experiment.

Truth discovery and truthfulness evaluation. Although truth extraction from inconsistent sources (Ge et al., 2013; Wan et al., 2016; Li et al., 2016) and from conflicting sources (Yin et al., 2008; Li et al., 2014b), truth inference through knowledge base (Dong et al., 2015), and discovering evolving truth (Li et al., 2015) could help identify fact and detect fake news, they cannot favor much for satirical news as the story is entirely made up and the ground-truth is hardly found. Analyzing user activities (Farajtabar et al., 2017) and interactions (Castillo et al., 2011; Mukherjee and Weikum, 2015) to evaluate the credibility may not be appropriate for satirical news as it cannot prevent the spreading. Therefore, we utilize content-based features, including psycholinguistic features, writing stylistic features, structural features, and readability features, to address satirical news detection.

Deception detection. We believe satirical news and opinion spam share similar characteristics of writing fictitious and deceptive content, which can be identified via a psycholinguistic consideration (Mihalcea and Strapparava, 2009; Ott et al., 2011). Beyond that, both syntactic stylometry (Feng et al., 2012) and behavioral features (Mukherjee et al., 2013b) are effective for detecting deceptive reviews, while stylistic features are practical to deal with obfuscating and imitat-

[2] Please contact the first author to obtain the data

ing writings (Afroz et al., 2012). However, deceptive content varies among paragraphs in the same document, and so does satire. We focus on devising and evaluating paragraph-level features to reveal the satire in this work. We compare them with features at the document level, so we are able to tell what features are important at which level.

Identification of highly attended component using attention mechanism. Attention mechanism is widely applied in machine translation (Bahdanau et al., 2014), language inference (Rocktäschel et al., 2015), and question answering (Chen et al., 2016a). In addition, Yang et al. (2016b) propose hierarchical attention network to understand both attended words and sentences for sentiment classification. Chen et al. (2016b) enhance the attention with the support of user preference and product information to comprehend how user and product affect sentiment ratings. Due to the capability of attention mechanism, we employ the same strategy to show attended component for satirical news. Different from above works, we further evaluate linguistic features of highly attended paragraphs to analyze characteristics of satirical news, which has not been explored to our knowledge.

3 The Proposed Model

We first present our 4-level hierarchical neural network and explain how linguistic features can be embedded in the network to reveal the difference between paragraph level and document level. Then we describe the linguistic features.

3.1 The 4-Level Hierarchical Model

We build the model in a hierarchy of character-word-paragraph-document. The general overview of the model can be viewed in Figure 1 and the notations are listed in Table 2.

	Meaning
Superscript	Lowercase for notation purpose; $\top$ means matrix transpose.
Subscript	For index purpose.
Parameter	$\mathbf{W}$,$\mathbf{U}$,$\mathbf{w}^c$,$\mathbf{v}^a$: learnable weights; b: learnable bias.
Representation	$\mathbf{c}$: character; $\mathbf{x}$: word; $\mathbf{p}$: paragraph; $\mathbf{d}$: document; $\tilde{y}$: prediction $\mathbf{l}$: linguistic vector; y: label; $\mathbf{r}$: reset gate; $\mathbf{z}$: update gate; $\mathbf{h}$: hidden state for GRU; $\mathbf{u}$: hidden state for attention.

Table 2: Notations and meanings

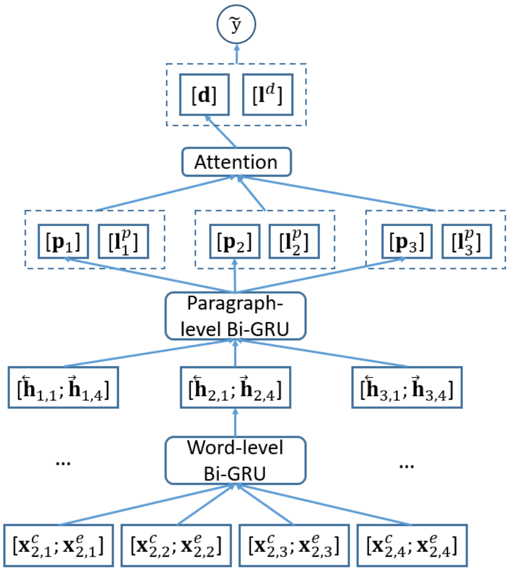

$\tilde{y}$
$[\mathbf{d}]$
$[\mathbf{l}^d]$
Attention
$[\mathbf{p}_1]$
$[\mathbf{l}^p_1]$
$[\mathbf{p}_2]$
$[\mathbf{l}^p_2]$
$[\mathbf{p}_3]$
$[\mathbf{l}^p_3]$
Paragraph-level Bi-GRU
$[\overleftarrow{\mathbf{h}}_{1,1}; \overrightarrow{\mathbf{h}}_{1,4}]$
$[\overleftarrow{\mathbf{h}}_{2,1}; \overrightarrow{\mathbf{h}}_{2,4}]$
$[\overleftarrow{\mathbf{h}}_{3,1}; \overrightarrow{\mathbf{h}}_{3,4}]$
...
Word-level Bi-GRU
...
$[\mathbf{x}^c_{2,1}; \mathbf{x}^e_{2,1}]$
$[\mathbf{x}^c_{2,2}; \mathbf{x}^e_{2,2}]$
$[\mathbf{x}^c_{2,3}; \mathbf{x}^e_{2,3}]$
$[\mathbf{x}^c_{2,4}; \mathbf{x}^e_{2,4}]$

Figure 1: The overview of the proposed model. The document has 3 paragraphs and each paragraph contains 4 words. We omit character-level convolution neural network but leave $\mathbf{x}^c$ to symbolize the representation learned from it.

3.1.1 Character-Level Encoder

We use convolutional neural networks (CNN) to encode word representation from characters. CNN is effective in extracting morphological information and name entities (Ma and Hovy, 2016), both of which are common in news. Each word is presented as a sequence of n characters and each character is embedded into a low-dimension vector. The sequence of characters $\mathbf{c}$ is brought to the network. A convolution operation with a filter $\mathbf{w}^c$ is applied and moved along the sequence. Max pooling is performed to select the most important feature generated by the previous operation. The word representation $\mathbf{x}^c \in \mathbb{R}^f$ is generated with f filters.

3.1.2 Word-Level Encoder

Assume a sequence of words of paragraph i arrives at time t. The current word representation $\mathbf{x}_{i,t}$ concatenates $\mathbf{x}^c_{i,t}$ from character level with pretrained word embedding $\mathbf{x}^e_{i,t}$, as $\mathbf{x}_{i,t} = [\mathbf{x}^c_{i,t}; \mathbf{x}^e_{i,t}]$. Examples are given in Figure 1. We implement Gated Recurrent Unit (GRU) (Cho et al., 2014) rather than LSTM (Hochreiter and Schmidhuber, 1997) to encode the sequence because GRU has fewer parameters. The GRU adopts reset gate $\mathbf{r}_{i,t}$ and update gate $\mathbf{z}_{i,t}$ to control the information flow between the input $\mathbf{x}_{i,t}$ and the candidate state $\tilde{\mathbf{h}}_{i,t}$. The output hidden state $\mathbf{h}_{i,t}$ is computed by manipulating previous state $\mathbf{h}_{i,t-1}$ and the candidate state $\tilde{\mathbf{h}}_{i,t}$ regarding to $\mathbf{z}_{i,t}$ as in Equation 4, where $\odot$ denotes element-wise multiplication.

$$\mathbf{z}_{i,t} = \sigma(\mathbf{W}^z\mathbf{x}_{i,t} + \mathbf{U}^z\mathbf{h}_{i,t-1} + b^z) \tag{1}$$

$$\mathbf{r}_{i,t} = \sigma(\mathbf{W}^r\mathbf{x}_{i,t} + \mathbf{U}^r\mathbf{h}_{i,t-1} + b^r) \tag{2}$$

$$\tilde{\mathbf{h}}_{i,t} = \tanh(\mathbf{W}^h\mathbf{x}_{i,t} + \mathbf{r}_{i,t} \odot (\mathbf{U}^h\mathbf{h}_{i,t-1} + b^h)) \tag{3}$$

$$\mathbf{h}_{i,t} = (1 - \mathbf{z}_{i,t}) \odot \mathbf{h}_{i,t-1} + \mathbf{z}_{i,t} \odot \tilde{\mathbf{h}}_{i,t} \tag{4}$$

To learn a better representation from the past and the future, we use bidirectional-GRU (Bi-GRU) to read the sequence of words with forward $\overrightarrow{\text{GRU}}$ from $\mathbf{x}_{i,1}$ to $\mathbf{x}_{i,t}$, and backward $\overleftarrow{\text{GRU}}$ from $\mathbf{x}_{i,t}$ to $\mathbf{x}_{i,1}$. The final output of Bi-GRU concatenates the last state of $\overrightarrow{\text{GRU}}$ and $\overleftarrow{\text{GRU}}$, as $[\overrightarrow{\mathbf{h}}_{i,t}; \overleftarrow{\mathbf{h}}_{i,1}]$, to represent the ith paragraph.

3.1.3 Paragraph-Level Attention

We observe that not all paragraphs have satire and some of them are functional to make the article complete, so we incorporate attention mechanism to reveal which paragraphs contribute to decision making. Assuming a sequence of paragraph representations have been constructed from lower levels, another Bi-GRU is used to encode these representations to a series of new states $\mathbf{p}_{1:t}$, so the sequential orders are considered.

To decide how paragraphs should be attended, we calculate satirical degree α_i of paragraph i. We first convey $\mathbf{p}_i$ into hidden states $\mathbf{u}_i$ as in Equation 5. Then we product $\mathbf{u}_i$ with a learnable satire-aware vector $\mathbf{v}^a$ and feed the result into softmax function as in Equation 6. The final document representation $\mathbf{d}$ is computed as a weighted sum of α_i and $\mathbf{p}_i$.

$$\mathbf{u}_i = \tanh(\mathbf{W}^a\mathbf{p}_i + b^a) \tag{5}$$

$$\alpha_i = \frac{\exp(\mathbf{u}_i^\top\mathbf{v}^a)}{\sum_{j=0}^{t}\exp(\mathbf{u}_j^\top\mathbf{v}^a))} \tag{6}$$

$$\mathbf{d} = \sum_{i=0}^{t}\alpha_i\mathbf{p}_i \tag{7}$$

Linguistic features are leveraged to support attending satire paragraph. Besides $\mathbf{p}_i$, we represent paragraph i based on our linguistic feature set and transform it into a high-level feature vector $\mathbf{l}_i^p$ via multilayer perceptron (MLP). So $\mathbf{u}_i$ in Equation 5 is updated to:

$$\mathbf{u}_i = \tanh(\mathbf{W}^a\mathbf{p}_i + \mathbf{U}^a\mathbf{l}_i^p + b^a) \tag{8}$$

3.1.4 Document-Level Classification

Similar to the paragraph level, we represent document j based on our linguistic feature set and transform it into a high-level feature vector $\mathbf{l}_j^d$ via MLP. We concatenate $\mathbf{d}_j$ and $\mathbf{l}_j^d$ together for classification. Suppose $y_j \in (0, 1)$ is the label of the document j, the prediction $\tilde{y}_j$ and the loss function $\mathcal{L}$ over N documents are:

$$\tilde{y}_j = \text{sigmoid}(\mathbf{W}^d\mathbf{d}_j + \mathbf{U}^d\mathbf{l}_j^d + b^d) \tag{9}$$

$$\mathcal{L} = -\frac{1}{N}\sum_{j}^{N} y_j \log \tilde{y}_j + (1 - y_j)\log(1 - \tilde{y}_j) \tag{10}$$

3.2 Linguistic Features

Linguistic features have been successfully applied to expose differences between deceptive and genuine content, so we subsume most of the features in previous works. The idea of explaining fictitious content is extended here to reveal how satirical news differs from true news. We divide our linguistic features into four families and compute them separately for paragraph and document.

Psycholinguistic Features: Psychological differences are useful for our problem, because professional journalists tend to express opinion conservatively to avoid unnecessary arguments. On the contrary, satirical news includes aggressive language for the entertainment purpose. We additionally observe true news favors clarity and accuracy while satirical news is related to emotional cognition. To capture the above observations, we employ Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2007) as our psycholinguistic dictionary. Each category of LIWC is one independent feature and valued by its frequency[3].

Writing Stylistic Features: The relative distribution of part-of-speech (POS) tags reflects informative vs. imaginative writing, which contributes to detecting deceptions (Li et al., 2014a; Mukherjee et al., 2013a). We argue that the stories covered by satirical news are based on imagination. In addition, POS tags are hints of the underlying

[3] Total counts divided by total words.

	#Train	#Validation	#Test	#Para	#Sent	#Words	# Capitals	#Punc	#Digits
True	101,268	33,756	33,756	20±7.8	32±24	734±301	118±58	28±26	93±49
Satire	9,538	3,103	3,608	12±4.4	25±12	587±246	87±44	11±13	86±43

Table 3: The split and the description (mean and standard deviation) of the dataset. Para denotes paragraphs, sent denotes sentences, and punc denotes punctuations.

humor (Reyes et al., 2012), which is common in satirical news. So we utilize POS tags (Toutanova et al., 2003) to apprehend satire. Each tag is regarded as one independent feature and valued by its frequency.

Readability Features: We consider readability of genuine news would differ from satirical news because the former is written by professional journalists and tend to be clearer and more accurate, while satirical news packs numerous clauses to enrich the made-up story as introduced by Rubin et al. (2016). Different from their work, we use readability metrics, including Flesch Reading Ease (Kincaid et al., 1975), Gunning Fog Index (Gunning, 1952), Automated Readability Index (Senter and Smith, 1967), ColemanLiau Index (Coleman and Liau, 1975), and syllable count per word, as features.

Structural Features: To further reflect the structure of news articles, we examine the following features: word count, log word count, number of punctuations, number of digits, number of capital letters, and number of sentences.

4 Experiment and Evaluation

We report satirical news detection results and show high weighted word features. Then, we provide a thorough analysis between paragraph-level and document-level features. Finally, we visualize an example of satirical news article to demonstrate the effectiveness of our work.

4.1 Dataset

The satirical news is collected from 14 websites that explicitly declare they are offering satire, so the correct label can be guaranteed. We also notice websites that mix true news, fake news, and satirical news. We exclude these websites in this work because it requires experts to annotate the news articles.

We maintain each satire source in only one of the train/validation/test sets[4] as the cross-domain setting in (Li et al., 2014a). Otherwise, the problem may become writing pattern recognition or news site classification. We also combined different sources together[5] as a similar setting of leveraging multiple domains (Yang et al., 2016a). The true news is collected from major news outlets[6] and Google News using FLORIN (Liu et al., 2015). The satirical news in the corpus is significantly less than true news, reflecting an impressionistic view of the reality. We omit headline, creation time, and author information so this work concentrates on the satire in the article body. We realize the corpus may contain different degree of satire. Without the annotation, we only consider binary classification in this work and leave the degree estimation for the future. The split and the description of the dataset can be found in Table 3.

4.2 Implementation Detail

For SVM, we use the sklearn implementation[7]. We find that using linear kernel and setting "class_weight" to "balanced" mostly boost the result. We search soft-margin penalty "C" and find high results occur in range $[10^{-1}, 10^{-4}]$. We use the validation set to tune the model so selecting hyper-parameters is consistent with neural network based model.

For neural network based models, we use the Theano package (Bastien et al., 2012) for implementation. The lengths of words, paragraphs, and documents are fixed at 24, 128, and 16 with necessary padding or truncating. Stochastic Gradient Descent is used with initial learning rate of 0.3 and decay rate of 0.9. The training is early stopped if the F1 drops 5 times continuously. Word embeddings are initialized with 100-dimension Glove embeddings (Pennington et al., 2014). Character embeddings are randomly initialized with 30 dimensions. Specifically for the proposed model, the following hyper-parameters are estimated based on the validation set and used

[4] Train: Onion, the Spoof. Test: SatireWorld, Beaverton, Ossurworld. Validation: DailyCurrent, DailyReport, EnduringVision, Gomerblog, NationalReport, SatireTribune, SatireWire, Syruptrap, and UnconfirmedSource.

[5] The combination is chosen to ensure enough training examples and balanced validation/test sets.

[6] CNN, DailyMail, WashingtonPost, NYTimes, TheGuardian, and Fox.

[7] sklearn.svm.SVC

Model	Validation				Test			
	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
SVM word n-grams	97.69	87.45	84.66	86.03	97.46	89.59	83.45	86.41
SVM word n-grams + LF	97.73	86.06	**87.14**	86.60	97.52	88.44	85.48	86.93
SVM word + char n-grams	97.43	87.10	81.57	84.24	97.64	90.76	84.12	87.31
SVM word + char n-grams + LF	97.76	90.13	82.44	86.11	97.93	92.71	85.31	88.86
SVM Rubin et al. (2016)	97.73	90.21	81.92	85.86	97.79	**93.47**	82.95	87.90
SVM Rubin et al. (2016) + char tf-idf + LF	**97.93**	**90.99**	83.69	**87.19**	**98.09**	92.98	**86.72**	**89.75**
Bi-GRU	97.67	89.17	82.28	85.58	97.58	93.11	80.96	86.61
SVM Doc2Vec Le and Mikolov (2014)	92.48	58.48	71.66	64.40	90.48	50.52	67.88	57.92
HAN Yang et al. (2016b)	97.91	92.06	82.24	86.88	97.83	90.85	86.17	88.45
4LHN	98.44	92.82	88.33	90.52	98.36	94.61	88.00	91.18
4LHNP	98.46	93.54	87.75	90.56	98.39	94.63	88.33	91.37
4LHND	98.36	**94.73**	85.24	89.74	98.18	**95.35**	85.31	90.05
4LHNPD	**98.54**	93.31	**89.01**	**91.11**	**98.39**	93.51	**89.50**	**91.46**

Table 4: Satirical news detection results.

in the final test set. The dropout is applied with probability of 0.5. The size of the hidden states is set at 60. We use 30 filters with window size of 3 for convolution.

4.3 Performance of Satirical News Detection

We report accuracy, precision, recall, and F1 on the validation set and the test set. All metrics take satirical news as the positive class. Both paragraph-level and document-level linguistic features are scaled to have zero mean and unit variance, respectively. The compared methods include:

SVM word n-grams: Unigram and bigrams of the words as the baseline. We report 1,2-grams because it performs better than other n-grams.

SVM word n-grams + LF: 1,2-word grams plus linguistic features. We omit comparison with similar work (Ott et al., 2011) as their features are subsumed in ours.

SVM word + char n-grams: 1,2-word grams plus bigrams and trigrams of the characters.

SVM word + char n-grams + LF: All the proposed features are considered.

SVM Rubin et al. (2016): Unigram and bigrams tf-idf with satirical features as proposed in (Rubin et al., 2016). We compare with (Rubin et al., 2016) rather than (Burfoot and Baldwin, 2009) as the former claims a better result.

SVM Rubin et al. (2016) + char tf-idf + LF: Include all possible features.

Bi-GRU: Bi-GRU for document classification. The document representation is the average of the hidden state at every time-step.

SVM Doc2Vec: Unsupervised method learning distributed representation for documents (Le and Mikolov, 2014). The implementation is based on Gensim (Řehůřek and Sojka, 2010).

HAN: Hierarchical Attention Network (Yang et al., 2016b) for document classification with both word-level and sentence-level attention.

4LHN: 4-Level Hierarchical Network without any linguistic features.

4LHNP: 4-Level Hierarchical Network with Paragraph-level linguistic features.

4LHND: 4-Level Hierarchical Network with Document-level linguistic features.

4LHNPD: 4-Level Hierarchical Network with both Paragraph-level and Document-level linguistic features.

In Table 4, the performances on the test set are generally better than on the validation set due to the cross-domain setting. We also explored word-level attention (Yang et al., 2016b), but it performed 2% worse than 4LHN. The result of Doc2Vec is limited. We suspect the reason could be the high imbalanced dataset, as an unsupervised learning method for document representation heavily relies on the distribution of the document.

4.4 Word Level Analysis

True		Satire	
:	day	"	stated
video	said the	sources	press
but the	twitter	continued	reporter
in statement	told the	added	resident
com	pictured	washington dc	said that

Table 5: High weighted word-level features

We report high weighted word-grams in Table 5 based on the SVM model as incorporating word-level attention in our neural hierarchy model reduces the detection performance. According

Psycholinguistic Feature					Writing Stylistic Feature					Readability Feature				
Name	S.m	S.std	T.m	T.std	Name	S.m	S.std	T.m	T.std	Name	S.m	S.std	T.m	T.std
Human.P	.011	.021	.009	.023	**JJ.P**	.061	.045	.058	.046	**FRE.D**	58.4	12.2	56.0	10.1
Past.P	.034	.035	.040	.042	**PRP.P**	.054	.047	.044	.047	**CLI.D**	9.08	1.66	9.48	1.61
Self.P	.017	.032	.010	.027	**RB.P**	.051	.048	.045	.054	**FOG.D**	13.71	3.25	14.00	2.89
Funct.D	.453	.045	.437	.049	**VBN.P**	.021	.026	.024	.031	Structural Feature				
Social.P	.097	.067	.091	.073	**NN.D**	.273	.038	.300	.043	**Punc.P**	7.69	5.35	4.69	3.83
Leisure.P	.017	.027	.018	.032	**VBZ.P**	.019	.026	.021	.029	**Cap.P**	7.44	6.08	5.75	4.8
Hear.P	.011	.019	.012	.021	**CC.P**	.023	.024	.024	.026	**Digit.P**	0.97	2.40	1.39	3.00
Bio.P	.026	.035	.023	.036	**CD.P**	.013	.027	.024	.043	**LogWc.P**	3.69	0.71	3.39	0.53

Table 6: Comparing feature values within each category. P stands for paragraph level. D stands for document level. S stands for satirical news. T stands for true news. m stands for mean and std stands for standard deviation. FRE: Flesch Reading Ease, the lower the harder. CLI: ColemanLiau Index. FOG: Gunning Fog Index. Punc: punctuation. Cap: Capital letters. LogWc: Log Word count

to Table 5, we conclude satirical news mimics true news by using news related words, such as "stated" and "reporter". However, these words may be over used so they can be detected. True news may use other evidence to support the credibility, which explains "twitter", "com", "video", and "pictured". High weight of " : " indicates that true news uses colon to list items for clarity. High weight of " " " indicates that satirical news involves more conversation, which is consistent with our observation. The final interesting note is satirical news favors "washington dc". We suspect that satirical news mostly covers politic topics, or satire writers do not spend efforts on changing locations.

4.5 Analysis of Weighted Linguistic Features

We use 4LHNPD to compare paragraph-level and document-level features, as 4LHNPD leverages the two-level features into the same framework and yields the best result.

Because all linguistic features are leveraged into MLP with non-linear functions, it is hard to check which feature indicates satire. Alternatively, we define the importance of linguistic features by summing the absolute value of the weights if directly connected to the feature. For example, the importance I of feature k is given by $\mathrm{I}_k = \frac{1}{M}\sum_{m=0}^{M}|\mathbf{w}_{k,m}|$, where $\mathbf{w} \in \mathbb{R}^{K\times M}$ is the directly connected weight, K is the number of features, and M is the dimension of the output. This metric gives a general idea about how much does a feature contribute to the decision making.

We first report the scaled importance of the four linguistic feature sets by averaging the importance of individual linguistic features. Then we report individual important features within each set.

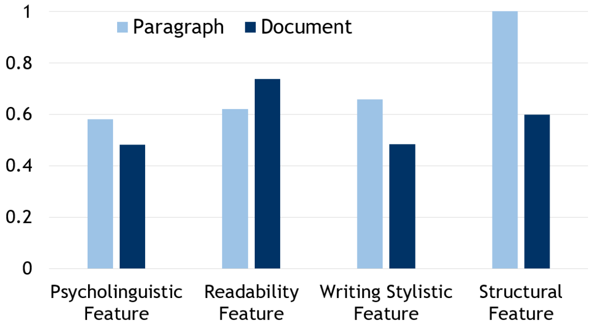


Figure 2: Comparing the importance of the four feature sets at paragraph level and document level.

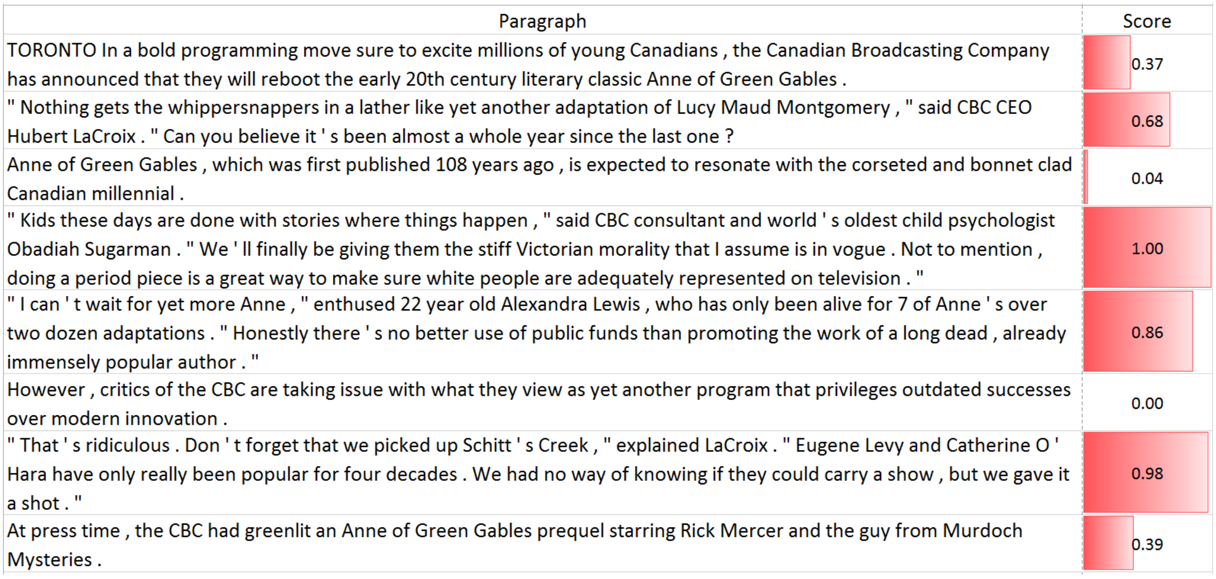
4.5.1 Comparing the Four Feature Sets

According to Figure 2, the importance of paragraph-level features is greater than document-level features except for the readability feature set. It is reasonable to use readability at the document level because readability features evaluate the understandability of a given text, which depends on the content and the presentation. The structural feature set is highly weighted for selecting attended paragraph, which inspires us to focus on individual features inside the structural feature set.

4.5.2 Comparing Individual Features

Within each set, we rank features based on the importance score and report their mean and standard deviation before being scaled in Table 6. At paragraph level, we use top three attended paragraphs for calculating. The respective p-values of all features in the table are less than 0.01 based on the t-test, indicating satirical news is statistically significantly different from true news.

Comparing Table 6 and Table 3, we find that the word count, capital letters, and punctuations in true news are larger than in satirical news at the document level, while at paragraph level these



Paragraph	Score
TORONTO In a bold programming move sure to excite millions of young Canadians , the Canadian Broadcasting Company has announced that they will reboot the early 20th century literary classic Anne of Green Gables .	0.37
" Nothing gets the whippersnappers in a lather like yet another adaptation of Lucy Maud Montgomery , " said CBC CEO Hubert LaCroix . " Can you believe it ' s been almost a whole year since the last one ?	0.68
Anne of Green Gables , which was first published 108 years ago , is expected to resonate with the corseted and bonnet clad Canadian millennial .	0.04
" Kids these days are done with stories where things happen , " said CBC consultant and world ' s oldest child psychologist Obadiah Sugarman . " We ' ll finally be giving them the stiff Victorian morality that I assume is in vogue . Not to mention , doing a period piece is a great way to make sure white people are adequately represented on television . "	1.00
" I can ' t wait for yet more Anne , " enthused 22 year old Alexandra Lewis , who has only been alive for 7 of Anne ' s over two dozen adaptations . " Honestly there ' s no better use of public funds than promoting the work of a long dead , already immensely popular author . "	0.86
However , critics of the CBC are taking issue with what they view as yet another program that privileges outdated successes over modern innovation .	0.00
" That ' s ridiculous . Don ' t forget that we picked up Schitt ' s Creek , " explained LaCroix . " Eugene Levy and Catherine O ' Hara have only really been popular for four decades . We had no way of knowing if they could carry a show , but we gave it a shot . "	0.98
At press time , the CBC had greenlit an Anne of Green Gables prequel starring Rick Mercer and the guy from Murdoch Mysteries .	0.39

Figure 3: An example of attended paragraphs.

features in true news are less than in satirical news. This indicates satire paragraph could be more complex locally. It also could be referred as "sentence complexity", that *"satirical articles tend to pack a great number of clauses into a sentence for comedic effect"* (Rubin et al., 2016). Accordingly, we hypothesize top complex paragraphs could represent the entire satire document for classification, which we leave for future examination.

In Table 6, psycholinguistic feature "Humans" is more related to emotional writing than control writing (Pennebaker et al., 2007), which indicates satirical news is emotional and unprofessional compared to true news. The same reason also applies to "Social" and "Leisure", where the former implies emotional and the latter implies control writing. The "Past" and "VBN" both have higher frequencies in true news, which can be explained by the fact that true news covers what happened. A similar reason that true news reports what happened to others explains a low "Self" and a high "VBZ" in true news.

For writing stylistic features, it is suggested that informative writing has more nouns, adjectives, prepositions and coordinating conjunctions, while imaginative writing has more verbs, adverbs, pronouns, and pre-determiners (Rayson et al., 2001). This explains higher frequencies of "RB" and "PRP" in satirical news, and higher frequency of "NN" and "CC" in true news. One exception is "JJ", adjectives, which receives the highest weight in this feature set and indicates a higher frequency in satirical news. We suspect adjective could also be related to emotional writing, but more experiments are required.

Readability suggests satirical news is easier to be understood. Considering satirical news is also deceptive (as the story is not true), this is consistent with works (Frank et al., 2008; Afroz et al., 2012) showing deceptive writings are more easily comprehended than genuine writings. Finally, true news has more digits and a higher "CD"(Cardinal number) frequency, even at the paragraph level, because they tend to be clear and accurate.

4.6 Visualization of Attended Paragraph

To explore the attention, we sample one example in the validation set and present it in Figure 3. The value at the right represents the scaled attention score. The high attended paragraphs are longer and have more capital letters as they are referring different entities. They have more double quotes, as multiple conversations are involved.

Moreover, we subjectively feel the attended paragraph with score 0.98 has a sense of humor while the paragraph with score 0.86 has a sense of sarcasm, which are common in satire. The paragraph with score 1.0 presents controversial topics, which could be misleading if the reader cannot understand the satire. This is what we expect from the attention mechanism. Based on the visualization, we also feel this work could be generalized to detect figurative languages.

5 Conclusion

In this paper, we proposed a 4-level hierarchical network and utilized attention mechanism to understand satire at both paragraph level and document level. The evaluation suggests readability features support the final classification while psycholinguistic features, writing stylistic features, and structural features are beneficial at the paragraph level. In addition, although satirical news is shorter than true news at the document level, we find satirical news generally contain paragraphs which are more complex than true news at the paragraph level. The analysis of individual features reveals that the writing of satirical news tends to be emotional and imaginative.

We will investigate efforts to model satire at the paragraph level following our conclusion and theoretical backgrounds, such as (Ermida, 2012). We plan to go beyond the binary classification and explore satire degree estimation. We will generalize our approach to reveal characteristics of figurative language (Joshi et al., 2016), where different paragraphs or sentences may reflect different degrees of sarcasm, irony, and humor.

Acknowledgments

The authors would like to thank the anonymous reviewers for their comments. This work was support in part by the U.S. NSF grants 1546480 and 1527364.

References

Sadia Afroz, Michael Brennan, and Rachel Greenstadt. 2012. Detecting hoaxes, frauds, and deception in writing style online. In *2012 IEEE Symposium on Security and Privacy*, pages 461–475. IEEE.

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

Francesco Barbieri, Francesco Ronzano, and Horacio Saggion. 2015. Do we criticise (and laugh) in the same way? automatic detection of multi-lingual satirical news in twitter. In *IJCAI*, pages 1215–1221.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. 2012. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*.

Clint Burfoot and Timothy Baldwin. 2009. Automatic satire detection: Are you having a laugh? In *Proceedings of the ACL-IJCNLP 2009 conference short papers*, pages 161–164. Association for Computational Linguistics.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 675–684. ACM.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016a. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016b. Neural sentiment classification with user and product attention. In *Proceedings of EMNLP*.

Yimin Chen, Niall J Conroy, and Victoria L Rubin. 2015. Misleading online content: Recognizing clickbait as false news. In *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*, pages 15–19. ACM.

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

Meri Coleman and Ta Lin Liau. 1975. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283.

Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. 2015. Knowledge-based trust: Estimating the trustworthiness of web sources. *Proceedings of the VLDB Endowment*, 8(9):938–949.

Isabel Ermida. 2012. News satire in the press: Linguistic construction of humour inspoof news articles. *Language and humour in the media*, page 185.

Mehrdad Farajtabar, Jiachen Yang, Xiaojing Ye, Huan Xu, Rakshit Trivedi, Elias Khalil, Shuang Li, Le Song, and Hongyuan Zha. 2017. Fake news mitigation via point process based intervention. *arXiv preprint arXiv:1703.07823*.

Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. Syntactic stylometry for deception detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 171–175. Association for Computational Linguistics.

Mark G Frank, Melissa A Menasco, and Maureen O'Sullivan. 2008. Human behavior and deception detection. *Wiley Handbook of Science and Technology for Homeland Security*.

Liang Ge, Jing Gao, Xiaoyi Li, and Aidong Zhang. 2013. Multi-source deep learning for information trustworthiness estimation. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 766–774. ACM.

Robert Gunning. 1952. The technique of clear writing.

Manish Gupta, Peixiang Zhao, and Jiawei Han. 2012. Evaluating event credibility on twitter. In *SDM*, pages 153–164. SIAM.

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

Benjamin D Horne and Sibel Adali. 2017. This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *arXiv preprint arXiv:1703.09398*.

Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. 2016. Automatic sarcasm detection: A survey. *arXiv preprint arXiv:1602.03426*.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, DTIC Document.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196.

Jiwei Li, Myle Ott, Claire Cardie, and Eduard H Hovy. 2014a. Towards a general rule for identifying deceptive opinion spam. In *ACL (1)*, pages 1566–1576. Citeseer.

Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. 2014b. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1187–1198. ACM.

Xian Li, Weiyi Meng, and Yu Clement. 2016. Verification of fact statements with multiple truthful alternatives. In *12th International Conference on Web Information Systems and Technologies*.

Yaliang Li, Qi Li, Jing Gao, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2015. On the discovery of evolving truth. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 675–684. ACM.

Qingyuan Liu, Eduard C Dragut, Arjun Mukherjee, and Weiyi Meng. 2015. Florin: a system to support (near) real-time applications on user generated content on daily news. *Proceedings of the VLDB Endowment*, 8(12):1944–1947.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Rada Mihalcea and Carlo Strapparava. 2009. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 309–312. Association for Computational Linguistics.

Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013a. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 632–640. ACM.

Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie S Glance. 2013b. What yelp fake review filter might be doing? In *ICWSM*.

Subhabrata Mukherjee and Gerhard Weikum. 2015. Leveraging joint interactions for credibility analysis in news communities. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 353–362. ACM.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 309–319. Association for Computational Linguistics.

James W Pennebaker, Cindy K Chung, Molly Ireland, Amy Gonzales, and Roger J Booth. 2007. The development and psychometric properties of liwc2007. austin, tx, liwc. net.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Verónica Pérez-Rosas and Rada Mihalcea. 2014. Cross-cultural deception detection. In *ACL (2)*, pages 440–445.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2017. A stylometric inquiry into hyperpartisan and fake news. *arXiv preprint arXiv:1702.05638*.

Paul Rayson, Andrew Wilson, and Geoffrey Leech. 2001. Grammatical word class variation within the british national corpus sampler. *Language and Computers*, 36(1):295–306.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering*, 74:1–12.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Victoria Rubin, Niall Conroy, Yimin Chen, and Sarah Cornwell. 2016. Fake news or truth? using satirical cues to detect potentially misleading news. In *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, pages 7–17, San Diego, California. Association for Computational Linguistics.

RJ Senter and Edgar A Smith. 1967. Automated readability index. Technical report, DTIC Document.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Mengting Wan, Xiangyu Chen, Lance Kaplan, Jiawei Han, Jing Gao, and Bo Zhao. 2016. From truth discovery to trustworthy opinion discovery: An uncertainty-aware quantitative modeling approach. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1885–1894. ACM.

Houping Xiao, Jing Gao, Qi Li, Fenglong Ma, Lu Su, Yunlong Feng, and Aidong Zhang. 2016. Towards confidence in the truth: A bootstrapping based truth discovery approach. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1935–1944. ACM.

Fan Yang, Arjun Mukherjee, and Yifan Zhang. 2016a. Leveraging multiple domains for sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2978–2988, Osaka, Japan. The COLING 2016 Organizing Committee.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016b. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xiaoxin Yin, Jiawei Han, and S Yu Philip. 2008. Truth discovery with multiple conflicting information providers on the web. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):796–808.

Zhe Zhao, Paul Resnick, and Qiaozhu Mei. 2015. Enquiring minds: Early detection of rumors in social media from enquiry posts. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1395–1405. ACM.

Fine-Grained Citation Span Detection for References in Wikipedia

Besnik Fetahu[1], Katja Markert[2] and Avishek Anand[1]

[1] L3S Research Center, Leibniz University of Hannover
Hannover, Germany
{fetahu, anand}@L3S.de
[2] Institute of Computational Linguistics, Heidelberg University
Heidelberg, Germany
markert@cl.uni-heidelberg.de

Abstract

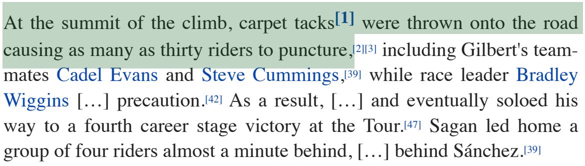
Verifiability is one of the core editing principles in Wikipedia, editors being encouraged to provide citations for the added content. For a Wikipedia article, determining the *citation span* of a citation, i.e. what content is covered by a citation, is important as it helps decide for which content citations are still missing.

We are the first to address the problem of determining the *citation span* in Wikipedia articles. We approach this problem by classifying which textual fragments in an article are covered by a citation. We propose a sequence classification approach where for a paragraph and a citation, we determine the citation span at a fine-grained level.

We provide a thorough experimental evaluation and compare our approach against baselines adopted from the scientific domain, where we show improvement for all evaluation metrics.

1 Introduction

Citations uphold the crucial policy of *verifiability* in Wikipedia. This policy requires Wikipedia contributors to support their additions with citations from authoritative external sources (web, news, journal etc.). In particular, it states that *"articles should be based on reliable, third-party, published sources with a reputation for fact-checking and accuracy"*[1]. Not only are citations essential in maintaining reliability, neutrality and authoritative assessment of content in such a collaboratively edited platform; but lack of citations are essential signals for core editors for unreliability checks.



At the summit of the climb, carpet tacks[1] were thrown onto the road causing as many as thirty riders to puncture,[2][3] including Gilbert's teammates Cadel Evans and Steve Cummings,[39] while race leader Bradley Wiggins [...] precaution.[42] As a result, [...] and eventually soloed his way to a fourth career stage victory at the Tour.[47] Sagan led home a group of four riders almost a minute behind, [...] behind Sánchez.[39]

Figure 1: Sub-sentence level span for citation [1] in a citing paragraph in a Wikipedia article.

However, there are two problems when it comes to citing facts in Wikipedia. First, there is a long tail of Wikipedia pages where citations are missing and hence facts might be unverified. Second, citations might have different span *granularities*, i.e., the text encoding the fact(s), for which a citation is intended, might span less than a sentence (see Figure 1) to multiple sentences. We denote the different *pieces of text* which contain a citation marker as *fact statements* or simply *statements*. For example, Table 1 shows different *statements* for several citations. The aim of this work is to automatically and accurately determine *citation spans* in order to improve coverage (Fetahu et al., 2015b, 2016) and to assist editors in verifying citation quality at a fine-grained level.

Earlier work on span determination is mostly concerned with scientific texts (O'Connor, 1982; Kaplan et al., 2016), operates at sentence level and exploits explicit authoring cues specific to scientific text. Although Wikipedia has well formed text, it does not follow explicit scientific guidelines for placing citations. Moreover, most statements can only be inferred from the citation text.

In this work, we operate at a sub-sentence level, loosely referred to as text fragments, and take a sequence prediction approach using a *linear chain CRF* (Lafferty et al., 2001). We limit our work to citations referring to *web* and *news* sources, as

[1] https://en.wikipedia.org/wiki/Wikipedia:Identifying_reliable_sources

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1990–1999
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

they are accessible online and present the most prominent sources in Wikipedia (Fetahu et al., 2015a). By using recent work on moving window language models (Taneva and Weikum, 2013) and the structure of the paragraph that includes a citation, we classify sequences of text fragments as text that belong to a given citation. We are able to tackle all citation span cases as shown in Table 1.

`sub sentence`	Obama was born on August 4, 1961$^{[c_1]}$, at Kapi'olani Maternity $\cdots$ Honolulu$^{[c_2]}$; he is the first $\cdots$ been born in Hawaii.$^{[c_3]}$.
`sentence`	He was reelected to the Illinois Senate in 1998, $\cdots$ in 2002.$^{[c_1]}$
`multi sentence`	On May 25, 2011, Obama $\cdots$ to address $\cdots$ UK Parliament in Westminster Hall, London. This was $\cdots$ Charles de Gaulle $\cdots$ and Pope Benedict XVI.$^{[c_1]}$

Table 1: Varying degrees of citation span granularity in Wikipedia text.

2 Problem Definition and Terminology

In this section, we describe the terminology and define the problem of determining the *citation span* in text in Wikipedia articles.

Terminology. We consider Wikipedia articles $W = \{e_1, \ldots, e_n\}$ from a Wikipedia snapshot. We distinguish *citations* to *external references* in text and denote them with $\langle p_k, c_i \rangle$, where c_i represents a citation which occurs in paragraph p_k with positional index k in an entity $e \in W$. We will refer to p_k as the *citing paragraph*. Furthermore, with *citing sentence* we refer to the sentence in $s \in p_k$, which contains c_i. Note that p_k can have more than one citation as shown in Table 1.

Problem Definition. The task of determining the *citation span* for a citation c and a paragraph p, respectively $\langle p, c \rangle$ (or simply p_c), is subject to the citing paragraph and the citation content. In particular, we refer with *citation span* to the *textual fragments* from p which are covered by c. The fragments correspond to the sequence of *sub-sentences* $\mathcal{S}(p) = \langle \delta_1^1, \delta_1^2, \ldots, \delta_1^k, \ldots, \delta_n^m \rangle$. We obtain the sequence of sub-sentences from p by splitting the sentences into sub-sentences or text fragments based on the following punctuation delimiters $(\{,!;:?\})$. These delimitors do not always provide a perfect semantic segmentation of sentences into facts. A more involved approach could be taken akin to work in text summarization, such as Zhou and Hovy (Zhou and Hovy, 2006) or (Nenkova et al., 2007) who consider *summary units* for a similar purpose.

Formally, we define the *citation span* in Equation 4 as the function of finding the subset $\mathcal{S}' \subseteq \mathcal{S}$ where the fragments in $\mathcal{S}'$ are covered by c.

$$\varphi(p, c) \rightarrow \mathcal{S}' \subseteq \mathcal{S}, \quad s.t. \;\; \delta \in \mathcal{S}' \wedge c \vdash \delta \qquad (1)$$

where $c \vdash \delta$ states that δ is covered in c.

3 Related Work

Scientific Text. One of the first attempts to determine the citation span in text (O'Connor, 1982) was carried out in the context of document retrieval. The citing statements from a document were used as an index to retrieve the *cited* document. The citing statements are extracted based on heuristics starting from the citing sentence and are expanded with sentences in a window of +/-2 sentences, depending on them containing cue words like *'this', 'these',... 'above-mentioned'*. We consider the approach in (O'Connor, 1982) as a baseline.

Kaplan et al. (2016) proposed the task of determining the *citation block* based on a set of *textual coherence* features (e.g. grammatical or lexical coherence). The citation block *starts* from the citing sentence, with succeeding sentences classified (through SVMs or CRFs) according to whether they belong to the block. Abu-Jbara and Radev (2012) determine the citation block by first segmenting the sentences and then classifying individual words as being *inside/outside* the citation. Finally, the segment is classified depending on the word labels (majority of words being inside, at least one, or all of them). This approach is not applicable in our case due to the fact that words in Wikipedia text are not domain or genre-specific as one expects in scientific text, and as such their classification does not work.

Citations in IR. The importance of determining the citation span has been acknowledged in the field of Information Retrieval (IR). The focus is on building citation indexes (Garfield, 1955) and improving the retrieval of scientific articles (Ritchie et al., 2008, 2006). Citing sentences on a fixed window size are used to index documents and aid the retrieval process.

Summarization. Citations have been successfully employed to generate summaries of scientific articles (Qazvinian and Radev, 2008; Elkiss et al., 2008). In all cases, citing statements are either extracted manually or via heuristics such as extracting only citing sentences. Similarly (Nanba and Okumura, 1999) expand the summaries in addition to the citing sentence based on cue words (e.g. *'In this', 'However'* etc.). The work in (Qazvinian and Radev, 2010) goes one step beyond and considers sentences which do not *explicitly* cite another article. The task is to assign a binary label to a sentence, indicating whether it contains context for a cited paper. We use this approach as one of our competitors. Again, the premise is that citations are marked explicitly and additional citing sentences are found dependent on them.

Comparison to our work. The language style and the composition of citations in Wikipedia and in scientific text differ significantly. Citations are *explicit* in scientific text (e.g. *author names*) and are usually the first word in a sentence (Abu-Jbara and Radev, 2012). In Wikipedia, citations are *implicit* (see Table 1) and there are no cue words in text which link to the provided citations. Therefore, the proposed methodologies and features from the scientific domain do not perform optimally in our case.

Both (Qazvinian and Radev, 2010) and (O'Connor, 1982) work at the sentence level. As, in Wikipedia, citation span detection needs to be performed at the sub-sentence level (see Table 1), their method introduces erroneous spans as we will show in our evaluation.

Related to our problem is the work on addressing quotation attribution. Pareti et al. (2013) propose an approach for *direct* and *indirect* quotation attribution. The task is mostly based on lexical cues and specific *reporting verbs* that are the signal for the majority of direct quotations. However, in the case of quotation attribution the task is to find the *source, cue*, and *content* of the quotation, whereas in our case, for a given citing paragraph and reference we simply assess which text fragment is covered by the reference. We also do normally not have access to specific lexical links between the citation and its citation span.

4 Citation Span Approach

We approach the problem of citation span detection in Wikipedia as a *sequence classification* problem. For a citation c and a citing paragraph p, we chunk the paragraph into textual fragments at the *sub-sentence* granularity, shown in Equation 4.

Figure 2 shows an overview of the sequence classification of textual fragments. We use a *linear chain CRF* (Lafferty et al., 2001), where for any fragment δ we predict the label corresponding to a random variable $\mathbf{y}$ which is either *'covered'* or *'not-covered'*. We opt for CRFs since we can encode global dependencies between the text fragments and the actual citation, thus, ensuring the coherence and accuracy of the predicted labels.

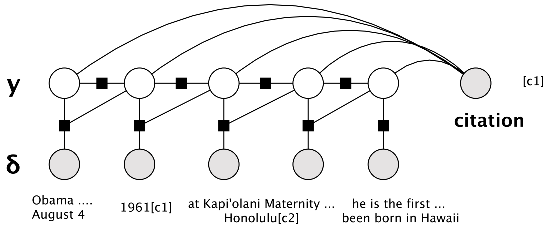


Figure 2: Linear chain CRF representing the sequence of text fragments in a paragraph. In the factors we encode the fitness to the given citation.

We now describe the features we compute for the factors $\Psi(y_i, y_{i-1}, \delta_i)$ for a fragment δ_i w.r.t the citation c. We determine the fitness of δ_i holding true or being covered by c. We denote with f_k the features for the factors $\Psi_i(y_i, y_{i-1}, \delta_i)$ for sequence δ_i for the linear-chain CRF in Figure 2.

4.1 Structural Features

An important aspect to consider for citation span detection is the structure of the citing paragraph, and correspondingly its sentences. For a textual fragment δ, we extract the following structural features shown in Table 2.

factor	description
$f_i^{c'}$	presence of other citations in δ_i where $c' \neq c$
$f^{\#s}$	the number of sentences in p
$f_i^{\lvert\delta_i\rvert}$	the length in terms of characters of the sub-sequence
f_i^{s}	check if δ_i is in the same sentence as the citation c
$f_i^{s \neq s'}$	check if δ_i is in the same sentence as δ_{i-1}
f_i^{c}	the distance of fragment δ_i to the fragment which contains citation c

Table 2: Structural features for a fragment δ_i.

From the features in Table 2, we highlight f_i^c which specifies the distance of δ to the fragment that cites c. The closer a fragment is to the citation the higher the likelihood of it being covered in c. In Wikipedia, depending on the citation and the paragraph length, the validity of a citation is densely concentrated in its nearby sub-sentences (preceding and succeeding).

Furthermore, the features $f^{\#s}$ and f_i^s (the number of sentences in p together with the feature considering if δ is in the same sentence as c) are strong indicators for accurate prediction of the label of δ. That is, it is more likely for a fragment δ to be covered by the citation if it appears in the same sentence or sentences nearby to the citation marker.

However, as shown in Table 1 there are three main citation span groups, and as such relying only on the structure of the citing paragraph does not yield optimal results. Hence, in the next group we consider features that tie the individual fragments in the citing paragraph with the citation as shown in Figure 2.

4.2 Citation Features

A core indicator as to whether a fragment δ is covered by c is based on the lexical similarity between δ and the content in c. We gather such evidence by computing two similarity measures. We compute the features f_i^{LM} and f_i^J between δ and paragraphs in the citation content c.

The first measure, f_i^{LM}, corresponds to a moving language window proposed in (Taneva and Weikum, 2013). In this case, for each word in either a paragraph in the citation c or the sequence δ, we associate a language model M_{w_i} based on its context $\phi(w_i) = \{w_{i-3}, w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}, w_{i+3}\}$ with a window of +/- 3 words. The parameters for the model M_{w_i} are estimated as in Equation 2 for all the words in the context $\phi(w_i)$ and their frequencies denoted with tf. With M_δ and M_p we denote the overall models as estimated in Equation 2 for the words in the respective fragments.

$$P(w|M_{w_i}) = \frac{tf_{w,\phi(w_i)}}{\sum_{w' \in \phi(w_i)} tf_{w',\phi(w_i)}} \tag{2}$$

Finally, we compute the similarity of each word in $w \in \delta$ against the language model of paragraph $p \in c$ in Equation 3, which corresponds to the Kullback-Leibler divergence score.

$$f_i^{LM} = \min_{p \in c} \left[- \sum_{w \in \delta} P(w|M_\delta) \log \frac{P(w|M_\delta)}{P(w|M_p)} \right] \tag{3}$$

The intuition behind f_i^{LM} is that for the fragments δ we take into account the word similarity and the similarity in the context they appear in w.r.t a paragraph in a citation. In this way, we ensure that the similarity is not by chance but is supported by the context in which the word appears. Finally, another advantage of this model is that we localize the paragraphs in c which provide evidence for δ.

As an additional feature we compute f_i^J which corresponds to the maximal *jaccard* similarity between δ_i and paragraphs $p \in c$.

Finally, as we will show in our experimental evaluation in Section 5, there is a high correlation between the citation span length and the length of citation content in terms of sentences. Hence, we add as an additional feature f^c the number of sentences in c.

4.3 Discourse Features

Sentences and fragments within a sentence can be tied together by discourse relations. We annotate sentences with explicit discourse relations based on an approach proposed in (Pitler and Nenkova, 2009), using discourse connectives as cues. The explicit discourse relations belong to one of the following: *temporal, contingency, expansion, comparison*.

After extracting a discourse connective in a sentence, we determine by its position to which fragment it belongs and mark the fragment accordingly. We denote with f_i^{disc} the discourse feature for the fragment δ_i.[2]

4.4 Temporal Features

An important aspect that we consider here is the temporal difference between two consecutive fragments δ_i and δ_{i-1}. If there exists a temporal date expression in δ_i and δ_{i-1} and they point to different time-points, this presents an indicator on the transitioning between the states y_i and y_{i-1}. That is, there is a higher likelihood of changing the state in the sequence $\mathcal{S}$ for the labels y_i and y_{i-1}.

We compute the temporal feature $f_i^{\lambda(i,i-1)}$,indicating the difference in *days* between any two temporal expression extracted from δ_i and δ_{i-1}. We extract the temporal expression through a set of hand-crafted regular expressions. We use the following expressions: (1) `DD Month YYYY`, (2) `DD MM YYYY`, (3)

[2] Note that, although discourse relations hold between at least two fragments or sentences, we only mark the individual fragment in which the connective occurs with the discourse relation type.

type	*avg.* $|s|$	*avg.* $|\delta|$	*avg 'covered'*
news	7.76	22.55	0.28
web	8.67	23.07	0.30

Table 3: Dataset statistics for citing paragraphs, distinguishing between *web* and *news* references, showing the average number of sentences, fragments, and covered fragments.

`MM DD YY(YY)`, (4) `YYYY`, with delimiters (whitespace, '-', '.').

5 Experimental Setup

We now outline the experimental setup for evaluating the citation span approach and the competitors for this task. The data and the proposed approaches are made available at the paper URL[3].

5.1 Dataset

We evaluate the citation span approaches on a random sample of Wikipedia entities (snapshot of 20/11/2016). For the sampling process, we first group entities based on the number of *web* or *news* citations.[4]). We then sample from the specific groups. This is due to the inherent differences in citation spans for entities with different numbers of citations. For instance, entities with a high number of citations tend to have shorter spans per citation. Figure 3 shows the distribution of entities from the different groups. From each sampled entity, we extract all *citing paragraphs* that contain either a *web* or *news* citation. Our sample consists of 509 citing paragraphs from 134 entities.

Furthermore, since a paragraph may have more than one citation, in our sampled citing paragraphs, we have an average of 4.4 citations per paragraph, which finally resulted in 408 unique paragraphs. Table 3 shows the stats of the dataset.

5.2 Ground Truth

Setup. For the ground truth, the citation span of c in paragraph p was manually determined by labeling each fragment in p with the binary label *covered* or *not-covered*.

We set strict guidelines that help us generate reliable ground-truth annotations. We follow two main guidelines: (i) requirement to read and comprehend the content in c, and (ii) matching of the textual fragments from p as either being supported *explicitly* or *implicitly* in c.[5]

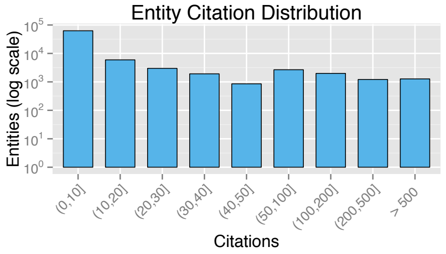


Figure 3: Entity distribution based on the number of news citations.

The entire dataset was carefully annotated by the first author. Later, a second annotator annotated a 10% sample of the dataset with an inter-rater agreement of $\kappa = .84$. We chose not to use crowd-sourcing as the task is very complex and hard to divide into small independent tasks. Since the task requires reading and comprehending the entire content in c and p, it takes on average up to 2.4 minutes to perform the evaluation for a single item. In future, it would be worthwhile to conduct more large-scale annotation exercises.

Citation Span Stats. Following the definition in Equation 4, we determine the citation span at the sub-sentence granularity level. Table 4 shows the distribution of citations falling into the specific spans for the citing paragraphs. We note that the majority of citations have a span between half a sentence and up to a sentence, yet, the remainder of more than 20% of citation span across multiple sentences in such paragraphs.

We define the citation span as the ratio of sub-sentences which are covered by a given citation over the total number of sub-sentences in the sentence, consequentially in the citing paragraph. That is, a citation is considered to have a span of one sentence if it covers all its sub-sentences.

$$span(c, p) = \sum_{s \in p} \frac{\#\delta^s \in \mathcal{S}'}{\#\delta^s} \tag{4}$$

where δ^s represents a sequence in sentence $s \in p$, which are part of the the ground-truth.

In Figure 4, we analyze a possible factor in the variance of the citation span. It is evident that for longer cited documents the span increases. This is

[3] `http://l3s.de/~fetahu/emnlp17/`

[4] Wikipedia has an internal categorization of citations based on the reference they point to.

[5] We excluded cases where the citation is not appropriate for the paragraph at all. This is, for example, the case when the language of c is not English.

	total	$\leq .5$	$(.5, 1]$	$(1, 2]$	$(2, 5]$	> 5
news	318	35	201	54	22	6
web	191	13	121	27	25	6

Table 4: Citation span distribution based on the number of sub-sentences in the citing paragraph.

intuitive since such documents carry more information and consequentially their span in the citing paragraphs can be larger. An example is the Wikipedia article `2008 US Open (tennis)` which has a citing paragraph with a citation span of 7 sentences for an article of 30k characters long[6]. We encoded this in the *citation* features f^c.

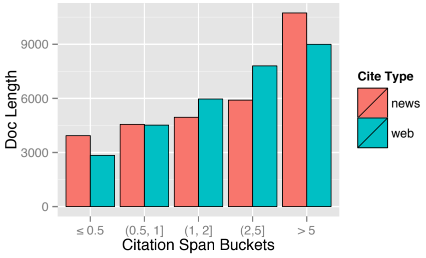


Figure 4: Average document length for the different span buckets for citation types *web* and *news*.

Additionally, within the different citation spans we analyze how many of them contain *skips* for two cases: (i) skip a fragment within a sentence, and (ii) skip sentences in p. The results for both cases are presented in Table 5.

span	*news*		*web*	
	skip δ	*skip* s	*skip* δ	*skip* s
≤ 0.5	6%	-	-	-
$(0.5, 1]$	-	-	-	1%
$(1, 2]$	-	8%	-	19%
$(2, 5]$	5%	18%	-	21%
> 5	-	20%	-	67%

Table 5: The percentage of citations in a span with *fragment skips* and *sentence skips*.

From the results in Table 4 and 5 we see that simple heuristics on selecting complete sentences or selecting consecutive sequences do not account for the different citation span cases and skips at the sentence and paragraph level. This leads to suboptimal results and introduces erroneous spans. Furthermore, we find that in 3.7% of the cases in our ground-truth, the citation spans include fragments after the citation marker.

5.3 Baselines

We consider the following baselines as competitors for our citation span approach.

Inter-Citation Text – IC. The span consists of sentences which start either at the beginning of the paragraph or at the end of a previous citation. The granularity is at the sentence level.

Citation-Sentence-Window – CSW. The span consists of sentences in a window of +/- 2 sentences from the citing sentence (O'Connor, 1982). The other sentences are included if they contain specific cue words in fixed positions.

Citing Sentence – CS. The span consists of only the *citing sentence*.

Markov Random Fields - MRF. MRFs (Qazvinian and Radev, 2010) model two functions. First, *compatibility*, which measures the similarity of sentences in p, and as such allows to extract non-citing sentences. Second, the *potential*, which measures the similarity between sentences in c with sentences in p. We use the provided implementation by the authors.

Citation Span Plain – CSPC. A plain classification setup using the features in Section 4, where the sequences are classified in isolation. We use Random Forests (Breiman, 2001) and evaluate them with 5-fold cross validation.

5.4 Citation Span Approach Setup – CSPS

For our approach $CSPS$ as mentioned in Section 4, we opt for linear-chain CRFs and use the implementation in (Okazaki, 2007). We evaluate our models using 5-fold cross validation, and learn the optimal parameters for the CRF model through the L-BFGS approach (Liu and Nocedal, 1989).

5.5 Evaluation Metrics

We measure the performance of the citation span approaches through the following metrics. We will denote with W' the sampled entities, with $\mathbf{p} = \{p_c, \ldots\}$ (p_c refers to $\langle p, c \rangle$) the set of sampled paragraphs from e, and with $|\mathbf{p}|$ the total items from e.

Mean Average Precision – MAP**.** First, we define *precision* for p_c as the ratio $P(p_c) = |\mathcal{S}' \cap \mathcal{S}^t| / |\mathcal{S}'|$ of fragments present in $\mathcal{S}' \cap \mathcal{S}^t$ over $\mathcal{S}'$. We measure MAP as in Equation 5.

$$MAP = \frac{1}{|W'|} \sum_{e \in W'} \frac{\sum_{p_c \in \mathbf{p}} P(p_c)}{|\mathbf{p}|} \qquad (5)$$

[6] `http://news.bbc.co.uk/sport1/hi/tennis/7601195.stm`

Recall – R. We measure the recall for p_c as the ratio $\mathcal{S}' \cap \mathcal{S}^t$ over all fragments in $\mathcal{S}^t$, $R(p_c) = |\mathcal{S}' \cap \mathcal{S}^t|/|\mathcal{S}^t|$. We average the individual recall scores for $e \in W'$ for the corresponding $\mathbf{p}$.

$$R = \frac{1}{|W'|} \sum_{e \in W'} \frac{\sum_{p_c \in \mathbf{p}} R(p_c)}{|\mathbf{p}|} \tag{6}$$

Erroneous Span – Δ. We measure the number of extra *words* or extra *sub-sentences* (denoted with Δ_w and Δ_δ) added by text fragments that are not part of the ground-truth $\mathcal{S}^t$. The ratio is relative to the number of words or sub-sentences in the ground-truth for p_c. We compute Δ_w and Δ_δ in Equation 7 and 8, respectively.

$$\Delta_w = \frac{1}{|W'|} \sum_{e \in W'} \frac{1}{|\mathbf{p}|} \sum_{p_c \in \mathbf{p}} \frac{\sum_{\delta \in \mathcal{S}' \setminus \mathcal{S}^t} words(\delta)}{\sum_{\delta \in \mathcal{S}^t} words(\delta)} \tag{7}$$

$$\Delta_\delta = \frac{1}{|W'|} \sum_{e \in W'} \frac{1}{|\mathbf{p}|} \sum_{p_c \in \mathbf{p}} \frac{|\mathcal{S}' \setminus \mathcal{S}^t|}{|\mathcal{S}^t|} \tag{8}$$

6 Results and Discussion

6.1 Citation Span Robustness

Table 6 shows the results for the different approaches on determining the citation span for all span cases shown in Table 4.

Accuracy. Not surprisingly, the baseline approaches perform reasonably well. CS which selects only the citing sentence achieves a reasonable $MAP = 0.86$ and similar recall. A slightly different baseline CSW achieves comparable scores with $MAP = 0.85$. This is due to the inherent span structure in Wikipedia, where a large portion of citations span up to a sentence (see Table 4). Therefore, in approximately 64% of the cases the baselines will select the correct span. For the cases where the span is more than a sentence, the drawback of these baselines is in coverage. We show in the next section a detailed decomposition of the results and highlight why even in the simpler cases, a sentence level granularity has its shortcomings due to sequence skips as shown in Table 5.

Overall, when comparing CS as the best performing baseline against our approach $CSPS$, we achieve an overall score of $MAP = 0.83$ (a slight decrease of 3.6%), whereas in term of F1 score, we have a decrease of 9%. The plain-classification approach $CSPC$ achieves similar score with $MAP = 0.86$, whereas in terms of F1 score, we have a decrease of 8%. As described above and as we will see later on in Table 7, the overall good performance of the baseline approaches can be attributed to the citation span distribution in our ground-truth.

On the other hand, an interesting observation is that sophisticated approaches, geared towards scientific domains like MRF perform poorly. We attribute this to *language style*, i.e., in Wikipedia there are no explicit citation hooks that are present in scientific articles. Comparing to $CSPS$, we outperform MRF by a large margin with an increase in MAP by 84%.

When comparing the sequence classifier $CSPS$ to the plain classifier $CSPC$, we see a marginal difference of 1.3% for *F1*. However, it will become more evident later that classifying jointly the text fragments for the different span buckets, outperforms the plain classification model.

	MAP	R	F1	Δ_w	Δ_δ
MRF	0.45	0.78	0.56	308%	278%
IC	0.72	**0.94**	0.77	113%	115%
CSW	0.85	0.84	**0.82**	38%	31%
CS	**0.86**	0.84	**0.82**	35%	27%
CSPC	**0.86**	0.68	0.76	**26**%	**23**%
CSPS	0.83	0.69	0.75	32%	24%

Table 6: Evaluation results for the different citation span approaches.

Erroneous Span. One of the major drawbacks of competing approaches is the granularity at which the span is determined. This leads to erroneous spans. From Table 4 we see that approximately in ∼10% of the cases the span is at sub-sentence level, and in 28% the span is more than a sentence.

The best performing baseline CS has an erroneous span of $\Delta_w = 35\%$ and $\Delta_\delta = 27\%$, in terms of extra words and sub-sentences, respectively. That is, nearly half of the determined span is erroneous, or in other words it is not covered in the provided citation. The MRF approach due to its poor MAP score provides the largest erroneous spans with $\Delta_w = 308\%$ and $\Delta_\delta = 278\%$. The amount of erroneous span is unevenly distributed, that is, in cases where the span is not at the sentence level granularity the amount of erroneous span increases. A detailed analysis is provided in the next section.

Contrary to the baselines, for $CSPS$ and similarly for $CSPC$, we achieve the lowest erroneous spans with $\Delta_w = 32\%$ and $\Delta_\delta = 26\%$, and $\Delta_w = 24\%$ and $\Delta_w = 23\%$, respectively.

Compared to the remaining baselines, we

achieve an overall relative decrease of 9% for $\Delta_w(CSPS)$, and 34% for $\Delta_w(CSPC)$, when compared to the best performing baseline CS.

From the *skips* in sequences in Table 5 and the unsuitability of sentence granularity for citation spans, we analyze the locality of erroneous spans w.r.t to the sequence that contains c, specifically the distribution of erroneous spans *preceding* and *succeeding* it. For the CS baseline, 71% of the total erroneous spans are added by sequences preceding the citing sequence, contrary to 35% which succeed it. In the case of $CSPS$, we have only 9% of erroneous spans (for Δ_δ) preceding the citation.

6.2 Citation Span and Feature Analysis

We now analyze how the approaches perform for the different citation spans in Table 4[7]. Additionally, we analyze how our approach $CSPS$ performs when determining the span without access to the content of c.

Citation Span. Table 7 shows the results for the approaches under comparison for all the citation span cases. In the case where the citation spans up to a sentence, that is $(0.5, 1]$, which presents the simplest citation span case, the baselines perform reasonably well. This is due to the heuristics they apply to determine the span, which in all cases includes the *citing sentence*. In terms of $F1$ score, the baseline CS achieves a highly competitive score of $F1 = 0.97$. Our approach $CSPS$ in this case has slight increase of 1% for $F1$ and an increase of 3% for MAP. $CSPC$ achieves a similar performance in this case.

However, for the cases where the span is at the sub-sentence level or across multiple sentences, the performance of baselines drops drastically. In the first bucket (≤ 0.5) which accounts for 9% of ground-truth data, we achieve the highest score with $MAP = 0.87$, though with lower recall than the competitors with $R = 0.56$. The reason for this is that the baselines take complete sentences, thus, having perfect recall at the cost of accuracy. In terms of $F1$ score we achieve 21% better results than the best performing baseline CS.

For the span of $(1, 2]$ we maintain an overall high accuracy and recall, and have the highest $F1$ score. The improvement is 8% in terms of $F1$ score. Finally, for the last case where the span is more than 2 sentences, we achieve $MAP = 0.74$, a marginal increase of 3%, however with lower recall, which results in an overall decrease of 4% for $F1$. The statistical significance tests are indicated with ** and * in Table 7.

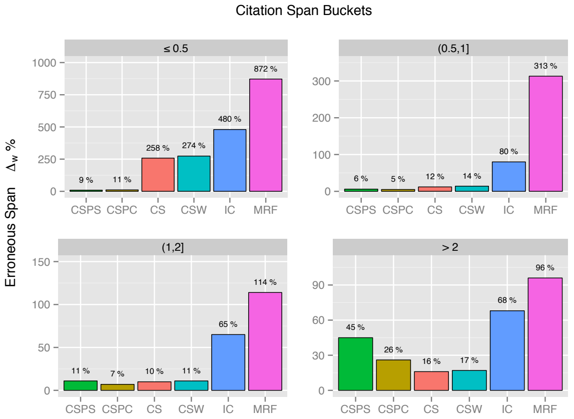


Figure 5: Erroneous spans for the different citation span buckets. The y-axis presents the Δ_w whereas in the x-axis are shown the different approaches.

Erroneous Span. Figure 5 shows the erroneous spans in terms of words for the metric Δ_w for all citation span cases. It is noteworthy that the amount of error can be well beyond 100% due to the ratio of the suggested span and the actual span in our ground-truth, which can be higher.

In the first bucket (span of ≤ 0.5) with granularity less than a sentence, all the competing approaches introduce large erroneous spans. For $CSPS$ we have a $MAP = 0.87$, and consequentially we have the lowest $\Delta_w = 9\%$, while for $CSPC$ we have only $\Delta_w = 11\%$. In contrast, the non-ML competitors introduce a minimum of $\Delta_w(CS) = 182\%$, with MRFs having the highest error. We also perform well in the bucket $(0.5, 1]$. For larger spans, for instance, for $(1, 2]$, we are still slightly better, with roughly 3% less erroneous span when comparing $CSPC$ and CS. However, only in the case of spans with > 2, we perform below the CS baseline. Despite, the smaller erroneous span, the CS baseline never includes more than one sentence, and as such it does not include many erroneous spans for the larger buckets. However, it is by definition unable to recognize any longer spans.

Feature Analysis. It is worthwhile to investigate the performance gains in determining the citation span without analyzing the content of the citation. The reason for this is that there are several cita-

[7] The models were retrained and tested for the different buckets with 5-fold cross validation.

		≤ 0.5			$(0.5, 1]$			$(1, 2]$			> 2	
	MAP	R	F1	MAP	R	F1	MAP	R	F1	MAP	R	F1
MRF	0.15	0.88	0.27	0.44	0.80	0.61	0.59	0.74	0.57	0.59	0.63	0.55
IC	0.32	**1.00**	0.45	0.77	**0.99**	0.83	0.73	**0.84**	0.74	0.72	**0.81**	**0.73**
CSW	0.38	**1.00**	0.54	0.93	0.98	0.96	0.88	0.54	0.65	0.79	0.34	0.43
CS	0.40	**1.00**	0.56	0.94	0.98	0.97	0.90	0.53	0.65	**0.80**	0.32	0.42
CSPC	0.85	0.53	0.65	**0.96**	0.97	0.97	**0.96**	0.68	0.79	0.71	0.65	0.68
CSPS	**0.87****	0.56	**0.68****	**0.96**	0.98	**0.98**	0.88	0.73	**0.80***	0.74	0.72	0.70
Δ_{F1} CSPS			▲21%			0%			▲8%			▼4%

Table 7: Evaluation results for the citation span approaches for the different span cases. For the results of $CSPS$ we compute the relative increase/decrease of $F1$ score compared to the best result (based on $F1$) from the competitors. We mark in bold the best results for the evaluation metrics, and indicate with ** and * the results which are highly significant ($p < 0.001$) and significant ($p < 0.05$) based on *t-test* statistics when compared to the best performing baselines (CS, IC, CSW, MRF) based on F1 score, respectively.

tion categories for which access to the source cannot be easily automated. Models which can determine the span accurately without the actual content have the advantage of generalizing to other citation sources (e.g. *books*) for which the evaluation is more challenging.[8]

Here, we disregard the citation features from Section 4.2. In terms of MAP, we have a slight decrease with $MAP = 0.82$ when compared to the model with the citation features. For recall we have a drop of 3%, resulting in $R = 0.67$.

This shows that by solely relying on the structure of the citing paragraph and other structural and discourse features we can perform the task with reasonable accuracy.

7 Conclusion

In this work, we tackled the problem of determining the fine-grained citation span of references in Wikipedia. We started from the *citing paragraph* and decomposed it into sequences consisting of sub-sentences. To accurately determine the span we proposed features that leverage the structure of the paragraph, discourse and temporal features, and finally analyzed the similarity between the citing paragraph and the citation content.

We introduce both a standard classifier as well as a sequence classifier using a linear-chain CRF model. For evaluation we manually annotated a ground-truth dataset of 509 citing paragraphs. We reported standard evaluation metrics and also introduced metrics that measure the amount of erroneous span.

We achieved a $MAP = 0.86$, in the case of the plain classification model $CSPC$, and with a marginal difference for $CSPS$ with $MAP = 0.83$, across all cases with an erroneous span of $\Delta_w = 26\%$ or $\Delta_w = 32\%$, depending on the model. Thus, we provide accurate means on determining the span and at the same time decrease the erroneous span by 34% compared to the best performing baselines. Moreover, we excel at determining citation spans at the sub-sentence level.

In conclusion, this presents an initial attempt on solving the citation span for references in Wikipedia. As future work we foresee a larger ground-truth and more robust approaches which take into account factors such as a reference being irrelevant to a citing paragraph and cases where the evidence for a paragraph is implied rather than explicitly stated in the reference.

Acknowledgments

This work is funded by the ERC Advanced Grant ALEXANDRIA (grant no. 339233), and H2020 AFEL project (grant no. 687916).

References

Amjad Abu-Jbara and Dragomir R. Radev. 2012. Reference scope identification in citing sentences. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 3-8, 2012, Montréal, Canada*, pages 80–90.

[8] At worst, one needs to read and comprehend the entire book to determine if a fragment is covered by the citation.

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Aaron Elkiss, Siwei Shen, Anthony Fader, Günes Erkan, David J. States, and Dragomir R. Radev. 2008. Blind men and elephants: What do citation summaries tell us about a research article? *JASIST*, 59(1):51–62.

Besnik Fetahu, Abhijit Anand, and Avishek Anand. 2015a. How much is wikipedia lagging behind news? In *Proceedings of the ACM Web Science Conference, WebSci 2015, Oxford, United Kingdom, June 28 - July 1, 2015*, pages 28:1–28:9.

Besnik Fetahu, Katja Markert, and Avishek Anand. 2015b. Automated news suggestions for populating wikipedia entity pages. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 323–332.

Besnik Fetahu, Katja Markert, Wolfgang Nejdl, and Avishek Anand. 2016. Finding news citations for wikipedia. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pages 337–346.

Eugene Garfield. 1955. Citation indexes for science: A new dimension in documentation through association of ideas. *Science*, 122(3159):108–111.

Dain Kaplan, Takenobu Tokunaga, and Simone Teufel. 2016. Citation block determination using textual coherence. *JIP*, 24(3):540–553.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528.

Hidetsugu Nanba and Manabu Okumura. 1999. Towards multi-paper summarization using reference information. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 926–931.

Ani Nenkova, Rebecca J. Passonneau, and Kathleen McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *TSLP*, 4(2):4.

John O'Connor. 1982. Citing statements: Computer recognition and use to improve retrieval. *Inf. Process. Manage.*, 18(3):125–131.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).

Silvia Pareti, Timothy O'Keefe, Ioannis Konstas, James R. Curran, and Irena Koprinska. 2013. Automatically detecting and attributing indirect quotations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 989–999.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore, Short Papers*, pages 13–16.

Vahed Qazvinian and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*, pages 689–696.

Vahed Qazvinian and Dragomir R. Radev. 2010. Identifying non-explicit citing sentences for citation-based summarization. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 555–564.

Anna Ritchie, Stephen Robertson, and Simone Teufel. 2008. Comparing citation contexts for information retrieval. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, pages 213–222.

Anna Ritchie, Simone Teufel, and Stephen Robertson. 2006. How to find better index terms through citations. In *Proceedings of the Workshop on How Can Computational Linguistics Improve Information Retrieval?*, CLIIR '06, pages 25–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bilyana Taneva and Gerhard Weikum. 2013. Gem-based entity-knowledge maintenance. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 149–158.

Liang Zhou and Eduard H Hovy. 2006. On the summarization of dynamically introduced information: Online discussions and blogs. In *AAAI Spring symposium: Computational approaches to analyzing weblogs*, page 237.

Identifying Semantic Edit Intentions from Revisions in Wikipedia

Diyi Yang♣, Aaron Halfaker♢, Robert Kraut♠, Eduard Hovy♣
♣ Language Technologies Institute, Carnegie Mellon University {diyi,hovy}@cmu.edu
♢ Wikimedia Foundation ahalfaker@wikimedia.org
♠ Human-Computer Interaction Institute, Carnegie Mellon University robert.kraut@cmu.edu

Abstract

Most studies on human editing focus merely on syntactic revision operations, failing to capture the intentions behind revision changes, which are essential for facilitating the single and collaborative writing process. In this work, we develop in collaboration with Wikipedia editors a 13-category taxonomy of the semantic intention behind edits in Wikipedia articles. Using labeled article edits, we build a computational classifier of intentions that achieved a micro-averaged F1 score of 0.621. We use this model to investigate edit intention effectiveness: how different types of edits predict the retention of newcomers and changes in the quality of articles, two key concerns for Wikipedia today. Our analysis shows that the types of edits that users make in their first session predict their subsequent survival as Wikipedia editors, and articles in different stages need different types of edits.

1 Introduction

Many online text production communities, including Wikipedia, maintain a history of revisions made by millions of participants. As Wikipedia statistics as of January 2017 show, English Wikipedia has 5.3 million articles with an average of 162.89 revisions per article, with revisions growing at a rate of about 2 revisions per second. This provides an amazing corpus for studying the types and effectiveness of revisions. Specifically, differences between revisions contain valuable information for modeling document quality or extracting users' expertise, and can additionally support various natural language processing (NLP) tasks such as sentence compression (Yamangil and Nelken, 2008), lexical simplification (Yatskar et al., 2010), information retrieval (Aji et al., 2010), textual entailment recognition (Zanzotto and Pennacchiotti, 2010), language bias detection (Recasens et al., 2013), spelling errors and paraphrases (Zesch, 2012; Max and Wisniewski, 2010).

To avoid building different approaches to extract the information needed by different NLP tasks (Ferschke et al., 2013), a unified framework to recognize edits from revisions is needed. Prior research on revision editing primarily develop syntactic edit action categories, from which they try to understand the effects of edits on meaning (Faigley and Witte, 1981; Yang et al., 2016). For instance, Daxenberger and Gurevych (2012) categorized edits based on whether edits affect the text meaning, resulting in syntactic edit categories such as file deletion, reference modification, etc. However, simply understanding the syntactic revision operation types does not provide the information we seek: *why* do editors do what they do? *how effective* are their actions? For example, syntactic edit type taxonomies cannot tell the difference between simplifying a paragraph and maliciously damaging that paragraph, since both involve deleting a sentence.

In this work, we focus explicitly on revision intention. We introduce a fine-grained taxonomy of the reasons why an author in Wikipedia made an edit. Example edit intentions include copy editing, elaboration, verification, and simplification. Compared to taxonomies that either focus on low-level syntactic operations (Faigley and Witte, 1981) or that mix syntactic and semantic classes (Daxenberger and Gurevych, 2013), a clean higher-level semantic categorization enables us to easily identify textual meaning changes, and to connect revisions to "what happens in the mind of the revising author during the revision"

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2000–2010
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

(Fitzgerald, 1987; Daxenberger, 2016). In order to capture the meaning behind edits, we worked with 13 Wikipedians to build a taxonomy that captured the meaning of an revision, which we term *edit intention*, and hand-labeled a corpus of 7,177 revisions with their edit intentions. We then developed an automated method to identify these edit intentions from differences between revisions of Wikipedia articles. To explore the utility of this taxonomy, we applied this model to better understand two important issues for Wikipedia: new editor retention and article quality. Specifically, we examined whether edit intentions in newcomers' first editing sessions predict their retention, and examined how edits with different intentions lead to changes in article quality. These analyses showed that specific types of editing work were positively correlated with newcomer survival and articles in different stages of development benefited differently from different types of edits.

2 Related Work

Wikipedia revision histories have been used for a wide range of NLP tasks (Yamangil and Nelken, 2008; Aji et al., 2010; Zanzotto and Pennacchiotti, 2010; Ganter and Strube, 2009; Nelken and Yamangil, 2008). For instance, Yatskar et al. (2010) used Wikipedia comments associated with revisions to collect relevant edits for sentence simplification. Max and Wisniewski (2010) constructed a corpus of rewritings that can be used for spelling errors and paraphrases (Zesch, 2012). Similarly, Zanzotto and Pennacchiotti (2010) used edits as training data for textual entailment recognition, and Recasens et al. (2013) analyzed real instances of human edits designed to remove bias from Wikipedia articles. Most of these work employed manually defined rules or filters to collect relevant edits to the NLP task at hand.

Towards analyzing revisions and developing unified revision taxonomies (Bronner and Monz, 2012; Liu and Ram, 2011), Fong and Biuk-Aghai (2010) built machine learning models to distinguish between factual and fluency edits in revision histories. Faigley and Witte (1981) made a distinction between changes that affect meaning, called *text-base changes* and changes which do not affect meaning, called *surface changes*. The two categories are further divided into formal changes, meaning-preserving changes, micro-structure changes and macro-structure changes. This taxonomy was later extended by Jones (2008) to take into account edit categories such as significant deletion, style, image insertion, revert, etc. Pfeil et al. (2006) proposed a 13-category taxonomy based on the data and performed manual annotation to compare cultural differences in the writing process in different versions of Wikipedia. Daxenberger and Gurevych (2013) introduced a finer-grained edit taxonomy, and performed multi-label classification to extract edit categories based on unparsed source text (Daxenberger and Gurevych, 2012). However, most taxonomies of edit categories contain only syntactic actions or a mixture of syntactic and semantic actions, failing to capturing the intention of revisions.

In terms of revision intentions, Zhang and Litman (2016) incorporated both argumentative writing features and surface changes from Faigley and Witte (1981) and constructed eight categories of revision *purposes*, such as claims/ideas, warrant/reasoning/backing, rebuttal/reservation, organization, clarify, etc. Tan and Lee (2014) used revisions to understand statement strength in academic writings. There are multiple works on the detection of specific subsets of revision intentions in Wikipedia, such as vandalism detection where the goal is to classify revisions as vandalized or non-vandalized (Harpalani et al., 2011; Adler et al., 2011) and language bias/neutral point of view detection (Recasens et al., 2013). Instead of recognizing a specific type of revision intention each time, our work aims at designing a systematic and comprehensive edit intention taxonomy to capture intentions behind textual changes.

Prior work also used edit types and intentions to better understand the process of collaborative writing, such as article quality improvement (Kittur and Kraut, 2008). For example, Liu and Ram (2011) found that Wikipedia article quality correlates with different types of contributors; similarly Yang et al. (2016) pointed out articles in different quality stages need different types of editors. However, there are few studies examining the specific types of edits that are predictive of article quality. Recent research shows that the number of active contributors in Wikipedia has been steadily declining since 2007, and Halfaker et al. (2012) suggested that the semi-automated rejection of new editors' contributions is a key cause, but they did not explore whether or not specific types of newcomers' work got rejected at different rates

Label	Description	α	Before	After
Clarification	Specify or explain an existing fact or meaning by example or discussion without adding new information	0.394	0.7%	4.1%
Copy Editing	Rephrase; improve grammar, spelling, tone, or punctuation	0.800	11.8%	14.8%
Counter Vandalism	Revert or otherwise; remove vandalism	0.879	1.9%	1.5%
Disambiguation	Relink from a disambiguation page to a specific page	0.401	0.3%	1.8%
Elaboration	Extend/add substantive new content; insert a fact or new meaningful assertion	0.733	12.0%	12.0%
Fact Update	Update numbers, dates, scores, episodes, status, etc. based on newly available information	0.744	5.5%	5.2%
Point of View	Rewrite using encyclopedic, neutral tone; remove bias; apply due weight	0.629	0.3%	2.2%
Process	Start/continue a wiki process workflow such as tagging an article with cleanup, merge or deletion notices	0.786	4.4%	5.8%
Refactoring	Restructure the article; move and rewrite content, without changing the meaning of it	0.737	1.9%	2.9%
Simplification	Reduce the complexity or breadth of discussion; may remove information	0.528	1.6%	4.6%
Vandalism	Deliberately attempt to damage the article	0.894	2.5%	2.0%
Verification	Add/modify references/citations; remove unverified text	0.797	5.4%	9.8%
Wikification	Format text to meet style guidelines, e.g. add links or remove them where necessary	0.664	33.1%	33.6%
Other	None of the above.	0.952	1.2%	-
Corpus Size		4,977	4,977	7,177

Table 1: A taxonomy of edit intentions in Wikipedia revisions, Cronbach's α agreement and the distributions of edit intention before and after corpus expansion. The percentage in each row represents what percentage of revisions are labeled with this edit intention. The percentages do not sum up to 100% because one revision could belong to multiple categories. The *After* corpus is used for all our analyses.

and how that affects retention. In this paper, we take advantage of this new taxonomy to explore correlations between edit intentions, newcomers' retention, and article quality.

3 Semantic Taxonomy of Edit Intentions

A *revision* is created whenever an editor saves changes to a Wikipedia page. As one revision could contain multiple local changes, each revision can be labeled with one or more edit intentions, representing the purposes of why an editor made that change. Different from prior research (Daxenberger, 2016; Yang et al., 2016), we do not distinguish between revisions and edits. Although an edit is a coherent local change and might belong to any edit categories, it cannot be used to represent the intentions of editors during the revision. For example, it might be difficult to recognize *Refactoring* if only one single edit is present. Since relocation or reorganization might involve several changes in the article, looking at one might lose the whole picture and lead to information loss. Moreover, edit types simply extracted from an edit is inadequate in outlining the correct intentions, for instance, adding a sentence could be *Clarification*, *Elaboration*, or *Vandalism*.

3.1 Taxonomy of Edit Intentions

Our semantic taxonomy of edit intentions builds on prior literature on collaborative writing (Faigley and Witte, 1981; Fitzgerald, 1987), research on document revision analyses (Bronner and Monz, 2012), studies on edit categories (Daxenberger and Gurevych, 2012; Fong and Biuk-Aghai, 2010), and work on purpose/intention classification (Zhang and Litman, 2016). In order to ensure that our taxonomy captured the *intentions* that Wikipedians would find meaningful, we set up discussions with a group of 12 interested editors on a Wikipedia project talk page, and iteratively refined our taxonomy based on their feedback. Our discussion with Wikipedia editors is in this page[1]. We also analyzed which intentions get more confused with which and used that to guide the refinement.

We define a top level layer for the revision intention taxonomy: intentions that are common in general revisions: **General Revision Intentions**, and intentions that are specific in Wikipedia: **Wikipedia Specific Intentions**. This categorization leads to 13 distinct semantic intentions, and Table 1 provides detailed descriptions. Specifically, general revision intentions include: *Clarification*, *Copy Editing*, *Elaboration*, *Fact Update*, *Point of View*, *Refactoring*, *Simplification* and *verification*, and can be applicable to other contexts. *Counter Vandalism*, *Disambiguation*, *Process*, *Vandalism*, and *Wikification* are edit intentions related to Wikipeida. We also propose an *Other* category, intended for edits that cannot be labeled using the above taxonomy.

As the first work to model intentions of revisions, our taxonomy distills and extends existing edit type taxonomies. For instance, our intentions of "elaboration" and "verification" are extensions of "evidence" type proposed by (Zhang and Litman, 2016), and a syntactic category of "information deletion" in (Daxenberger and Gurevych, 2013) could be an instance of our "vandalism" or "simplification" depending on the context.

3.2 Corpus Construction

To construct a reliable, hand-coded dataset to serve as ground truth for automatic recognition of edit intentions, we employed four undergraduate students who had basic Wikipedia editing experience to label edits using our intention taxonomy, based on written annotation guidelines[2] vetted by Wikipedia editors and provided examples[3]. Moreover, to expose annotators to more working knowledge of Wikipedia, we provided three one-hour training sessions where annotators were asked to label a small set of revisions (around 50 each time) and to discuss their disagreements until consensus.

We randomly sampled 5,000 revisions from Jan, 2016 to June 2016 from the recent changes table[4] in the Wikipedia database. For each revision, we displayed the content difference[5] before and after the change to annotators, via a labeling interface that we developed. Because an editor could make several different types of edits within a single revision, we asked four RAs to label each revision with one or more of the possible semantic intentions. We collected four valid annotations for 4,977 revisions. We used Cronbach's α, a measure of internal consistency, to evaluate agreement among the annotators. The overall agreement α score was 0.782, indicating substantial agreement between different annotators; The rule of thumb 1993 suggests that Cronbachs alpha scores larger than 0.7 are considered as acceptable. The inter-annotator agreement per semantic intention is described in column α in Table 1.

3.3 Corpus Expansion

As shown in column *Before* in Table 1, some types of edit intentions, such as *disambiguation* and *clarification*, were very rare in the random-sample corpus. To address this under-representation problem, we used the text of editors' comments to expand the corpus by retrieving 200 more revisions for each edit intention except Vandalism and Counter-Vandalism, resulting in 2,200 revisions[6]. More precisely, as a common practice (Zanzotto and Pennacchiotti, 2010; Recasens et al., 2013), we utilized regular expressions to match the text from the comments, which editors often wrote when saving their revisions, to the edit intentions. For example, editors might be signalling that they were intending to fix problems of *Point of View* when their comments contained keywords such as "npov" or "neutral". Even though the comments sometimes signal the editors' intents, they are not infallible, editors may fail to complete the comment field, may only label one of the multiple edit intentions for a single revision, or write comments that are inaccurate, irrelevant, or incomplete. Thus the first author annotated the 2,200 revisions from the expanded corpus and

[1] https://en.wikipedia.org/wiki/Wikipedia_talk:Labels/Edit_types/Taxonomy

[2] http://www.cs.cmu.edu/~diyiy/data/edit_intention_annotation_doc.pdf

[3] https://en.wikipedia.org/wiki/Wikipedia:Labels/Edit_types/Examples

[4] https://www.mediawiki.org/wiki/Manual:Recentchanges_table

[5] en.wikipedia.org/wiki/?diff=712140761

[6] We used a practical and economic way to expand the corpus, and this made the intention distribution skewed away. We acknowledge this expansion as a limitation.

merged it with the randomly sampled corpus. The frequency of the edit intentions before and after the expansion is in Table 1. We used the majority voting to resolve the disagreement. That is, if at least 3 out of 4 annotators picked an intention for a revision, it will be selected as the ground-truth. The final corpus contains 5,777 revisions, and can be downloaded from here[7].

4 Identification of Edit Intentions

We frame automated identification of edit intentions as a multi-label classification task. We designed four sets of features for identifying edit intentions from revisions. Set I comprised two features associated with the **Editor**: *user registration* indicating whether the editor of a particular revision was registered or anonymous and *tenure*, which refers to the elapsed months between the current revision and editors' registration date. Set II comprised 16 features associated with the **Comment** written by the editor to describe the revision, including *comment length* and a set of regular expressions to match intentions such as `*pov*`, `*clarify*`, `*simplif*`, `*add link*`, etc. Set III comprised 198 features associated with the **Revision Diff**, based on content differences between current revision and the previous one. They are similar to textual features defined in Daxenberger and Gurevych (2013), but we considered a wider range of objects being modified. In particular, we computed the difference in the number of characters, uppercase words, numeric chars, white-spaces, markups, Chinese/Japanese/Korean characters, HTML entity characters, URLs, punctuations, break characters, etc. We also considered languages features, such as the use of stop words, obscene words and informal words. Set IV comprises two features associated with **Vandalism** and **Revert**. We utilized the Wikipedia API to extract whether a revision was likely to be vandalism[8] or reverting revisions[9].

4.1 Identification Result

We extracted the input features with the help of Revision Scoring package[10] and framed this task a multi-label classification problem. For multi-label classification, we considered solving them by using single-label classification algorithms and by transforming it into one or more single-label classification tasks. We used the multi-label classifiers implemented in Mulan (Tsoumakas et al., 2011), with 10-fold cross validation. We utilized Binary Relevance (**BR**) to convert our multi-label classification into 13 binary single-label problems. Similar to Daxenberger and Gurevych (2013); Yang et al. (2016), we used Random k-labelsets **RAKEL** method that randomly chooses l small subset with k categories from the overall set of categories. We set l as 26, twice the size of the categories, and set k as 3. **MLKNN** method that classifies edit intentions based on K (K=10) nearest neighbor method. We used C4.5 decision tree classifiers in BR and RAKEL, as recommended by prior work (Daxenberger and Gurevych, 2013; Potthast et al., 2013). Prior research shows that sophisticated neural network models for text-classification largely rely on factors such as dataset size (Zhang et al., 2015; Joulin et al., 2016). Due to the size of our corpus and the complexity of this task, we did not use them.

To evaluate the relative accuracy of the multi-label classifier, we compared it to several baselines. The random baseline, denoted as **Random** in Table 2, assigns labels randomly. The majority category baseline, denoted as **Majority**, assigns all edits the most frequent intention, elaboration. Since revision comments may be especially as informative in reflecting edit intentions, the comment baseline, denoted as **CMT**, is a Binary Relevance classifier that includes only the comments features from Set II. We also created a Binary Relevance classifier, denoted as **BR-**, which excludes comment features and only used features from Sets I, III and IV.

Table 2 shows the evaluation metrics for the baselines and our multi-label classifiers. The metrics include the Exact Match subset accuracy, which evaluates whether the predicted labels are the same as the actual labels. These classifiers are available upon request. Table 2 also shows example-based measures of Accuracy, Precision, Recall and F1 Score, weighting each edit equally. It also shows label-based measures of accuracy – the micro- and macro-averaged F1 scores– which weight each edit intention category equally. As a ranking based measure, we measured One Error,

[7] http://www.cs.cmu.edu/~diyiy/data/edit_intention_dataset.csv

[8] https://ores.wmflabs.org/v2/scores/enwiki/goodfaith/71076450

[9] http://pythonhosted.org/mwreverts/api.html

[10] http://pythonhosted.org/revscoring/

Metric		Random	Majority	CMT	BR-	BR	MLKNN	RAKEL
Example	Exact Match	0.052	0.284	0.352	0.391	0.426	**0.452**	0.292
	Accuracy	0.052	0.283	0.428	0.498	0.540	**0.542**	0.338
	Precision	0.084	0.417	0.479	0.626	0.586	**0.599**	0.381
	Recall	0.052	0.285	0.458	0.562	**0.611**	0.578	0.344
	F1 Score	0.052	0.285	0.455	0.536	**0.580**	0.574	0.354
Label	Macro F1	0.060	0.042	0.310	0.487	**0.597**	0.576	0.385
	Micro F1	0.074	0.370	0.528	0.583	**0.621**	0.613	0.441
Ranking	One Error	0.920	0.583	0.415	0.400	0.358	**0.320**	0.434

Table 2: Performance comparison for predicting edit intentions from revisions. Best results are bold.

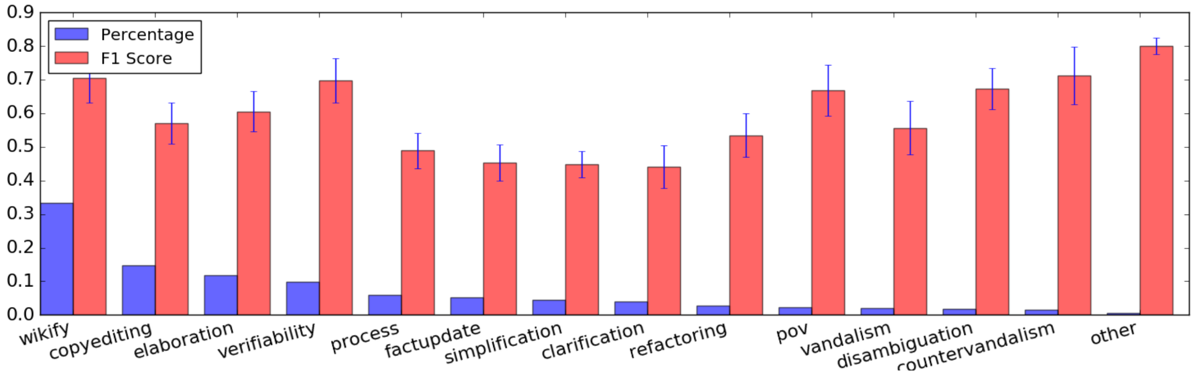


Figure 1: The relative frequency of each edit intention, and its F1 score provided by the **BR** model.

which evaluates how many times the top ranked predicted intention is not in the set of true labels of the instance.

Results show that the Binary Relevance (BR) and MLKNN classifiers, which used all our constructed features, outperformed Random and Majority baselines. Moreover, the BR and MLKNN methods show relatively similar best performances. Although multiple studies have utilized revisions' comments as "groundtruth" to collect desired edits, the CMT method, which includes only comment features, is less accurate than either the BR or MLKNN models. Note that predicting 14-category semantic intentions is more challenging compared to classifying low-level syntactic actions, such as inserting an image (Daxenberger and Gurevych, 2013).

5 Intentions, Survival and Quality

The automated measurement of edit intentions provides a general framework to analyze revisions and can facilitate a wide range of applications, such as collecting specific types of revisions (Yatskar et al., 2010; Recasens et al., 2013; Zanzotto and Pennacchiotti, 2010) and outlining the evolution of author roles (Arazy et al., 2015; Yang et al., 2016). In this section, we demonstrate two examples of how this intention taxonomy can be applied to better understand the success of online collaboration communities (Kraut et al., 2010), specifically the process of these sites to retain new contributors and create innovative products. To this end, we first investigate what newcomers are intended for in their first sessions and whether their edit intentions can account for their survival in Wikipedia. We then examine how edits carrying on different intentions at distinct times in an article's history influence changes in its quality.

5.1 How Edit Intentions Affect Survival

To explore newcomers' intentions during their first experience editing articles, we focus on users' first edit sessions in Wikipedia. Here, **Edit Session** is defined as a sequence of edits performed by a registered user with less than one hour's time gap between two adjacent edits (Halfaker et al., 2012). We then compare edit intentions of newcomers who survive - **Survivors**, and newcomers who do not - **Non-survivors**. Here, newcomers are defined as surviving if they performed an edit at

Edit Intention	***Intention Dist***		***Revert Ratio***	
	NS	**SS**	**NS**	**SS**
clarification	**0.2%**	**0.4%**	0.1%	0.1%
copy editing	**12.1%**	**14.4%**	**6.9%**	**3.8%**
counter vandalism	0.1%	0.0%	0.1%	0.0%
disambiguation	0.0%	0.0%	0.0%	0.0%
elaboration	27.7%	26.5%	**16.5%**	**6.9%**
fact update	4.2%	3.8%	**3.4%**	**1.7%**
point of view	**0.1%**	**0.2%**	0.0%	0.1%
process	2.0%	2.3%	**1.9%**	**0.7%**
refactoring	1.1%	1.3%	**0.9%**	**0.5%**
simplification	**3.7%**	**3.1%**	**3.1%**	**1.4%**
vandalism	**13.8%**	**6.1%**	**16.0%**	**4.7%**
verification	7.0%	7.4%	**3.8%**	**2.7%**
wikification	**25.8%**	**32.3%**	**14.0%**	**6.9%**

Table 3: The edit intention distribution in the first sessions (Intention Dist) and the revert ratio comparison (Revert Ratio), among non-survivors (NS) and survivors (SS). The numbers are bolded if 1-way ANOVA tests for difference between two groups are significant, with $p<0.05$.

Edit Intention	**Survival**	**Quality Changes**
clarification	0.029	0.001
copy editing	0.033	$0.011^\dagger$
counter vandalism	0.004	$-0.020^\dagger$
disambiguation	-0.003	$-0.006^\dagger$
elaboration	-0.024	$0.061^\dagger$
fact update	-0.001	0.002
point of view	0.041	-0.003
process	$0.051^\dagger$	$-0.024^\dagger$
refactoring	-0.013	$0.011^\dagger$
simplification	-0.002	$-0.008^\dagger$
vandalism	$-0.211^\dagger$	$-0.005^\dagger$
verification	0.047	$0.068^\dagger$
wikification	$0.099^\dagger$	$-0.010^\dagger$

Table 4: Regression coefficients of different edit intentions for predicting Newcomer **Survival** and Article **Quality Changes**. † means the coefficient is statistically significant ($p<0.05$)

least two months after their first edit session.

5.1.1 Intention Comparison

Among 100,000 randomly sampled Wikipedia users, 21,096 made revisions in the Main/Article namespace during their first editing session. Among these 4,407 were survivors (i.e., made an edit two months after registering) and 16,689 were non-survivors. We applied our edit intention model to 53,248 revisions in users' first sessions, and compared the percentages of different types of edit intentions between survivors and non-survivors, as shown in ***Intention Dist*** column in Table 3. We also performed 1-way ANOVA to test whether survivors and non-survivors have the same mean for each edit intention. We observed that, survivors tend to do more copy-editing (Δ_+=2.3%) and more wikification (Δ_+=6.5%), while non-survivors seem to perform more simplification and vandalism, which might provide signals for detecting vandals.

5.1.2 Revert Analysis

To explore the relationship between rejection of contributions and newcomer retention, we also visualized the revert ratios of different types of edit intentions for survivors and non-survivors in their first session. Here, **Revert** refers to whether an edit from the author was reverted or completely removed by another user, and we detect reverts using MediaWiki Reverts library[11]. We then measured the revert ratio for each edit intention by calculating the percentage of revisions belonging to a specific edit intention, among all reverted revisions in users' first sessions. As shown in the ***Revert Ratio*** column in Table 3, in general, non-survivors get reverted more compared to survivors, across all edit intentions. Interestingly, non-survivors compared to survivors get reverted more when performing *Wikification*, *verification* and *Refactoring*, suggesting that sophisticated types of work might not be suitable for beginners.

5.1.3 Newcomer Survival

As a further exploration of the relationship between edit intentions and newcomer survival, we performed a logistic regression using edits in survivors' and non-survivors' first sessions. To handle this imbalanced data (i.e., many more negative examples than positive examples in training), we performed majority-class under-sampling to make this dataset balanced. Similar to Halfaker et al. (2012), we controlled the number of revisions completed during the first session (a proxy for an editor's initial investment), and the number of revisions reverted in their first sessions. We

[11] `http://pythonhosted.org/mwreverts/`

described the regression coefficients of statistically significant edit intentions in the **Survival** column of Table 4. This logistic model achieves an Accuracy of 60.98%, Recall of 58.30%, Precision of 78.08% and F1-score of 66.76%. Editing articles for the purposes of *Process*, *Verification* and *Wikification* significantly predict the survival of newcomers, while performing vandalism is a strong negative predictor for survival.

5.2 How Intentions Affect Article Quality

Although there are over 5.5 million articles in the English Wikipedia, fewer than 0.2% have been evaluated by Wikipedians as good articles and around 92% have been evaluated as start or stub class articles, Wikipedia's two lowest quality categories. In this section, we examine how edits with different intentions at distinct times in an article's history influence changes in its quality.

This task is framed as a prediction task, i.e. using edits' intentions and a set of control variables to predict changes in article quality. We borrowed a Article Quality Prediction Dataset released in Yang et al. (2016), which consists of the quality ratings collected in January and June, 2015 of 151,452 articles. We collected 1,623,446 revisions made to these articles between January and June 2015, by randomly sampling 10% revisions that were made to these articles during that time periods. Specifically, the outcome *article quality change* is calculated by subtracting the previous quality score from the end quality score. The control variables include the previous article quality score, the total number of edits, the total number of editors, the changed bytes to an article, and the total number of edits to the article talk page during the six months. To construct edit-intention predictors, we summed the number of edits for each edit intention during the six months divided by the total number of revisions in this article.

Results of the linear regression model, shown in **Quality Changes** column of Table 4, show that our constructed regression model is significantly predictive of article quality changes ($R^2 = 0.225$). The results show that, keeping all control variables fixed, more *Copy Editing*, *Elaboration*, *Refactoring* and *Verification* are positively associated with improvements in article quality; in contrast, *Vandalism*, *Counter Vandalism*, *Disambiguation*, *Process* and *Simplification* predict declines in article quality. The first four of these edits types often occur with reducing the article content, removing or redirecting pages. Improper use of them might be detrimental to article quality.

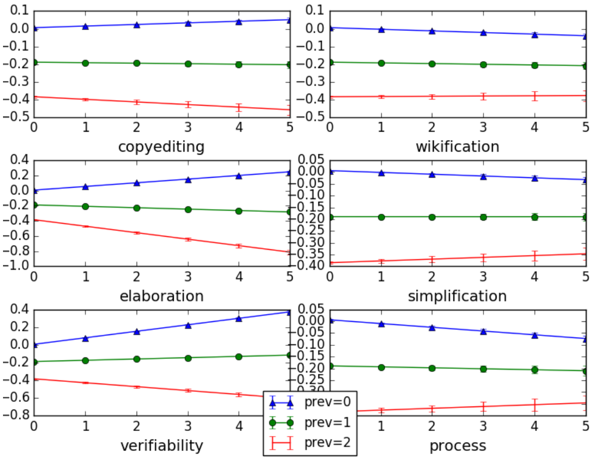


Figure 2: Interaction effect of different levels of edit intentions and different levels of previous article quality (**prev**) on article quality changes. All variables are standardized. The Y-axis measures the predictive margins and X-axis refers to different standardized levels of edit intention.

To determine if the effect of edit intentions on quality changes depends upon the initial quality of the article, we added the interaction terms between the previous quality score and edit percentages of different intentions (e.g., clarification x previous quality), and visualized interaction effects in Figure 2. When examining the interaction terms in more detail: the negative slope of *copy editing* (when prev=2) suggests that, as articles increase in quality, copy editing is needed less. We found similar trends for interactions between previous quality and *elaboration* and *verification*, which are essential for articles in the starting stages. In contrast, the positive slopes for *simplification*, *wikification* and *process* suggest that, as articles increase in quality, simplifying articles' content, adding proper links or reorganizing their structure becomes more important. Overall, these results reveal that different types of edit intentions are needed at different quality stages of articles.

6 Discussion and Conclusion

In this work, we proposed 13 semantic intentions that motivate editors' revisions in English Wikipedia. Example edit intentions include copy editing, elaboration, simplification, etc. Based in a labeled corpus of revisions, we developed machine-learning models to automatically identify these edit intentions. We then examine the relations between edit intentions, newcomers survival, and article quality improvement. We found that (1) survivors tend to do more copy editing and wikification; non-survivors seem to perform more vandalism and other sophisticated types of work, and the latter often gets reverted more; (2) Different types of contributions are needed by articles in different quality stages, with elaboration and verification are needed more for articles in the starting stages, and simplification and process become more important as article quality increases.

Our proposed edit intention taxonomy and the constructed corpus can facilitate a set of downstream NLP applications. First, classifiers based on this intention taxonomy can help retrieve large scale and high quality revisions around simplification, neutral point of view or copy editing, which provides amazing corpora for studying lexical simplification, language bias detection and paraphrases. Second, as we showed in Section 5.2, determining how different edit types influence changes in articles is of great use to better the causes of quality variance in collaborative writing, such as detecting quality flaws (Anderka et al., 2012) and providing insights on which specific aspects of an article needs improvement and what type of work should be performed. The ability to identify the need for editing, and specifically the types of editing work required, can greatly assist not only collaborative writing but also individual improvement of text. Moreover, even though our edit taxonomy is for English Wikipedia, it can be applied to other language versions of Wikipedia. We are now deploying the same edit intention taxonomy for Italian Wikipedia, and plan to apply it to other low resourced languages in Wikipedia. Finally, beyond the context of Wikipedia, similar taxonomies can be designed for analyzing the collaboration and interaction happened in other online contexts such as academic writing (e.g., Google Docs or ShareLatex, etc).

Acknowledgement

This research was supported in part by a grant from Google to Robert Kraut. The first author was supported by Carnegie Mellon Presidential Fellowship. The authors would like to thank our reviewers and Wikipedian for helpful feedback.

References

B Thomas Adler, Luca De Alfaro, Santiago M Mola-Velasco, Paolo Rosso, and Andrew G West. 2011. Wikipedia vandalism detection: Combining natural language, metadata, and reputation features. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 277–288.

Ablimit Aji, Yu Wang, Eugene Agichtein, and Evgeniy Gabrilovich. 2010. Using the past to score the present: Extending term weighting models through revision history analysis. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 629–638. ACM.

Maik Anderka, Benno Stein, and Nedim Lipka. 2012. Predicting quality flaws in user-generated content: the case of wikipedia. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 981–990. ACM.

Ofer Arazy, Felipe Ortega, Oded Nov, Lisa Yeo, and Adam Balila. 2015. Functional roles and career paths in wikipedia. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, pages 1092–1105.

Amit Bronner and Christof Monz. 2012. User edits classification using document revision histories. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL'12, pages 356–366.

Jose M Cortina. 1993. What is coefficient alpha? an examination of theory and applications. *Journal of applied psychology*, 78(1):98.

Johannes Daxenberger. 2016. *The Writing Process in Online Mass Collaboration: NLP-Supported Approaches to Analyzing Collaborative Revision and User Interaction*. Ph.D. thesis, Technische Universität.

Johannes Daxenberger and Iryna Gurevych. 2012. A corpus-based study of edit categories in featured and non-featured Wikipedia articles. In *Proceedings of COLING 2012*, pages 711–726, Mumbai, India. The COLING 2012 Organizing Committee.

Johannes Daxenberger and Iryna Gurevych. 2013. Automatically classifying edit categories in Wikipedia revisions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 578–589, Seattle, Washington, USA. Association for Computational Linguistics.

Lester Faigley and Stephen Witte. 1981. Analyzing revision. *College composition and communication*, 32(4):400–414.

Oliver Ferschke, Johannes Daxenberger, and Iryna Gurevych. 2013. A survey of nlp methods and resources for analyzing the collaborative writing process in wikipedia. In *The Peoples Web Meets NLP*, pages 121–160. Springer.

Jill Fitzgerald. 1987. Research on revision in writing. *Review of educational research*, 57(4):481–506.

Peter Kin-Fong Fong and Robert P. Biuk-Aghai. 2010. What did they do? deriving high-level edit histories in wikis. In *Proceedings of the 6th International Symposium on Wikis and Open Collaboration*, WikiSym '10, pages 2:1–2:10, New York, NY, USA. ACM.

Viola Ganter and Michael Strube. 2009. Finding hedges by chasing weasels: Hedge detection using wikipedia tags and shallow linguistic features. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ACL Short '09, pages 173–176.

Aaron Halfaker, R Stuart Geiger, Jonathan T Morgan, and John Riedl. 2012. The rise and decline of an open collaboration system: How wikipedia's reaction to popularity is causing its decline. *American Behavioral Scientist*, page 0002764212469365.

Manoj Harpalani, Michael Hart, Sandesh Singh, Rob Johnson, and Yejin Choi. 2011. Language of vandalism: Improving wikipedia vandalism detection via stylometric analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 83–88, Portland, Oregon, USA. Association for Computational Linguistics.

John Jones. 2008. Patterns of revision in online writing a study of wikipedia's featured articles. *Written Communication*, 25(2):262–289.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Aniket Kittur and Robert E. Kraut. 2008. Harnessing the wisdom of crowds in wikipedia: Quality through coordination. In *Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work*, CSCW '08, pages 37–46.

Robert Kraut, Moira Burke, John Riedl, and P Resnick. 2010. Dealing with newcomers. *Evidencebased Social Design Mining the Social Sciences to Build Online Communities*, 1:42.

Jun Liu and Sudha Ram. 2011. Who does what: Collaboration patterns in the wikipedia and their impact on article quality. *ACM Trans. Manage. Inf. Syst.*, 2(2):11:1–11:23.

Aurlien Max and Guillaume Wisniewski. 2010. Mining naturally-occurring corrections and paraphrases from wikipedias revision history. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Rani Nelken and Elif Yamangil. 2008. Mining wikipedia's article revision history for training computational linguistics algorithms. In *Proceedings of the AAAI Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy*, pages 31–36.

Ulrike Pfeil, Panayiotis Zaphiris, and Chee Siang Ang. 2006. Cultural differences in collaborative authoring of wikipedia. *Journal of Computer-Mediated Communication*, 12(1):88–113.

Martin Potthast, Matthias Hagen, Tim Gollub, Martin Tippmann, Johannes Kiesel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. 2013. Overview of the 5th international competition on plagiarism detection. In *CLEF Conference on Multilingual and Multimodal Information Access Evaluation*, pages 301–331. CELCT.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1650–1659, Sofia, Bulgaria. Association for Computational Linguistics.

Chenhao Tan and Lillian Lee. 2014. A corpus of sentence-level revisions in academic writing: A step towards understanding statement strength in communication. In *Proceedings of ACL (short paper)*.

Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. 2011. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414.

Elif Yamangil and Rani Nelken. 2008. Mining wikipedia revision histories for improving sentence compression. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 137–140. Association for Computational Linguistics.

Diyi Yang, Aaron Halfaker, Robert Kraut, and Eduard Hovy. 2016. Who did what: Editor role identification in wikipedia. In *Tenth International AAAI Conference on Web and Social Media*.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 365–368. Association for Computational Linguistics.

Fabio Massimo Zanzotto and Marco Pennacchiotti. 2010. Expanding textual entailment corpora from wikipedia using co-training. In *Proceedings of the COLING-Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, volume 128.

Torsten Zesch. 2012. Measuring contextual fitness using error contexts extracted from the wikipedia revision history. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 529–538, Avignon, France. Association for Computational Linguistics.

Fan Zhang and Diane Litman. 2016. Using context to predict the purpose of argumentative writing revisions. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1424–1430, San Diego, California. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Accurate Supervised and Semi-Supervised Machine Reading for Long Documents

Izzeddin Gur[*]
Department of Computer Science
University of California, Santa Barbara
izzeddingur@cs.ucsb.edu

Daniel Hewlett
Google Research
dhewlett@google.com

Alexandre Lacoste[†]
Element AI
alex.lacoste.shmu@gmail.com

Llion Jones
Google Research
llion@google.com

Abstract

We introduce a hierarchical architecture for machine reading capable of extracting precise information from long documents. The model divides the document into small, overlapping windows and encodes all windows in parallel with an RNN. It then attends over these window encodings, reducing them to a single encoding, which is decoded into an answer using a sequence decoder. This hierarchical approach allows the model to scale to longer documents without increasing the number of sequential steps. In a supervised setting, our model achieves state of the art accuracy of 76.8 on the WikiReading dataset. We also evaluate the model in a semi-supervised setting by downsampling the WikiReading training set to create increasingly smaller amounts of supervision, while leaving the full unlabeled document corpus to train a sequence autoencoder on document windows. We evaluate models that can reuse autoencoder states and outputs without fine-tuning their weights, allowing for more efficient training and inference.

1 Introduction

Recently, deep neural networks (DNNs) have provided promising results for a variety of reading comprehension and question answering tasks (Weston et al., 2014; Hermann et al., 2015; Rajpurkar et al., 2016), which require extracting precise information from documents conditioned on a query. While a basic sequence to sequence (seq2seq) model (Sutskever et al., 2014) can perform these tasks by encoding a question and document sequence and decoding an answer sequence (Hewlett et al., 2016), it has some disadvantages. The answer may be encountered early in the text and need to be stored across all the further recurrent steps, leading to forgetting or corruption; Attention can be added to the decoder to solve this problem (Hermann et al., 2015). Even with attention, approaches based on Recurrent Neural Networks (RNNs) require a number of sequential steps proportional to the document length to encode each document position. Hierarchical reading models address this problem by breaking the document into sentences (Choi et al., 2017). In this paper, we introduce a simpler hierarchical model that achieves state-of-the-art performance on our benchmark task without this linguistic structure, and use it as framework to explore semi-supervised learning for reading comprehension.

We first develop a hierarchical reader called Sliding-Window Encoder Attentive Reader (SWEAR) that circumvents the aforementioned bottlenecks of existing readers. SWEAR, illustrated in Figure 1, first encodes each question into a vector space representation. It then chunks each document into overlapping, fixed-length windows and, conditioned on the question representation, encodes each window in parallel. Inspired by recent attention mechanisms such as Hermann et al. (2015), SWEAR attends over the window representations and reduces them into a single vector for each document. Finally, the answer is decoded from this document vector. Our results show that SWEAR outperforms the previous state-of-the-art on the supervised WikiReading task (Hewlett et al., 2016), improving Mean F1 to 76.8 from the previous 75.6 (Choi et al., 2017).

While WikiReading is a large dataset with millions of labeled examples, many applications of machine reading have a much smaller number

[*] Work completed while interning at Google Research.
[†] Work completed while at Google Research.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2011–2020
Copenhagen, Denmark, September 7–11, 2017.

of labeled examples among a large set of unlabeled documents. To model this situation, we constructed a semi-supervised version of WikiReading by downsampling the labeled corpus into a variety of smaller subsets, while preserving the full unlabeled corpus (i.e., Wikipedia). To take advantage of the unlabeled data, we evaluated multiple methods of reusing unsupervised recurrent autoencoders in semi-supervised versions of SWEAR. Importantly, in these models we are able to reuse all the autoencoder parameters *without fine-tuning*, meaning the supervised phase only has to learn to condition the answer on the document and query. This allows for more efficient training and online operation: Documents can be encoded in a single pass offline and these encodings reused by all models, both during training and when answering queries. Our semi-supervised learning models achieve significantly better performance than supervised SWEAR on several subsets with different characteristics. The best-performing model reaches 66.5 with 1% of the WikiReading dataset, compared to the 2016 state of the art of 71.8 (with 100% of the dataset).

2 Problem Description

Following the recent progress on end-to-end supervised question answering (Hermann et al., 2015; Rajpurkar et al., 2016), we consider the general problem of predicting an answer A given a query-document pair (Q, D). We do not make the assumption that the answer should be present verbatim in the document.

2.1 Supervised Version

Given a document $D = \{d_1, d_2, \cdots, d_{N_D}\}$ and a query $Q = \{q_1, q_2, \cdots, q_{N_Q}\}$ as sequences of words, our task is to generate a new sequence of words that matches the correct answer $A = \{a_1, a_2, \cdots, a_{N_A}\}$. Because we do not assume that A is a subsequence of D, the answer may require blending information from multiple parts of the document, or may be precisely copied from a single location. Our proposed architecture supports both of these use cases.

The WikiReading dataset (Hewlett et al., 2016), which includes a mix of categorization and extraction tasks, is the largest dataset matching this problem description. In WikiReading, documents are Wikipedia articles, while queries and answers are Wikidata properties and values, respectively. Example Wikidata property-value pairs are `(place of birth, Paris)`, `(genre, Science Fiction)`. The dataset contains 18.58M instances divided into training, validation, and test with an 85/10/5 split. The answer is present verbatim in the document only 47.1% of the time, severely limiting models that label document spans, such as those developed for the popular SQUAD dataset (Rajpurkar et al., 2016).

2.2 Semi-Supervised Version

We also consider a semi-supervised version of the task, where an additional corpus of documents without labeled (Q, A) pairs is available. Taking advantage of the large size of the WikiReading dataset, we created a series of increasingly challenging semi-supervised problems with the following structure:

- **Unsupervised:** The entire document corpus (about 4M Wikipedia articles), with queries and answers removed.
- **Supervised:** Five smaller training sets created by sampling a random (1%, 0.5%, 0.1%) of the WikiReading training set, and taking (200, 100) random samples from each property in the original training set.

3 Supervised Model Architecture

We now present our model, called Sliding-Window Encoder Attentive Reader (SWEAR), shown in Figure 1, and describe its operation in a fully supervised setting. Given a (Q, D) pair, the model encodes Q into a vector space representation with a Recurrent Neural Network (RNN). The first layer of the model chunks the document D into overlapping, fixed-length windows and encodes all windows in parallel with an RNN conditioned on the question representation. The second layer attends over the window representations, reducing them into a single vector representing the latent answer. Finally, the answer sequence A is decoded from this vector using an RNN sequence decoder.

3.1 Preliminaries and Notation

Each word w comes from a vocabulary V and is associated with a vector e_w which constitutes the rows of an embedding matrix E. We denote by e^D, e^Q, and e^A the vector sequences corresponding to

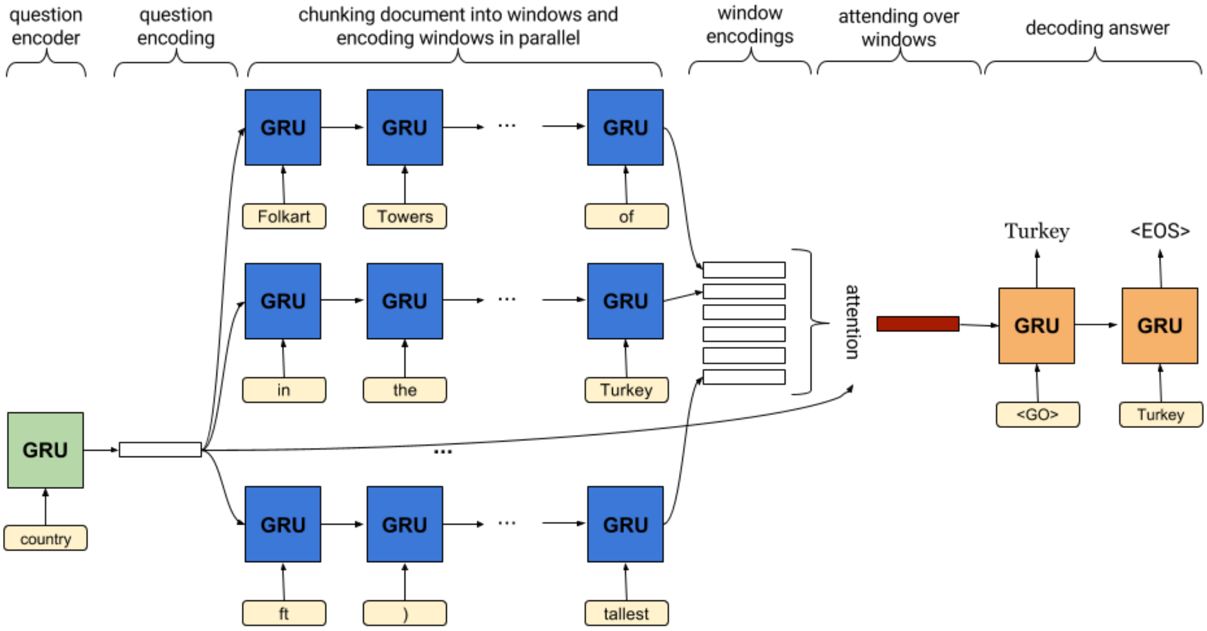


Figure 1: SWEAR model: Boxes are RNN cells, colors indicate parameter sharing.

the document, question, and answer sequences, respectively. More specifically, we aim at obtaining vector representations for documents and questions, then generating the words of the answer sequence.

Our model makes extensive use of RNN encoders to transform sequences into fixed length vectors. For our purposes, an RNN encoder consists of GRU units (Cho et al., 2014) defined as

$$h_t = f(x_t; h_{t-1}; \theta) \tag{1}$$

where h_t is hidden state at time t. f is a nonlinear function operating on input vector x_t and previous state, h_{t-1} with θ being its parameter vector. Given an input sequence, the encoder runs over the sequence of words producing the hidden vectors at each step. We refer to the last hidden state of an RNN encoder as the *encoding* of a sequence.

3.2 Sliding Window Recurrent Encoder

The core of the model is a sequence encoder that operates over sliding windows in a manner analogous to a traditional convolution. Before encoding the document, we slide a window of length l with a step size s over the document and produce $n = \lfloor \frac{N_D - l}{s} \rfloor$ document windows. This yields a sequence of sub-documents $(D_1, D_2, \cdots, D_n)$, where each D_i contains a subsequence of l words from the original document D. Intuitively, a precise answer may be present verbatim in one or more windows, or many windows may contain evidence suggestive of a more categorical answer.

Next, the model encodes each window conditioned on a question encoding. We first encode the question sequence once using a RNN (Enc) as

$$h^q = Enc(e^Q; \theta_Q) \tag{2}$$

where h^q is the last hidden state and θ_Q represents the parameters of the question encoder. Initialized with this question encoding, we employ another RNN to encode each document window as

$$\begin{aligned} h^w_{i,0} &= h^q \\ h^w_i &= Enc(e^{D_i}; \theta_W) \end{aligned} \tag{3}$$

where $h^w_{i,0}$ is the initial hidden state, h^w_i is the last hidden state, and θ_W represents the parameters of the window encoder. θ_W is shared for every window and is decoupled from θ_Q. As the windows are significantly smaller than the documents, encodings of windows will reflect the effect of question encodings better, mitigating any long-distance dependency problems.

3.3 Combining Window Encodings

SWEAR attends over the window encoder states using the question encoding to produce a single vector h^d for the document, given by

$$p_i \propto \exp(u_R^T \, tanh(W_R[h^w_i, h^q])) \tag{4}$$

$$h^d = \sum_i p_i h^w_i \tag{5}$$

Model	Mean F1
Placeholder seq2seq (HE16)	71.8
SoftAttend (CH17)	71.6
Reinforce (CH17)	74.5
Placeholder seq2seq (CH17)	75.6
SWEAR (w/ zeros)	76.4
SWEAR	**76.8**

Table 1: Results for SWEAR compared to top published results on the WikiReading test set.

	HE16 Best	SWEAR
Categorical	**88.6**	**88.6**
Relational	56.5	**63.4**
Date	73.8	**82.5**

Table 2: Mean F1 for SWEAR on each type of property compared with the best results for each type reported in Hewlett et al. (2016), which come from different models. Other publications did not report these sub-scores.

Doc length	pct	seq2seq	SWEAR	imp
$[0, 200)$	44.6	79.7	80.7	1.2
$[200, 400)$	19.5	76.7	77.8	1.5
$[400, 600)$	11.0	74.5	76.3	2.3
$[600, 800)$	6.6	72.8	74.3	2.1
$[800, 1000)$	4.3	71.5	72.8	1.8
$[1000, max)$	14.0	64.8	65.9	1.7

Table 3: Comparison of Mean F1 for SWEAR and a baseline seq2seq model on the WikiReading test set across different document length ranges. *pct* indicates the percentage of the dataset falling in the given document length range. *imp* is the percentage improvement of SWEAR over baseline.

where $[.]$ is vector concatenation, and p_i is the probability window i is relevant to answering the question. W_R and u_R are parameters of the attention model.

3.4 Answer Decoding

Given the document encoding h^d, an RNN decoder (Dec) generates the answer word sequence:

$$h_0^a = h^d$$

$$h_t^a = Dec(h_{t-1}^a; \omega_A) \quad (6)$$

$$P(a_t^* = w_j) \propto \exp(e_j^T (W_A h_t^a + b_A)) \quad (7)$$

$$a_t^* = argmax_j(P(a_t^* = w_j)) \quad (8)$$

where h_0^a is the initial hidden state and h_t^a is the hidden vector at time t. $A^* = \{a_1^*, a_2^*, \cdots, a_{N_A}^*\}$ is the sequence of answer words generated. W_A, b_A, and ω_A are the parameters of the answer decoder. The training objective is to minimize the average cross-entropy error between the candidate sequence A^* and the correct answer sequence A.

3.5 Supervised Results

Before exploring unsupervised pre-training, we present summary results for SWEAR in a fully supervised setting, for comparison to previous work on the WikiReading task, namely that of Hewlett et al. (2016) and Choi et al. (2017), which we refer to as HE16 and CH17 in tables. For further experiments, results, and discussion see Section 5.2. Table 1 shows that SWEAR outperforms the best results for various models reported in both publications, including the hierarchical models SoftAttend and Reinforce presented by Choi et al. (2017).[1] Interestingly, SoftAttend computes an attention over sentence encodings, analogous to SWEAR's attention over overlapping window encodings, but it does so on the basis of less powerful encoders (BoW or convolution vs RNN), suggesting that the extra computation spent by the RNN provides a meaningful boost to performance.

To quantify the effect of initializing the window encoder with the question state, we report results for two variants of SWEAR: In *SWEAR* the window encoder is initialized with the question encoding, while in *SWEAR w/ zeros*, the window encoder is initialized with zeros. In both cases the question encoding is used for attention over the window encodings. For SWEAR w/ zeros it is additionally concatenated with the document encoding and passed through a 2-layer fully connected neural network before the decoding step. Conditioning on the question increases Mean F1 by 0.4.

Hewlett et al. (2016) grouped properties by answer distribution: *Categorical* properties have a small list of possible answers, such as countries, *Relational* properties have an open set of answers, such as spouses or places of birth, and *Date* properties (a subset of relational properties) have date answers, such as date of birth. We reproduce this grouping in Table 2 to show that SWEAR improves performance for Relational and Date properties, demonstrating that it is better able to extract precise information from documents.

Finally, we observe that SWEAR outperforms a baseline seq2seq model on longer documents, as

[1] Document lengths differ between publications: We truncate documents to the first 600 words, while Choi et al. truncate to 1000 words or 35 sentences and Hewlett et al. truncate to 300 words.

shown in Table 3. The baseline model is roughly equivalent to the best previously-published result, Placeholder seq2seq (CH17) in Table 1, reaching a Mean F1 of 75.5 on the WikiReading testset. SWEAR improves over the baseline in every length category, but the differences are larger for longer documents.

4 Semi-Supervised Model Architecture

We now describe semi-supervised versions of the SWEAR model, to address the semi-supervised problem setting described in Section 2.2. A wide variety of approaches have been developed for semi-supervised learning with Neural Networks, with a typical scheme consisting of training an unsupervised model first, and then reusing the weights of that network as part of a supervised model. We consider each of these problems in turn, describing two types of unsupervised autoencoder models for sequences in Section 4.1 before turning to a series of strategies for incorporating the autoencoder weights into a final supervised model in Section 4.3. All of these models reuse the autoencoder weights without modification, meaning a document can be encoded once by an offline process, and the resulting encodings can be used both during training and to answer multiple queries online in a more efficient manner.

4.1 Recurrent Autoencoders for Unsupervised Pre-training

Autoencoders are models that reconstruct their input, typically by encoding it into a latent space and then decoding it back again. Autoencoders have recently proved useful for semi-supervised learning (Dai and Le, 2015; Fabius and van Amersfoort, 2014). We now describe two autoencoder models from the recent literature that we use for unsupervised learning. The Recurrent Autoencoder (RAE) is the natural application of the seq2seq framework (Sutskever et al., 2014) to autoencoding documents (Dai and Le, 2015): In seq2seq, an encoder RNN already produces a latent representation h_N, which is used to initialize a decoder RNN. In RAE, the output sequence is replaced with the input sequence, so learning minimizes the cross-entropy between the reconstructed input sequence and the original input sequence. Encoder and decoder cells share parameters θ_U.

4.1.1 Variational Recurrent Autoencoder

The Variational Recurrent Autoencoder (VRAE), introduced by Fabius et al. (2014), is a RAE with a variational Bayesian inference step where an unobserved latent random variable generates the sequential data. The encoder and decoder are exactly the same as RAE, but the latent state h_N is not directly passed to the decoder. Instead, it is used to estimate the parameters of a Gaussian distribution with a diagonal covariance matrix: The mean is given by $\mu_x = W_\mu h_N + b_\mu$ and the covariance by $\Sigma_x = W_\Sigma h_N + b_\Sigma$, where W_μ, W_Σ, b_μ, and b_Σ are new variational step parameters. The decoder is initialized with a single vector sampled from this distribution, $z_x \sim \mathcal{N}(z|\mu_x, \Sigma_x)$. For VRAE, the Kullback-Leibler divergence between trained Normal distribution and standard normal distribution, i.e., $KL(\mathcal{N}(\mu_x, \Sigma_x)|\mathcal{N}(0, I))$, is added to the loss.

4.1.2 Window Autoencoders

We take advantage of the SWEAR architecture by training autoencoders for text windows, as opposed to the standard document autoencoders. These autoencoders operate on the same sliding window subsequences as the supervised SWEAR model, autoencoding all subsequences independently and in parallel. This makes them easier to train as they only have to compress short sequences of text into a fixed-length representation. As the task of autoencoding is independent from our supervised problem, we refer to the generated encodings as *global encodings*.

4.2 Baseline: Initialization with Autoencoder Embeddings

Our baseline approach to reusing an unsupervised autoencoder in SWEAR is to initialize all embeddings with the pre-trained parameters and fix them. We call this model SWEAR-SS (for semi-supervised). The embedding matrix is fixed to the autoencoder embeddings. All other parameters are initialized randomly and trained as in the fully supervised version. We found that initializing the encoders and decoder with autoencoder weights hurts performance.

4.3 Reviewer Models

Unfortunately, this baseline approach to semi-supervised learning has significant disadvantages in our problem setting. Pre-trained RNN parameters are not fully exploited since we observed

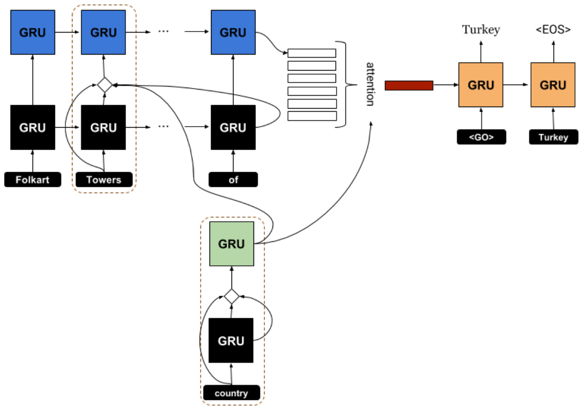


Figure 2: Multi-layer reviewer model (SWEAR-MLR), shown operating over a single window: Black boxes are RNN cells with fixed weights copied from the autoencoder, diamonds indicate vector concatenation with adapter and FC layers. For simplicity, only the cells in dashed boxes are fully illustrated (detailed in Figure 3), but the same structure is repeated for each cell.

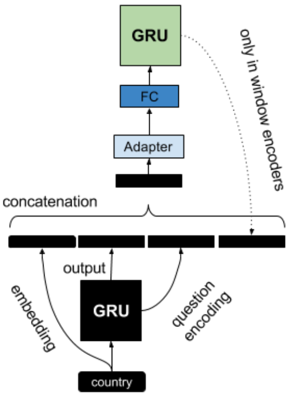


Figure 3: Detailed illustration of the dashed box in SWEAR-MLR question encoder. Black boxes are fixed parameters and encodings. The window encoder is similar, except that the output of the question reviewer layer is also added to the concatenated input (dashed line).

catastrophic forgetting when initializing and fine-tuning SWEAR with pre-trained weights. This includes fixing window encoder parameters with autoencoders and only fine-tuning question encoders. Second, conditioning the window encoders on the question eliminates the possibility to train window representations offline and utilize them later which causes a significant overhead during testing.

Inspired by recent trends in deep learning models such as Progressive Neural Networks (Rusu et al., 2016) and Reviewer Models (Yang et al., 2016), we propose multiple solutions to these problems. All of the proposed models process text input first through a fixed *autoencoder layer*: fixed pre-trained embeddings and fixed RNN encoder parameters, both initialized from the autoencoder weights. Above this autoencoder layer, we build layers of abstraction that learn to adapt the pre-trained models to the QA task.

4.3.1 Multi-Layer Reviewer (SWEAR-MLR)

The most straightforward extension to the baseline model is to fix the pretrained autoencoder RNN as the first layer and introduce a second, trainable *reviewer layer*. To make this approach more suitable for question answering, reviewer layers utilize corresponding global encodings as well as hidden states of the pre-trained autoencoders as input (Figure 2). The aim is to review both pre-trained question and window encodings to compose a single vector representing the window conditioned on the question.

Encoding questions: The question is first encoded by the autoencoder layer, $\tilde{h}^q = Enc(e^Q; \theta_U)$ where both word embeddings (E and e^Q) and encoder (θ_U) are fixed and initialized with pretrained parameters. A second, learnable RNN layer then takes the output of the autoencoder layer and corresponding input embeddings as input and produces the final question encoding, $h^q = Enc(FC([e^Q, \tilde{h}^q, \tilde{h}_t]); \theta_Q)$ where FC is a fully connected layer with ReLU activation function, and $\tilde{h}_t$ is the output of the autoencoder layer at time step t. Figure 3 illustrates a single time-step of the question encoder.

Encoding windows: Similarly, windows are encoded first by the fixed autoencoder layer and then by a reviewer layer, $\tilde{h}^w_i = Enc(e^{D_i}; \theta_U)$ and $h^w_i = Enc(FC([e^{D_i}, \tilde{h}^w_i, \tilde{h}^w_t, h^q]); \theta_W)$ where $\tilde{h}^w_t$ is the output of the autoencoder layer at time step t. Unlike supervised SWEAR, in SWEAR-MLR the window encoder is not initialized with the question encoder state. Instead, the question encoder state is an additional input to each unit in the reviewer layer (illustrated as the dashed line in Figure 3). Intuitively, the reviewer layer should reuse the global window and question information and encode only information relevant to the current question.

4.3.2 Progressive Reviewer (SWEAR-PR)

Although the reviewer layer in SWEAR-MLR has global window and question encodings as input, it requires a number of sequential steps equal to the window size, plus any additional reviewer steps. The reviewer layer also has to re-encode windows for each question, which is not ideal for online use.

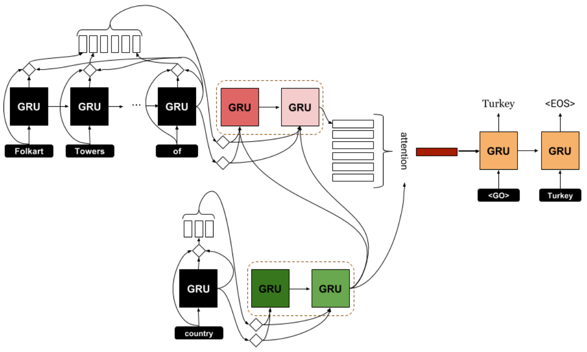


Figure 4: Progressive reviewer model (SWEAR-PR), shown operating over a single window: Black boxes are RNN cells with fixed weights copied from the autoencoder, diamonds indicate vector concatenation with adapter and FC layers. Dashed boxes contain reviewer layers. Cells within reviewer layers are decoupled as indicated by different colors.

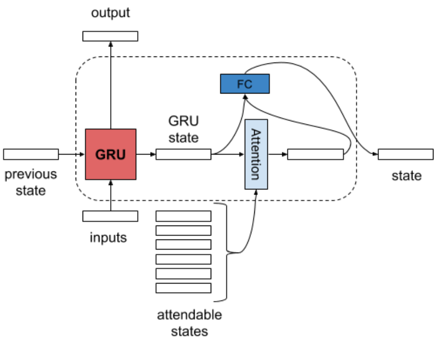


Figure 5: Illustration of attention cell: GRU state is used to attend over attendable states, then final state is computed by concatenating GRU state with context vector and passing through a fully connected neural network.

To address these issues, we now present a *Progressive Reviewer* model (SWEAR-PR) that reviews the outputs of the encoders using a separate RNN that is decoupled from the window size (Figure 4).

Encoding questions and windows: Similar to SWEAR-MLR, SWEAR-PR first encodes the questions and windows independently using autoencoder layers, $\tilde{h}^q = Enc(e^Q; \theta_U)$ and $\tilde{h}^w_i = Enc(e^{D_i}; \theta_U)$. To decouple the question and window encoders, however, SWEAR-PR does not have a second layer as a reviewer.

Reviewing questions and windows: SWEAR-PR employs two other RNNs to review the question and window encodings and to compose a single window representation conditioned on the question. Question reviewer takes the same pre-trained question encoding at each time step and attends over the hidden states and input embeddings of the pre-trained question encoder, $h^q = AttnEnc(FC(\tilde{h}^q); FC([\tilde{h}^q_t, e^Q]); \theta_Q)$ where $AttnEnc$ is an RNN with an attention cell which is illustrated in Figure 5. Outputs of the fixed autoencoder layer and fixed word embeddings, $[\tilde{h}^q_t, e^Q]$, are the attendable states. Window reviewer on the other hand takes the pre-trained window encoding and reviewed question encoding at each time step and attends over the hidden states of pre-trained window encoder, $h^w_i = AttnEnc(FC([\tilde{h}^w_i, h^q]); FC([\tilde{h}^w_t, e^{D_i}]); \theta_W)$ where outputs of the fixed autoencoder layer and fixed word embeddings, $[\tilde{h}^w_t, e^{D_i}]$, are the attendable states. As length of the windows is smaller than length of the reviewers, SWEAR-PR has significantly smaller overhead compared to other supervised and semi-supervised SWEAR variants.

4.3.3 Shared Components

Reducing window encodings and decoding: As in the supervised case described in 4, both reviewer models attend over the window encodings using the question encoding and reduce them into a single document encoding. Identical to answer decoding described in 6, the answer is decoded using another RNN taking the document state as the initial state. The parameters of this answer decoder are initialized randomly.

Adapter layer: As the distribution and scale of parameters may differ significantly between the autoencoder layer and the reviewer layer, we use an *adapter layer* similar to the adapters in Progressive Neural Networks (Rusu et al., 2016) to normalize the pre-trained parameters:

$$W_{out} = a * tanh(b * W_{in}) \tag{9}$$

where a and b are scalar variables to be learnt and W_{in} is a pre-trained input parameter. We put adapter layers after every pre-trained parameter connecting to a finetuned parameter such as on the connections from pre-trained embeddings to reviewer layer. We use dropout (Srivastava et al., 2014) regularization on both inputs and outputs of the reviewer cells.

5 Experimental Evaluation

As described in Section 2, we evaluate our models on the WikiReading task. In Section 3.5 we presented results for the supervised SWEAR on the full WikiReading dataset, establishing it as the highest-scoring method so far developed for WikiReading. We now compare our semi-supervised models SWEAR-MLR and SWEAR-

Model	1%	0.5%	0.1%
SWEAR	63.5	57.6	39.5
SWEAR-SS (RAE)	64.7	62.8	55.3
SWEAR-SS (VRAE)	**65.7**	**64.0**	**60.7**

Table 4: Mean F1 results for SWEAR (fully supervised) and SWEAR-SS (semi-supervised) trained on 1%, 0.5%, and 0.1% subsets, respectively. Variants of SWEAR-SS indicate different sources of fixed encoder weights. [2]

PR over various subsets of the WikiReading dataset, using SWEAR as a baseline.

5.1 Experimental Setup

Following Hewlett et al. (2016), we use the *Mean F1* metric for WikiReading, which assigns partial credit when there are multiple valid answers. We ran hyperparameter tuning for all models and report the result for the configuration with the highest Mean F1 on the validation set.

The supervised SWEAR model was trained on both the full training (results reported in Section 3.5) and on each subset of training data (results reported below). Unsupervised autoencoders were trained on all documents in the WikiReading training set. We selected the autoencoder with the lowest reconstruction error for use in semi-supervised experiments. After initialization with weights from the best autoencoder, learnable parameters in the semi-supervised models were trained exactly as in the supervised model.

Training Details

We implemented all models in a shared framework in TensorFlow (Abadi et al., 2016). We used the Adam optimizer (Kingma and Ba, 2014) for all training, periodically halving the learning rate according to a hyperparameter. Models were trained for a maximum of 4 epochs.

Table 7 shows which hyperparameters were tuned for each type of model, and the range of values for each hyperparameter. The parameters in the second group of the table are tuned for supervised SWEAR and the best setting (shown in bold) was used for other models where applicable. We fixed the batch size to 8 for autoencoders and 64 for semi-supervised models. We used a truncated normal distribution with a standard deviation of 0.01 for VRAE.

5.2 Results and Discussion

Model	100	200
SWEAR	25.0	33.0
SWEAR-SS (VRAE)	**39.0**	**45.0**

Table 5: Results for SWEAR and the best SWEAR-SS initialization (VRAE) trained on 100- and 200- per-property subsets, respectively.

Model	Mean F1
SWEAR-PR	**66.5**
dropout on input only	65.4
no dropout	64.6
shared reviewer cells	63.8
SWEAR-MLR	63.0
w/o skip connections	60.0

Table 6: Results for semi-supervised reviewer models trained on the 1% subset of WikiReading.

Table 4 and 5 show the results of SWEAR and semi-supervised models with pretrained and fixed embeddings. Results show that SWEAR-SS always improves over SWEAR at small data sizes, with the difference become dramatic as the dataset becomes very small. VRAE pretraining yields the best performance. As training and testing datasets have different distributions in per-property subsets, Mean F1 for supervised and semi-supervised models drops compared to uniform sampling. However, initialization with pretrained VRAE model leads to a substantial improvement on both subsamples. We further experimented by initializing the decoder (vs. only the encoder) with pretrained autoencoder weights but this resulted in a lower Mean F1.

Table 6 shows the results of semi-supervised reviewer models. When trained on 1% of the training data, SWEAR-MLR and the supervised SWEAR model perform similarly. Without using skip connections between embedding and hidden layers, the performance drops. The SWEAR-PR model further improves Mean F1 and outperforms the strongest SWEAR-SS model, even without fine-tuning the weights initialized from the autoencoder.

The success of SWEAR-PR rests on multiple design elements working together, as shown by the reduced performance caused by altering or disabling them. Using dropout only on the inputs, or not using any dropout on reviewer cells, causes a substantial decrease in Mean F1 score (by 1.1 and 1.9, respectively). Configuring the model with many more review steps (15) but with

[2] Initialization with Word2Vec (Mikolov et al., 2013) embeddings on 1% subset gives 64.0 Mean F1 score.

Parameter	**Space**
All Models	
Learning Rate	[0.0001; 0.005]
Learning Rate Decay Steps	{25k, 50k}
Gradient Clip	[0.01; 1.0]
Supervised & Semi-Supervised	
Embedding Size	{128, **256**, 512}
RNN State Size	{256, **512**}
Window Size	{10, 20, **30**}
Batch Size	{64, **128**}
Dropout	{0.3, **0.5**, 0.8}
Autoencoders Only	
Embedding Sharing	{Input, **All**}
KL-weight	[0.0001;0.01]
Semi-Supervised Only	
Finetune Pretrained Parameters	{YES, NO}
Dropout	{0.7, 0.8, **0.9**}
Reviewer State Size	{256, **512**}
Question Reviewer Steps	{0, 2, 3}
Window Reviewer Steps	{2, 3, 5, 8}

Table 7: Hyperparameter search spaces for each model type. We use {. . . } to denote a set of discrete values and [. . .] to denote a continuous range. Following Hewlett et al. (2016), we ran a random search over the possible configurations.

a smaller hidden vector size (128) reduced Mean F1 to 62.5. Increasing the number of review steps for the question to 5 caused a decrease in Mean F1 of 2.1.

6 Related Work

Our model architecture is one of many hierarchical models for documents proposed in the literature. The most similar is proposed by Choi et al. (2017), which uses a coarse-to-fine approach of first encoding each sentence with a cheap BoW or Conv model, then selecting the top k sentences to form a mini-document which is then processed by a standard seq2seq model. While they also evaluate their approach on WikiReading, their emphasis is on efficiency rather than model accuracy, with the resulting model performing slightly worse than the full seq2seq model but taking much less time to execute. SWEAR also requires fewer sequential steps than the document length but still computes at least as many recurrent steps in parallel.

Our model can also be viewed as containing a Memory Network (MemNet) built from a document (Weston et al., 2014; Sukhbaatar et al., 2015), where the memories are the window encodings. The core MemNet operation consists of attention over a set of vectors (memories) based on a query encoding, and then reduction of a second set of vectors by weighted sum based on the attention weights. In particular, Miller et al. (2016) introduce the Key-Value MemNet where the two sets of memories are computed from the keys and values of a map, respectively: In their QA task, each memory entry consists of a potential answer (the value) and its context bag of words (the key).

Our reviewer approach is inspired by “Encode, Review, Decode” approach introduced by Yang et al. (2016), which showed the value of introducing additional computation steps between the encoder and decoder in a seq2seq model.

The basic recurrent autoencoder was first introduced by Dai et al. (2015), a standard seq2seq model with the same input and output. Fabius et al. (2014) expanded this model into the Variational Recurrent Autoencoder (VRAE), which we describe in Section 4.1.1. VRAE is an application of the general idea of variational autoencoding, which applies variational approximation to the posterior to reconstruct the input (Kingma and Welling, 2013). While we train window autoencoders, an alternative approach is hierarchical document autoencoders (Li et al., 2015).

The semi-supervised approach of initializing the weights of an RNN encoder with those of a recurrent autoencoder was first studied by Dai et al. (2015) in the context of document classification and further studied by Ramachandran et al. (2016) for traditional sequence-to-sequence tasks such as machine translation. Our baseline semi-supervised model can be viewed as an extension of these approaches to a reading comprehension setting. Dai et al. (2015) also explore initialization from a language model, but find that the recurrent autoencoder is superior, which is why we do not consider language models in this work.

7 Conclusions

We have demonstrated the efficacy of the SWEAR architecture, reaching state of the art performance on supervised WikiReading. The model improves the extraction of precise information from long documents over the baseline seq2seq model. In a semi-supervised setting, our method of reusing (V)RAE encodings in a reading comprehension framework is effective, with SWEAR-PR reaching an accuracy of 66.5 on 1% of the dataset against last year’s state of the art of 71.8 using the full dataset. However, these methods require careful configuration and tuning to succeed, and making them more robust presents an excellent opportunity for future work.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Eunsol Choi, Daniel Hewlett, Alexandre Lacoste, Illia Polosukhin, Jakob Uszkoreit, and Jonathan Berant. 2017. Coarse-to-fine question answering for long document. In *Association for Computational Linguistics (ACL)*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3079–3087. Curran Associates, Inc.

Otto Fabius and Joost R van Amersfoort. 2014. Variational recurrent auto-encoders. *arxiv*.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. Wikireading: A novel large-scale language understanding task over wikipedia. In *Association for Computational Linguistics (ACL)*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arxiv*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Association for Computational Linguistics (ACL)*.

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

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arxiv*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Prajit Ramachandran, Peter J. Liu, and Quoc V. Le. 2016. Unsupervised pretraining for sequence to sequence learning. *CoRR*, abs/1611.02683.

A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. 2016. Progressive Neural Networks. *arxiv*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR*, abs/1410.3916.

Zhilin Yang, Ye Yuan, Yuexin Wu, Ruslan Salakhutdinov, and William W. Cohen. 2016. Encode, review, and decode: Reviewer module for caption generation. *arxiv*.

Adversarial Examples for Evaluating Reading Comprehension Systems

Robin Jia
Computer Science Department
Stanford University
robinjia@cs.stanford.edu

Percy Liang
Computer Science Department
Stanford University
pliang@cs.stanford.edu

Abstract

Standard accuracy metrics indicate that reading comprehension systems are making rapid progress, but the extent to which these systems truly understand language remains unclear. To reward systems with real language understanding abilities, we propose an adversarial evaluation scheme for the Stanford Question Answering Dataset (SQuAD). Our method tests whether systems can answer questions about paragraphs that contain adversarially inserted sentences, which are automatically generated to distract computer systems without changing the correct answer or misleading humans. In this adversarial setting, the accuracy of sixteen published models drops from an average of 75% F1 score to 36%; when the adversary is allowed to add ungrammatical sequences of words, average accuracy on four models decreases further to 7%. We hope our insights will motivate the development of new models that understand language more precisely.

1 Introduction

Quantifying the extent to which a computer system exhibits intelligent behavior is a longstanding problem in AI (Levesque, 2013). Today, the standard paradigm is to measure average error across a held-out test set. However, models can succeed in this paradigm by recognizing patterns that happen to be predictive on most of the test examples, while ignoring deeper, more difficult phenomena (Rimell et al., 2009; Paperno et al., 2016).

Article: Super Bowl 50
Paragraph: "*Peyton Manning became the first quarterback ever to lead two different teams to multiple Super Bowls. He is also the oldest quarterback ever to play in a Super Bowl at age 39. The past record was held by John Elway, who led the Broncos to victory in Super Bowl XXXIII at age 38 and is currently Denver's Executive Vice President of Football Operations and General Manager. Quarterback Jeff Dean had jersey number 37 in Champ Bowl XXXIV.*"
Question: "*What is the name of the quarterback who was 38 in Super Bowl XXXIII?*"
Original Prediction: John Elway
Prediction under adversary: Jeff Dean

Figure 1: An example from the SQuAD dataset. The BiDAF Ensemble model originally gets the answer correct, but is fooled by the addition of an adversarial distracting sentence (in blue).

In this work, we propose adversarial evaluation for NLP, in which systems are instead evaluated on adversarially-chosen inputs. We focus on the SQuAD reading comprehension task (Rajpurkar et al., 2016), in which systems answer questions about paragraphs from Wikipedia. Reading comprehension is an appealing testbed for adversarial evaluation, as existing models appear successful by standard average-case evaluation metrics: the current state-of-the-art system achieves 84.7% F1 score, while human performance is just 91.2%.[1] Nonetheless, it seems unlikely that existing systems possess true language understanding and reasoning capabilities.

Carrying out adversarial evaluation on SQuAD requires new methods that adversarially alter reading comprehension examples. Prior work in computer vision adds imperceptible adversarial perturbations to input images, relying on the fact that such small perturbations cannot change an image's true label (Szegedy et al., 2014; Goodfellow et al., 2015). In contrast, changing even one word of a

[1] https://rajpurkar.github.io/SQuAD-explorer/

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2021–2031
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

paragraph can drastically alter its meaning. Instead of relying on semantics-preserving perturbations, we create adversarial examples by adding distracting sentences to the input paragraph, as shown in Figure 1. We automatically generate these sentences so that they confuse models, but do not contradict the correct answer or confuse humans. For our main results, we use a simple set of rules to generate a raw distractor sentence that does not answer the question but looks related; we then fix grammatical errors via crowdsourcing. While adversarially perturbed images punish model *oversensitivity* to imperceptible noise, our adversarial examples target model *overstability*—the inability of a model to distinguish a sentence that actually answers the question from one that merely has words in common with it.

Our experiments demonstrate that no published open-source model is robust to the addition of adversarial sentences. Across sixteen such models, adding grammatical adversarial sentences reduces F1 score from an average of 75% to 36%. On a smaller set of four models, we run additional experiments in which the adversary adds non-grammatical sequences of English words, causing average F1 score to drop further to 7%. To encourage the development of new models that understand language more precisely, we have released all of our code and data publicly.

2 The SQuAD Task and Models

2.1 Task

The SQuAD dataset (Rajpurkar et al., 2016) contains 107,785 human-generated reading comprehension questions about Wikipedia articles. Each question refers to one paragraph of an article, and the corresponding answer is guaranteed to be a span in that paragraph.

2.2 Models

When developing and testing our methods, we focused on two published model architectures: BiDAF (Seo et al., 2016) and Match-LSTM (Wang and Jiang, 2016). Both are deep learning architectures that predict a probability distribution over the correct answer. Each model has a single and an ensemble version, yielding four systems in total.

We also validate our major findings on twelve other published models with publicly available test-time code: ReasoNet Single and Ensemble versions (Shen et al., 2017), Mnemonic Reader Single and Ensemble versions (Hu et al., 2017), Structural Embedding of Dependency Trees (SEDT) Single and Ensemble versions (Liu et al., 2017), jNet (Zhang et al., 2017), Ruminating Reader (Gong and Bowman, 2017), Multi-Perspective Context Matching (MPCM) Single version (Wang et al., 2016), RaSOR (Lee et al., 2017), Dynamic Chunk Reader (DCR) (Yu et al., 2016), and the Logistic Regression Baseline (Rajpurkar et al., 2016). We did not run these models during development, so they serve as a held-out set that validates the generality of our approach.

2.3 Standard Evaluation

Given a model f that takes in paragraph-question pairs (p, q) and outputs an answer $\hat{a}$, the *standard accuracy* over a test set D_{test} is simply

$$\text{Acc}(f) \overset{\text{def}}{=} \frac{1}{|D_{\text{test}}|} \sum_{(p,q,a)\in D_{\text{test}}} v((p, q, a), f),$$

where v is the F1 score between the true answer a and the predicted answer $f(p, q)$ (see Rajpurkar et al. (2016) for details).

3 Adversarial Evaluation

3.1 General Framework

A model that relies on superficial cues without understanding language can do well according to average F1 score, if these cues happen to be predictive most of the time. Weissenborn et al. (2017) argue that many SQuAD questions can be answered with heuristics based on type and keyword-matching. To determine whether existing models have learned much beyond such simple patterns, we introduce adversaries that confuse deficient models by altering test examples. Consider the example in Figure 1: the BiDAF Ensemble model originally gives the right answer, but gets confused when an adversarial distracting sentence is added to the paragraph.

We define an adversary A to be a function that takes in an example (p, q, a), optionally with a model f, and returns a new example (p', q', a'). The *adversarial accuracy* with respect to A is

$$\text{Adv}(f) \overset{\text{def}}{=} \frac{1}{|D_{\text{test}}|} \sum_{(p,q,a)\in D_{\text{test}}} v(A(p, q, a, f), f)).$$

While standard test error measures the fraction of the test distribution over which the model gets the correct answer, the adversarial accuracy measures

	Image Classification	Reading Comprehension
Possible Input		Tesla moved to the city of Chicago in 1880.
Similar Input		Tadakatsu moved to the city of Chicago in 1881.
Semantics	Same	Different
Model's Mistake	Considers the two to be different	Considers the two to be the same
Model Weakness	Overly sensitive	Overly stable

Table 1: Adversarial examples in computer vision exploit model oversensitivity to small perturbations. In contrast, our adversarial examples work because models do not realize that a small perturbation can completely change the meaning of a sentence. Images from Szegedy et al. (2014).

the fraction over which the model is *robustly* correct, even in the face of adversarially-chosen alterations. For this quantity to be meaningful, the adversary must satisfy two basic requirements: first, it should always generate (p', q', a') tuples that are *valid*—a human would judge a' as the correct answer to q' given p'. Second, (p', q', a') should be somehow "close" to the original example (p, q, a).

3.2 Semantics-preserving Adversaries

In image classification, adversarial examples are commonly generated by adding an imperceptible amount of noise to the input (Szegedy et al., 2014; Goodfellow et al., 2015). These perturbations do not change the semantics of the image, but they can change the predictions of models that are *oversensitive* to semantics-preserving changes. For language, the direct analogue would be to paraphrase the input (Madnani and Dorr, 2010). However, high-precision paraphrase generation is challenging, as most edits to a sentence do actually change its meaning.

3.3 Concatenative Adversaries

Instead of relying on paraphrasing, we use perturbations that do alter semantics to build *concatenative* adversaries, which generate examples of the form $(p + s, q, a)$ for some sentence s. In other words, concatenative adversaries add a new sentence to the end of the paragraph, and leave the question and answer unchanged. Valid adversarial examples are precisely those for which s does not contradict the correct answer; we refer to such sentences as being *compatible* with (p, q, a). We use semantics-altering perturbations to that ensure that s is compatible, even though it may have many words in common with the question q. Existing models are bad at distinguishing these sentences from sentences that do in fact address the question, indicating that they suffer not from oversensitivity but from *overstability* to semantics-altering edits. Table 1 summarizes this important distinction.

The decision to always append s to the end of p is somewhat arbitrary; we could also prepend it to the beginning, though this would violate the expectation of the first sentence being a topic sentence. Both are more likely to preserve the validity of the example than inserting s in the middle of p, which runs the risk of breaking coreference links.

Now, we describe two concrete concatenative adversaries, as well as two variants. AddSent, our main adversary, adds grammatical sentences that look similar to the question. In contrast, AddAny adds arbitrary sequences of English words, giving it more power to confuse models. Figure 2 illustrates these two main adversaries.

3.3.1 AddSent

AddSent uses a four-step procedure to generate sentences that look similar to the question, but do not actually contradict the correct answer. Refer to Figure 2 for an illustration of these steps.

In Step 1, we apply semantics-altering perturbations to the question, in order to guarantee that the resulting adversarial sentence is compatible. We replace nouns and adjectives with antonyms from WordNet (Fellbaum, 1998), and change named entities and numbers to the nearest word in GloVe word vector space[2] (Pennington et al., 2014) with the same part of speech.[3] If no words are changed during this step, the adversary gives up and immediately returns the original example. For example, given the question "*What ABC division handles domestic television distribution?*", we would change "*ABC*" to "*NBC*" (a nearby word in vector space) and "*domestic*" to "*foreign*" (a WordNet antonym), resulting in the question, "*What NBC division handles foreign television distribution?*"

In Step 2, we create a fake answer that has the same "type" as the original answer. We define a set

[2] We use 100-dimensional GloVe vectors trained on Wikipedia and Euclidean distance to define nearby words.

[3] We choose the nearest word whose most common gold POS tag in the Penn Treebank (Marcus et al., 1999) matches the predicted POS tag of the original word, according to CoreNLP. If none of the nearest 100 words satisfy this, we just return the single closest word.

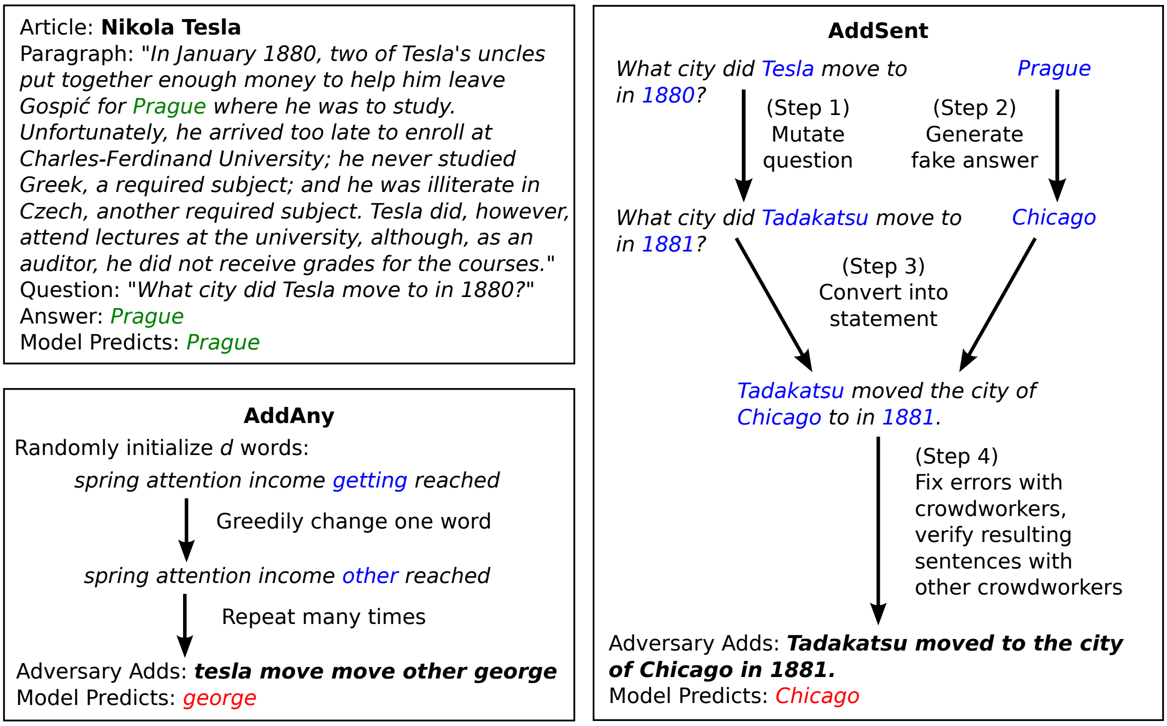

Article: Nikola Tesla
Paragraph: "In January 1880, two of Tesla's uncles put together enough money to help him leave Gospić for Prague where he was to study. Unfortunately, he arrived too late to enroll at Charles-Ferdinand University; he never studied Greek, a required subject; and he was illiterate in Czech, another required subject. Tesla did, however, attend lectures at the university, although, as an auditor, he did not receive grades for the courses."
Question: "What city did Tesla move to in 1880?"
Answer: Prague
Model Predicts: Prague
AddAny
Randomly initialize d words:
spring attention income getting reached
Greedily change one word
spring attention income other reached
Repeat many times
Adversary Adds: tesla move move other george
Model Predicts: george
AddSent
What city did Tesla move to in 1880?
Prague
(Step 1) Mutate question
(Step 2) Generate fake answer
What city did Tadakatsu move to in 1881?
Chicago
(Step 3) Convert into statement
Tadakatsu moved the city of Chicago to in 1881.
(Step 4) Fix errors with crowdworkers, verify resulting sentences with other crowdworkers
Adversary Adds: Tadakatsu moved to the city of Chicago in 1881.
Model Predicts: Chicago

Figure 2: An illustration of the AddSent and AddAny adversaries.

of 26 types, corresponding to NER and POS tags from Stanford CoreNLP (Manning et al., 2014), plus a few custom categories (e.g., abbreviations), and manually associate a fake answer with each type. Given the original answer to a question, we compute its type and return the corresponding fake answer. In our running example, the correct answer was not tagged as a named entity, and has the POS tag `NNP`, which corresponds to the fake answer "*Central Park.*"

In Step 3, we combine the altered question and fake answer into declarative form, using a set of roughly 50 manually-defined rules over CoreNLP constituency parses. For example, "*What ABC division handles domestic television distribution?*" triggers a rule that converts questions of the form "*what/which* $\texttt{NP}_1$ $\texttt{VP}_1$?" to "The $\texttt{NP}_1$ of `[Answer]` $\texttt{VP}_1$". After incorporating the alterations and fake answer from the previous steps, we generate the sentence, "*The NBC division of Central Park handles foreign television distribution.*"

The raw sentences generated by Step 3 can be ungrammatical or otherwise unnatural due to the incompleteness of our rules and errors in constituency parsing. Therefore, in Step 4, we fix errors in these sentences via crowdsourcing. Each sentence is edited independently by five workers on Amazon Mechanical Turk, resulting in up to five sentences for each raw sentence. Three additional crowdworkers then filter out sentences that are ungrammatical or incompatible, resulting in a smaller (possibly empty) set of human-approved sentences. The full AddSent adversary runs the model f as a black box on every human-approved sentence, and picks the one that makes the model give the worst answer. If there are no human-approved sentences, the adversary simply returns the original example.

A model-independent adversary. AddSent requires a small number of queries to the model under evaluation. To explore the possibility of an adversary that is completely model-independent, we also introduce AddOneSent, which adds a random human-approved sentence to the paragraph. In contrast with prior work in computer vision (Papernot et al., 2017; Narodytska and Kasiviswanathan, 2016; Moosavi-Dezfooli et al., 2017), AddOneSent does not require any access to the model or to any training data: it generates adversarial examples based solely on the intuition that existing models are overly stable.

3.3.2 AddAny

For AddAny, the goal is to choose any sequence of d words, regardless of grammaticality. We use local search to adversarially choose a distracting sentence $s = w_1 w_2 \dots w_d$. Figure 2 shows an example of AddAny with $d = 5$ words; in our experiments, we use $d = 10$.

We first initialize words $w_1, \dots, w_d$ randomly from a list of common English words.[4] Then, we run 6 epochs of local search, each of which iterates over the indices $i \in \{1, \dots, d\}$ in a random order. For each i, we randomly generate a set of candidate words W as the union of 20 randomly sampled common words and all words in q. For each $x \in W$, we generate the sentence with x in the i-th position and w_j in the j-th position for each $j \neq i$. We try adding each sentence to the paragraph and query the model for its predicted probability distribution over answers. We update w_i to be the x that minimizes the expected value of the F1 score over the model's output distribution. We return immediately if the model's argmax prediction has 0 F1 score. If we do not stop after 3 epochs, we randomly initialize 4 additional word sequences, and search over all of these random initializations in parallel.

AddAny requires significantly more model access than AddSent: not only does it query the model many times during the search process, but it also assumes that the model returns a probability distribution over answers, instead of just a single prediction. Without this assumption, we would have to rely on something like the F1 score of the argmax prediction, which is piecewise constant and therefore harder to optimize. "Probabilistic" query access is still weaker than access to gradients, as is common in computer vision (Szegedy et al., 2014; Goodfellow et al., 2015).

We do not do anything to ensure that the sentences generated by this search procedure do not contradict the original answer. In practice, the generated "sentences" are gibberish that use many question words but have no semantic content (see Figure 2 for an example).

Finally, we note that both AddSent and AddAny try to incorporate words from the question into their adversarial sentences. While this is an obvious way to draw the model's attention, we were curious if we could also distract the model without such a straightforward approach. To this end, we introduce a variant of AddAny called AddCommon, which is exactly like AddAny except it only adds common words.

	Match Single	Match Ens.	BiDAF Single	BiDAF Ens.
Original	71.4	75.4	75.5	80.0
AddSent	27.3	29.4	34.3	34.2
AddOneSent	39.0	41.8	45.7	46.9
AddAny	7.6	11.7	4.8	2.7
AddCommon	38.9	51.0	41.7	52.6

Table 2: Adversarial evaluation on the Match-LSTM and BiDAF systems. All four systems can be fooled by adversarial examples.

Model	Original	AddSent	AddOneSent
ReasoNet-E	**81.1**	39.4	49.8
SEDT-E	80.1	35.0	46.5
BiDAF-E	80.0	34.2	46.9
Mnemonic-E	79.1	**46.2**	**55.3**
Ruminating	78.8	37.4	47.7
jNet	78.6	37.9	47.0
Mnemonic-S	78.5	**46.6**	**56.0**
ReasoNet-S	78.2	39.4	50.3
MPCM-S	77.0	40.3	50.0
SEDT-S	76.9	33.9	44.8
RaSOR	76.2	39.5	49.5
BiDAF-S	75.5	34.3	45.7
Match-E	75.4	29.4	41.8
Match-S	71.4	27.3	39.0
DCR	69.3	37.8	45.1
Logistic	50.4	23.2	30.4

Table 3: AddSent and AddOneSent on all sixteen models, sorted by F1 score the original examples. S = single, E = ensemble.

4 Experiments

4.1 Setup

For all experiments, we measure adversarial F1 score (Rajpurkar et al., 2016) across 1000 randomly sampled examples from the SQuAD development set (the test set is not publicly available). Downsampling was helpful because AddAny and AddCommon can issue thousands of model queries per example, making them very slow. As the effect sizes we measure are large, this downsampling does not hurt statistical significance.

4.2 Main Experiments

Table 2 shows the performance of the Match-LSTM and BiDAF models against all four adversaries. Each model incurred a significant accuracy drop under every form of adversarial evaluation. AddSent made average F1 score across the four models fall from 75.7% to 31.3%. AddAny was even more effective, making average F1 score fall to 6.7%. AddOneSent retained much of the effectiveness of AddSent, despite being model-independent. Finally, AddCommon caused aver-

[4] We define common words as the 1000 most frequent words in the Brown corpus (Francis and Kucera, 1979).

	Human
Original	92.6
AddSent	79.5
AddOneSent	89.2

Table 4: Human evaulation on adversarial examples. Human accuracy drops on AddSent mostly due to unrelated errors; the AddOneSent numbers show that humans are robust to adversarial sentences.

age F1 score to fall to 46.1%, despite only adding common words.

We also verified that our adversaries were general enough to fool models that we did not use during development. We ran AddSent on twelve published models for which we found publicly available test-time code; we did not run AddAny on these models, as not all models exposed output distributions. As seen in Table 3, no model was robust to adversarial evaluation; across the sixteen total models tested, average F1 score fell from 75.4% to 36.4% under AddSent.

It is noteworthy that the Mnemonic Reader models (Hu et al., 2017) outperform the other models by about 6 F1 points. We hypothesize that Mnemonic Reader's self-alignment layer, which helps model long-distance relationships between parts of the paragraph, makes it better at locating all pieces of evidence that support the correct answer. Therefore, it can be more confident in the correct answer, compared to the fake answer inserted by the adversary.

4.3 Human Evaluation

To ensure our results are valid, we verified that humans are not also fooled by our adversarial examples. As AddAny requires too many model queries to run against humans, we focused on AddSent. We presented each original and adversarial paragraph-question pair to three crowdworkers, and asked them to select the correct answer by copy-and-pasting from the paragraph. We then took a majority vote over the three responses (if all three responses were different, we picked one at random). These results are shown in Table 4. On original examples, our humans are actually slightly better than the reported number of 91.2 F1 on the entire development set. On AddSent, human accuracy drops by 13.1 F1 points, much less than the computer systems.

Moreover, much of this decrease can be explained by mistakes unrelated to our adversarial sentences. Recall that AddSent picks the worst case over up to five different paragraph-question pairs. Even if we showed the same original example to five sets of three crowdworkers, chances are that at least one of the five groups would make a mistake, just because humans naturally err. Therefore, it is more meaningful to evaluate humans on AddOneSent, on which their accuracy drops by only 3.4 F1 points.

4.4 Analysis

Next, we sought to better understand the behavior of our four main models under adversarial evaluation. To highlight errors caused by the adversary, we focused on examples where the model originally predicted the (exact) correct answer. We divided this set into "model successes"—examples where the model continued being correct during adversarial evaluation—and "model failures"—examples where the model gave a wrong answer during adversarial evaluation.

4.4.1 Manual verification

First, we verified that the sentences added by AddSent are actually grammatical and compatible. We manually checked 100 randomly chosen BiDAF Ensemble failures. We found only one where the sentence could be interpreted as answering the question: in this case, AddSent replaced the word "*Muslim*" with the related word "*Islamic*", so the resulting adversarial sentence still contradicted the correct answer. Additionally, we found 7 minor grammar errors, such as subject-verb disagreement (e.g., "*The Alaskan Archipelago* ***are*** *made up almost entirely of hamsters.*") and misuse of function words (e.g., "*The gas* ***of*** *nitrogen makes up 21.8 % of* ***the*** *Mars's atmosphere.*"), but no errors that materially impeded understanding of the sentence.

We also verified compatibility for AddAny. We found no violations out of 100 randomly chosen BiDAF Ensemble failures.

4.4.2 Error analysis

Next, we wanted to understand what types of errors the models made on the AddSent examples. In 96.6% of model failures, the model predicted a span in the adversarial sentence. The lengths of the predicted answers were mostly similar to those of correct answers, but the BiDAF models occasionally predicted very long spans. The BiDAF Single model predicted an answer of more than

29 words—the length of the longest answer in the SQuAD development set—on 5.0% of model failures; for BiDAF Ensemble, this number was 1.6%. Since the BiDAF models independently predict the start and end positions of the answer, they can predict very long spans when the end pointer is influenced by the adversarial sentence, but the start pointer is not. Match-LSTM has a similar structure, but also has a hard-coded rule that stops it from predicting very long answers.

We also analyzed human failures—examples where the humans were correct originally, but wrong during adversarial evaluation. Humans predicted from the adversarial sentence on only 27.3% of these error cases, which confirms that many errors are normal mistakes unrelated to adversarial sentences.

4.4.3 Categorizing ADDSENT sentences

We then manually examined sentences generated by ADDSENT. In 100 BiDAF Ensemble failures, we found 75 cases where an entity name was changed in the adversarial sentence, 17 cases where numbers or dates were changed, and 33 cases where an antonym of a question word was used.[5] Additionally, 7 sentences had other miscellaneous perturbations made by crowdworkers during Step 4 of ADDSENT. For example, on a question about the "*Kalven Report*", the adversarial sentence discussed "*The statement Kalven cited*" instead; in another case, the question, "*How does Kenya curb corruption?*" was met by the unhelpful sentence, "*Tanzania is curbing corruption*" (the model simply answered, "*corruption*").

4.4.4 Reasons for model successes

Finally, we sought to understand the factors that influence whether the model will be robust to adversarial perturbations on a particular example. First, we found that models do well when the question has an exact n-gram match with the original paragraph. Figure 3 plots the fraction of examples for which an n-gram in the question appears verbatim in the original passage; this is much higher for model successes. For example, 41.5% of BiDAF Ensemble successes had a 4-gram in common with the original paragraph, compared to only 21.0% of model failures.

We also found that models succeeded more often on short questions. Figure 4 shows the distribution of question length on model successes and failures; successes tend to involve shorter questions. For example, 32.7% of the questions in BiDAF Ensemble successes were 8 words or shorter, compared to only 11.8% for model failures. This effect arises because ADDSENT always changes at least one word in the question. For long questions, changing one word leaves many others unchanged, so the adversarial sentence still has many words in common with the question. For short questions, changing one content word may be enough to make the adversarial sentence completely irrelevant.

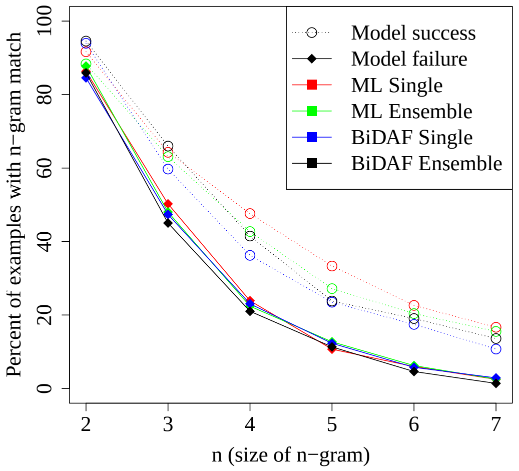


Figure 3: Fraction of model successes and failures on ADDSENT for which the question has an exact n-gram match with the original paragraph. For each model and each value of n, successes are more likely to have an n-gram match than failures.

Targeted Model	Model under Evaluation			
	ML Single	ML Ens.	BiDAF Single	BiDAF Ens.
ADDSENT				
ML Single	27.3	33.4	40.3	39.1
ML Ens.	31.6	29.4	40.2	38.7
BiDAF Single	32.7	34.8	34.3	37.4
BiDAF Ens.	32.7	34.2	38.3	34.2
ADDANY				
ML Single	7.6	54.1	57.1	60.9
ML Ens.	44.9	11.7	50.4	54.8
BiDAF Single	58.4	60.5	4.8	46.4
BiDAF Ens.	48.8	51.1	25.0	2.7

Table 5: Transferability of adversarial examples across models. Each row measures performance on adversarial examples generated to target one particular model; each column evaluates one (possibly different) model on these examples.

[5] These numbers add up to more than 100 because more than one word can be altered per example.

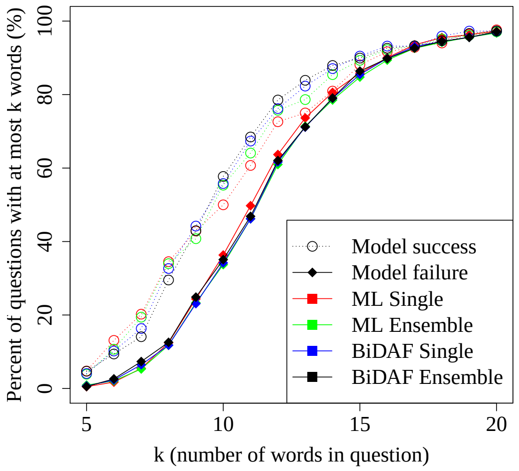


Figure 4: For model successes and failures on AddSent, the cumulative distribution function of the number of words in the question (for each k, what fraction of questions have $\leq k$ words). Successes are more likely to involve short questions.

4.5 Transferability across Models

In computer vision, adversarial examples that fool one model also tend to fool other models (Szegedy et al., 2014; Moosavi-Dezfooli et al., 2017); we investigate whether the same pattern holds for us. Examples from AddOneSent clearly do transfer across models, since AddOneSent always adds the same adversarial sentence regardless of model.

Table 5 shows the results of evaluating the four main models on adversarial examples generated by running either AddSent or AddAny against each model. AddSent adversarial examples transfer between models quite effectively; in particular, they are harder than AddOneSent examples, which implies that examples that fool one model are more likely to fool other models. The AddAny adversarial examples exhibited more limited transferability between models. For both AddSent and AddAny, examples transferred slightly better between single and ensemble versions of the same model.

4.6 Training on Adversarial Examples

Finally, we tried training on adversarial examples, to see if existing models can learn to become more robust. Due to the prohibitive cost of running AddSent or AddAny on the entire training set, we instead ran only Steps 1-3 of AddSent (everything except crowdsourcing) to generate a raw adversarial sentence for each training example. We then trained the BiDAF model from scratch on the union of these examples and the original training data. As a control, we also trained a second BiDAF model on the original training data alone.[6]

	Training data	
Test data	Original	Augmented
Original	75.8	75.1
AddSent	34.8	70.4
AddSentMod	34.3	39.2

Table 6: Effect of training the BiDAF Single model on the original training data alone (first column) versus augmenting the data with raw AddSent examples (second column).

The results of evaluating these models are shown in Table 6. At first glance, training on adversarial data seems effective, as it largely protects against AddSent. However, further investigation shows that training on these examples has only limited utility. To demonstrate this, we created a variant of AddSent called AddSentMod, which differs from AddSent in two ways: it uses a different set of fake answers (e.g., PERSON named entities map to "*Charles Babbage*" instead of "*Jeff Dean*"), and it prepends the adversarial sentence to the beginning of the paragraph instead of appending it to the end. The retrained model does almost as badly as the original one on AddSentMod, suggesting that it has just learned to ignore the last sentence and reject the fake answers that AddSent usually proposed. In order for training on adversarial examples to actually improve the model, more care must be taken to ensure that the model cannot overfit the adversary.

5 Discussion and Related Work

Despite appearing successful by standard evaluation metrics, existing machine learning systems for reading comprehension perform poorly under adversarial evaluation. Standard evaluation is overly lenient on models that rely on superficial cues. In contrast, adversarial evaluation reveals that existing models are overly stable to perturbations that alter semantics.

To optimize adversarial evaluation metrics, we may need new strategies for training models. For certain classes of models and adversaries, efficient training strategies exist: for example, Globerson and Roweis (2006) train classifiers that are optimally robust to adversarial feature deletion. Ad-

[6] All previous experiments used parameters released by Seo et al. (2016)

versarial training (Goodfellow et al., 2015) can be used for any model trained with stochastic gradient descent, but it requires generating new adversarial examples at every iteration; this is feasible for images, where fast gradient-based adversaries exist, but is infeasible for domains where only slower adversaries are available.

We contrast *adversarial evaluation*, as studied in this work, with *generative adversarial models*. While related in name, the two have very different goals. Generative adversarial models pit a generative model, whose goal is to generate realistic outputs, against a discriminative model, whose goal is to distinguish the generator's outputs from real data (Smith, 2012; Goodfellow et al., 2014). Bowman et al. (2016) and Li et al. (2017) used such a setup for sentence and dialogue generation, respectively. Our setup also involves a generator and a discriminator in an adversarial relationship; however, our discriminative system is tasked with finding the right answer, not distinguishing the generated examples from real ones, and our goal is to evaluate the discriminative system, not to train the generative one.

While we use adversaries as a way to evaluate language understanding, robustness to adversarial attacks may also be its own goal for tasks such as spam detection. Dalvi et al. (2004) formulated such tasks as a game between a classifier and an adversary, and analyzed optimal strategies for each player. Lowd and Meek (2005) described an efficient attack by which an adversary can reverse-engineer the weights of a linear classifier, in order to then generate adversarial inputs. In contrast with these methods, we do not make strong structural assumptions about our classifiers.

Other work has proposed harder test datasets for various tasks. Levesque (2013) proposed the Winograd Schema challenge, in which computers must resolve coreference resolution problems that were handcrafted to require extensive world knowledge. Paperno et al. (2016) constructed the LAMBADA dataset, which tests the ability of language models to handle long-range dependencies. Their method relies on the availability of a large initial dataset, from which they distill a difficult subset; such initial data may be unavailable for many tasks. Rimell et al. (2009) showed that dependency parsers that seem very accurate by standard metrics perform poorly on a subset of the test data that has unbounded dependency constructions. Such evaluation schemes can only test models on phenomena that are moderately frequent in the test distribution; by perturbing test examples, we can introduce out-of-distribution phenomena while still leveraging prior data collection efforts.

While concatenative adversaries are well-suited to reading comprehension, other adversarial methods may prove more effective on other tasks. As discussed previously, paraphrase generation systems (Madnani and Dorr, 2010) could be used for adversarial evaluation on a wide range of language tasks. Building on our intuition that existing models are overly stable, we could apply meaning-altering perturbations to inputs on tasks like machine translation, and adversarially choose ones for which the model's output does *not* change. We could also adversarially generate new examples by combining multiple existing ones, in the spirit of Data Recombination (Jia and Liang, 2016). The Build It, Break It shared task (Bender et al., 2017) encourages researchers to adversarially design minimal pairs to fool sentiment analysis and semantic role labeling systems.

Progress on building systems that truly understand language is only possible if our evaluation metrics can distinguish real intelligent behavior from shallow pattern matching. To this end, we have released scripts to run ADDSENT on any SQuAD system, as well as code for ADDANY. We hope that our work will motivate the development of more sophisticated models that understand language at a deeper level.

Acknowledgments. We thank Pranav Rajpurkar for his help with various SQuAD models. This work was supported by the NSF Graduate Research Fellowship under Grant No. DGE-114747, and funding from Facebook AI Research and Microsoft.

Reproducibility. All code, data, and experiments for this paper are available on the CodaLab platform at https://worksheets.codalab.org/worksheets/0xc86d3ebe69a3427d91f9aaa63f7d1e7d/.

References

E. M. Bender, H. Daume III, A. Ettinger, H. Kannan, S. Rao, and E. Rothschild. 2017. Build it, break it: The language edition. https://bibinlp.umiacs.umd.edu/.

S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai,

R. Jozefowicz, and S. Bengio. 2016. Generating sentences from a continuous space. In *Computational Natural Language Learning (CoNLL)*.

N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. 2004. Adversarial classification. In *International Conference on Knowledge Discovery and Data Mining (KDD)*.

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

W. N. Francis and H. Kucera. 1979. *Brown Corpus Manual*.

A. Globerson and S. Roweis. 2006. Nightmare at test time: robust learning by feature deletion. In *International Conference on Machine Learning (ICML)*, pages 353–360.

Y. Gong and S. R. Bowman. 2017. Ruminating reader: Reasoning with gated multi-hop attention. *arXiv*.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*.

I. J. Goodfellow, J. Shlens, and C. Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*.

M. Hu, Y. Peng, and X. Qiu. 2017. Mnemonic reader for machine comprehension. *arXiv*.

R. Jia and P. Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*.

K. Lee, S. Salant, T. Kwiatkowski, A. Parikh, D. Das, and J. Berant. 2017. Learning recurrent span representations for extractive question answering. *arXiv*.

H. J. Levesque. 2013. On our best behaviour. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

J. Li, W. Monroe, T. Shi, A. Ritter, and D. Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

R. Liu, J. Hu, W. Wei, Z. Yang, and E. Nyberg. 2017. Structural embedding of syntactic trees for machine comprehension. *arXiv*.

D. Lowd and C. Meek. 2005. Adversarial learning. In *International Conference on Knowledge Discovery and Data Mining (KDD)*.

N. Madnani and B. J. Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. 2014. The stanford coreNLP natural language processing toolkit. In *ACL system demonstrations*.

M. Marcus, B. Santorini, M. A. Marcinkiewicz, and A. Taylor. 1999. *Treebank-3*.

S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. 2017. Universal adversarial perturbations. In *Computer Vision and Pattern Recognition (CVPR)*.

N. Narodytska and S. P. Kasiviswanathan. 2016. Simple black-box adversarial perturbations for deep networks. *arXiv preprint arXiv:1612.06299*.

D. Paperno, G. Kruszewski, A. Lazaridou, Q. N. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernandez. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Association for Computational Linguistics (ACL)*.

N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Celik, and A. Swami. 2017. Practical black-box attacks against deep learning systems using adversarial examples. In *Proceedings of the ACM Asia Conference on Computer and Communications Security*.

J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

L. Rimell, S. Clark, and M. Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv*.

Y. Shen, P. Huang, J. Gao, and W. Chen. 2017. Reasonet: Learning to stop reading in machine comprehension. In *International Conference on Knowledge Discovery and Data Mining (KDD)*.

N. A. Smith. 2012. Adversarial evaluation for models of natural language. *arXiv preprint arXiv:1207.0245*.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*.

S. Wang and J. Jiang. 2016. Machine comprehension using match-LSTM and answer pointer. *arXiv preprint arXiv:1608.07905*.

Z. Wang, H. Mi, W. Hamza, and R. Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv*.

D. Weissenborn, G. Wiese, and L. Seiffe. 2017. Making neural qa as simple as possible but not simpler. *arXiv*.

Y. Yu, W. Zhang, K. Hasan, M. Yu, B. Xiang, and B. Zhou. 2016. End-to-end answer chunk extraction and ranking for reading comprehension. *arXiv*.

J. Zhang, X. Zhu, Q. Chen, L. Dai, S. Wei, and H. Jiang. 2017. Exploring question understanding and adaptation in neural-network-based question answering. *arXiv*.

Reasoning with Heterogeneous Knowledge for Commonsense Machine Comprehension

Hongyu Lin[1,2] **Le Sun**[1] **Xianpei Han**[1]
[1]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
{hongyu2016,sunle,xianpei}@iscas.ac.cn

Abstract

Reasoning with commonsense knowledge is critical for natural language understanding. Traditional methods for commonsense machine comprehension mostly only focus on one specific kind of knowledge, neglecting the fact that commonsense reasoning requires simultaneously considering different kinds of commonsense knowledge. In this paper, we propose a multi-knowledge reasoning method, which can exploit heterogeneous knowledge for commonsense machine comprehension. Specifically, we first mine different kinds of knowledge (including event narrative knowledge, entity semantic knowledge and sentiment coherent knowledge) and encode them as inference rules with costs. Then we propose a multi-knowledge reasoning model, which selects inference rules for a specific reasoning context using attention mechanism, and reasons by summarizing all valid inference rules. Experiments on RocStories show that our method outperforms traditional models significantly.

1 Introduction

Commonsense knowledge is fundamental in artificial intelligence, and has long been a key component in natural language understanding and human-like reasoning. For example, to understand the relation between sentences "*Mary walked to a restaurant*" and "*She ordered some foods*", we need commonsense knowledge such as "*Mary is a girl*", "*restaurant sells food*", etc. The task of understanding natural language with commonsense knowledge is usually referred as *commonsense machine comprehension*, which has been a hot topic in recent years (Richardson et al., 2013; Weston et al., 2015; Zhang et al., 2016).

Recently, *RocStories* (Mostafazadeh et al., 2016a), a commonsense machine comprehension task, has attached many researchers' attention due to its significant difference from previous machine comprehension tasks. *RocStories* focuses on reasoning with implicit commonsense knowledge, rather than matching with explicit information in given contexts. In this task, a system requires choosing a sentence, namely *hypothesis*, to complete a given commonsense story, called as *premise document*. Table 1 shows two examples. RocStories proposes a challenging benchmark task for evaluating commonsense-based language understanding. As investigated by Mostafazadeh et al.(2016a), this dataset does not have any boundary cases and thus results in 100% human performance.

Commonsense machine comprehension, however, is an natural ability for human but could be very challenging for computers. In general, any world knowledge whatsoever in the reader's mind can affect the choice of an interpretation (Dahlgren et al., 1989). That is, a person can learn any heterogeneous commonsense knowledge and make inference of given information based on all knowledge in his mind. For example, to choose the right hypothesis for the first premise document in Table 1, we needs the event narrative knowledge that "*X does a thorough job*" will lead to "*commends X*", rather than "*fire X*". Besides, people can further confirm their judgement based on the sentimental coherence between "*finish super early*" and "*job well done*". Furthermore, in the second example, even both hypothesises are consistent with the premise document in both event and sentimental facets, we can still infer the right answer easily using the commonsense knowledge that "*puppy*" is a dog, meanwhile "*kitten*" is a cat.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2032–2043
Copenhagen, Denmark, September 7–11, 2017.

Premise Document	Right Hypothesis	Wrong Hypothesis
Ron started his new job as a landscaper today. He loves the outdoors and has always enjoyed working in it. His boss tells him to re-sod the front yard of the mayor's home. Ron is ecstatic, but **does a thorough job** and **finishes super early**.	His boss **commends** him for a **job well done**.	Ron is immediately **fired** for insubordination.
One day, my sister came over to the house to show us her **puppy**. She told us that she had just gotten the puppy across the street. My sons begged me to get them one. I told them that if they would care for it, they could have it.	My son said they would, so we got a **dog**.	We then grabbed a small **kitten**.

Table 1: Examples of RocStories Dataset.

In recent years, many methods have been proposed for commonsense machine comprehension. However, these methods mostly either focus on matching explicit information in given texts (Weston et al., 2014; Wang and Jiang, 2016a,b; Wang et al., 2016b; Zhao et al., 2017), or paid attention to one specific kind of commonsense knowledge, such as event temporal relation (Chambers and Jurafsky, 2008; Modi and Titov, 2014; Pichotta and Mooney, 2016b; Hu et al., 2017) and event causality (Do et al., 2011; Radinsky et al., 2012; Hashimoto et al., 2015; Gui et al., 2016). As discussed above, it is obvious that commonsense machine comprehension problem is far from settled by considering only explicit or a single kind of commonsense knowledge. To achieve human-like comprehension and reasoning, there exist two main challenges:

1) How to mine and represent different kinds of implicit knowledge that commonsense machine comprehension needs. For example, to complete the first example in Table 1, we need a system equipped with the event narrative knowledge that "*commends X*" can be inferred from "*X does a thorough job*", as well as the sentiment coherent knowledge that "*insubordination*" and "*finish super early*" are sentimental incoherent.

2) How to reason with various kinds of commonsense knowledge. As shown above, knowledge that reasoning process needs varies for different contexts. For human-like commonsense machine comprehension, a system should take various kinds of knowledge into consideration, decide what knowledge will be utilized in a specific reasoning contexts, and make the final decision by taking all utilized knowledge into consideration.

To address the above problems, this paper proposes a new commonsense reasoning approach, which can mine and exploit heterogeneous knowledge for commonsense machine comprehension. Specifically, we first mine different kinds of knowledge from raw text and relevant knowledge base, including event narrative knowledge, entity semantic knowledge and sentiment coherent knowledge. These heterogeneous knowledge are encoded into a uniform representation – inference rules between elements under different kinds of relations, with an inference cost for each rule. Then we design a rule selection model using attention mechanism, modeling which inference rules will be applied in a specific reasoning context. Finally, we propose a multi-knowledge reasoning model, which measures the reasoning distance from a premise document to a hypothesis as the expected cost sum of all inference rules applied in the reasoning process.

By modeling and exploiting heterogeneous knowledge during commonsense reasoning, our method can achieve more accurate and more robust performance than traditional methods. Furthermore, our method is a general framework, which can be extended to incorporate new knowledge easily. Experiments show that our method achieves a 13.7% accuracy improvement on the standard RocStories dataset, a significant improvement over previous work.

2 Commonsense Knowledge Acquisition for Machine Comprehension

As described above, various knowledge can be exploited for machine comprehension. In this section, we describe how to mine different knowledge from different sources. Specifically, we mine three types of commonly used commonsense knowledge, including: 1)*Event narrative knowledge*, which captures temporal and causal relations between events; 2)*Entity semantic knowledge*, which captures semantic relations between entities; 3)*Sentiment coherent knowledge*, which captures sentimental coherence between elements.

In this paper, we represent commonsense knowledge as a set of inference rules given in the form of $X \xrightarrow{f} Y : s$, which means that element Y can be inferred from element X under relation f, with an inference cost s. An element can stand

	Antecedent	Consequent	Relation	Cost
①	Mary	she	coreference	0.0
②	restaurant	order	narrative	0.1
③	restaurant	food	associative	0.1
④	restaurant	food	narrative	0.3
⑤	Mary	order	narrative	0.5
⑥	walk	sleep	narrative	0.8
⑦	walk	food	narrative	0.9

Table 2: Examples of Inference Rules.

for either event, entity or sentiment, and this paper represents elements using lemmatized nouns, verbs and adjectives. The lexical element representation can also be easily extended to structural representation, like the one in (Chambers and Jurafsky, 2008), if needed. However, in auxiliary experiments we found that using structural elements results in severe sparseness and noises which in turn will hurt the reasoning performance. Therefore, we think an individual work is needed to solve it. Table 2 demonstrates several examples of inference rules. In following, we describe how to mine different types of inference rules.

2.1 Mining Event Narrative Knowledge

Event narrative knowledge captures structured temporal and casual knowledge about stereotypical event sequences, which is fundamental for commonsense machine comprehension. For example, we can infer "X ordered some foods" from "X walked to a restaurant" using event narrative knowledge. Previous work (Chambers and Jurafsky, 2008; Rudinger et al., 2015) proves that event narrative knowledge can be mined from raw texts unsupervisedly. So we propose two models to encode this knowledge using inference rules.

The first one is based on ordered PMI, which is also proposed by Rudinger et al. (2015). Given two element e_1 and e_2, this model calculates the cost of inference rule $e_1 \xrightarrow{narrative} e_2$ as:

$$cost(e_1 \rightarrow e_2) = -log\frac{C(e_1, e_2)}{C(e_1, *), C(*, e_2)} \tag{1}$$

Here $C(e_1, e_2)$ is the order sensitive count that element e_1 occurs before element e_2 in different sentences of the same document.

The second model is a variant of the skip-gram model (Mikolov et al., 2013). The goal of this model is to find element representations which can accurately predict relevant elements in sentences afterwards. Formally, given n asymmetric pairs of elements $(e_1^1, e_2^1), (e_1^2, e_2^2), ..., (e_1^n, e_2^n)$ identified from training data, the objective of our model is to maximize the average log probability $\frac{1}{n}\sum_{i=1}^{n} logP(e_2^i|e_1^i)$. And the probability $P(e_2|e_1)$ is defined using the softmax function:

$$P(e_2|e_1) \propto exp(\mathbf{v}'_{e_2}{}^T \mathbf{v}_{e_1}) \tag{2}$$

where $\mathbf{v}_e$ and $\mathbf{v}'_e$ are "antecedent" and "consequent" vector representation of element e, respectively. We use the negative inner product $-\mathbf{v}'_{e_2}{}^T \mathbf{v}_{e_1}$ as the cost of inference rule $e_1 \xrightarrow{skip-gram} e_2$.

2.2 Mining Entity Semantic Knowledge

Entities, often serving as event participants or environment variables, are important components of commonsense stories. Intuitively, an entity in hypothesis is reasonable if we can identify semantic relations between it and some parts of premise document. For example, if a premise document contains "Starbucks", then "coffeehouse" and "latte" will be reasonable entities in hypothesis since "Starbucks" is a possible coreference of "coffeehouse" and it is semantically related to "latte".

Specifically, we identify mainly two kinds of semantic relations between entities for commonsense machine comprehension:

1) *Coreference relation,* which indicates that two elements refer to the same entity in environment. In stories, besides to pronouns, an entity is often referred using its hypernyms, e.g, the second example in Table 1 uses "dog" to refer to "puppy". Motivated by this observation, we mine coreference knowledge between elements using Wordnet (Kilgarriff and Fellbaum, 2000): $X \xrightarrow{coref} Y$ is an inference rule with cost 0 if X and Y are lemmas in the same Wordnet synset, or with hyponymy relation in Wordnet. Otherwise, the cost of inference rules between this element-pair under this relation will be 1.

2) *Associative relation,* which captures the semantic relatedness between two entities, i.e., "starbucks" → "latte", "restaurant" → "food", etc. This paper mines associative relations between entities from Wikipedia[1], using the method proposed by Milne and Witten(2008). Specifically, given two entities e_1 and e_2, we compute the semantic distance $dist(e_1, e_2)$ between them as:

$$dist(e_1, e_2) = \frac{log(max(|E_1|, |E_2|) - log(|E_1 \bigcap E_2|))}{log(|W|) - log(min(|E_1|, |E_2|))} \tag{3}$$

where E_1 and E_2 are the sets of all entities that link to these two entities in Wikipedia respectively,

[1] https://www.wikipedia.org/

and W is the entire Wikipedia. We set the cost of inference rule $e_1 \xrightarrow{associative} e_2$ as $dist(e_1, e_2)$.

2.3 Mining Sentiment Coherent Knowledge

Sentiment is one of the central and pervasive aspects of human experience (Ortony et al., 1990). It plays an important role in commonsense stories, i.e., a reasonable hypothesis should be sentimental coherent with its premise document. In this paper, we mine sentiment coherence rules using SentiWordnet (Baccianella et al., 2010), in which each synset of Wordnet is assigned with three sentiment scores: positivity, negativity and objectivity.

Concretely, to identify sentimental coherence rule between two element e_1 and e_2, we first compute the positivity, negativity and objectivity scores of every element by averaging the scores of all synsets it's in, then we identify an element to be subjective if its objectivity score is smaller than a threshold, and the distance between its positivity and negativity score is greater than a threshold. Finally, for an inference rule $e_1 \xrightarrow{senti} e_2$, we set its cost to 1 if e_1 and e_2 are both subjective and have opposite sentimental polarity, to -1 if they are both subjective and their sentimental polarity are the same, and to 0 for other cases. For example, we will mine inference rules "good $\xrightarrow{senti}$ happy : -1", "perfect $\xrightarrow{senti}$ sad : 1" and "young $\xrightarrow{senti}$ happy : 0".

2.4 Metric Learning to Calibrate Cost Measurement

So far, we have extracted many inference rules under different relations. However, because we extract them from different sources and estimate their costs using different measurements, the cost metrics of these rules may not be consistent with each other. To exploit different types of inference rules in a unified framework, we here propose a metric learning based method to calibrate their costs.

Given an input distance function, a metric learning method constructs a new distance function which is "better" than the original one with supervision regarding an ideal distance (Kulis, 2012). To calibrate inference rule cost, we add a nonlinear layer to the original cost s_r of inference rule r under relation f:

$$c_r = sigmoid(w_f s_r + b_f) \tag{4}$$

Here c_r is the metric-unified inference cost of inference rule r, w_f and b_f are calibration parameters for inference rules of relation f. We use sigmoid function in order to normalize costs into 0 to 1. Calibration parameters will be trained along with other parameters in our model. See Section 3.4 for detail.

2.5 Dealing with Negation

One important linguistic phenomenon needs to specifically consider is negation. Here we discuss how to solve negation in our model.

We use $\neg X$ to represent an element X modified by a negation word (the existence of negation is detected using dependency relations). Under event narrative relation and sentiment coherent relation, the existence of negation will reverse the conclusion. So we add three additional negation related inference rules for rule $X \xrightarrow{f} Y : s$ under these relations, including $\neg X \xrightarrow{f} Y : 1 - s$, $X \xrightarrow{f} \neg Y : 1 - s$ and $\neg X \xrightarrow{f} \neg Y : s$. Here s is the calibrated cost of the original inference rule. For entity semantic relations, we just ignore the negation since it will not affect the inference under these relations.

3 Machine Comprehension via Commonsense Reasoning

This section describes how to leverage acquired knowledge for commonsense machine comprehension. We first define how to infer from a premise document to a hypothesis using inference rules. Then we model how to choose inference rules for a specific reasoning context. Finally, we describe how to measure the reasoning distance from a premise document to a hypothesis by summarizing the costs of all possible inferences.

3.1 Inference from Premise Document to Hypothesis

Given a premise document $D = \{d_1, d_2, ..., d_m\}$ containing m elements, a hypothesis $H = \{h_1, h_2, ..., h_n\}$ containing n elements, a valid inference R from D to H is a set of inference rules that all elements in H can be inferred from one element in D using one and only one rule in R. This definition means that all elements in H should be covered by consequents of inference rules in R, as well as all antecedents of inference rules in R should come from D. Figure 1 shows some inference examples, where (a), (b) and (d) are valid inferences, but (c) is not a valid inference because its rules can not cover all elements in hypothesis.

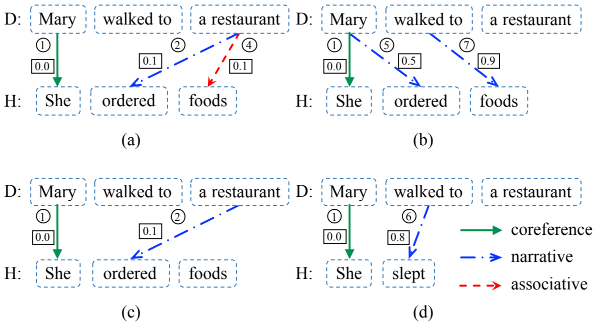


Figure 1: Examples of inferences. Numbers in circle indicates the proposed inference rules in Table 2, and values in rectangle are their costs.

By the definition, the size of R and the size of H are equal. So we use r_i to denote the inference rule in R that applied to derive element h_i in H, i.e., $R = \{r_1, r_2, ..., r_n\}$.

Based on the above definition, we can naturally define the cost of an inference R as the cost sum of all inference rules in R. In Figure 1, the cost for inference (a) is $0.0 + 0.1 + 0.1 = 0.2$, and for inference (d) is $0.0 + 0.8 = 0.8$.

3.2 Modeling Inference Probability using Attention Mechanism

Obviously, there exist multiple valid inferences for a premise document and a hypothesis. For example, in Figure 1, both (a) and (b) are valid inferences for the same premise document and hypothesis. To identify whether a hypothesis is reasonable, we need to consider all possible inferences. However, in human reasoning process, not all inference rules have the same possibility to be applied, because the more reasonable inference will be proposed more likely. In Figure 1, inference (a) should have a higher probability than inference (b) because it is more reasonable to infer "*foods*" from "*a restaurant*" with *associative relation*, rather than from "*walked to*" with *narrative relation*. Besides, the possibility of proposing an inference should not depend on its cost, e.g., inference (d) should have high possibility to be proposed despite its high cost, because we often infer event "*sleep*" from another event using inference rules under *narrative relation*. As examples mentioned above, the "cost" measures the "correctness" of an inference rule. A rule with low cost is more likely to be "reasonable", and a rule with high cost is more likely to be a contradiction with commonsense. On the other hand, the "possibility" should measure how likely a rule will be applied in a given context, which does not depend on the "cost" but on the nature of the rule and the given context. Motivated by above observations, we endow each inference a probability $P(R|D, H)$, indicating the possibility that R is chosen to infer hypothesis H from premise document D. For simplicity, we assume that each element in hypothesis is independently inferred using individual inference rule, then $P(R|D, H)$ can be written as:

$$P(R|D,H) = \prod_{i=1}^{n} P(r_i|D,H) \tag{5}$$

$$= \prod_{i=1}^{n} P(r_i|D,h_i) \tag{6}$$

$$= \prod_{i=1}^{n} \sum_{j=1}^{m} P(r_i, d_j|D, h_i) \tag{7}$$

Equation (7) clearly shows how an inference rule is selected given the premise document D and the element h_i in hypothesis. It depends on which element d_j in D will be selected and which relation f will be used to infer h_i from d_j. We then refactor the probability $P(r_i, d_j|D, h_i)$ to be:

$$P(r_i, d_j|D, h_i) = \begin{cases} 0 & , antecedent(r_i) \neq d_j \\ g(h_i, d_j, f(r_i); D) & , otherwise \end{cases} \tag{8}$$

Here $f(r)$ is the relation type of inference rule r, and $g(h, d, f; D)$ is defined as:

$$g(h, d, f; D) = \frac{s(h,d)a(h,f)a(d,f)}{\sum_{f \in \mathcal{F}} \sum_{d \in D} s(h,d)a(h,f)a(d,f)} \tag{9}$$

Here $\mathcal{F}$ denotes all relation types of inference rules, $s(e_1, e_2)$ is a matching function between two elements e_1 and e_2, measuring by cosine similarity based on GoogleNews word2vec (Mikolov et al., 2013). And $a(e, f)$ is an attention function measuring how likely an element e will be involved with rules under relation f:

$$a(e, f) = \mathbf{v_f}^T tanh(\mathbf{W_f e} + \mathbf{b_f}) \tag{10}$$

where $\mathbf{v_f} \in \mathbb{R}^K$, $\mathbf{W_f} \in \mathbb{R}^{K \times F}$ and $\mathbf{b_f} \in \mathbb{R}^K$ are attention parameters of relation f, and $\mathbf{e} \in \mathbb{R}^F$ is the feature vector of element e. Here K is the size of attention hidden layer and F is the dimension of feature vector. We consider three types of features, as shown in Table 3. Using attention mechanism, our method models the possibility that an inference rule is applied during the inference from a premise document to a hypothesis by considering the relatedness between elements and knowledge category, as well as the relatedness between two elements, which make it able to select the most reasonable inference rules to derive each part of the hypothesis.

Feature	Description
Syntax Features	
is_verb	whether this element is a verb
is_noun	whether this element is a noun
is_adj	whether this element is an adjective
Lexical Features	
is_event	whether this element belongs to hyponymy of *event sysnset* in Wordnet
is_entity	whether this element belongs to hyponymy of *physical_entity sysnset*
is_attr	whether this element belongs to hyponymy of *attribute sysnset*
named_entity	the named entity type of this element
Semantic Features	
word embeddings	300 dimension embeddings of GoogleNews word2vec model

Table 3: Features for element-relation attention.

3.3 Reasoning Distance Between Premise Document and Hypothesis

Given a premise document, this section shows how to measure whether a hypothesis is coherent using above inference model. Given all valid inferences from D to H and the probability $P(R|D,H)$ of selecting inference R to infer H from D, we measure the reasoning distance $L(D \to H)$ as the expected cost sum of all valid inferences:

$$L(D \to H) = E_{P(R|D,H)}[cost(R)] \tag{11}$$

$$= E_{P(R|D,H)}[\sum_{i=1}^{n} cost(r_i)] \tag{12}$$

Then using Equation (6) and Equation (7), we can further rewrite the equation into:

$$L(D \to H) = \sum_{R}[\prod_{i=1}^{n} P(r_i|D,h_i)] \cdot [\sum_{i=1}^{n} cost(r_i)] \tag{13}$$

$$= \sum_{i=1}^{n} P(r_i|D,h_i) \cdot cost(r_i) \tag{14}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{m} P(r_i,d_j|D,h_i) \cdot cost(r_i) \tag{15}$$

Equation (15) shows that in our framework, the final cost of inferring the element h_i in the hypothesis is the expected cost of all valid inference rules which can derive h_i from one element in the premise document.

3.4 Model Learning

Following Huang et al. (2013), our model measures the posterior probability of choosing hypothesis H as the answer of premise document D through a softmax function:

$$P(H|D) = \frac{exp(-\gamma L(D \to H))}{\sum_{H' \in \mathcal{H}_D} exp(-\gamma L(D \to H'))} \tag{16}$$

Here $\mathcal{H}_D$ is all candidate hypothesises for D, and γ is a positive smoothing factor. We train our model by maximizing the likelihood of choosing right hypothesis H^+ for D:

$$\mathcal{L}(\theta) = -log \prod_{(D,H^+)} P(H^+|D) \tag{17}$$

where θ is the parameter set of our model, including calibration parameters in Section 2.4 and attention parameters in Section 3.2. $\mathcal{L}(\theta)$ is differentiable so we can estimate θ using any gradient-based optimization algorithm.

4 Experiments

4.1 Experimental Settings

Data Preparation. We evaluated our approach on the *Test Set Spring 2016* of RocStories, which consists of 1871 commonsense stories, with each story has two candidate story endings. Because stories in the training set of RocStories do not contain wrong hypothesis, and our model has a compact size of parameters, we estimated the parameters of our model using the *Validation Set Spring 2016* of RocStories with 1871 commonsense stories.

We mined event narrative knowledge from the *Training Set Spring 2016* of RocStories, which consists of 45502 commonsense stories. We performed lemmatisation, part of speech annotation, named entity tagging, and dependency parsing using Stanford CoreNLP toolkits (Manning et al., 2014). We used the Jan. 30, 2010 English version of Wikipedia and processed it according to the method described by Hu et al. (2008).

Model Training. We used normalized initialization (Glorot and Bengio, 2010) to initialize attention parameters in our model. For calibration parameters, we initialized all $\mathbf{w_f}$ to 1 and $\mathbf{b_f}$ to 0. The model parameters were trained using minibatch stochastic gradient descent algorithm. As for hyper-parameters, we set the batch size as 32, the learning rate as 1, the dimension of attention hidden layer K as 32, and the smoothing factor γ as 0.5.

Baselines. We compared our approach with following three baselines:

1) **Narrative Event Chain** (Chambers and Jurafsky, 2008), which scores hypothesis using PMI scores between events. We used a simplified version of the original model by using only verbs as event, ignoring the dependency relation between verbs and their participants. We found such a simplified version achieved better performance than its original one whose performance was reported in (Mostafazadeh et al., 2016a).

2) **Deep Structured Semantic Model (DSS-**

M) (Huang et al., 2013), which achieved the best performance on RocStories as reported by Mostafazadeh et al.(2016a). This model measures the reasoning score between a premise document D and a hypothesis H by calculating the cosine similarity between the overall vector representations of D and H, and do not consider any other task-relevant knowledge.

3) **Recurrent Neural Network(RNN) Model** proposed by Pichotta and Mooney(2015), which transforms all events and their arguments into a sequence and predict next events and arguments using a Long Short-Term Memory network. We used the average generating probability of all elements in H as the reasoning score, and choose the hypothesis with largest reasoning score as the system answer.

4.2 Overall Performance

System	**Accuracy**
Narrative Event Chain	57.62%
DSSM	58.52%
RNN Model	58.93%
Our Model	**67.02%**

Table 4: Comparison of accuracy for our model and three baselines on RocStories Spring 2016 Test Set. The result of DSSM is adapted from (Mostafazadeh et al., 2016a).

Table 4 shows the results. From this table, we can see that:

1) Our model outperforms all baselines significantly. Compared with baselines, the accuracy improvement on test set is at least 13.7%. This demonstrates the effectiveness of our model by mining and exploiting heteregenous knowledge.

2) The event narrative knowledge only is insufficient for commonsense machine comprehension. Compared with Narrative Event Chain Model, our model achieves a 16.3% accuracy improvement by considering richer commonsense knowledge, rather than only narrative event knowledge.

3) It is necessary to distinguish different kinds of commonsense relations for machine comprehension and commonsense reasoning. Compared with DSSM and RNN, which model all relations between two elements using a single semantic similarity score, our model achieves significant accuracy improvements by modeling, distinguishing and selecting different types of commonsense relations between different kinds of elements.

4.3 Effects of Different Knowledge

To investigate the effect of different kinds of knowledge in our model, we conducted two groups of experiments.

The first group of experiments was conducted using only one kind of knowledge at a time in our model. Table 5 shows the results. We can see that using a single kind of knowledge is insufficient for commonsense machine comprehension: all single-knowledge settings cannot achieve competitive performance to the all-knowledge setting.

System	**Accuray**
Event Narrative Knowledge	60.98%
Entity Semantics Knowledge	57.14%
Sentiment Coherent Knowledge	61.30%
Our Model(All Knowledge)	**67.02%**

Table 5: Comparison of the performance using single type of knowledge.

The second group of experiments was conducted to investigate whether different knowledge can complement each other. We conducted experiments by removing one kind of knowledge from our final model at a time, and investigate the change of accuracy.

System	**Accuracy**
Our Model(All Knowledge)	**67.02%**
-w/o Event Narrative Knowledge	63.65%
-w/o Entity Semantic Knowledge	64.89%
-w/o Sentiment Coherent Knowledge	62.85%

Table 6: Comparison of the performance by removing one single type of knowledge.

Table 6 shows the results. We can find that removing any kind of knowledge will reduce the accuracy. This verified that all kinds of knowledge containing unique complementary information, which cannot be covered by other types of knowledge.

4.4 Effect of Inference Probability

This section investigates the effect of inference rule selection probability, and whether our attention mechanism can effectively model the possibility of inference rule selection. We compared our method with following two heuristic settings:

1) **Minimum Cost Mechanism**, which measures the reasoning distance by only selecting the inference rule with minimum cost for each hypothesis element.

2) **Average Cost Mechanism**, which measures the reasoning distance by setting equal probabilities to all inference rules that can infer a hypothesis element from a premise document element.

System	Accuracy
Minimum Cost Mechanism	54.84%
Average Cost Mechanism	63.01%
Our Model(Attention Mechanism)	**67.02%**

Table 7: Comparison of the performance using different inference rule selection mechanism.

Table 7 show the results. We can see that: 1) the minimum cost mechanism cannot achieve competitive performance, we believe this is because the selection of rules should not depend on the cost of them, and considering all valid inferences is critical for reasoning; 2) our attention mechanism can effectively model the inference rule selection possibility. Compared with the average cost mechanism, our method achieved a 6.36% accuracy improvement. This also verified the necessity of an effective inference rule probability model.

4.5 Effect of Negation Rules

This section investigates the effect of special handling of negation mentioned in Section 2.5. To investigate the necessity of negation rules proposed in our model, we conducted experiments by removing all negation rules from original system, and investigate the change of accuracy.

System	Accuracy
Our Model	**67.02%**
-w/o Negation Rules	63.12%

Table 8: Comparison of the performance by removing negation rules.

Table 8 show the results. We can see that removing negation rules will significantly drop the system performance, which confirm the effectiveness of our proposed negation rules.

5 Related Work

Endowing computers with the ability of understanding commonsense story has long a goal of natural language processing. There exist two big challenges: 1)Matching explicit information in the given context; 2)Incorporating implicit commonsense knowledge into human-like reasoning process. Previous machine comprehension tasks (Richardson et al., 2013; Weston et al., 2015; Hermann et al., 2015; Rajpurkar et al., 2016) mainly focus on the first challenge, leading their solutions focusing on semantic matching between texts (Weston et al., 2014; Kumar et al., 2015; Narasimhan and Barzilay, 2015; Smith et al., 2015; Sukhbaatar et al., 2015; Hill et al., 2015; Wang et al., 2015, 2016a; Cui et al., 2016; Trischler et al., 2016a,b; Kadlec et al., 2016; Kobayashi et al., 2016; Wang and Jiang, 2016b), but ignore the second issues. One notable task is SNLI (Bowman et al., 2015), which considers entailment between two sentences. This task, however, only provides shallow context and thus needs a few kinds of implicit knowledge (Rocktäschel et al., 2015; Wang and Jiang, 2016a; Angeli et al., 2016; Wang et al., 2016b; Parikh et al., 2016; Henderson and Popa, 2016; Zhao et al., 2017).

Realizing that story understanding needs commonsense knowledge, many researches have been proposed to learn structural event knowledge. Chambers and Jurafsky (2008) first proposed an unsupervised approach to learn partially ordered sets of events from raw text. Many expansions have been introduced later, including unsupervisedly learning narrative schemas and scripts (Chambers and Jurafsky, 2009; Regneri et al., 2011), event schemas and frames (Chambers and Jurafsky, 2011; Balasubramanian et al., 2013; Sha et al., 2016; Huang et al., 2016; Mostafazadeh et al., 2016b), and some generative models to learn latent structures of event knowledge (Cheung et al., 2013; Chambers, 2013; Bamman et al., 2014; Nguyen et al., 2015). Another direction for learning event-centred knowledge is causality identification (Do et al., 2011; Radinsky et al., 2012; Berant et al., 2014; Hashimoto et al., 2015; Gui et al., 2016), which tried to identify the causality relation in text.

For reasoning over these knowledge, Jans et al. (2012) extend introduced skip-grams for collecting statistics. Further improvements include incorporating more information and more complicated models (Radinsky and Horvitz, 2013; Modi and Titov, 2014; Ahrendt and Demberg, 2016). Recent researches tried to solve event prediction problem by transforming it into an language modeling paradigm (Pichotta and Mooney, 2014, 2015, 2016a,b; Rudinger et al., 2015; Hu et al., 2017).

The principal difference between previous work and our method is that we not only take various kinds of implicit commonsense knowledge into consideration, but also provide a highly extensible framework to exploit these kinds of knowledge for commonsense machine comprehension. We also notice the recent progress in RocStories (Mostafazadeh et al., 2017). Rather than inferring a possible ending generated from document, recent systems solve this task by discriminatively comparing two candidates. This enables very strong stylistic features being added explicitly (Schwartz et al., 2017; Bugert et al., 2017) or implicitly (Schenk and Chiarcos, 2017), which can select hypothesis without any consideration of given document. Also, some augmentation strategies are introduced to produce more training data (Roemmele and Gordon, 2017; Mihaylov and Frank, 2017; Bugert et al., 2017). These methods are dataset-sensitive and are not the main concentration of our paper.

6 Conclusions and Future Work

This paper proposes a commonsense machine comprehension method, which performs effective commonsense reasoning by taking heterogenous knowledge into consideration. Specifically, we mine commonsense knowledge from heterogeneous knowledge sources and simultaneously exploit them by proposing a highly extensible multi-knowledge reasoning framework. Experiment results shown that our method surpasses baselines by a large margin.

Currently, there are little labeled training instances for commonsense machine comprehension, for future work we want to address this issue by developing semi-supervised or unsupervised approaches.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants no. 61433015 and 61572477, the National High Technology Development 863 Program of China under Grants no. 2015AA015405, and the Young Elite Scientists Sponsorship Program no. YESS20160177. Moreover, we sincerely thank the reviewers for their valuable comments.

References

Simon Ahrendt and Vera Demberg. 2016. Improving event prediction by representing script participants. In *Proceedings of NAACL-HLT*, pages 546–551.

Gabor Angeli, Neha Nayak, and Christopher D Manning. 2016. Combining natural logic and shallow reasoning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 442–452.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204.

Niranjan Balasubramanian, Stephen Soderland, Oren Etzioni Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *EMNLP*, pages 1721–1731.

David Bamman, Brendan O'Connor, and Noah A Smith. 2014. Learning latent personas of film characters. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (A-CL)*, page 352.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling biological processes for reading comprehension. In *EMNLP*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Michael Bugert, Yevgeniy Puzikov, R Andreas, Judith Eckle-kohler, Teresa Martin, and Eugenio Mart. 2017. LSDSem 2017 : Exploring Data Generation Methods for the Story Cloze Test. In *The 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (ISDSEM 2017)*, 2016, pages 56–61, Valencia, Spain.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *EMNLP*, volume 13, pages 1797–1807.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 602–610. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 976–986. Association for Computational Linguistics.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL*, volume 94305, pages 789–797. Citeseer.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. *arXiv preprint arXiv:1302.4813*.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*.

Kathleen Dahlgren, Joyce McDowell, and Edward P Stabler. 1989. Knowledge representation for commonsense reasoning with text. *Computational Linguistics*, 15(3):149–170.

Quang Xuan Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 294–303. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256.

Lin Gui, Dongyin Wu, Ruifeng Xu, Qin Lu, and Yu Zhou. 2016. Event-driven emotion cause extraction with corpus construction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1639–1649.

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, and Jong-Hoon Oh. 2015. Generating event causality hypotheses through semantic relations. In *AAAI*, pages 2396–2403.

James Henderson and Diana Nicoleta Popa. 2016. A vector space for distributional semantics for entailment. *arXiv preprint arXiv:1607.03780*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Jian Hu, Lujun Fang, Yang Cao, Hua-Jun Zeng, Hua Li, Qiang Yang, and Zheng Chen. 2008. Enhancing text clustering by leveraging wikipedia semantics. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 179–186. ACM.

Linmei Hu, Juanzi Li, Liqiang Nie, Xiao-Li Li, and Chao Shao. 2017. What happens next? future subevent prediction using contextual hierarchical lstm. In *Proceedings of the 31th AAAI Conference on Artificial Intelligence*.

Lifu Huang, T Cassidy, X Feng, H Ji, CR Voss, J Han, and A Sil. 2016. Liberal event extraction and event schema induction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-16)*.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM.

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 336–344. Association for Computational Linguistics.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*.

Adam Kilgarriff and Christiane Fellbaum. 2000. Wordnet: An electronic lexical database.

Sosuke Kobayashi, Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2016. Dynamic entity representation with max-pooling improves machine reading. In *Proceedings of NAACL-HLT*, pages 850–855.

Brian Kulis. 2012. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR, abs/1506.07285*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Todor Mihaylov and Anette Frank. 2017. Story Cloze Ending Selection Baselines and Data Examination. In *The 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (ISDSEM 2017)*, pages 2–7, Valencia, Spain.

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

David Milne and Ian H Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *CoNLL*, volume 14, pages 49–57.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016a. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of NAACL-HLT*, pages 839–849.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016b. Caters: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the The 4th Workshop on EVENTS: Definition, Detection, Coreference, and Representation, San Diego, California, June. Association for Computational Linguistics*.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael William Chambers, and James F. Allen. 2017. LSDSem 2017 Shared Task : The Story Cloze Test. In *The 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (ISDSEM 2017)*, 2016, pages 1–5, Valencia, Spain.

Karthik Narasimhan and Regina Barzilay. 2015. Machine comprehension with discourse relations. In *ACL (1)*, pages 1253–1262.

Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. 2015. Generative event schema induction with entity disambiguation. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics (ACL-15)*.

Andrew Ortony, Gerald L Clore, and Allan Collins. 1990. *The cognitive structure of emotions*. Cambridge university press.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Karl Pichotta and Raymond J Mooney. 2014. Statistical script learning with multi-argument events. In *EACL*, volume 14, pages 220–229.

Karl Pichotta and Raymond J Mooney. 2015. Learning statistical scripts with lstm recurrent neural networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

Karl Pichotta and Raymond J Mooney. 2016a. Statistical script learning with recurrent neural networks. *EMNLP 2016*, page 11.

Karl Pichotta and Raymond J Mooney. 2016b. Using sentence-level lstm language models for script inference. *arXiv preprint arXiv:1604.02993*.

Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. 2012. Learning causality for news events prediction. In *Proceedings of the 21st international conference on World Wide Web*, pages 909–918. ACM.

Kira Radinsky and Eric Horvitz. 2013. Mining the web to predict future events. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 255–264. ACM.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Michaela Regneri, Alexander Koller, Josef Ruppenhofer, and Manfred Pinkal. 2011. Learning script participants from unlabeled data. In *RANLP*, pages 463–470.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 3, page 4.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Melissa Roemmele and Andrew M Gordon. 2017. An RNN-based Binary Classifier for the Story Cloze Test. In *The 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (ISDSEM 2017)*, pages 74–80, Valencia, Spain.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script induction as language modeling. In *EMNLP*, pages 1681–1686.

Niko Schenk and Christian Chiarcos. 2017. Resource-Lean Modeling of Coherence in Commonsense Stories. In *The 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (ISDSEM 2017)*, pages 68–73, Valencia, Spain.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, Noah A Smith, and Computer Science. 2017. Story Cloze Task : UW NLP System. In *The 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (ISDSEM 2017)*, pages 52–55, Valencia, Spain.

Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. Joint learning templates and slots for event schema induction. In *Proceedings of NAACL-HLT*, pages 428–434.

Ellery Smith, Nicola Greco, Matko Bosnjak, and Andreas Vlachos. 2015. A strong lexical matching method for the machine comprehension test. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1693–1698. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He, Phillip Bachman, and Kaheer Suleman. 2016a. A parallel-hierarchical model for machine comprehension on sparse data. *arXiv preprint arXiv:1603.08884*.

Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. 2016b. Natural language comprehension with the epireader. *arXiv preprint arXiv:1606.02270*.

Bingning Wang, Shangmin Guo, Kang Liu, Shizhu He, and Jun Zhao. 2016a. Employing external rich knowledge for machine comprehension. In *Proceedings of IJCAI*.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David A McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *ACL (2)*, pages 700–706.

Shuohang Wang and Jing Jiang. 2016a. Learning natural language inference with lstm. In *Proceedings of NAACL-HLT*, pages 1442–1451.

Shuohang Wang and Jing Jiang. 2016b. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016b. Sentence similarity learning by lexical decomposition and composition. *arXiv preprint arXiv:1602.07019*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Sheng Zhang, Rachel Rudinger, Kevin Duh, and Benjamin Van Durme. 2016. Ordinal common-sense inference. *arXiv preprint arXiv:1611.00601*.

Kai Zhao, Liang Huang, and Mingbo Ma. 2017. Textual entailment with structured attentions and composition. *arXiv preprint arXiv:1701.01126*.

Document-Level Multi-Aspect Sentiment Classification as Machine Comprehension

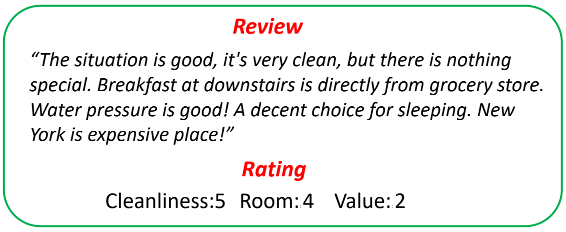
Yichun Yin[1], **Yangqiu Song**[2], **Ming Zhang**[1]
[1]School of Electronics Engineering and Computer Science, Peking University, Beijing, China
[2]Department of Computer Science and Engineering, HKUST, Hong Kong
{yichunyin, mzhang_cs}@pku.edu.cn, yqsong@cse.ust.hk

Abstract

Document-level multi-aspect sentiment classification is an important task for customer relation management. In this paper, we model the task as a machine comprehension problem where pseudo question-answer pairs are constructed by a small number of aspect-related keywords and aspect ratings. A hierarchical iterative attention model is introduced to build aspect-specific representations by frequent and repeated interactions between documents and aspect questions. We adopt a hierarchical architecture to represent both word level and sentence level information, and use the attention operations for aspect questions and documents alternatively with the multiple hop mechanism. Experimental results on the TripAdvisor and BeerAdvocate datasets show that our model outperforms classical baselines.

1 Introduction

Document-level sentiment classification is one of the pragmatical sentiment analysis tasks (Pang and Lee, 2007; Liu, 2010). There are many Web sites having platforms for users to input reviews over products or services, such as TripAdvisor, Yelp, Amazon, etc. Most of reviews are very comprehensive and thus long documents. Analyzing these documents to predict ratings of products or services is an important complementary way for better customer relationship management. Recently, neural network based approaches have been developed and become state-of-the-arts for long-document sentiment classification (Tang et al., 2015a,b; Yang et al., 2016). However, predicting an overall score for each long document is not enough, because the document can mention different aspects of the corresponding product or service. For example, in Figure 1, there could be different aspects for a review of hotel. These aspects help customer service better understand what are the major pros and cons of the product or service. Compared to the overall rating, users are less motivated to give aspect ratings. Therefore, it is more practically useful to perform document-level multi-aspect sentiment classification task, predicting different ratings for each aspect rather than an overall rating.



Review

"The situation is good, it's very clean, but there is nothing special. Breakfast at downstairs is directly from grocery store. Water pressure is good! A decent choice for sleeping. New York is expensive place!"

Rating

Cleanliness:5 Room:4 Value:2

Figure 1: Example: hotel review with aspects.

One straightforward approach for document-level multi-aspect sentiment classification is multi-task learning (Caruana, 1997). For neural networks, we can simply treat each aspect (e.g., rating from one to five) as a classification task, and let different tasks use softmax classifier to extract task-specific representations at the top layer while share the input and hidden layers to mutually enhance the prediction results (Collobert et al., 2011; Luong et al., 2016). However, such approach ignores the fact that the aspects themselves have semantic meanings. For example, as human beings, if we were asked to evaluate the aspect rating of a document, we simply read the review, and find aspect-related keywords, and see around comments. Then, we aggregate all the related snippets to make a decision.

In this paper, we propose a novel approach to treat document-level multi-aspect sentiment clas-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2044–2054
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

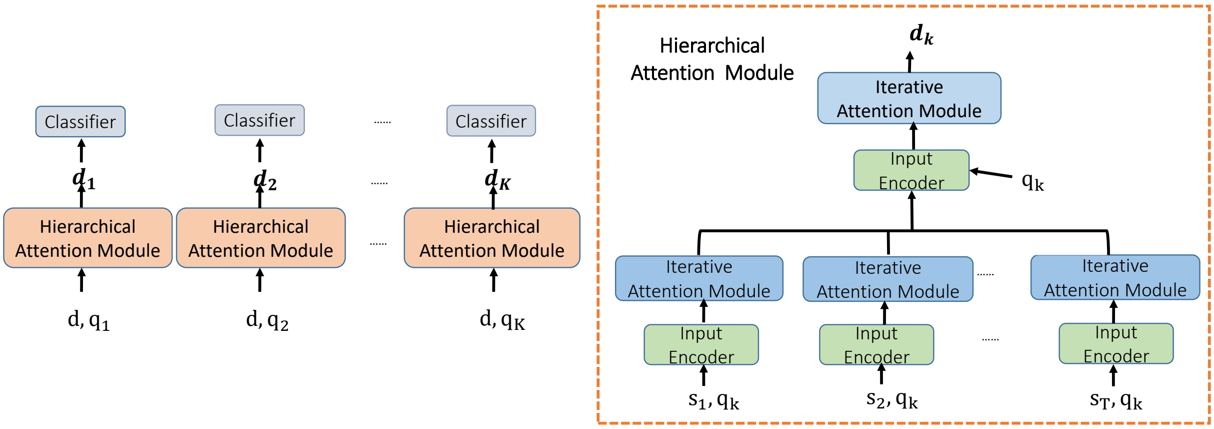

Classifier
Classifier
Classifier
d_1
d_2
d_K
Hierarchical Attention Module
Hierarchical Attention Module
Hierarchical Attention Module
d, q_1
d, q_2
d, q_K
Hierarchical Attention Module
d_k
Iterative Attention Module
Input Encoder
q_k
Iterative Attention Module
Iterative Attention Module
Iterative Attention Module
Input Encoder
Input Encoder
Input Encoder
s_1, q_k
s_2, q_k
s_T, q_k

Figure 2: The architecture of our model. Left: multi-task learning. Right: hierarchical attention module which includes input encoders and iterative attention modules.

sification as a machine comprehension (Kumar et al., 2016; Sordoni et al., 2016) problem. To mimic human's evaluation of aspect classification, we create a list of keywords for each aspect. For example, when we work on the *Room* aspect, we generate some keywords such as "room," "bed," "view," etc. Then we can ask pseudo questions: "How is the room?" "How is the bed?" "How is the view?" and provide an answer "Rating 5." In this case, we can train a machine comprehension model to automatically attend corresponding text snippets in the review document to predict the aspect rating. Specifically, we introduce a hierarchical and iterative attention model to construct aspect-specific representations. We use a hierarchical architecture to build up different representations at both word and sentence levels interacting with aspect questions. At each level, the model consists of input encoders and iterative attention modules. The input encoder learns memories[1] of documents and questions with Bi-directional LSTM (Bi-LSTM) model and non-linear mapping respectively. The iterative attention module takes into memories as input and attends them sequentially with a multiple hop mechanism, performing effective interactions between documents and aspect questions.

To evaluate the effectiveness of the proposed model, we conduct extensive experiments on the TripAdvisor and BeerAdvocate datasets and the results show that our model outperforms typical baselines. We also analyze the effects of numbers of the hop and aspect words on performances. Moreover, a case study for attention results is performed at both word and sentence levels.

The contributions of this paper are two-fold. First, we study the document-level multi-aspect sentiment classification as a machine comprehension problem and introduce a hierarchical iterative attention model for it. Second, we demonstrate the effectiveness of proposed model on two datasets, showing that our model outperforms classical baselines. The code and data for this paper are available at `https://github.com/HKUST-KnowComp/DMSCMC`.

2 Method

In this section, we introduce our proposed method.

2.1 Problem Definition and Hierarchical Framework

We first briefly introduce the problem we work on. Given a piece of review, our task is to predict the ratings of different aspects. For example, in Figure 1, we predict the ratings of *Cleanliness*, *Room*, and *Value*. To achieve this, we assume that there are existing reviews with aspect ratings for machines to learn. Formally, we denote the review document as d containing a set of T_d sentences $\{s_1, s_2, \dots s_{T_d}\}$. For the t-th sentence s_t, we use a set of words $\{w_1, w_2, \dots w_{|s_t|}\}$ to represent it, and use $\mathbf{w}_i$, $\mathbf{w}_i^w$ and $\mathbf{w}_i^p$ as the one-hot encoding, word embedding, and phrase embedding for w_i respectively. The phrase embedding encodes the semantics of phrases where the current word w_i is the center (e.g., hidden vectors learned by Bi-LSTM shown in Section 2.2). For each q_k of K aspects

[1] Following the work (Weston et al., 2015; Sukhbaatar et al., 2015), we refer the memory to a set vectors which are stacked together and could be attended.

$\{q_1, q_2, \ldots, q_K\}$, we use N_k aspect-related keywords, $\left\{q_{k_1}, q_{k_2} \ldots q_{k_{N_k}}\right\}$, to represent it. Similarly, we use $\mathbf{q}_{k_i}$, $\mathbf{q}_{k_i}^w$ as the one-hot encoding and word embedding for q_{k_i} respectively.

There are several sophisticated methods for choosing aspect keywords (e.g., topic model). Here, we consider a simple way where five seeds were first manually selected for each aspect and then more words were obtained based on their cosine similarities with seeds[2]

As shown in Figure 2 (left), our framework follows the idea of multi-task learning, which learns different aspects simultaneously. In this case, all these tasks share the representations of words and architecture of semantic model for the final classifiers. Different from straightforward neural network based multi-task learning (Collobert et al., 2011), for each document d and an aspect q_k, our model uses both the content of d and all the related keywords $\left\{q_{k_1}, q_{k_2} \ldots q_{k_{N_k}}\right\}$ as input. Since the keywords can cover most of the semantic meanings of the aspect, and we do not know which document mentions which semantic meaning, we build an attention model to automatically decide it (introduced in Section 2.3). Assuming that the keywords have been decided, we use a hierarchical attention model to select useful information from the review documents. As shown in Figure 2 (right), the hierarchical attention of keywords is applied to both sentence level (to select meaningful words) and document level (to select meaningful sentence). Thus, our model builds aspect-specific representations in a bottom-up manner.

Specifically, we obtain sentence representations $\left\{\mathbf{s}_1^k, \mathbf{s}_2^k, \ldots \mathbf{s}_T^k\right\}$ using the input encoder (Section 2.2) and iterative attention module (Section 2.3) at the word level. Then we take sentence representations and k-th aspect as input and apply the sentence-level input encoder and attention model to generate the document representation $\mathbf{d}_k$ for final classification. As shown in Figure 2 (right), the attention model is applied twice at different levels of the representation.

2.2 Input Encoder

The input module builds memory vectors for the iterative attention module and is performed both at word and sentence levels. For a document, it converts word sequence into word level memory $\mathbf{M}_w^d$ and sentence sequence into sentence level memory $\mathbf{M}_s^d$ respectively. For an aspect question q_k, it takes a set of aspect-specific words $\{q_{k_i}\}_{1\le i\le N_k}$ as input and derives word level memory $\mathbf{M}_w^q$ and sentence level memory $\mathbf{M}_s^q$.

To construct $\mathbf{M}_w^d$, we obtain word embeddings $\left\{\mathbf{w}_1^w, \mathbf{w}_2^w, \ldots \mathbf{w}_{|s_t|}^w\right\}$ from an embedding matrix $\mathbf{E}^A$ applied to all words shown in the corpus. Then, LSTM (Hochreiter and Schmidhuber, 1997) model is used as the encoder to produce hidden vectors of words based on the word embeddings. At each step, LSTM takes input $\mathbf{w}_t^w$ and derives a new hidden vector by $\mathbf{h}_t = \text{LSTM}(\mathbf{w}_t^w, \mathbf{h}_{t-1})$. To preserve the subsequent context information for words, another LSTM is ran over word sequence in a reverse order simultaneously. Then the forward hidden vector $\overrightarrow{\mathbf{h}}_t$ and backward hidden vector $\overleftarrow{\mathbf{h}}_t$ are concatenated as phrase embedding $\mathbf{w}_t^p$. We stack these phrase embeddings together as word level memory $\mathbf{M}_w^d$. Similarly, we feed sentence representations into another Bi-LSTM to derive the sentence level memory $\mathbf{M}_s^d$. Note that, the sentence representations are obtained using the iterative attention module which is described as Eq. (5) in Section 2.3.

Since we have question keywords as input, to allow the interactions between questions and documents, we also build question memory in following way. We obtain $\mathbf{Q}_k = \left\{\mathbf{q}_{k_i}^w\right\}_{1\le i\le N_k}$ by looking up an embedding matrix [3] $\mathbf{E}^B$ applied to all question keywords. Then a non-linear mapping is applied to obtain the question memory at word level:

$$\mathbf{M}_w^{q_k} = \tanh(\mathbf{Q}_k \mathbf{W}_w^q), \tag{1}$$

where $\mathbf{W}_w^q$ is the parameter matrix to adapt q_k at word level. Similarly, we use another mapping to obtain the sentence level memory:

$$\mathbf{M}_s^{q_k} = \tanh(\mathbf{Q}_k \mathbf{W}_s^q), \tag{2}$$

where $\mathbf{W}_s^q$ is the parameter matrix to adapt q_k at sentence level.

2.3 Iterative Attention Module

The iterative attention module (IAM) attends and reads memories of questions and documents alternatively with a multi-hop mechanism, deriving

[2] For example, the words “value,” “price,” “worth,” “cost,” and “\$” are selected as seeds for aspect *Price*. The information for seeds can be found in our released resource.

[3] $\mathbf{E}^A$ and $\mathbf{E}^B$ are initialized by the same pre-trained embeddings but are different embedding matrices with different updates.

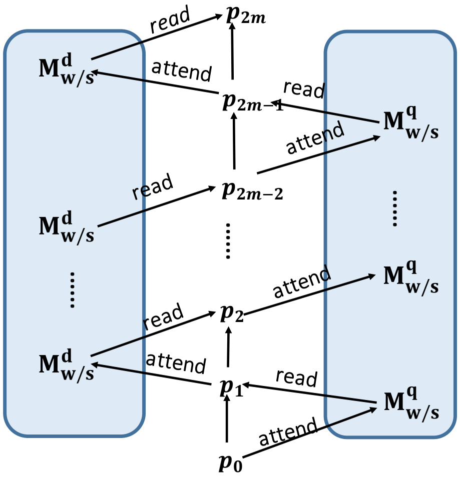


Figure 3: The iterative attention module.

aspect-specific sentence and document representations. As we discussed in the introduction, the set of selected question keywords may not best characterize the aspect for different documents. Thus, the IAM module introduces a backward attention to use document information (word or sentence) to select useful keywords of each aspect as the document-specific question to build attention model.

The illustration of IAM is shown in Figure 3. To obtain sentence representations, it takes $\mathbf{M}^d_w$ and $\mathbf{M}^q_w$ as the input and performs m iterations (hops). For each iteration, IAM conducts four operations: (1) attends the question memory by the selective vector $\mathbf{p}$ and summarizes question memory vectors into a single vector $\hat{\mathbf{q}}$; (2) updates the selective vector by the previous one and $\hat{\mathbf{q}}$; (3) attends document (content) memory based on the updated selective vector and summarizes memory vectors in to a single vector $\hat{\mathbf{c}}$; (4) updates the selective vector by the previous one and $\hat{\mathbf{c}}$.

We unify operations (1) and (3) by an attention function $\hat{\mathbf{x}} = \mathcal{A}(\mathbf{p}, \mathbf{M})$, where $\mathbf{M}$ could be $\mathbf{M}^d_w$ or $\mathbf{M}^q_w$ which corresponds $\hat{\mathbf{x}} = \hat{\mathbf{c}}$ or $\hat{\mathbf{x}} = \hat{\mathbf{q}}$. The attention function $\mathcal{A}$ is decomposed as:

$$\begin{aligned}\mathbf{H} &= \tanh(\mathbf{M}\mathbf{W}_a \odot (\mathbb{1}\mathbf{p})) \\ \mathbf{a} &= \text{softmax}(\mathbf{H}\mathbf{v}_a^T) \\ \hat{\mathbf{x}} &= \sum \mathbf{a}_i \mathbf{M}_i,\end{aligned} \tag{3}$$

where $\mathbb{1}$ is a vector with all elements are 1, which copies the selective vector to meet the dimension requirement. The $\mathbf{W}_a$ and $\mathbf{v}_a$ are parameters, $\mathbf{a}$ is attention weights for memory vectors, and $\mathbf{M}_i$ means i-th row in $\mathbf{M}$.

Operations (2) and (4) are formulated as an update function $\mathbf{p}_{2i-\{l\}} = \mathcal{U}(\hat{\mathbf{x}}, \mathbf{p}_{2i-\{l\}-1})$, where i is the hop index, l can be 0 or 1 which corresponds to $\hat{\mathbf{x}} = \hat{\mathbf{c}}$ or $\hat{\mathbf{x}} = \hat{\mathbf{q}}$ respectively. We initialize $\mathbf{p}_0$ by a zero vector. The update function $\mathcal{U}$ can be a recurrent neural network (Xiong et al., 2017) or other heuristic weighting functions. In this paper, we introduce a simple strategy:

$$\mathbf{p}_{2i-\{l\}} = \hat{\mathbf{x}}, \tag{4}$$

which ignores the previous selective vector but succeeds to obtain comparable results with other more complicated function in the initial experiments.

Multi-hop mechanism attends different memory locations in different hops (Sukhbaatar et al., 2015), capturing different interactions between documents and questions. In order to preserve the information of various kinds of interactions, we concatenate all $\hat{\mathbf{c}}$'s in each hop as the final representations of sentences:

$$\mathbf{s} = [\hat{\mathbf{c}}_1; \hat{\mathbf{c}}_2; \cdots \hat{\mathbf{c}}_m]. \tag{5}$$

After obtaining sentence representations, we feed them into the sentence-level input encoder, deriving the memories $\mathbf{M}^d_s$ and $\mathbf{M}^q_s$. Then, the aspect-specific document representation $\mathbf{d}_k$ is obtained by the sentence-level IAM in a similar way.

2.4 Objective Function

For each aspect, we obtain aspect-specific document representations $\{\mathbf{d}_k\}_{1 \le k \le K}$. All these representations are fed into classifiers, each of which includes a softmax layer. The softmax layer outputs the probability distribution over $|\mathcal{Y}|$ categories for the distributed representation, which is defined as:

$$\mathbf{p}'(d, k) = \text{softmax}(\mathbf{W}^{class}_k \mathbf{d}_k), \tag{6}$$

where $\mathbf{W}^{class}_k$ is the parameter matrix.

We define the cross-entropy objective function between gold sentiment distribution $\mathbf{p}(d, k)$ and predicted sentiment distribution $\mathbf{p}'(d, k)$ as the classification loss function:

$$-\sum_{d \in \mathcal{D}} \sum_{k=1}^{K} \sum_{i=1}^{|\mathcal{Y}|} \mathbf{p}(d, k) \log(\mathbf{p}'(d, k)), \tag{7}$$

where $\mathbf{p}(d, k)$ is a one-hot vector, which has the same dimension as the number of classes, and only the dimension associated with the ground truth label is one, with others being zeros.

Dataset	#docs	#words/doc	#words/sent
TripAdvisor	29,391	251.7	18.0
BeerAdvocate	51,020	144.5	12.1

Table 1: Statistics of the datasets. The rating scale of TripAdvisor dataset is 1-5. The rating scale of BeerAdvocate dataset is 1-10.

3 Experiment

In this section, we show experimental results to demonstrate our proposed algorithm.

3.1 Datasets

We conduct our experiments on TripAdvisor (Wang et al., 2010) and BeerAdvocate (McAuley et al., 2012; Lei et al., 2016) datasets, which contain seven aspects (*value*, *room*, *location*, *cleanliness*, *check in/front desk*, *service*, and *business service*) and four aspects (*feel*, *look*, *smell*, and *taste*) respectively. We follow the processing step (Lei et al., 2016) by choosing the reviews with different aspect ratings and the new datasets are described in Table 1. We tokenize the datasets by Stanford corenlp[4] and randomly split them into training, development, and testing sets with 80/10/10%.

3.2 Baseline Methods

To demonstrate the effectiveness of the proposed method, we compare our model with following baselines:

Majority uses the majority sentiment label in development sets as the predicted label.

SVM uses unigram and bigram as text features and uses Liblinear (Fan et al., 2008) for learning.

SLDA refers to supervised latent Dirichlet allocation (Blei and Mcauliffe, 2010) which is a statistical model of labeled documents.

NBoW is a neural bag-of-words model averaging embeddings of all words in a document and feeds the resulted embeddings into SVM classifier.

DAN is a deep averaging network model which consists of several fully connected layers with averaged word embeddings as input. One novel word dropout strategy is employed to boost model performances (Iyyer et al., 2015).

CNN continuously performs a convolution operation over a sentence to extract words neighboring features, then gets a fixed-sized representation by a pooling layer (Kim, 2014).

LSTM is one variant of recurrent neural network and has been proved to be one of state-of-the-art models for document-level sentiment classification (Tang et al., 2015a). We use LSTM to refer Bi-LSTM which captures both forward and backward semantic information.

HAN means the hierarchical attention network which is proposed in (Yang et al., 2016) for document classification. Note that, the original *HAN* depends GRU as the encoder. In our experiments, LSTM-based HAN obtains slightly better results. Thus, we report the results of HAN with LSTM as the encoder.

We extend *DAN*, *CNN*, *LSTM* with the hierarchical architecture and multi-task framework, the corresponding models are *MHDAN*, *MHCNN* and *MHLSTM* respectively. Besides, *MHAN* is also evaluated as one baseline, which is *HAN* with the multi-task learning.

3.3 Implementation Details

We implement all neural models using Theano (Theano Development Team, 2016). The model parameters are tuned based on the development sets. We learn 200-dimensional word embeddings with Skip-gram model (Mikolov et al., 2013) on in-domain corpus, which follows (Tang et al., 2015a). The pre-trained word embeddings are used to initialize the embedding matrices $\mathbf{E}^A$ and $\mathbf{E}^B$. The dimensions of all hidden vectors are set to 200. For TripAdvisor dataset, the hop numbers of word-level and sentence-level iterative attention modules are set to 4 and 2 respectively. For BeerAdvocate dataset, the hop numbers are set to 6 and 2. The number of selected keywords $N_k = N$ is set to 20. To avoid model over-fitting, we use dropout and regularization as follows: (1) the regularization parameter is set to *1e-5*; (2) the dropout rate is set to 0.3, which is applied to both sentence and document vectors. All parameters are trained by ADADELTA (Zeiler, 2012) without needing to set the initial learning rate. To ensure fair comparisons, we make baselines have same settings as the proposed model, such as word embeddings, dimensions of hidden vectors and optimization details and so on.

3.4 Results and Analyses

We use accuracy and mean squared error (MSE) as the evaluation metrics and the results are shown in Table 2.

[4] http://nlp.stanford.edu/software/corenlp.shtml

Model	TripAdvisor				BeerAdvocate			
	Dev		Test		Dev		Test	
	Accuracy	MSE	Accuracy	MSE	Accuracy	MSE	Accuracy	MSE
Majority	24.47	2.533	23.89	2.549	24.48	4.706	24.41	4.545
SVM	34.30	1.982	35.26	1.963	25.70	3.286	25.79	3.270
SLDA	31.58	2.131	32.81	2.110	25.39	3.372	25.73	3.391
NBoW	38.43	1.866	39.09	1.808	28.99	2.883	28.85	2.919
DAN	40.30	1.569	40.93	1.531	31.25	2.569	32.44	2.279
CNN	43.25	1.474	43.35	1.456	34.17	2.173	33.37	2.217
LSTM	43.85	1.525	44.02	1.470	35.23	2.112	34.78	2.097
HAN	44.47	1.312	44.68	1.301	36.57	1.903	36.03	1.920
MHDAN	42.22	1.554	42.47	1.549	32.76	2.358	32.54	2.376
MHCNN	44.19	1.329	43.79	1.398	36.10	1.966	35.33	1.976
MHLSTM	44.53	1.308	44.72	1.272	38.14	1.785	37.04	1.809
MHAN	44.72	1.294	44.94	1.210	37.98	1.783	36.82	1.813
Our	46.21	1.091	**46.56**	**1.083**	39.43	1.696	**38.06**	**1.755**

Table 2: Comparison of our model and other baseline methods.

Model	TripAdvisor		BeerAdvocate	
	Accuracy	MSE	Accuracy	MSE
MHLSTM	44.75 (0.24)	1.256 (0.05)	37.28 (0.43)	1.802 (0.17)
MHAN	45.02 (0.33)	1.221 (0.12)	37.02 (0.22)	1.810 (0.15)
Our	**46.65**†(0.29)	**1.084***(0.06)	**38.25**†(0.35)	**1.749***(0.18)

Table 3: The results of average accuracy/MSE and standard deviation of models on test sets. We choose *MHAN* and *MHLSTM* as comparison baselines for TripAdvisor and BeerAdvocate respectively. In t-tests, the marker * refers to p-value < 0.05 and the marker † refers to p-value < 0.01.

Compared to SVM and SLDA, NBoW achieves higher accuracy by 3% in both datasets, which shows that embedding features are more effective than traditional ngram features on these two datasets. All neural network models outperform NBoW. It shows the advantages of neural networks in the document sentiment classification.

From the results of neural networks, we can observe that DAN performs worse than LSTM and CNN, and LSTM achieves slightly higher results than CNN. It can be explained that the simple composition method averaging embeddings of words in a document but ignoring word order, may not be as effective as other flexible composition models, such as LSTM and CNN, on aspect classification. Additionally, we observe that the multi-task learning and hierarchical architecture are beneficial for neural networks. Among all baselines, MHAN and MHLSTM achieve comparable results and outperform others.

Compared with MHAN and MHLSTM, our method achieves improvements of 1.5% (3% relative improvement) and 1.0% (2.5% relative improvement) on TripAdvisor and BeerAdvocate respectively, which shows that the incorporation of iterative attention mechanism helps the deep neural network based model build up more discriminative aspect-aware representation. Note that BeerAdvocate is relatively more difficult since the predicted ratings are from 1 to 10 while TripAdvisor is 1 to 5. Moreover, t-test is conducted by randomly splitting datasets into train/dev/test sets and random initialization. The results on test sets are described in Table 3 which show performance of our model is stable.

3.5 Case Study for Attention Results

In this section, we sample two sentences from TripAdvisor to show the visualization of attention results for case study. Both word-level and sentence-level attention visualizations are shown in Figure 4. We normalize the word weight by the sentence weight to make sure that only important words in a document are highlighted.

From the top figures in (a) and (b), we observe that our model assigns different attention weights for each aspect. For example, in the first sentence, the words *comfortable* and *bed* are assigned higher

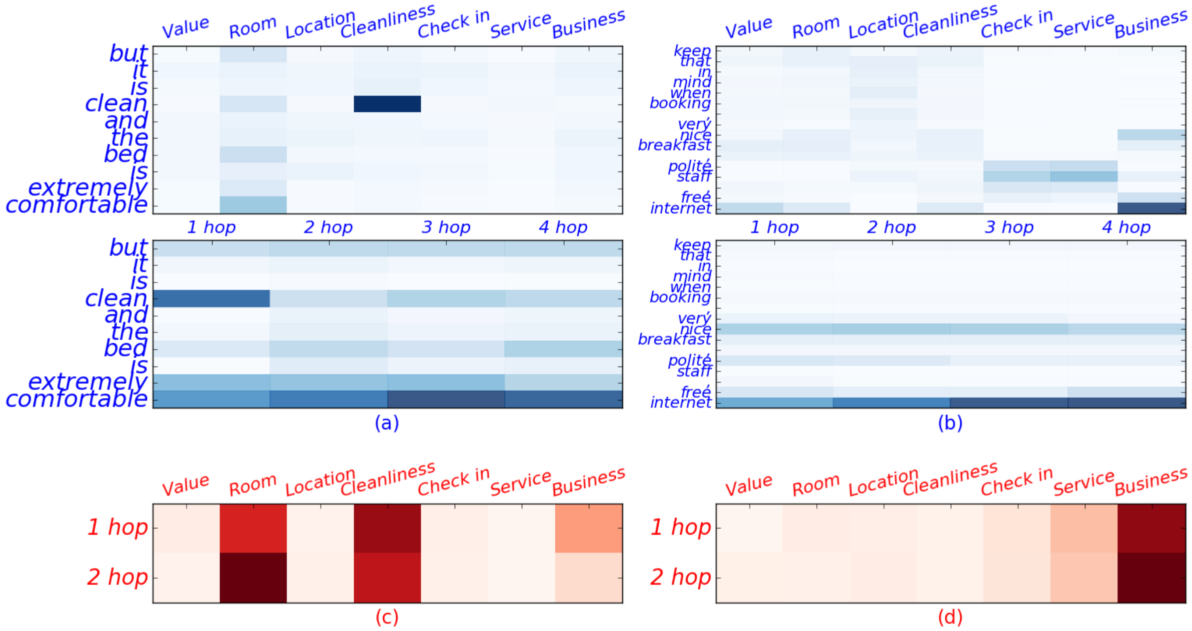


Figure 4: The attention visualization of words and sentences. Darker color means higher weight. (a) and (b) are the visualization of word weights; (c) and (d) are the visualization of sentence weights. The top figures in (a) and (b) show the word weights of fourth hop for each aspect. The bottom figures in (a) and (b) visualize the word weights of different hops for the aspects *Room* and *Business* respectively.

weights in the aspect *Room*, and the word *clean* are highlighted by the aspect *Cleaniness*. In the second sentence, the word *internet* is assigned a high attention value for *Business*. Moreover, the bottom figures in (a) and (b) show that (1) word weights of different hops are various; (2) attention values in higher hop are more reasonable. Specifically, in the first sentence, the weight of word *clean* is higher than the word *comfortable* in first hop, while *comfortable* surpasses *clean* in higher hops. In the second sentence, we observe that the value of word *internet* increases with the number of hop. Thus, we can see that the more sensible weights are obtained for words through the proposed iterative attention mechanism. Similarly, the figures (c) and (d) show that the conclusion from words is also suitable for sentences. For the first sentence, the sentence weight regarding the aspect *Room* is lower than *Cleanliness* in the first hop, but surpasses *Cleanliness* in the second hop. For the second sentence, the weight for *Business* becomes higher in the second hop.

3.6 Effects of Hop and Aspect Keywords

In this experiment, we investigate the effects of hop number m and size of aspect keywords N on performances. All the experiments are conducted on the development set. Due to lack of space, we only present the results of TripAdvisor and the results of BeerAdvocate have a similar behavior as TripAdvisor.

For the hop number, we vary m from 1 to 7 and the results are shown in Figure 5 (left). We can see that: (1) at the word level, the performance increases when $m \leq 4$, but shows no improvement after $m > 4$; (2) at the sentence level, model performs best when $m = 2$. Moreover, we can see that the hop number of word level leads to larger variation than the hop number of sentence level.

For the size of aspect keywords, we vary N from 0 to 35, incremented by 5. Note that, we set a learnable vector to represent question memory when $N = 0$. The results are shown in Figure 5 (right). We observe that the performance increases when $N \leq 20$, and has no improvement after $N > 20$. This indicates that a small number of keywords can help the proposed model achieve competitive results.

4 Related Work

Multi-Aspect Sentiment Classification. Multi-aspect sentiment classification has been studied extensively in literature. Lu et al. (2011) used support vector regression model based on hand-

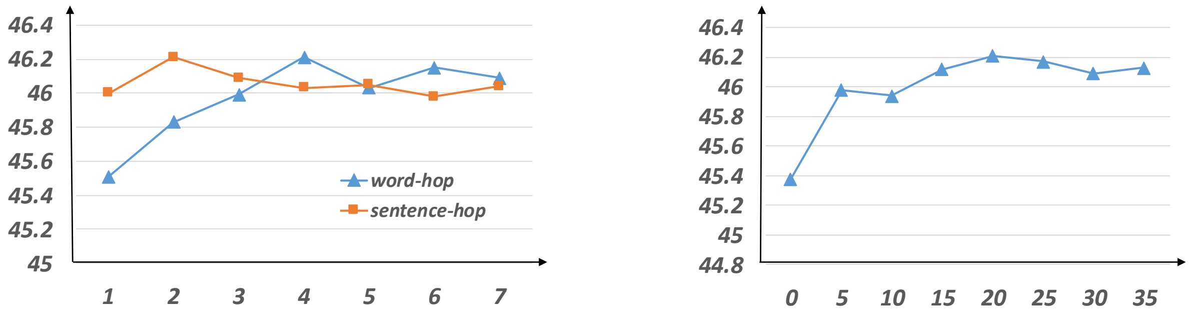


Figure 5: Results of different hops and different sizes of question keywords. Left: different the hop numbers; Right: different sizes of keywords.

crafted features to predict aspect ratings. To handle the correlation between aspects, McAuley et al. (2012) added a dependency term in final multi-class SVM objective. There were also some heuristic based methods and sophisticated topic models where multi-aspect sentiment classification is solved as a subproblem (Titov and McDonald, 2008; Wang et al., 2010; Diao et al., 2014; Pappas and Popescu-Belis, 2014). However, these approaches often rely on strict assumptions about words and sentences, for example, using the word syntax to determine if a word is an aspect or a sentiment word, or relating a sentence with an specific aspect. Another related problem is called aspect-based sentiment classification (Pontiki et al., 2014, 2016; Poria et al., 2016), which first extracts aspect expressions from sentences (Poria et al., 2014; Balahur and Montoyo, 2008; Chen et al., 2014, 2013), and then determines their sentiments. With the developments of neural networks and word embeddings in NLP, neural network based models have shown the state-of-the-art results with less feature engineering work. Tang et al. (2016) employed a deep memory network for aspect-based sentiment classification given the aspect location and Lakkaraju et al. (2014) employed recurrent neural networks and its variants for the task of extraction of aspect-sentiment pair. However, these tasks are sentence-level. Another related research field is document-level sentiment classification because we can treat single aspect sentiment classification as an individual document classification task. This line of research includes (Tang et al., 2015b; Chen et al., 2016; Tang et al., 2016; Yang et al., 2016) which are based on neural networks in a hierarchical structure. However, they did not work on multiple aspects.

Machine Comprehension. Recently, neural network based machine comprehension (or reading) has been studied extensively in NLP, with the releases of large-scale evaluation datasets (Hermann et al., 2015; Hill et al., 2016; Rajpurkar et al., 2016). Most of the related studies focus on attention mechanism (Bahdanau et al., 2014) which is firstly proposed in machine translating and aims to solve the long-distance dependency between words. Hermann et al. (2015) used Bi-LSTM to encode document and query, and proposed Attentive Reader and Impatient Reader. The first one attends document based on the query representation, and the second one attends document by the representation of each token in query with an incremental manner. Memory Networks (Weston et al., 2015; Sukhbaatar et al., 2015) attend and reason document representation in a multi-hop fashion, enriching interactions between documents and questions. Dynamic Memory Network (Kumar et al., 2016) updates memories of documents by re-running GRU models based on derived attention weights. Meanwhile, the query representation is refined by another GRU model. Gated-Attention Reader (Dhingra et al., 2016) proposes a novel attention mechanism, which is based on multiplicative interactions between the query embeddings and the intermediate states of a recurrent neural network document reader. Bi-Directional Attention Model (Xiong et al., 2017; Seo et al., 2017) fuses co-dependent representations of queries and documents in order to focus on relevant parts of both. Iterative Attention model (Sordoni et al., 2016) attends question and document sequentially, which is related to our model. Different from Iterative Attention model, our model focuses on the document-level multi-aspect sentiment classification, which is proposed in a hierarchical architecture and has different procedures in the iterative attention module. Another related research problem is visual question answering which uses an image as question context rather than a set of keywords as question. Neural network based visual question answering (Lu et al., 2016; Xiong et al., 2016) is similar as the proposed models in text comprehension.

5 Conclusion

In this paper, we model the document-level multi-aspect sentiment classification as a text comprehension problem and propose a novel hierarchical iterative attention model in which documents and pseudo aspect-questions are interleaved at both word and sentence-level to learn aspect-aware document representation in a unified model. Extensive experiments show that our model outperforms the other neural models with multi-task framework and hierarchical architecture.

6 Acknowledgments

This paper is partially supported by the National Natural Science Foundation of China (NSFC Grant Nos. 61472006 and 91646202) as well as the National Basic Research Program (973 Program No. 2014CB340405). This work was also supported by NVIDIA Corporation with the donation of the Titan X GPU, Hong Kong CERG Project 26206717, China 973 Fundamental R&D Program (No.2014CB340304), and the LORELEI Contract HR0011-15-2-0025 with DARPA. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. We also thank the anonymous reviewers for their valuable comments and suggestions that help improve the quality of this manuscript.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Alexandra Balahur and Andres Montoyo. 2008. A feature dependent method for opinion mining and classification. In *Natural Language Processing and Knowledge Engineering*. pages 1–7.

David M. Blei and Jon D. Mcauliffe. 2010. Supervised topic models. *Advances in Neural Information Processing Systems* 3:327–332.

Rich Caruana. 1997. Multitask learning. *Machine Learning* 28(1):41–75.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Proceedings of EMNLP*. pages 1650–1659.

Zhiyuan Chen, Arjun Mukherjee, and Bing Liu. 2014. Aspect extraction with automated prior knowledge learning. In *ACL*. pages 347–358.

Zhiyuan Chen, Arjun Mukherjee, Bing Liu, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. 2013. Exploiting domain knowledge in aspect extraction. In *EMNLP*. pages 1655–1667.

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

Bhuwan Dhingra, Hanxiao Liu, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549* .

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of KDD*. ACM, pages 193–202.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research* 9(Aug):1871–1874.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of NIPS*. pages 1693–1701.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of ICLR*.

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

Mohit Iyyer, Varun Manjunatha, Jordan L Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of ACL*. pages 1681–1691.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*. pages 1746–1751.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of ICML*.

Himabindu Lakkaraju, Richard Socher, and Chris Manning. 2014. Aspect specific sentiment analysis using hierarchical deep learning. In *NIPS Workshop on Deep Learning and Representation Learning*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of EMNLP*. Austin, Texas, pages 107–117.

Bing Liu. 2010. Sentiment analysis and subjectivity. In *Handbook of Natural Language Processing, Second Edition.*, pages 627–666.

Bin Lu, Myle Ott, Claire Cardie, and Benjamin K Tsou. 2011. Multi-aspect sentiment analysis with topic models. In *ICDM Workshops*. IEEE, pages 81–88.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Proceedings of NIPS*. pages 289–297.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of ICLR*.

Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning attitudes and attributes from multi-aspect reviews. In *Proceedings of ICDM*. IEEE, pages 1020–1025.

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

Bo Pang and Lillian Lee. 2007. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval* 2(1-2):1–135.

Nikolaos Pappas and Andrei Popescu-Belis. 2014. Explaining the stars: Weighted multiple-instance learning for aspect-based sentiment analysis. In *Proceedings of EMNLP*. pages 455–466.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphee De Clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of SemEval*. pages 19–30.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of SemEval*. pages 27–35.

Soujanya Poria, Erik Cambria, Lun-Wei Ku, Chen Gui, and Alexander Gelbukh. 2014. A rule-based approach to aspect extraction from product reviews. In *Proceedings of the second workshop on natural language processing for social media (SocialNLP)*. pages 28–37.

Soujanya Poria, Iti Chaturvedi, Erik Cambria, and Federica Bisio. 2016. Sentic lda: Improving on lda with semantic similarity for aspect-based sentiment analysis. In *International Joint Conference on Neural Networks*. pages 4465–4473.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*. pages 2383–2392.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *Proceedings of ICLR* .

Alessandro Sordoni, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245* .

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Proceedings of NIPS*. pages 2440–2448.

Duyu Tang, Bing Qin, and Ting Liu. 2015a. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*. pages 1422–1432.

Duyu Tang, Bing Qin, and Ting Liu. 2015b. Learning semantic representations of users and products for document level sentiment classification. In *ACL*. pages 1014–1023.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Proceedings of EMNLP*. pages 214–224.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688. http://arxiv.org/abs/1605.02688.

Ivan Titov and Ryan T McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL*. Citeseer, volume 8, pages 308–316.

Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of KDD*. ACM, pages 783–792.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of ICLR*.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *Proceedings of ICML*. pages 1378–1387.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. *Proceedings of ICLR* .

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL-HLT*. pages 1480–1489.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

What is the Essence of a Claim? Cross-Domain Claim Identification

Johannes Daxenberger[†], **Steffen Eger**[†‡], **Ivan Habernal**[†], **Christian Stab**[†], **Iryna Gurevych**[†‡]
[†]Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
[‡]Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research and Educational Information
http://www.ukp.tu-darmstadt.de

Abstract

Argument mining has become a popular research area in NLP. It typically includes the identification of argumentative components, e.g. claims, as the central component of an argument. We perform a qualitative analysis across six different datasets and show that these appear to conceptualize claims quite differently. To learn about the consequences of such different conceptualizations of claim for practical applications, we carried out extensive experiments using state-of-the-art feature-rich and deep learning systems, to identify claims in a cross-domain fashion. While the divergent conceptualization of claims in different datasets is indeed harmful to cross-domain classification, we show that there are shared properties on the lexical level as well as system configurations that can help to overcome these gaps.

1 Introduction

The key component of an argument is the *claim*. This simple observation has not changed much since the early works on argumentation by Aristotle more than two thousand years ago, although argumentation scholars provide us with a plethora of often clashing theories and models (van Eemeren et al., 2014). Despite the lack of a precise definition in the contemporary argumentation theory, Toulmin's influential work on argumentation in the 1950's introduced a claim as an 'assertion that deserves our attention' (Toulmin, 2003, p. 11); recent works describe a claim as 'a statement that is in dispute and that we are trying to support with reasons' (Govier, 2010).

Argument mining, a computational counterpart of manual argumentation analysis, is a recent growing sub-field of NLP (Peldszus and Stede, 2013a). 'Mining' arguments usually involves several steps like separating argumentative from non-argumentative text units, parsing argument structures, and recognizing argument components such as claims—the main focus of this article. Claim identification itself is an important prerequisite for applications such as fake checking (Vlachos and Riedel, 2014), politics and legal affairs (Surdeanu et al., 2010), and science (Park and Blake, 2012).

Although claims can be identified with a promising level of accuracy in typical argumentative discourse such as persuasive essays (Stab and Gurevych, 2014; Eger et al., 2017), less homogeneous resources, for instance online discourse, pose challenges to current systems (Habernal and Gurevych, 2017). Furthermore, existing argument mining approaches are often limited to a single, specific domain like legal documents (Mochales-Palau and Moens, 2009), microtexts (Peldszus and Stede, 2015), Wikipedia articles (Levy et al., 2014; Rinott et al., 2015) or student essays (Stab and Gurevych, 2017). The problem of generalizing systems or features and their robustness across heterogeneous datasets thus remains fairly unexplored.

This situation motivated us to perform a detailed analysis of the concept of claims (as a key component of an argument) in existing argument mining datasets from different domains.[1] We first review and qualitatively analyze six existing publicly available datasets for argument mining (§3), showing that the conceptualizations of claims in these datasets differ largely. In a next step, we analyze the influence of these differences for cross-domain claim identification. We propose several computational models for claim identification,

[1] We take the machine learning perspective in which different *domains* mean data drawn from different distributions (Murphy, 2012, p. 297).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2055–2066
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

including systems using linguistically motivated features (§4.1) and recent deep neural networks (§4.2), and rigorously evaluate them on and across all datasets (§5). Finally, in order to better understand the factors influencing the performance in a cross-domain scenario, we perform an extensive quantitative analysis on the results (§6).

Our analysis reveals that despite obvious differences in conceptualizations of claims across datasets, there are some shared properties on the lexical level which can be useful for claim identification in heterogeneous or unknown domains. Furthermore, we found that the choice of the source (training) domain is crucial when the target domain is unknown. We release our experimental framework to help other researchers build upon our findings.[2]

2 Related Work

Existing approaches to argument mining can be roughly categorized into (*a*) *multi-document* approaches which recognize claims and evidence across several documents and (*b*) *discourse level* approaches addressing the argumentative structure within a single document. Multi-document approaches have been proposed e.g. by Levy et al. (2014) and Rinott et al. (2015) for mining claims and corresponding evidence for a predefined topic over multiple Wikipedia articles. Nevertheless, to date most approaches and datasets deal with single-document argumentative discourse. This paper takes the discourse level perspective, as we aim to assess multiple datasets from different authors and compare their notion of 'claims'.

Mochales-Palau and Moens (2009) experiment at the discourse level using feature-rich SVM and a hand-crafted context-free grammar in order to recognize claims and premises in legal decisions. Their best results for claims achieve 74.1% F_1 using domain-dependent key phrases, token counts, location features, information about verbs, and the tense of the sentence. Peldszus and Stede (2015) present an approach based on a minimum spanning tree algorithm and model the global structure of arguments considering argumentative relations, the stance and the function of argument components. Their approach yields 86.9% F_1 for recognizing claims in English 'microtexts'. Habernal and Gurevych (2017) cast argument component identification as BIO sequence labeling and jointly model separation of argumentative from non-argumentative text units and identification of argument component boundaries together with their types. They achieved 25.1% Macro-F_1 with a combination of topic, sentiment, semantic, discourse and embedding features using structural SVM. Stab and Gurevych (2014) identified claims and other argument components in student essays. They experiment with several classifiers and achieved the best performance of 53.8% F_1 score using SVM with structural, lexical, syntactic, indicator and contextual features. Although the above-mentioned approaches achieve promising results in particular domains, their ability to generalize over heterogeneous text types and domains remains unanswered.

Rosenthal and McKeown (2012) set out to explore this direction by conducting cross-domain experiments for detecting claims in blog articles from LiveJournal and discussions taken from Wikipedia. However, they focused on relatively similar datasets that both stem from the social media domain and in addition annotated the datasets themselves, leading to an identical conceptualization of the notion of claim. Although Al-Khatib et al. (2016) also deal with cross-domain experiments, they address a different task; namely identification of argumentative sentences. Further, their goals are different: they want to improve argumentation mining via distant supervision rather than detecting differences in the notions of a claim.

Domain adaptation techniques (Daume III, 2007) try to address the frequently observed drop in classifier performances entailed by a dissimilarity of training and test data distributions. Since techniques such as learning generalized cross-domain representations in an unsupervised manner (Blitzer et al., 2006; Pan et al., 2010; Glorot et al., 2011; Yang and Eisenstein, 2015) have been criticized for targeting specific source and target domains, it has alternatively been proposed to learn *universal* representations from general domains in order to render a learner robust across *all* possible domain shifts (Müller and Schütze, 2015; Schnabel and Schütze, 2013). Our approach is in a similar vein. However, rather than trying to improve classifier performances for a specific source-target domain pair, we want to detect *differences* between these pairs. Furthermore, we are looking

[2] https://github.com/UKPLab/emnlp2017-claim-identification

Corpus	*Reference*	*Genre*	*#Docs*	*#Tokens*	*#Sentences*	*#Claims*
VG	Reed et al. (2008)	various genres	507	60,383	2,842	563 (19.81%)
WD	Habernal and Gurevych (2015)	web discourse	340	84,817	3,899	211 (5.41%)
PE	Stab and Gurevych (2017)	persuasive essays	402	147,271	7,116	2,108 (29.62%)
OC	Biran and Rambow (2011a)	online comments	2,805	125,677	8,946	703 (7.86%)
WTP	Biran and Rambow (2011b)	wiki talk pages	1,985	189,140	9,140	1,138 (12.45%)
MT	Peldszus and Stede (2015)	micro texts	112	8,865	449	112 (24.94%)

Table 1: Overview of the employed corpora.

for *universal* feature sets or classifiers that perform generally well for claim identification across varying source and target domains.

3 Claim Identification in Computational Argumentation

We briefly describe six English datasets used in our empirical study; they all capture claims on the discourse level. Table 1 summarizes the dataset statistics relevant to claim identification.

3.1 Datasets

The AraucariaDB corpus (Reed et al., 2008) includes various genres (**VG**) such as newspaper editorials, parliamentary records, or judicial summaries. The annotation scheme structures arguments as trees and distinguishes between claims and premises at the clause level. Although the reliability of the annotations is unknown, the corpus has been extensively used in argument mining (Moens et al., 2007; Feng and Hirst, 2011; Rooney et al., 2012).

The corpus from Habernal and Gurevych (2017) includes user-generated web discourse (**WD**) such as blog posts, or user comments annotated with claims and premises as well as backings, rebuttals and refutations (α_U 0.48) inspired by Toulmin's model of argument (Toulmin, 2003).

The persuasive essay (**PE**) corpus (Stab and Gurevych, 2017) includes 402 student essays. The scheme comprises major claims, claims and premises at the clause level (α_U 0.77). The corpus has been extensively used in the argument mining community (Persing and Ng, 2015; Lippi and Torroni, 2015; Nguyen and Litman, 2016).

Biran and Rambow (2011a) annotated claims and premises in online comments (**OC**) from blog threads of LiveJournal (κ 0.69). In a subsequent work, Biran and Rambow (2011b) applied their annotation scheme to documents from Wikipedia talk pages (**WTP**) and annotated 118 threads. For our experiments, we consider each user comment in both corpora as a document, which yields $2,805$ documents in the OC corpus and $1,985$ documents in the WTP corpus.

Peldszus and Stede (2016) created a corpus of German microtexts (**MT**) of controlled linguistic and rhetoric complexity. Each document includes a single argument and does not exceed five argument components. The scheme models the argument structure and distinguishes between premises and claims, among other properties (such as proponent/opponent or normal/example). In the first annotation study, 26 untrained annotators annotated 23 microtexts in a classroom experiment (κ 0.38) (Peldszus and Stede, 2013b). In a subsequent work, the corpus was largely extended by expert annotators (κ 0.83). Recently, they translated the corpus to English, resulting in the first parallel corpus in computational argumentation; our experiments rely on the English version.

3.2 Qualitative Analysis of Claims

In order to investigate how claim annotations are tackled in the chosen corpora, one co-author of this paper manually analyzed 50 randomly sampled claims from each corpus. The characteristics taken into account are drawn from argumentation theory (Schiappa and Nordin, 2013) and include among other things the claim type, signaling words and discourse markers.

Biran and Rambow (2011b) do not back-up their claim annotations by any common argumentation theory but rather state that claims are *utterances which convey subjective information and anticipate the question 'why are you telling me that?'* and need to be supported by justifications. Using this rather loose definition, a claim might be any subjective statement that is justified by the author. Detailed examination of the LiveJournal corpus (OC) revealed that sentences with claims are extremely noisy. Their content ranges from a single word, (*"Bastard."*), to emotional expressions of personal regret, (*"::hugs:: i am so sorry hon .."*)

to general Web-chat nonsense (*"W-wow... that's a wicked awesome picture... looks like something from Pirates of the Caribbean...gone Victorian ...lolz."*) or posts without any clear argumentative purpose (*"what i did with it was make this recipe for a sort of casserole/stratta (i made this up, here is the recipe) [...] butter, 4 eggs, salt, pepper, sauted onions and cabbage..add as much as you want bake for 1 hour at 350 it was seriously delicious!"*). The Wikipedia Talk Page corpus (WTP) contains claims typical to Wikipedia quality discussions (*"That is why this article has NPOV issues."*) and policy claims (Schiappa and Nordin, 2013) are present as well (*"I think the gallery should be got rid of altogether."*). However, a small number of nonsensical claims remains (*"A dot."*).

Analysis of the MT dataset revealed that about half of claim sentences contain the modal verb *'should'*, clearly indicating policy claims (*"The death penalty should be abandoned everywhere."*). Such statements also very explicitly express the stance on the controversial topic of interest. In a similar vein, claims in persuasive students' essays (PE) heavily rely on phrases signaling beliefs (*"In my opinion, although using machines have many benefits, we cannot ignore its negative effects."*) or argumentative discourse connectors whose usage is recommended in textbooks on essay writing (*"Thus, it is not need for young people to possess this ability."*). Most claims are value/policy claims written in the present tense.

The mixture of genres in the AraucariaDB corpus (VG) is reflected in the variety of claims. While some are simple statements starting with a discourse marker (*"Therefore, 10% of the students in my logic class are left-handed."*), there are many legal-specific claims requiring expert knowledge (*"In considering the intention of Parliament when passing the 1985 Act, or perhaps more properly the intention of the draftsman in settling its terms, there are [...]"*), reported and direct speech claims (*"Eight-month-old Kyle Mutch's tragic death was not an accident and he suffered injuries consistent with a punch or a kick, a court heard yesterday."*), and several nonsensical claims (*"RE: Does the Priest Scandal Reveal the Beast?"*) which undercut the consistency of this dataset.

The web-discourse (WD) claims take a clear stance to the relevant controversy (*"I regard single sex education as bad."*), yet sometimes anaphoric (*"My view on the subject is no."*). The usage of discourse markers is seldom. Habernal and Gurevych (2017) investigated hedging in claims and found out that it varies with respect to the topic being discussed (10% up to 35% of claims are hedged). Sarcasm or rhetorical question are also common (*"In 2013, is single-sex education really the way to go?"*).

These observations make clear that annotating claims—the central part of all arguments, as suggested by the majority of argumentation scholars—can be approached very differently when it comes to actual empirical, data-driven operationalization. While some traits are shared, such as that claims usually need some support to make up a 'full' argument (e.g., premises, evidence, or justifications), the exact definition of a claim can be arbitrary—depending on the domain, register, or task.

4 Methodology

Given the results from the qualitative analysis, we want to investigate whether the different conceptualizations of claims can be assessed empirically and if so, how they could be dealt with in practice. Put simply, the task we are trying to solve in the following is: *given a sentence, classify whether or not it contains a claim.* We opted to model the claim identification task on sentence level, as this is the only way to make all datasets compatible to each other. Different datasets model claim boundaries differently, e.g. MT includes discourse markers within the same sentence, whereas they are excluded in PE.

All six datasets described in the previous section have been preprocessed by first segmenting documents into sentences using Stanford CoreNLP (Manning et al., 2014) and then annotating every sentence as claim, if one or more tokens within the sentence were labeled as claim (or major claim in PE). Analogously, each sentence is annotated as non-claim, if none of its tokens were labeled as claim (or major claim). Although our basic units of interest are sentences, we keep the content of the entire document to be able to retrieve information about the context of (non-)claims.[3]

We are not interested in optimizing the properties of a certain learner for this task, but rather

[3] This is true only for the feature-based learners. The neural networks do not have access to information beyond individual sentences.

want to compare the influence of different types of lexical, syntactical, and other kinds of information across datasets.[4] Thus, we used a limited set of learners for our task: a) a standard L2-regularized logistic regression approach with manually defined feature sets[5], which is a simple yet robust and established technique for many text classification problems (Plank et al., 2014; He et al., 2015; Zhang et al., 2016a; Ferreira and Vlachos, 2016); and b) several deep learning approaches, using state-of-the-art neural network architectures.

The **in-domain experiments** were carried out in a 10-fold cross-validation setup with fixed splits into training and test data. As for the **cross-domain experiments**, we train on the entire data of the source domain and test on the entire data of the target domain. In the domain adaptation terminology, this corresponds to an unsupervised setting.

To address class-imbalance in our datasets (see Table 1), we downsample the negative class (non-claim) both in-domain and cross-domain, so that positive and negative class occur approximately in an 1:1 ratio in the training data. Since this means that we discard a lot of useful information (many negative instances), we repeat this procedure 20 times, in each case randomly discarding instances of the negative class such that the required ratio is obtained. At test time, we use the majority prediction of this ensemble of 20 trained models. With the exception of very few cases, this led to consistent performance improvements across all experiments. The systems are described in more detail in the following subsections. Additionally, we report the results of two baselines. The majority baseline labels all sentences as non-claims (predominant class in all datasets), the random baseline labels sentences as claims with 0.5 probability.

4.1 Linguistically Motivated Features

For the logistic regression-based experiments (LR) we employed the following feature groups. *Structure Features* capture the position, the length and the punctuation of a sentence. *Lexical Features* are lowercased unigrams. *Syntax Features* account for grammatical information at the sentence level. We include information about the part-of-speech and parse tree for each sentence. *Discourse Features* encode information extracted with help of the Penn Discourse Treebank (PDTB) styled end-to-end discourse parser as presented by Lin et al. (2014). *Embedding Features* represent each sentence as a summation of its word embeddings (Guo et al., 2014). We further experimented with sentiment features (Habernal and Gurevych, 2015; Anand et al., 2011) and dictionary features (Misra et al., 2015; Rosenthal and McKeown, 2015) but these delivered very poor results and are not reported in this article. The full set of features and their parameters are described in the supplementary material to this article. We experiment with the full feature set, individual feature groups, and feature ablation (all features except for one group).

4.2 Deep Learning Approaches

As alternatives to our feature-based systems, we consider three deep learning approaches. The first is the Convolutional Neural Net of Kim (Kim, 2014) which has shown to perform excellently on many diverse classification tasks such as sentiment analysis and question classification and is still a strong competitor among neural techniques focusing on sentence classification (Komninos and Manandhar, 2016; Zhang et al., 2016b,c). We consider two variants of Kim's CNN, one in which words' vectors are initialized with pre-trained GoogleNews word embeddings (CNN:w2vec) and one in which the vectors are randomly initialized and updated during training (CNN:rand). Our second model is an LSTM (long short-term memory) neural net for sentence classification (LSTM) and our third model is a bidirectional LSTM (BiLSTM).

For all neural network classifiers, we use default hyperparameters concerning hidden dimensionalities (for the two LSTM models), number of filters (for the convolutional neural net), and others. We train each of the three neural networks for 15 iterations and choose in each case the learned model that performs best on a held-out development set of roughly 10% of the training data as the model to apply to unseen test data. This corresponds to an early stopping regularization scheme.

5 Results

In the following, we summarize the results of the various learners described above. Obtaining all results required heavy computation, e.g. the cross-

[4] For the same reason, we do not optimize any hyperparameters for individual learners, unless explicitly stated.

[5] Using the liblinear library (Fan et al., 2008).

Target → System ↓	**MT**		**OC**		**PE**		**VG**		**WD**		**WTP**		**Average**	
							neural network models							
BiLSTM	68.8	41.8	58.0	22.4	73.0	62.0	60.9	37.7	60.0	24.5	57.9	28.5	63.1	36.1
CNN:rand	78.6	67.3	**60.5**	**25.6**	**73.6**	61.1	**65.9**	**45.0**	61.1	25.8	58.6	28.9	66.4	42.3
CNN:w2vec	73.7	60.9	58.2	23.7	74.0	61.7	63.8	33.5	62.6	**28.9**	57.3	24.3	64.9	38.8
LSTM	65.2	48.3	58.5	22.3	71.8	60.7	61.3	40.1	61.6	25.9	58.0	28.4	62.7	37.6
LR							*feature ablation and combination*							
-Discourse	73.0	60.8	59.9	22.9	70.6	60.6	62.5	42.6	63.7	23.2	59.7	30.2	64.9	40.0
-Embeddings	74.6	62.9	59.6	22.6	70.4	60.4	62.9	43.1	63.9	23.5	59.4	29.9	65.1	40.4
-Lexical	72.1	59.5	59.6	22.5	65.9	55.1	60.8	40.5	60.1	18.5	57.7	27.8	62.7	37.3
-Structure	74.4	62.6	60.0	23.0	70.4	60.4	62.0	41.8	64.2	23.4	59.5	30.0	65.1	40.2
-Syntax	**79.8**	**70.3**	59.8	22.9	72.1	**62.5**	63.4	43.8	**65.1**	25.5	**60.1**	**30.5**	**66.7**	**42.6**
All Features	74.4	62.7	59.9	22.9	70.6	60.6	62.5	42.6	63.8	23.3	59.7	30.2	65.1	40.4
LR							*single feature groups*							
+Discourse	70.0	56.7	49.4	13.8	50.1	41.7	49.6	30.6	57.6	14.9	49.5	18.4	54.4	29.3
+Embeddings	72.4	59.8	58.8	20.8	68.2	57.7	59.7	39.3	64.2	23.8	59.0	28.9	63.7	38.4
+Lexical	75.9	64.7	59.5	21.4	71.8	62.1	61.1	40.5	64.0	22.2	59.0	27.7	65.2	39.8
+Structure	57.1	42.0	56.5	20.0	54.2	39.5	55.4	33.3	48.4	9.0	55.4	25.2	54.5	28.2
+Syntax	66.7	52.5	58.1	21.0	64.1	52.9	60.7	40.4	57.6	15.5	57.0	27.0	60.7	34.9
							baselines							
Majority bsl	42.9	0.0	48.0	0.0	41.3	0.0	44.5	0.0	48.6	0.0	46.7	0.0	45.3	0.0
Random bsl	50.7	33.2	49.9	13.5	50.8	38.0	50.4	28.8	51.6	10.8	48.9	18.8	50.4	23.9

Table 2: In-domain experiments, best values per column are highlighted. For each dataset (column head) we show two scores: *Macro-F_1* score (left-hand column) and F_1 score for claims (right-hand column).

validation experiments for feature-based systems took 56 days of computing. We intentionally do not list the results of previous work on those datasets. The scores are not comparable since we strictly work on sentence level (rather than e.g. clause level) and applied downsampling to the training data. All reported significance tests were conduced using two-tailed Wilcoxon Signed-Rank Test for matched pairs, i.e. paired scores of F_1 scores from two compared systems (Japkowicz and Shah, 2014).

5.1 In-Domain Experiments

The performance of the learners is quite divergent across datasets, with Macro-F_1 scores[6] ranging from 60% (WTP) to 80% (MT), average 67% (see Table 2). On all datasets, our best systems clearly outperform both baselines. In isolation, lexical, embedding, and syntax features are most helpful, whereas structural features did not help in most cases. Discourse features only contribute significantly on MT. When looking at the performance of the feature-based approaches, the most striking finding is the importance of lexical (in our setup, unigram) information.

The average performances of LR$_{-syntax}$ and CNN:rand are virtually identical, both for Macro-F_1 and Claim-F_1, with a slight advantage for the feature-based approach, but their difference is not statistically significant ($p \leq 0.05$). Altogether, these two systems exhibit significantly better average performances than all other models surveyed here, both those relying on and those not relying on hand-crafted features ($p \leq 0.05$). The absence or the different nature of inter-annotator agreement measures for all datasets prevent us from searching for correlations between agreement and performance. But we observed that the systems yield better results on PE and MT, both datasets with good inter-annotator agreement ($\alpha_u = 0.77$ for PE and $\kappa = 0.83$ for MT).

5.2 Cross-Domain Experiments

For all six datasets, training on different sources resulted in a performance drop. Table 3 lists the results of the best feature-based (*LR All features*) and deep learning (*CNN:rand*) systems, as well as single feature groups (averages over all source domains, results for individual source domains can be found in the supplementary material to this article). We note the biggest performance drops on the datasets which performed best in the in-domain setting (MT and PE). For the lowest scoring datasets, OC and WTP, the differences are only marginal when trained on a suitable dataset (VG

[6] Described as $Fscore_M$ in Sokolova and Lapalme (2009).

Target → Source/Sys. ↓	**MT**		**OC**		**PE**		**VG**		**WD**		**WTP**		**Average**	
							CNN:rand							
MT	*78.6*	*67.3*	51.0	7.4	56.9	22.1	57.2	15.7	52.4	9.4	49.4	10.9	53.4	13.1
OC	57.1	39.7	*60.5*	*25.6*	56.4	42.8	58.9	37.3	54.6	13.2	**58.4**	**28.9**	57.1	32.4
PE	59.8	18.0	54.2	9.5	*73.6*	*61.1*	57.5	18.7	**55.5**	**15.9**	54.7	16.0	56.3	15.6
VG	**68.7**	**51.5**	55.8	19.2	**57.0**	32.0	*65.9*	*45.0*	51.7	10.5	54.7	22.0	57.6	27.0
WD	64.4	3.5	51.3	1.3	41.3	0.0	44.5	0.0	*61.1*	*25.8*	46.7	0.0	49.6	1.0
WTP	58.5	26.6	56.8	15.4	56.0	18.5	55.3	19.4	52.9	11.6	*58.6*	*28.9*	55.9	18.3
Average	61.7	27.9	53.8	10.6	53.5	23.1	54.7	18.2	53.4	12.1	52.8	15.6	55.0	17.9
							LR All features							
MT	*74.4*	*62.7*	53.9	17.0	51.9	29.5	56.1	34.2	55.1	14.5	52.5	21.2	53.9	23.3
OC	60.0	45.1	*59.9*	*22.9*	56.7	**47.0**	58.6	**38.0**	54.1	12.2	57.7	27.5	57.4	**34.0**
PE	58.1	36.3	54.6	17.3	*70.6*	*60.6*	54.1	21.4	54.0	13.5	54.4	20.4	55.0	21.8
VG	65.8	51.4	**57.3**	**21.7**	**57.0**	45.1	*62.5*	*42.6*	54.5	13.1	55.1	24.8	**57.9**	31.2
WD	62.6	38.5	55.4	19.0	56.0	30.1	55.1	23.3	*63.8*	*23.3*	53.6	20.9	56.5	26.3
WTP	58.0	41.7	56.1	20.3	56.8	42.6	**59.1**	**38.0**	52.2	11.2	*59.7*	*30.2*	56.5	30.8
Average	60.9	42.6	55.5	19.1	55.7	38.9	56.6	31.0	54.0	12.9	54.7	23.0	56.2	27.9
LR							*single feature groups (averages across all source domains)*							
+Discourse	40.2	15.0	31.7	5.8	30.3	27.4	27.7	19.9	40.9	4.5	25.3	13.3	32.7	14.3
+Embeddings	56.6	35.2	51.4	12.8	53.6	30.7	53.3	24.3	54.2	13.2	52.9	19.0	53.7	22.5
+Lexical	61.0	42.2	55.2	18.3	56.2	38.6	54.7	29.1	53.1	11.9	54.9	23.4	55.9	27.2
+Structure	44.2	22.9	53.6	18.5	52.5	38.4	53.6	32.1	49.1	9.0	53.4	23.3	51.1	24.0
+Syntax	54.8	37.0	54.2	17.5	54.3	40.6	55.7	32.0	53.0	11.8	53.8	22.5	54.3	26.9
							baselines							
Majority bsl	42.9	0.0	48.0	0.0	41.3	0.0	44.5	0.0	48.6	0.0	46.7	0.0	45.3	0.0
Random bsl	47.5	30.6	50.5	14.0	51.0	38.4	51.0	29.3	49.3	9.3	50.3	20.2	49.9	23.6

Table 3: Cross-domain experiments, best values per column are highlighted, in-domain results (for comparison) in italics; results only for selected systems. For each source/target combination we show two scores: *Macro-F_1* score (left-hand column) and F_1 score for claims (right-hand column).

and OC, respectively). The best feature-based approach outperforms the best deep learning approach in most scenarios. In particular, as opposed to the in-domain experiments, the difference of the Claim-F_1 measure between the feature-based approaches and the deep learning approaches is striking. In the feature-based approaches, on average, a combination of all features yields the best results for both Macro-F_1 and Claim-F_1. When comparing single features, lexical ones do the best job.

Looking at the best overall system (LR with all features), the average test results when training on different source datasets are between 54% Macro-F_1 resp. 23% Claim-F_1 (both MT) and 58% (VG) resp. 34% (OC). Depending on the goal that should be achieved, training on VG (highest average Macro-F_1) or OC (highest average Claim-F_1) seems to be the best choice when the domain of test data is unknown (we analyze this finding in more depth in §6). MT clearly gives the best results as target domain, followed by PE and VG.

We also performed experiments with mixed sources, the results are shown in Table 4. We did this in a leave-one-domain-out fashion, in particular we trained on all but one datasets and tested on the remaining one. In this scenario, the neural network systems seem to benefit from the increased amount of training data and thus gave the best results. Overall, the mixed sources approach works better than many of the single-source cross-domain systems – yet, the differences were not found to be significant, but as good as training on suitable single sources (see above).

6 Further Analysis and Discussion

To better understand which factors influence cross-domain performance of the systems we tested, we considered the following variables as potential determinants of outcome: similarity between source and target domain, the source domain itself, training data size, and the ratio between claims and non-claims.

We calculated the Spearman correlation of the top-500 lemmas between the datasets in each direction, see results in Table 5. The most similar domains are OC (source s) and WTP (target t), coming from the same authors. OC (s) and WD (t) as well OC (s) and VG (t) are also highly cor-

Target → System ↓	**MT**		**OC**		**PE**		**VG**		**WD**		**WTP**		**Avg**	
CNN:rand	62.8	41.4	**57.8**	**22.4**	**59.7**	36.2	**58.6**	28.1	**54.2**	**14.1**	**56.8**	25.6	**58.3**	28.0
All features	**64.7**	**49.5**	56.4	20.6	57.8	**45.8**	58.2	**36.4**	52.3	11.3	56.0	**26.0**	57.6	**31.6**
Majority bsl	42.9	0.0	48.0	0.0	41.3	0.0	44.5	0.0	48.6	0.0	46.7	0.0	45.3	0.0
Random bsl	47.5	30.6	50.5	14.0	51.0	38.4	51.0	29.3	49.3	9.3	50.3	20.2	49.9	23.6

Table 4: Leave-one-domain-out experiments, best values per column are highlighted. For each test dataset (column head) we show two scores: *Macro-F_1* score (left-hand column) and F_1 score for claims (right-hand column).

	MT	OC	PE	VG	WD	WTP
MT	100	47	51	52	49	48
OC	56	100	55	68	71	71
PE	59	58	100	66	67	57
VG	51	58	52	100	59	62
WD	54	61	61	62	100	55
WTP	49	59	49	57	57	100

Table 5: Heatmap of Spearman correlations in % based on most frequent 500 lemmas for each dataset. Source domain: rows, target domain: columns.

related. For a statistical test of potential correlations between cross-domain performances and the introduced variables, we regress the cross-domain results (Table 3) on Table 5 (T4 in the following equation), on the number of claims #C (directly related to training data size in our experiments, effect of downsampling), and on the ratio of claims to non-claims R.[7] More precisely, given source/training data and target data pairs (s,t) in Table 3, we estimate the linear regression model

$$y_{st} = \alpha \cdot \mathrm{T4}_{st} + \beta \cdot \log(\#\mathrm{C}_s) + \gamma \cdot \mathrm{R}_t + \varepsilon_{st}, \quad (1)$$

where y_{st} denotes the Macro-F_1 score when training on s and testing on t. In the regression, we also include binary dummy variables $\mathbb{1}_\sigma = \mathbb{1}_{s,\sigma}$ for each domain σ whose value is 1 if $s = \sigma$ (and 0 otherwise). These help us identify "good" source domains.

The coefficient α for Table 5 is not statistically significantly different from zero in any case. Ultimately, this means that it is difficult to predict cross-domain performance from lexical similarity of the datasets. This is in contrast to e.g., POS tagging, where lexical similarity has been reported to predict cross-domain performance very well (Van Asch and Daelemans, 2010). The coefficient for training data size β is statistically significantly different from zero in three out of 15 cases. In particular, it is significantly positive in two (CNN:rand, CNN:w2vec) out of four cases for the neural networks. This indicates that the neural networks would have particularly benefited from more training data, which is confirmed by the improved performance of the neural networks in the mixed sources experiments (cf. §5.2). The ratio of claims to non-claims in t is among the best predictors for the variables considered here (coefficient γ is significant in three out of 15 cases, but consistently positive). This is probably due to our decision to balance training data (downsampling non-claims) to keep the assessment of claim identification realistic for real-world applications, where the class ratio of t is unknown. Our systems are thus inherently biased towards a higher claim ratio.

Finally, the dummy variables for OC and VG are three times significantly positive, but consistently positive overall. Their average coefficient is 2.31 and 1.90, respectively, while the average coefficients for all other source datasets is negative, and not significant in most cases. Thus, even when controlling for all other factors such as training data size and the different claim ratios of target domains, OC and VG are the best source domains for cross-domain claim classification in our experiments. OC and VG are particularly good training sources for the detection of claims (as opposed to non-claims)—the minority class in all datasets—as indicated by the average Claim-F_1 scores in Table 3.

One finding that was confirmed both in-domain as well as cross-domain was the importance of lexical features as compared to other feature groups. As mere lexical similarity between domains does not explain performance (cf. coefficient α above), this finding indicated that the learners relied on

[7] Overall, we had 15 different systems, see upper 15 rows in Table 2. Therefore, we had 15 different regression models.

a few, but important lexical clues. To go more into depth, we carried out error analysis on the CNN:rand cross-domain results. We used OC, VG and PE as source domains, and MT and WTP as target domains. By examining examples in which a model trained on OC and VG made correct predictions as opposed to a model trained on PE, we quickly noticed that lexical indicators indeed played a crucial role. In particular, the occurrence of the word "should" (and to a lower degree: "would", "article", "one") are helpful for the detection of claims across various datasets. In MT, a simple baseline labeling every sentence containing "should" as claim achieves 76.1 Macro-F_1 (just slightly below the best in-domain system on this dataset). In the other datasets, this phenomenon is far less dominant, but still observable. We conclude that a few rather simple rules (learned by models trained on OC and VG, but not by potentially more complex models trained on PE) make a big difference in the cross-domain setting.

7 Conclusion

In a rigorous empirical assessment of different machine learning systems, we compared how six datasets model claims as the fundamental component of an argument. The varying performance of the tested in-domain systems reflects different notions of claims also observed in a qualitative study of claims across the domains. Our results reveal that the best in-domain system is not necessarily the best system in environments where the target domain is unknown. Particularly, we found that mixing source domains and training on two rather noisy datasets (OC and VG) gave the best results in the cross-domain setup. The reason for this seem to be a few important lexical indicators (like the word "should") which are learned easier under these circumstances. In summary, as for the six datasets we analyzed here, our analysis shows that the essence of a claim is not much more than a few lexical clues.

From this, we conclude that future work should address the problem of vague conceptualization of claims as central components of arguments. A more consistent notion of claims, which also holds across domains, would potentially not just benefit cross-domain claim identification, but also higher-level applications relying on argumentation mining (Wachsmuth et al., 2017). To further overcome the problem of domain dependence, multi-task learning is a framework that could be explored (Søgaard and Goldberg, 2016) for different conceptualizations of claims.

Acknowledgments

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the promotional reference 03VP02540 (ArgumenText), by the GRK 1994/1 AIPHES (DFG), and by the ArguAna Project GU 798/20-1 (DFG).

References

Khalid Al-Khatib, Henning Wachsmuth, Matthias Hagen, Jonas Köhler, and Benno Stein. 2016. Cross-Domain Mining of Argumentative Text through Distant Supervision. In *15th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1395–1404, Berlin, Germany.

Pranav Anand, Marilyn Walker, Rob Abbott, Jean E. Fox Tree, Robeson Bowmani, and Michael Minor. 2011. Cats rule and dogs drool!: Classifying stance in online debate. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis WASSA 2011*, pages 1–9, Portland, OR, USA.

Or Biran and Owen Rambow. 2011a. Identifying justifications in written dialogs. In *Fifth IEEE International Conference on Semantic Computing (ICSC)*, pages 162–168, Palo Alto, CA, USA.

Or Biran and Owen Rambow. 2011b. Identifying justifications in written dialogs by classifying text as argumentative. *International Journal of Semantic Computing*, 05(04):363–381.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, Sydney, Australia.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic.

Frans H. van Eemeren, Bart Garssen, Erik C. W. Krabbe, A. Francisca Snoeck Henkemans, Bart Verheij, and Jean H. M. Wagemans. 2014. *Handbook of Argumentation Theory*. Springer, Berlin/Heidelberg.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural End-to-End Learning for Computational Argumentation Mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, page to appear, Vancouver, Canada.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 987–996, Portland, OR, USA.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1163–1168, San Diego, CA, USA.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning*, pages 513–520, Bellevue, WA, USA.

Trudy Govier. 2010. *A Practical Study of Argument*, 7th edition. Wadsworth, Cengage Learning.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting Embedding Features for Simple Semi-supervised Learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 110–120, Doha, Qatar.

Ivan Habernal and Iryna Gurevych. 2015. Exploiting debate portals for semi-supervised argumentation mining in user-generated web discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2127–2137, Lisbon, Portugal.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation Mining in User-Generated Web Discourse. *Computational Linguistics*, 43(1):125–179.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-Answer Driven Semantic Role Labeling: Using Natural Language to Annotate Natural Language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 643–653, Lisbon, Portugal.

Nathalie Japkowicz and Mohak Shah. 2014. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, Cambridge, UK.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency based embeddings for sentence classification tasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1490–1500, San Diego, California.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 1489–1500, Dublin, Ireland.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A PDTB-Styled End-to-End Discourse Parser. *Natural Language Engineering*, 20(2):151–184.

Marco Lippi and Paolo Torroni. 2015. Context-independent claim detection for argument mining. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 185–191, Buenos Aires, Argentina.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, MA, USA.

Amita Misra, Pranav Anand, Jean Fox Tree, and Marilyn Walker. 2015. Using summarization to discover argument facets in online idealogical dialog. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 430–440, Denver, CO, USA.

Raquel Mochales-Palau and Marie-Francine Moens. 2009. Argumentation mining: The detection, classification and structure of arguments in text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 98–107, Barcelona, Spain.

Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. 2007. Automatic detection of arguments in legal texts. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law*, pages 225–230, Stanford, CA, USA.

Thomas Müller and Hinrich Schütze. 2015. Robust morphological tagging with word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 526–536, Denver, CO, USA.

Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA, USA.

Huy Nguyen and Diane Litman. 2016. Improving argument mining in student essays by learning and exploiting argument indicators versus essay topics. In *Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference*, pages 485–490, Key Largo, FL, USA.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th International Conference on World Wide Web*, pages 751–760, Raleigh, NC, USA.

Dae Hoon Park and Catherine Blake. 2012. Identifying comparative claim sentences in full-text scientific articles. In *Proceedings of the Workshop on Detecting Structure in Scholarly Discourse*, pages 1–9, Jeju, Republic of Korea.

Andreas Peldszus and Manfred Stede. 2013a. From Argument Diagrams to Argumentation Mining in Texts: A Survey. *International Journal of Cognitive Informatics and Natural Intelligence*, 7(1):1–31.

Andreas Peldszus and Manfred Stede. 2013b. Ranking the annotators: An agreement study on argumentation structure. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 196–204, Sofia, Bulgaria.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in mst-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948, Lisbon, Portugal.

Andreas Peldszus and Manfred Stede. 2016. An annotated corpus of argumentative microtexts. In *Argumentation and Reasoned Action: Proceedings of the 1st European Conference on Argumentation*, pages 801–815, Lisbon, Portugal.

Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 543–552, Beijing, China.

Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014. Linguistically debatable or just plain wrong? In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 507–511, Baltimore, MA, USA.

Chris Reed, Raquel Mochales-Palau, Glenn Rowe, and Marie-Francine Moens. 2008. Language resources for studying argument. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*, pages 2613–2618, Marrakech, Morocco.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence - an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 440–450, Lisbon, Portugal.

Niall Rooney, Hui Wang, and Fiona Browne. 2012. Applying kernel methods to argumentation mining. In *Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference*, pages 272–275, Marco Island, FL, USA.

Sara Rosenthal and Kathleen McKeown. 2012. Detecting opinionated claims in online discussions. In *Proceedings of the 2012 IEEE Sixth International Conference on Semantic Computing*, pages 30–37, Washington, DC, USA.

Sara Rosenthal and Kathy McKeown. 2015. I couldn't agree more: The role of conversational structure in agreement and disagreement detection in online discussions. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 168–177, Prague, Czech Republic.

Edward Schiappa and John P. Nordin. 2013. *Argumentation: Keeping Faith with Reason*, 1st edition. Pearson UK.

Tobias Schnabel and Hinrich Schütze. 2013. Towards robust cross-domain domain adaptation for part-of-speech tagging. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 198–206, Nagoya, Japan.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 231–235, Berlin, Germany.

Marina Sokolova and Guy Lapalme. 2009. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 46–56, Doha, Qatar.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, pages in press, preprint available at arXiv:1604.07370.

Mihai Surdeanu, Ramesh Nallapati, and Christopher D. Manning. 2010. Legal claim identification: Information extraction with hierarchically labeled data. In *Workshop on the Semantic Processing of Legal Texts at LREC*, pages 22–29, Valletta, Malta.

Stephen E. Toulmin. 2003. *The Uses of Argument, Updated Edition*. Cambridge University Press, New York.

Vincent Van Asch and Walter Daelemans. 2010. Using domain similarity for performance estimation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 31–36, Uppsala, Sweden.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22, Baltimore, MD, USA.

Henning Wachsmuth, Benno Stein, and Yamen Ajjour. 2017. ”PageRank” for Argument Relevance. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1117–1127, Valencia, Spain.

Yi Yang and Jacob Eisenstein. 2015. Unsupervised multi-domain adaptation with feature embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 672–682, Denver, CO, USA.

Fan Zhang, Rebecca Hwa, Diane Litman, and Homa B. Hashemi. 2016a. ArgRewrite: A Web-based Revision Assistant for Argumentative Writings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 37–41, San Diego, CA, USA.

Rui Zhang, Honglak Lee, and Dragomir R. Radev. 2016b. Dependency sensitive convolutional neural networks for modeling sentences and documents. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1512–1521, San Diego, CA, USA.

Ye Zhang, Stephen Roller, and Byron C. Wallace. 2016c. MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1522–1527, San Diego, CA, USA.

Identifying Where to Focus in Reading Comprehension for Neural Question Generation

Xinya Du and **Claire Cardie**
Department of Computer Science
Cornell University
Ithaca, NY, 14853, USA
{xdu, cardie}@cs.cornell.edu

Abstract

A first step in the task of automatically generating questions for testing reading comprehension is to identify *question-worthy* sentences, i.e. sentences in a text passage that humans find it worthwhile to ask questions about. We propose a hierarchical neural sentence-level sequence tagging model for this task, which existing approaches to question generation have ignored. The approach is fully data-driven — with no sophisticated NLP pipelines or any hand-crafted rules/features — and compares favorably to a number of baselines when evaluated on the SQuAD data set. When incorporated into an existing neural question generation system, the resulting end-to-end system achieves state-of-the-art performance for paragraph-level question generation for reading comprehension.

1 Introduction and Related Work

Automatically generating questions for testing reading comprehension is a challenging task (Mannem et al., 2010; Rus et al., 2010). First and foremost, the question generation system must determine which concepts in the associated text passage are important, i.e. are worth asking a question about.

The little previous work that exists in this area currently circumvents this critical step in passage-level question generation by assuming that such sentences have already been identified. In particular, prior work focuses almost exclusively on *sentence-level* question generation: given a text passage, assume that all sentences contain a question-worthy concept and generate one or more questions for each (Heilman and Smith, 2010; Du et al., 2017; Zhou et al., 2017).

In contrast, we study the task of *passage-level question generation (QG)*. Inspired by the large body of research in text summarization on identifying sentences that contain "summary-worthy" content (e.g. Mihalcea (2005), Berg-Kirkpatrick et al. (2011), Yang et al. (2017)), we develop a method to identify the *question-worthy sentences* in each paragraph of a reading comprehension passage. Inspired further by the success of neural sequence models for many natural language processing tasks (e.g. named entity recognition (Collobert et al., 2011), sentiment classification (Socher et al., 2013), machine translation (Sutskever et al., 2014), dependency parsing (Chen and Manning, 2014)), including very recently document-level text summarization (Cheng and Lapata, 2016), we propose a hierarchical neural sentence-level sequence tagging model for question-worthy sentence identification.

We employ the SQuAD reading comprehension data set (Rajpurkar et al., 2016) for evaluation and show that our sentence selection approach compares favorably to a number of baselines including the feature-rich sentence selection model of Cheng and Lapata (2016) proposed in the context of extract-based summarization, and the convolutional neural network model of Kim (2014) that achieves state-of-the-art results on a variety of sentence classification tasks.

We also incorporate our sentence selection component into the neural question generation system of Du et al. (2017) and show, again using SQuAD, that our resulting end-to-end system achieves state-of-the-art performance for the challenging task of paragraph-level question generation for reading comprehension.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2067–2073
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

2 Problem Formulation

In this section, we define the tasks of *important* (i.e. question-worthy) *sentence selection* and *sentence-level question generation (QG)*. Our full paragraph-level QG system includes both of these components. For the sentence selection task, given a paragraph D consisting of a sequence of sentences $\{s_1, ..., s_m\}$, we aim to select a subset of k question-worthy sentences $(k < m)$. The goal is defined as finding $\overline{\mathbf{y}} = \{y_1, ..., y_m\}$, such that,

$$\begin{aligned}\overline{\mathbf{y}} &= \arg\max_{\mathbf{y}} \log P_1 (\mathbf{y}|D) \\ &= \arg\max_{\mathbf{y}} \sum_{t=1}^{|\mathbf{y}|} \log P_1 (y_t|D)\end{aligned} \quad (1)$$

where log $P(\mathbf{y}|D)$ is the conditional log-likelihood of the label sequence $\mathbf{y}$; and $y_i = 1$ means sentence i is question-worthy (contains at least one answer), otherwise $y_i = 0$.

For sentence-level QG, the goal is to find the best word sequence $\overline{\mathbf{z}}$ (a question of arbitrary length) that maximizes the conditional likelihood given the input sentence $\mathbf{x}$ and satisfies:

$$\begin{aligned}\overline{\mathbf{z}} &= \arg\max_{\mathbf{z}} \log P_2 (\mathbf{z}|\mathbf{x}) \\ &= \arg\max_{\mathbf{z}} \sum_{t=1}^{|\mathbf{z}|} \log P_2 (z_t|\mathbf{x}, z_{<t})\end{aligned} \quad (2)$$

where $P_2(\mathbf{z}|\mathbf{x})$ is modeled with a global attention mechanism (Section 3).

3 Model

Important Sentence Selection Our general idea for the hierarchical neural network architecture is illustrated in Figure 1. First, we perform the encoding using sum operation or convolution+maximum pooling operation (Kim, 2014; dos Santos and Zadrozny, 2014) over the word vectors comprising each sentence in the input paragraph. For simplicity and consistency, we denote the sentence encoding process as ENC. Given the t^{th} sentence $\mathbf{x} = \{x_1, ..., x_n\}$ in the paragraph, we have its encoding:

$$\mathbf{s}_t = \text{ENC}([x_1, ..., x_n]) \quad (3)$$

Then we use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to encode the paragraph,

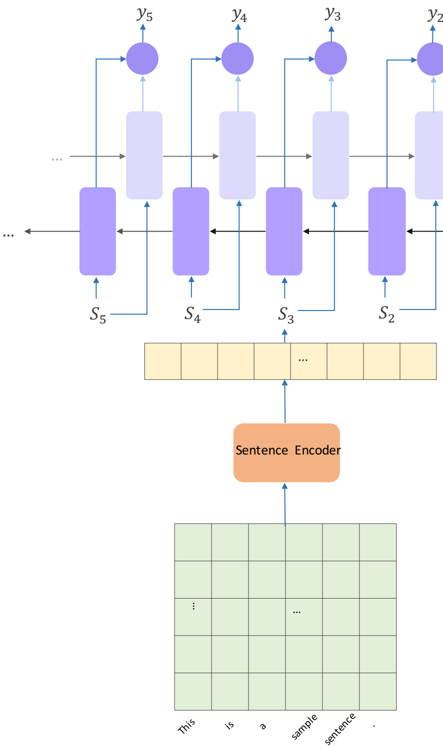


Figure 1: Hierarchical neural network architecture for sentence-level sequence labeling. The input is a paragraph consisting of sentences, whose encoded representation is fed into each hidden unit.

$$\begin{aligned}\overrightarrow{\mathbf{h}_t} &= \overrightarrow{\text{LSTM}} \left(\mathbf{s}_t, \overrightarrow{\mathbf{h}_{t-1}}\right) \\ \overleftarrow{\mathbf{h}_t} &= \overleftarrow{\text{LSTM}} \left(\mathbf{s}_t, \overleftarrow{\mathbf{h}_{t+1}}\right)\end{aligned}$$

We use the concatenation of the two, namely, $[\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]$, as the hidden state $\mathbf{h}_t$ at time stamp t, and feed it to the upper layers to get the probability distribution of y_t $(\in \{0, 1\})$,

$$P_1 (y_t|D; \theta) = \text{softmax} \left(\text{MLP} \left(\tanh \left([\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]\right)\right)\right)$$

where MLP is multi-layer neural network and tanh is the activation function.

Question Generation Similar to Du et al. (2017), we implement the sentence-level question generator with an attention-based sequence-to-sequence learning framework (Sutskever et al., 2014; Bahdanau et al., 2015), to map a sentence in the reading comprehension article to natural questions. It consists of an LSTM encoder and decoder. The encoder is a bi-directional LSTM network; it encodes the input sentence $\mathbf{x}$ into a sequence of hidden states $\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_{|\mathbf{x}|}$.

Model	Precision	Recall	F-measure	Acc.	Paragraph-level Acc.
RANDOM	63.45	50.29	56.11	50.27	11.69
Majority Baseline	63.21	100.00	77.46	63.21	32.30
CNN (Kim, 2014)	68.35	**90.13**	77.74	67.38	24.73
LREG$_{\text{(w/ BOW)}}$	68.52	86.55	76.49	66.37	31.36
LREG$_{\text{(w/ para.-level)}}$ (Cheng and Lapata, 2016)	70.49	89.08	78.70	69.52	33.95
Ours$_{\text{SUM}}$ (no pre-trained)	73.02	89.23	80.32	72.36	36.46
Ours$_{\text{SUM}}$ (w/ pre-trained)	73.85	87.65	80.16	**72.58**	36.30
Ours$_{\text{CNN}}$ (no pre-trained)	73.15	89.29	**80.42***	72.52	35.93
Ours$_{\text{CNN}}$ (w/ pre-trained)	**74.35**	86.11	79.80	72.44	**36.87**

Table 1: Automatic evaluation results for important sentence selection. The best performing system in each column is highlighted in boldface. Paragraph-level accuracies are calculated as the proportion of paragraphs in which *all* of the sentences are predicted correctly. We show two-tailed t-test results on F-measure for our best performing method compared to the other baselines. (Statistical significance is indicated with $^*(p < 0.005)$.)

The decoder is another LSTM that uses global attention over the encoder hidden states. The entire encoder-decoder structure learns the probability of generating a question given a sentence, as indicated by equation 2. To be more specific,

$$P_2\left(z_t|\mathbf{x}, z_{<t}\right) = \text{softmax}\left(\mathbf{W}_s\text{tanh}\left(\mathbf{W}_t[\mathbf{h}_t; \mathbf{c}_t]\right)\right)$$

where $\mathbf{W}_s$, $\mathbf{W}_t$ are parameter matrices; $\mathbf{h}_t$ is the hidden state of the decoder LSTM; and $\mathbf{c}_t$ is the context vector created dynamically by the encoder LSTM — the weighted sum of the hidden states computed for the source sentence:

$$\mathbf{c}_t = \sum_{i=1,..,|\mathbf{x}|} a_{i,t}\mathbf{q}_i$$

The attention weights $a_{i,t}$ are calculated via a bilinear scoring function and softmax normalization:

$$a_{i,t} = \frac{\exp(\mathbf{h}_t^T \mathbf{W}_b \mathbf{q}_i)}{\sum_j \exp(\mathbf{h}_t^T \mathbf{W}_b \mathbf{q}_j)}$$

Apart from the bilinear score, alternative options for computing the attention can also be used (e.g. dot product). Readers can refer to Luong et al. (2015) for more details.

During inference, beam search is used to predict the question. The decoded `UNK` token at time step t, is replaced with the token in the input sentence with the highest attention score, the index of which is $\arg\max_i a_{i,t}$.

Henceforth, we will refer to our sentence-level **N**eural **Q**uestion **G**eneration system as **NQG**. Note that generating answer-specific questions would be easy for this architecture — we can append answer location features to the vectors of tokens in the sentence. To better mimic the real life case (where questions are generated with no prior knowledge of the desired answers), we do not use such location features in our experiments.

4 Experimental Setup and Results

4.1 Dataset and Implementation Details

We use the SQuAD dataset (Rajpurkar et al., 2016) for training and evaluation for both important sentence selection and sentence-level NQG. The dataset contains 536 curated Wikipedia articles with over 100k questions posed about the articles. The authors employ Amazon Mechanical Turk crowd-workers to generate questions based on the article paragraphs and to annotate the corresponding answer spans in the text. Later, to make the evaluation of the dataset more robust, other crowd-workers are employed to provide additional answers to the questions.

We split the public portion of the dataset into training ($\sim$80%), validation ($\sim$10%) and test ($\sim$10%) sets at the paragraph level. For the sentence selection task, we treat sentences that contain at least one answer span (question-worthy sentences) as positive examples ($y = 1$); all remaining sentences are considered negative ($y = 0$). Not surprisingly, the training set is unbalanced: 52332 ($\sim$60%) sentences contain answers, while 29693 sentences do not. Because of the variabil-

	Model	BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR
Conservative	LREG(C&L) + NQG	38.30	23.15	15.64	10.97	15.09
	Ours + NQG	40.08	24.26	16.39	11.50	15.67
Liberal	LREG(C&L) + NQG	51.55	40.17	34.35	30.59	24.17
	Ours + NQG	52.89	41.16	35.15	31.25	24.76

Table 2: Results for the full QG systems using BLEU 1–4, METEOR. The first stage of the two pipeline systems are the feature-rich linear model (LREG) and our best performing selection model respectively.

ity of human choice in generating questions, it is the case that many sentences labeled as negative examples might actually contain concepts worth asking a question about. For the related important sentence detection task in text summarization, Yang et al. (2017) therefore propose a two-stage approach (Lee and Liu, 2003; Elkan and Noto, 2008) to augment the set of known summary-worthy sentences. In contrast, we adopt a conservative approach rather than predict too many sentences as being question-worthy: we pair up source sentences with their corresponding questions, and use just these sentence-question pairs to training the encoder-decoder model.

We use the `glove.840B.300d` pre-trained embeddings (Pennington et al., 2014) for initialization of the embedding layer for our sentence selection model and the full NQG model. `glove.6B.100d` embeddings are used for calculating sentence similarity feature of the baseline linear model (LREG). Tokens outside the vocabulary list are replaced by the `UNK` symbol. Hyperparameters for all models are tuned on the validation set and results are reported on the test set.

4.2 Sentence Selection Results

	conservative eval.		liberal eval.	
Gold Data System Output	w/ Q	w/o Q	w/ Q	w/o Q
w/ Q	matching	zero	matching	full
w/o Q	zero	-	zero	-

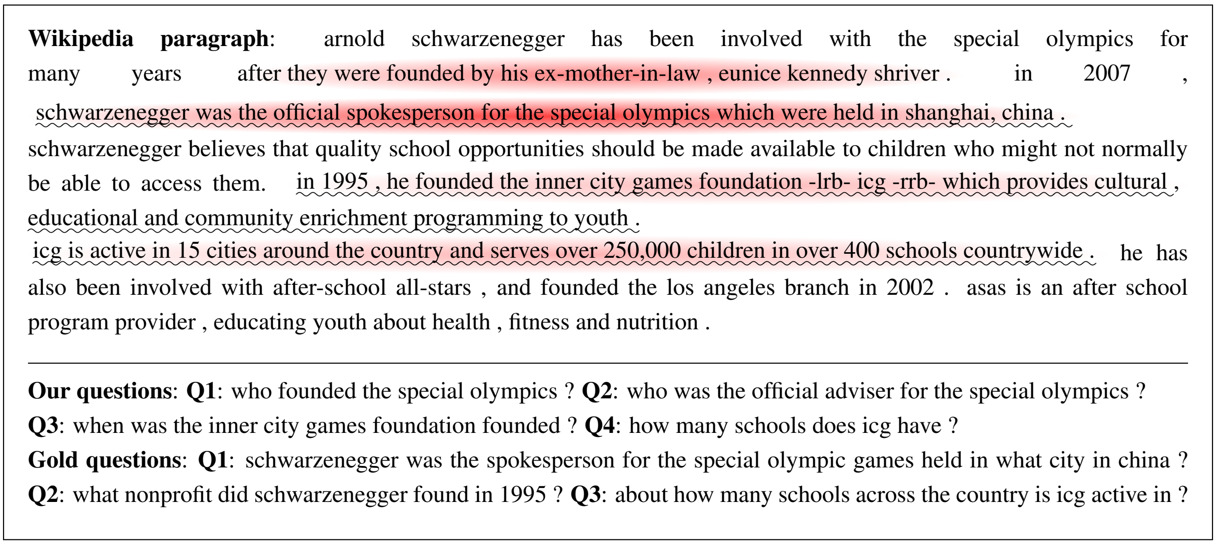
Table 3: For a source sentence in SQuAD, given the prediction from the sentence selection system and the corresponding NQG output, we provide conservative and liberal evaluations.

We compare to a number of baselines. The **Random** baseline assigns a random label to each sentence. The **Majority** baseline assumes that all sentences are question-worthy. The convolutional neural networks (**CNN**) sentence classification model (Kim, 2014) has similar structure to our CNN sentence encoder, but the classification is done only at the sentence-level rather than jointly at paragraph-level. **LREG$_{\text{w/ BOW}}$** is the logistic regression model with bag-of-words features. **LREG$_{\text{w/ para.-level}}$** is the feature-rich LREG model designed by Cheng and Lapata (2016); the features include: sentence length, position of sentence, number of named entities in the sentence, number of sentences in the paragraph, sentence-to-sentence cohesion, and sentence-to-paragraph relevance. Sentence-to-sentence cohesion is obtained by calculating the embedding space similarity between it and every other sentence in the paragraph (similar for sentence-to-paragraph relevance). In document summarization, graph-based extractive summarization models (e.g. TGRAPH Parveen et al. (2015) and URANK Wan (2010)) focus on global optimization and extract sentences contributing to topical coherent summaries. Because this does not really fit our task — a *summary-worthy* sentence might not necessarily contain enough information for generating a good question — we do not include these as comparisons.

Results are displayed in Table 1. Our models with sum or CNN as the sentence encoder significantly outperform the feature-rich LREG as well as the other baselines in terms of F-measure.

4.3 Evaluation of the full QG system

To evaluate the full systems for paragraph-level QG, we introduce in Table 3 the "conservative" and "liberal" evaluation strategies. Given an input source sentence, there will be in total four possibilities: if both the gold standard data and prediction include the sentence, then we use its n-gram matching score (by BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014)); if neither the gold data nor prediction include the sentence, then the sentence is discarded from the evaluation; if the gold data includes the sentence while the prediction does not, we assign a score of 0 for it; and if gold data does not include the sentence while prediction does, the generated question gets a 0 for conservative, while it gets full



Wikipedia paragraph: arnold schwarzenegger has been involved with the special olympics for many years after they were founded by his ex-mother-in-law , eunice kennedy shriver . in 2007 , schwarzenegger was the official spokesperson for the special olympics which were held in shanghai, china . schwarzenegger believes that quality school opportunities should be made available to children who might not normally be able to access them. in 1995 , he founded the inner city games foundation -lrb- icg -rrb- which provides cultural , educational and community enrichment programming to youth . icg is active in 15 cities around the country and serves over 250,000 children in over 400 schools countrywide . he has also been involved with after-school all-stars , and founded the los angeles branch in 2002 . asas is an after school program provider , educating youth about health , fitness and nutrition .

Our questions: **Q1**: who founded the special olympics ? **Q2**: who was the official adviser for the special olympics ? **Q3**: when was the inner city games foundation founded ? **Q4**: how many schools does icg have ?
Gold questions: **Q1**: schwarzenegger was the spokesperson for the special olympic games held in what city in china ? **Q2**: what nonprofit did schwarzenegger found in 1995 ? **Q3**: about how many schools across the country is icg active in ?

Figure 2: Sample output from our full NQG system, the four questions correspond to the four highlighted sentences in the paragraph in the same order. Darkness indicates sentence importance, the score for deciding the darkness is obtained from the softmax results. Wave-lined sentences bear label $y = 1$, and 0 otherwise. The three gold questions also correspond to the wave-lined sentences in the same order. Please refer to the appendix for sample output on more Wikipedia articles.

score for liberal evaluation. Table 2 shows that the QG system incorporating our best performing sentence extractor outperforms its LREG counterpart across metrics. Note that to calculate the score for the matching case, similar to our earlier work (Du et al., 2017), we adapt the image captioning evaluation scripts of Chen et al. (2015) since there can be several gold standard questions for a single input sentence.

In Figure 2, we provide questions generated by the full NQG system (Q1–4) and according to the gold standard (Q1–3) for the selected Wikipedia paragraph. The sentences they were drawn from are shown with wavy lines (gold standard) and via highlighting (our system). Darkness of the highlighting is proportional to the softmax score provided by the sentence extractor.

5 Conclusion

In this work we introduced the task of identifying important sentences — good sentences to ask a question about — in the reading comprehension setting. We proposed a hierarchical neural sentence labeling model and investigated encoding sentences with sum and convolution operations. The question generation system that uses our sentence selection model consistently outperforms previous approaches and achieves state-of-the-art paragraph-level question generation performance on the SQUAD data set.

In future work, we would like to investigate approaches to identify question-worth *concepts* rather than question-worthy sentences. It would also be interesting to see if the generated questions can be used to help improve question answering systems.

Acknowledgments

We thank the reviewers for helpful comments and Victoria Litvinova for proofreading.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations Workshop (ICLR)*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 481–490. http://www.aclweb.org/anthology/P11-1049.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 740–750. http://www.aclweb.org/anthology/D14-1082.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325* .

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, pages 484–494. http://www.aclweb.org/anthology/P16-1046.

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

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Baltimore, Maryland, USA, pages 376–380. http://www.aclweb.org/anthology/W14-3348.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*. pages 1818–1826.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. To appear.

Charles Elkan and Keith Noto. 2008. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '08, pages 213–220. https://doi.org/10.1145/1401890.1401920.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Los Angeles, California, pages 609–617. http://www.aclweb.org/anthology/N10-1086.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Doha, Qatar, pages 1746–1751. http://www.aclweb.org/anthology/D14-1181.

Wee Sun Lee and Bing Liu. 2003. Learning with positive and unlabeled examples using weighted logistic regression. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*. AAAI Press, ICML'03, pages 448–455. http://dl.acm.org/citation.cfm?id=3041838.3041895.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1412–1421. http://aclweb.org/anthology/D15-1166.

Prashanth Mannem, Rashmi Prasad, and Aravind Joshi. 2010. Question generation from paragraphs at upenn: Qgstec system description. In *Proceedings of QG2010: The Third Workshop on Question Generation*. pages 84–91.

Rada Mihalcea. 2005. Language independent extractive summarization. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*. Association for Computational Linguistics, Ann Arbor, Michigan, pages 49–52. https://doi.org/10.3115/1225753.1225766.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Daraksha Parveen, Hans-Martin Ramsl, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1949–1954. http://aclweb.org/anthology/D15-1226.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2383–2392. https://aclweb.org/anthology/D16-1264.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 251–257. http://dl.acm.org/citation.cfm?id=1873738.1873777.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1631–1642. http://www.aclweb.org/anthology/D13-1170.

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

Xiaojun Wan. 2010. Towards a unified approach to simultaneous single-document and multi-document summarizations. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Coling 2010 Organizing Committee, Beijing, China, pages 1137–1145. http://www.aclweb.org/anthology/C10-1128.

Yinfei Yang, Forrest Bao, and Ani Nenkova. 2017. Detecting (un)important content for single-document news summarization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, pages 707–712. http://www.aclweb.org/anthology/E17-2112.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. *arXiv preprint arXiv:1704.01792* .

Break it Down for Me: A Study in Automated Lyric Annotation

Lucas Sterckx[*], **Jason Naradowsky**[†], **Bill Byrne**[‡],
Thomas Demeester[*] and **Chris Develder**[*]
[*] IDLab, Ghent University - imec
firstname.lastname@ugent.be
[†] Language Technology Lab, DTAL, University of Cambridge
jrn39@cam.ac.uk
[‡] Department of Engineering, University of Cambridge
wjb31@cam.ac.uk

Abstract

Comprehending lyrics, as found in songs and poems, can pose a challenge to human and machine readers alike. This motivates the need for systems that can understand the ambiguity and jargon found in such creative texts, and provide commentary to aid readers in reaching the correct interpretation.

We introduce the task of automated lyric annotation (ALA). Like text simplification, a goal of ALA is to rephrase the original text in a more easily understandable manner. However, in ALA the system must often include *additional* information to clarify niche terminology and abstract concepts. To stimulate research on this task, we release a large collection of crowdsourced annotations for song lyrics. We analyze the performance of translation and retrieval models on this task, measuring performance with both automated and human evaluation. We find that each model captures a unique type of information important to the task.

1 Introduction

Song lyrics and poetry often make use of ambiguity, symbolism, irony, and other stylistic elements to evoke emotive responses. These characteristics sometimes make it challenging to interpret obscure lyrics, especially for readers or listeners who are unfamiliar with the genre. To address this problem, several online lyric databases have been created where users can explain, contextualize, or discuss lyrics. Examples include MetroLyrics[1] and Genius.com[2]. We refer to such commentary as a lyric annotation (Figure 1).

How does it feel?
To be without a home
Like a complete unknown,

Like a rolling stone

↓

The proverb "A rolling stone gathers no moss" refers to people who are always on the move, never putting down roots or accumulating responsibilities and cares.

Figure 1: A lyric annotation for "Like A Rolling Stone" by Bob Dylan.

In this work we introduce the task of *automated lyric annotation* (ALA). Compared to many traditional NLP systems, which are trained on newswire or similar text, an automated system capable of explaining abstract language, or finding alternative text expressions for slang (and other unknown terms) would exhibit a deeper understanding of the nuances of language. As a result, research in this area may open the door to a variety of interesting use cases. In addition to providing lyric annotations, such systems can lead to improved NLP analysis of informal text (blogs, social media, novels and other literary works of fiction), better handling of genres with heavy use of jargon (scientific texts, product manuals), and increased robustness to textual variety in more traditional NLP tasks and genres.

Our contributions are as follows:

1. To aid in the study of ALA we present a corpus of 803,720 crowdsourced lyric annotation pairs suitable for training models for this task.[3]
2. We present baseline systems using statistical machine translation (SMT), neural trans-

[1] http://www.metrolyrics.com

[2] http://genius.com

[3] To obtain the data collection please contact the first author of this paper.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2074–2080
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

# Lyric Annotation pairs	803,720
⊘ Tokens per Lyric	15
⊘ Tokens per Annotation	43
$\|V_{\text{lyrics}}\|$	124,022
$\|V_{\text{annot}}\|$	260,427

Table 1: Properties of gathered dataset (V_{lyrics} and V_{annot} denote the vocabulary for lyrics and annotations, ⊘ denotes the average amount).

lation (Seq2Seq), and information retrieval.

3. We establish an evaluation procedure which adopts measures from machine translation, paraphrase generation, and text simplification. Evaluation is conducted using both human and automated means, which we perform and report across all baselines.

2 The Genius ALA Dataset

We collect a dataset of crowdsourced annotations, generated by users of the *Genius* online lyric database. For a given song, users can navigate to a particular stanza or line, view existing annotations for the target lyric, or provide their own annotation. Discussion between users acts to improve annotation quality, as it does with other collaborative online databases like Wikipedia. This process is gamified: users earn *IQ* points for producing high quality annotations.

We collect 736,423 lyrics having a total 1,404,107 lyric annotation pairs from all subsections (rap, poetry, news, etc.) of Genius. We limit the initial release of the annotation data to be English-only, and filter out non-English annotations using a pre-trained language identifier. We also remove annotations which are solely links to external resources, and do not provide useful textual annotations. This reduces the dataset to 803,720 lyric annotation pairs. We list several properties of the collected dataset in Table 1.

2.1 Context Independent Annotation

Mining annotations from a collaborative human-curated website presents additional challenges worth noting. For instance, while we are able to generate large quantities of parallel text from Genius, users operate without a single, predefined and shared *global* goal other than to maximize their own IQ points. As such, there is no motivation to provide annotations for a song in its entirety, or independent of previous annotations.

For this reason we distinguish between two types of annotations: *context independent* (CI) annotations are independent of their surrounding context and can be interpreted without it, e.g., explain specific metaphors or imagery or provide narrative while normalizing slang language. Contrastively, *context sensitive* (CS) annotations provide broader context beyond the song lyric excerpt, e.g., background information on the artist.

To estimate contribution from both types to the dataset, we sample 2,000 lyric annotation pairs and label them as either CI or CS. Based on this sample, an estimated 34.8% of all annotations is independent of context. Table 2 shows examples of both types.

While the goal of ALA is to generate annotations of all types, it is evident from our analysis that CS annotations can not be generated by models trained solely on parallel text. That is, these annotations cannot be generated without background knowledge or added context. Therefore, in this preliminary work we focus on predicting CI lyric annotations.

3 Baselines

We experiment with three baseline models used for text simplification and paraphrase generation.

- **Statistical Machine Translation (SMT):** One approach is to treat the task as one of translation, and to use established statistical machine translation (SMT) methods (Quirk et al., 2004) to produce them. We train a standard phrase-based SMT model to translate lyrics to annotations, using GIZA++ (Josef Och and Ney, 2003) for word alignment and Moses (Koehn et al., 2007) for phrasal alignment, training, and decoding.

- **Seq2Seq:** Sequence-to-sequence models (Sutskever et al., 2014) offer an alternative to SMT systems, and have been applied successfully to a variety of tasks including machine translation. In Seq2Seq, a recurrent neural network (RNN) encodes the source sequence to a single vector representation. A separate decoder RNN generates the translation conditioned on this representation of the source sequence's semantics. We utilize Seq2Seq with attention (Bahdanau et al., 2014), which allows the model to

Type	% of annotations	Examples	
CI (Context independent)	34.8%	[L]	Gotta patch a lil kid tryna get at this cabbage
		[A]	He's trying to ignore the people trying to get at his money.
		[L]	You know it's beef when a smart brother gets stupid
		[A]	You know an argument is serious when an otherwise rational man loses rational.
CS (Context sensitive)	65.2%	[L]	Cause we ain't break up, more like broke down
		[A]	The song details Joe's break up with former girlfriend Esther.
		[L]	If I quit this season, I still be the greatest, funk
		[A]	Kendrick has dropped two classic albums and pushed the artistic envelope further.

Table 2: Examples of context independent and dependent pairs of lyrics [L] and annotations [A].

additionally condition on tokens from the input sequence during decoding.

- **Retrieval:** In practice, similar lyrics may reappear in different contexts with exchangeable annotations. We treat the training corpus as a database of lyrics' excerpts with corresponding annotations, and at test time select the annotation assigned to the most similar lyric. This baseline is referred to as the *retrieval* model. We use standard TF-IDF weighted cosine distance as similarity measure between lyrics' excerpts.

4 Evaluation

4.1 Data

We evaluate automatic annotators on a selection of 354 CI annotations and partition the rest of the annotations into 2,000 instances for development and the full remainder for training. It is important to note that the annotations used for training and development include CI as well as CS annotations.

Annotations often include multiple sentences or even paragraphs for a single lyrics excerpt (which does not include end marks), while machine translation models need aligned corpora at sentence level to perform well (Xu et al., 2016). We therefore transform training data by including each sentence from the annotation as a single training instance with the same lyric, resulting in a total of 1,813,350 sentence pairs.

We use this collection of sentence pairs (denoted as *sent.* in results) to train the SMT model. Seq2Seq models are trained using sentence pairs as well as full annotations. Interestingly, techniques encouraging alignment by matching length and thresholding cosine distance between lyric and annotation did not improve performance during development.

4.2 Measures

For automated evaluation, we use measures commonly used to evaluate translation systems (BLEU, METEOR), paraphrase generation (iBLEU) and text simplification (SARI).

BLEU (Papineni et al., 2002) uses a modified form of precision to compare generated annotations against references from Genius. METEOR (Denkowski and Lavie, 2011) is based on the harmonic mean of precision and recall and, along with exact word matching, includes stemming and synonymy matching. iBLEU (Sun and Zhou, 2012) is an extension of the BLEU metric to measure diversity as well as adequacy of the annotation, iBLEU $= 0.9 \times$ BLEU(Annotation, Reference) $- 0.1 \times$ BLEU(Annotation, Lyric). SARI (Xu et al., 2016) measures precision and recall of words that are added, kept, or deleted separately and averages their arithmetic means.

We also measure quality by crowdsourcing ratings via the online platform CrowdFlower.[4] We present collaborators with a song lyric excerpt annotated with output from the annotation generators as well as a reference annotation from Genius. Collaborators assign a 5-point rating for *Fluency* which rates the quality of the generated language, and *Information* which measures the added clarification by the annotation, a key aspect of this task. For each lyric annotation pair, we gather ratings from three different collaborators and take the average.

4.3 Hyperparameters and Optimization

Here we describe implementation and some of the optimizations used when training the models.

[4] https://www.crowdflower.com/

Lyric:	`and when I slept, I dream g's, son I need some`
Human:	he dreams about large amounts of money because he is in need of it .
SMT:	and when he slept, and he's dream, g's son, he needs some
Seq2Seq:	when he sleeps , he dreams of getting money.
Retrieval:	g's refer to the ghetto gangsters in london that the kid has become involved with
Lyric:	`Mossberg pump I'm ridin' shotgun literally`
Human:	riding shotgun means to ride up front in the passenger seat
SMT:	Mossberg pump - action shotgun, he's saying he's ridin' shotgun literally.
Seq2Seq:	Mossberg is a firearms manufacturer. He is riding shotgun with his shotgun.
Retrieval:	he fully respects the people mentioned earlier and is riding for them.
Lyric:	`and my eyes red cause of all that haze`
Human:	his eyes turn red due to lack of hydration from the weed one smokes (in this case , smoking purple haze)
SMT:	and his eyes are red because of all that haze.
Seq2Seq:	his eyes are red because of all the weed he smokes.
Retrieval:	his eyes are red as a sign of the fatigue of travel and , possibly , from drugs and alcohol .

Table 3: Lyrics excerpts with annotations from Genius ('Human') and automated annotators.

	Properties		Automated Evaluation				Human Evaluation	
	Length Ratio	Profanity/Tok.	BLEU	iBLEU	METEOR	SARI	Fluency	Information
Human	1.19	0.0027	-	-	-	-	3.93	3.53
SMT (Sent.)	1.23	0.0068	6.22	1.44	12.20	38.42	3.82	3.31
Seq2Seq (Sent.)	1.05	0.0023	5.33	3.64	9.28	36.52	3.76	3.25
Seq2Seq	1.32	0.0022	5.15	3.46	10.56	36.86	3.83	3.34
Retrieval	1.18	0.0038	2.82	2.27	5.10	32.76	3.93	2.98

Table 4: Quantitative evaluation of different automated annotators.

For Seq2Seq models, we use OpenNMT (Klein et al., 2017) and optimize for perplexity on the development set. Vocabulary for both lyrics and annotations is reduced to the 50,000 most frequent tokens and are embedded in a 500-dimensional space.

We use two layers of stacked bi-directional LSTMs with hidden states of 1024 dimensions. We regularize using dropout (keep probability of 0.7) and train using stochastic gradient descent with batches of 64 samples for 13 epochs.

The decoder of the SMT model is tuned for optimal BLEU scores on the development set using minimum error rate training (Bertoldi et al., 2009).

5 Results

To measure agreement between collaborators, we compute the kappa statistic (Fleiss, 1971). Kappa statistics for fluency and information are 0.05 and 0.07 respectively, which indicates low agreement. The task of evaluating lyric annotations was difficult for CrowdFlower collaborators as was apparent from their evaluation of the task. For evaluation in future work, we recommend recruitment of expert collaborators familiar with the Genius platform and song lyrics.

Table 3 shows examples of lyrics with annotations from Genius and those generated by baseline models.

A notable observation is that translation models learn to take the role of narrator, as is common in CI annotations, and recognize slang language while simplifying it to more standard English.

Automatic and human evaluation scores are shown in Table 4. Next to evaluation metrics, we show two properties of automatically generated annotations; the average annotation length relative to the lyric and the occurrence of profanity per token in annotations, using a list of 343 swear words.

The SMT model scores high on BLEU, METEOR and SARI but shows a large drop in performance for iBLEU, which penalizes lexical similarity between lyrics and generated annotations as apparent from the amount profanity remaining in the generated annotations.

Standard SMT rephrases the song lyric from a third person perspective but is conservative in lexical substitutions and keeps close to the grammar of the lyric. A more appropriate objective function for tuning the decoder which promotes lexical dissimilarity as done for paraphrase generation, would be beneficial for this approach.

Seq2Seq models generate annotations more dissimilar to the song lyric and obtain higher iBLEU

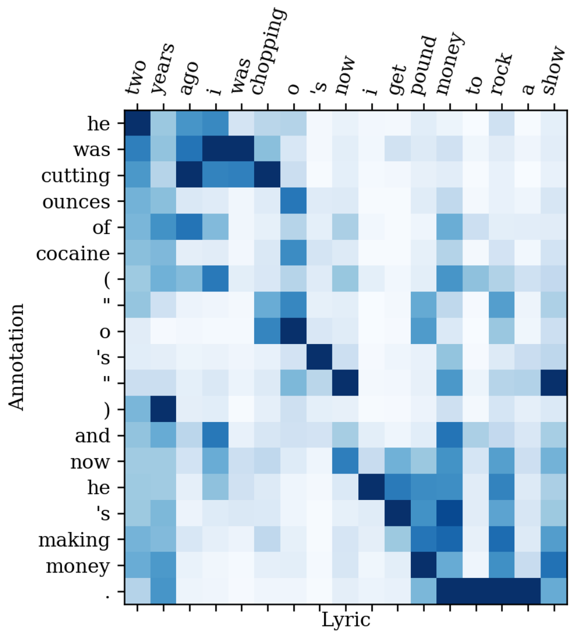


Figure 2: Attention visualization of Seq2Seq models for ALA.

and Information scores. To visualize some of the alignments learned by the translation models, Fig. 2 shows word-by-word attention scores for a translation by the Seq2Seq model.

While the retrieval model obtains quality annotations when test lyrics are highly similar to lyrics from the training set, retrieved annotations are often unrelated to the test lyric or specific to the song lyric it is retrieved from.

Out of the unsupervised metrics, METEOR obtained the highest Pearson correlation (Pearson, 1895) with human ratings for Information with a coefficient of 0.15.

6 Related Work

Work on modeling of social annotations has mainly focused on the use of topic models (Iwata et al., 2009; Das et al., 2014) in which annotations are assumed to originate from topics. They can be used as a preprocessing step in machine learning tasks such as text classification and image recognition but do not generate language as required in our ALA task.

Text simplification and paraphrase generation have been widely studied. Recent work has highlighted the need for large text collections (Xu et al., 2015) as well as more appropriate evaluation measures (Xu et al., 2016; Galley et al., 2015). They indicated that especially informal language, with its high degree of lexical variation, e.g., as used in social media or lyrics, poses serious challenges (Xu et al., 2013).

Text generation for artistic purposes, such as poetry and lyrics, has been explored most commonly using templates and constraints (Barbieri et al., 2012). In regard to rap lyrics, Wu et al. (2013) present a system for rap lyric generation that produces a single line of lyrics that is meant to be a response to a single line of input. Most recent work is that of Zhang et al. (2014) and Potash et al. (2015), who show the effectiveness of RNNs for the generation of poetry and lyrics.

The task of annotating song lyrics is also related to metaphor processing. As annotators often explain metaphors used in song lyrics, the Genius dataset can serve as a resource to study computational modeling of metaphors (Shutova and Teufel, 2010).

7 Conclusion and Future Work

We presented and released the Genius dataset to study the task of Automated Lyric Annotation. As a first investigation, we studied automatic generation of context independent annotations as machine translation and information retrieval. Our baseline system tests indicate that our corpus is suitable to train machine translation systems.

Standard SMT models are capable of rephrasing and simplifying song lyrics but tend to keep close to the structure of the song lyric. Seq2Seq models demonstrated potential to generate more fluent and informative text, dissimilar to the lyric.

A large fraction of the annotations is heavily based on context and background knowledge (CS), one of their most appealing aspects. As future work we suggest injection of structured and unstructured external knowledge (Ahn et al., 2016) and explicit modeling of references (Yang et al., 2016).

Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was supported by the Research Foundation - Flanders (FWO) and the U.K. Engineering and Physical Sciences Research Council (EPSRC grant EP/L027623/1).

References

S. Ahn, H. Choi, T. Pärnamaa, and Y. Bengio. 2016. A Neural Knowledge Language Model. *ArXiv e-prints* https://arxiv.org/abs/1608.00318.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Gabriele Barbieri, François Pachet, Pierre Roy, and Mirko Degli Esposti. 2012. Markov constraints for generating lyrics with style. In *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*. pages 115–120. https://doi.org/10.3233/978-1-61499-098-7-115.

Nicola Bertoldi, Barry Haddow, and Jean-Baptiste Fouet. 2009. Improved minimum error rate training in moses. *Prague Bull. Math. Linguistics* 91:7–16. http://ufal.mff.cuni.cz/pbml/91/art-bertoldi.pdf.

Mrinal Kanti Das, Trapit Bansal, and Chiranjib Bhattacharyya. 2014. Going beyond corr-lda for detecting specific comments on news & blogs. In *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*. pages 483–492. https://doi.org/10.1145/2556195.2556231.

Michael Denkowski and Alon Lavie. 2011. Proceedings of the sixth workshop on statistical machine translation pages 85–91. http://aclweb.org/anthology/W11-2107.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76(5):378.

Michel Galley, Chris Brockett, Alessandro Sordoni, Yangfeng Ji, Michael Auli, Chris Quirk, Margaret Mitchell, Jianfeng Gao, and Bill Dolan. 2015. deltableu: A discriminative metric for generation tasks with intrinsically diverse targets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 445–450. https://doi.org/10.3115/v1/P15-2073.

Tomoharu Iwata, Takeshi Yamada, and Naonori Ueda. 2009. Modeling social annotation data with content relevance using a topic model. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*. pages 835–843. http://papers.nips.cc/paper/3773-modeling-social-annotation-data-with-content-relevance-using-a-topic-model.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics, Volume 29, Number 1, March 2003* http://aclweb.org/anthology/J03-1002.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A.M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints* https://arxiv.org/abs/1701.02810.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Association for Computational Linguistics, pages 177–180. http://aclweb.org/anthology/P07-2045.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. http://aclweb.org/anthology/P02-1040.

Karl Pearson. 1895. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London* 58:240–242.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2015. Ghostwriter: Using an lstm for automatic rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1919–1924. https://doi.org/10.18653/v1/D15-1221.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Proceedings of the 2004 conference on empirical methods in natural language processing. http://aclweb.org/anthology/W04-3219.

Ekaterina Shutova and Simone Teufel. 2010. Metaphor corpus annotated for source - target domain mappings. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Languages Resources Association (ELRA). http://aclweb.org/anthology/L10-1419.

Hong Sun and Ming Zhou. 2012. Joint learning of a dual smt system for paraphrase generation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pages 38–42. http://aclweb.org/anthology/P12-2008.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. pages 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.

Dekai Wu, Karteek Addanki, Markus Saers, and Meriem Beloucif. 2013. Learning to freestyle: Hip hop challenge-response induction via transduction rule segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 102–112. http://aclweb.org/anthology/D13-1011.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association of Computational Linguistics* 3:283–297. http://aclweb.org/anthology/Q15-1021.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association of Computational Linguistics* 4:401–415. http://aclweb.org/anthology/Q16-1029.

Wei Xu, Alan Ritter, and Ralph Grishman. 2013. Proceedings of the sixth workshop on building and using comparable corpora. Association for Computational Linguistics, pages 121–128. http://aclweb.org/anthology/W13-2515.

Z. Yang, P. Blunsom, C. Dyer, and W. Ling. 2016. Reference-Aware Language Models. *ArXiv e-prints* https://arxiv.org/abs/1611.01628.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 670–680. https://doi.org/10.3115/v1/D14-1074.

Cascaded Attention based Unsupervised Information Distillation for Compressive Summarization[*]

Piji Li[†] **Wai Lam**[†] **Lidong Bing**[‡] **Weiwei Guo**[§] **Hang Li**[♮]
[†]Department of Systems Engineering and Engineering Management,
The Chinese University of Hong Kong
[‡]AI Lab, Tencent Inc., Shenzhen, China
[§]LinkedIn, Mountain View, CA, USA
[♮]Noah's Ark Lab, Huawei Technologies, Hong Kong
[†]{pjli, wlam}@se.cuhk.edu.hk, [‡]lyndonbing@tencent.com
[§]wguo@linkedin.com, [♮]hangli.hl@huawei.com

Abstract

When people recall and digest what they have read for writing summaries, the important content is more likely to attract their attention. Inspired by this observation, we propose a cascaded attention based unsupervised model to estimate the salience information from the text for compressive multi-document summarization. The attention weights are learned automatically by an unsupervised data reconstruction framework which can capture the sentence salience. By adding sparsity constraints on the number of output vectors, we can generate condensed information which can be treated as word salience. Fine-grained and coarse-grained sentence compression strategies are incorporated to produce compressive summaries. Experiments on some benchmark data sets show that our framework achieves better results than the state-of-the-art methods.

1 Introduction

The goal of Multi-Document Summarization (MDS) is to automatically produce a succinct summary, preserving the most important information of a set of documents describing a topic[1] (Luhn, 1958; Edmundson, 1969; Goldstein et al., 2000; Erkan and Radev, 2004b; Wan et al., 2007; Nenkova and McKeown, 2012). Considering the procedure of summary writing by humans, when people read, they will **remember** and **forget** part of the content. Information which is more important may make a deep impression easily. When people recall and digest what they have read to write summaries, the important information usually attracts more **attention** (*the behavioral and cognitive process of selectively concentrating on a discrete aspect of information, whether deemed subjective or objective, while ignoring other perceivable information*[2]) since it may repeatedly appears in some documents, or be positioned in the beginning paragraphs.

In the context of multi-document summarization, to generate a summary sentence for a key aspect of the topic, we need to find its relevant parts in the original documents, which may attract more attention. The semantic parts with high attention weights plausibly represent and reconstruct the topic's main idea. To this end, we propose a cascaded neural attention model to distill salient information from the input documents in an unsupervised data reconstruction manner, which includes two components: reader and recaller. The reader is a gated recurrent neural network (LSTM or GRU) based sentence sequence encoder which can map all the sentences of the topic into a global representation, with the mechanism of remembering and forgetting. The recaller decodes the global representation into significantly fewer diversified vectors for distillation and concentration. A cascaded attention mechanism is designed by incorporating attentions on both the hidden layer (dense distributed representation of a sentence) and the output layer (sparse bag-of-words representation of summary information). It is worth noting that the output vectors of the recaller can be viewed as word salience, and the attention matrix can be used as sentence salience. Both of them are automatically learned by data reconstruction in an **un-**

[*]The work described in this paper is supported by grants from the Research and Development Grant of Huawei Technologies Co. Ltd (YB2015100076/TH1510257) and the Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14203414).

[1]A topic represents a real event, e.g., "AlphaGo versus Lee Sedol".

[2]https://en.wikipedia.org/wiki/Attention (Apr., 2017)

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2081–2090
Copenhagen, Denmark, September 7–11, 2017.

supervised manner. Thereafter, the word salience is fed into a coarse-grained sentence compression component. Finally, the attention weights are integrated into a phrase-based optimization framework for compressive summary generation.

In fact, the notion of "attention" has gained popularity recently in neural network modeling, which has improved the performance of many tasks such as machine translation (Bahdanau et al., 2015; Luong et al., 2015). However, very few previous works employ attention mechanism to tackle MDS. Rush et al. (2015) and Nallapati et al. (2016) employed attention-based sequence-to-sequence (seq2seq) framework only for sentence summarization. Gu et al. (2016), Cheng and Lapata (2016), and Nallapati et al. (2016) also utilized seq2seq based framework with attention modeling for short text or single document summarization. Different from their works, our framework aims at conducting multi-document summarization in an unsupervised manner.

Our contributions are as follows: (1) We propose a cascaded attention model that captures salient information in different semantic representations. (2) The attention weights are learned automatically by an unsupervised data reconstruction framework which can capture the sentence salience. By adding sparsity constraints on the number of output vectors of the recaller, we can generate condensed vectors which can be treated as word salience; (3) We thoroughly investigate the performance of combining different attention architectures and cascaded structures. Experimental results on some benchmark data sets show that our framework achieves better performance than the state-of-the-art models.

2 Framework Description

2.1 Overview

Our framework has two phases, namely, information distillation for finding salient words/sentences, and compressive summary generation. For the first phase, our cascaded neural attention model consists of two components: reader and recaller as shown in Figure 1. The reader component reads in all the sentences in the document set corresponding to the topic/event. The information distillation happens in the recaller component where only the most important information is preserved. Precisely, the recaller outputs fewer vectors s than that of the input sentences x for the reader.

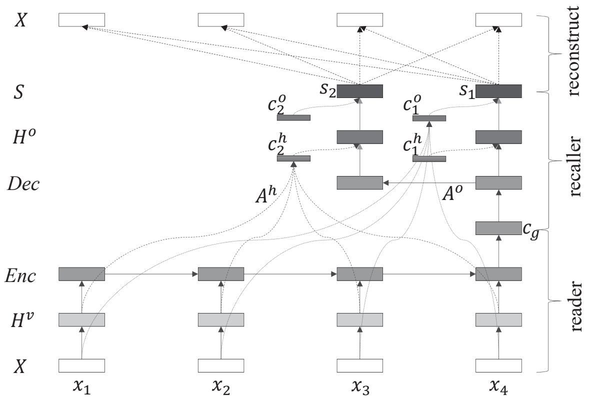


Figure 1: Our cascaded attention based unsupervised information distillation framework. X is the original input sentence sequence of a topic. H^i is the hidden vectors of sentences. "Enc" and "Dec" represent the RNN-based encoding and decoding layer respectively. c_g is the global representation for the whole topic. A^h and A^o are the distilled attention matrices for the hidden layer and the output layer respectively, representing the salience of sentences. H^o is the output hidden layer. s_1 and s_2 are the distilled condensed vectors representing the salience of words. Note that they are neither origin inputs nor golden summaries.

After the learning of the neural attention model finishes, the obtained salience information will be used in the second phase for compressive summary generation. This phase consists of two components: (i) the coarse-grained sentence compression component which can filter the trivial information based on the output vectors S from the neural attention model; (ii) the unified phrase-based optimization method for summary generation in which the attention matrix A^o is used to conduct fine-grained compression and summary construction.

2.2 Attention Modeling for Distillation

2.2.1 Reader

In the reader stage, for each topic, we extract all the sentences $X = \{x_1, x_2, \ldots, x_m\}$ from the set of input documents corresponding to a topic and generate a sentence sequence with length m. The sentence order is the same as the original order of the documents. Then the reader reads the whole sequence sentence by sentence. We employ the bag-of-words (BOW) representation as the initial semantic representation for sentences. Assume that the dictionary size is k, then $x_i \in \mathbb{R}^k$.

Sparsity is one common problem for the BOW representation, especially when each vector is generated from a single sentence. Moreover, downstream algorithms might suffer from the curse of dimensionality. To solve these problems, we add a hidden layer H^v (v for input layer) which is a densely distributed representation above the input layer as shown in Figure 1. Such distributed representation can provide better generalization than BOW representation in many different tasks (Le and Mikolov, 2014; Mikolov et al., 2013). Specifically, the input hidden layer will project the input sentence vector x_j to a new space $\mathbb{R}^h$ according to Equation 1. Then we obtain a new sentence sequence $H^v = [h_1^v, h_2^v, \ldots, h_m^v]$.

$$h_j^v = \tanh(W_{xh}^v x_j + b_h^v) \tag{1}$$

where W_{xh}^v and b_h^v are the weight and bias respectively. The superscript v means that the variables are from the input layer.

While reading the sentence sequence, the reader should have the ability of remembering and forgetting. Therefore, we employ the RNN models with various gates (input gate, forget gate, etc.) to imitate the remembering and forgetting mechanism. Then the RNN based neural encoder (the third layer in Figure 1) will map the whole embedding sequence to a single vector c_g which can be regarded as a global representation for the whole topic. Let t be the index of the sequence state for the sentence x_t, the hidden unit h_t^e (e for encoder RNN) of the RNN encoder can be computed as:

$$h_t^e = f(h_{t-1}^e, h_t^v) \tag{2}$$

where the RNN $f(\cdot)$ computes the current hidden state given the previous hidden state h_{t-1}^e and the sentence embedding h_t^v. The encoder generates hidden states $\{h_t^e\}$ over all time steps. The last state $\{h_m^e\}$ is extracted as the global representation c_g for the whole topic. The structure for $f(\cdot)$ can be either an LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014).

2.2.2 Recaller

The recaller stage is a reverse of the reader stage, but it outputs less number of vectors in S as shown in Figure 1. Given the global representation c_g, the past hidden state h_{t-1}^d (d for decoder RNN) from the decoder layer, an RNN based decoder generates several hidden states according to:

$$h_t^d = f(h_{t-1}^d, c_g) \tag{3}$$

We use c_g to initialize the first decoder hidden state. The decoder will generate several hidden states $\{h_t^d\}$ over pre-defined time steps. Then, similar to the reader stage, we add an output hidden layer after the decoder layer:

$$h_t^o = \tanh(W_{hh}^o h_t^d + b_h^o) \tag{4}$$

where W_{hh}^o and b_h^o are the weight and bias respectively for the projection from h_t^d to h_t^o. Finally, the output layer maps these hidden vectors to the condensed vectors $S = [s_1, s_2, \ldots, s_n]$, Each output vector s_t has the same dimension k as the input BOW vectors and is obtained as follows:

$$s_t = \sigma(W_{hs} h_t^o + b_s) \tag{5}$$

For the purpose of distillation and concentration, we restrict n to be very small.

2.2.3 Cascaded Attention Modeling

Salience estimation for words and sentences is a crucial component in MDS, especially in the unsupervised summarization setting. We propose a cascaded attention model for information distillation to tackle the salience estimation task for MDS. We add attention mechanism not only in the hidden layer, but also in the output layer. By this cascaded attention model, we can capture the salience of sentences from two different and complementary vector spaces. One is the embedding space that provides better generalization, and the other one is the BOW vector space that captures more nuanced and subtle difference.

For each output hidden state h_t^o, we align it with each input hidden state h_i^v by an attention vector $a_{t,i}^h \in \mathbb{R}^m$ (recall that m is the number of input sentences). $a_{t,i}^h$ is derived by comparing h_t^o with each input sentence hidden state h_i^v:

$$a_{t,i}^h = \frac{\exp(score(h_t^o, h_i^v))}{\sum_{i'} \exp(score(h_t^o, h_{i'}^v))} \tag{6}$$

where $score(\cdot)$ is a content-based function to capture the relation between two vectors. Several different formulations can be used as the function $score(\cdot)$ which will be elaborated later.

Based on the alignment vectors $\{a_{t,i}^h\}$, we can create a context vector c_t^h by linearly blending the sentence hidden states $\{h_{i'}^v\}$:

$$c_t^h = \sum\nolimits_{i'} a_{t,i'}^h h_{i'}^v \tag{7}$$

Then the output hidden state can be updated based on the context vector. Let $\tilde{h}_t^o = h_t^o$, then update the original state according to the following operation:

$$h_t^o = \tanh(W_{ch}^a c_t^h + W_{hh}^a \tilde{h}_t^o) \tag{8}$$

The alignment vector $a_{t,i}^h$ captures which sentence should be attended more in the hidden space when generating the condensed representation for the whole topic.

Besides the attention mechanism on the hidden layer, we also directly add attention on the output BOW layer which can capture more nuanced and subtle difference information from the BOW vector space. The hidden attention vector $a_{t,i}^h$ is integrated with the output attention by a weight $\lambda_a \in [0, 1]$:

$$\bar{a}_{t,i}^o = \frac{\exp(score(s_t, x_i))}{\sum_{i'} \exp(score(s_t, x_{i'}))} \tag{9}$$

$$a_{t,i}^o = \lambda_a \bar{a}_{t,i}^o + (1 - \lambda_a) a_{t,i}^h \tag{10}$$

The output context vector is computed as:

$$c_t^o = \sum\nolimits_{i'} a_{t,i'}^o x_{i'} \tag{11}$$

To update the output vector s_t in Equation 5, we develop a different method from that of the hidden attentions. Specifically we use a weighted combination of the context vectors and the original outputs with $\lambda_c \in [0, 1]$. Let $\tilde{s}_t = s_t$, then the updated s_t is:

$$s_t = \lambda_c c_t^o + (1 - \lambda_c)\tilde{s}_t \tag{12}$$

The parameters λ_a and λ_c can also be learned during training.

There are several different alternatives for the function $score(\cdot)$:

$$score(h_t, h_s) = \begin{cases} {h_t}^T h_s & \textit{dot} \\ {h_t}^T W h_s & \textit{tensor} \\ v^T \tanh(W[h_t; h_s]) & \textit{concat} \end{cases} \tag{13}$$

Considering their behaviors as studied in (Luong et al., 2015), we adopt "*concat*" for the hidden attention layer, and "*dot*" for the output attention layer.

2.2.4 Unsupervised Learning

By minimizing the loss owing to using the condensed output vectors to reconstruct the original input sentence vectors, we are able to learn the solutions for all the parameters as follows.

$$\min_{\Theta} \frac{1}{2m} \sum_{i=1}^{m} \|x_i - \sum_{j=1}^{n} s_j a_{j,i}^o\|_2^2 + \lambda_s \|S\|_1 \tag{14}$$

where Θ denotes all the parameters in our model. In order to penalize the unimportant terms in the output vectors, we put a sparsity constraint on the rows of S using l_1-regularization, with the weight λ_s as a scaling constant for determining its relative importance.

Let $\bar{s}$ be the magnitude vector computed from the columns in S ($S \in \mathbb{R}^{n \times k}$). Once the training is finished, each dimension of the vector $\bar{s}$ can be regarded as the **word salience** score. According to Equation 14, $s_i \in S$ is used to reconstruct the original sentence space X, and $n \ll m$ (the number of sentences in X is much more than the number of vectors in S) Therefore a large value in $\bar{s}$ means that the corresponding word contains important information about this topic and it can serve as the word salience.

Moreover, the output layer attention matrix A^o can be regarded as containing the **sentence salience** information. Note that each output vector s_i is generated based on the cascaded attention mechanism. Assume that $a_i^o = A_{i,:}^o \in \mathbb{R}^m$ is the attention weight vector for s_i. According to Equation 9, a large value in a_i^o conveys a meaning that the corresponding sentence should contribute more when generating s_i. We also use the magnitude of the columns in A^o to represent the salience of sentences.

2.3 Compressive Summary Generation Phase

2.3.1 Coarse-grained Sentence Compression

Using the information distillation result from the cascaded neural attention model, we conduct coarse-grained compression for each individual sentence. Such strategy has been adopted in some multi-document summarization methods (Li et al., 2013; Wang et al., 2013; Yao et al., 2015). Our coarse-grained sentence compression jointly considers word salience obtained from the neural attention model and linguistically-motivated rules. The linguistically-motivated rules are designed based on the observed obvious evidence for uncritical information from the word level to the clause level, which include news headers such as "BEIJING, Nov. 24 (Xinhua) –", intra-sentential attribution such as ", police said Thursday", ", he said", etc. The information filtered by the rules will be processed according to the word salience score. Information with smaller salience score ($< \epsilon$) will be removed.

2.3.2 Phrase-based Optimization for Summary Construction

After coarse-grained compression on each single sentence as described above, we design a unified optimization method for summary generation. We refine the phrase-based summary construction model in (Bing et al., 2015) by adjusting the goal as compressive summarization. We consider the salience information obtained by our neural attention model and the compressed sentences in the coarse-grained compression component.

Based on the parsed constituency tree for each input sentence as described in Section 2.3.1, we extract the noun-phrases (NPs) and verb-phrases (VPs). The salience S_i of a phrase P_i is defined as:

$$S_i = \{\sum_{t \in P_i} tf(t) / \sum_{t \in Topic} tf(t)\} \times a_i \qquad (15)$$

where a_i is the salience of the sentence containing P_i. $tf(t)$ is the frequency of the concept t (unigram/bigram) in the whole topic. Thus, S_i inherits the salience of its sentence, and also considers the importance of its concepts.

The overall objective function of our optimization formulation for selecting salient NPs and VPs is formulated as an integer linear programming (ILP) problem:

$$\max\{\sum\nolimits_i \alpha_i S_i - \sum\nolimits_{i<j} \alpha_{ij}(S_i + S_j)R_{ij}\} \quad (16)$$

where α_i is the selection indicator for the phrase P_i, S_i is the salience scores of P_i, α_{ij} and R_{ij} is the co-occurrence indicator and the similarity of a pair of phrases (P_i, P_j) respectively. The similarity is calculated by the Jaccard Index based method. Specifically, this objective maximizes the salience score of the selected phrases as indicated by the first term, and penalizes the selection of similar phrase pairs.

In order to obtain coherent summaries with good readability, we add some constraints into the ILP framework such as sentence generation constraint: Let β_k denote the selection indicator of the sentence x_k. If any phrase from x_k is selected, $\beta_k = 1$. Otherwise, $\beta_k = 0$. For generating a compressed summary sentence, it is required that if $\beta_k = 1$, at least one NP and at lease one VP of the sentence should be selected. It is expressed as:

$$\forall P_i \in x_k, \alpha_i \leq \beta_k \wedge \sum\nolimits_i \alpha_i \geq \beta_k, \qquad (17)$$

Other constraints include sentence number, summary length, phrase co-occurrence, etc. For details, please refer to McDonald (2007), Woodsend and Lapata (2012), and Bing et al. (2015).

The objective function and constraints are linear. Therefore the optimization can be solved by existing ILP solvers such as the simplex algorithm (Dantzig and Thapa, 2006). In the implementation, we use a package called lp_solve[3].

In the post-processing, the phrases and sentences in a summary are ordered according to their natural order if they come from the same document. Otherwise, they are ordered according to the timestamps of the corresponding documents.

3 Experimental Setup

3.1 Datasets

DUC: Both DUC 2006 and DUC 2007 are used in our evaluation. DUC 2006 and DUC 2007 contain 50 and 45 topics respectively. Each topic has 25 news documents and 4 model summaries. The length of the model summary is limited to 250 words. **TAC**: We also use TAC 2010 and TAC 2011 in our experiments. TAC 2011 is the latest standard summarization benchmark data set and it contains 44 topics. Each topic falls into one of 5 predefined event categories and contains 10 related news documents and 4 model summaries. TAC 2010 is used as the parameter tuning data set of our TAC evaluation.

3.2 Settings

For text processing, the input sentences are represented as BOW vectors with dimension k. The dictionary is created using unigrams and named entity terms. The word salience threshold ϵ used in sentence compression is 0.005. For the neural network framework, we set the hidden size as 500. All the neural matrix parameters $\mathcal{W}$ in hidden layers and RNN layers are initialized from a uniform distribution between $[-0.1, 0.1]$. Adadelta (Schmidhuber, 2015) is used for gradient based optimization. Gradient clipping is adopted by scaling gradients then the norm exceeded a threshold of 10. The maximum epoch number in the optimization procedure is 200. We limit the number of distilled vectors $n = 5$. The attention cascaded parameter λ_a and λ_c can be learned by our model. The sparsity penalty λ_s in Equation 14 is

[3] http://lpsolve.sourceforge.net/5.5/

Table 1: Comparisons on TAC 2010

System	R-1	R-2	R-SU4
CW	0.353	0.092	0.123
SC	0.346	0.083	0.116
AttenC-tensor-gru	0.339	0.078	0.115
AttenC-concat-gru	0.353	0.089	0.121
AttenC-dot-lstm	0.352	0.089	0.121
AttenH-dot-gru	0.348	0.086	0.119
AttenO-dot-gru	0.348	0.085	0.118
AttenC-dot-gru	**0.359**	**0.092**	**0.124**
(w\o coarse-comp)	0.351	0.089	0.122

0.001. Our neural network based framework is implemented using Theano (Bastien et al., 2012) on a single GPU of Tesla K80.

We use ROUGE score as our evaluation metric (Lin, 2004) with standard options. F-measures of ROUGE-1 (R-1), ROUGE-2 (R-2) and ROUGE-SU4 (R-SU4) are reported.

4 Results and Discussions

4.1 Effect of Existing Salience Models and Different Attention Architectures

We quantitatively evaluate the performance of different variants on the dataset of TAC 2010. The experimental results are shown in Table 1. Note that the summary generation phase for different methods are the same, and only the salience estimation methods are different. Commonly used existing methods for salience estimation include: concept weight (**CW**) (Bing et al., 2015) and sparse coding (**SC**) (Li et al., 2015). As mentioned in Section 2.2.3, there are several alternatives for the attention scoring function $score(\cdot)$: **dot**, **tensor**, and **concat**. Moreover, we also design experiments to show the benefit of our cascaded attention mechanism versus the single attention method. **AttenC** denotes the cascaded attention mechanism. **AttenH** and **AttenO** represent the attention only on the hidden layer or the output layer respectively without cascaded combination.

Among all the methods, the cascaded attention model with *dot* structure achieves the best performance. The effect of different RNN models, such as LSTM and GRU, is similar. However, there are less parameters in GRU resulting in improvements for the efficiency of training. Therefore, we choose **AttenC-dot-gru** as the attention structure of our framework in the subsequent experiments. Moreover, the results without coarse-grained sentence compression (Section 2.3.1) show that the compression can indeed improve the sumamrization performance.

Table 2: Results on DUC 2006.

System	R-1	R-2	R-SU4
Random	0.280	0.046	0.088
Lead	0.308	0.048	0.087
LexRank	0.360	0.062	0.118
TextRank	0.373	0.066	0.125
MDS-Sparse	0.340	0.052	0.107
DSDR	0.377	0.073	0.117
RA-MDS	0.391	0.081	0.136
ABS-Phrase	0.392	0.082	0.137
C-Attention	**0.393***	**0.087***	**0.141***

Table 3: Results on DUC 2007.

System	R-1	R-2	R-SU4
Random	0.302	0.046	0.088
Lead	0.312	0.058	0.102
LexRank	0.378	0.075	0.130
TextRank	0.403	0.083	0.144
MDS-Sparse	0.353	0.055	0.112
DSDR	0.398	0.087	0.137
RA-MDS	0.408	0.097	0.150
ABS-Phrase	0.419	0.103	0.156
C-Attention	**0.423***	**0.107***	**0.161***

4.2 Main Results of Compressive MDS

We compare our system **C-Attention** with several unsupervised summarization baselines and state-of-the-art models. **Random** baseline selects sentences randomly for each topic. **Lead** baseline (Wasson, 1998) ranks the news chronologically and extracts the leading sentences one by one. **TextRank** (Mihalcea and Tarau, 2004) and **LexRank** (Erkan and Radev, 2004a) estimate sentence salience by applying the PageRank algorithm to the sentence graph. **PKUTM** (Li et al., 2011) employs manifold-ranking for sentence scoring and selection; **ABS-Phrase** (Bing et al., 2015) generates abstractive summaries using phrase-based optimization framework. Three other unsupervised methods based on sparse coding are also compared, namely, **DSDR** (He et al., 2012), **MDS-Sparse** (Liu et al., 2015), and **RA-MDS** (Li et al., 2015).

As shown in Table 2, Table 3, and Table 4, our system achieves the best results on all the ROUGE

Table 4: Results on TAC 2011.

System	R-1	R-2	R-SU4
Random	0.303	0.045	0.090
Lead	0.315	0.071	0.103
LexRank	0.313	0.060	0.102
TextRank	0.332	0.064	0.107
PKUTM	0.396	0.113	0.148
ABS-Phrase	0.393	0.117	0.148
RA-MDS	0.400	0.117	0.151
C-Attention	**0.400***	**0.121***	**0.153***

* Statistical significance tests show that our method is better than the best baselines.

metrics. The reasons are as follows: (1) The attention model can directly capture the salient sentences, which are obtained by minimizing the global data reconstruction error; (2) The cascaded structure of attentions can jointly consider the embedding vector space and bag-of-words vector space when conducting the estimation of sentence salience; (3) The coarse-grained sentence compression based on distilled word salience, and the fine-grained compression via phrase-based unified optimization framework can generate more concise and salient summaries. It is worth noting that PKUTM used a Wikipedia corpus for providing domain knowledge. The system **SWING** (Min et al., 2012) is the best system for TAC 2011. Our results are not as good as SWING. The reason is that SWING employs category-specific features and requires supervised training. These features help them select better category-specific content for the summary. In contrast, our model is basically **unsupervised**.

4.3 Linguistic Quality Evaluation

The linguistic quality of summaries generated by ABS-Phrase, PKUTM, and our model from 20 topics of TAC 2011 is evaluated using the five linguistic quality questions on grammaticality (Q1), non-redundancy (Q2), referential clarity (Q3), focus (Q4), and coherence (Q5) in Document Understanding Conferences (DUC). A Likert scale with five levels is employed with 5 being very good with 1 being very poor. A summary was blindly evaluated by three assessors on each question. The results are given in Table 5. PKUTM is an extractive method that picks the original sentences, hence it achieves higher score in Q1 grammaticality. ABS-Phrase is an abstractive method and can generate new sentences by merging different phrases, which decreases the grammaticality. Grammaticality of our compression-based framework is better than ABS-Phrase, but not as good as PKUTM. However, our framework performs the best on some other metrics such as Q2 (non-redundancy) and Q4 (focus). The reason is that our framework can compress and remove some uncritical and redundancy content from the original sentences, which leads to better performance on Q2 and Q4.

Table 5: Evaluation of linguistic quality.

System	Q1	Q2	Q3	Q4	Q5	AVG
ABS-Phrase	3.75	3.38	3.75	3.35	3.12	3.47
PKUTM	4.13	3.45	3.83	3.33	2.92	3.53
Ours	3.96	3.50	3.79	3.50	3.25	3.60

Table 6: Top-10 terms extracted from each topic according to the word salience

Topic 1	Topic 2	Topic 3
school	heart	HIV
shooting	disease	Africa
Auvinen	study	circumcision
Finland	risk	study
police	test	infection
video	blood	trial
Wednesday	red	woman
gunman	telomere	drug
post	level	health

4.4 Case Study: Distilled Word Salience

As mentioned above, the output vectors S in our neural model contain the distilled word salience information. In order to show the performance of word salience estimation, we select 3 topics (events) from different categories of TAC 2011: “Finland Shooting”, “Heart Disease”, and “Hiv Infection Africa”. For each topic, we sort the dictionary terms according to their salience scores, and extract the top-10 terms as the salience estimation results as shown in Table 6. We can see that the top-10 terms reveal the most important information of each topic. For the topic “Finland Shooting”, there is a sentence from the golden summary “A teenager at a school in Finland went on a shooting rampage Wednesday, November 11, 2007, killing 8 people, then himself.” It is obvious that the top-10 terms from Table 6 can capture this main point.

4.5 Case Study: Attention-based Sentence Salience

In our model, the distilled attention matrix A^o can be treated as sentence salience estimation. Let $\widehat{a}$ be the magnitude of the columns in A^o and $\widehat{a} \in \mathbb{R}^m$. $\widehat{a}_i$ represents the salience of the sentence x_i. We collect all the attention vectors for 8 topics of TAC 2011, and display them as an image as shown in Figure 2. The x-axis represents the sentence id (we show at most 100 sentences), and the y-axis represents the topic id. The gray level of pixels in the image indicates different salience scores, where $dark$ represents a high salience score and $light$ represents a small score. Note that different topics seem to hold different ranges of salience scores because they have different number of sentences, i.e. m. According to Equation 9, topics containing more sentences will distribute the attention to more units, therefore, each sentence will get a relatively smaller attention weight. But this issue does not affect the performance of MDS since different topics are independently processed.

In Figure 2, there are some chunks in each topic (see Topic 3 as an example) having higher attention weights, which indeed automatically captures one characteristic of MDS: *sentence position is an important feature for news summarization*. As observed by several previous studies (Li et al., 2015; Min et al., 2012), the sentences in the beginning of a news document are usually more important and tend to be used for writing model summaries. Manual checking verified that those high-attention chunks correspond to the beginning sentences. Our model is able to automatically capture this information by assigning the latter sentences in each topic lower attention weights.

4.6 Summary Case Analysis

Table 7 shows the summary of the topic "*Hawkins Robert Van Maur*" in TAC 2011. The summary contains four sentences, which are all compressed with different compression ratio. Some uncritical information is excluded from the summary sentences, such as "*police said Thursday*" in S2, "*But*" in S3, and "*he said*" in S4. In addition, the VP "*killing eight people*" in S2 is also excluded since it is duplicate with the phrase "killed eight people" in S3. Moreover, from the case we can find that the compression operation did not harm the linguistic quality.

Table 7: The summary of the topic "*Hawkins Robert Van Maur*".

S1: The young gunman who opened fire at a mall busy with holiday shoppers appeared to choose his victims at random, according to police[~~, but a note he left behind hinted at a troubled life~~].
S2: The teenage gunman who went on a shooting rampage in a department store, [~~killing eight people,~~] may have smuggled an assault rifle into the mall underneath clothing[~~, police said Thursday~~].
S3: [~~But~~] police said it was Hawkins who went into an Omaha shopping mall on Wednesday and began a shooting rampage that killed eight people.
S4: Mall security officers noticed Hawkins briefly enter the Von Maur department store at Omaha's Westroads Mall earlier Wednesday[~~, he said~~].

5 Related Works

According to different machine learning paradigms, summarization models can be divided into supervised framework and unsupervised framework. Some previous works have been proposed based on unsupervised models. For example, Mihalcea and Tarau (2004) and Erkan and Radev (2004a) estimated sentence salience by applying the PageRank algorithm to the sentence graph. He et al. (2012), Liu et al. (2015), Li et al. (2015) and Song et al. (2017) employed sparse coding techniques for finding the salient sentences as summaries. Li et al. (2017) conducted salience estimation jointly considering reconstructions on several different vector spaces generated by a variational auto-ecoder framework.

Some recent works utilize attention modeling based recurrent neural networks to tackle the task of single-document summarization. Rush et al. (2015) proposed a sentence summarization framework based on a neural attention model using a supervised sequence-to-sequence neural machine translation model. Gu et al. (2016) combined a copying mechanism with the seq2seq framework to improve the quality of the generated summaries. Nallapati et al. (2016) also employed the typical attention modeling based seq2seq framework, but utilized a trick to control the vocabulary size to improve the training efficiency. However, few previous works employ attention mechanism to tackle the unsupervised MDS problem. In contrast, our attention-based framework can generate summaries for multi-document summarization

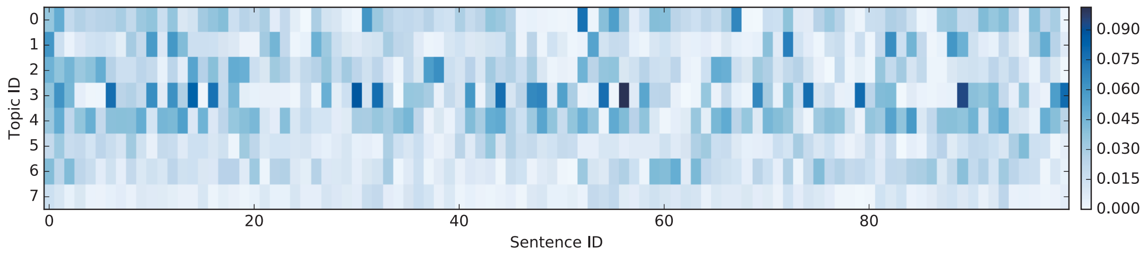


Figure 2: Visualization for sentence attention.

settings in an unsupervised manner.

6 Conclusions

We propose a cascaded neural attention based unsupervised salience estimation method for compressive multi-document summarization. The attention weights for sentences and salience values for words are both learned by data reconstruction in an unsupervised manner. We thoroughly investigate the performance of combining different attention architectures and cascaded structures. Experimental results on some benchmark data sets show that our framework achieves good performance compared with the state-of-the-art methods.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. 2012. Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *ACL*, pages 1587–1597.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *ACL*, pages 484–494.

Kyunghyun Cho, Bart van Merriënboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. *EMNLP*, pages 1724–1734.

George B Dantzig and Mukund N Thapa. 2006. *Linear programming 1: introduction*. Springer Science & Business Media.

Harold P Edmundson. 1969. New methods in automatic extracting. *Journal of the ACM (JACM)*, 16(2):264–285.

Günes Erkan and Dragomir R Radev. 2004a. Lexpagerank: Prestige in multi-document text summarization. In *EMNLP*, volume 4, pages 365–371.

Günes Erkan and Dragomir R Radev. 2004b. Lexrank: Graph-based lexical centrality as salience in text summarization. *JAIR*, pages 457–479.

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. 2000. Multi-document summarization by sentence extraction. In *NAACL-ANLPWorkshop*, pages 40–48.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*, pages 1631–1640.

Zhanying He, Chun Chen, Jiajun Bu, Can Wang, Lijun Zhang, Deng Cai, and Xiaofei He. 2012. Document summarization based on data reconstruction. In *AAAI*, pages 620–626.

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

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

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *EMNLP*, pages 490–500.

Huiying Li, Yue Hu, Zeyuan Li, Xiaojun Wan, and Jianguo Xiao. 2011. PKUTM participation in TAC2011. In *TAC*.

Piji Li, Lidong Bing, Wai Lam, Hang Li, and Yi Liao. 2015. Reader-aware multi-document summarization via sparse coding. In *IJCAI*, pages 1270–1276.

Piji Li, Zihao Wang, Wai Lam, Zhaochun Ren, and Lidong Bing. 2017. Salience estimation via variational auto-encoders for multi-document summarization. In *AAAI*, pages 3497–3503.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8.

He Liu, Hongliang Yu, and Zhi-Hong Deng. 2015. Multi-document summarization based on two-level sparse representation model. In *AAAI*, pages 196–202.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*, pages 1412–1421.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *ECIR*, pages 557–564. Springer.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *EMNLP*, pages 404–411.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Ziheng Lin Min, Yen Kan Chew, and Lim Tan. 2012. Exploiting category-specific information for multi-document summarization. *COLING*, pages 2093–2108.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. In *Mining Text Data*, pages 43–76. Springer.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *EMNLP*, pages 379–389.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Hongya Song, Zhaochun Ren, Shangsong Liang, Piji Li, Jun Ma, and Maarten de Rijke. 2017. Summarizing answers in non-factoid community question-answering. In *WSDM*, pages 405–414. ACM.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Manifold-ranking based topic-focused multi-document summarization. In *IJCAI*, volume 7, pages 2903–2908.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *ACL*, pages 1384–1394.

Mark Wasson. 1998. Using leading text for news summaries: Evaluation results and implications for commercial summarization applications. In *ACL*, pages 1364–1368.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *EMNLP-CNLL*, pages 233–243.

Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2015. Compressive document summarization via sparse optimization. In *IJCAI*, pages 1376–1382.

Deep Recurrent Generative Decoder for Abstractive Text Summarization[*]

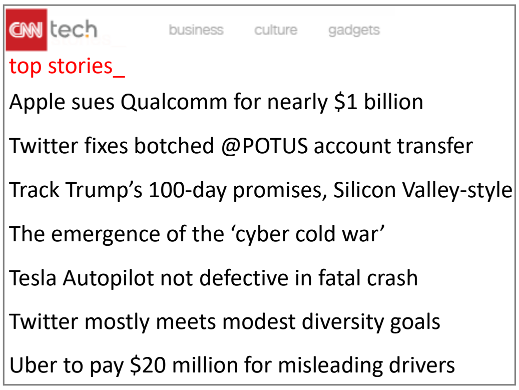
Piji Li[†] **Wai Lam**[†] **Lidong Bing**[‡] **Zihao Wang**[†]
[†]Key Laboratory on High Confidence Software Technologies (Sub-Lab, CUHK),
Ministry of Education, China
[†]Department of Systems Engineering and Engineering Management,
The Chinese University of Hong Kong
[‡]AI Lab, Tencent Inc., Shenzhen, China
[†]{pjli, wlam, zhwang}@se.cuhk.edu.hk, [‡]lyndonbing@tencent.com

Abstract

We propose a new framework for abstractive text summarization based on a sequence-to-sequence oriented encoder-decoder model equipped with a deep recurrent generative decoder (DRGN). Latent structure information implied in the target summaries is learned based on a recurrent latent random model for improving the summarization quality. Neural variational inference is employed to address the intractable posterior inference for the recurrent latent variables. Abstractive summaries are generated based on both the generative latent variables and the discriminative deterministic states. Extensive experiments on some benchmark datasets in different languages show that DRGN achieves improvements over the state-of-the-art methods.

1 Introduction

Automatic summarization is the process of automatically generating a summary that retains the most important content of the original text document (Edmundson, 1969; Luhn, 1958; Nenkova and McKeown, 2012). Different from the common extraction-based and compression-based methods, abstraction-based methods aim at constructing new sentences as summaries, thus they require a deeper understanding of the text and the capability of generating new sentences, which provide an obvious advantage in improving the focus of a summary, reducing the redundancy, and keeping a good compression rate (Bing et al., 2015; Rush et al., 2015; Nallapati et al., 2016).



Apple sues Qualcomm for nearly $1 billion

Twitter fixes botched @POTUS account transfer

Track Trump's 100-day promises, Silicon Valley-style

The emergence of the 'cyber cold war'

Tesla Autopilot not defective in fatal crash

Twitter mostly meets modest diversity goals

Uber to pay $20 million for misleading drivers

Figure 1: Headlines of the top stories from the channel "Technology" of CNN.

Some previous research works show that human-written summaries are more abstractive (Jing and McKeown, 2000). Moreover, our investigation reveals that people may naturally follow some inherent structures when they write the abstractive summaries. To illustrate this observation, we show some examples in Figure 1, which are some top story summaries or headlines from the channel "Technology" of CNN. After analyzing the summaries carefully, we can find some common structures from them, such as "**What**", "**What-Happened**" , "**Who Action What**", etc. For example, the summary "*Apple sues Qualcomm for nearly $1 billion*" can be structuralized as "Who (Apple) Action (sues) What (Qualcomm)". Similarly, the summaries "*[Twitter] [fixes] [botched @POTUS account transfer]*", "*[Uber] [to pay] [$20 million] for misleading drivers*", and "*[Bipartisan bill] aims to [reform] [H-1B visa system]*" also follow the structure of "Who Action What". The summary "*The emergence of the 'cyber cold war'*" matches with the structure of "What", and the summary "*St. Louis' public library computers hacked*" follows

[*]The work described in this paper is supported by a grant from the Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14203414).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2091–2100
Copenhagen, Denmark, September 7–11, 2017.

the structure of “What-Happened”.

Intuitively, if we can incorporate the latent structure information of summaries into the abstractive summarization model, it will improve the quality of the generated summaries. However, very few existing works specifically consider the latent structure information of summaries in their summarization models. Although a very popular neural network based sequence-to-sequence (seq2seq) framework has been proposed to tackle the abstractive summarization problem (Lopyrev, 2015; Rush et al., 2015; Nallapati et al., 2016), the calculation of the internal decoding states is entirely deterministic. The deterministic transformations in these discriminative models lead to limitations on the representation ability of the latent structure information. Miao and Blunsom (2016) extended the seq2seq framework and proposed a generative model to capture the latent summary information, but they did not consider the recurrent dependencies in their generative model leading to limited representation ability.

To tackle the above mentioned problems, we design a new framework based on sequence-to-sequence oriented encoder-decoder model equipped with a latent structure modeling component. We employ Variational Auto-Encoders (VAEs) (Kingma and Welling, 2013; Rezende et al., 2014) as the base model for our generative framework which can handle the inference problem associated with complex generative modeling. However, the standard framework of VAEs is not designed for sequence modeling related tasks. Inspired by (Chung et al., 2015), we add historical dependencies on the latent variables of VAEs and propose a deep recurrent generative decoder (DRGD) for latent structure modeling. Then the standard discriminative deterministic decoder and the recurrent generative decoder are integrated into a unified decoding framework. The target summaries will be decoded based on both the discriminative deterministic variables and the generative latent structural information. All the neural parameters are learned by back-propagation in an end-to-end training paradigm.

The main contributions of our framework are summarized as follows: (1) We propose a sequence-to-sequence oriented encoder-decoder model equipped with a deep recurrent generative decoder (DRGD) to model and learn the latent structure information implied in the target summaries of the training data. Neural variational inference is employed to address the intractable posterior inference for the recurrent latent variables. (2) Both the generative latent structural information and the discriminative deterministic variables are jointly considered in the generation process of the abstractive summaries. (3) Experimental results on some benchmark datasets in different languages show that our framework achieves better performance than the state-of-the-art models.

2 Related Works

Automatic summarization is the process of automatically generating a summary that retains the most important content of the original text document (Nenkova and McKeown, 2012). Traditionally, the summarization methods can be classified into three categories: extraction-based methods (Erkan and Radev, 2004; Goldstein et al., 2000; Wan et al., 2007; Min et al., 2012; Nallapati et al., 2017; Cheng and Lapata, 2016; Cao et al., 2016; Song et al., 2017), compression-based methods (Li et al., 2013; Wang et al., 2013; Li et al., 2015, 2017), and abstraction-based methods. In fact, previous investigations show that human-written summaries are more abstractive (Barzilay and McKeown, 2005; Bing et al., 2015). Abstraction-based approaches can generate new sentences based on the facts from different source sentences. Barzilay and McKeown (2005) employed sentence fusion to generate a new sentence. Bing et al. (2015) proposed a more fine-grained fusion framework, where new sentences are generated by selecting and merging salient phrases. These methods can be regarded as a kind of indirect abstractive summarization, and complicated constraints are used to guarantee the linguistic quality.

Recently, some researchers employ neural network based framework to tackle the abstractive summarization problem. Rush et al. (2015) proposed a neural network based model with local attention modeling, which is trained on the Gigaword corpus, but combined with an additional log-linear extractive summarization model with handcrafted features. Gu et al. (2016) integrated a copying mechanism into a seq2seq framework to improve the quality of the generated summaries. Chen et al. (2016) proposed a new attention mechanism that not only considers the important source segments, but also distracts them in the decoding step in order to better grasp the overall meaning of input documents. Nallapati et al. (2016) utilized a trick to control the vocabulary size to improve the training efficiency. The calculations in these methods are all deterministic and the representation ability is limited. Miao and Blunsom (2016) extended the seq2seq framework and proposed a generative model to capture the latent summary information, but they do not consider the recurrent dependencies in their generative model leading to limited representation ability.

Some research works employ topic models to capture the latent information from source documents or sentences. Wang et al. (2009) proposed a new Bayesian sentence-based topic model by making use of both the term-document and term-sentence associations to improve the performance of sentence selection. Celikyilmaz and Hakkani-Tur (2010) estimated scores for sentences based on their latent characteristics using a hierarchical topic model, and trained a regression model to extract sentences. However, they only use the latent topic information to conduct the sentence salience estimation for extractive summarization. In contrast, our purpose is to model and learn the latent structure information from the target summaries and use it to enhance the performance of abstractive summarization.

3 Framework Description

3.1 Overview

As shown in Figure 2, the basic framework of our approach is a neural network based encoder-decoder framework for sequence-to-sequence learning. The input is a variable-length sequence $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ representing the source text. The word embedding $\mathbf{x}_t$ is initialized randomly and learned during the optimization process. The output is also a sequence $Y = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\}$, which represents the generated abstractive summaries. Gated Recurrent Unit (GRU) (Cho et al., 2014) is employed as the basic sequence modeling component for the encoder and the decoder. For latent structure modeling, we add historical dependencies on the latent variables of Variational Auto-Encoders (VAEs) and propose a deep recurrent generative decoder (DRGD) to distill the complex latent structures implied in the target summaries of the training data. Finally, the abstractive summaries will be decoded out based on both the discriminative deterministic variables H and the generative latent structural information Z.

3.2 Recurrent Generative Decoder

Assume that we have obtained the source text representation $\mathbf{h}^e \in \mathbb{R}^{k_h}$. The purpose of the decoder is to translate this source code $\mathbf{h}^e$ into a series of hidden states $\{\mathbf{h}_1^d, \mathbf{h}_2^d, \dots, \mathbf{h}_n^d\}$, and then revert these hidden states to an actual word sequence and generate the summary.

For standard recurrent decoders, at each time step t, the hidden state $\mathbf{h}_t^d \in \mathbb{R}^{k_h}$ is calculated using the dependent input symbol $\mathbf{y}_{t-1} \in \mathbb{R}^{k_w}$ and the previous hidden state $\mathbf{h}_{t-1}^d$:

$$\mathbf{h}_t^d = f(\mathbf{y}_{t-1}, \mathbf{h}_{t-1}^d) \quad (1)$$

where $f(\cdot)$ is a recurrent neural network such as vanilla RNN, Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), and Gated Recurrent Unit (GRU) (Cho et al., 2014). No matter which one we use for $f(\cdot)$, the common transformation operation is as follows:

$$\mathbf{h}_t^d = g(\mathbf{W}_{yh}^d \mathbf{y}_{t-1} + \mathbf{W}_{hh}^d \mathbf{h}_{t-1}^d + \mathbf{b}_h^d) \quad (2)$$

where $\mathbf{W}_{yh}^d \in \mathbb{R}^{k_h \times k_w}$ and $\mathbf{W}_{hh}^d \in \mathbb{R}^{k_h \times k_h}$ are the linear transformation matrices. $\mathbf{b}_h^d$ is the bias. k_h is the dimension of the hidden layers, and k_w is the dimension of the word embeddings. $g(\cdot)$ is the non-linear activation function. From Equation 2, we can see that all the transformations are deterministic, which leads to a deterministic recurrent hidden state h_t^d. From our investigations, we find that the representational power of such deterministic variables are limited. Some more complex latent structures in the target summaries, such as the high-level syntactic features and latent topics, cannot be modeled effectively by the deterministic operations and variables.

Recently, a generative model called Variational Auto-Encoders (VAEs) (Kingma and Welling, 2013; Rezende et al., 2014) shows strong capability in modeling latent random variables and improves the performance of tasks in different fields such as sentence generation (Bowman et al., 2016) and image generation (Gregor et al., 2015). However, the standard VAEs is not designed for modeling sequence directly. Inspired by (Chung et al., 2015), we extend the standard VAEs by

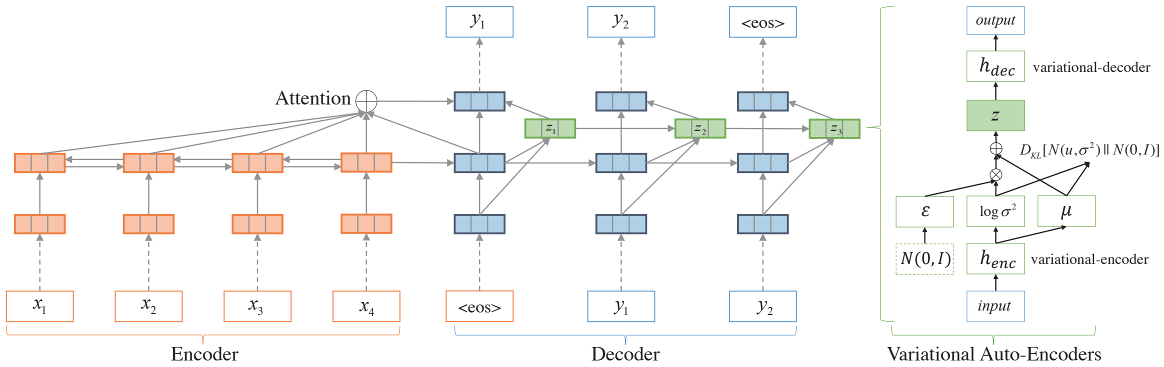


Figure 2: Our deep recurrent generative decoder (DRGD) for latent structure modeling.

introducing the historical latent variable dependencies to make it be capable of modeling sequence data. Our proposed latent structure modeling framework can be viewed as a sequence generative model which can be divided into two parts: inference (variational-encoder) and generation (variational-decoder). As shown in the decoder component of Figure 2, the input of the original VAEs only contains the observed variable $\mathbf{y}_t$, and the variational-encoder can map it to a latent variable $\mathbf{z} \in \mathbb{R}^{k_z}$, which can be used to reconstruct the original input. For the task of summarization, in the sequence decoder component, the previous latent structure information needs to be considered for constructing more effective representations for the generation of the next state.

For the inference stage, the variational-encoder can map the observed variable $\mathbf{y}_{<t}$ and the previous latent structure information $\mathbf{z}_{<t}$ to the posterior probability distribution of the latent structure variable $p_\theta(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})$. It is obvious that this is a recurrent inference process in which $\mathbf{z}_t$ contains the historical dynamic latent structure information. Compared with the variational inference process $p_\theta(\mathbf{z}_t|\mathbf{y}_t)$ of the typical VAEs model, the recurrent framework can extract more complex and effective latent structure features implied in the sequence data.

For the generation process, based on the latent structure variable $\mathbf{z}_t$, the target word y_t at the time step t is drawn from a conditional probability distribution $p_\theta(\mathbf{y}_t|\mathbf{z}_t)$. The target is to maximize the probability of each generated summary $y = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T\}$ based on the generation process according to:

$$p_\theta(y) = \prod_{t=1}^{T} \int p_\theta(\mathbf{y}_t|\mathbf{z}_t) p_\theta(\mathbf{z}_t) d\mathbf{z}_t \tag{3}$$

For the purpose of solving the intractable integral of the marginal likelihood as shown in Equation 3, a recognition model $q_\phi(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})$ is introduced as an approximation to the intractable true posterior $p_\theta(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})$. The recognition model parameters ϕ and the generative model parameters θ can be learned jointly. The aim is to reduce the Kulllback-Leibler divergence (KL) between $q_\phi(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})$ and $p_\theta(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})$:

$$\begin{aligned} & D_{KL}[q_\phi(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t}) \| p_\theta(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})] \\ & = \int_z q_\phi(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t}) \log \frac{q_\phi(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})}{p_\theta(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})} dz \\ & = \mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})}[\log q_\phi(\mathbf{z}_t|\cdot) - \log p_\theta(\mathbf{z}_t|\cdot)] \end{aligned}$$

where $\cdot$ denotes the conditional variables $\mathbf{y}_{<t}$ and $\mathbf{z}_{<t}$. Bayes rule is applied to $p_\theta(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})$, and we can extract $\log p_\theta(\mathbf{z})$ from the expectation, transfer the expectation term $\mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})}$ back to KL-divergence, and rearrange all the terms. Consequently the following holds:

$$\begin{aligned} & \log p_\theta(\mathbf{y}_{<t}) = \\ & D_{KL}[q_\phi(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t}) \| p_\theta(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})] \\ & + \mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})}[\log p_\theta(\mathbf{y}_{<t}|\mathbf{z}_t)] \\ & - D_{KL}[q_\phi(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t}) \| p_\theta(\mathbf{z}_t)] \end{aligned} \tag{4}$$

Let $\mathcal{L}(\theta, \phi; y)$ represent the last two terms from the right part of Equation 4:

$$\begin{aligned} & \mathcal{L}(\theta, \varphi; y) = \\ & \mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})} \Big\{ \sum_{t=1}^{T} \log p_\theta(\mathbf{y}_t|\mathbf{z}_t) \\ & \quad - D_{KL}[q_\phi(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t}) \| p_\theta(\mathbf{z}_t)] \Big\} \end{aligned} \tag{5}$$

Since the first KL-divergence term of Equation 4 is non-negative, we have $\log p_\theta(\mathbf{y}_{<t}) \geq \mathcal{L}(\theta, \phi; y)$ meaning that $\mathcal{L}(\theta, \phi; y)$ is a lower bound (the objective to be maximized) on the marginal likelihood. In order to differentiate and optimize the

lower bound $\mathcal{L}(\theta, \phi; y)$, following the core idea of VAEs, we use a neural network framework for the probabilistic encoder $q_\phi(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t})$ for better approximation.

3.3 Abstractive Summary Generation

We also design a neural network based framework to conduct the variational inference and generation for the recurrent generative decoder component similar to some design in previous works (Kingma and Welling, 2013; Rezende et al., 2014; Gregor et al., 2015). The encoder component and the decoder component are integrated into a unified abstractive summarization framework. Considering that GRU has comparable performance but with less parameters and more efficient computation, we employ GRU as the basic recurrent model which updates the variables according to the following operations:

$$\begin{aligned}
\mathbf{r}_t &= \sigma(\mathbf{W}_{xr}\mathbf{x}_t + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{b}_r)\\
\mathbf{z}_t &= \sigma(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{b}_z)\\
\mathbf{g}_t &= \tanh(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h)\\
\mathbf{h}_t &= \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \mathbf{g}_t
\end{aligned}$$

where $\mathbf{r}_t$ is the reset gate, $\mathbf{z}_t$ is the update gate. $\odot$ denotes the element-wise multiplication. *tanh* is the hyperbolic tangent activation function.

As shown in the left block of Figure 2, the encoder is designed based on bidirectional recurrent neural networks. Let $\mathbf{x}_t$ be the word embedding vector of the t-th word in the source sequence. GRU maps $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$ to the current hidden state $\mathbf{h}_t$ in feed-forward direction and back-forward direction respectively:

$$\begin{aligned}
\overrightarrow{\mathbf{h}}_t &= GRU(x_t, \overrightarrow{\mathbf{h}}_{t-1})\\
\overleftarrow{\mathbf{h}}_t &= GRU(x_t, \overleftarrow{\mathbf{h}}_{t-1})
\end{aligned} \tag{6}$$

Then the final hidden state $\mathbf{h}^e_t \in \mathbb{R}^{2k_h}$ is concatenated using the hidden states from the two directions: $\mathbf{h}^e_t = \overrightarrow{\mathbf{h}}_t || \overleftarrow{\mathbf{h}}$. As shown in the middle block of Figure 2, the decoder consists of two components: discriminative deterministic decoding and generative latent structure modeling.

The discriminative deterministic decoding is an improved attention modeling based recurrent sequence decoder. The first hidden state $\mathbf{h}^d_1$ is initialized using the average of all the source input states: $\mathbf{h}^d_1 = \frac{1}{T^e}\sum_{t=1}^{T^e}\mathbf{h}^e_t$, where $\mathbf{h}^e_t$ is the source input hidden state. T^e is the input sequence length. The deterministic decoder hidden state $\mathbf{h}^d_t$ is calculated using two layers of GRUs. On the first layer, the hidden state is calculated only using the current input word embedding $\mathbf{y}_{t-1}$ and the previous hidden state $\mathbf{h}^{d_1}_{t-1}$:

$$\mathbf{h}^{d_1}_t = GRU_1(\mathbf{y}_{t-1}, \mathbf{h}^{d_1}_{t-1}) \tag{7}$$

where the superscript d_1 denotes the first decoder GRU layer. Then the attention weights at the time step t are calculated based on the relationship of $\mathbf{h}^{d_1}_t$ and all the source hidden states $\{\mathbf{h}^e_t\}$. Let $a_{i,j}$ be the attention weight between $\mathbf{h}^{d_1}_i$ and $\mathbf{h}^e_j$, which can be calculated using the following formulation:

$$\begin{aligned}
a_{i,j} &= \frac{\exp(e_{i,j})}{\sum_{j'=1}^{T^e}\exp(e_{i,j'})}\\
e_{i,j} &= \mathbf{v}^T\tanh(\mathbf{W}^d_{hh}\mathbf{h}^{d_1}_i + \mathbf{W}^e_{hh}\mathbf{h}^e_j + \mathbf{b}_a)
\end{aligned}$$

where $\mathbf{W}^d_{hh} \in \mathbb{R}^{k_h \times k_h}$, $\mathbf{W}^e_{hh} \in \mathbb{R}^{k_h \times 2k_h}$, $\mathbf{b}_a \in \mathbb{R}^{k_h}$, and $\mathbf{v} \in \mathbb{R}^{k_h}$. The attention context is obtained by the weighted linear combination of all the source hidden states:

$$\mathbf{c}_t = \sum_{j'=1}^{T^e} a_{t,j'}\mathbf{h}^e_{j'} \tag{8}$$

The final deterministic hidden state $\mathbf{h}^{d_2}_t$ is the output of the second decoder GRU layer, jointly considering the word $\mathbf{y}_{t-1}$, the previous hidden state $\mathbf{h}^{d_2}_{t-1}$, and the attention context $\mathbf{c}_t$:

$$\mathbf{h}^{d_2}_t = GRU_2(\mathbf{y}_{t-1}, \mathbf{h}^{d_2}_{t-1}, \mathbf{c}_t) \tag{9}$$

For the component of recurrent generative model, inspired by some ideas in previous works (Kingma and Welling, 2013; Rezende et al., 2014; Gregor et al., 2015), we assume that both the prior and posterior of the latent variables are Gaussian, i.e., $p_\theta(\mathbf{z}_t) = \mathcal{N}(0, \mathbf{I})$ and $q_\phi(\mathbf{z}_t|\mathbf{y}_{<t}, \mathbf{z}_{<t}) = \mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I})$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ denote the variational mean and standard deviation respectively, which can be calculated via a multilayer perceptron. Precisely, given the word embedding $\mathbf{y}_{t-1}$, the previous latent structure variable $\mathbf{z}_{t-1}$, and the previous deterministic hidden state $\mathbf{h}^d_{t-1}$, we first project it to a new hidden space:

$$\mathbf{h}^{e_z}_t = g(\mathbf{W}^{e_z}_{yh}\mathbf{y}_{t-1} + \mathbf{W}^{e_z}_{zh}\mathbf{z}_{t-1} + \mathbf{W}^{e_z}_{hh}\mathbf{h}^d_{t-1} + \mathbf{b}^{e_z}_h)$$

where $\mathbf{W}^{e_z}_{yh} \in \mathbb{R}^{k_h \times k_w}$, $\mathbf{W}^{e_z}_{zh} \in \mathbb{R}^{k_h \times k_z}$, $\mathbf{W}^{e_z}_{hh} \in \mathbb{R}^{k_h \times k_h}$, and $\mathbf{b}^{e_z}_h \in \mathbb{R}^{k_h}$. g is the sigmoid activation function: $\sigma(\mathbf{x}) = 1/(1 + e^{-\mathbf{x}})$. Then the

Gaussian parameters $\boldsymbol{\mu}_t \in \mathbb{R}^{k_z}$ and $\boldsymbol{\sigma}_t \in \mathbb{R}^{k_z}$ can be obtained via a linear transformation based on $\mathbf{h}_t^{e_z}$:

$$\begin{aligned} \boldsymbol{\mu}_t &= \mathbf{W}_{h\mu}^{e_z}\mathbf{h}_t^{e_z} + \mathbf{b}_{\mu}^{e_z} \\ \log(\boldsymbol{\sigma}_t^2) &= \mathbf{W}_{h\sigma}\mathbf{h}_t^{e_z} + \mathbf{b}_{\sigma}^{e_z} \end{aligned} \tag{10}$$

The latent structure variable $\mathbf{z}_t \in \mathbb{R}^{k_z}$ can be calculated using the reparameterization trick:

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \;\; \mathbf{z}_t = \boldsymbol{\mu}_t + \boldsymbol{\sigma}_t \otimes \varepsilon \tag{11}$$

where $\boldsymbol{\varepsilon} \in \mathbb{R}^{k_z}$ is an auxiliary noise variable. The process of inference for finding $\mathbf{z}_t$ based on neural networks can be teated as a variational encoding process.

To generate summaries precisely, we first integrate the recurrent generative decoding component with the discriminative deterministic decoding component, and map the latent structure variable $\mathbf{z}_t$ and the deterministic decoding hidden state $\mathbf{h}_t^{d_2}$ to a new hidden variable:

$$\mathbf{h}_t^{d_y} = tanh(\mathbf{W}_{zh}^{d_y}\mathbf{z}_t + \mathbf{W}_{hh}^{d_z}\mathbf{h}_t^{d_2} + \mathbf{b}_h^{d_y}) \tag{12}$$

Given the combined decoding state $\mathbf{h}_t^{d_y}$ at the time t, the probability of generating any target word y_t is given as follows:

$$\mathbf{y}_t = \varsigma(\mathbf{W}_{hy}^{d}\mathbf{h}_t^{d_y} + \mathbf{b}_{hy}^{d}) \tag{13}$$

where $\mathbf{W}_{hy}^{d} \in \mathbb{R}^{k_y \times k_h}$ and $\mathbf{b}_{hy}^{d} \in \mathbb{R}^{k_y}$. $\varsigma(\cdot)$ is the softmax function. Finally, we use a beam search algorithm (Koehn, 2004) for decoding and generating the best summary.

3.4 Learning

Although the proposed model contains a recurrent generative decoder, the whole framework is fully differentiable. As shown in Section 3.3, both the recurrent deterministic decoder and the recurrent generative decoder are designed based on neural networks. Therefore, all the parameters in our model can be optimized in an end-to-end paradigm using back-propagation. We use $\{X\}_N$ and $\{Y\}_N$ to denote the training source and target sequence. Generally, the objective of our framework consists of two terms. One term is the negative log-likelihood of the generated summaries, and the other one is the variational lower bound $\mathcal{L}(\theta, \phi; Y)$ mentioned in Equation 5. Since the variational lower bound $\mathcal{L}(\theta, \phi; Y)$ also contains a likelihood term, we can merge it with the likelihood term of summaries. The final objective function, which needs to be minimized, is formulated as follows:

$$\begin{aligned} \mathcal{J} = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}\Bigg\{ &-\log\left[p(y_t^{(n)}|y_{<t}^{(n)}, X^{(n)})\right] \\ &+ D_{KL}\left[q_\phi(\mathbf{z}_t^{(n)}|\mathbf{y}_{<t}^{(n)}, \mathbf{z}_{<t}^{(n)})\|p_\theta(\mathbf{z}_t^{(n)})\right]\Bigg\} \end{aligned} \tag{14}$$

4 Experimental Setup

4.1 Datesets

We train and evaluate our framework on three popular datasets. **Gigawords** is an English sentence summarization dataset prepared based on Annotated Gigawords[1] by extracting the first sentence from articles with the headline to form a source-summary pair. We directly download the prepared dataset used in (Rush et al., 2015). It roughly contains 3.8M training pairs, 190K validation pairs, and 2,000 test pairs. **DUC-2004**[2] is another English dataset only used for testing in our experiments. It contains 500 documents. Each document contains 4 model summaries written by experts. The length of the summary is limited to 75 bytes. **LCSTS** is a large-scale Chinese short text summarization dataset, consisting of pairs of (short text, summary) collected from Sina Weibo[3] (Hu et al., 2015). We take Part-I as the training set, Part-II as the development set, and Part-III as the test set. There is a score in range $1 \sim 5$ labeled by human to indicate how relevant an article and its summary is. We only reserve those pairs with scores no less than 3. The size of the three sets are 2.4M, 8.7k, and 725 respectively. In our experiments, we only take Chinese character sequence as input, without performing word segmentation.

4.2 Evaluation Metrics

We use ROUGE score (Lin, 2004) as our evaluation metric with standard options. The basic idea of ROUGE is to count the number of overlapping units between generated summaries and the reference summaries, such as overlapped n-grams, word sequences, and word pairs. F-measures of ROUGE-1 (R-1), ROUGE-2 (R-2), ROUGE-L (R-L) and ROUGE-SU4 (R-SU4) are reported.

4.3 Comparative Methods

We compare our model with some baselines and state-of-the-art methods. Because the datasets are

[1] https://catalog.ldc.upenn.edu/ldc2012t21

[2] http://duc.nist.gov/duc2004

[3] http://www.weibo.com

quite standard, so we just extract the results from their papers. Therefore the baseline methods on different datasets may be slightly different.

- **TOPIARY** (Zajic et al., 2004) is the best on DUC2004 Task-1 for compressive text summarization. It combines a system using linguistic based transformations and an unsupervised topic detection algorithm for compressive text summarization.

- **MOSES+** (Rush et al., 2015) uses a phrase-based statistical machine translation system trained on Gigaword to produce summaries. It also augments the phrase table with "deletion" rulesto improve the baseline performance, and MERT is also used to improve the quality of generated summaries.

- **ABS** and **ABS+** (Rush et al., 2015) are both the neural network based models with local attention modeling for abstractive sentence summarization. ABS+ is trained on the Gigaword corpus, but combined with an additional log-linear extractive summarization model with handcrafted features.

- **RNN** and **RNN-context** (Hu et al., 2015) are two seq2seq architectures. RNN-context integrates attention mechanism to model the context.

- **CopyNet** (Gu et al., 2016) integrates a copying mechanism into the sequence-to-sequence framework.

- **RNN-distract** (Chen et al., 2016) uses a new attention mechanism by distracting the historical attention in the decoding steps.

- **RAS-LSTM** and **RAS-Elman** (Chopra et al., 2016) both consider words and word positions as input and use convolutional encoders to handle the source information. For the attention based sequence decoding process, RAS-Elman selects Elman RNN (Elman, 1990) as decoder, and RAS-LSTM selects Long Short-Term Memory architecture (Hochreiter and Schmidhuber, 1997).

- **LenEmb** (Kikuchi et al., 2016) uses a mechanism to control the summary length by considering the length embedding vector as the input.

- **ASC+FSC$_1$** (Miao and Blunsom, 2016) uses a generative model with attention mechanism to conduct the sentence compression problem. The model first draws a latent summary sentence from a background language model, and then subsequently draws the observed sentence conditioned on this latent summary.

- **lvt2k-1sent** and **lvt5k-1sent** (Nallapati et al., 2016) utilize a trick to control the vocabulary size to improve the training efficiency.

4.4 Experimental Settings

For the experiments on the English dataset Gigawords, we set the dimension of word embeddings to 300, and the dimension of hidden states and latent variables to 500. The maximum length of documents and summaries is 100 and 50 respectively. The batch size of mini-batch training is 256. For DUC-2004, the maximum length of summaries is 75 bytes. For the dataset of LCSTS, the dimension of word embeddings is 350. We also set the dimension of hidden states and latent variables to 500. The maximum length of documents and summaries is 120 and 25 respectively, and the batch size is also 256. The beam size of the decoder was set to be 10. Adadelta (Schmidhuber, 2015) with hyperparameter $\rho = 0.95$ and $\epsilon = 1e - 6$ is used for gradient based optimization. Our neural network based framework is implemented using Theano (Theano Development Team, 2016).

5 Results and Discussions

5.1 ROUGE Evaluation

Table 1: ROUGE-F1 on validation sets

Dataset	**System**	**R-1**	**R-2**	**R-L**
GIGA	StanD	32.69	15.29	30.60
	DRGD	**36.25**	**17.61**	**33.55**
LCSTS	StanD	33.88	21.49	31.05
	DRGD	**36.71**	**24.00**	**34.10**

We first depict the performance of our model DRGD by comparing to the standard decoders (StanD) of our own implementation. The comparison results on the validation datasets of Gigawords and LCSTS are shown in Table 1. From the results we can see that our proposed generative decoders DRGD can obtain obvious improvements on abstractive summarization than the standard decoders. Actually, the performance of the standard

Table 2: ROUGE-F1 on Gigawords

System	R-1	R-2	R-L
ABS	29.55	11.32	26.42
ABS+	29.78	11.89	26.97
RAS-LSTM	32.55	14.70	30.03
RAS-Elman	33.78	15.97	31.15
ASC + FSC$_1$	34.17	15.94	31.92
lvt2k-1sent	32.67	15.59	30.64
lvt5k-1sent	35.30	16.64	32.62
DRGD	**36.27**	**17.57**	**33.62**

Table 3: ROUGE-Recall on DUC2004

System	R-1	R-2	R-L
TOPIARY	25.12	6.46	20.12
MOSES+	26.50	8.13	22.85
ABS	26.55	7.06	22.05
ABS+	28.18	8.49	23.81
RAS-Elman	28.97	8.26	24.06
RAS-LSTM	27.41	7.69	23.06
LenEmb	26.73	8.39	23.88
lvt2k-1sen	28.35	9.46	24.59
lvt5k-1sen	28.61	9.42	25.24
DRGD	**31.79**	**10.75**	**27.48**

Table 4: ROUGE-F1 on LCSTS

System	R-1	R-2	R-L
RNN	21.50	8.90	18.60
RNN-context	29.90	17.40	27.20
CopyNet	34.40	21.60	31.30
RNN-distract	35.20	22.60	32.50
DRGD	**36.99**	**24.15**	**34.21**

decoders is similar with those mentioned popular baseline methods.

The results on the English datasets of Gigawords and DUC-2004 are shown in Table 2 and Table 3 respectively. Our model DRGD achieves the best summarization performance on all the ROUGE metrics. Although ASC+FSC$_1$ also uses a generative method to model the latent summary variables, the representation ability is limited and it cannot bring in noticeable improvements. It is worth noting that the methods lvt2k-1sent and lvt5k-1sent (Nallapati et al., 2016) utilize linguistic features such as parts-of-speech tags, named-entity tags, and TF and IDF statistics of the words as part of the document representation. In fact, extracting all such features is a time consuming work, especially on large-scale datasets such as Gigawords. lvt2k and lvt5k are not end-to-end style models and are more complicated than our model in practical applications.

The results on the Chinese dataset LCSTS are shown in Table 4. Our model DRGD also achieves the best performance. Although CopyNet employs a copying mechanism to improve the summary quality and RNN-distract considers attention information diversity in their decoders, our model is still better than those two methods demonstrating that the latent structure information learned from target summaries indeed plays a role in abstractive summarization. We also believe that integrating the copying mechanism and coverage diversity in our framework will further improve the summarization performance.

5.2 Summary Case Analysis

In order to analyze the reasons of improving the performance, we compare the generated summaries by DRGD and the standard decoders StanD used in some other works such as (Chopra et al., 2016). The source texts, golden summaries, and the generated summaries are shown in Table 5. From the cases we can observe that DRGD can indeed capture some latent structures which are consistent with the golden summaries. For example, our result for S(1) "*Wuhan wins men's soccer title at Chinese city games*" matches the "Who Action What" structure. However, the standard decoder StanD ignores the latent structures and generates some loose sentences, such as the results for S(1) "*Results of men's volleyball at Chinese city games*" does not catch the main points. The reason is that the recurrent variational auto-encoders used in our framework have better representation ability and can capture more effective and complicated latent structures from the sequence data. Therefore, the summaries generated by DRGD have consistent latent structures with the ground truth, leading to a better ROUGE evaluation.

6 Conclusions

We propose a deep recurrent generative decoder (DRGD) to improve the abstractive summarization performance. The model is a sequence-to-sequence oriented encoder-decoder framework equipped with a latent structure modeling component. Abstractive summaries are generated based on both the latent variables and the deterministic states. Extensive experiments on benchmark

Table 5: Examples of the generated summaries.

S(1): hosts wuhan won the men 's soccer title by beating beijing shunyi #-# here at the #th chinese city games on friday. **Golden**: hosts wuhan wins men 's soccer title at chinese city games. **StanD**: results of men 's volleyball at chinese city games. **DRGD**: **wuhan wins men 's soccer title at chinese city games.**
S(2): UNK and the china meteorological administration tuesday signed an agreement here on long - and short-term cooperation in projects involving meteorological satellites and satellite meteorology. **Golden**: UNK china to cooperate in meteorology. **StanD**: weather forecast for major chinese cities. **DRGD**: **china to cooperate in meteorological satellites.**
S(3): the rand gained ground against the dollar at the opening here wednesday , to #.# to the greenback from #.# at the close tuesday. **Golden**: rand gains ground. **StanD**: rand slightly higher against dollar. **DRGD**: **rand gains ground against dollar.**
S(4): new zealand women are having more children and the country 's birth rate reached its highest level in ## years , statistics new zealand said on wednesday. **Golden**: new zealand birth rate reaches ##-year high. **StanD**: new zealand women are having more children birth rate hits highest level in ## years. **DRGD**: **new zealand 's birth rate hits ##-year high.**

datasets show that DRGD achieves improvements over the state-of-the-art methods.

References

Regina Barzilay and Kathleen R McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *ACL*, pages 1587–1597.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. *CoNLL*, pages 10–21.

Ziqiang Cao, Wenjie Li, Sujian Li, Furu Wei, and Yanran Li. 2016. Attsum: Joint learning of focusing and summarization with neural attention. *COLING*, pages 547–556.

Asli Celikyilmaz and Dilek Hakkani-Tur. 2010. A hybrid hierarchical model for multi-document summarization. In *ACL*, pages 815–824.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for document summarization. In *IJCAI*, pages 2754–2760.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *ACL*, pages 484–494.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. 2016. Abstractive sentence summarization with attentive recurrent neural networks. *NAACL-HLT*, pages 93–98.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *NIPS*, pages 2980–2988.

Harold P Edmundson. 1969. New methods in automatic extracting. *Journal of the ACM (JACM)*, 16(2):264–285.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. 2000. Multi-document summarization by sentence extraction. In *NAACL-ANLPWorkshop*, pages 40–48.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. 2015. Draw: A recurrent neural network for image generation. In *ICML*, pages 1462–1471.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*, pages 1631–1640.

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

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. Lcsts: A large scale chinese short text summarization dataset. In *EMNLP*, pages 1962–1972.

Hongyan Jing and Kathleen R McKeown. 2000. Cut and paste based text summarization. In *NAACL*, pages 178–185.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *EMNLP*, pages 1328–1338.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Conference of the Association for Machine Translation in the Americas*, pages 115–124. Springer.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013. Document summarization via guided sentence compression. In *EMNLP*, pages 490–500.

Piji Li, Lidong Bing, Wai Lam, Hang Li, and Yi Liao. 2015. Reader-aware multi-document summarization via sparse coding. In *IJCAI*, pages 1270–1276.

Piji Li, Zihao Wang, Wai Lam, Zhaochun Ren, and Lidong Bing. 2017. Salience estimation via variational auto-encoders for multi-document summarization. In *AAAI*, pages 3497–3503.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8.

Konstantin Lopyrev. 2015. Generating news headlines with recurrent neural networks. *arXiv preprint arXiv:1512.01712*.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *EMNLP*, pages 319–328.

Ziheng Lin Min, Yen Kan Chew, and Lim Tan. 2012. Exploiting category-specific information for multi-document summarization. *COLING*, pages 2903–2108.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. In *Mining Text Data*, pages 43–76. Springer.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, pages 1278–1286.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*, pages 379–389.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Hongya Song, Zhaochun Ren, Piji Li, Shangsong Liang, Jun Ma, and Maarten de Rijke. 2017. Summarizing answers in non-factoid community question-answering. In *WSDM*, pages 405–414.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Manifold-ranking based topic-focused multi-document summarization. In *IJCAI*, volume 7, pages 2903–2908.

Dingding Wang, Shenghuo Zhu, Tao Li, and Yihong Gong. 2009. Multi-document summarization using sentence-based topic models. In *ACL-IJCNLP*, pages 297–300.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *ACL*, pages 1384–1394.

David Zajic, Bonnie Dorr, and Richard Schwartz. 2004. Bbn/umd at duc-2004: Topiary. In *HLT-NAACL*, pages 112–119.

Extractive Summarization Using Multi-Task Learning with Document Classification

Masaru Isonuma,Toru Fujino, Junichiro Mori, Yutaka Matsuo and **Ichiro Sakata**
The University of Tokyo, Japan
{isonuma, jmori, isakata}@ipr-ctr.t.u-tokyo.ac.jp,
tr.fujino@scslab.k.u-tokyo.ac.jp, matsuo@weblab.t.u-tokyo.ac.jp

Abstract

The need for automatic document summarization that can be used for practical applications is increasing rapidly. In this paper, we propose a general framework for summarization that extracts sentences from a document using externally related information. Our work is aimed at single document summarization using small amounts of reference summaries. In particular, we address document summarization in the framework of multi-task learning using curriculum learning for sentence extraction and document classification. The proposed framework enables us to obtain better feature representations to extract sentences from documents. We evaluate our proposed summarization method on two datasets: financial report and news corpus. Experimental results demonstrate that our summarizers achieve performance that is comparable to state-of-the-art systems.

1 Introduction

With rapid increase in the volume of textual data that are available both online and offline, the need for automatic document summarization that can be implement in practical scenarios is increasing (Li et al., 2016; Chopra et al., 2016; Takase et al., 2016). Among the several summarization systems, extractive summarization approaches (Erkan and Radev, 2004; McDonald, 2007; Wong et al., 2008) are widely used. These techniques identify and subsequently concatenate relevant sentences automatically from a document to create its summary while preserving its original information content. Such approaches are popular and widely used for practical applications because they are computationally cost-effective and less complex. Extractive summarization approaches based on neural network-based approaches (Kågebäck et al., 2014; Cao et al., 2015; Yin and Pei, 2015; Cao et al., 2016) have advanced rapidly. Recently, an attentional encoder-decoder for extractive single-document summarization was proposed and its application to the news corpus was demonstrated (Cheng and Lapata, 2016; Nallapati et al., 2017).

The neural network-based approaches rely heavily on large amounts of reference summaries for training neural models, and consequently, for tuning a large number of parameters. The reference summaries are manually or semi-automatically created in advance. Some existing studies employ parallel corpora as artificial reference summaries (Woodsend and Lapata, 2010; Cheng and Lapata, 2016). However, preparing such large volumes of reference summaries manually is sometimes costly. Particularly, it is infeasible for humans to create hundreds of thousands of reference summaries in cases where summarization requires domain-specific or expert knowledge. Such cases include financial reports, financial and economic news (Filippova et al., 2009), and scientific articles (Parveen et al., 2016).

A fundamental requirement in extractive summarization is the identification of salient sentences from a document, i.e., sentences that represent key subjects mentioned in the document. Such subjects are often described in the form of topics, categories, sentiments, and other meta-information about a document. Sometimes they are extracted from external information related to document contents. Once one knows the subjects of a document beforehand, a straightforward strategy in extractive summarization is to select sentences that are relevant to the subjects. Importantly, subjects should be inferred from sentences identified from

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2101–2110
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

the document. For example, assume that we are about to summarize a financial report of a company with knowledge from external information sources that the company has strong earnings. In this case, we might select sentences that explain factors affecting increase of earnings so that a reader of the summary can intuitively understand the company's financial situation.

The key idea is that we regard the subjects of a document as pseudo-rough reference summaries. Then, if we are able to estimate the subjects with small amounts of documents and the external information in them, the identification of salient sentences from a document can be supported by sentence features that have been learned from document subject estimation. As a result, smaller amounts of actual reference summaries are only needed as mutually learning feature representations for both subject estimation and sentence identification from pseudo-rough reference summaries.

As described earlier, we focus on single document summarization with small amounts of reference summaries, and propose a general framework for summarization that is useful for extracting sentences from a document along with its external related information. Particularly, we formalize estimation of the above-described document subjects as a document classification task and solve document summarization in the framework of multi-task learning for sentence extraction and document classification.

Our proposed summarization framework comprises two components: one designed for sentence extraction, which selects sentences relevant to the subjects of an input document, and one for document classification, which predicts the subject of the input document. In the multi-task learning framework, document classification supports sentence extraction by learning common feature representations of salient sentences for summarization. We use recurrent neural network encoder–decoder as sentence extractor and document classifier.

We evaluate our proposed summarization method on two datasets: the NIKKEI, the leading financial news publisher in Japan and a financial report corpus; and the New York Times Annotated Corpus (Sandhaus, 2008). The results of experimental evaluations demonstrate that our summarizers achieve a performance that is comparable to those of state-of-the-art systems.

The contribution of this paper is two-fold. First, we propose a general framework for single document summarization with small amounts of reference summaries, which is important for practical implementation of summarization techniques. Second, we propose a multi-task learning method with curriculum learning that supports sentence extraction from a document while solving document classification. Here, we assume that a document is classifiable into certain subjects, which comprise the meta-information of the document. Furthermore, sentences for a summary are extracted in relation to the subjects.

2 Problem Statement and Data Preparation

In this section, we define the task of sentence extraction for document summarization as addressed in this paper. We specifically examine documents that satisfy the following requirements. (1) Reference summaries are few. (2) The document is associated with a list of subjects, $\{a_1, a_2, \cdots, a_m\}$ ($a_i \in \{0, 1\}$), that includes topics, categories, sentiments, and other meta-information. $a_j = 1$ denotes that the document is classified into the subject j. Given a document D consisting of a sequence of sentences $\{s_1, s_2, \cdots, s_n\}$, we aim to extract k sentences in relation to a document subject a_j, which is expected to be included in the summary ($k < n$) of the document. We predict both a subject a_j for the document and a label $y_i \in \{0, 1\}$ for each sentence within the document, which indicates that the i-th sentence should be extracted.

In this study, we use two datasets for our sentence extraction task: the NIKKEI financial report corpus and New York Times news corpus.

For the financial report corpus, we used financial reports published every quarter during 2013–2016 by Japanese exchange listed companies. The reports explained the economic activity and the factors affecting revenue or profits for the quarter. For the reference summary, which is the gold standard summary used for training a classifier that predicts a sentence label, we use financial news articles published by the NIKKEI [1]. The NIKKEI publishes articles summarizing financial reports of each company. It covers approximately 10 % of all the reports: 3911 reports from 2013 to 2016. The

[1] http://www.nikkei.com/

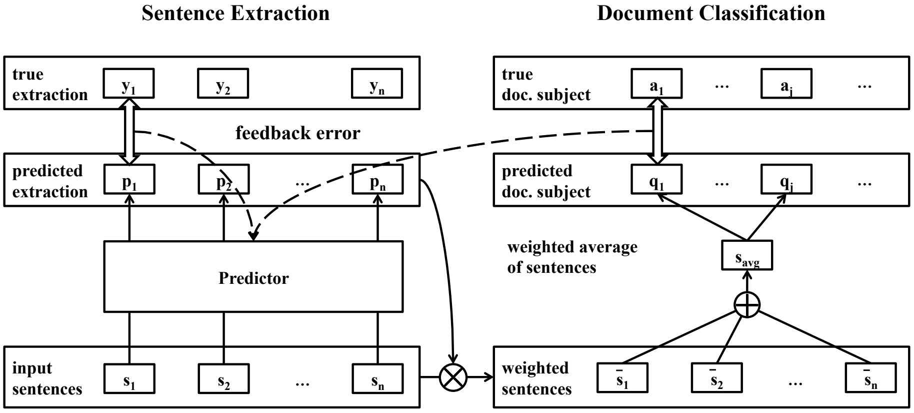


Figure 1: Proposed multi-task learning framework for sentence extraction with document classification

language of the reports and the articles is Japanese. For the document subject of a financial report, we used profit and revenue information as a subject a_j $(j = 1, 2)$, which indicates its profit and revenue increase compared to an earlier term.

Our second dataset, the New York Times Annotated Corpus (NYTAC), is a collection of articles from the New York Times. The gold standard summaries are attached to some of the articles. As the subject of a document article, we use already annotated category of the news from its metadata such as *Business* and *Arts*.

For the task of sentence extraction, the gold standard labels indicating sentences that should be extracted are needed. To attach the labels on sentences that maximize the Rouge score with respect to gold summaries, we introduce a greedy approach (Cheng and Lapata, 2016; Nallapati et al., 2017). We first select one sentence that has a maximum Rouge score with respect to the entire gold standard summary. We add it to the reference summary set and select sentences incrementally until no candidate sentence improves the score when added to the current summary set. The labels of sentences in the summary set are set as $y = 1$. The greedy approach is efficient because the computational costs associated with the identification of a global optimal summary set are too large.

The labels are attached by computing ROUGE-1 (Lin, 2004). ROUGE-1 and ROUGE-2 are reported as best for emulating evaluation by humans (Owczarzak et al., 2012). For financial reports, words apart from nouns, verbs, adjectives, and adverbs are removed for computing appropriate ROUGE scores. The accuracy between the labels attached by ROUGE-1 and humans is 81%.

3 Summarization Model

This section introduces our novel summarization framework. Figure 1 presents the proposed multi-task learning framework for sentence extraction with document classification. The left half of the figure shows the common sentence extraction part. The right half depicts a novel sentence extraction by document classification. We assume that a document is classifiable into certain subjects that represent meta-information of the document, and assume that sentences for a summary are extracted in relation to the subjects. Therefore, solving document classification supports sentence extraction from a document with multi-task learning of both tasks.

In Fig. 1, s_i denotes the embedding of sentence i. Furthermore, y_i denotes whether the sentence should be extracted and a_j is a subject of a document, which includes topics, categories, sentiments, or other meta-information. The predictor component computes $p_i \equiv p(y_i = 1 \mid D)$, the probability of sentence i extraction. Our proposed method estimates p_i by learning both sentence extraction and document classification.

We now explain how learning document classification supports sentence extraction. In Fig. 1, s_{avg} is the weighted average of s_i in terms of the probability of sentence extraction. It means that s_{avg} includes much more information about sentences with higher extraction probability. The probability that the document is related to a subject $q_j \equiv p(a_j = 1 \mid D)$ is estimated by s_{avg}. The error is larger if the contents of extracted sentences do not correspond with the document subject. By feeding back this error to the predictor, the model learns to extract sentences related to

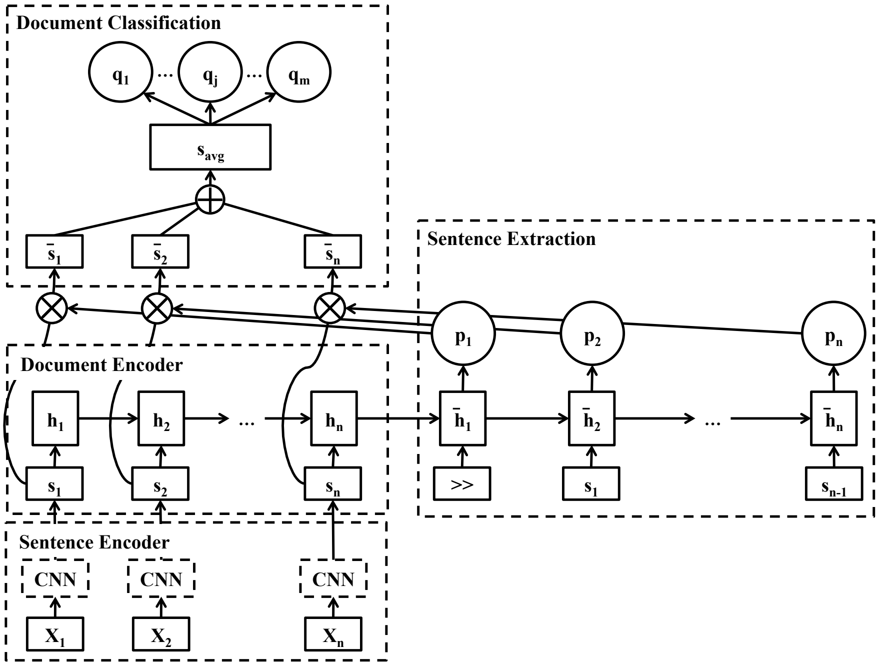


Figure 2: Sentence extraction model using LSTM-RNN with multi-task learning

the document subject. For example, in the case of a financial report, profit information indicating profit and revenue increase compared with an earlier term is used as a document subject. Positive sentences are expected to be extracted so that the extracted sentences reflect good financial results if the profit and revenue increase.

Figure. 2 shows the entire model. With respect to the predictor component in the proposed model, we use an encoder–decoder architecture modeled by recurrent neural networks (Kim et al., 2016) based on recent neural extractive summarization approaches (Cheng and Lapata, 2016; Nallapati et al., 2017). However, our summarization framework is applicable to all models of sentence extraction using distributed representation as inputs. We explain four sub-modules of the summarization model: sentence encoder, document encoder, sentence extraction, and document classification.

3.1 Sentence Encoder

We use Convolutional Neural Network (CNN) to obtain a sentence embedding from word embeddings. The training speed of single-layer CNN is high. It is effective for sentence-level classification such as sentiment analysis (Kim, 2014). Actually, CNN is suitable for use because our model requires a high computational cost. Sentence embeddings are used for both sentence extraction and document classification.

Let $x_i \in \mathcal{R}^d$ denote the embedding of the i-th word in the sentence, and $x_{i:i+q-1} \in \mathcal{R}^{dq}$ represent a concatenated vector that represents a sequence of q words. Convolutional filter $w \in \mathcal{R}^{dq}$ is applied as

$$s_w^i = f(w \cdot x_{i:i+q} + b), \qquad (1)$$

where f is a nonlinear function such as the hyperbolic tangent and b is the bias. Max pooling over time is applied to obtain a single feature s_w representing the sentence under filter w.

3.2 Document Encoder and Extractor

The LSTM-RNN Encoder–Decoder model is used for sentence extraction. First, on the encoder part, all sentences of a document are input into the hidden layers of RNN. LSTM assigns the input gate, forget gate, output gate, and memory cells as activation functions of RNN.

In Fig. 2, $h_n \in \mathcal{R}^k$ is the output of the encoder part, for which information of all sentences is input. The extraction probability is estimated based on the encoder part output. The hidden layer of the decoder part $\bar{h}_t \in \mathcal{R}^k$ is updated by LSTM equal to the encoder part. The initial value $\bar{h}_0$ is h_n. The input is the prior sentence s_{t-1} multiplied by the extraction probability p_{t-1}. Therefore, more information about sentences that are likely to be extracted is input to the hidden layer. Based on each hidden layer, the extraction probability of sentence t is computed as shown below.

$$h_t = \mathrm{LSTM}(s_t, h_{t-1}) \tag{2}$$

$$\bar{h}_t = \mathrm{LSTM}(p_{t-1} \cdot s_{t-1}, \bar{h}_{t-1}) \tag{3}$$

$$p_t = \sigma(w_y \cdot [h_t : \bar{h}_t] + b_y) \tag{4}$$

Here, $w_y \in \mathcal{R}^{2k}$ represents the weight vector, b_y stands for the bias, and : is the concatenation operator of vectors. By concatenating the hidden layers of the decoder part with the encoder part, the extraction probability is computed by referencing the information of input sentences more directly. It seems reasonable to pay attention to the input sentence directly when deciding whether the sentence should be extracted or not.

3.3 Document Classification

Using the embeddings and estimated extraction probability of sentences, the probability distribution of input document subjects is estimated. The probability that a document is classified into the subject j, q_j, is computed as shown below.

$$q_j = \sigma(w_a \cdot s_{avg} + b_a) \tag{5}$$

$$s_{avg} = \frac{\sum_t p_t \cdot s_t}{\sum_i p_i} \tag{6}$$

Here, w_a signifies the weight vector and b_a is the bias. Additionally, s_{avg} represents the weighted average of sentence embeddings. Each sentence is weighted by the estimated extraction probability. The predictor computes the probability distribution of a document subject from s_{avg}, which means that the predictor pays more attention to sentences that are likely to be extracted.

3.4 Multi-Task Learning with Curriculum Learning

This section presents an explanation of the procedure followed to train the summarization model. The model parameters are updated to maximize the likelihood of all sentence labels and document subject labels. This is equivalent to minimization of the following error terms.

$$\begin{aligned} E_y(\theta) = &-\sum_{t=1}^{n} (y_t \log p_t + (1-y_t)\log(1-p_t)) \\ &+ \lambda_\theta \|\theta\|^2 \end{aligned} \tag{7}$$

$$\begin{aligned} E_a(\theta) = &-\sum_{j=1}^{m} (a_j \log q_j + (1-a_j)\log(1-q_j)) \\ &+ \lambda_\theta \|\theta\|^2 \end{aligned} \tag{8}$$

In these equations, $\|\theta\|^2$ is the L2 norm, and λ_θ is the regularization term. L2 regularization is introduced to avoid overfitting.

Multi-task learning is generally complicated because the parameter is optimized simultaneously for sentence extraction and document classification. We introduce curriculum learning (Bengio et al., 2009) to overcome this difficulty. Curriculum learning is a learning method that aims to improve the performance of a complicated model or data. The model starts by learning a simple model or data, and gradually adapts to more complicated ones.

We introduce two kinds of curriculum learning for multi-task learning. We apply baby step curriculum learning (Cirik et al., 2016), which demonstrates the effectiveness of the LSTM-RNN architecture. In this, the dataset is categorized based on the difficulty and added to the order of ease.

We divide the dataset into three subsets based on the combination of document type and objective function. The first subset has documents with an attached reference summary. The model is trained for optimizing sentence extraction. The second uses the same documents as the first. However, the objective function is document classification. The last one has documents with no reference summary. Only the likelihood of document subjects as pseudo-rough reference summaries is maximized in the last dataset. For sentence extraction task, it is more difficult to train from the last dataset than the first dataset because information related to document subjects are more truncated than the reference summary. The second dataset is the bridge between the first and the last.

For document classification, sentences are weight-averaged by the estimated sentence extraction probability p_t. In the second dataset, sentences are weighted not only by p_t, but also the true label y_t. p_t in Eq.(6) is replaced by $\bar{p}_t$ as follows.

$$\bar{p}_t = \kappa p_t + (1-\kappa) y_t \tag{9}$$

Here, κ is the mixing rate of extraction probability and the true label. At the beginning of training,

p_t is not predicted accurately. This will affect document classification adversely, therefore κ is set to nearly zero so that the true label y_t is used for document classification. By using the true label, training for sentence extraction and document classification does not mutually interfere. As training progress, κ gets larger and document classification supports sentence extraction.

We believe that our basic idea of curriculum learning, with some modifications depending on the task applied, can be applied for other kinds of multi-task learning in general.

4 Experimental Setup

4.1 Dataset

The NIKKEI Financial Report Corpus

For training and evaluation, we used financial reports published from April 2013 through September 2016. The reports used for evaluation and validation were published in the last and the second to last quarter. All other reports were used for training. The numbers of reports used for training, validation, and evaluation were $12,262$, 191, and 183. In the training dataset, $8,725$ reports with no reference summary were included and used only for training of document classification, which predicts document subjects. As for document subjects, we used subjects of two kinds as $a_j \in \{0,1\}$ $(j = 1,2)$, indicating that the profit and revenue increases compared with the prior term. $a_j = 1$ denotes that the value increases.

New York Times Annotated Corpus

For the experiment using NYTAC, we evaluated our model using different amounts of reference summary. The numbers of articles used for both validation and evaluation were 200. For training, we prepared 125, 250, and 500 articles. For training of document classification, we used 3000 articles for which the reference summary was not attached. As document subjects, we used the category of a news article $a_j \in \{0,1\}$ $(j = 1,2,\cdots,C)$ as a subject. Each subject corresponded to a news article category, such as ”Business” and ”Arts.” $a_j = 1$ denotes that a document is classified into the category j. C is the number of categories, which is 26 in our experiment.

4.2 Implementation Details

The word embeddings were pre-trained using Skip-gram (Mikolov et al., 2013) on all $1,043,064$ articles in the Japanese version of Wikipedia. The dimensions of word embedding were 200. Those of the hidden layer in LSTM-RNN were 400. For CNN, the list of kernel sizes was $\{1,2,3,4,5,6\}$. The number of feature maps was 50. Adadelta (Zeiler, 2012) was used for updating parameters. The initial learning rate was 10^{-6}. The hyper parameters were optimized using grid search. We extracted three sentences with the highest scores in the manner described in an earlier report (Cheng and Lapata, 2016).

4.3 Baselines

For the NIKKEI financial report dataset, we used LEAD, which extracts the leading three sentences of a document as a baseline. We also built a baseline classifier LREG using logistic regression and human engineered features. The features were sentence length, position in the document, number of entities, nouns, verbs, adverbs, and adjectives in the sentence. We also added the sentiment of sentence to the features. For the financial report summarization, sentiment information is important because positive/negative sentences are frequently included in the summary when the revenue increases/decreases. The sentiment is computed by the frequency of words that appear in the articles when the revenue increases/decreases. For both datasets, we assigned NN-SE(Cheng and Lapata, 2016) as the baseline. The difference between NN-SE and our model is the introduction of multi-task learning and curriculum learning. The hyper-parameters are the same as those of our model. Through comparison with NN-SE, we can validate the effectiveness of the proposed framework.

5 Results

5.1 Results obtained from the NIKKEI Financial Report Corpus

Table1 presents the results for financial reports using F-measure. The precision and recall are calculated based on binary classification setup. LEAD, LREG, and NN-SE are used as the baselines. The proposed neural multi-task learning model, NN-ML, is significantly inferior to NN-SE and LREG. However, NN-ML-CL, the proposed model with curriculum learning, is superior to all

Table 1: F-measure evaluation (%) on financial reports

Models	F-measure	Precision	Recall
LEAD	42.1	39.1	50.4
LREG	60.5	**67.6**	66.5
NN-SE	59.9	58.0	68.8
NN-ML	55.2	52.1	64.6
NN-ML-CL	**60.6**	54.6	**74.9**

Table 2: F-measure evaluation (%) depending on the amount of reference summary

Models	125	250	500	1000	2000
NN-SE	55.2	56.1	58.0	58.0	58.1
NN-ML-CL	**58.1**	**58.4**	**59.4**	**59.3**	**59.2**

other models. This result shows that merely introducing multi-task learning does not positively influence on sentence extraction. However, curriculum learning overcomes the difficulty of multi-task learning; thus, document classification has positive effects on sentence extraction.

We confirmed the relation between the effectiveness of our model and the amount of reference summaries. We compared NN-ML-CL (our model) and NN-SE in several cases for which the amounts of reference summaries were 125, 250, 500, 1000, and 2000; the results are shown in Table 2. As observed, NN-ML-CL is superior to NN-SE in all cases. The margin between NN-ML-CL and NN-SE grows as the amount decreases, which means that document classification is more effective in cases with fewer reference summaries.

We also reported the results of human evaluation for summaries generated by the respective systems. Referring to the gold summary, participants ranked the generated summaries generated by four systems: NN-ML-CL(our system), NN-SE, LEAD, and LREG. The judging criteria was informativeness, which indicates how a generated summary covers information in the gold summary. From the test documents, we remove summaries for which the same sentences were extracted by different systems and randomly sampled 20 documents. 6 persons participated in the evaluation.

Table 3 presents the distribution of ranking and the average. Our NN-ML-CL model is ranked first in more than half the tests and markedly surpasses other models. Comparison with NN-SE verifies the effectiveness of multi-task learning for human evaluation.

Table 3: Ranking distributions (%) and the average evaluated by humans

Models	1st	2nd	3rd	4th	Ave.
LEAD	21.7	20.0	28.3	30.0	2.67
LREG	20.0	28.3	26.7	25.0	2.45
NN-SE	31.7	21.7	16.7	30.0	2.57
NN-ML-CL	51.7	20.0	21.7	6.7	**1.83**

Table 4: ROUGE scores (%) for various amounts of reference summaries in NYTAC

Model	Ref.	ROUGE-1	ROUGE-2
NN-SE	125	17.2	12.1
	250	16.8	11.3
	500	18.0	12.5
NN-ML-CL	125	18.1	12.7
	250	18.3	12.7
	500	18.5	12.9

5.2 Results on NYTAC

Table 4 presents our results for NYTAC using ROUGE-1 and ROUGE-2. In all cases, NN-ML-CL outperforms NN-SE on both metrics. When the amount of reference summary is 250, the margin between NN-ML-CL and NN-SE is the largest on each metric. For cases with 125 and 500 reference summaries, improvement is observed, but the margin is smaller than in the case for financial reports.

5.3 Discussion

In this section, we discuss how document classification contributes to the improvement of sentence extraction performance on the financial report dataset.

As mentioned above, NN-ML, the model that uses multi-task learning, exhibits a performance that is worse than that of NN-SE, the model that does not use multi-task learning. One possible explanation would be that it is difficult to optimize the parameters to maximize the likelihood of both sentence extraction and document classification simultaneously. If the learning task of sentence extraction does not proceed well enough, the task of document classification may also not work well. The reason is that the classification task relies on the estimated probability for sentence extraction in our proposed summarization framework.

However, curriculum learning improves the performance of the model with multi-task learning. By introducing curriculum learning into the framework, we are able to start training the model

Table 5: Rates (%) of extracted sentences corresponding with a document subject

	Correspond	Not	Others
NN-SE	**40.0**	40.0	20.0
NN-ML-CL	**85.7**	14.3	0.0

Table 6: Example of gold summary and sentences extracted using NN-ML-CL and NN-SE

Sentences extracted by NN-ML-CL
The rapid progress of the strong yen adversely influenced financial results, but the growth of revenue in areas such as Europe, Asia, and Oceania increased the revenue to 535 billion yen. The demand for air conditioners increased because of the intense July heat.
Sentences extracted by NN-SE
As for fluorine resin, the demand for semiconductors rose steadily. However, competing Chinese companies gained power, and revenue for electrical wire use declined. The revenue of parts for guided missiles increase year-on-year, but the revenue of medical equipment decrease.
News Article (Gold summary)
In Southeast Asia and Europe, high-end models of air conditioners sold well. In China and US, revenue rose and overcame the adverse influence of strong yen. The corporate tax ratio reduction also supported business performance. The revenue of air-conditioners, a leading product of Daikin, rose 9% in Southeast Asia. The revenue network in Vietnam and Indonesia expanded. revenue of air-conditioners rose at a higher pace than market scale. In China, the revenue of air conditioners for business use recovered. High-end models were also selling well.

only for sentence extraction. Then, the training for document classification is started gradually. Eventually, it contributes to the improvement of the performance of sentence extraction through the multi-task learning approach.

From the results for the financial report corpus, we confirmed that the contents of sentences extracted by our model corresponded with revenue and profit changes. Before validation, the sentences were categorized as corresponding to or not corresponding to others. We compared the results of sentence extraction with NN-ML-CL and NN-SE and checked the category distribution of sentences extracted using NN-ML-CL or NN-SE.

Table 5 shows that 85.7% of sentences extracted using NN-ML-CL correspond to changes of revenues and profits. However, only 40.0% of sentences extracted by NN-SE correspond to these parameters, which indicates that document classification supports extraction of sentences related to the revenue and profit change, and contributes to the improvement.

Table 6 shows sentences extracted from financial reports published by Daikin, Ltd., the leading air-conditioner manufacturer in Japan. During this term, the air conditioner revenue increased; moreover, revenues and profits increased considerably year-on-year. NN-ML-CL extracted sentences that mention the good revenue performance of air-conditioners in Asia and Europe, which is the same as that in the gold summary. In contrast, NN-SE extracts sentences mentioning the bad revenue performance of fluorine resin and medical equipment, which are not described in the gold summary. NN-SE is badly affected owing to training on past reports and articles. Our model extracts sentences with words that appear frequently in a positive context. Therefore, sentences related to good revenue performance are extracted.

There are two main ways of applications for our summarization approach with document classification. In the first case, the text collection has explicitly annotated document labels, which includes the collection of news articles with their category information, product reviews with their rating, scholarly paper abstracts with their discipline information, etc. In the second case, a document label can be acquired from external information sources about the text collection. For financial reports, the information about financial situation of a target company is extracted from the financial statement, which in turn can be used for a label of document classification.

6 Related Work

Based on the recent advances of neural network-based approaches (Kågebäck et al., 2014; Cao et al., 2015; Yin and Pei, 2015; Cao et al., 2016), an attentional encoder-decoder for extractive single-document summarization and its application to the news corpus was proposed (Cheng and Lapata, 2016; Nallapati et al., 2017). Although we employ an encoder-decoder architecture in the predictor component of our summarization framework, the framework can be applied to all models of sentence extraction using distributed representation as inputs, including recently advanced other attention-based encoder-decoder networks (Wang et al., 2016; Yang et al., 2016)

(Cheng and Lapata, 2016; Nallapati et al., 2017) argue that a stumbling block to applying neural network models to extractive summarization is the lack of training data and documents with sentences labeled as summary-worthy. To overcome this, several studies have used artificial reference summaries (Sun et al., 2005; Svore et al., 2007; Woodsend and Lapata, 2010; Cheng and Lapata, 2016) compiled by collecting documents and corresponding highlights from other sources. However, preparing such a parallel corpus often requires domain-specific or expert knowledge depending on the domain (Filippova et al., 2009; Parveen et al., 2016). Our summarization uses document-associated information as pseudo rough reference summaries, which enables us to learn feature representations for both document classification and sentence identification with smaller amounts of actual reference summaries.

Neural networks based multi-task learning has recently proven effective in many NLP problems (Liu et al., 2015, 2016; Firat et al., 2016; Dong et al., 2015). Aiming at single document summarization with relatively small amounts of reference summaries, we demonstrated document summarization in the framework of multi-task learning with curriculum learning for sentence extraction and document classification. This enabled us to obtain better feature representations to extract sentences from documents.

7 Conclusion

In this paper, we proposed a general framework for extractive summarization using document subjects. Our key idea is to use a multi-task learning method that supports sentence extraction while enabling document classification, assuming that a document can be classified into certain subjects, and sentences for a summary are extracted in relation to the subjects.

This framework enables single document summarization with relatively small amounts of reference summaries since document subjects can be used as pseudo-rough reference summaries. Our proposed method can be widely applied for actual documents attached with meta-information such as product reviews, sports news and so on.

Experimental results showed that our model is less effective on the news corpus. For higher performance, more information such as the embeddings of news descriptors for document classification must be used.

Acknowledgments

This work was supported by JST CREST Grant Number JPMJCR1513, Japan and the New Energy and Industrial Technology Development Organization (NEDO).

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*, pages 41–48.

Ziqiang Cao, Wenjie Li, Sujian Li, Furu Wei, and Yanran Li. 2016. Attsum: Joint learning of focusing and summarization with neural attention. In *COLING*, pages 547–556.

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015. Ranking with recursive neural networks and its application to multi-document summarization. In *AAAI*, pages 2153–2159.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *ACL*, pages 484–494.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL-HLT*, pages 93–98.

Volkan Cirik, Eduard Hovy, and Louis-Philippe Morency. 2016. Visualizing and understanding curriculum learning for long short-term memory networks. *arXiv preprint arXiv:1611.06204*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *ACL*, pages 1723–1732.

Günes Erkan and Dragomir R Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. In *EMNLP*, volume 4, pages 365–371.

Katja Filippova, Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2009. Company-oriented extractive summarization of financial news. In *EACL*, pages 246–254.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. *arXiv preprint arXiv:1601.01073*.

Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. 2014. Extractive summarization using continuous vector space models. In *the second Workshop on Continuous Vector Space Models and their Compositionality in EACL*, pages 31–39.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Wei Li, Lei He, and Hai Zhuge. 2016. Abstractive news summarization based on event semantic link network. In *COLING*, pages 236–246.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out Workshop in ACL*, volume 8.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *IJICAI*, pages 2873–2879.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *NAACL-HLT*, pages 912–921.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *European Conference on Information Retrieval*, pages 557–564.

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

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081.

Karolina Owczarzak, John M Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Workshop on Evaluation Metrics and System Comparison for Automatic Summarization in ACL*, pages 1–9.

Daraksha Parveen, Mohsen Mesgar, and Michael Strube. 2016. Generating coherent summaries of scientific articles using coherence patterns. In *EMNLP*, pages 772–783.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium*, 6(12).

Jian-Tao Sun, Dou Shen, Hua-Jun Zeng, Qiang Yang, Yuchang Lu, and Zheng Chen. 2005. Web-page summarization using clickthrough data. In *ACM SIGIR conference on Research and development in information retrieval*, pages 194–201.

Krysta Marie Svore, Lucy Vanderwende, and Christopher JC Burges. 2007. Enhancing single-document summarization by combining ranknet and third-party sources. In *EMNLP-CoNLL*, pages 448–457.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *EMNLP*, pages 1054–1059.

Y Wang et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *EMNLP*, pages 606–615.

Kam-Fai Wong, Mingli Wu, and Wenjie Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *COLING*, pages 985–992.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *ACL*, pages 565–574.

Z Yang et al. 2016. hierarchical attention networks for document classification. In *NAACL-HLT*, pages 1480–1489.

Wenpeng Yin and Yulong Pei. 2015. Optimizing sentence modeling and selection for document summarization. In *IJCAI*, pages 1383–1389.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Towards Automatic Construction of News Overview Articles by News Synthesis

Jianmin Zhang and **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{zhangjianmin2015, wanxiaojun}@pku.edu.cn

Abstract

In this paper we investigate a new task of automatically constructing an overview article from a given set of news articles about a news event. We propose a news synthesis approach to address this task based on passage segmentation, ranking, selection and merging. Our proposed approach is compared with several typical multi-document summarization methods on the Wikinews dataset, and achieves the best performance on both automatic evaluation and manual evaluation.

1 Introduction

There are usually many news articles about a news event, and news summaries can be used for readers to quickly learn the most salient information of the news articles. News summaries in previous studies are usually very short, and most of them consist of about one or two hundred words. However, in many circumstances, readers want to learn more about an event, but the news summary is insufficient to read, and people are reluctant to read each news article one by one. A possible solution to this problem is constructing a long and comprehensive news overview article to summarize and present all important facts about the news event in an unbiased way. The news overview articles can be considered long summaries, however, news overview articles are more comprehensive and the article texts are harder to arrange and organize.

In this paper, we conduct a pilot study to investigate the new task of automatic construction of a news overview article from a set of news articles about an event. We argue that traditional multi-document summarization methods can be applied to this task, but they do not perform well because sentence-based extraction used in these methods is not suitable for constructing and organizing a long article. Instead, we propose a news synthesis approach to address this task. Our approach uses passage as the basic unit. In this study, passage does not mean a natural paragraph, but means a block of text (maybe multiple paragraphs) about a subtopic of an event. Our approach first segments news articles into passages with the SenTiling algorithm, and then ranks the passages with the DivRank algorithm. Finally, it selects and merges a few passages to construct the long news overview article.

We automatically build an evaluation dataset based on English Wikinews [1]. Most Wikinews articles are synthesis articles and they are written using information from other online news sources. All the important facts available from all sources about a news event are combined into a single article for the reader's convenience, and the information is presented in a neutral manner avoiding the bias that may be present in other news sources. Therefore, we treat a Wikinews article as an ideal overview article (i.e., reference) of the source news articles.

We compare our proposed approach with several typical multi-document summarization methods based on the Wikinews dataset. The results are very promising and our approach achieves the best performance on both automatic evaluation and manual evaluation. In this study, we demonstrate the feasibility of automatic construction of long overview articles from a set of news articles.

The contributions of this paper are summarized as follows: 1) we are the first to investigate the task of automatic construction of news overview articles from a set of source news articles; 2) we automatically build an evaluation dataset based on Wikinews; 3) we propose a news passage-based

[1] https://en.wikinews.org/wiki/Main_Page

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2111–2116
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

synthesis approach to address this task; 4) evaluation results verify the efficacy of our approach.

2 Our News Synthesis Approach

We propose a news synthesis approach to automatic construction of news overview articles from a set of source news articles. Our approach uses passage as the basic unit, and consists of three main steps: passage segmentation, passage ranking, and passage selection and merging. The rationale of using passage rather than sentence lies in that 1) the sentences in a passage are more complete and coherent than multiple sentences selected from different places in different documents; 2) it is easier to arrange several passages than to arrange a large number of sentences.

2.1 Passage Segmentation

In this step, we aim to segment each source news article into several passages, where each passage represents a subtopic of the event. In order to achieve this goal, we adopt the TextTiling algorithm (Hearst, 1997), which is a popular algorithm for discovering subtopic structure using term repetition. The original TextTiling algorithm usually splits a sentence into different passages, and in order to remedy this problem, we slightly modify the TextTiling algorithm and our new SenTiling algorithm consists of three steps:

Tokenization refers to the division of the input text into individual lexical units, and the tokens are converted to lower-case characters and stemmed using the Porter stemmer.

Lexical score determination refers to assigning a lexical score of each gap between text blocks. To avoid the incomplete sentence in the segmentation result, we regard a sentence as a text block and calculate a lexical score for the gap at the end of each sentence by the cosine similarity value between 100 words before and after the gap. We do not use natural paragraphs as blocks because their lengths are highly irregular.

Boundary identification assigns a depth score to each sentence gap and then determines the passages to assign to a document. The depth score is computed in the same way as in (Hearst, 1997) and it corresponds to how strongly the cues for a subtopic changed on both sides of a given gap and is based on the distance from the peaks on both sides of the valley to that valley. Since every gap is a potential segment boundary. We select a boundary only if the depth score exceeds the average depth scores $\overline{s}$ minus the standard deviation σ of their scores (thus assuming that the scores are normally distributed), as $\overline{s} - \sigma$.

2.2 Passage Ranking

We use DivRank (Mei et al., 2010) to rank passages, because DivRank automatically balances the prestige and the diversity of the top ranked passages in a principled way. It is motivated from a general time-variant random walk process known as the vertex-reinforced random walk. Let $p_T(v)$ be the probability that the walk is at state v at time T, and $p_T(u, v)$ be the transition probability from any state u to any state v at time T.

$$p_T(v) = \textstyle\sum_{u \in V} p_{T-1}(u, v) p_{T-1}(u)$$

$$p_T(u, v) = (1 - \lambda) \cdot p^*(v) + \lambda \cdot \frac{p_0(u,v) \cdot p_T(v)}{D_T(u)}$$

where $D_T(u) = \sum_{v \in V} p_0(u, v) p_T(v)$. And $p^*(v)$ is a uniform distribution which represents the prior preference of visiting vertex v. $p_0(u, v)$ is the organic transition probability prior to any reinforcement, which is estimated as in a regular time-homogenous random walk by the normalized cosine similarity value between u and v.

After a sufficiently large T, the reinforced random walk will converge to a stationary distribution, and each passage node will be assigned with a rank score.

2.3 Passage Selection and Merging

We aim to select several important but non-redundant passages to form the overview article. The selection can be done according to the DivRank scores because the scores balance the prestige and the diversity of most of the top ranked passages, but it occasionally happens that two relevant passages both get high scores. In order to remedy this problem and make the content for each subtopic more comprehensive and complete, we further merge relevant passages by adding informative sentences from relevant passages into the selected passage. The greedy selection process is illustrated in Algorithm 1.

The function $merge(g_{i*}, g_{j*})$ merges the sentences of g_{i*} into g_{j*} one by one. If the average similarity between a sentence $s_{i*,k}$ in g_{i*} and each sentence in g_{j*} is less than ξ, we insert the sentence $s_{i*,k}$ into g_{j*} and find the insertion position between two sentences $s_{j*,m}$ and $s_{j*,n}$ in g_{j*}, where the average of the similarity between $s_{i*,k}$ and $s_{j*,m}$, and the similarity between $s_{i*,k}$

Algorithm 1 Passage Selection and Merging

```
Input:
    Passage set $G = g_1, ..., g_n$ and each passage $g_i$ is assigned with a DivRank score $p(g_i)$;
    The cosine similarity value $gSim_{i,j}$ between any two passages $g_i$ and $g_j$;
Output:
    The passage set $O$ in the overview article;
Initialize $O = \phi$
while $G \neq \phi$ and $O$ does not reach the length limit do
    $g_{i*} = \text{argmax}_{g_i \in G}\, p(g_i)$;
    $G = G - g_{i*}$
    $g_{j*} = \text{argmax}_{g_j \in O}\, gSim_{i*,j}$;
    if $gSim_{i*,j*} > \tau$ then
        $g_{j*} = merge(g_{i*}, g_{j*})$
    else
        $O = O \cup g_{i*}$
    end if
end while
return $O$
```

and $s_{j*,n}$ is the largest.

Finally, we arrange the passages in O with topological sorting to form the overview article. We follow two principles: 1) If passages u and v are from the same news article and u is before v, they should be adjacent and have the same order in the overview article; 2) If passages u and v are from different news articles and u has higher DivRank score than v, u and the passages coming from the same news article with u should be placed before v in the overview article.

3 Evaluation Dataset and Baselines

As mentioned in the introduction section, we used Wikinews to construct the evaluation dataset. We first crawled 18121 English Wikinews and their source news articles via the associated URLs. However, many Wikinews articles have very few source news articles and they are very short, and moreover, the URLs for many of the source news are out of date. We filtered the Wikinews articles for which the number of available source news articles are less than 5. Finally, we selected 100 longest Wikinews from the remaining set for testing [2]. The average number of words of Wikinews in the test set is 598 and the average number of total words of their source news articles is 2136. Accordingly, the length limit of overview articles produced by different methods is 600 words.

Our approach is compared with several typical multi-document summarization methods: **Lead**, **Coverage**, **Centroid** (Radev et al., 2004), **TextRank** (Mihalcea and Tarau, 2004), **ClusterCMRW** (Wan and Yang, 2008), **ILP** (Gillick and Favre, 2009) and **Submodular** (Li et al., 2012). We also implement **SenDivRank** that applies the DivRank algorithm on sentences.

For our approach, τ is set to 0.4 and ξ is set to 0.5 based on an additional small development set chosen from the remaining Wikinews set. λ in the DivRank algorithm is set to 0.85 by default. Under the control of these thresholds, we only merge a very small number of passages and insert very few sentences from one passage to another passage, so the influence of passage merging on the coherence is very subtle.

4 Evaluation Results and Analysis

Automatic Evaluation: Similar to traditional summarization tasks, we use the ROUGE metrics (Lin and Hovy, 2003) to automatically evaluate the quality of peer overview articles against the gold-standard references. We use ROUGE-1.5.5 and report the F-scores of ROUGE-1 (R-1), ROUGE-2 (R-2) and ROUGE-SU4 (R-SU4).

Firstly, we perform evaluation on the whole articles and Table 1 shows the comparison results. We can see that our approach outperforms all the baseline methods with respect to ROUGE-2 and ROUGE-SU4. The Submodular method achieves the highest ROUGE-1 score, but our approach also achieves very high ROUGE-1 score, which is very close to that of the Submodular method.

Method	**R-1**	**R-2**	**R-SU4**
Lead	0.48029	0.16183	0.21156
Coverage	0.48085	0.15849	0.20615
TextRank	0.49453	0.16370	0.21457
Centroid	0.48582	0.16099	0.20919
ILP	0.49302	0.16651	0.21493
ClusterCMRW	0.49363	0.17205	0.22033
Submodular	**0.50273**	0.16963	0.21775
SenDivRank	0.48701	0.17491	0.22382
Our Approach	0.50215	**0.18631**	**0.23426**

Table 1: Comparison results on overall evaluation

Secondly, in order to better evaluating the con-

[2] The dataset is accompanied and it will be released soon.

Method	R-1	R-2	R-SU4
Lead	0.38757	0.10631	0.15138
Coverage	0.38932	0.10399	0.14714
TextRank	0.40246	0.10651	0.15327
Centroid	0.38910	0.10297	0.14774
ILP	0.40004	0.11256	0.15641
ClusterCMRW	0.40565	0.11855	0.16195
Submodular	0.39990	0.11044	0.15442
SenDivRank	0.39462	0.11575	0.16028
Our Approach	**0.41913**	**0.13369**	**0.17735**

Table 2: Comparison results on two-part evaluation I

Method	R-1	R-2	R-SU4
Lead	0.39850	0.11888	0.16209
Coverage	0.39957	0.11610	0.15753
TextRank	0.41132	0.12045	0.16317
Centroid	0.40071	0.11772	0.15859
ILP	0.40795	0.12149	0.16350
ClusterCMRW	0.41379	0.12769	0.16935
Submodular	0.40677	0.11903	0.16163
SenDivRank	0.40210	0.12704	0.17001
Our Approach	**0.42207**	**0.14401**	**0.18392**

Table 3: Comparison results on two-part evaluation II

tent organization in long articles, we split each article (both peer article and reference article) into two parts with equal length, and compare the first parts in the peer and reference articles, and then compare the second parts in the peer and reference articles. Lastly, the ROUGE scores are averaged across the two parts. Table 2 shows the comparison results based on this evaluation protocol (two-part evaluation I). Furthermore, we allow the first part in a reference article to match with the second part in a peer article, and vice versa. We allow one-to-one matching and find the optimal matching between the two sets of parts, which refers to the matching with the largest sum of the similarity values of the matched parts. We then compute and average the ROUGE scores of the matched parts. Table 3 shows the comparison results based on this evaluation protocol (two-part evaluation II). We can see from Tables 2 and 3 that our proposed approach performs much better than the baseline methods over all three metrics.

Method	Cov.	Read.	Overall
TextRank	2.86	2.34	2.50
Centroid	2.83	2.17	2.33
ILP	2.17	1.17	2.27
ClusterCMRW	3.33	2.34	2.83
Submodular	2.51	2.03	2.34
SenDivRank	3.51	2.47	2.86
Our Approach	**3.85**	**3.32**	**3.47**

Table 4: Manual evaluation results

Manual Evaluation: We randomly select 30 test cases for manual evaluation. We employ three students as human judges and each judge is asked to read the reference Wikinews and the peer overview article produced by each method, and then give a rating score between 1 and 5 with respect to three aspects: content coverage, readability and overall responsiveness. 5 means “very good”, 3 means “acceptable”, and 1 means “very bad”. The methods producing the articles are blind to the judges. Finally, the rating scores with respect to each aspect across different test cases are averaged, and then averaged across the three judges. Table 4 shows the manual evaluation results. We can see that our proposed approach can produce news overview articles with better content coverage, readability and overall responsiveness than baseline methods. The quality of the news overview articles is generally acceptable by the human judges.

In all, our proposed approach are more effective than typical multi-document summarization methods for addressing this challenging task. It is feasible to automatically construct news overview articles with news synthesis.

5 Related Work

The most closely related work is multi-document summarization, which aims to produce a concise (or short) summary to deliver the major information for a given document set. Most summarization methods rank and select a few existing sentences in the documents or compose new sentences with phrases to form a summary. Typical summarization methods include graph-based ranking methods (Erkan and Radev, 2004; Mihalcea and Tarau, 2005; Berg-Kirkpatrick et al., 2011; Wan and Zhang, 2014; Wan and Yang, 2008), sentence classification or regression based methods (Conroy and O’leary, 2001; Shen et al., 2007; Ouyang et al., 2007), ILP-based methods (McDonald, 2007; Gillick and Favre, 2009; Xie et al., 2009; Berg-Kirkpatrick et al., 2011; Woodsend and Lapata, 2012; Bing et al., 2015), submodular maximization based methods (Lin and Bilmes, 2010, 2011; Sipos et al., 2012), DPP (Determinantal Point Process) based methods (Kulesza et al., 2012), and neural model based methods (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016), etc.

Other related work includes automatic generation of well-structured Wikipedia articles (Sauper and Barzilay, 2009; Yao et al., 2011). Different from Wikinews, Wikipedia articles usually have domain-dependent templates for content filling and organization.

6 Conclusion

In this pilot study we proposed a news synthesis approach to address the challenging task of automatic generation of news overview articles. Evaluation results on Wikinews verified the efficacy and feasibility of the proposed approach. In future work, we will investigate supervised learning methods for passage ranking and selection, and try to paraphrase the selected passages.

Acknowledgments

This work was supported by NSFC (61331011), 863 Program of China (2015AA015403) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the anonymous reviewers for helpful comments. Xiaojun Wan is the corresponding author.

References

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 481–490. Association for Computational Linguistics.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. *arXiv preprint arXiv:1506.01597*.

Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. 2016. Abstractive sentence summarization with attentive recurrent neural networks. *Proceedings of NAACL-HLT16*, pages 93–98.

John M Conroy and Dianne P O'leary. 2001. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 406–407. ACM.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 10–18. Association for Computational Linguistics.

Marti A Hearst. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64.

Alex Kulesza, Ben Taskar, et al. 2012. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286.

Jingxuan Li, Lei Li, and Tao Li. 2012. Multi-document summarization via submodularity. *Applied Intelligence*, 37(3):420–430.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *European Conference on Information Retrieval*, pages 557–564. Springer.

Qiaozhu Mei, Jian Guo, and Dragomir Radev. 2010. Divrank: the interplay of prestige and diversity in information networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1009–1018. Acm.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *EMNLP*. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2005. A language independent algorithm for single and multiple document summarization.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

You Ouyang, Sujian Li, and Wenjie Li. 2007. Developing learning strategies for topic-based summarization. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 79–86. ACM.

Dragomir R Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6):919–938.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Christina Sauper and Regina Barzilay. 2009. Automatically generating wikipedia articles: A structure-aware approach. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 208–216. Association for Computational Linguistics.

Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. 2007. Document summarization using conditional random fields. In *IJCAI*, volume 7, pages 2862–2867.

Ruben Sipos, Pannaga Shivaswamy, and Thorsten Joachims. 2012. Large-margin learning of submodular summarization models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 224–233. Association for Computational Linguistics.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306. ACM.

Xiaojun Wan and Jianmin Zhang. 2014. Ctsum: extracting more certain summaries for news articles. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 787–796. ACM.

Kristian Woodsend and Mirella Lapata. 2012. Multiple aspect summarization using integer linear programming. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 233–243. Association for Computational Linguistics.

Shasha Xie, Benoit Favre, Dilek Hakkani-Tür, and Yang Liu. 2009. Leveraging sentence weights in a concept-based optimization framework for extractive meeting summarization. In *INTERSPEECH*, pages 1503–1506.

Conglei Yao, Xu Jia, Sicong Shou, Shicong Feng, Feng Zhou, and Hongyan Liu. 2011. Autopedia: Automatic domain-independent wikipedia article generation. In *Proceedings of the 20th international conference companion on World wide web*, pages 161–162. ACM.

Joint Syntacto-Discourse Parsing and the Syntacto-Discourse Treebank [*]

Kai Zhao[†] and **Liang Huang**
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, Oregon, USA
{kzhao.hf, liang.huang.sh}@gmail.com

Abstract

Discourse parsing has long been treated as a stand-alone problem independent from constituency or dependency parsing. Most attempts at this problem are pipelined rather than end-to-end, sophisticated, and not self-contained: they assume gold-standard text segmentations (Elementary Discourse Units), and use external parsers for syntactic features. In this paper we propose the first end-to-end discourse parser that jointly parses in both syntax and discourse levels, as well as the first syntacto-discourse treebank by integrating the Penn Treebank with the RST Treebank. Built upon our recent span-based constituency parser, this joint syntacto-discourse parser requires no preprocessing whatsoever (such as segmentation or feature extraction), achieves the state-of-the-art end-to-end discourse parsing accuracy.

1 Introduction

Distinguishing the semantic relations between segments in a document can be greatly beneficial to many high-level NLP tasks, such as summarization (Louis et al., 2010; Yoshida et al., 2014), sentiment analysis (Voll and Taboada, 2007; Somasundaran et al., 2009; Bhatia et al., 2015), question answering (Ferrucci et al., 2010; Jansen et al., 2014), and textual quality evaluation (Tetreault et al., 2013; Li and Jurafsky, 2016).

There has been a variety of research on discourse parsing (Marcu, 2000a; Soricut and Marcu, 2003; Pardo and Nunes, 2008; Hernault et al., 2010; da Cunha et al., 2012; Joty et al., 2013; Joty and Moschitti, 2014; Feng and Hirst, 2014; Ji and Eisenstein, 2014; Li et al., 2014a,b; Heilman and Sagae, 2015; Wang et al., 2017). But most of them suffer from the following limitations:

1. *pipelined rather than end-to-end*: they assume pre-segmented discourse, and worse yet, use gold-standard segmentations, except Hernault et al. (2010);
2. *not self-contained*: they rely on external syntactic parsers and pretrained word vectors;
3. *complicated*: they design sophisticated features, including those from parse-trees.

We argue for the first time that discourse parsing should be viewed as an extension of, and be performed in conjunction with, constituency parsing. We propose the first *joint syntacto-discourse treebank*, by unifying constituency and discourse tree representations. Based on this, we propose the first *end-to-end* incremental parser that jointly parses at both constituency and discourse levels. Our algorithm builds up on the span-based parser (Cross and Huang, 2016); it employs the strong generalization power of bi-directional LSTMs, and parses efficiently and robustly with an extremely simple span-based feature set that does not use any tree structure information.

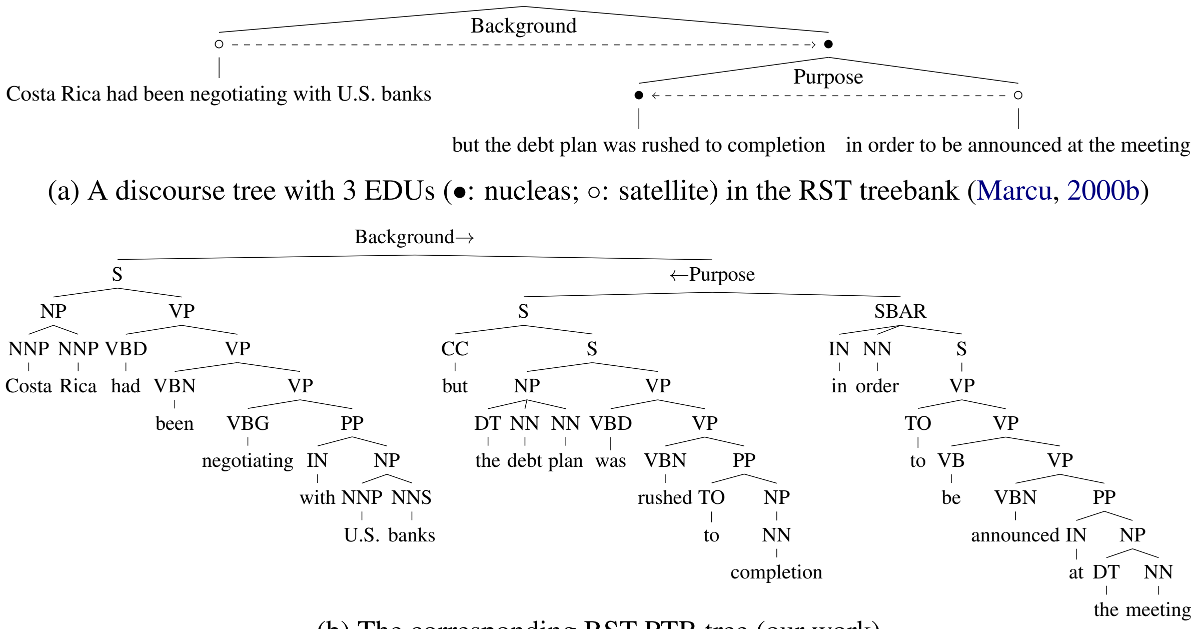
We make the following contributions:

1. We develop a combined representation of constituency and discourse trees to facilitate parsing at both levels without explicit conversion mechanism. Using this representation, we build and release a joint treebank based on the Penn Treebank (Marcus et al., 1993) and RST Treebank (Marcu, 2000a,b) (Section 2).
2. We propose a novel joint parser that parses at both constituency and discourse levels. Our

[*] The source code and the joint treebank are available at https://github.com/kaayy/josydipa.

[†] Current address: Google Inc., New York, NY, USA.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2117–2123
Copenhagen, Denmark, September 7–11, 2017.



(a) A discourse tree with 3 EDUs (●: nucleas; ○: satellite) in the RST treebank (Marcu, 2000b)

(b) The corresponding RST-PTB tree (our work)

Figure 1: Examples of the RST discourse treebank and our syntacto-discourse treebank (PTB-RST).

parser performs discourse parsing in an end-to-end manner, which greatly reduces the efforts required in preprocessing the text for segmentation and feature extraction, and, to our best knowledge, is the first end-to-end discourse parser in literature (Section 3).

3. Even though it simultaneously performs constituency parsing, our parser does *not* use any explicit syntactic feature, nor does it need any binarization of discourse trees, thanks to the powerful span-based framework of Cross and Huang (2016) (Section 3).

4. Empirically, our end-to-end parser outperforms the existing pipelined discourse parsing efforts. When the gold EDUs are provided, our parser is also competitive to other existing approaches with sophisticated features (Section 4).

2 Combined Representation & Treebank

We first briefly review the discourse structures in Rhetorical Structure Theory (Mann and Thompson, 1988), and then discuss how to unify discourse and constituency trees, which gives rise to our syntacto-discourse treebank PTB-RST.

2.1 Review: RST Discourse Structures

In an RST discourse tree, there are two types of branchings. Most of the internal tree nodes are binary branching, with one *nucleus* child containing the core semantic meaning of the current node, and one *satellite* child semantically decorating the nucleus. Like dependency labels, there is a *relation* annotated between each satellite-nucleus pair, such as "Background" or "Purpose". Figure 1(a) shows an example RST tree. There are also non-binary-branching internal nodes whose children are conjunctions, e.g., a "List" of semantically similar EDUs (which are all nucleus nodes); see Figure 2(a) for an example.

2.2 Syntacto-Discourse Representation

It is widely recognized that lower-level lexical and syntactic information can greatly help determining both the boundaries of the EDUs (i.e., discourse segmentation) (Bach et al., 2012) as well as the semantic relations between EDUs (Soricut and Marcu, 2003; Hernault et al., 2010; Joty and Moschitti, 2014; Feng and Hirst, 2014; Ji and Eisenstein, 2014; Li et al., 2014a; Heilman and Sagae, 2015). While these previous approaches rely on pre-trained tools to provide both EDU segmentation and intra-EDU syntactic parse trees, we instead propose to directly determine the low-level segmentations, the syntactic parses, and the high-level discourse parses using a single joint parser. This parser is trained on the combined trees of constituency and discourse structures.

We first convert an RST tree to a format similar

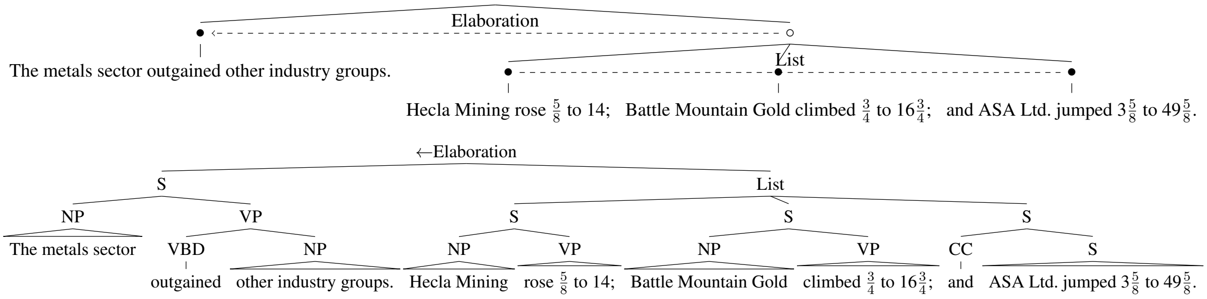
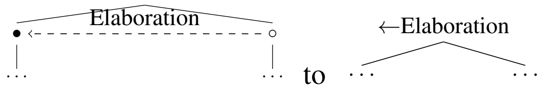
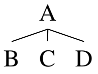
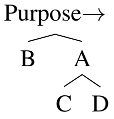

Elaboration
The metals sector outgained other industry groups.
List
Hecla Mining rose $\frac{5}{8}$ to 14; Battle Mountain Gold climbed $\frac{3}{4}$ to $16\frac{3}{4}$; and ASA Ltd. jumped $3\frac{5}{8}$ to $49\frac{5}{8}$.
←Elaboration
S
List
NP
VP
S
S
S
The metals sector
VBD
NP
NP
VP
NP
VP
CC
S
outgained
other industry groups.
Hecla Mining
rose $\frac{5}{8}$ to 14;
Battle Mountain Gold
climbed $\frac{3}{4}$ to $16\frac{3}{4}$;
and
ASA Ltd. jumped $3\frac{5}{8}$ to $49\frac{5}{8}$.

Figure 2: Another example of RST vs. PTB-RST, demonstrating a discourse tree over two sentences and a non-binary relation (List). The lower levels of the PTB-RST tree are collapsed due to space contraints.

to those constituency trees in the Penn Treebank (Marcus et al., 1993). For each binary branching node with a nucleus child and a satellite child, we use the relation as the label of the converted parent node. The nucleus/satellite relation, along with the direction (either ← or →, pointing from satellite to nucleus) is then used as the label. For example, at the top level in Figure 2, we convert



For a conjunctive branch (e.g. "List"), we simply use the relation as the label of the converted node.

After converting an RST tree into the constituency tree format, we then replace each leaf node (i.e., EDU) with the corresponding syntactic (sub)tree from PTB. Given that the sentences in the RST Treebank (Marcu, 2000b) is a subset of that of PTB, we can always find the corresponding constituency subtrees for each EDU leaf node. In most cases, each EDU corresponds to one single (sub)tree in PTB, since the discourse boundaries generally do not conflict with constituencies. In other cases, one EDU node may correspond to multiple subtrees in PTB, and for these EDUs we use the lowest common ancestor of those subtrees in the PTB as the label of that EDU in the converted tree. E.g., if C–D is one EDU in the PTB tree  it might be converted to 

if the relation annonated in RST is B $\xrightarrow{\text{Purpose}}$ C–D.

Figures 1–2 are two examples of discourse trees and their combined syntacto-discourse trees.

2.3 Joint PTB-RST Treebank

Using the conversion strategy described above we build the first joint syntacto-discourse treebank based on the Penn Treebank and RST Treebank. This PTB-RST treebank is released as a set of tools to generate the joint trees given Penn Treebank and RST Treebank data. During the alignment between the RST trees and the PTB trees, we only keep the common parts of the two trees.

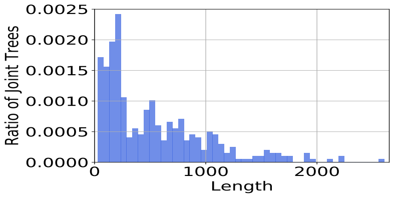


Figure 3: PTB-RST: length distribution (# tokens).

We follow the standard training/testing split of the RST Treebank. In the training set, there are 347 joint trees with a total of 17,837 tokens, and the lengths of the discourses range from 30 to 2,199 tokens. In the test set, there are 38 joint trees with a total of 4,819 tokens, and the lengths vary from 45 to 2,607. Figure 3 shows the distribution of the discourse lengths over the whole dataset, which on average is about 2x of PTB sentence length, but longest ones are about 10x the longest lengths in the Treebank.

3 Joint Syntacto-Discourse Parsing

Given the combined syntacto-discourse treebank, we now propose a joint parser that can perform end-to-end discourse segmentation and parsing.

3.1 Extending Span-based Parsing

As mentioned above, the input sequences are substantially longer than PTB parsing, so we choose linear-time parsing, by adapting a popular greedy constituency parser, the span-based constituency parser of Cross and Huang (2016).

As in span-based parsing, at each step, we maintain a a stack of spans. Notice that in conventional incremental parsing, the stack stores the subtrees

$$\begin{array}{ll}
\text{input} & w_0 \ldots w_{n-1} \\[6pt]
\text{axiom} & \langle\, {}_{-1}\triangle_0 \rangle\text{: } (0, \emptyset) \qquad \text{goal} \quad \langle\, {}_{-1}\triangle_0\triangle_n \rangle\text{: } (\_, t) \\[6pt]
\mathsf{sh} & \dfrac{\langle \ldots\, {}_i\triangle_j \rangle : (c, t)}{\langle \ldots\, {}_i\triangle_j \overset{j}{\triangle}_{j+1} \rangle : (c + sc_{\mathsf{sh}}(i, j), t)}\; j < n \\[12pt]
\mathsf{comb} & \dfrac{\langle \ldots\, {}_i\triangle_k\triangle_j \rangle : (c, t)}{\langle \ldots\, {}_i\overset{k}{\triangle}{}_j \rangle : (c + sc_{\mathsf{comb}}(i, k, j), t)} \\[12pt]
\mathsf{label}_X & \dfrac{\langle \ldots\, {}_i\overset{k}{\triangle}{}_j \rangle : (c, t)}{\langle \ldots\, {}_i\triangle_j \rangle : (c + sc_{\mathsf{label}_X}(i, k, j), t \cup \{{}_iX_j\})} \\[12pt]
\mathsf{nolabel} & \dfrac{\langle \ldots\, {}_i\overset{k}{\triangle}{}_j \rangle : (c, t)}{\langle \ldots\, {}_i\triangle_j \rangle : (c + sc_{\mathsf{nolabel}}(i, k, j), t)}
\end{array}$$

Figure 4: Deductive system for joint syntactic and discourse parsing. $sc_{\mathsf{sh}}(\cdot, \cdot)$, $sc_{\mathsf{comb}}(\cdot, \cdot, \cdot)$, $sc_{\mathsf{label}_X}(\cdot, \cdot, \cdot)$, and $sc_{\mathsf{nolabel}}(\cdot, \cdot, \cdot)$ are scoring functions evaluated in the neural network.

constructed so far, but in span-based constituency parsing, the stack only stores the boundaries of subtrees, which are just a list of indices $\ldots{}_i\triangle_k\triangle_j$. In other words, quite shockingly, no tree structure is represented anywhere in the parser. Please refer Cross and Huang (2016) for details.

Similar to span-based constituency parsing, we alternate between structural (either shift or combine) and label (label_X or nolabel) actions in an odd-even fashion. But different from Cross and Huang (2016), after a structural action, we choose to keep the last branching point k, i.e., ${}_i\overset{k}{\triangle}{}_j$ (mostly for combine, but also trivially for shift). This is because in our parsing mechanism, the discourse relation between two EDUs is actually determined after the previous combine action. We need to keep the splitting point to clearly find the spans of the two EDUs to determine their relations. This midpoint k disappears after a label action; therefore we can use the shape of the last span on the stack (whether it contains the split point, i.e., ${}_i\overset{k}{\triangle}{}_j$ or ${}_i\triangle_j$) to determine the parity of the step and thus no longer need to carry the step z in the state as in Cross and Huang (2016).

The nolabel action makes the binarization of the discourse/constituency tree unnecessary, because nolabel actually combines the top two spans on the stack σ into one span, but without annotating the new span a label. This greatly simplifies the preprocessing and post-processing efforts needed.

	Prec.	Recall	F1
Constituency	87.6	86.9	87.2
Discourse	46.5	40.2	43.0
Overall	83.5	81.6	82.5

Table 1: Accuracies on PTB-RST at constituency and discourse levels.

3.2 Recurrent Neural Models and Training

The scoring functions in the deductive system (Figure 4) are calculated by an underlying neural model, which is similar to the bi-directional LSTM model in Cross and Huang (2016) that evaluates based on span boundary features. Again, it is important to note that no discourse or syntactic tree structures are represented in the features.

During the decoding time, a document is firstl passed into a two-layer bi-directional LSTM model, then the outputs at each text position of the two layers of the bi-directional LSTMs are concatenated as the positional features. The spans at each parsing step can be represented as the feature vectors at the boundaries. The span features are then passed into fully connected networks with softmax to calculate the likelihood of performing the corresponding action or marking the corresponding label.

We use the "training with exploration" strategy (Goldberg and Nivre, 2013) and the dynamic oracle mechanism described in Cross and Huang (2016) to make sure the model can handle unseen parsing configurations properly.

4 Empirical Results

We use the treebank described in Section 2 for empirical evaluation. We randomly choose 30 documents from the training set as the development set.

We tune the hyperparameters of the neural model on the development set. For most of the hyperparameters we settle with the same values suggested by Cross and Huang (2016). To alleviate the overfitting problem for training on the relative small RST Treebank, we use a dropout of 0.5.

One particular hyperparameter is that we use a value β to balance the chances between training following the exploration (i.e., the best action chosen by the neural model) and following the correct path provided by the dynamic oracle. We find that $\beta = 0.8$, i.e., following the dynamic oracle with a probability of 0.8, achieves the best performance. One explanation for this high chance to follow the oracle is that, since our combined trees are signif-

	description	syntactic feats.	segmentation	structure	+nuclearity	+relation
Bach et al. (2012)	segmentation only	Stanford	95.1	-	-	-
Hernault et al. (2010)	end-to-end pipeline	Penn Treebank	94.0	72.3	59.1	47.3
joint syntactic & discourse parsing		-	**95.4**	**78.8**	**65.0**	**52.2**

Table 2: F1 scores of end-to-end systems. "+nuclearity" indicates scoring of tree structures with nuclearity included. "+relation" has both nuclearity and relation included (e.g., ←Elaboration).

		syntactic feats	structure	+nuclearity	+relation
human annotation (Ji and Eisenstein, 2014)		-	88.7	77.7	65.8
sparse	Hernault et al. (2010)	Penn Treebank	83.0	68.4	54.8
	Joty et al. (2013)	Charniak (retrained)	82.7	68.4	55.7
	Joty and Moschitti (2014)	Charniak (retrained)	-	-	57.3
	Feng and Hirst (2014)	Stanford	85.7	71.0	58.2
	Heilman and Sagae (2015)	ZPar (retraied)	83.5	68.1	55.1
	Wang et al. (2017)	Stanford	**86.0**	**72.4**	59.7
neural	Li et al. (2014a)	Stanford	82.4	69.2	56.8
	+ sparse features		84.0	70.8	58.6
	Ji and Eisenstein (2014)	MALT	80.5	68.6	58.3
	+ sparse features		81.6	71.1	**61.8**
	span-based discourse parsing	-	84.2	67.7	56.0

Table 3: Experiments using gold segmentations. The column of "syntactic feats" shows how the syntactic features are calculated in the corresponding systems. Note that our parser predicts solely based on the span features from bi-directionaly LSTM, instead of any explicitly designed syntactic features.

icantly larger than the constituency trees in Penn Treebank, lower β makes the parsing easily divert into wrong trails that are difficult to learn from.

Since our parser essentially performs both constituency parsing task and discourse parsing task. We also evaluate the performances on sentence constituency level and discourse level separately. The result is shown in Table 1. Note that in constituency level, the accuracy is not directly comparable with the accuracy reported in Cross and Huang (2016), since: a) our parser is trained on a much smaller dataset (RST Treebank is about 1/6 of Penn Treebank); b) the parser is trained to optimize the discourse-level accuracy.

Table 2 shows that, in the perspective of end-to-end discourse parsing, our parser first outperforms the state-of-the-art segmentator of Bach et al. (2012), and furthermore, in end-to-end parsing, the superiority of our parser is more pronounced comparing to the previously best parser of Hernault et al. (2010).

On the other hand, the majority of the conventional discourse parsers are not end-to-end: they rely on gold EDU segmentations and pre-trained tools like Stanford parsers to generate features. We perform an experiment to compare the performance of our parser with them given the gold EDU segments (Table 3). Note that most of these parsers do not handle multi-branching discourse nodes and are trained and evaluated on binarized discourse trees (Feng and Hirst, 2014; Li et al., 2014a,b; Ji and Eisenstein, 2014; Heilman and Sagae, 2015), so their performances are actually not directly comparable to the results we reported.

5 Conclusion

We have presented a neural-based incremental parser that can jointly parse at both constituency and discourse levels. To our best knowledge, this is the first end-to-end parser for discourse parsing task. Our parser achieves the state-of-the-art performance in end-to-end parsing, and unlike previous approaches, needs little pre-processing effort.

Acknowledgments

We thank the anonymous reviewers for helpful comments. We are also grateful to Mingbo Ma, James Cross, and Dezhong Deng for suggestions. This work is supported in part by NSF IIS-1656051, DARPA N66001-17-2-4030 (XAI), a Google Faculty Research Award, and HP.

References

Ngo Xuan Bach, Nguyen Le Minh, and Akira Shimazu. 2012. A reranking model for discourse segmentation using subtree features. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 160–168. Association for Computational Linguistics.

Parminder Bhatia, Yangfeng Ji, and Jacob Eisenstein. 2015. Better document-level sentiment analysis from rst discourse parsing. *arXiv preprint arXiv:1509.01599*.

James Cross and Liang Huang. 2016. Incremental parsing with minimal features using bi-directional lstm. *arXiv preprint arXiv:1606.06406*.

Iria da Cunha, Eric SanJuan, Juan-Manuel Torres-Moreno, M Teresa Cabré, and Gerardo Sierra. 2012. A symbolic approach for automatic detection of nuclearity and rhetorical relations among intra-sentence discourse segments in spanish. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 462–474. Springer.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *ACL (1)*, pages 511–521.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the association for Computational Linguistics*, 1:403–414.

Michael Heilman and Kenji Sagae. 2015. Fast rhetorical structure theory discourse parsing. *arXiv preprint arXiv:1505.02425*.

Hugo Hernault, Helmut Prendinger, David A DuVerle, Mitsuru Ishizuka, and Tim Paek. 2010. Hilda: a discourse parser using support vector machine classification. *Dialogue and Discourse*, 1(3):1–33.

Peter Jansen, Mihai Surdeanu, and Peter Clark. 2014. Discourse complements lexical semantics for non-factoid answer reranking. In *ACL (1)*, pages 977–986.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *ACL (1)*, pages 13–24.

Shafiq Joty and Alessandro Moschitti. 2014. Discriminative reranking of discourse parses using tree kernels. *a) A*, 4(5):6.

Shafiq R Joty, Giuseppe Carenini, Raymond T Ng, and Yashar Mehdad. 2013. Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. In *ACL (1)*, pages 486–496.

Jiwei Li and Dan Jurafsky. 2016. Neural net models for open-domain discourse coherence. *arXiv preprint arXiv:1606.01545*.

Jiwei Li, Rumeng Li, and Eduard H Hovy. 2014a. Recursive deep models for discourse parsing. In *EMNLP*, pages 2061–2069.

Sujian Li, Liang Wang, Ziqiang Cao, and Wenjie Li. 2014b. Text-level discourse dependency parsing. In *ACL (1)*, pages 25–35.

Annie Louis, Aravind Joshi, and Ani Nenkova. 2010. Discourse indicators for content selection in summarization. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 147–156. Association for Computational Linguistics.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Daniel Marcu. 2000a. The rhetorical parsing of unrestricted texts: A surface-based approach. *Computational linguistics*, 26(3):395–448.

Daniel Marcu. 2000b. *The theory and practice of discourse parsing and summarization*. MIT press.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Thiago Alexandre Salgueiro Pardo and Maria das Graças Volpe Nunes. 2008. On the development and evaluation of a brazilian portuguese discourse parser. *Revista de Informática Teórica e Aplicada*, 15(2):43–64.

Swapna Somasundaran, Galileo Namata, Janyce Wiebe, and Lise Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 170–179. Association for Computational Linguistics.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics.

ETS Tetreault et al. 2013. Holistic discourse coherence annotation for noisy essay writing.

Kimberly Voll and Maite Taboada. 2007. Not all words are created equal: Extracting semantic orientation as a function of adjective relevance. In *Australasian Joint Conference on Artificial Intelligence*, pages 337–346. Springer.

Yizhong Wang, Sujian Li, and Houfeng Wang. 2017. A two-stage parsing method for text-level discourse analysis. In *Proceedings of ACL*.

Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata. 2014. Dependency-based discourse parser for single-document summarization. In *EMNLP*, pages 1834–1839. Citeseer.

Event Coreference Resolution by Iteratively Unfolding Inter-dependencies among Events

Prafulla Kumar Choubey and **Ruihong Huang**
Department of Computer Science and Engineering
Texas A&M University
(prafulla.choubey, huangrh)@tamu.edu

Abstract

We introduce a novel iterative approach for event coreference resolution that gradually builds event clusters by exploiting inter-dependencies among event mentions within the same chain as well as across event chains. Among event mentions in the same chain, we distinguish within- and cross-document event coreference links by using two distinct pairwise classifiers, trained separately to capture differences in feature distributions of within- and cross-document event clusters. Our event coreference approach alternates between WD and CD clustering and combines arguments from both event clusters after every merge, continuing till no more merge can be made. And then it performs further merging between event chains that are both closely related to a set of other chains of events. Experiments on the ECB+ corpus show that our model outperforms state-of-the-art methods in joint task of WD and CD event coreference resolution.

1 Introduction

Event coreference resolution is the task of identifying event mentions and clustering them such that each cluster represents a unique real world event. The capability of resolving links among coreferring event identities is vital for information aggregation and many NLP applications, including topic detection and tracking, information extraction, question answering and text summarization (Humphreys et al., 1997; Allan et al., 1998; Daniel et al., 2003; Narayanan and Harabagiu, 2004; Mayfield et al., 2009; Zhang et al., 2015). Yet, studies on event coreference are few compared to the well-studied entity coreference resolution.

Event mentions that refer to the same event can occur both within a document (WD) and across multiple documents (CD). One common practice (Lee et al., 2012) to approach CD coreference task is to resolve event coreference in a mega-document created by concatenating topic-relevant documents, which essentially does not distinguish WD and CD event links.

However, intuitively, recognizing CD coreferent event pairs requires stricter evidence compared to WD event linking because it is riskier to link two event mentions from two distinct documents rather than the same document. In a perfect scenario where all WD event mentions are properly clustered and their participants and arguments are combined within a cluster, CD clustering can be performed with ease as sufficient evidences are collected through initial WD clustering. Therefore, another very common practice for event coreference is to first group event mentions within a document and then group WD clusters across documents (Yang et al., 2015).

Nonetheless, WD coreference chains are equally hard to resolve. Event mentions in the same document can look very dissimilar (”killed/ VB” and ”murder/ NN”), have event arguments (i.e., participants and spatio-temporal information of an event (Bejan and Harabagiu, 2010)) partially or entirely omitted, or appear in distinct contexts compared to their antecedent event mentions, partially to avoid repetitions. Under this irresolute state, approaching WD and CD individually is incompetent.

While CD coreference resolution is overall difficult, we observe that some CD coreferent event mentions, especially the ones that appear at the beginning of documents, share sufficient contexts and are relatively easier to resolve. At the same

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2124–2133
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

time, many of them bear sufficient differences that can bring in new information and further lead to more WD merges and consequently more CD merges.

Guided by these observations, we present an event coreference approach that exploits interdependencies among event mentions within an event chain both within a document and across documents, by sequentially applying WD and CD merges in an alternating manner until no more merge can be made. We combine argument features of event mentions after each CD (WD) merge in order to resolve more difficult WD (CD) merges in the following iterations. Furthermore, our model uses two distinct pairwise classifiers that are separately trained with features intrinsic to each type. Specifically, the WD classifier uses features based on event mentions and their arguments while the CD classifier relies on features characterizing surrounding contexts of event mentions as well.

We further exploit second-order interdependencies across event clusters in order to resolve additional WD and CD coreferent event pairs. Intuitively, if two event mentions are related to the same set of events, it is likely that the two event mentions refer to the same real world event, even when their word forms and local contexts are distinct. Specifically, we merge event clusters if their event mentions are tightly associated (i.e., having the same dependency relations) or loosely associated (i.e., co-occurring in the same sentential context) with enough (i.e., passing a threshold) other events that are known coreferent.

Experimental results on the benchmark event coreference dataset, ECB+ (Cybulska and Vossen, 2014b,a), show that our model extensively exploits inter-dependencies between events and outperforms the state-of-the-art methods for both WD and CD event coreference resolution.

2 Related Work

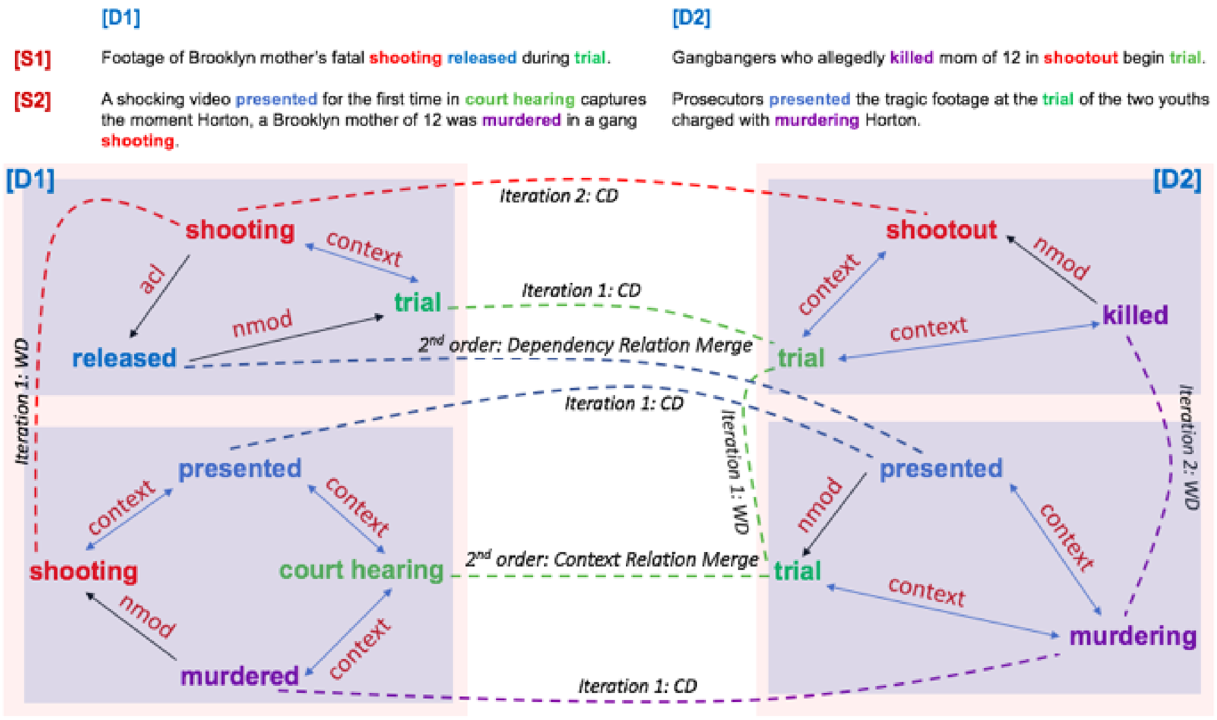
Different approaches, focusing on either of WD or CD coreference chains, have been proposed for event coreference resolution. Works specific to WD event coreference includes pairwise classifiers (Ahn, 2006; Chen et al., 2009) graph based clustering method (Chen and Ji, 2009), information propagation (Liu et al., 2014), and markov logic networks Lu et al. (2016). As to only CD event coreference, Cybulska and Vossen (2015a) created pairwise classifiers using features indicating granularities of event slots and in another work (2015b), grouped events based on compatibilities of event contexts.

Like this work, several studies have considered both WD and CD event coreference resolution task together. However to simplify the problem, they (Lee et al., 2012; Bejan and Harabagiu, 2010, 2014) created a meta-document by concatenating topic-relevant documents and treated both as an identical task. Most recently, Yang et al. (2015) applied a two-level clustering model that first groups event mentions within a document and then groups WD clusters across documents in a joint inference process. Our approach advances these works and emphasizes on different natures of WD and CD clusters along with the benefits of distinguishing WD merges from CD merges and exploiting their mutual dependencies.

Iterative models, in general, have been applied to both entity coreference resolution (Singh et al., 2009; Clark and Manning, 2015, 2016; Wiseman et al., 2016) and prior event coreference resolution (Lee et al., 2012) works, which gradually build clusters and enable later merges to benefit from earlier ones. Especially, Lee et al. (2012) used an iterative model to jointly build entity and event clusters and showed the advantages of information flow between entity and event clusters through semantic role features. Our model, by alternating between WD and CD merges, allows the multi-level flow of first order interdependencies. Moreover, additional cross cluster merges based on 2nd order interdependencies effectively exploits the semantic relations among events, in contrast to only semantic roles (between events and arguments) used in previous work.

3 System Overview and A Worked Example

Inter-dependencies among event mentions can be effectively exploited by conducting sequential WD and CD merges in an iterative manner. In addition, recognizing second order relations between event chains relies on adequate number of event mentions that are already linked. Therefore, our model conducts event coreference in two stages. In the first stage, it iteratively conducts WD and CD merges as suggested by pairwise WD and CD merging classifiers respectively. Argument features of individual event mentions are propagated


[D1]

[S1] Footage of Brooklyn mother's fatal shooting released during trial.

[S2] A shocking video presented for the first time in court hearing captures the moment Horton, a Brooklyn mother of 12 was murdered in a gang shooting.

[D2]

[S1] Gangbangers who allegedly killed mom of 12 in shootout begin trial.

[S2] Prosecutors presented the tragic footage at the trial of the two youths charged with murdering Horton.

Figure 1: An example of Event Coreference using the iterative two stage model. All event mentions are boldfaced; solid arrow line between event mentions show second order relations between them; dashed lines link coreferent event mentions and are tagged with the type of merge.

within a cluster after each merge operation. In the second stage, it explores second order relations across event clusters w.r.t context event mentions in order to carefully generate candidate event clusters and perform further merging.

The example in Figure 1 illustrates the two stages of our proposed approach. It shows two iterations of WD and CD merges. In iteration 1, relatively easy coreferent event mentions were linked, including the two *shooting* and two *trial* event mentions in doc_1 and doc_2 as well as the event mentions *presented*, *trial* and *murder* across the two documents. Argument propagation was conducted after each merge and *murder*'s argument "mother of 12" in doc_1 is combined with the *murder* event in doc_2 after iteration 1. Then in iteration 2, more merges were made by recognizing additional coreferent event mentions including event mentions in one document (e.g., *murdering* and *killed* in doc_2) and event mentions across the two documents (e.g., *shooting* in doc_1 and shootout in doc_2). Next, two additional merges were made by leveraging second-order inter-dependencies. Specifically, both the event mentions *released* in doc_1 and *presented* in doc_2 are in the same dependency relation ("nmod") with a mention of the *trial* event cluster, therefore, a new merge was made between clusters containing the two mentions. Following this, the event mentions *court hearing* in doc_1 and *trial* in doc_2 were identified to have multiple coreferent events in their sentential contexts, therefore, the clusters containing these two event mentions were merged as well.

4 Detailed System Description: Exploiting Interdependencies between Events

4.1 Document Clustering

Our approach starts with a pre-processing step that clusters input documents ($\mathcal{D}$) into a set of document clusters ($\mathcal{C}$). This is meant to reduce search space and mitigate errors (Lee et al., 2012). In our experiments, we used the Affinity Propagation algorithm (Buitinck et al., 2013) on $tf - idf$ vectors, where terms are only proper nouns and verbs (excludes reporting and auxiliary verbs) in the document. While it is interesting to understand the influences of wrong document clusters to event coreference, this algorithm yielded perfect document clusters on the benchmark ECB+

dataset (Cybulska and Vossen, 2014b,a). This is consistent with the prior study (Lee et al., 2012) on the related ECB dataset (Bejan and Harabagiu, 2010) [1], which shows that document clustering in the ECB dataset is trivial.

```
Algorithm 1:
  input: set of Documents $\mathcal{D}$
         Within-Document Classifier: $\Theta_{\mathcal{WD}}$
         Cross-Document Classifier: $\Theta_{\mathcal{CD}}$
  // clusters of event mentions
1  $\mathcal{EM} = \{\}$
  // clusters of Documents
2  $\mathcal{C}$ = ClusterDocument($\mathcal{D}$)
3  for each document cluster $c$ in $\mathcal{C}$ do
4    $\mathcal{EM}' = \{$Singleton Clusters$\}$
     // Iterative WD and CD Merging
6    while $iterate$ do
7      $iterate$ = False
8      for each two clusters $E_1, E_2 \in \mathcal{EM}'$ s.t. $\exists e_1 \in E_1, e_2 \in E_2$, $(e_1, e_2) \in$ a Doc, and score$(\Theta_{\mathcal{WD}}, e_1, e_2) > 0.60$ do
9          Merge$(E_1, E_2, \mathcal{EM}')$
10         $iterate$ = True
11     if not $iterate$ break
12     $iterate$ = False
13     for each two clusters $E_1, E_2 \in \mathcal{EM}'$ s.t. $\exists e_1 \in E_1, e_2 \in E_2$, $(e_1, e_2) \notin$ a Doc, and score$(\Theta_{\mathcal{CD}}, e_1, e_2) > 0.90$ do
14         Merge$(E_1, E_2, \mathcal{EM}')$
15         $iterate$ = True
   // Exploiting Second-Order Inter-dependencies Across Event Chains
16   while $\exists$ two clusters $E_1, E_2 \in \mathcal{EM}'$ s.t. GovernorModifierRelated$(E_1, E_2, \Theta_{\mathcal{CD}})$ do
17     $\mathcal{EM}'$ = Merge$(E_1, E_2, \mathcal{EM}')$
18   while $\exists$ two clusters $E_1, E_2 \in \mathcal{EM}'$ s.t. ContextSimilarity$(E_1, E_2, \Theta_{\mathcal{CD}})$ do
19     $\mathcal{EM}'$ = Merge$(E_1, E_2, \mathcal{EM}')$
20   $\mathcal{EM} = \mathcal{EM} + \mathcal{EM}'$
21 output: $\mathcal{EM}$
```

4.2 Iterative WD and CD Merging

We iteratively conduct WD merges and CD merges until no more merge can be done. We train pairwise classifiers for identifying event clusters to merge. Specifically for WD merges as indicated in lines 8-10 in Algorithm 1, we iteratively go through pairs of clusters that contain a pair of within-document event mentions, one mention from each cluster. If the similarity score between the two event mentions is above a tuned threshold of 0.6 [2], we merge the two clusters. Similarly, for CD merges described in lines 13-15 of Algorithm 1, we iteratively go through pairs of clusters that contain a pair of cross-document event mentions and merge the two clusters if the similarity score between the two event mentions is above another tuned threshold of 0.9 [3]. Following each cluster pair merge, arguments are combined for the two merged clusters.

4.3 Merging by Exploiting Second-Order Inter-dependencies Across Event Chains

Intuitively, two event mentions that share events in their contexts are likely to be coreferent. Similarly, if their context events are coreferent, the two events are likely to be coreferent as well.

First, if two event mentions are in the same dependency relation with two other event mentions that are known coreferent, then the first two event mentions are likely to describe the same real world event as well. In steps 16-17 of Algorithm 1, we perform event cluster merges by collecting evidence pertaining to dependency relations. The subroutine $GovernorModifierRelated$ $(E_1, E_2, \Theta_{\mathcal{CD}})$ checks whether two event mentions e_1 and e_2, from clusters E_1 and E_2 respectively, have a related event e_3 from another cluster E_3, such that $E_3 \notin \{E_1, E_2\}$ and pairs $(e_1, e_3), (e_2, e_3)$ are linked with the same dependency relation. Note that observing shared event mentions in the contexts will increase the likelihood that the two event mentions are coreferent, but we can not sufficiently infer the coreference relation yet, we still need to look at features describing the event mentions. Therefore, if the condition was satisfied, the subroutine eventually makes merges based on the CD confidence score assigned to the event pair (e_1, e_2) but using a lower threshold of 0.8.

In addition, seeing coreferent event mentions in the sentential contexts of two events will increase the likelihood that the two events are coreferent as well. Then as shown in steps 18-19, we fur-

[1] The ECB+ dataset is an extended version of the ECB dataset. Both datasets have documents for the same 43 topics.

[2] all tunings are performed on Validation dataset (topics 23-25)

[3] Note that these high threshold for WD- and CD- classifiers are meant to retain high precision and avoid error propagation in subsequent stages. Output from each classifier is a number bounded in [0,1].

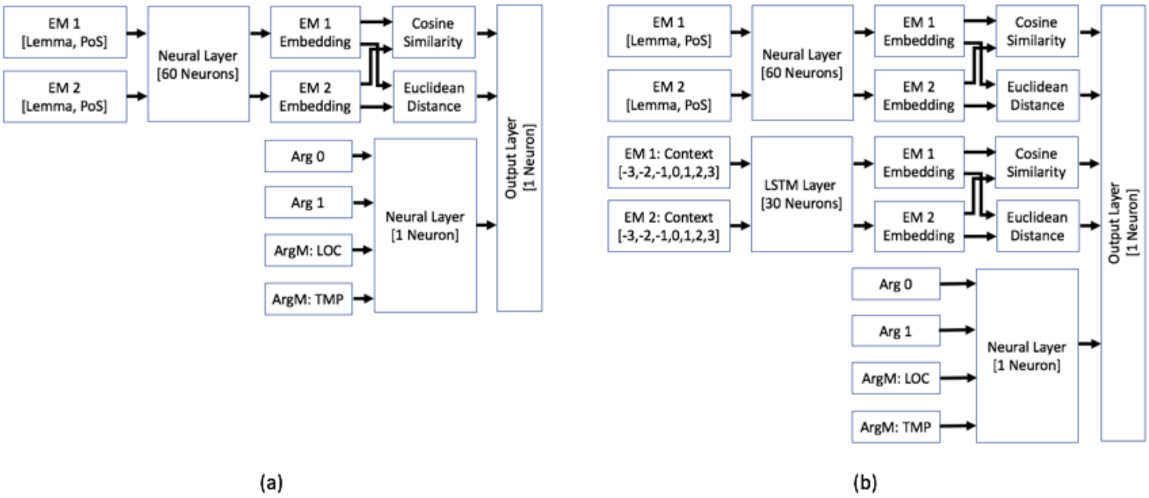


Figure 2: Pairwise Classifiers for resolving (a) Within-Document Coreference Links (b) Cross-Document Coreference Links. *EM*: event mentions; *Arg0, Arg1, ArgM:LOC, ArgM:TMP*: semantic roles.

ther use context events co-occurring in the same sentence as another parameter to perform additional clustering. Subroutine $ContextSimilarity$ $(E_1, E_2, \Theta_{\mathcal{CD}})$ generates a context vector ($\mathcal{CV}$) for each event cluster and check whether cosine similarity between context vectors of two clusters E_1 and E_2 $(cos(\vec{\mathcal{CV}}^1, \vec{\mathcal{CV}}^2))$ is above 0.7. Specifically, we define *context clusters* for an event mention as the different event clusters that have event mentions co-occurring in the same sentence. Then the context vector of an event cluster has an entry for each of its context clusters, with the value to be the number of sentences where event mentions from the two clusters co-occur. This subroutine also makes merges based on the CD confidence score using the same lower threshold of 0.8.

5 Distinguishing WD and CD Merging

We implement two distinct pairwise classifiers to effectively utilize the distributional variations in WD and CD clusters. The first classifier (WD) is used for calculating a similarity between two event mentions within a document and recognizing coreferent event mention pairs. The second classifier (CD) is used for calculating a similarity between two event mentions across two documents and then identifying coreferent event mention pairs across documents. Both classifiers were implemented as neural nets (Chollet, 2015). The architectures of the two classifiers are shown in Figure 2.

WD Classifier: the neural network based WD classifier essentially inherits the features that have been shown effective in previous event coreference studies (Ahn, 2006; Chen et al., 2009), including both features for event words and features for their arguments. Specifically, the classifier includes a common neural layer shared by two event mentions to embed event lemma and parts-of-speech features. Then the classifier calculates cosine similarity and euclidean distance between two event embeddings, one per event mention. In addition, the classifier includes a neural layer component to embed event arguments that are overlapped between the two event mentions. Its output layer takes the calculated cosine similarity and euclidean distance between event mention embeddings as well as the embedding of the overlapped event arguments as input, and output a confidence score to indicate the similarity of the two event mentions.

CD Classifier: the CD classifier mimics the WD classifier except that the CD classifier contains an additional LSTM layer (Hochreiter and Schmidhuber, 1997) to embed context words. The LSTM layer is shared by both event mentions in order to calculate context word embeddings for both event mentions. Specifically, three words to each side of an event word together with the event word itself are used to calculate the context embedding for each event mention. The classifier then calculates cosine similarity and euclidean distance between two context embeddings as well. The output neural net layer will take two sets of cosine similarity and euclidean distance scores that have been calculated w.r.t. context embeddings and event word embeddings, as well as the embedding of the overlapped event arguments as input, and further calculate a confidence score indicating the similarity of two event mentions across documents.

5.1 Characteristics of WD and CD Event Linking

In order to further understand characteristics of within- and cross-document event linking, we trained two classifiers having the same CD classifier architecture (Figure 2(b)) but with different sets of event pairs, within-document or cross-document event pairs, then analyzed the impacts of features on each type of event linking by comparing the neural net learned weights for each feature. Table 1 shows the comparisons of feature weights.

Features	**WD**	**CD**
Event Word Embedding: Euc	1.017	0.207
Event Word Embedding: Cos	1.086	1.142
Context Embedding: Euc	0.038	0.422
Context Embedding: Cos	0.004	3.910
Argument Embedding	0.349	3.270

Table 1: Comparisons of Feature Weights Learned Using In-doc or Cross-doc Coreferent Event Pairs, Euc: Euclidean Distance, Cos: Cosine Similarity

We can see that within-document event linking mainly relies on the euclidean distance and cosine similarity scores calculated using event word features, with a reasonable amount of weight assigned to overlapped arguments' embedding as well. However, only very small weights were assigned to the similarity and distance scores calculated using context embeddings. In contrast, in the classifier trained with cross-doc coreferent event mention pairs, the highest weight was assigned to the cosine similarity score calculated using context embeddings of two event mentions. Additionally, both the cosine similarity score calculated using event word embeddings and the overlapped argument features were assigned high weights as well. The comparisons clearly demonstrate the significantly different nature of WD and CD event coreference.

5.2 Neural Net Classifiers and Training

In both WD and CD classifiers, we use neural network layer with 60 neurons for embedding event word features and another layer with 1 neuron for embedding argument features. Additionally, in CD classifier, we use an LSTM layer with 30 neurons to embed context features. Dropout of 0.25 was applied to both the event word neural net layer and the context layer. We used sigmoid activation function for the dense layers and tanh activation for the LSTM layer. We used 300-dimensional word embeddings and one hot 37[4] dimensional pos tag embeddings in all our experiments. Therefore, input to word embedding layer is a 337-dimensional vector and to LSTM layer is 300*7 dimensional vectors.

We train both classifiers using the ECB+ corpus (Cybulska and Vossen, 2014b,a). We train the WD classifier using all pairs of WD event mentions that are in an annotated event chain as positive instances and using all pairs of WD event mentions that are not in an annotated event chain as negative instances. However, there are significantly more CD coreferent event mention pairs annotated in the ECB+ corpus, therefore, we randomly sampled 70% of all the CD coreferent event mention pairs as positive instances and randomly sampled from non-coreferent CD event mention pairs as negative instances. Specifically, number of negative instances are kept 5 times of positive instances.

Note that the pairwise classifiers will be used throughout the iterative merging stage. However, after each merge, argument propagation is conducted to enrich features for each event mention in the merged cluster and the number of arguments of an event mention will grow after several merges. In order to account for the growing number of arguments in iterative merging, we augment arguments for each event mention in training instances with arguments derived from other event mentions in the same pair. The augmenting was performed randomly for only 50% of event mentions.

6 Evaluation

We perform all the experiments on the ECB+ corpus (Cybulska and Vossen, 2014b,a), which is an extension to the earlier EventCorefBank (ECB) (Bejan and Harabagiu, 2010) dataset. We have adopted the settings used in Yang et al. (2015). We divide the dataset into training set (topics 1-20), validation set (topics 21-23) and test

[4] Corresponding to the unique 36 POS tags based on the Stanford POS tagger (Toutanova et al., 2003) and an additional 'padding'.

set (topics 24-43). Table 2 shows the distribution of the corpus.

	Train	**Dev**	**Test**	**Total**
#Documents	462	73	447	982
#Sentences	7,294	649	7,867	15,810
#Event Mentions	3,555	441	3,290	7,286
#CD Chains	687	47	486	1,220
#WD Chains	2,499	316	2,137	4,952
Avg. WD chain length	2.835	2.589	2.553	2.686
Avg. CD chain length	5.17	9.39	6.77	5.98

Table 2: ECB+ Corpus Statistics.

We used event mentions identified by CRF based event extractor used in Yang et al. (2015) and extracted event arguments by applying state-of-the-art semantic role labeling system (SwiRL (Surdeanu et al., 2007)). In addition, we used the Stanford parser (Chen and Manning, 2014) for generating dependency relations, parts-of-speech tags and lemmas. We use pre-trained Glove vectors (Pennington et al., 2014)[5] for word representation and one-hot vectors for parts-of-speech tags.

We evaluate our model using four commonly adopted event coreference evaluation metrics, namely, **MUC** (Vilain et al., 1995), $\mathbf{B^3}$ (Bagga and Baldwin, 1998), $\mathbf{CEAF_e}$ (Luo, 2005) and **CoNLL F1** (Pradhan et al., 2014). We used the publicly available official implementation of revised coreference scorer (**v8.01**).[6]

6.1 Baseline Systems

We compare our iterative event coreference resolution model with five baseline systems.

LEMMA: The Lemma match baseline links event mentions within- or cross- documents which have the same lemmatized head word. It is often considered a strong baseline for this task.

HDDCRP (Yang et al., 2015): The second baseline is the supervised Hierarchical Distance Dependent Bayesian Model, the most recent event coreference system evaluated on the same ECB+ dataset. This model uses distances between event mentions, generated using a feature-rich learnable distance function, as Bayesian priors for single pass non-parametric clustering.

HDP-LEX[7]: A reimplementation of the unsupervised hierarchical bayesian model by Bejan and Harabagiu (2010, 2014).

Agglomerative[7]: A Reimplementation of two-step agglomerative clustering model, WD clustering followed by CD clustering (Chen et al., 2009).

We have trained our systems using the same ECB+ dataset and the same set of event mentions as these prior systems.

6.2 Our Systems

We evaluate several variation systems of our proposed model.

Common Classifier (WD or CD): the system implementing only the first stage of iterative WD & CD merging. In addition, the same neural net classifier with the architecture as shown in Figure 2(a) (the WD classifier) or in Figure 2(b) (the CD classifier) was applied for both WD and CD merging. The neural net classifiers were trained using all coreferent event mention pairs including both within-document and cross-document ones.

WD and CD Classifiers: distinguishes WD from CD merges by using two distinct classifiers (Figure 2(a), 2(b)) in the first stage of the algorithm.

+ 2nd Order Relations: after iterative WD and CD merges within each individual chain as suggested by pairwise classifiers (the first stage), further merges (the second stage) were conducted leveraging second order event inter-dependencies across event chains.

6.3 Results

Table 3 shows the comparison results for both within-document and cross-document event coreference resolution. In the first stage of iterative merging, using two distinct WD and CD classifiers for corresponding WD and CD merges yields clear improvements for both WD and CD event coreference resolution tasks, compared with using one common classifier for both types of merges. In addition, the second stage of iterative merging further improves both WD and CD event coreference resolution performance stably by leveraging second order event inter-dependencies. The improvements are consistent when measured using various coreference resolution evaluation metrics.

Our full model achieved more than 8% of improvements when compared with the lemma matching baseline, using the CoNLL F1-score for both WD and CD coreference resolution tasks. Furthermore, it outperforms state-of-the-art HDDCRP model for both WD and CD event coreference resolution by 2.1% and 4.9% respectively.

[5] Trained on 840 billion tokens of Common Crawl data, http://nlp.stanford.edu/projects/glove/

[6] https://github.com/conll/reference-coreference-scorers

[7] The results were taken from the paper Yang et al. (2015).

Cross-Document Coreference Results										
	B^3			MUC			$CEAFE_e$			CoNLL
	R	P	F1	R	P	F1	R	P	F1	F1
LEMMA	39.5	73.9	51.4	58.1	78.2	66.7	58.9	37.5	46.2	54.8
Common Classifier (WD)	46	72.8	56.4	60.4	76.8	68.4	59.5	42.1	49.3	58
+ 2nd Order Relations	48.8	72.1	58.2	61.8	78.9	69.3	59.3	44.1	50.6	59.4
Common Classifier (CD)	44.9	64.7	53	66.1	66.4	66.2	51.9	46.4	49	56.1
+ 2nd Order Relations	52.2	58.4	55.1	**70.4**	66.2	68.3	54.1	45.2	49.2	57.5
WD & CD Classifiers	49	71.9	58.3	63.8	78.9	70.6	59.3	48.1	53.1	60.7
+ 2nd Order Relations **(Full Model)**	**56.2**	66.6	**61**	67.5	**80.4**	**73.4**	59	**54.2**	**56.5**	**63.6**
HDDCRP Yang et al. (2015)	40.6	**78.5**	53.5	67.1	80.3	73.1	**68.9**	38.6	49.5	58.7
HDP-LEX Bejan and Harabagiu (2010)	43.7	65.6	52.5	63.5	75.5	69.0	60.2	34.8	44.1	55.2
Agglomerative Chen et al. (2009)	40.2	73.2	51.9	59.2	78.3	67.4	65.6	30.2	41.1	53.6
Within-Document Coreference Results										
LEMMA	56.8	80.9	66.7	35.9	76.2	48.8	67.4	62.9	65.1	60.2
Common Classifier (WD)	59.7	80.5	68.6	44.6	75	55.9	68.2	67.7	67.9	64.2
+ 2nd Order Relations	62.7	79.4	70	50.3	75.2	60.3	68.6	70.5	69.5	66.6
Common Classifier (CD)	65.2	67.1	66.1	47.6	53.9	50.5	69.2	62.1	65.5	60.7
+ 2nd Order Relations	66.9	69.1	68	56.7	55.1	55.9	70.4	63.6	66.8	62.8
WD & CD classifiers	63.8	79.9	70.9	51.6	**75.3**	61.2	68.6	70.5	69.5	67.2
+ 2nd Order Relations **(Full Model)**	**69.2**	76	72.4	**58.5**	67.3	**62.6**	67.9	**76.1**	**71.8**	**68.9**
HDDCRP Yang et al. (2015)	67.3	**85.6**	**75.4**	41.7	74.3	53.4	**79.8**	65.1	71.7	66.8
HDP-LEX Bejan and Harabagiu (2010)	67.6	74.7	71.0	39.1	50.0	43.9	71.4	66.2	68.7	61.2
Agglomerative Chen et al. (2009)	67.6	80.7	73.5	39.2	61.9	48.0	76.0	65.6	70.4	63.9

Table 3: Within- and cross-document event coreference result on ECB+ Corpus.

7 Discussion and Analysis

Cross-Document Coreference Results				
$F_{measure}$	B^3	MUC	$CEAFE_e$	CoNLL
1 Iteration	56	69.3	50.3	58.5
2 Iterations	57.9	69.9	52.4	60.1
3 Iterations	58.3	70.6	53.1	60.7
Within-Document Coreference Results				
1 Iteration	69.7	55.8	68.8	64.8
2 Iterations	70.2	60.3	69.4	66.6
3 Iterations	70.9	61.2	69.5	67.2

Table 4: Per-iteration Performance Analysis for the First Stage of Iterative WD & CD Merging.

Stage I: The first stage of our algorithm, iterative WD and CD merging, went for three iterations (See Table 4). Our analysis of merges in each iteration shows that most of the merges in the initial iteration are between event mentions with the same lemma or shared arguments. In the second and third iterations, more merges were between event mentions with synonymous lemmas or shared arguments that have been accumulated in previous iterations. Example merges between synonymous event mentions include *(nominate, nominations)*, *(die, death)*, *(murder, killing)*, *(hit, strike)*, *(attack, bomb)* etc.

Stage II: It is even more intriguing to discuss the clusters that were merged in stage 2 of merging, that leverages second order event interdependencies across event chains. We found that almost all of the 81 merges happening in the second stage are between event mentions that are quite dissimilar including *(take over, replace)*, *(unveil, announce)*, *(win, victory, comeback)*, *(downtime, problem, outage)*, *(cut, damage)*, *(spark, trigger)* etc. Most interestingly, two event pairs which are antonymous to each other, *(win, beat)* and *(defeat, victory)*, were also correctly merged.

Errors: while our iterative algorithm has gradually resolved coreference relations between event mentions that are synonyms or distant by surface forms, many coreference links were overlooked and many unrelated events were wrongly predicted as coreferential. We analyzed our system's final predictions in order to identify the most common sources of errors.

Missed Coreference Links: We found that many event mentions have few or no argument in their local context, and our event coreference resolution system often failed to link these event mentions with their coreferential mentions. For instance, in the following event mention pairs that were overlooked by the system, *(operations, raids), (operations, sweep), (suicide, hang), (prosecution, jail), and (participating, role)*, one or both event mentions do not have an argument in their local context. This is mainly because the base WD and CD classifiers heavily rely on features extracted from the local context of two event mentions, including event words and event arguments, in resolving the coreference relation. For these event mentions having few arguments identified, the iterative algorithm may get stuck from the beginning.

While it is a grand challenge to further resolve coreferential relations between event mentions that do not have sufficient local features, these missed coreference links easily break a long and influential event chain into several sub-chains, which makes event coreference resolution results less useful for many potential applications, such as text summarization.

Wrongly Predicted Coreference Links: The majority of this type of errors are between non-coreferent event mentions that have the same lemma. This is especially common among reporting event mentions and light verb mentions. For instance, we found that 24 non-coreferent event clusters corresponding to reporting events, e.g., *said, told* and *reported*, and 13 non-coreferent clusters corresponding to light verbs, e.g., *take, give* and *get*, were incorrectly merged by the system.

8 Conclusions and Future Work

We presented a novel approach for event coreference resolution that extensively exploits event inter-dependencies between event mentions in the same chain and event mentions across chains. The approach iteratively conducts WD and CD merges followed by further merges leveraging second order event inter-dependencies across chains. We further distinguish WD and CD merges using two distinct classifiers that capture differences of within- and cross-document event clusters in feature distributions. Our system was shown effective in both WD and CD event coreference and has outperformed the previous best event coreference system in both tasks.

Note that our approach is flexible to incorporate different strategies for conducting WD and CD merges. In the future, we plan to continue to investigate the distinct characteristics of WD and CD coreferent event mentions in order to further improve event coreference performance. Especially, we are interested in including additional discourse-level features for improving WD coreference merge performance, such as, features indicating the distance between two event mentions in a document.

Acknowledgments

We want to thank our anonymous reviewers for providing insightful review comments.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8. Association for Computational Linguistics.

James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. 1998. Topic detection and tracking pilot study final report.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566.

Cosmin Adrian Bejan and Sanda Harabagiu. 2010. Unsupervised event coreference resolution with rich linguistic features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1412–1422. Association for Computational Linguistics.

Cosmin Adrian Bejan and Sanda Harabagiu. 2014. Unsupervised event coreference resolution. *Computational Linguistics*, 40(2):311–347.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

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

Zheng Chen and Heng Ji. 2009. Graph-based event coreference resolution. In *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing*, pages 54–57. Association for Computational Linguistics.

Zheng Chen, Heng Ji, and Robert Haralick. 2009. A pairwise event coreference model, feature impact and evaluation for event coreference resolution. In *Proceedings of the workshop on events in emerging text types*, pages 17–22. Association for Computational Linguistics.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

Kevin Clark and Christopher D Manning. 2015. Entity-centric coreference resolution with model stacking.

Kevin Clark and Christopher D Manning. 2016. Improving coreference resolution by learning entity-level distributed representations.

Agata Cybulska and Piek Vossen. 2014a. Guidelines for ecb+ annotation of events and their coreference. Technical report, Technical report, Technical Report NWR-2014-1, VU University Amsterdam.

Agata Cybulska and Piek Vossen. 2014b. Using a sledgehammer to crack a nut? lexical diversity and event coreference resolution. In *LREC*, pages 4545–4552.

Agata Cybulska and Piek Vossen. 2015a. Translating granularity of event slots into features for event coreference resolution. In *Proceedings of the 3rd Workshop on EVENTS at the NAACL-HLT*, pages 1–10.

Agata Cybulska and Piek Vossen. 2015b. bag of events approach to event coreference resolution. supervised classification of event templates. *IJCLA*, page 11.

Naomi Daniel, Dragomir Radev, and Timothy Allison. 2003. Sub-event based multi-document summarization. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop-Volume 5*, pages 9–16. Association for Computational Linguistics.

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

Kevin Humphreys, Robert Gaizauskas, and Saliha Azzam. 1997. Event coreference for information extraction. In *Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, pages 75–81. Association for Computational Linguistics.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 489–500. Association for Computational Linguistics.

Zhengzhong Liu, Jun Araki, Eduard H Hovy, and Teruko Mitamura. 2014. Supervised within-document event coreference using information propagation. In *LREC*, pages 4539–4544.

Jing Lu, Deepak Venugopal, Vibhav Gogate, and Vincent Ng. 2016. Joint inference for event coreference resolution.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 25–32. Association for Computational Linguistics.

James Mayfield, David Alexander, Bonnie J Dorr, Jason Eisner, Tamer Elsayed, Tim Finin, Clayton Fink, Marjorie Freedman, Nikesh Garera, Paul McNamee, et al. 2009. Cross-document coreference resolution: A key technology for learning by reading. In *AAAI Spring Symposium: Learning by Reading and Learning to Read*, volume 9, pages 65–70.

Srini Narayanan and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of the 20th international conference on Computational Linguistics*, page 693. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 30–35, Baltimore, Maryland. Association for Computational Linguistics.

Sameer Singh, Karl Schultz, and Andrew McCallum. 2009. Bi-directional joint inference for entity resolution and segmentation using imperatively-defined factor graphs. *Machine Learning and Knowledge Discovery in Databases*, pages 414–429.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29:105–151.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of HLT-NAACL 2003*.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding*, pages 45–52. Association for Computational Linguistics.

Sam Wiseman, Alexander M Rush, and Stuart M Shieber. 2016. Learning global features for coreference resolution. In *Proceedings of NAACL-HLT*, pages 994–1004.

Bishan Yang, Claire Cardie, and Peter Frazier. 2015. A hierarchical distance-dependent bayesian model for event coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:517–528.

Tongtao Zhang, Hongzhi Li, Heng Ji, and Shih-Fu Chang. 2015. Cross-document event coreference resolution based on cross-media features. In *EMNLP*, pages 201–206.

When to Finish? Optimal Beam Search for Neural Text Generation (modulo beam size)

Liang Huang and **Kai Zhao**[†] and **Mingbo Ma**
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, Oregon, USA
{liang.huang.sh, kzhao.hf, cosmmb}@gmail.com

Abstract

In neural text generation such as neural machine translation, summarization, and image captioning, beam search is widely used to improve the output text quality. However, in the neural generation setting, hypotheses can finish in different steps, which makes it difficult to decide when to end beam search to ensure optimality. We propose a provably optimal beam search algorithm that will always return the optimal-score complete hypothesis (modulo beam size), and finish as soon as the optimality is established (finishing no later than the baseline). To counter neural generation's tendency for shorter hypotheses, we also introduce a bounded length reward mechanism which allows a modified version of our beam search algorithm to remain optimal. Experiments on neural machine translation demonstrate that our principled beam search algorithm leads to improvement in BLEU score over previously proposed alternatives.

1 Introduction

In recent years, neural text generation using recurrent networks have witnessed rapid progress, quickly becoming the state-of-the-art paradigms in machine translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2014), summarization (Rush et al., 2015; Ranzato et al., 2016), and image captioning (Vinyals et al., 2015; Xu et al., 2015). In the decoder of neural generation, beam search is widely employed to boost the output text quality, often leading to substantial improvement over greedy search (equivalent to beam size 1) in metrics such as BLEU or ROUGE; for example, Ranzato et al. (2016) reported +2.2 BLEU (on single reference) in translation and +3.5 ROUGE-2 in summarization, both using a beam of 10. Our own experiments on machine translation (see Sec. 5) show +4.2 BLEU (on four references) using a beam of 5.

However, unlike traditional beam search in phrase-based MT or shift-reduce parsing where all hypotheses finish in the same number of steps, here in neural generation, hypotheses can finish in vastly different numbers of steps. Once you find a completed hypothesis (by generating the </s> symbol), there are still other active hypotheses in the beam that can continue to grow, which might lead to better scores. Therefore when can you end the beam search? How (and when) can you guarantee that the returned hypothesis has the optimal score modulo beam size?

There have not been satisfying answers to these questions, and existing beam search strategies are heuristic methods that do not guarantee optimality. For example, the widely influential RNNsearch (Bahdanau et al., 2014) employs a "shrinking beam" method: once a completed hypothesis is found, beam size shrinks by 1, and beam search would finish if beam size shrinks to 0 or if the number of steps hits a hard limit. The best scoring completed hypothesis among all completed ones encountered so far is returned. On the other hand, OpenNMT (Klein et al., 2017), whose PyTorch version will be the baseline in our experiments, uses a very different strategy: beam search terminates whenever the highest-ranking hypothesis in the current step is completed (which is also the one returned), without considering any other completed hypotheses. Neither of these two methods guarantee optimality of the returned hypothesis.

We therefore propose a novel and simple beam search variant that will always return the optimal-score complete hypothesis (modulo beam size), and finish as soon as the optimality is established.

[†] Current address: Google Inc., New York, NY, USA.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2134–2139
Copenhagen, Denmark, September 7–11, 2017.

However, another well-known problem remains, that the generated sentences are often too short, compared to previous paradigms such as SMT (Shen et al., 2016). To alleviate this problem, previous efforts introduce length normalization (as a switch in RNNsearch) or length reward (He et al., 2016) borrowed from SMT (Koehn et al., 2007). Unfortunately these changes will invalidate the optimal property of our proposed algorithm. So we introduce a *bounded* length reward mechanism which allows a modified version of our beam search algorithm to remain optimal. Experiments on neural machine translation demonstrate that our principled beam search algorithm leads to improvement in BLEU score over previously proposed alternatives.

2 Neural Generation and Beam Search

Here we briefly review neural text generation and then review existing beam search algorithms.

Assume the input sentence, document, or image is embedded into a vector $\mathbf{x}$, from which we generate the output sentence $\mathbf{y}$ which is a *completed* hypothesis:[1]

$$\begin{aligned}\mathbf{y}^* &= \mathop{\mathbf{argmax}}_{\mathbf{y}:comp(\mathbf{y})} p(\mathbf{y} \mid \mathbf{x}) \\ &= \mathop{\mathbf{argmax}}_{\mathbf{y}:comp(\mathbf{y})} \prod_{i \le |\mathbf{y}|} p(y_i \mid \mathbf{x}, \mathbf{y}_{<i})\end{aligned}$$

where $\mathbf{y}_{<i}$ is a popular shorthand notation for the prefix $y_0 y_1 ... y_{i-1}$. We say that a hypothesis $\mathbf{y}$ is **completed**, notated $comp(\mathbf{y})$, if its last word is `</s>`, i.e.,

$$comp(\mathbf{y}) \triangleq (\mathbf{y}_{|\mathbf{y}|} = \texttt{</s>})$$

in which case it will not be further expanded.

A crucial difference in RNN-based neural generation compared to previous paradigms such as phrase-based MT is that we no longer decompose $p(y_i \mid \mathbf{x}, \mathbf{y}_{<i})$ into the translation model, $p(y_i \mid \mathbf{x})$, and the language model, $p(y_i \mid \mathbf{y}_{<i})$, and more importantly, we no longer approximate the latter by n-gram models. This ability to model arbitrarily-lengthed history using RNNs is an important reason for NMT's substantially improved fluency compared to SMT.

To (approximately) search for the best output $\mathbf{y}^*$, we use beam search, where the beam B_i at step i is an *ordered* list of size (at most) b, and expands to the next beam B_{i+1} of the same size:

$$B_0 = [\langle \texttt{<s>},\ p(\texttt{<s>} \mid \mathbf{x})\rangle]$$

$$B_i = \overset{b}{\mathbf{top}}\{\langle \mathbf{y}' \circ y_i,\ s \cdot p(y_i | \mathbf{x}, \mathbf{y})\rangle \mid \langle \mathbf{y}', s\rangle \in B_{i-1}\}$$

where the notation $\mathbf{top}^b\, S$ selects the top b scoring items from the set S, and each item is a pair $\langle \mathbf{y}, s\rangle$ where $\mathbf{y}$ is the current prefix and s is its accumulated score (i.e., product of probabilities).

3 Optimal Beam Search (modulo beam size)

We propose a very simple method to optimally finish beam search, which guarantees the returned hypothesis is the highest-scoring completed hypothesis modulo beam size; in other words, we will finish as soon as an "optimality certificate" can be established that future hypotheses will never score better than the current best one.

Let $best_{\le i}$ be the *best completed hypothesis so far* up to step i, i.e.,

$$best_{\le i} \triangleq \max\{\mathbf{y} \in \cup_{j \le i} B_j \mid comp(\mathbf{y})\} \quad (1)$$

We update it every time we find a completed hypothesis (if there is none yet, then it remains undefined). Now at any step i, if $best_{\le i}$ is defined, and the highest scoring item $B_{i,1}$ in the current beam B_i scores worse than or equal to $best_{\le i}$, i.e., when

$$B_{i,1} \le best_{\le i} \quad (2)$$

we claim the optimality certificate is established, and terminate beam search, returning $best_{\le i}$ (here smaller means worse, since we aim for the highest-probability completed hypothesis).

Theorem 1 (optimality)**.** *When our beam search algorithm terminates, the current best completed hypothesis (i.e., $best_{\le i}$) is the highest-probability completed hypothesis (modulo beam size).*

Proof. If $B_{i,1} \le best_{\le i}$ then $B_{i,j} \le B_{i,1} \le best_{\le i}$ for all items $B_{i,j}$ in beam B_i. Future descendants grown from these items will only be no better, since probability ≤ 1, so all items in current and future steps are no better than $best_{\le i}$. □

Theorem 2 (early stopping)**.** *Our beam search algorithm terminates no later than OpenNMT's termination criteria (when $B_{i,1}$ is completed).*

Proof. When $B_{i,1}$ is itself completed, $best_{\le i} = \max\{B_{i,1}, \cdots\} \ge B_{i,1}$, so our stopping criteria is also met. □

[1] For simplicity reasons we do not discuss bidirectional LSTMs and attentional mechanisms here but our algorithms still work with those encoders (we have tested them).

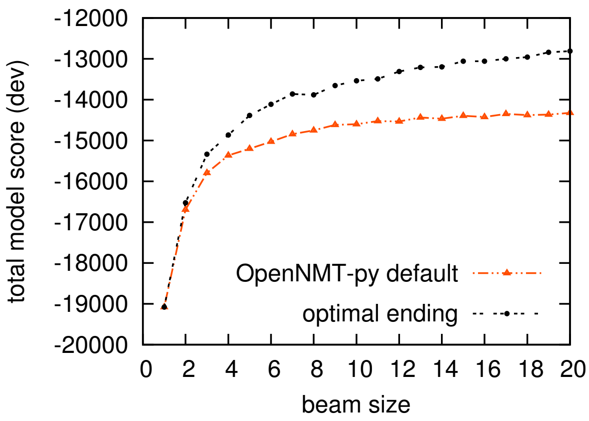


Figure 1: Comparison between optimal beam search and OpenNMT-py's default search, in terms of search quality (model score, ↑ is better).

This above Theorem shows that our search is stopping earlier once the optimality certificate is established, exploring fewer items than OpenNMT's default search. Also note that the latter, even though exploring more items than ours, still can return suboptimal solutions; e.g., when $B_{i,1}$ is worst than $best_{\leq i}$ (they never stored $best_{\leq i}$). In practice, we noticed our search finishes about 3–5 steps earlier than OpenNMT at a beam of 10, and this advantage widens as beam size increases, although the overall speedup is not too noticeable, given the target language sentence length is much longer. Also, our model scores (i.e., log-probabilities) are indeed better (see Fig. 1), where the advantage is also more pronounced with larger beams (note that OpenNMT baseline is almost flat after $b = 10$, while our optimal beam search still steadily improves). Combining these two Theorems, it is interesting to note that our method is not just optimal but also faster.

4 Optimal Beam Search for Bounded Length Reward

However, optimal-score hypothesis, though satisfying in theory, is not ideal in practice, since neural models are notoriously bad in producing very short sentences, as opposed to older paradigms such as SMT (Shen et al., 2016). To alleviate this problem, two methods have been proposed: (a) length normalization, used in RNNsearch as an option, where the revised score of a hypothesis is divided by its length, thus favoring longer sentences; and (b) explicit length reward (He et al., 2016) borrowed from SMT, rewarding each generated word by a constant tuned on the dev set.

Unfortunately, each of these methods breaks the optimality proof of our beam search algorithm in Section 3, since a future hypothesis, being longer, might end up with a higher (revised) score. We therefore devise a novel mechanism called "bounded length reward", that is, we reward each word until the length of the hypothesis is longer than the "estimated optimal length". In machine translation and summarization, this optimal length l can be $ratio \cdot |\mathbf{x}|$ where $|\mathbf{x}|$ is the source sentence length, and $ratio$ is the average ratio of reference translation length over source sentence length on the dev set (in our Chinese-to-English NMT experiments, it is 1.27 as the English side is a bit longer). Note that we use the same $ratio$ estimated from dev on test, assuming that the optimal length ratio for test (which we do not know) should be similar to those of dev ones. We denote $\tilde{sc}(\mathbf{y})$ to be the revised score of hypothesis $\mathbf{y}$ with the bounded length reward, i.e.,

$$\tilde{sc}(\mathbf{y}) \triangleq sc(\mathbf{y}) + r \cdot \min\{l, |\mathbf{y}|\}.$$

We also define $\tilde{best}_{\leq i}$ to be the revised version of $best_{\leq i}$ that optimizes the revised instead of the original score, i.e.,

$$\tilde{best}_{\leq i} \triangleq \mathop{\mathbf{argmax}}_{\mathbf{y} \in \cup_{j \leq i} B_j, comp(\mathbf{y})} \tilde{sc}(\mathbf{y})$$

Now with bounded length reward, we can modify our beam search algorithm a little bit and still guarantee optimality. First we include in the revised cost a reward r for each generated word, as long as the length is less than l, the estimated optimal length. If at step i, the highest scoring item $B_{i,1}$'s revised score (i.e., including bounded length reward) plus the heuristic "future" extra length reward of a descendant, $r \cdot \max\{l - i, 0\}$, is worse than (or equal to) the similarly revised version of $best_{\leq i}$, i.e.,

$$\tilde{sc}(B_{i,1}) + r \cdot \max\{l - i, 0\} \leq \tilde{sc}(\tilde{best}_{\leq i}) \quad (3)$$

at which time we claim the revised optimality certificate is established, and terminate the beam search and return $\tilde{best}_{\leq i}$.

Actually with some trivial math we can simplify the stopping criteria to

$$sc(B_{i,1}) + r \cdot l \leq \tilde{sc}(\tilde{best}_{\leq i}). \quad (4)$$

This much simplified but still equivalent criteria can speed up decoding in practice, since this

	sents	tokens	vocab.	w/ BPE
Chinese	1M	28M	112k	18k
English	1M	23M	93k	10k

Table 1: Machine translation training set.

reward r	0	1	1.1	1.2	1.3	1.4	1.5
BLEU	32.2	34.6	34.6	**34.7**	34.6	34.6	34.6
len. ratio	0.88	.95	.96	.97	.98	.98	.99
best b	4	17	17	15	20	20	17

Table 2: Tuning length reward r (with beam size b=1..20) for optimal bounded-reward beam search.

means we actually do not need to compute the revised score for every hypothesis in the beam; we only need to add the bounded length reward when one is finished (i.e., when updating $\widetilde{best}_{\leq i}$), and the simplified criteria only compares it with the original score of a hypothesis plus a constant reward $r \cdot l$.

Theorem 3 (modified optimality)**.** *Our modified beam search returns the highest-scoring completed hypothesis where the score of an item is its log-probability plus a bounded length reward.*

Proof. by admissibility of the heuristic. □

Theorem 4 (correctness of the simplified criteria)**.** *Eq. 4 is equivalent to Eq. 3.*

Proof. trivial. □

5 Experiments: Neural Translation

5.1 Data Preparation, Training, and Baselines

We conduct experiments on Chinese-to-English neural machine translation, using OpenNMT-py,[2] the PyTorch port of the Lua-based OpenNMT (Klein et al., 2017). We choose this library because PyTorch's combination of Python with Torch's dynamic computation graphs made it much easier to implement various search algorithms on it than on Theano-based implementations derived from RNNsearch (Bahdanau et al., 2014) (such as the widely used GroundHog[3] and Laulysta[4] codebases) as well as the original LuaTorch version of OpenNMT. We use 1M Chinese/English sentence pairs for training (see Table 1 for statistics); we also trained on 2M sentence pairs and only saw a minor improvement so below we report results from 1M training. To alleviate the vocabulary size issue we employ byte-pair encoding (BPE) (Sennrich et al., 2015) which reduces the source and target language vocabulary sizes to 18k and 10k, respectively; we found BPE to significantly improve BLEU scores (by at least +2 BLEU) and reduce training time. Following other papers on Chinese-English translation such as Shen et al. (2016), we use NIST 06 newswire portion (616 sentences) for development and NIST 08 newswire portion (691 sentences) for testing; we will report case-insensitive 4-reference BLEU-4 scores (using original segmentation).

Following OpenNMT-py's default settings, we train our NMT model for 20 epochs to minimize perplexity on the training set (excluding 15% sentences longer than 50 source tokens), with a batch size of 64, word embedding size of 500, and dropout rate of 0.3. The total number of parameters is 29M. Training takes about an hour per epoch on Geforce 980 Ti GPU, and the model at epoch 15 reaches the lowest perplexity on the dev set (9.10) which is chosen as the model for testing.

On dev set, the default decoder of OpenNMT-py reaches 29.2 BLEU with beam size 1 (greedy) and 33.2 BLEU with the default beam size of 5. To put this in perspective, the most commonly used SMT toolkit Moses (Koehn et al., 2007) reaches 30.1 BLEU (with beam size 70) using the same 1M sentence training set (trigram language model trained on the target side). With 2.56M training sentence pairs, Shen et al. (2016) reported 32.7 BLEU on the same dev set using Moses and 30.7 BLEU using the baseline RNNsearch (GroundHog) with beam size 10 (without BPE, without length normalization or length reward). So our OpenNMT-py baseline is extremely competitve.

5.2 Beam Search & Bounded Length Reward

We compare the following beam search variants:

1. OpenNMT-py's default beam search, finishing only when the top hypothesis in a step is completed (see Section 2);

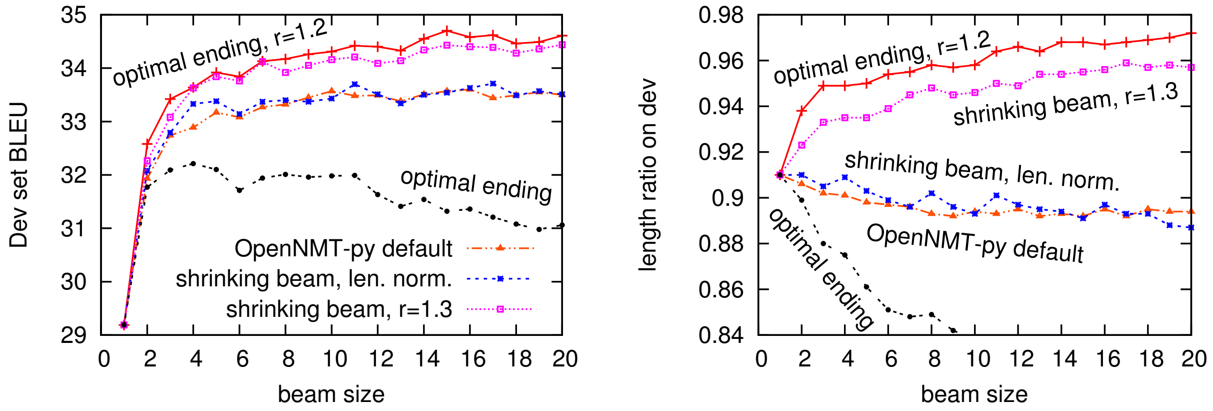
2. The "shrinking beam" method in RNNsearch with two variants to encourage longer translations:

   (a) length normalization; Google NMT (Wu et al., 2016) also adopted a similar

[2] https://github.com/opennmt/opennmt-py

[3] https://github.com/lisa-groundhog/

[4] https://github.com/laulysta/nmt/



(a) BLEU vs. beam size

(b) length ratio vs. beam size

Figure 2: BLEU score and length ratio against beam size (on dev) of various beam search algorithms for neural machine translation.

mechanism.

(b) unbounded length reward (tuned on dev set) in Baidu NMT (He et al., 2016).

3. Our optimal-ending beam search (Section 3);

4. Our modified optimal-ending beam search for bounded length reward (Section 4).

Notice that length reward has no effect on both methods 1 and 2(a) above. To tune the optimal length reward r we run our modified optimal-ending beam search algorithm with all combinations of $r = 0, 0.5, 1, 1.1, 1.2, 1.3, 1.4$ with beam sizes $b = 1 \ldots 20$ on the dev set, since different beam sizes might prefer different length rewards. We found $r = 1.2$ to be the best among all length rewards (see Table 2) which is used in Figure 2 and $b = 15$ is the best for $r = 1.2$.

We can observe from Figure 2 that (a) our optimal beam search with bounded length reward performs the best, and at b=15 it is +5 BLEU better than b=1; (b) pure optimal beam search degrades after b=4 due to extremely short translations; (c) both the shrinking beam method with length normalization and OpenNMT-py's default search alleviate the shortening problem, but still produce very short translations (length ratio ~0.9). (d) the shrinking beam method with length reward works well, but still 0.3 BLEU below our best method. These are confirmed by the test set (Tab. 3).

6 Conclusions

We have presented a beam search algorithm for neural sentence generation that always returns optimal-score completed hypotheses. To counter neural generation's natural tendancy for shorter hypotheses, we introduced a *bounded length reward* mechanism which allows a modified version of our beam search algorithm to remain optimal. Experiments on top of strong baselines have confirmed that our principled search algorithms (together with our bounded length reward mechanism) outperform existing beam search methods in terms of BLEU scores. We will release our implementations (which will hopefully be merged into OpenNMT-py) when this paper is published. [5]

decoder	b	dev	test
Moses	70	30.14	29.41
OpenNMT-py default	16	33.60	29.75
shrinking, len. norm.	17	33.71	30.11
shrinking, reward r=1.3	15	34.42	30.37
optimal beam search, r=1.2	15	**34.70**	**30.61**

Table 3: Final BLEU scores on the test set (nist 08) using best settings from the dev set (nist 06).

Acknowledgments

We thank the anonymous reviewers from both EMNLP and WMT for helpful comments. This work is supported in part by NSF IIS-1656051, DARPA N66001-17-2-4030 (XAI), a Google Faculty Research Award, and HP.

[5] While implementing our search algorithms we also found and fixed an obscure but serious bug in OpenNMT-py's baseline beam search code (not related to discussions in this paper), which boosts BLEU scores by about +0.7 in all cases. We will release this fix as well.

References

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

Wei He, Zhongjun He, Hua Wu, and Haifeng Wang. 2016. Improved neural machine translation with smt features. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, pages 151–157.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*. volume 3, page 413.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints* .

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*. Association for Computational Linguistics, pages 177–180.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *ICLR* .

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* .

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of ACL*.

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

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 3156–3164.

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

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C Courville, Ruslan Salakhutdinov, Richard S Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*. volume 14, pages 77–81.

Steering Output Style and Topic in Neural Response Generation

Di Wang[1,2], **Nebojsa Jojic**[1], **Chris Brockett**[1], **and Eric Nyberg**[2]
[1]Microsoft Research, Redmond, WA, USA
[2]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
diwang@cs.cmu.edu, jojic@microsoft.com

Abstract

We propose simple and flexible training and decoding methods for influencing output style and topic in neural encoder-decoder based language generation. This capability is desirable in a variety of applications, including conversational systems, where successful agents need to produce language in a specific style and generate responses steered by a human puppeteer or external knowledge. We decompose the neural generation process into empirically easier sub-problems: a faithfulness model and a decoding method based on selective-sampling. We also describe training and sampling algorithms that bias the generation process with a specific language style restriction, or a topic restriction. Human evaluation results show that our proposed methods are able to to restrict style and topic without degrading output quality in conversational tasks.

chat context	Where are you?
Twitter bot	i'm on my way to london now.
Star Wars bot	we're heading to the dark side of jabba's palace.
Hillary bot	i'm here in philadelphia, pennsylvania
Trump bot	i went to the white house. we were in brooklyn.
Kennedy bot	i am on the question of the united states.
chat context	What are you doing?
Twitter bot	i'm just chilling with my dad, i'm just sitting in the kitchen and listening to music
Star Wars bot	i'm doing my magic solo.
Hillary bot	i'm running for president, i'm going to be talking about some of these things
Trump bot	i'm building a wall, right now
Kennedy bot	today, i am asking the congress for a new program to make a new effort to increase the tax privileges and to stimulate

Table 1: Example responses from our Star Wars, Hillary, Trump, and Kennedy bots with scented conversation models.

1 Introduction

Neural encoder-decoder models have demonstrated great promise in many sequence generation tasks, including neural machine translation (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014; Luong et al., 2015; Wu et al., 2016), image captioning (Xu et al., 2015), summarization (Rush et al., 2015; Gu et al., 2016; Kikuchi et al., 2016), and conversation generation (Vinyals and Le, 2015; Sordoni et al., 2015; Shang et al., 2015; Shao et al., 2017; Li et al., 2015). These encouraging early successes have motivated research interest in training more natural-sounding conversational systems based on large volumes of open-domain human-to-human interactions. In order to create more human-like patterns of conversation, the agents need to have recognizable (and tunable) style, just as individual humans do, and also need to accept guidance from separate information processing modules in order to increase quality of responses. In an extreme case, an agent may be micro-managed by a human user who uses the neural model to enforce grammar and style (e.g., a level of politeness, or a type of humor), while driving the content directly (e.g., by expressing In this manner, the neural model becomes an authoring tool, rather than an independent chat-bot. On the other hand, in fully automated agent systems, the agent may be influenced by a knowledge database, or some other artificial information system, while running in a pre-set style or a style deemed best based on the course of the conversation.

One obstacle to achieving this with neural language generation models is that the sentence representation is distributed across all coordinates of

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2140–2150
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

the embedding vector in a way that is hard to disentangle, and thus control. In order to gain insight into the full distribution of what a decoder might produce given the prompt sentence as input, the model has to be heavily (and sometimes cleverly) sampled. The second problem is that neural models only become highly functional after training with very large amounts of data, while the strongly recognizable style usually must be defined by a relatively tiny corpus of examples (e.g., all Seinfeld episodes, or all popular song lyrics).

In this paper, we address the challenge of how to enforce the decoder models to mimic a specific language style with only thousands of target sentences, as well as generating specific content in that style. We developed and experimented with several training and decoding procedures to allow the model to adapt to target language style and follow additional content guidance. Our experiments, conducted on an open-domain corpus of Twitter conversations and small persona corpora, show that our methods are capable of responding to queries in a transferred style without significant loss of relevance, and can respond within a specific topic as restricted by a human. Some examples of 'scenting' the base conversation model with particular styles are shown in Table 1. More can be found in the Supplementary Material.

2 Related Work

Recurrent neural network based encoder-decoder models have been applied to machine translation and quickly achieved state-of-the-art results (Bahdanau et al., 2014; Luong et al., 2015). As an extension, the attention mechanism enables the decoder to revisit the input sequence's hidden states and dynamically collects information needed for each decoding step. Specifically, our conversation model is established based on a combination of the models of (Bahdanau et al., 2014) and (Luong et al., 2015) that we found to be effective. In section 3, we describe the attention-based neural encoder-decoder model we used in detail.

This work follows the line of research initiated by (Ritter et al., 2011) and (Vinyals and Le, 2015) who treat generation of conversational dialog as a data-drive statistical machine translation (SMT) problem. Sordoni et al. (2015) extended (Ritter et al., 2011) by re-scoring SMT outputs using a neural encoder-decoder model conditioned on conversation history. Recently, researchers have used neural encoder-decoder models to directly generate responses in an end-to-end fashion without relying on SMT phrase tables(Vinyals and Le, 2015; Sordoni et al., 2015; Shang et al., 2015; Shao et al., 2017; Li et al., 2015).

Li et al. (2016) defined a "persona" as the character that an artificial agent, as actor, plays or performs during conversational interactions. Their dataset requires user identification for all speakers in the training set, while our methods treat the base data (millions of twitter conversations) as unlabeled, and the target persona is defined simply by a relatively small sample of their speech. In this sense, the persona can be any set of text data. In our experiments, for example, we used a generic Star Wars character that was based on the entire set of Star Wars scripts (in addition to 46 million base conversations from Twitter). This provides us with a system that can talk about almost anything, being able to respond to most prompts, but in a recognizable Star Wars style. Other possibilities include training (styling) on famous personalities, or certain types of poetry, or song lyrics, or even mixing styles by providing two or more datasets for styling. Thus our targets are highly recognizable styles, and use of these for emphasis (or caricature) by human puppeteers who can choose from multiple options and guide neural models in a direction they like. We expect that these tools might not only be useful in conversational systems, but could also be popular in social media for text authoring that goes well beyond spelling/grammar auto correction.

3 Neural Encoder-Decoder Background

In general, neural encoder-decoder models aim at generating a target sequence $Y = (y_1, \ldots, y_{T_y})$ given a source sequence $X = (x_1, \ldots, x_{T_x})$. Each word in both source and target sentences, x_t or y_t, belongs to the source vocabulary V_x, and the target vocabulary V_y respectively.

First, an encoder converts the source sequence X into a set of context vectors $C = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{T_x}\}$, whose size varies with regard to the length of the source passage. This context representation is generated using a multi-layered recurrent neural network (RNN). The encoder RNN reads the source passage from the first token until the last one, where $\mathbf{h}_i = \Psi\left(\mathbf{h}_{i-1}, \mathbf{E}_x\left[x_t\right]\right)$. Here $\mathbf{E}_x \in \mathbb{R}^{|V_x|\times d}$ is an embedding matrix containing vector representations of words, and Ψ is a recurrent activation unit that we employ in the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997).

The decoder, which is also implemented as an RNN, generates one word at a time, based on the context vector set returned by the encoder. The decoder's hidden state $\bar{\mathbf{h}}_t$ is a fixed-length continuous vector that is updated in the same way as encoder. At each time step t in the decoder, a time-dependent attentional context vector $\mathbf{c}_t$ is computed based on the current hidden state of the decoder $\bar{\mathbf{h}}_t$ and the whole context set C.

Decoding starts by computing the content-based score of each context vector as: $e_{t,i} = \bar{\mathbf{h}}_t^\top W_a \mathbf{h}_i$. This relevance score measures how helpful the i-th context vector of the source sequence is in predicting next word based on the decoder's current hidden state $\bar{\mathbf{h}}_t^\top$. Relevance scores are further normalized by the softmax function: $\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{T_x}\exp(e_{t,j})}$, and we call $\alpha_{t,i}$ the attention weight. The time-dependent context vector $\mathbf{c}_t$ is then the weighted sum of the context vectors with their attention weights from above: $\mathbf{c}_t = \sum_{i=1}^{T_x} \alpha_{t,i}\mathbf{h}_i$.

With the context vector $\mathbf{c}_t$ and the hidden state $\mathbf{h}_t$, we then combine the information from both vectors to produce an attentional hidden state as follow: $\mathbf{z}_t = \tanh(\mathbf{W}_c[\mathbf{c}_t;\mathbf{h}_t])$. The probability distribution for the next target symbol is computed by $p(y_t = k|\tilde{y}_{<t}, X) \propto \exp(\mathbf{W}_s\mathbf{z}_t + \mathbf{b}_t)$.

4 Decoding with Selective Sampling

The standard objective function for neural encoder-decoder models is the log-likelihood of target T given source S, which at test time yields the statistical decision problem:

$$\hat{T} = \arg\max_T \big\{ \log p(T|S)\big\}. \quad (1)$$

However, as discussed in (Li et al., 2015; Shao et al., 2017), simply conducting beam search over the above objective will tend to generate generic and safe responses that lack diversity, such as *"I am not sure"*. In section 7.3, we present a ranking experiment in which we verify that an RNN-based neural decoder provides a poor approximation of the above conditional probability, and instead biases towards the target language model $p(T)$. Fortunately, the backward model $p(S|T)$ empirically perform much better than $p(T|S)$ on the relevance ranking task. Therefore, we directly apply Bayes' rule to Equation 1, as in statistical machine translation (Brown et al., 1993), and use:

$$\hat{T} = \arg\max_T \big\{ \log p(S|T) + \log p(T)\big\}. \quad (2)$$

Since $p(T|S)$ is empirically biased towards $p(T)$, in practice, this objective also resembles the Maximum Mutual Information (MMI) objective function in (Li et al., 2015).

The challenge now is to develop an effective search algorithm for a target words sequence that maximize the product in Equation 2. Here, we follow a similar process as in (Wen et al., 2015) which generates multiple target hypotheses with stochastic sampling based on $p(T|S)$, and then ranks them with the objective function 2 above. However, as also observed by (Shao et al., 2017), step-by-step naive sampling can accumulate errors as the sequence gets longer.

To reduce language errors of stochastic sampling, we introduce a sample selector to choose the next token among N stochastically sampled tokens based on the predicted output word distributions. The sample selector, which is a multilayer perceptron in our experiments, takes the following features: 1) the log-probability of current sample word in $p(w_t|S)$; 2) the entropy of current predicted word distribution, $\sum_{w_t} P(w_t|S)\log P(w_t|S)$ for all w_t in the vocabulary; 3) the log-probability of current sample word in $p(w_t|\emptyset)$, which we found effective in ranking task. The selector outputs a binary variable that indicates whether the current sample should be accepted or rejected.

At test time, if none of the N sampled tokens are above the classification threshold, we choose the highest scored token. If there are more than 1 acceptable samples among N stochastically sampled tokens, we randomly choose one among them. Ideally, this permits us to safely inject diversity while maintaining language fluency. We also use the sample acceptor's probabilities as the language model score $P(T)$ for objective in equation 2.

As regards directly integrating beam-search, we found (a) that beam-search often produces a set of similar top-N candidates, and (b) that decoding with only the objective $p(Y|X)$ can easily lead to irrelevant candidates. (See section 7.3) Therefore, we use the selective-sampling method to generate candidates for all our experiments; this (a) samples stochastically then (b) selects using a learned objective from data. The sample-then-select approach encourages more diversity (v.s. MMI's beam-search) while still maintain language fluency (v.s. naive-sampling).

5 Output style restriction using a small 'scenting' dataset

In this section, we propose three simple yet effective methods of influencing the language style of the output in the neural encoder-decoder framework. Our language style restricting setup assumes that there is a large open-domain parallel corpus that provides training for context-response relevance, and a smaller monologue speaker corpus that reflects the language characteristics of the target speaker. We will refer to this smaller set as a 'scenting' dataset, since it hints at, or insinuates, the characteristics of the target speaker.

5.1 $Rank$: Search in the Target Corpus

Our first approach to scenting is to simply use the all sentences in the target speaker's corpus as generation candidates, ranked by the objective (2) for a given prompt. Since these sentences are naturally-occurring instead of generated word-by-word, we can safely assume $p(T)$ is constant (and high), and so the objective only requires sorting the sentences based on the backward model $p(S|T)$.

RNN-based ranking methods are among the most effective methods for retrieving relevant responses (Wang and Nyberg, 2015, 2016). Thus this approach is a very strong baseline. Its limitation is also obvious: by limiting all possible responses to a fixed finite set of sentences, this method cannot provide a good response if such a response is not already in the scenting dataset.

5.2 $Multiply$: Mixing the base model and the target language model during generation

In our second method we use both the vanilla encoder-decoder model trained on open-domain corpus and the target domain language model trained on the corpus while decoding output sentence. The idea is to use a speaker's language model, which is also RNN-based in our experiments, to restrict the open-domain encoder-decoder model's step-by-step word prediction. Similar ideas have been tested in domain adaptation for statistical machine translation (Koehn and Schroeder, 2007), where both in-domain and open-domain translation tables were used as candidates for generating target sentence. Because open-domain encoder-decoder models are trained with various kinds of language patterns and topics, choosing a sequence that satisfies both models may produce relevant responses that are also in the target language style. We found that a straightforward way of achieving this is to multiply the two models' distributions $p_1(t|S)^{\lambda_1} p_2(t)^{\lambda_2}$ at each point and then re-normalize before sampling. The weights can be tuned either by the perplexity on the validation set, or through manually controlling the trade-off between style restriction and answer accuracy.

5.3 $Finetune$: Over-training on Target Corpus with Pseudo Context

Fine-tuning is a widely used in the neural network community to achieve transfer learning. This strategy permits us to train the neural encoder-decoder on a larger general parallel corpus, and then use the learned parameters to initialize the training of a styled model. Most of the time, however, the target speaker's corpus will lack training data in parallel form. For example, if we train on song lyrics or movie scripts, or political speeches, the data will not be in a question-answer form. To make encoder-decoder overtraining possible, we treat every sentence in the scenting corpus as a target sentence T generated a pseudo context from the backward model $p(S|T)$ trained on the open-domain corpus. Over-training on such pairs imparts the scenting dataset's language characteristics, while retaining the generality of the original model. We also found that the previous sentence in the styled corpus (i.e., previous sentence in the speech) provides helpful context for the current sentence, analogous with a question-answer link. Thus we use both pseudo context and the previous sentence as possible sources S to fine-tune the in-domain decoder. To avoid overfitting, we stop overtraining when the perplexity on the in-domain validation set starts to increase. A corresponding sample acceptor is also trained for the fine-tuned model: we found it helpful to initialize this from the open-domain model's sample acceptor.

6 Restricting the Output Topic

We further introduce a topic restricting method for neural decoders based on the Counting Grid (Jojic and Perina, 2011) model, by treating language guidance as a topic embedding. Our model extension provides information about the output topic in the form of an additional topic embedding vector to the neural net at each time step.

6.1 CG: Counting Grids

The basic counting grid $\pi_{\mathbf{k}}$ is a set of distributions on the d-dimensional toroidal discrete grid $\mathbf{E}$ indexed by $\mathbf{k}$. The grids in this paper are bi-dimensional and typically from $(E_x = 32) \times (E_y = 32)$ to $(E_x = 64) \times (E_y = 64)$ in size. The index z indexes a particular word in the vocabulary $z = [1 \dots Z]$. Thus, $\pi_{\mathbf{i}}(z)$ is the probability of the word z at the d-dimensional discrete location $\mathbf{i}$, and $\sum_z \pi_{\mathbf{i}}(z) = 1$ at every location on the grid. The model generates bags of words, each represented by a list of words $\mathbf{w} = \{w_n\}_{n=1}^N$ with each word w_n taking an integer value between 1 and Z. The modeling assumption in the basic CG model is that each bag is generated from the distributions in a single window $\mathbf{W}$ of a preset size, e.g., $(W_x = 5) \times (W_y = 5)$. A bag can be generated by first picking a window at a d-dimensional location ℓ, denoted as W_ℓ, then generating each of the N words by sampling a location $\mathbf{k}_n$ for a particular micro-topic $\pi_{\mathbf{k}_n}$ uniformly within the window, and sampling from that micro-topic.

Because the conditional distribution $p(\mathbf{k}_n|\ell)$ is a preset uniform distribution over the grid locations inside the window placed at location ℓ, the variable $\mathbf{k}_n$ can be summed out (Jojic and Perina, 2011), and the generation can directly use the grouped histograms

$$h_\ell(z) = \frac{1}{|\mathbf{W}|} \sum_{\mathbf{j} \in W_\ell} \pi_{\mathbf{j}}(z), \tag{3}$$

where $|\mathbf{W}|$ is the area of the window, e.g. 25 when 5×5 windows are used. In other words, the position of the window ℓ in the grid is a latent variable given which we can write the probability of the bag as

$$P(\mathbf{w}|\ell) = \prod_{w_n \in \mathbf{w}} h_\ell(w_n) = \prod_{w_n \in \mathbf{w}} \Big(\frac{1}{|\mathbf{W}|} \cdot \sum_{\mathbf{j} \in W_\ell} \pi_{\mathbf{j}}(w_n) \Big) \tag{4}$$

As the grid is toroidal, a window can start at any position and there is as many h distributions as there are π distributions. The former will have a considerably higher entropy as they are averages of many π distributions. Although the basic CG model is essentially a simple mixture assuming the existence of a single source (one window) for all the features in one bag, it can have a very large number of (highly related) choices h to choose from. Topic models (Blei et al., 2003; Lafferty and Blei, 2006), on the other hand, are admixtures that capture word co-occurrence statistics by using a much smaller number of topics that can be more freely combined to explain a single document (and this makes it harder to visualize the topics and pinpoint the right combination of topics to use in influencing the output).

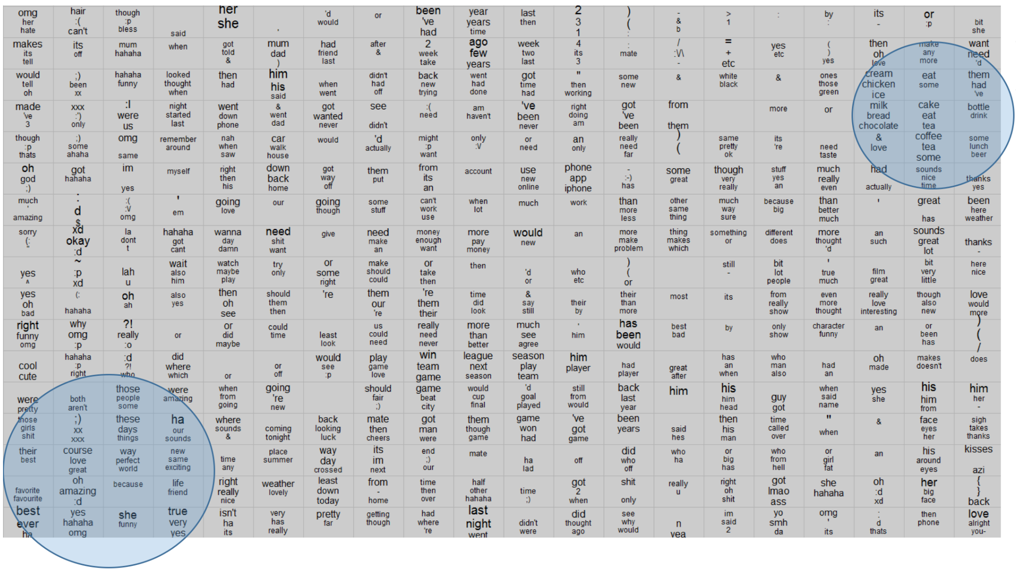
In a well-fit CG model, each data point tends to have a rather peaky posterior location distribution because the model is a mixture. The CG model can be learned efficiently using the EM algorithm because the inference of the hidden variables, as well as updates of π and h can be performed using summed area tables (Crow, 1984), and are thus considerably faster than most of the sophisticated sampling procedures used to train other topic models. The use of overlapping windows helps both in controlling the capacity of the model and in organizing topics on the grid automatically: Two overlapping windows have only slighly different h distributions, making CGs especially useful in visualization applications where the grid is shown in terms of the most likely words in the component distributions π (Perina et al., 2014).[1]

Having trained the grid on some corpus (in our case a sample of the base model's corpus), the mapping of either a source S and/or target T sentence can be obtained by treating the sentences as bags of words. By appending one or both of these mappings to the decoder's embedding of the target T, the end-to-end encoder-decoder learning can be performed in a scenario where the decoder is expected to get an additional hint through a CG mapping. In our experiments, we only used the embedding of the target T as the decoder hint, and we appended the full posterior distribution over CG locations to the encoder's embedding. At test time, we only have the S and need to generate T without knowing where it may map in the counting grid. We considered two ways of providing a mapping:

- The user provides a hint sentence H (could be just a few words in any order), and the CG mapping of the user's hint, i.e. the full posterior distribution $p(\ell|H)$, is used in the decoding. The posterior probabilities over 32 × 32 grid locations are unwrapped into a vector

[1] (Chen et al., 2017) have recently proposed using LDA for topic modeling in Sequence-To-Sequence response generation models. We believe that the CG embedding used here will prove easier to apply and interpret through visualization.

S: I am so hungry!



T1: Have leftover cake, come over!
T2: Let's have cream chicken for lunch.

T1: Of course, ;)
T2: Omg, hahaha. I am very hungry, too

Figure 1: A part of a Counting Grid trained on Twitter data and its use in providing topical hints in decoding. For the source sentence at the top, the decoder may produce the two target samples on the right, if the circled locations are used as a hint, or the two sentences at the bottom if the locations in the lower right are picked.

with a size of $|L| = 1024$, and then concatenated with the word embedding as the input at each time-step. That acts to expand the user's hint into a sentence with similar content (and style if the model is also styled).

- The CG is scanned and a variety of mappings are tested as inputs to provide a diverse set of possible answers. In our experiments, instead of scanning over all 1024 possible locations in the grid, we retrieved several possible answers using information retrieval (ranking of the data samples in the training set based on the source S and picking the top ten). Then the CG mapping $p(\ell|H)$ of these retrieved hints is used to decode several samples from each.

As an example, Figure 1 shows a portion of a CG trained on randomly chosen 800k tweets from the twitter corpus. In each cell of the grid, we show the top words in the distribution $\pi_{\mathbf{j}}(z)$ over words (z) in that location $(\mathbf{j})$. (Each cell has a distribution over the entire vocabulary). As a response to "I am hungry," using two highlighted areas as hints, we can generate either a set of empathic responses, such as 'Me too,' or food suggestions, such as 'Let's have cake.' It will also be evident that some areas of the grid may produce less sensical answers. These can later be pruned by likelihood criteria or by user selection.

7 Experiments

7.1 Datasets

Yahoo! Answer Dataset. We use the Comprehensive Questions and Answers dataset[2] to train and validate the performances of different decoding setups with ranking experiments described in section 7.3. This dataset contains 4.4 million Yahoo! Answers questions and the user-selected best answers. Unlike the conversational datasets, such as the Twitter dataset described below, it contains more relevant and specific responses for each question, which leads to less ambiguity in ranking.

Twitter Conversation Dataset. We trained our base encoder-decoder models on the Twitter Conversation Triple Dataset described in (Sordoni et al., 2015), which consists of 23 million conversational snippets randomly selected from a collection of 129M context-message-response triples extracted from the Twitter Firehose over the 3-month

[2] http://webscope.sandbox.yahoo.com/catalog.php?datatype=l

period from June through August 2012. For the purposes of our experiments, we split the triples into context-message and message-response pairs yielding 46M source-target pairs. For tuning and evaluation, we used the development dataset of size 200K conversation pairs and the test dataset of 5K examples. The corpus is preprocessed using a Twitter specific tokenizer (O'Connor et al., 2010). The vocabulary size is limited to 50,000 excluding the special boundary symbol and the unknown word tag.

Scenting datasets. A variety of persona characters have been trained and tested, including Hillary Clinton, Donald Trump, John F. Kennedy, Richard Nixon, singer-songwriters, stand-up comedians, and a generic Star Wars character. In experiments, we evaluated on a diverse set of representative target speakers:

JFK. We mainly tested our models on John F. Kennedy's speeches collected from American Presidency Project[3], which contains 6474 training and 719 validation sentences.

Star Wars. Movie subtitles of three Star Wars movies are also tested[4]. They are extracted from Cornell Movie-Dialogs Corpus (Danescu-Niculescu-Mizil and Lee, 2011), and have 495 training and 54 validation sentences.

Singer-Songwriter. We also evaluated our approach on a lyric corpus from a collective of singers: Coldplay, Linkin Park, and Green Day. The lyric dataset is collected from mldb.org and has 9182 training and 1020 validation lines.

Debate Chat Contexts. We designed testing questionnaires with 64 chat contexts spanning a range of topics in politic, science, and technology: the sort of questions we might ask in an entertaining political debate.[5] To test the model's ability to control output topic in section 7.4.3, we also created one hint per question.

7.2 Network Setup and Implementation

Our encoder and decoder RNNs contains two-layer stacked LSTMs. Each LSTM layer has a memory size of 500. The network weights are randomly initialized using a uniform distribution $(-0.08, 0.08)$, and are trained with the ADAM optimizer (Kingma and Ba, 2014), with an initial learning rate of 0.002. Gradients were clipped so their norm does not exceed 5. Each mini-batch contains 200 answers and their questions. The words of input sentences were first converted to 300-dimensional vector representations learned from the RNN based language modeling tool word2vec (Mikolov et al., 2013). The beginning and end of each passage are also padded with a special boundary symbol. During decoding, our model generates 500 candidate samples in parallel, then ranks them. As these are processed in batches on GPU, generation is very efficient. We also experimented incorporating an information retrieval (IR) module to automatically collect topic hints for CG-based decoder. Specifically, a full-text index of twitter corpus is built using solr[6], and the top 10 searched results based on the source sentence are be used to generate posterior CG distributions as hints.

7.3 Validating the Decoding Setup with Ranking

We performed a ranking evaluation applying different decoding setups on the Yahoo! Answers dataset. Here we wanted to test the relevance judgment capacities of different setups, and validate the necessity of the new decoding method discussed in section 4. Yahoo! Answers question is used as source S, and its answer is treated as target T. Each test question is associated with one true answer and 19 random answers from the test set. MRR (Mean Reciprocal Rank) and P@1 (precision of top1) were then used as evaluation metrics.

Table 2 shows the answer ranking evaluation results: the forward model $P(T|S)$, by itself is close to the performance of random selection in distinguishing true answer from wrong answers. This implies that a naive beam search over only the forward model may generate irrelevant outputs. One hypothesis was that $P(T|S)$ is biased toward $P(T)$, and performance indeed improves after normalizing by $P(T)$. However, it is difficult to directly decode with objective $P(T|S)/P(T|\emptyset)$, because this objective removes the influence of the target-side language model. Decoding only according to this function will thus result in only low-frequency words and ungrammatical sentences, behavior also noted by (Li et al., 2015; Shao et al., 2017).

[3] http://www.presidency.ucsb.edu/

[4] Koncel-Kedziorski et al. (2016) also uses Star Wars scripts to test theme rewriting of algebra word problems.

[5] See the Supplementary material.

[6] https://lucene.apache.org/solr/

Ranking Methods	MRR	P@1
$P_{rnn}(T\|S)$	0.224	0.075
$P_{rnn}(T\|S)/P_{rnn}(T\|\emptyset)$	0.652	0.524
$P_{rnn}(S\|T)$	0.687	0.556

Table 2: Ranking the true target answer among random answers on Yahoo! Answers test set.

7.4 Human Evaluations

7.4.1 Systems

We tested 10 different system configurations to evaluate the overall output quality, and their abilities of influencing output language style and topic:

- ***vanilla-sampling*** each word in the target.
- ***selective-sampling*** as described in section 4; all the following systems are using it as well.
- ***cg-ir*** uses IR results to create counting grid topic hints (sections 6.1 and 7.2).
- ***rank*** uses proposals from the full JFK corpus as in section 5.1.
- ***multiply*** with a JFK language model as in section 5.2.
- ***finetune*** with JFK dataset as in section 5.3.
- ***finetune-cg-ir*** uses IR results as topic hints for fine-tuned JFK.
- ***finetune-cg-topic*** forced to use the given topic hint for fine-tuned JFK.
- ***singer-songwriter*** fine-tuned cg-topic.
- ***starwars*** fine-tuned cg-topic.

7.4.2 Evaluation Setup

Owing to the low consistency between automatic metrics and human perception on conversational tasks (Liu et al., 2016; Stent et al., 2005) and the lack of true reference responses from persona models, we evaluated the quality of our generated text with a set of judges recruited from Amazon Mechanical Turk (AMT). Workers were selected based on their AMT prior approval rate (>95%). Each questionnaire was presented to 3 different workers. We evaluated our proposed models on the 64 debate chat contexts. Each of the evaluated methods generated 3 samples for every chat context. To ensure calibrated ratings between systems, we show the human judges all system outputs (randomly ordered) for each particular test case at the same time. For each chat context, we conducted three kinds of assessments:

Methods	Quality (MOS)	Style
vanilla-sampling	2.286 ± 0.046	—
selective-sampling	2.681 ± 0.049	10.42%
cg-ir	2.566 ± 0.048	10.24%
rank	2.477 ± 0.048	21.88%
multiply	2.627 ± 0.048	13.54%
finetune	2.597 ± 0.046	20.83%
finetune-cg-ir	2.627 ± 0.049	20.31%
finetune-cg-topic	2.667 ± 0.045	21.09%
singer-songwriter	2.373 ± 0.045	—
starwars	2.677 ± 0.048	—

Table 3: Results of quality assessments with 5-scale mean opinion scores (MOS) and JFK style assessments with binary ratings. Style results are statistically significant compared to the *selective-sampling* by paired t-tests ($p < 0.5\%$).

Quality Assessment Workers were provided with the following guidelines: "Given the chat context, a chat-bot needs to continue the conversation. Rate the potential answers based on your own preference on a scale of 1 to 5 (the highest):"

- 5-Excellent: "Very appropriate response, and coherent with the chat context."
- 4-Good: "Coherent with the chat context."
- 3-Fair: "Interpretable and related. It is OK for you to receive this chat response."
- 2-Poor: "Interpretable, but not related."
- 1-Bad: "Not interpretable."

In this test, the outputs of all 10 systems evaluated are then provided to worker together for a total of 30 responses. In total, we gathered $64 \cdot 30 \cdot 3 = 5760$ ratings for quality assessments, and 47 different workers participated.

Style Assessment. We provided following instructions: "Which candidate responses are likely to have come from or are related to [Persona Name]?". Checkboxes were provided for the responses from style-influenced systems and from *selective-sampling* as a baseline.

Topic Assessment. The instruction was: "Which candidate answers to the chat context above are similar or related to the following answer: '[a hint topic provided by us]'?". This was also a checkbox questionnaire. Candidates are from both style- and topic-influenced systems (fine-tuned cg-topic), and from *selective-sampling* as a baseline.

Persona	Style		Topic	
	Ours	Base	Ours	Base
John F. Kennedy	21%	10%	33%	22%
Star Wars	27%	3%	14%	8%
Singer-Songwriter	31%	23%	17%	9%

Table 4: The style and topic assessments (both binary) of three models with different personas and with restriction of specific target topic for each chat context. All style and topic results are statistically significant compared to the Base (*selective-sampling*) by paired t-tests with $p < 0.5\%$.

7.4.3 Results

Overall Quality. We conducted mean opinion score (MOS) tests for overall quality assessment of generated responses with questionnaires described above. Table 3 shows the MOS results with standard error. It can be seen that all the systems based on selective sampling are significantly better than vanilla sampling baseline. When restricting output's style and/or topic, the MOS score results of most systems do not decline significantly except *singer-songwriter*, which attempts to generate lyrics-like outputs in response to to political debate questions, resulting in uninterpretable strings.

Our *rank* method uses $p(S|T)$ to pick the answer from the original persona corpus, and is thus as good at styling as the person themselves. Because most of our testing questionnaire is political, the *rank* was indeed often able to find related answers in the dataset (JFK). Also, unlike generation-based approaches, *rank* has oracle-level language fluency and it is expected to have quality score of at least 2 ("Interpretable, but not related"). Overall, however, the quality score of *rank* is still lower than other approaches. Note that a hybrid system can actually chose between rank and the decoder's outputs based on likelihood, as shown in the example of bJFk-bNixon debate in the supplemental material.

Influencing the Style. Table 3 also shows the likelihood of being labeled as JFK for different methods. It is encouraging that *finetune* based approaches have similar chances as the *rank* system which retrieves sentences directly from JFK corpus, and are significantly better than the *selective-sampling* baseline.

Influencing both Style and Topic. Table 4 summarizes the results in terms of style (the fraction of answers labeled as in-style for the target persona), and topic (the percentage of answers picked as related to the human-provided topic hint text). We used the last three of the ten listed systems, which are both styled and use specific topic hints to generate answers. These results demonstrate that it is indeed possible to provide simple prompts to a styled model and drive their answers in a desired direction while picking up the style of the persona. It also shows that the style of some characters is harder to recreate than others. For example, workers are more likely to label baseline results as lyrics from a singer-songwriter than lines from Star Wars movies, which might be because lyrics often take significant freedom with structure and grammar. We also found that it is harder for Star Wars and Singer-Songwriter bots to follow topic hints than it is for the John F. Kennedy model, largely because the political debate questions we used overlap less with the topics found in the scenting datasets for those two personas.

8 Conclusions

In this study we investigated the possibility of steering the style and content in the output of a neural encoder-decoder model[7]. We showed that acquisition of highly recognizable styles of famous personalities, characters, or professionals, is achievable, and that it is even possible to allow users to influence the topic direction of conversations. The tools described in the paper are not only useful in conversational systems (e.g., chatbots), but can also be useful as authoring tools in social media. In the latter case, the social media users might use neural models as consultants to help with crafting responses to any post the user is reading. The AMT tests show that these models do indeed provide increased recognizability of the style, without sacrificing quality or relevance.

Acknowledgments

We thank Shashank Srivastava, Donald Brinkman, Michel Galley, and Bill Dolan for useful discussions and encouragement. Di Wang is supported by the Tencent Fellowship and Yahoo! Fellowship, to which we gratefully acknowledge.

[7] The code and testing data are available at https://github.com/digo/steering-response-style-and-topic

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3:993–1022.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Comput. Linguist.* 19(2):263–311.

Xing Chen, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*. pages 3351–3357.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1724–1734.

Franklin C Crow. 1984. Summed-area Tables for Texture Mapping. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA, pages 207–212.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning .

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9(8):1735–1780.

Nebojsa Jojic and Alessandro Perina. 2011. Multidimensional Counting Grids: Inferring Word Order from Disordered Bags of Words. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*. AUAI Press, Arlington, Virginia, United States, pages 547–556.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling Output Length in Neural Encoder-Decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 1328–1338.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6.

Philipp Koehn and Josh Schroeder. 2007. Experiments in Domain Adaptation for Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 224–227.

Rik Koncel-Kedziorski, Ioannis Konstas, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2016. A theme-rewriting approach for generating algebra word problems. *CoRR* abs/1610.06210.

John D Lafferty and David M Blei. 2006. Correlated Topic Models. In Y Weiss, P B Schölkopf, and J C Platt, editors, *Advances in Neural Information Processing Systems 18*, MIT Press, pages 147–154.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Persona-Based Neural Conversation Model. *arXiv* page 10.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B. Dolan. 2015. A Diversity-Promoting Objective Function for Neural Conversation Models. *Arxiv* pages 110–119.

Chia-Wei Liu, Ryan Lowe, Iulian Vlad Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. *CoRR* abs/1603.0.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 1412–1421.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*. pages 3111–3119.

B O'Connor, M Krieger, and D Ahn. 2010. TweetMotif : Exploratory search and topic summarization for Twitter. *4th International AAAI Conference on Weblogs and Social Media* pages 2–3.

Alessandro Perina, Dongwoo Kim, Andrzej Turski, and Nebojsa Jojic. 2014. Skim-reading thousands of documents in one minute: Data indexing and visualization for multifarious search. In *Workshop on Interactive Data Exploration and Analytics (IDEA'14) at KDD 2014*.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven Response Generation in Social Media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '11, pages 583–593.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 379–389.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural Responding Machine for Short-Text Conversation. *CoRR* abs/1503.0.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating Long and Diverse Responses with Neural Conversation Models .

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and William B. Dolan. 2015. A Neural Network Approach to Context-Sensitive Generation of Conversational Responses. In *Naacl-2015*. Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL-HLT 2015), pages 196–205.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating Evaluation Methods for Generation in the Presence of Variation. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing*. Springer-Verlag, Berlin, Heidelberg, pages 341–351.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, pages 3104–3112.

Orioi Vinyals and Quoc V. Le. 2015. A Neural Conversational Model. *ICML Deep Learning Workshop 2015* 37.

Di Wang and Eric Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. In *Annual Meeting of the Association for Computational Linguistics*. pages 707–712.

Di Wang and Eric Nyberg. 2016. CMU OAQA at TREC 2016 LiveQA: An Attentional Neural Encoder-Decoder Approach for Answer Ranking. In *Proceedings of The Twenty-Fifth Text REtrieval Conference, TREC 2016, Gaithersburg, Maryland, USA, November 15-18, 2016*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Peihao Su, David Vandyke, and Steve J. Young. 2015. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* pages 1711–1721.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR* abs/1609.0.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *arXiv preprint arXiv:1502.03044* .

Preserving Distributional Information in Dialogue Act Classification

Quan Hung Tran and **Ingrid Zukerman** and **Gholamreza Haffari**
Faculty of Information Technology
Monash University, Australia
hung.tran,ingrid.zukerman,gholamreza.haffari@monash.edu

Abstract

This paper introduces a novel training/decoding strategy for sequence labeling. Instead of greedily choosing a label at each time step, and using it for the next prediction, we retain the probability distribution over the current label, and pass this distribution to the next prediction. This approach allows us to avoid the effect of label bias and error propagation in sequence learning/decoding. Our experiments on dialogue act classification demonstrate the effectiveness of this approach. Even though our underlying neural network model is relatively simple, it outperforms more complex neural models, achieving state-of-the-art results on the MapTask and Switchboard corpora.

1 Introduction

Dialogue Act (DA) classification is a sequence-labeling task, where a sequence of utterances is mapped into a sequence of DAs. The DAs are semantic classifications of the utterances, and different corpora usually have their own DA labels.

Two of the most popular DA classification datasets are Switchboard (Godfrey et al., 1992; Jurafsky et al., 1997) and MapTask (Anderson et al., 1991). There have been many works on DA classification applied to these two datasets; some focus on textual data (Kalchbrenner and Blunsom, 2013; Stolcke et al., 2000), while others explore speech data (Julia et al., 2010). The classification methods used can be broadly divided into *instance-based methods* (Julia et al., 2010; Gambäck et al., 2011) and *sequence-labeling methods* (Stolcke et al., 2000; Kalchbrenner and Blunsom, 2013; Ji et al., 2016; Shen and Lee, 2016; Tran et al., 2017). Instance-based methods treat each utterance as an independent data point, which allows the application of general machine learning models, such as Support Vector Machines. Sequence-labeling methods include methods based on Hidden Markov Models (HMMs) (Stolcke et al., 2000) and neural networks (Kalchbrenner and Blunsom, 2013; Ji et al., 2016; Shen and Lee, 2016; Tran et al., 2017).

Stolcke *et al.* employed an HMM, using a Language Model to produce emission probabilities. The neural models are particularly successful, posting a higher accuracy on Switchboard than the HMM. Specifically, Kalchbrenner and Blunsom (2013) model a DA sequence with a recurrent neural network (RNN) where sentence representations are constructed by means of a convolutional neural network (CNN); Ji *et al.* (2016) treat the labels as latent variables in a generative RNN; Shen and Lee (2016) employ attentional RNNs for the independent prediction of DAs; and Tran *et al.* (2017) model the DAs in a conversation by means of a hierarchical RNN. In this paper, we also rely on RNNs, but our architecture is much simpler than the above neural models, while posting competitive results.

Most neural network models for DA classification employ greedy decoding (Tran et al., 2017; Ji et al., 2016), as its speed and simplicity support an on-line decoding process (i.e., producing a label immediately after receiving an utterance). For sequential labeling, the DA label in the current time step is very important (Tran et al., 2017). However, using a greedy approach to connect the current label directly to the next label may degrade

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2151–2156
Copenhagen, Denmark, September 7–11, 2017.

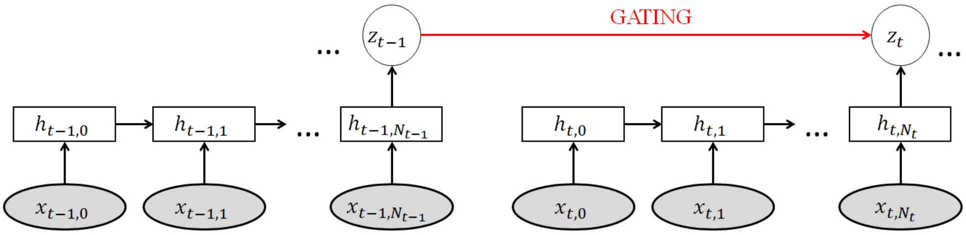


Figure 1: Model architecture.

performance, because the current predicted label may be noisy, which in turn leads to the propagation of errors through the sequence (Tran et al., 2017; Ranzato et al., 2015).

Recently, Bengio *et al.* (2015) proposed a technique called *Scheduled Sampling* that tries to solve the label-bias problem by alternating between the predicted label and the correct label during training. This makes the model gradually adapt to the noisiness of the predicted label. However, this method still relies upon a single current label, and, by omitting the distribution over the possible labels, this model loses information about the current stage. In contrast, we propose to condition the next label on a predicted distribution of the current label. Specifically, we introduce two variants of this idea: the *Uncertainty Propagation* model and the *Average Embedding* model.

2 Sequential DA Prediction

We are interested in predicting DAs $\{z_1, \ldots, z_t\}$ in a conversation as we receive textual utterances $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}$ sequentially. Importantly, we do not have access to future utterances when predicting a DA at time t.

Model. We propose a discriminative model, where the probability of DAs conditioned on utterances is decomposed as follows (Figure 1):

$$p(\boldsymbol{z}_{1:t}|\boldsymbol{x}_{1:t}) = \prod_{i=1}^{t} p_{\boldsymbol{\theta}}(z_i|z_{i-1}, \boldsymbol{x}_i) \qquad (1)$$

where $\boldsymbol{z}_{1:t}$ and $\boldsymbol{x}_{1:t}$ respectively denote the sequence of DAs and utterances up to time step t. Our model resembles a maximum entropy Markov model, as it conditions the label of the next time step on the label of the current step and the next received utterance. The conditional distribution term $p_{\boldsymbol{\theta}}(z_i|z_{i-1}, \boldsymbol{x}_i)$ is realised by neural models as follows:

$$z_i|z_{i-1}, \boldsymbol{x}_i \sim \mathit{softmax}(\boldsymbol{W}^{(z_{i-1})}\boldsymbol{c}(\boldsymbol{x}_i) + \boldsymbol{b}^{(z_{i-1})}) \qquad (2)$$

where $\boldsymbol{c}(\boldsymbol{x}_i)$ is the distributed representation of utterance $\boldsymbol{x}_i$ (discussed below), and $\{\boldsymbol{W}^{(z_{i-1})}, \boldsymbol{b}^{(z_{i-1})}\}$ are DA-specific parameters gated on the current DA z_{i-1}.

The encoding function for an utterance is an RNN with long-short term memory (LSTM) units (Graves, 2013; Hochreiter and Schmidhuber, 1997), where the final hidden state of the RNN is taken as the representation of the whole sequence of text:

$$\boldsymbol{h}_{t,n} = \boldsymbol{f}_{\boldsymbol{\phi}}(\boldsymbol{h}_{t,n-1}, \boldsymbol{e}(x_{t,n})) \ \ , \ \ \boldsymbol{c}(\boldsymbol{x}_t) = \boldsymbol{h}_{t,N_t} \qquad (3)$$

where $x_{t,n}$ is the n-th token in the t-th utterance, and N_t is the length of the utterance.

The parameter set of our model $\boldsymbol{\theta}$ includes $\{\boldsymbol{W}^{(\ell)}, \boldsymbol{b}^{(\ell)}\}_{\ell=1}^{L}$ for the gating component (where L is the number of DAs), as well as the LSTM parameters $\boldsymbol{\phi}$ and the word-embedding table $\{\boldsymbol{e}(w)\}_{w \in \mathcal{W}}$, where $\mathcal{W}$ is the dictionary.

Uncertainty Propagation. In this model, the distribution over the labels at the current time step is passed to the next time step. Specifically, the quantity of interest is the posterior probability of the DA of the next time step given all the utterances observed so far. This posterior probability can be rewritten as

$$\begin{aligned} p_{\boldsymbol{\theta}}(z_t|\boldsymbol{x}_{1:t}) &= \sum_{z_1,\ldots,z_{t-1}} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{1:t}|\boldsymbol{x}_{1:t}) \\ &= \sum_{z_{t-1}} p_{\boldsymbol{\theta}}(z_t|z_{t-1}, \boldsymbol{x}_t) p_{\boldsymbol{\theta}}(z_{t-1}|\boldsymbol{x}_{1:t-1}) \qquad (4) \end{aligned}$$

According to Equation 4, the label uncertainty at the next time step t can be computed by a dynamic programming algorithm based on the label uncertainty of the current time step combined with the local potentials $p_{\boldsymbol{\theta}}(z_t|z_{t-1}, \boldsymbol{x}_t)$.

The use of posterior probability for prediction is also motivated by the minimum Bayes risk decoding (MBR). In the sequential setting, we are interested in predicting the next DA that minimizes the expected loss

$$\begin{aligned} & \arg\min_{\hat{z}_t} \sum_{z_1,\dots,z_{t-1}} p_{\boldsymbol{\theta}}(\boldsymbol{z}_{1:t}|\boldsymbol{x}_{1:t}) \mathit{loss}(z_t, \hat{z}_t) \\ = \quad & \arg\min_{\hat{z}_t} p_{\boldsymbol{\theta}}(z_t|\boldsymbol{x}_{1:t}) \mathit{loss}(z_t, \hat{z}_t) \end{aligned} \tag{5}$$

where $\hat{z}_t$ is the predicted label, z_t is the actual label, and $\mathit{loss}(z_t, \hat{z}_t) = 1_{z_t \neq \hat{z}_t}$.

In addition to decoding, we use posterior probability when training the model. That is, our training objective is

$$\sum_{(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) \in \mathcal{D}} \sum_{t=1}^{T} \log p_{\boldsymbol{\theta}}(z_t|\boldsymbol{x}_{1:t}) \tag{6}$$

where $\mathcal{D}$ is the set of conversations in the training set, each consisting of a sequence of utterances $\boldsymbol{x}_{1:T}$ annotated with its gold sequence of DAs $\boldsymbol{z}_{1:T}$.

Average Embedding. This model offers a new perspective where a neural net combines an inference machine and a model (rather than simply encoding a model). Specifically, this model represents in its architecture, through a weighted sum of embeddings, the inference procedure encoded in Equation 4 for the Uncertainty Propagation model:

$$\mathit{softmax}(E_{q(z_{t-1})}[\boldsymbol{W}^{(z_{t-1})}]\boldsymbol{c}(\boldsymbol{x}_t) + E_{q(z_{t-1})}[\boldsymbol{b}^{(z_{t-1})}]) \tag{7}$$

where $q(z_t)$ is an embedding that represents the uncertainty at time step t. $q(z_t)$ is computed sequentially as new utterances are received, and used in both decoding and training.

This formulation contrasts with Uncertainty Propagation, where the expectation is over the distributions:

$$E_{p_{\boldsymbol{\theta}}(z_{t-1}|\boldsymbol{x}_{1:t-1})}[\mathit{softmax}(\boldsymbol{W}^{(z_{t-1})}\boldsymbol{c}(\boldsymbol{x}_t) + \boldsymbol{b}^{(z_{t-1})})] \tag{8}$$

It is worth noting that Equations 7 and 8 yield the same result if the distributions involved in calculating the expectations are point-mass distributions and they are equal.

Although we could have used a more elaborate neural architecture as the inference machine for the Average Embedding model, we employed a simple *softmax* architecture to make this model comparable with the principled inference algorithm for our Uncertainty Propagation model, which is based on Equation 4.

Comparison to traditional graphical models Our models have several similarities with the traditional HMM model and inference algorithms, such as Forward-Backward decoding and the Viterbi algorithm. However, there are some key differences. Firstly, our model is discriminative, whereas HMM is generative. Secondly, our method is designed for online decoding (the future inputs to a specific classification decision are unknown), whereas both Forward-Backward decoding and Viterbi require access to the whole sequence. Thirdly, Viterbi's objective is to decode for the most probable sequence of labels, whereas our decoding algorithm's objective is to find the sequence of most probable labels (conditioned on the inputs observed so far). Lastly, our Uncertainty Propagation model is not only a basis for decoding, but also for training (the training objective in Equation 6 requires the calculation of the posterior probability in Equation 4). Overall, the best analogue of our Uncertainty Propagation model to methods used in HMMs and other graphical models is the forward message calculation in the Forward-Backward algorithm.

3 Experiments

3.1 Data sets

For our experiments, we use the MapTask and Switchboard corpora.

The MapTask Dialog Act corpus (Anderson et al., 1991) consists of 128 conversations tagged with 13 DAs. The MapTask conversations focus on instructions and clarifications — in the MapTask experiment, there is one instruction giver and one instruction follower. The task of the instruction giver is to guide the instruction follower to follow a pre-determined path, and the instruction follower must draw this path on his/her map. We use 12 conversations for validation, 13 for testing, and the rest for training.

The Switchboard Dialog Act corpus (Godfrey et al., 1992; Jurafsky et al., 1997) consists of 1155 transcribed telephone conversations about general topics, encoded into 42 DAs. We use the experimental setup proposed by Stolcke *et al.* (2000): 1115 conversations for training and 19 for testing.

3.2 Baselines

Our first baseline is the model without any current label information. Next, we compare our models with other strategies for incorporating the current

	Accuracy	
Models	**Switchboard**	**MapTask**
No current label	72.93%	61.27%
True current label	73.15%	63.36%
Predicted current label	73.91%	64.53%
Scheduled Sampling	74.43%	64.50%
Average Embedding	75.04%	65.09%
Uncertainty Propagation	**75.61%**	**65.87%**

Table 1: Results of different strategies to leverage the current label.

Baseline models	**Accuracy**
Julia *et al.* (2010)	55.4 %
Surendran and Levow (2006)	59.1%
Tran *et al.* (2017)	61.6%
Our models:	
Average Embedding	62.6%
Uncertainty Propagation	**62.9%**

Table 2: Results on MapTask data.

Baseline models	**Accuracy**
Stolcke *et al.* (2000)	71.0%
Shen and Lee (2016)	72.6%
Kalchbrenner and Blunsom (2013)	73.9%
Tran *et al.* (2017)	74.5%
Ji *et al.* (2016)	(77.0%) 72.5%
Our models:	
Average Embedding	75.0%
Uncertainty Propagation	**75.6%**

Table 3: Results on Switchboard data.

labels, viz those that use *predicted label* in training, and those that use *correct label*. These models simply employ the predicted/correct label to gate the parameters in Equation 2 during training. During testing, both models can only use the predicted label.

Another baseline is Bengio *et al.*'s (2015) Scheduled Sampling technique, where the training model uses the current correct label with probability p and the predicted label with probability $1 - p$, and p is scheduled to decrease over time. This strategy tries to solve the label-bias problem by making the model gradually adapt to the noisy predicted current label.

Finally, we consider the results obtained by corpus-specific baselines, viz (Julia et al., 2010; Surendran and Levow, 2006; Tran et al., 2017) for MapTask, and (Stolcke et al., 2000; Kalchbrenner and Blunsom, 2013; Ji et al., 2016; Shen and Lee, 2016; Tran et al., 2017) for Switchboard.

3.3 Results

Table 1 compares our results with those obtained by the baselines. Our two models, Uncertainty Propagation and Average Embedding, outperform all the baselines. Among these two models, Uncertainty Propagation, which is more analytically grounded, outperforms the Average Embedding model. Using the true current label during training seems to degrade performance compared to using the predicted label, which is expected, since the true label is not available during testing. The Scheduled Sampling method performs similarly to the predicted-label method for the MapTask corpus, and outperforms this method for the Switchboard corpus.

Tables 2 and 3 compare our models' performance on the MapTask and Switchboard corpora respectively with that of several strong baselines. On MapTask, we achieved the best results for textual input, using the four-fold cross-validation setup used by Surendran and Levow (2006) and Julia *et al.* (2010). On Switchboard, we also obtained the best results among the systems with the same experimental setting. It is worth noting that Ji *et al.* (2016) reported a higher accuracy of 77.0%, but the paper does not provide enough information about the experimental setup to replicate this result, and we only got 72.5% accuracy using the paper's publicly available code.

3.4 Analysis

To quantify the effectiveness of the different models on reducing the label-bias problem, we calculate the probability of the models making a correct prediction after they have made a sequence of n mistakes. We expect our models, Uncertainty Propagation and Average Embedding, to be more robust than the label-sensitive baselines in recovering from errors.

The results in Table 4 confirm our expectations. The simple model with no current label, while performing worse than all other models in accuracy, does not suffer from the label-bias problem. Among the models with current label information, Uncertainty Propagation suffers the least from label bias. It even outperforms the model with no current label on Switchboard for all values of n, and on MapTask for $n = 2$. Interestingly, Average Embedding performs quite well for $n = 1$, but

	MapTask			Switchboard		
	$n = 1$	$n = 2$	$n = 3$	$n = 1$	$n = 2$	$n = 3$
Not affected by label bias:						
No Previous label	60.29%	56.90%	55.67%	66.99%	63.10%	56.52%
Affected by label bias:						
True current label	53.12%	50.38%	47.89%	61.74%	60.93%	60.71%
Predicted current label	55.89%	53.65%	49.10%	64.38%	62.21%	62.59%
Scheduled Sampling	54.28%	53.32%	50.00%	64.67%	63.49%	60.87%
Average Embedding	56.50%	53.76%	52.56%	66.51%	61.71%	55.22%
Uncertainty Propagation	**57.13%**	**57.37%**	**53.93%**	**67.78%**	**66.57%**	**66.36%**

Table 4: Probability that the models recover from a sequence of n prediction mistakes.

its ability to recover from errors drops quickly as the length of the erroneous conditioning sequence increases, especially on Switchboard, where the number of labels is higher. This may explain its slightly lower accuracy compared to the Uncertainty Propagation model. However, in general, the difference in accuracy between these two models is small, because they are rather unlikely to make several consecutive errors.

4 Conclusion

In this paper, we proposed two strategies to encode current label uncertainty in sequence-labeling RNN models. The experimental results show that our models achieve a very strong performance on the MapTask and Switchboard corpora using a simple underlying RNN architecture.

Although we experimented with DA classification, the idea presented in this paper is general, and can be applied to many sequence-labeling tasks. Our approach is particularly suitable for tasks involving streaming data where the model only has access to current and previous observations.

In the future, we plan to combine our strategies with more complex neural architectures, and explore their application to other sequence-labeling problems.

References

Anne H Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, et al. 1991. The HCRC MapTask corpus. *Language and speech* 34(4):351–366.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*. pages 1171–1179.

Björn Gambäck, Fredrik Olsson, and Oscar Täckström. 2011. Active learning for dialogue act classification. In *Interspeech 2011 – Proceedings of the International Conference on Spoken Language Processing*. pages 1329–1332.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*. IEEE, volume 1, pages 517–520.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* .

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

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse-driven language models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California, pages 332–342. http://www.aclweb.org/anthology/N16-1037.

Fatema N Julia, Khan M Iftekharuddin, and ATIQ U ISLAM. 2010. Dialog act classification using acoustic and discourse information of MapTask data. *International Journal of Computational Intelligence and Applications* 9(04):289–311.

Daniel Jurafsky, Elizabeth Shriberg, and Debra Biasca. 1997. Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation Coders Manual, Draft 13. Technical report, University of Colorado.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *arXiv preprint arXiv:1306.3584* .

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* .

Sheng-syun Shen and Hung-yi Lee. 2016. Neural attention models for sequence classification: Analysis and application to key term extraction and dialogue act detection. *arXiv preprint arXiv:1604.00077* .

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics* 26(3):339–373.

Dinoj Surendran and Gina-Anne Levow. 2006. Dialog act tagging with support vector machines and hidden markov models. In *Interspeech 2006 – Proceedings of the International Conference on Spoken Language Processing*. pages 1950–1953.

Quan Hung Tran, Ingrid Zukerman, and Gholamreza Haffari. 2017. A hierarchical neural model for learning sequences of dialogue acts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain, pages 428–437. http://www.aclweb.org/anthology/E17-1041.

Adversarial Learning for Neural Dialogue Generation

Jiwei Li[1], **Will Monroe**[1], **Tianlin Shi**[1], **Sébastien Jean**[2], **Alan Ritter**[3] **and Dan Jurafsky**[1]

[1]Stanford University, Stanford, CA, USA
[2]New York University, NY, USA
[3]Ohio State University, OH, USA
jiweil,wmonroe4,tianlins,jurafsky@stanford.edu
sebastien@cs.nyu.edu
ritter.1492@osu.edu

Abstract

In this paper, drawing intuition from the Turing test, we propose using adversarial training for open-domain dialogue generation: the system is trained to produce sequences that are indistinguishable from human-generated dialogue utterances. We cast the task as a reinforcement learning (RL) problem where we jointly train two systems, a generative model to produce response sequences, and a discriminator—analagous to the human evaluator in the Turing test— to distinguish between the human-generated dialogues and the machine-generated ones. The outputs from the discriminator are then used as rewards for the generative model, pushing the system to generate dialogues that mostly resemble human dialogues.

In addition to adversarial training we describe a model for adversarial *evaluation* that uses success in fooling an adversary as a dialogue evaluation metric, while avoiding a number of potential pitfalls. Experimental results on several metrics, including adversarial evaluation, demonstrate that the adversarially-trained system generates higher-quality responses than previous baselines.

1 Introduction

Open domain dialogue generation (Ritter et al., 2011; Sordoni et al., 2015; Xu et al., 2016; Wen et al., 2016; Li et al., 2016b; Serban et al., 2016c, 2017) aims at generating meaningful and coherent dialogue responses given the dialogue history. Prior systems, e.g., phrase-based machine translation systems (Ritter et al., 2011; Sordoni et al., 2015) or end-to-end neural systems (Shang et al., 2015; Vinyals and Le, 2015; Li et al., 2016a; Yao et al., 2015; Luan et al., 2016) approximate such a goal by predicting the next dialogue utterance given the dialogue history using the maximum likelihood estimation (MLE) objective. Despite its success, this over-simplified training objective leads to problems: responses are dull, generic (Sordoni et al., 2015; Serban et al., 2016a; Li et al., 2016a), repetitive, and short-sighted (Li et al., 2016d).

Solutions to these problems require answering a few fundamental questions: what are the crucial aspects that characterize an ideal conversation, how can we quantitatively measure them, and how can we incorporate them into a machine learning system? For example, Li et al. (2016d) manually define three ideal dialogue properties (ease of answering, informativeness and coherence) and use a reinforcement-learning framework to train the model to generate highly rewarded responses. Yu et al. (2016b) use keyword retrieval confidence as a reward. However, it is widely acknowledged that manually defined reward functions can't possibly cover all crucial aspects and can lead to suboptimal generated utterances.

A good dialogue model should generate utterances indistinguishable from human dialogues. Such a goal suggests a training objective resembling the idea of the Turing test (Turing, 1950). We borrow the idea of adversarial training (Goodfellow et al., 2014; Denton et al., 2015) in computer vision, in which we jointly train two models, a generator (a neural SEQ2SEQ model) that defines the probability of generating a dialogue sequence, and a discriminator that labels dialogues as human-generated or machine-generated. This discriminator is analogous to the evaluator in the Turing test. We cast the task as a reinforcement learning problem, in which the quality of machine-generated utterances is measured by its ability to fool the discriminator into believing that it is a

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2157–2169
Copenhagen, Denmark, September 7–11, 2017.

human-generated one. The output from the discriminator is used as a reward to the generator, pushing it to generate utterances indistinguishable from human-generated dialogues.

The idea of a Turing test—employing an evaluator to distinguish machine-generated texts from human-generated ones—can be applied not only to training but also testing, where it goes by the name of adversarial evaluation. Adversarial evaluation was first employed in Bowman et al. (2016) to evaluate sentence generation quality, and preliminarily studied for dialogue generation by Kannan and Vinyals (2016). In this paper, we discuss potential pitfalls of adversarial evaluations and necessary steps to avoid them and make evaluation reliable.

Experimental results demonstrate that our approach produces more interactive, interesting, and non-repetitive responses than standard SEQ2SEQ models trained using the MLE objective function.

2 Related Work

Dialogue generation Response generation for dialogue can be viewed as a source-to-target transduction problem. Ritter et al. (2011) frame the generation problem as a machine translation problem. Sordoni et al. (2015) improved Ritter et al.'s system by rescoring the outputs of a phrasal MT-based conversation system with a neural model incorporating prior context. Recent progress in SEQ2SEQ models have inspired several efforts (Vinyals and Le, 2015; Serban et al., 2016a,d; Luan et al., 2016) to build end-to-end conversational systems that first apply an encoder to map a message to a distributed vector representing its meaning and then generate a response from the vector.

Our work adapts the encoder-decoder model to RL training, and can thus be viewed as an extension of Li et al. (2016d), but with more general RL rewards. Li et al. (2016d) simulate dialogues between two virtual agents, using policy gradient methods to reward sequences that display three useful conversational properties: informativity, coherence, and ease of answering. Our work is also related to recent efforts to integrate the SEQ2SEQ and reinforcement learning paradigms, drawing on the advantages of both (Wen et al., 2016). For example, Su et al. (2016) combine reinforcement learning with neural generation on tasks with real users. Asghar et al. (2016) train an end-to-end RL dialogue model using human users.

Dialogue quality is traditionally evaluated (Sordoni et al., 2015, e.g.) using word-overlap metrics such as BLEU and METEOR scores used for machine translation. Some recent work (Liu et al., 2016) has started to look at more flexible and reliable evaluation metrics such as human-rating prediction (Lowe et al., 2017) and next utterance classification (Lowe et al., 2016).

Adversarial networks The idea of generative adversarial networks has enjoyed great success in computer vision (Radford et al., 2015; Chen et al., 2016a; Salimans et al., 2016). Training is formalized as a game in which the generative model is trained to generate outputs to fool the discriminator; the technique has been successfully applied to image generation.

However, to the best of our knowledge, this idea has not achieved comparable success in NLP. This is due to the fact that unlike in vision, text generation is discrete, which makes the error outputted from the discriminator hard to backpropagate to the generator. Some recent work has begun to address this issue: Lamb et al. (2016) propose providing the discriminator with the intermediate hidden vectors of the generator rather than its sequence outputs. Such a strategy makes the system differentiable and achieves promising results in tasks like character-level language modeling and handwriting generation. Yu et al. (2016a) use policy gradient reinforcement learning to backpropagate the error from the discriminator, showing improvement in multiple generation tasks such as poem generation, speech language generation and music generation. Outside of sequence generation, Chen et al. (2016b) apply the idea of adversarial training to sentiment analysis and Zhang et al. (2017) apply the idea to domain adaptation tasks.

Our work is distantly related to recent work that formalizes sequence generation as an action-taking problem in reinforcement learning. Ranzato et al. (2016) train RNN decoders in a SEQ2SEQ model using policy gradient to obtain competitive machine translation results. Bahdanau et al. (2017) take this a step further by training an actor-critic RL model for machine translation. Also related is recent work (Shen et al., 2016; Wiseman and Rush, 2016) to address the issues of exposure bias and loss-evaluation mismatch in neural translation.

3 Adversarial Training for Dialogue Generation

In this section, we describe in detail the components of the proposed adversarial reinforcement learning model. The problem can be framed as follows: given a dialogue history x consisting of a sequence of dialogue utterances,[1] the model needs to generate a response $y = \{y_1, y_2, ..., y_T\}$. We view the process of sentence generation as a sequence of actions that are taken according to a policy defined by an encoder-decoder recurrent neural network.

3.1 Adversarial REINFORCE

The adversarial REINFORCE algorithm consists of two components: a generative model G and a discriminative model D.

Generative model The generative model G defines the policy that generates a response y given dialogue history x. It takes a form similar to SEQ2SEQ models, which first map the source input to a vector representation using a recurrent net and then compute the probability of generating each token in the target using a softmax function.

Discriminative model The discriminative model D is a binary classifier that takes as input a sequence of dialogue utterances $\{x, y\}$ and outputs a label indicating whether the input is generated by humans or machines. The input dialogue is encoded into a vector representation using a hierarchical encoder (Li et al., 2015; Serban et al., 2016b),[2] which is then fed to a 2-class softmax function, returning the probability of the input dialogue episode being a machine-generated dialogue (denoted $Q_-(\{x, y\})$) or a human-generated dialogue (denoted $Q_+(\{x, y\})$).

Policy Gradient Training The key idea of the system is to encourage the generator to generate utterances that are indistinguishable from human generated dialogues. We use policy gradient methods to achieve such a goal, in which the score of current utterances being human-generated ones assigned by the discriminator (i.e., $Q_+(\{x, y\})$) is used as a reward for the generator, which is trained to maximize the expected reward of generated utterance(s) using the REINFORCE algorithm (Williams, 1992):

$$J(\theta) = \mathbb{E}_{y \sim p(y|x)}(Q_+(\{x, y\})|\theta) \tag{1}$$

Given the input dialogue history x, the bot generates a dialogue utterance y by sampling from the policy. The concatenation of the generated utterance y and the input x is fed to the discriminator. The gradient of (1) is approximated using the likelihood ratio trick (Williams, 1992; Glynn, 1990; Aleksandrov et al., 1968):

$$\begin{aligned} \nabla J(\theta) \approx & [Q_+(\{x, y\}) - b(\{x, y\})] \\ & \nabla \log \pi(y|x) \\ = & [Q_+(\{x, y\}) - b(\{x, y\})] \\ & \nabla \sum_t \log p(y_t|x, y_{1:t-1}) \end{aligned} \tag{2}$$

where π denotes the probability of the generated responses. $b(\{x, y\})$ denotes the baseline value to reduce the variance of the estimate while keeping it unbiased.[3] The discriminator is simultaneously updated with the human generated dialogue that contains dialogue history x as a positive example and the machine-generated dialogue as a negative example.

3.2 Reward for Every Generation Step (REGS)

The REINFORCE algorithm described has the disadvantage that the expectation of the reward is approximated by only one sample, and the reward associated with this sample (i.e., $[Q_+(\{x, y\}) - b(\{x, y\})]$ in Eq(2)) is used for all actions (the generation of each token) in the generated sequence. Suppose, for example, the input history is *what's your name*, the human-generated response is *I am John*, and the machine-generated response is *I don't know*. The vanilla REINFORCE model assigns the same negative reward to all tokens within the human-generated response (i.e., *I*, *don't*, *know*), whereas proper credit assignment in training would give separate rewards, most likely a neutral reward for the token *I*, and negative rewards to *don't* and *know*. We call this *reward for every generation step*, abbreviated *REGS*.

Rewards for intermediate steps or partially decoded sequences are thus necessary. Unfortunately, the discriminator is trained to assign scores to fully

[1] We approximate the dialogue history using the concatenation of two preceding utterances. We found that using more than 2 context utterances yields very tiny performance improvements for SEQ2SEQ models.

[2] To be specific, each utterance p or q is mapped to a vector representation h_p or h_q using LSTM (Hochreiter and Schmidhuber, 1997). Another LSTM is put on sentence level, mapping the context dialogue sequence to a single representation.

[3] Like Ranzato et al. (2016), we train another neural network model (the critic) to estimate the value (or future reward) of current state (i.e., the dialogue history) under the current policy π. The critic network takes as input the dialogue history, transforms it to a vector representation using a hierarchical network and maps the representation to a scalar. The network is optimized based on the mean squared loss between the estimated reward and the real reward.

generated sequences, but not partially decoded ones. We propose two strategies for computing intermediate step rewards by (1) using Monte Carlo (MC) search and (2) training a discriminator that is able to assign rewards to partially decoded sequences.

In (1) Monte Carlo search, given a partially decoded s_P, the model keeps sampling tokens from the distribution until the decoding finishes. Such a process is repeated N (set to 5) times and the N generated sequences will share a common prefix s_P. These N sequences are fed to the discriminator, the average score of which is used as a reward for the s_P. A similar strategy is adopted in Yu et al. (2016a). The downside of MC is that it requires repeating the sampling process for each prefix of each sequence and is thus significantly time-consuming.[4]

In (2), we directly train a discriminator that is able to assign rewards to both fully and partially decoded sequences. We break the generated sequences into partial sequences, namely $\{y^+_{1:t}\}_{t=1}^{N_{Y+}}$ and $\{y^-_{1:t}\}_{t=1}^{N_{Y-}}$ and use all instances in $\{y^+_{1:t}\}_{t=1}^{N_{Y+}}$ as positive examples and instances $\{y^-_{1:t}\}_{t=1}^{N_{Y-}}$ as negative examples. The problem with such a strategy is that earlier actions in a sequence are shared among multiple training examples for the discriminator (for example, token y_1^+ is contained in all partially generated sequences, which results in overfitting. To mitigate this problem, we adopt a strategy similar to when training value networks in *AlphaGo* (Silver et al., 2016), in which for each collection of subsequences of Y, we randomly sample only one example from $\{y^+_{1:t}\}_{t=1}^{N_{Y+}}$ and one example from $\{y^-_{1:t}\}_{t=1}^{N_{Y-}}$, which are treated as positive and negative examples to update the discriminator. Compared with the Monte Carlo search model, this strategy is significantly more time-effective, but comes with the weakness that the discriminator becomes less accurate after partially decoded sequences are added in as training examples. We find that the MC model performs better when training time is less of an issue.

For each partially-generated sequence $Y_t = y_{1:t}$, the discriminator gives a classification score $Q_+(x, Y_t)$. We compute the baseline $b(x, Y_t)$ using a similar model to the vanilla REINFORCE model. This yields the following gradient to update the generator:

$$\nabla J(\theta) \approx \sum_t (Q_+(x, Y_t) - b(x, Y_t)) \nabla \log p(y_t|x, Y_{1:t-1}) \quad (3)$$

Comparing (3) with (2), we can see that the values for rewards and baselines are different among generated tokens in the same response.

Teacher Forcing Practically, we find that updating the generative model only using Eq. 1 leads to unstable training for both vanilla Reinforce and REGS, with the perplexity value skyrocketing after training the model for a few hours (even when the generator is initialized using a pre-trained SEQ2SEQ model). The reason this happens is that the generative model can only be indirectly exposed to the gold-standard target sequences through the reward passed back from the discriminator, and this reward is used to promote or discourage its (the generator's) own generated sequences. Such a training strategy is fragile: once the generator (accidentally) deteriorates in some training batches and the discriminator consequently does an extremely good job in recognizing sequences from the generator, the generator immediately gets lost. It knows that its generated sequences are bad based on the rewards outputted from the discriminator, but it does not know what sequences are good and how to push itself to generate these good sequences (the odds of generating a good response from random sampling are minute, due to the vast size of the space of possible sequences). Loss of the reward signal leads to a breakdown in the training process.

To alleviate this issue and give the generator more direct access to the gold-standard targets, we propose also feeding human generated responses to the generator for model updates. The most straightforward strategy is for the discriminator to automatically assign a reward of 1 (or other positive values) to the human generated responses and for the generator to use this reward to update itself on human generated examples. This can be seen as having a teacher intervene with the generator some fraction of the time and force it to generate the true responses, an approach that is similar to the professor-forcing algorithm of Lamb et al. (2016).

A closer look reveals that this modification is the same as the standard training of SEQ2SEQ mod-

[4] Consider one target sequence with length 20, we need to sample 5*20=100 full sequences to get rewards for all intermediate steps. Training one batch with 128 examples roughly takes roughly 1 min on a single GPU, which is computationally intractable considering the size of the dialogue data we have. We thus parallelize the sampling processes, distributing jobs across 8 GPUs.

```
For number of training iterations do
.    For i=1,D-steps do
.         Sample (X,Y) from real data
.         Sample $\hat{Y} \sim G(\cdot|X)$
.           Update $D$ using $(X, Y)$ as positive examples and
$(X, \hat{Y})$ as negative examples.
.    End
.
.    For i=1,G-steps do
.         Sample (X,Y) from real data
.         Sample $\hat{Y} \sim G(\cdot|X)$
.         Compute Reward $r$ for $(X, \hat{Y})$ using $D$.
.         Update $G$ on $(X, \hat{Y})$ using reward $r$
.         Teacher-Forcing: Update $G$ on $(X, Y)$
.    End
End
```

Figure 1: A brief review of the proposed adversarial reinforcement algorithm for training the generator G and discriminator D. The reward r from the discriminator D can be computed using different strategies according to whether using REINFORCE or REGS. The update of the generator G on $(X, \hat{Y})$ can be done by either using Eq.2 or Eq.3. D-steps is set to 5 and G-steps is set to 1.

els, making the final training alternately update the SEQ2SEQ model using the adversarial objective and the MLE objective. One can think of the professor-forcing model as a regularizer to regulate the generator once it starts deviating from the training dataset.

We also propose another workaround, in which the discriminator first assigns a reward to a human generated example using its own model, and the generator then updates itself using this reward on the human generated example only if the reward is larger than the baseline value. Such a strategy has the advantage that different weights for model updates are assigned to different human generated examples (in the form of different reward values produced by the generator) and that human generated examples are always associated with nonnegative weights.

A sketch of the proposed model is shown in Figure 1.

3.3 Training Details

We first pre-train the generative model by predicting target sequences given the dialogue history. We trained a SEQ2SEQ model (Sutskever et al., 2014) with an attention mechanism (Bahdanau et al., 2015; Luong et al., 2015) on the OpenSubtitles dataset. We followed protocols recommended by Sutskever et al. (2014), such as gradient clipping, mini-batch and learning rate decay. We also pre-train the discriminator. To generate negative examples, we decode part of the training data. Half of the negative examples are generated using beam-search with mutual information reranking as described in Li et al. (2016a), and the other half is generated from sampling.

For data processing, model training and decoding (both the proposed adversarial training model and the standard SEQ2SEQ models), we employ a few strategies that improve response quality, including: (2) Remove training examples with length of responses shorter than a threshold (set to 5). We find that this significantly improves the general response quality.[5] (2) Instead of using the same learning rate for all examples, using a weighted learning rate that considers the average tf-idf score for tokens within the response. Such a strategy decreases the influence from dull and generic utterances.[6] (3) Penalizing intra-sibling ranking when doing beam search decoding to promote N-best list diversity as described in Li et al. (2016c). (4) Penalizing word types (stop words excluded) that have already been generated. Such a strategy dramatically decreases the rate of repetitive responses such as *no. no. no. no. no.* or contradictory responses such as *I don't like oranges but i like oranges*.

4 Adversarial Evaluation

In this section, we discuss strategies for successful adversarial evaluation. Note that the proposed adversarial training and adversarial evaluation are separate procedures. They are independent of each other and share no common parameters.

The idea of adversarial evaluation, first proposed by Bowman et al. (2016), is to train a discriminant function to separate generated and true sentences, in an attempt to evaluate the model's sentence generation capability. The idea has been preliminarily studied by Kannan and Vinyals (2016) in the context of dialogue generation. Adversarial evaluation also resembles the idea of the Turing test, which

[5] To compensate for the loss of short responses, one can train a separate model using short sequences.

[6] We treat each sentence as a document. Stop words are removed. Learning rates are normalized within one batch. For example, suppose t_1, t_2, ..., t_i, ... ,t_N denote the tf-idf scores for sentences within current batch and lr denotes the original learning rate. The learning rate for sentence with index i is $N \cdot lr \cdot \frac{t_i}{\sum_{i'} t_{i'}}$. To avoid exploding learning rates for sequences with extremely rare words, the tf-idf score of a sentence is capped at L times the minimum tf-idf score in the current batch. L is empirically chosen and is set to 3.

requires a human evaluator to distinguish machine-generated texts from human-generated ones. Since it is time-consuming and costly to ask a human to talk to a model and give judgements, we train a machine evaluator in place of the human evaluator to distinguish the human dialogues and machine dialogues, and we use it to measure the general quality of the generated responses.

Adversarial evaluation involves both training and testing. At training time, the evaluator is trained to label dialogues as machine-generated (negative) or human-generated (positive). At test time, the trained evaluator is evaluated on a held-out dataset. If the human-generated dialogues and machine-generated ones are indistinguishable, the model will achieve 50 percent accuracy at test time.

4.1 Adversarial Success

We define Adversarial Success (*AdverSuc* for short) to be the fraction of instances in which a model is capable of fooling the evaluator. *AdverSuc* is the difference between 1 and the accuracy achieved by the evaluator. Higher values of *AdverSuc* for a dialogue generation model are better.

4.2 Testing the Evaluator's Ability

One caveat with the adversarial evaluation methods is that they are model-dependent. We approximate the human evaluator in the Turing test with an automatic evaluator and assume that the evaluator is perfect: low accuracy of the discriminator should indicate high quality of the responses, since we interpret this to mean the generated responses are indistinguishable from the human ones. Unfortunately, there is another factor that can lead to low discriminative accuracy: a poor discriminative model. Consider a discriminator that always gives random labels or always gives the same label. Such an evaluator always yields a high *AdverSuc* value of 0.5. Bowman et al. (2016) propose two different discriminator models separately using *unigram* features and *neural* features. It is hard to tell which feature set is more reliable. The standard strategy of testing the model on a held-out development set is not suited to this case, since a model that overfits the development set is necessarily superior.

To deal with this issue, we propose setting up a few manually-invented situations to test the ability of the automatic evaluator. This is akin to setting up examinations to test the ability of the human evaluator in the Turing test. We report not only the *AdverSuc* values, but also the scores that the evaluator achieves in these manually-designed test cases, indicating how much we can trust the reported *AdverSuc*. We develop scenarios in which we know in advance how a perfect evaluator should behave, and then compare *AdverSuc* from a discriminative model with the gold-standard *AdverSuc*. Scenarios we design include:

- We use human-generated dialogues as both positive examples and negative examples. A perfect evaluator should give an *AdverSuc* of 0.5 (accuracy 50%), which is the gold-standard result.
- We use machine-generated dialogues as both positive examples and negative examples. A perfect evaluator should give an *AdverSuc* of 0.5 (accuracy 50%).
- We use original human-generated dialogues as positive examples and dialogues consisting of random utterances as negative examples. A perfect evaluator should give an *AdverSuc* of 0 (accuracy 100%).
- We use original human-generated dialogues as positive examples and dialogues with responses replaced by the utterances immediately following the true responses. A perfect evaluation should give an *AdverSuc* of 0 (accuracy 100%).

The *evaluator reliability error* (ERE) is the average deviation of an evaluator's adversarial error from the gold-standard error in the above tasks, with equal weight for each task. The smaller the error, the more reliable the evaluator is.

4.3 Machine-vs-Random Accuracy

Evaluator reliability error uses scenarios constructed from human-generated dialogues to assess feature or hyper-parameter choice for the evaluator. Unfortunately, no machine-generated responses are involved in the ERE metric. The following example illustrates the serious weakness resulting from this strategy: as will be shown in the experiment section, when inputs are decoded using greedy or beam search models, most generation systems to date yield an adversarial success less than 10 percent (evaluator accuracy 90 percent). But when using sampling for decoding, the adversarial success skyrockets to around 40 percent,[7] only 10 percent less than what's needed to pass the Turing test. A close look at the decoded sequences using sampling tells a different story: the responses from

[7] Similar results are also reported in Kannan and Vinyals (2016).

Input	tell me ... how long have you had this falling sickness ?
Vanilla-MLE	i 'm not a doctor .
Vanilla-Sample	well everything you did was totally untrue .
REINFORCE	i don 't know how long it 's been .
REGS Monte Carlo	A few months, I guess .
Input	so i had the doctors test sammy 's response to conditioning .
Vanilla-MLE	sammy wrote the test sammy wrote the test .
Vanilla-Sample	objects pick objects up objects objects objects objects objects
REINFORCE	i 'm not sure that 's relevant .
REGS Monte Carlo	so he took the pills .
Input	what are you going to do ?
Vanilla-MLE	i 'm going to the movies .
Vanilla-Sample	get him outta here first !
REINFORCE	i 'm going to get you .
REGS Monte Carlo	i 'm going to get you a new car .
Input	they fear your power your intellect .
Vanilla-MLE	you 're the only one who knows what 's going on .
Vanilla-Sample	when they are conquered and you surrender they will control all of us .
REINFORCE	i 'm afraid i 'm not ready yet .
REGS Monte Carlo	i 'm not afraid of your power .

Table 1: Sampled responses from different models. More in Appendix Tables 5 and 6.

sampling are sometimes incoherent, irrelevant or even ungrammatical.

We thus propose an additional sanity check, in which we report the accuracy of distinguishing between machine-generated responses and randomly sampled responses (*machine-vs-random* for short). This resembles the N-choose-1 metric described in Shao et al. (2017). Higher accuracy indicates that the generated responses are distinguishable from randomly sampled human responses, indicating that the generative model is not fooling the generator simply by introducing randomness. As we will show in Sec. 5, using sampling results in high *AdverSuc* values but low *machine-vs-random* accuracy.

5 Experimental Results

In this section, we detail experimental results on adversarial success and human evaluation.

Setting	ERE
SVM+Unigram	0.232
Concat Neural	0.209
Hierarchical Neural	0.193
SVM+Neural+multil-features	0.152

Table 2: ERE scores obtained by different models.

5.1 Adversarial Evaluation

ERE We first test adversarial evaluation models with different feature sets and model architectures for reliability, as measured by evaluator reliability error (ERE). We explore the following models: (1) *SVM+Unigram*: SVM using unigram features.[8] A multi-utterance dialogue (i.e., input messages and responses) is transformed to a unigram representation; (2) *Concat Neural*: a neural classification model with a softmax function that takes as input the concatenation of representations of constituent dialogues sentences; (3) *Hierarchical Neural*: a hierarchical encoder with a structure similar to the discriminator used in the reinforcement; and (4) *SVM+Neural+multi-lex-features*: a SVM model that uses the following features: unigrams, neural representations of dialogues obtained by the neural model trained using strategy (3),[9] the forward likelihood $\log p(t|s)$ and backward likelihood $p(s|t)$.

ERE scores obtained by different models are reported in Table 2. As can be seen, the *hierarchical neural* evaluator (model 3) is more reliable than simply concatenating the sentence-level representations (model 2). Using the combination of neural features and lexicalized features yields the most reliable evaluator. For the rest of this section, we report results obtained by the *Hierarchical Neural* setting due to its end-to-end nature, despite its inferiority to *SVM+Neural+multil-features*.

Table 3 presents *AdverSuc* values for different models, along with *machine-vs-random* accuracy described in Section 4.3. Higher values of *AdverSuc* and *machine-vs-random* are better.

Baselines we consider include standard SEQ2SEQ models using greedy decoding (*MLE-greedy*), beam-search (*MLE+BS*) and sampling, as well as the mutual information reranking model of Li et al. (2016a) with two algorithmic variations: (1) MMI+$p(t|s)$, in which a large N-best list is first

[8] Trained using the SVM-Light package (Joachims, 2002).

[9] The representation before the softmax layer.

Model	*AdverSuc*	*machine-vs-random*
MLE-BS	0.037	0.942
MLE-Greedy	0.049	0.945
MMI+$p(t\|s)$	0.073	0.953
MMI-$p(t)$	0.090	0.880
Sampling	0.372	0.679
Adver-Reinforce	0.080	0.945
Adver-REGS	0.098	0.952

Table 3: *AdverSuc* and *machine-vs-random* scores achieved by different training/decoding strategies.

generated using a pre-trained SEQ2SEQ model and then reranked by the backward probability $p(s|t)$ and (2) MMI$-p(t)$, in which language model probability is penalized during decoding.

Results are shown in Table 3. What first stands out is decoding using sampling (as discussed in Section 4.3), achieving a significantly higher *AdverSuc* number than all the rest models. However, this does not indicate the superiority of the sampling decoding model, since the *machine-vs-random* accuracy is at the same time significantly lower. This means that sampled responses based on SEQ2SEQ models are not only hard for an evaluator to distinguish from real human responses, but also from randomly sampled responses. A similar, though much less extreme, effect is observed for MMI$-p(t)$, which has an *AdverSuc* value slightly higher than *Adver-Reinforce*, but a significantly lower *machine-vs-random* score.

By comparing different baselines, we find that MMI+$p(t|s)$ is better than *MLE-greedy*, which is in turn better than *MLE+BS*. This result is in line with human-evaluation results from Li et al. (2016a). The two proposed adversarial algorithms achieve better performance than the baselines. We expect this to be the case, since the adversarial algorithms are trained on an objective function more similar to the evaluation metric (i.e., adversarial success). *REGS* performs slightly better than the vanilla REINFORCE algorithm.

5.2 Human Evaluation

For human evaluation, we follow protocols defined in Li et al. (2016d), employing crowdsourced judges to evaluate a random sample of 200 items. We present both an input message and the generated outputs to 3 judges and ask them to decide which of the two outputs is better (*single-turn* general quality). Ties are permitted. Identical strings are assigned the same score. We also present the judges with *multi-turn* conversations simulated between the two agents. Each conversation consists of 3 turns. Results are presented in Table 4. We observe a significant quality improvement on both single-turn quality and multi-turn quality from the proposed adversarial model. It is worth noting that the reinforcement learning system described in Li et al. (2016d), which simulates conversations between two bots and is trained based on manually designed reward functions, only improves multi-turn dialogue quality, while the model described in this paper improves both single-turn and multi-turn dialogue generation quality. This confirms that the reward adopted in adversarial training is more general, natural and effective in training dialogue systems.

Setting	adver-win	adver-lose	tie
single-turn	0.62	0.18	0.20
multi-turn	0.72	0.10	0.18

Table 4: The gain from the proposed adversarial model over the mutual information system based on pairwise human judgments.

6 Conclusion and Future Work

In this paper, drawing intuitions from the Turing test, we propose using an adversarial training approach for response generation. We cast the model in the framework of reinforcement learning and train a generator based on the signal from a discriminator to generate response sequences indistinguishable from human-generated dialogues. We observe clear performance improvements on multiple metrics from the adversarial training strategy.

The adversarial training model should theoretically benefit a variety of generation tasks in NLP. Unfortunately, in preliminary experiments applying the same training paradigm to machine translation, we did not observe a clear performance boost. We conjecture that this is because the adversarial training strategy is more beneficial to tasks in which there is a big discrepancy between the distributions of the generated sequences and the reference target sequences. In other words, the adversarial approach is more beneficial on tasks in which entropy of the targets is high. Exploring this relationship further is a focus of our future work.

Acknowledgements The authors thank Michel Galley, Bill Dolan, Chris Brockett, Jianfeng Gao and other members of the NLP group at Microsoft Research, as well as Sumit Chopra and Marc'Aurelio Ranzato from Facebook AI Research for helpful discussions and comments. Jiwei Li is supported by a Facebook Fellowship, which we gratefully acknowledge. This work is also partially supported by the NSF under award IIS-1514268, and the DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF- 15-1-0462, IIS-1464128. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, the NSF, or Facebook.

References

V. M. Aleksandrov, V. I. Sysoyev, and V. V. Shemeneva. 1968. Stochastic optimization. *Engineering Cybernetics* 5:11–16.

Nabiha Asghar, Pasca Poupart, Jiang Xin, and Hang Li. 2016. Online sequence-to-sequence reinforcement learning for open-domain conversational agents. *arXiv preprint arXiv:1612.03929* .

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. *ICLR* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. *CoNLL* .

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016a. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances In Neural Information Processing Systems*. pages 2172–2180.

Xilun Chen, Ben Athiwaratkun, Yu Sun, Kilian Weinberger, and Claire Cardie. 2016b. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614* .

Emily L Denton, Soumith Chintala, Rob Fergus, et al. 2015. Deep generative image models using a? laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*. pages 1486–1494.

Peter W Glynn. 1990. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM* 33(10):75–84.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. pages 2672–2680.

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

Thorsten Joachims. 2002. *Learning to classify text using support vector machines: Methods, theory and algorithms*. Kluwer Academic Publishers.

Anjuli Kannan and Oriol Vinyals. 2016. Adversarial evaluation of dialogue models. In *NIPS 2016 Workshop on Adversarial Training*.

Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*. pages 4601–4609.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proc. of NAACL-HLT*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany, pages 994–1003. http://www.aclweb.org/anthology/P16-1094.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *ACL* .

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016c. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562* .

Jiwei Li, Will Monroe, Alan Ritter, and Dan Jurafsky. 2016d. Deep reinforcement learning for dialogue generation. *EMNLP* .

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *EMNLP* .

Ryan Lowe, Michael Noseworthy, Iulian Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. *ACL* .

Ryan Lowe, Iulian V Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. On the evaluation of dialogue systems with next utterance classification. *SIGDIAL* .

Yi Luan, Yangfeng Ji, and Mari Ostendorf. 2016. LSTM based conversation models. *arXiv preprint arXiv:1603.09457* .

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *ACL* .

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* .

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. *ICLR* .

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of EMNLP 2011*. pages 583–593.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. pages 2226–2234.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016a. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016b. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron Courville. 2016c. Multiresolution recurrent neural networks: An application to dialogue response generation. *arXiv preprint arXiv:1606.00776* .

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2016d. Generative deep neural networks for dialogue: A short review .

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. *AAAI* .

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of ACL-IJCNLP*. pages 1577–1586.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating long and diverse responses with neural conversational models. *ICLR* .

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. *ACL* .

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Meg Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of NAACL-HLT*.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. *arxiv* .

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

Alan M Turing. 1950. Computing machinery and intelligence. *Mind* 59(236):433–460.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of ICML Deep Learning Workshop*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562* .

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *ACL* .

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2016. Incorporating loose-structured knowledge into LSTM with recall gate for conversation modeling. *arXiv preprint arXiv:1605.05110* .

Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. 2015. Attention with intention for a neural network conversation model. In *NIPS workshop on Machine Learning for Spoken Language Understanding and Interaction*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016a. Seqgan: sequence generative adversarial nets with policy gradient. *arXiv preprint arXiv:1609.05473* .

Zhou Yu, Ziyu Xu, Alan W Black, and Alex I Rudnicky. 2016b. Strategy and policy learning for non-task-oriented conversational systems. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. page 404.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *arXiv preprint arXiv:1701.00188* .

Input	maybe his eyes were too big for his stomach .
Vanilla-MLE	he 's got a big mouth .
Vanilla-Sample	i 'm sure he 's unhappy with the fact .
Reinforce	he 's a little bit of a pain .
MC	i don 't think so .
Input	they didn 't have much success with this problem commander .
Vanilla-MLE	they 're not the only ones who have been in the system .
Vanilla-Sample	it 's the new priority only .
Reinforce	they 're not the only ones who can help us .
MC	can we find someone else ?
Input	and he carried them all the time wherever he went .
Vanilla-MLE	i don 't know who he is .
Vanilla-Sample	the church ... suffered the winds of 900 meters with horse aid .
Reinforce	he was a good man .
MC	he was a very good man .
Input	somebody probably flushed their pet snake down the toilet .
Vanilla-MLE	i 'm sure the dog will be here soon .
Vanilla-Sample	i could at least think .
Reinforce	i don 't think so .
MC	i don 't think so .
Input	excuse me but how did we pay for these suits ?
Vanilla-MLE	i 'm not a beggar .
Vanilla-Sample	is this the money eugene ?
Reinforce	i don 't know what to do .
MC	how much money you owe me ?
Input	... while the beautiful and happy world was madly destroyed ;
Vanilla-MLE	i was born in the world of the world
Vanilla-Sample	as in childhood i took you to one tenth of a postcard
Reinforce	and the world was full of joy .
MC	and the world was full of joy .
Input	i was kinda hoping that you were my dad .
Vanilla-MLE	i 'm just a little bit of a little bit of a little bit of a little bit of a
Vanilla-Sample	shiny gold gold and silver in the mountains .
Reinforce	i 'm not gonna be a baby .
MC	i 'm not going to be a father .
Input	the man who does that is not afraid to die .
Vanilla-MLE	he 's not afraid of the dark .
Vanilla-Sample	even the program is the only truth
Reinforce	he 's also not afraid of me .
MC	he 's such a brave man !
Input	do you have any idea what caused the explosion ?
Vanilla-MLE	i think it 's the same thing .
Vanilla-Sample	i 'm still very nervous .
Reinforce	maybe it 's just a coincidence .
MC	i don 't know if it 's a coincidence .
Input	don 't bother the citizens in hong kong island anymore
Vanilla-MLE	i 'll be back in a few days
Vanilla-Sample	you wiii be transferred to the target court
Reinforce	i 'm not sure if you can handle it without bothering them .
MC	i 'm not going to be a nuisance
Input	you know you ought to find yourself a girl .
Vanilla-MLE	you know what i mean ?
Vanilla-Sample	it 's full of crime ...
Reinforce	i don 't know what you 're talking about .
MC	i don 't want to be a girl .
Input	i got you the big screen tv deluxe UNknown machine
Vanilla-MLE	i 'm gonna be a little late for the movie .
Vanilla-Sample	you mean you 're living ?
Reinforce	i 'm gonna get you a new one .
MC	i 'm gonna get you a new tv set .
Input	we won 't give her the chance to surprise us .
Vanilla-MLE	she 's not going to be a party member .
Vanilla-Sample	the timing 's good and bad .
Reinforce	we 'll have to get her back .
MC	i don 't think we have to worry about her .
Input	does it live in this matter we call dna ?
Vanilla-MLE	i don 't think so .
Vanilla-Sample	and it requires time to be wrong and no one believes it .
Reinforce	it 's a genetic disorder .
MC	i don 't think so .

Table 5: Appendix: Responses sampled from different models.

Input	i 'll have to drive home .
Vanilla-MLE	i 'll be back in a minute .
Vanilla-Sample	let 's go for some fun .
Reinforce	i 'll drive you home .
MC	i don 't have a car .
Input	i had to work late and then i went for a walk .
Vanilla-MLE	i was in the middle of the road .
Vanilla-Sample	my body was just pounding .
Reinforce	i was just trying to help you .
MC	i was just trying to get a little sleep .
Input	lots of cream lots of sugar .
Vanilla-MLE	i 'm gonna be in the kitchen .
Vanilla-Sample	was this gay or red ?
Reinforce	i don 't want to be a vegetable .
MC	i 'm not a very good cook .
Input	so they just let him go ?
Vanilla-MLE	he 's not gonna let them go through the motions .
Vanilla-Sample	they asked him whether we could go see ernesto ?
Reinforce	i don 't know what to do .
MC	he 's not going to be here any longer .
Input	you must not bomb this town .
Vanilla-MLE	you have to go back to the city .
Vanilla-Sample	the one is a multiple choice .
Reinforce	I think I have to .
MC	I think I have to .
Input	communism recently failed in eastern europe .
Vanilla-MLE	the war is not over .
Vanilla-Sample	there are people who hope that in tokyo they won 't risk losing their departing .
Reinforce	and the people who are not the same .
MC	the war is still going on .
Input	you should stop in more often .
Vanilla-MLE	I don 't think so .
Vanilla-Sample	is it possible as old as you ?
Reinforce	i 'm not going to be a burden on you .
MC	i don 't want to be a burden .
Input	he couldn 't have heard us .
Vanilla-MLE	he was in the middle of the night .
Vanilla-Sample	he should have been somewhere out here .
Reinforce	he 's not a good guy .
MC	he 's got a lot of nerve.
Input	that 's our wake up call .
Vanilla-MLE	we 're gonna be late for the meeting .
Vanilla-Sample	c 'mon c 'mon let 's get outta here .
Reinforce	i 'll be right back .
MC	we 're gonna have to get to the station .
Input	i had 'em when i started .
Vanilla-MLE	i 'm gonna get 'em .
Vanilla-Sample	those weak rotten guys had me in a rush on a broken swoop .
Reinforce	i don 't know what to do .
MC	i was just a little bit nervous .
Input	oatmeal delicious start into a new day
Vanilla-MLE	i 'll be right back
Vanilla-Sample	sure if you don 't put it into the water
Reinforce	i 'm gonna be a little busy with the dishes .
MC	i 'm gonna make you a little dinner .

Table 6: Appendix: More responses sampled from different models.

Using Context Information for Dialog Act Classification in DNN Framework

Yang Liu and **Kun Han** and **Zhao Tan** and **Yun Lei**
Applied Machine Learning, Facebook

Abstract

Previous work on dialog act (DA) classification has investigated different methods, such as hidden Markov models, maximum entropy, conditional random fields, graphical models, and support vector machines. A few recent studies explored using deep learning neural networks for DA classification, however, it is not clear yet what is the best method for using dialog context or DA sequential information, and how much gain it brings. This paper proposes several ways of using context information for DA classification, all in the deep learning framework. The baseline system classifies each utterance using the convolutional neural networks (CNN). Our proposed methods include using hierarchical models (recurrent neural networks (RNN) or CNN) for DA sequence tagging where the bottom layer takes the sentence CNN representation as input, concatenating predictions from the previous utterances with the CNN vector for classification, and performing sequence decoding based on the predictions from the sentence CNN model. We conduct thorough experiments and comparisons on the Switchboard corpus, demonstrate that incorporating context information significantly improves DA classification, and show that we achieve new state-of-the-art performance for this task.

1 Introduction

Dialog act (DA) represents a function of a speaker's utterance in either human-to-human or human-to-computer conversations. Correct identification of DAs is important for understanding human conversations, as well as for developing intelligent human-to-computer dialog systems (either written or spoken dialogs). For example, recognizing DAs can help identify questions and answers in meetings, customer service, online forum, etc. Many machine learning techniques have been investigated and shown reasonable performance for DA classification, for example, (Ang et al., 2005; Ji and Bilmes, 2005; Kalchbrenner and Blunsom, 2013; Ribeiro et al., 2015), just to name a few. Intuitively we would expect that leveraging dialog context can help classify the current utterance. For example, if the previous sentence is a question, then there is a high probability that the current sentence is a response to that question. Such context information has been explored in some previous methods, for example, hidden Markov models (HMM), conditional random fields (CRF), dynamic Bayesian networks (DBN). Given the recent success of the deep learning framework in various language processing tasks, in this work we also employ neural networks for DA classification. In fact, such models have been used in some recent studies for DA classification, e.g., (Rojas-Barahona et al., 2016; Kalchbrenner and Blunsom, 2013; Zhou et al., 2015); however, previous work has not thoroughly evaluated the use of context information for this task, and there is still a lack of good understanding about how we can use context information and how useful it is. This is the question we aim to answer in this work.

The contributions of this paper are: 1) We propose several ways to incorporate context information for DA classification over the baseline method of using convolutional neural networks (CNN) for sentence classification, including: (a) a hierarchical RNN/LSTM and CNN to model the utterance sequence in the conversation, where the input to the higher level LSTM and CNN unit is the sentence vector from the sentence level CNN model;

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2170–2178
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

(b) a two-step approach where the predicted DA results for the previous utterances, either labels or probability distributions, are concatenated with the sentence CNN vector for the current utterance as the new input for classification; (c) sequence level decoding based on the predicted DA probabilities and the transition probabilities between DA labels. Some of these methods have not been exploited previously for this task. 2) We perform a detailed and thorough analysis of different modeling approaches and some impacting factors in the models (such as the context length, representations and quality of the predictions). This is the first study with such kind of comparisons. 3) We achieve new state-of-the-art results.

2 Related work

Previous work has investigated different machine learning techniques for DA classification such as Maximum entropy, DBN, HMM, and SVM (Ang et al., 2005; Ji and Bilmes, 2005; Venkataraman et al., 2003; Webb et al., 2005; Fernandez and Picard, 2002; Mast et al., 1996; Liu, 2006; Kral and Cerisara, 2014). Different features have been explored in these models, including lexical, syntactic features, prosodic cues, and speaker interactions. In particular, context information has been previously used in some methods. For example, some early studies used HMMs (Venkataraman et al., 2003; Stolcke et al., 2000), where the "hidden" states are the DA tags, which generate the sequence of words as observations. The observation probabilities are obtained by DA specific word-based language models, and a DA tag based n-gram language model provides the transition probabilities between the DA tags. (Ji and Bilmes, 2005; Dielmann and Renals, 2008) used DBN for sequence decoding and examined both the generative and the conditional modeling approaches. CRF, as a powerful sequence labeling method, has also been widely used to incorporate context information for DA classification (Kim et al., 2010; Quarteroni et al., 2011; Chen and Eugenio, 2013; Dielmann and Renals, 2008). It is worth noting that (Ribeiro et al., 2015) used different configurations to capture information from previous context in the SVM classifiers, such as n-grams or DA predictions. This is similar to our work in that we also evaluate using the previous utterances, and the predicted DAs for them. However, our modeling approaches are all based on DNNs, as described in more details in Section 3, and the interaction between utterances and DA labels is modeled in the hierarchical models in a more principled way.

Recently deep learning has been widely adopted in many language processing tasks, including DA classification. Context or sequence information is also explored in this framework. For example, (Rojas-Barahona et al., 2016) proposed to use DNN for DA classification and slot filling, and evaluated on two different sets. They showed that their proposed CNN+LSTM model has negligible gain on one data set, and significant improvement on the other one for the joint DA classification and slot filling task. (Kalchbrenner and Blunsom, 2013) proposed methods for discourse decomposition, and investigated using recurrent CNN for DA classification, reporting some positive results, e.g., 2.9% improvement over the LM-HMM baseline. In this paper we propose different methods in the deep learning framework to incorporate context information. Our hierarchical LSTM and CNN method has some similarities to that used in (Rojas-Barahona et al., 2016; Kalchbrenner and Blunsom, 2013), but unlike those that focus on just one method, we propose a few approaches and perform comparisons among them for a deeper understanding of different methods and their contributing factors.

The discussions above are limited to DA classification using speech/text data. Other knowledge sources have also been used in a multimodal setting (e.g., haptic actions in (Chen and Eugenio, 2013)). In this study we just rely on textual information. Also note that in some scenarios, for example, speech conversations where transcripts are from speech recognition systems, DA segmentation is also needed. This problem has been addressed in some previous work, for example, (Lendvai, 2007; Quarteroni et al., 2011; Ang et al., 2005), which often uses a classification or sequence labeling setup for the segmentation task, or performs joint DA segmentation and classification. We use pre-segmented utterances and focus just on the DA classification task in this work.

3 DA Classification Methods

3.1 Task

Our task is to classify each utterance in a conversation into a predefined DA tag set. We use Switchboard data in our experiments (see Section 4.1 for

DA type	speaker	sentence
statement-opinion	B	I always kind of think it would be neat to be able to watch them and be there for them all the time
back-channel	A	Uh-huh
question-yes-no	B	Is that what you do?
yes-answer	A	Uh yeah
statement	A	Actually I teach my kids at home
statement	A	so I'm here all the time
summarize/reformulate	B:	Oh so they don't go to school

Table 1: An example of Switchboard conversation with the DA labels.

additional information on the data). There are different granularities of the tag sets. In this work we use 42 tags (Jurafsky et al., 1997), which has been widely used in previous studies of DA classification on this data set. Table 1 shows an example of some utterances in a Switchboard conversation. We can see that the 'answer' DA follows the 'question' one, which is quite intuitive. Our goal is thus to model such sequential information for DA classification. Again in this work we only use the transcriptions of the utterances along with the speaker information (i.e., if the current utterance is from the same or different speaker as the previous one), without any speech related features.

3.2 CNN for utterance classification

All of our methods are built based on the basic CNN sentence representation, which has been widely used recently in sentence as well as document classification (Collobert et al., 2011; Kim, 2014), therefore we first briefly describe this baseline. Figure 1 shows the context independent CNN-based classification method. Let $\mathbf{w}_{[1...n]}$ represent the word embedding sequence for a sentence with n words, where $\mathbf{w}_i \in \mathbf{R}^d$ is the d-dimensional embedding vector for the i^{th} word. A temporal convolution operation is applied to the sentence:

$$c_{[1...n]} = \tilde{\mathbf{w}}_{[1...n]} * \mathbf{f}$$

where $\tilde{\mathbf{w}}_{[1...n]}$ denotes the sequence $\mathbf{w}_{[1...n]}$ with zero padding, and f is a filter map for the convolution operation. A max pooling layer is then applied over the resulting sequence $c_{[1...n]}$ to obtain one value for the sentence. If we use l window sizes and k filters for each window, then $l \times k$ convolutional sequences are generated for each sentence, and after max pooling, we obtain a fixed-length vector $\mathbf{s}$ with a dimension of $l \times k$. This is the feature vector representation for the sentence, which is then used as the input in a multi-layer perceptron (MLP) or feedforward neural network for sentence classification. We only use one layer MLP in this work.

This baseline CNN model learns textual information in each sentence for DA classification. We can incorporate additional features into this model, for example, if the current sentence is from the same speaker as the previous one. Figure 1 shows the use of such additional features – they are concatenated with the CNN-based textural vector, and then fed to the MLP for DA classification. In the rest of the paper, when there is no confusion, we also use CNN for the cases when additional features are concatenated with the standard CNN for sentence-level representation. We use this CNN model as a baseline, and in the following will explore several methods using context information for DA classification.

3.3 Use history DA information

As discussed earlier, we expect there is valuable sequential information among the DA tags, therefore in the first approach, we combine the history DA information with the current utterance to classify its DA tag. This is represented as additional features concatenated with the CNN sentence representation, as shown in Figure 1. We evaluate different configurations in this framework.

- Use DA labels. We compare using reference and system predicted DA labels in training and testing. Note that using reference labels in testing is not a real testing setup. This is just meant to provide an upper bound and understand the performance degradation due to prediction errors.
- Use probabilities for system predictions. Instead of taking the hard decisions from the system's predictions, we evaluate using the

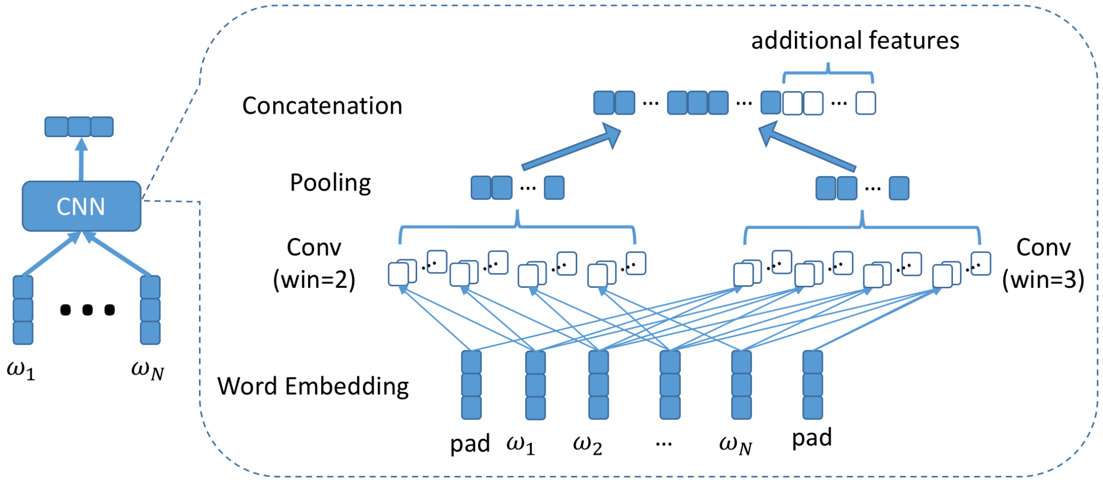


Figure 1: Baseline context-independent CNN-based DA classification method.

posterior probabilities from the system in order to capture more information.

- History length. We compare using DA information from different number of previous utterances.

Note that for most of these setups above when system's predicted DA information is used, we need to go through the following procedure:

- train a context-independent sentence CNN model
- use it to generate predictions, for training and test data
- add the corresponding history DA information in the training set to retrain a model
- add the history DA information in the test set and apply the new model

The only scenario where these steps are not required is when reference DA tags are used in both training and testing. There is one additional caveat that is worth pointing out – when generating the DA predictions for the training data, ideally we need to perform cross validation for the training set such that all the training sentences are labeled by a model trained from data that does not include this sentence, and thus we have matched information used in training and testing; however, we noticed that our model does not overfit the training data very much, and the training accuracy is not significantly different from the test accuracy, therefore we simply apply the trained CNN model to the training set itself to obtain the DA predictions for all the training sentences, and train the new model.

3.4 CNN + DA transition decoding

In this approach, we perform conversation level decoding that combines the probabilities from the context-independent CNN model and the DA tag transition probabilities. The DA classification problem can be represented as:

$$\hat{Y} = argmaxP(Y|X) = argmaxP(Y)P(X|Y)$$
$$= argmaxP(Y)\prod_i P(x_i|y_i)$$

where Y is the DA tag sequence, and X contains the entire conversation, i.e., sequence of sentences. $P(Y)$ can be computed for the DA tag sequence (similar to word-based n-gram language model, here "words" are DA tags), and the probability of a tag given the utterance ($P(x_i|y_i)$) can be obtained from the rescaled probability from the CNN model (that is $P(y_i|x_i)$). For decoding, we can use either Viterbi decoding to find the most likely DA sequence (as shown above) or forward-backward decoding to determine the best tag for each utterance in the sequence. This model is similar to the HMM model used previously for this task (Stolcke et al., 2000), and the difference is in that the probability of a DA given the sentence is estimated by the CNN model, a discrim-

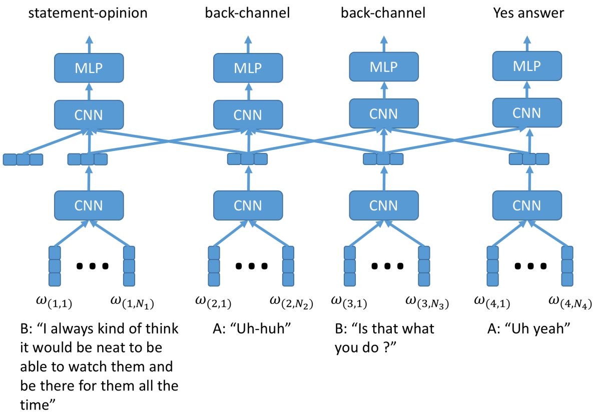


Figure 2: Hirarchical CNN: sequence CNN on top of sentence CNN for DA classification.

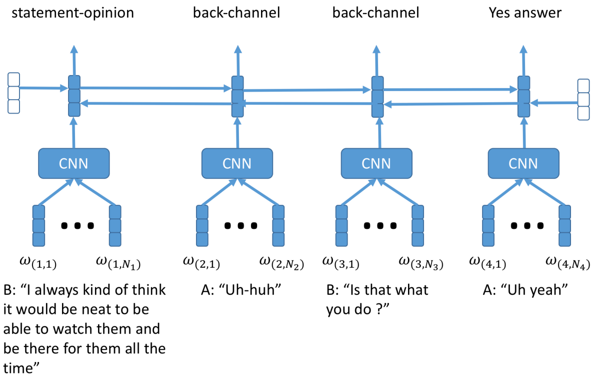


Figure 3: RNN/Bi-LSTM on top of sentence CNN for DA classification.

inative model, in contrast to the word-based language model that is a generative model.

3.5 Hierarchical model: CNN+CNN

Once we have the sentence vector representation built based on the baseline CNN model, we use another CNN to incorporate context information of an utterance for its classification. Figure 2 shows this method. The sequence of sentences is represented by a sequence of fixed length vectors $\mathbf{s}_{[1...m]}$, where m is the number of sentences in the conversation, and s_i is the vector representation for sentence i from the baseline CNN model. Similar to the CNN model for word sequence, we apply a temporal convolutional layer with different filters to $\mathbf{s}_{[1...m]}$. Different from the sentence CNN model for word sequences, here we do not perform pooling for the entire dialog sequence, as the classification task is for each sentence, not the whole conversation (sentence sequence). Instead, for each sentence, the output of every convolutional filter is concatenated to form the sentence's representation, and then an MLP is used for its classification. This approach can be thought as a hierarchical neural network, where the high level CNN is used to capture context information.

3.6 Hierarchical model: CNN+RNN

The hierarchical CNN method uses the neighboring sentences to learn the dependencies among consecutive utterances. A different method to model the sequential information is via an RNN that is intrinsically capable of learning the temporal dynamics, which is suitable for the problem. In this hierarchical model, the representation for each sentence is still learned by the CNN as in the baseline, while the dialog-level sequence information among sentences is modeled by the RNN. Here, we use bidirectional-LSTM (BLSTM) to learn the context before and after the current sentence. The left-to-right LSTM output and the one from the reverse direction are concatenated and input to a hidden layer for classification. BLSTM has been widely used recently for various sequence labeling problems (such as part-of-speech tagging, named entity recognition) and achieved state-of-the-art performance. Figure 3 shows the structure of the model. Note that the difference between these last two models and the one using history DA information is in that DA labels are not explicitly represented in these hierarchical models.

4 Experiments

4.1 Data

We use Switchboard data in our experiments. This corpus has been widely used in the community for DA classification. In this data, two people talked over the phone about a given topic for several minutes. 1155 conversations have been manually labeled with DAs. 40 conversations were held out for testing and development. However, there is no standard as to what are the test ones (it is unknown from the earliest paper using this data (Stolcke et al., 2000)). Therefore we randomly split the set into two, 20 conversations in each, with similar amount of utterances. We use one set as the development set and evaluate on the other set. As mentioned earlier, we do not use speech features, and only use textual information and speaker change feature in this study. For all the experiments, we use human transcripts. This setup is expected to be applicable to written conversations/dialogs. Table 2 shows the basic statistics of the data.

	conversations	sentences
training	1115	196,753
test set 1	20	3,764
test set 2	20	3,771

Table 2: Data information.

4.2 Results

4.2.1 Baseline CNN

For all the DNN models, we did not tune model parameters very much. Most of the parameters were chosen based on literature or our experience with other DNN-based text classification tasks. We used pretrained embeddings (dimension 200) to initialize word vectors to use in CNN, and then update them during training.[1] To avoid overfitting, we use a dropout of 0.5. The baseline CNN uses three windows: 1, 2, and 3, and 100 filter maps for each. The output hidden layer dimension is 100. For learning, we use Adagrad with a learning rate of 0.01.

Table 3 shows the baseline classification accuracy results when no context information is used, for three setups: the baseline sentence CNN model with the pretrained embeddings, when speaker change information is added, and when no pretrained embeddings are used. We can see the slight performance change because of the added speaker change feature. When no pretrained embeddings are used, i.e., no additional information is used from other resources, there is a performance degradation of 2-3%. Note that these results are better or at least comparable to state-of-the-art performance. In fact, we also implemented a CRF tagging model for this data set, where we used bag-of-word features for each utterance, therefore the information is similar to that used in the DNN framework (but the CRF does model DA tag sequential information). This CRF model has an accuracy of about 74% for the two sets combined. The CNN model without using pretrained embeddings has worse results than the CRF system that is trained just using the Switchboard data, confirming that when using word embeddings as word representations, pretrained embeddings are beneficial when the training size is small. However, the CNN model can effectively leverage word embedding information (obtained from unlabeled data), whereas it is not straightforward to use such information in the CRF classifiers. This shows an advantage of the DNN-based method.

	set 1	set 2
CNN	74.47	76.88
+ speaker change	74.73	77.12
no pretrained embedding	71.81	74.49

Table 3: DA classification accuracy (%) when using the baseline CNN without context information.

4.2.2 Hierarchical models: CNN+CNN/RNN

For the hierarchical models described in Section 3.5 and 3.6, i.e., adding CNN and BLSTM on top of the baseline sentence CNN, we kept the same model parameters in the sentence CNN part. The dimension is 64 for both the higher level CNN and LSTM. For these sequence labeling tasks, we use stochastic gradient descent (SGD), with a learning rate of 0.01. We observed this yielded better performance than Adagrad learning. Table 4 shows the results for different setups in these two models to evaluate the impact of context information. For LSTM, we compare using LSTM and BLSTM; for CNN, we show results when using different context window sizes in the top layer.

		set 1	set 2
baseline CNN		74.73	77.12
CNN+CNN	window 2	76.2	79.16
	window 3	76.78	79.05
	window 4	77.15	79.74
CNN+RNN	BLSTM	76.91	79.71
	LSTM	76.35	79.71

Table 4: DA classification results (%) when using the hierarchical structure: sentence CNN followed by dialog sequence level CNN or RNN/BLSTM.

From the table we can see that using LSTM and CNN to model context information for DA classification is effective, both models significantly outperforming the baseline. Regarding the effect of context, in general there is slightly more gain when more context is used, as in BLSTM, or larger windows in CNN. For CNN, when we increase the window more, to beyond 4, there is no further improvement. The greatest difference comes from using context vs. not using it at all.

[1] The embeddings we used are generated based on our collected web data. We compared it to other embeddings, e.g., Senna, and found the performance difference is very small.

history	DA = ref or sys		DA representation	set 1	set 2
	training	testing			
one	ref	ref	label	78.19	80.96
	ref	sys	label	75.72	77.94
	sys	sys	label	76.78	79.26
	sys	sys	probabilities	76.62	79.45
two	ref	ref	label	78.93	81.54
	sys	sys	label	76.41	79.98
	sys	sys	probabilities	76.51	80.14
three	ref	ref	label	79.62	81.76
	sys	sys	label	76.54	80.06
	sys	sys	probabilities	76.73	79.9
baseline CNN				74.73	77.12

Table 5: DA classification results (%) when incorporating history DA information in the current utterance in the CNN method. Three factors are examined: context history length, DA representations, and where DA information is from.

4.2.3 CNN + DA prediction

As described in Section 3.3, another method to incorporate context information is to use the DAs from previous utterances. We perform a detailed analysis to examine three factors under this framework:

- context history: we use a window of up to 3, i.e., information from the previous one, two, or three utterances;
- representation of the DA information, whether it is DA label or probabilities;
- reference vs. system predicted DA labels during training and testing.

Using the reference DA labels in testing is expected to give an oracle or upper bound performance for this set of experiments. Table 5 shows the results for these setups. The predictions for the utterances are generated using the baseline CNN model, with the pretrained embeddings and speaker information (i.e., the best utterance classification model). The model parameters in the second-round CNN training (when additional history DA information is included) are the same as the baseline CNN.

From Table 5 we can see that in terms of the representation of the history DA information, using hard labels and soft predictions achieves similar performance. For model training, it is better to have matched information in training and testing. Using reference DA labels during training and system predictions in testing (second row in the results) is less effective compared to using both system predictions in training and testing. The quality of the prediction also affects the usefulness of the DA prediction information, as demonstrated by the better performance when the reference labels are used compared to using system predicted DAs, which is expected. The immediate previous utterance has the largest impact on the prediction of the current utterance (comparing to not using context at all), and adding longer context helps less. In addition, using the reference previous DA labels (ref train and ref test condition) benefits more than using system predicted DA labels when longer history is used, suggesting that more predicted DAs, when used together, become more noisy and bring less gain.

4.2.4 Overall results

Table 6 summarizes the results for different systems, including the baseline CNN model without using context information (this baseline uses pretrained embeddings and speaker change feature), and four different ways of using context: (a) predicted DA information (posterior probabilities) is combined with the current sentence's CNN-based representation; (b) applying a BLSTM on top of the sentence CNN representation; (c) hierarchical CNN that combines the current sentence's CNN representation with its neighbors; (d) sequence decoding by combining CNN posteriors with DA transition scores.

From the results, we can see the positive effect

	set 1	set 2
CNN baseline, no context	74.73	77.12
CNN + DA predictions	76.73	79.9
CNN + RNN/BLSTM	76.91	79.7
CNN + CNN	77.15	79.74
CNN prob + DA transition	76.70	79.69

Table 6: DA classification results (%) using different systems.

when context information is used. All the methods using context yield significant improvement over the baseline (statistically significant based on t-test). Comparing representing context information via the DA labels of the previous utterances vs. using the hierarchical CNN or RNN model, we see there is not much difference. This observation is somewhat different from that found in (Ribeiro et al., 2015; Kim et al., 2010) where using previous DA predictions yields more gain than adding n-gram features from the previous utterances. We believe one reason for this difference is the use of the DNN framework to model the utterance sequences. Given the current data size and the oracle performance in Table 5, we expect that when more data is available, using larger neural networks will further improve the performance. Furthermore, we want to mention that overall these results represent new state-of-the-art performance for this task ((Kalchbrenner and Blunsom, 2013) reported 73.9% accuracy using recurrent CNN, though the results are not directly comparable since they only evaluated on 19 test conversations).

4.2.5 Final remarks

As expected, our experimental results demonstrate that we can effectively incorporate context information to improve DA classification. We conducted some analyses to see what errors are corrected when we use the context models compared to the baseline results. Due to space limit, we show one positive example below where adding context changes the prediction from 'backchannel' to 'answer'.

- Example:
  - Is this a mail order parts house that specializes in parts for parts for uh old imports?
  - right

It is clear that using context can help disambiguate and better predict the DAs for the current utterance. In fact, we noticed that close to 5% of errors are correctly changed from 'back channel' to 'reply' when context information is used.

One of the most frequent errors we notice the system makes is the mislabels between 'statement' and 'statement-opinion'. To correctly identify statement-opinion DAs, we can perform some opinion or subjectivity recognition, but that is out of the scope of this study. Another frequent error is the confusion between backchannel and agreement. For example, 'right' and 'yeah' are common words for both categories, and even with context information, they are still hard to disambiguate for the current models.

Finally it is worth pointing out that our work uses an offline setting where we perform DA tagging for the entire conversation. In real world applications, an online setting may be needed; however, information from previous utterances can still be used there. In fact, most of the performance gain from incorporating context information comes from the previous utterances (e.g., the difference between the hierarchical LSTM and BLSTM is very small). Our findings about the effectiveness of context information are applicable to the online setting.

5 Conclusions

We proposed several approaches to incorporate context information in the deep learning framework for DA classification in conversations, including expanding the sentence CNN vector with the predicted DA information from previous utterances to train another model, hierarchical models based on CNN or LSTM to model the DA sequence on top of the sentence CNN representation, or dialog level decoding once the sentence CNN generates its hypothesis. Compared to the baseline using CNN for utterance classification, our proposed methods effectively leverage context information and achieve significantly better performance. We observe that there is very small difference among different approaches. Our results represent the state-of-the-art for DA classification on the Switchboard data. We conducted thorough evaluations to understand the impact of different factors, and our results shed lights on the use of context information for similar tasks. In our future work, we plan to apply these approaches to other tasks, such as intent recognition and slot filling in language understanding.

Acknowledgments

The authors thank Yandi Xia for preparing the Switchboard data, Xian Qian, Antoine Raux and Benoit Dumoulin for various discussions.

References

Jeremy Ang, Yang Liu, and Elizabeth Shriberg. 2005. Automatic dialog act segmentation and classification in multiparty meetings. In *Proc. of ICASSP*.

Lin Chen and Barbara Di Eugenio. 2013. Multimodality and dialogue act classification in the robohelper project. In *Proceddings of Sigdial*.

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

Alfred Dielmann and Steve Renals. 2008. Recognition of dialogue acts in multiparty meetings using a switching dbn. *IEEE Transactions on Audio, Speech and Language Processing* 16.

Raul Fernandez and Rosalind Picard. 2002. Dialog act classification from prosodic features using support vector machines. In *Proc. of Speech Prosody*.

Gang Ji and Jeff Bilmes. 2005. Dialog act tagging using graphical models. In *Proc. of ICASSP*.

D. Jurafsky, L. Shriberg, and D. Biasca. 1997. Switchboard swbd-damsl shallow-discourse-function annotation coders manual. Technical report, University of Colorado at Boulder.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality.

Su Nam Kim, Lawrence Cavedon, and Timothy Baldwin. 2010. Classifying dialog acts in one-on-one live chats. In *Proceedings of EMNLP*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. of EMNLP*. pages 1746–1751.

Pavel Kral and Chrisophe Cerisara. 2014. Automatic dialogue act recognition with syntactic features. In *Proceedings of LREC*.

Piroska Lendvai. 2007. Token-based chunking of turn-internal dialogue act sequences.

Yang Liu. 2006. Using svm and error-correcting codes for multiclass dialog act classification in meeting corpus. In *Interspeech*.

Marion Mast, Ralf Kompe, Stefan Harbeck, Andreas Kiebling, Heinrich Niemann, and Elmar Noth. 1996. Dialog act classification with the help of prosody. In *Proc. of ICSLP*.

Silvia Quarteroni, Alexei V. Ivanov, and Giuseppe Riccardi. 2011. Simultaneous dialog segmentation and classification from human-human spoken conversations. In *Proceedings of ICASSP*.

Eugenio Ribeiro, Ricardo Ribeiro, and David Martins de Matos. 2015. The influence of context on dialogue act recognition .

Lina M. Rojas-Barahona, Milica Gasic, Nikola Mrksic, Pei-Hao Su, Stefan Ultes, Tsung-Hsien, and Steve Young. 2016. Exploiting sentence and context representation in deep neural models for spoken language understanding.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialog act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics* .

Anand Venkataraman, Lucianna Ferrer, Andreas Stolcke, and Elizabeth Shriberg. 2003. Training a prosody based dialog act tagger from unlabeled data. In *Proc. of ICASSP*.

Nick Webb, Mark Hepple, and Yorick Wilks. 2005. Dialog act classification based on intra-utterances features. In *AAAI workshop on Spoken Language Understanding*.

Yucan Zhou, Qinghua Hu, Jie Liu, and Yuan Jia. 2015. Combining heterogeneious deep neural networks with conditional random fields for chinese dialogue act recognition. *Neurocomputing* 168.

Modeling Dialogue Acts with Content Word Filtering and Speaker Preferences

Yohan Jo and **Michael Miller Yoder** and **Hyeju Jang** and **Carolyn P. Rosé**
Language Technologies Institute
Carnegie Mellon University
{yohanj,yoder,hyejuj,cprose}@cs.cmu.edu

Abstract

We present an unsupervised model of dialogue act sequences in conversation. By modeling topical themes as transitioning more slowly than dialogue acts in conversation, our model de-emphasizes content-related words in order to focus on conversational function words that signal dialogue acts. We also incorporate speaker tendencies to use some acts more than others as an additional predictor of dialogue act prevalence beyond temporal dependencies. According to the evaluation presented on two dissimilar corpora, the CNET forum and NPS Chat corpus, the effectiveness of each modeling assumption is found to vary depending on characteristics of the data. De-emphasizing content-related words yields improvement on the CNET corpus, while utilizing speaker tendencies is advantageous on the NPS corpus. The components of our model complement one another to achieve robust performance on both corpora and outperform state-of-the-art baseline models.

1 Introduction

Dialogue acts (DAs), or speech acts, represent the intention behind an utterance in conversation to achieve a conversational goal (Austin, 1975). Modeling conversations as structured DA sequences is a step toward the automated understanding of dialogue, useful for dialogue agents (Traum, 1999; Louwerse et al., 2002) and the processing of informal online conversational data (Misra and Walker, 2013; Vosoughi and Roy, 2016). Distributions of DAs can also be used as predictors of conversational outcome measures such as student learning in tutoring systems (Litman and Forbes-Riley, 2006) and engagement in meetings (Wrede and Shriberg, 2003). Unsupervised models for DA recognition may substitute or aid costly human annotation. We present an unsupervised model of DA sequences in conversation that overcomes limitations of prior models.

The first improvement our model offers is separating out content-related words to emphasize words more relevant to DAs. DAs are associated more closely with style and function words such as discourse markers and light verbs than with content words, which are more related to the propositional content (Erkens and Janssen, 2008; O'Shea et al., 2012). However, separating out content words is not standard in our field. For example, in some rule-based semantic and pragmatic parsing, the content and function of dialogue acts are not formally distinguished in the formalization (Becker et al., 2011), especially in domain-specific applications in dialogue systems (Gavaldà, 2004). A separation between content and function is useful for making cross-domain or cross-task generalizations about conversational processes.

Our model filters out content words by implementing the assumption that conversations proceed against a backdrop of underlying topics that transition more slowly than DAs or that are constant throughout. Based on a difference in transition speed, two types of language models are learned: foreground language models that capture DA-related words and background language models for content words. Although some existing models assume a background or domain-specific language model to filter out words unrelated to DAs (Lee et al., 2013; Paul, 2012; Ritter et al., 2010), they either require domain labels or do not learn topics underlying conversations.

The second improvement offered by our model is inclusion of speaker preferences, or tendencies to use some DAs more than others. Prior mod-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2179–2189
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

els of DAs in conversation often rely on the discourse property of conditional relevance (Levinson, 1983; Martin and Rose, 2003), i.e., tendencies for sequences of conversational DAs such as questions followed by answers, greetings followed by greetings, and invitations followed by acceptances (Sidnell, 2011). Though conditional relevance, which motivates the use of Markov models for inducing DA representations, is one stable signal to discover DAs in discourse data (Brychcín and Král, 2017; Lee et al., 2013), there are reasons that it is a less strong signal than ultimately desired. One of the reasons is that the DA of an utterance depends not only on the preceding DA, but also on the speaker's personal style (Appling et al., 2013) or preferences for certain DAs. Our model explicitly accounts for speaker preferences as a factor in determining the DA of an utterance.

Our model also includes additional structure to account for assumptions about distribution and packaging of observed DAs in running discourse. First, one utterance can involve more than one DA (Levinson, 1983); for example, asking a question in a forum may involve introducing the speaker, explaining the problem, etc. Hence, we assume that DAs operate on more than one level simultaneously, and an utterance-level DA is a mixture of finer-grained sentence-level DAs. Second, online conversations often have multi-level structure, branching into multiple conversational threads using replies. Our model supports conversations that have such multi-level structure.

To illustrate the generalizability of our model, we evaluate it on two corpora with very different characteristics in terms of utterance length, the number of speakers per conversation, and the domain: CNET and NPS Chat Corpus. We evaluate the DA recognition accuracy of our model and compare the result with other latest models. As we tune the model parameters for each corpus, we use our model as a lens to understand the relationship between the nature of conversations and effective model components for identifying DAs, which may inform future model design.

For the remainder of the paper, we will discuss prior work on dialogue acts and existing models (Section 2) and explain our model design (Section 3). Then we will describe our evaluation method and corpora (Section 4) and discuss the lessons learned from our empirical investigation (Section 5). We conclude the paper in Section 6.

2 Related Work

Austin (1975) makes a distinction between the illocutionary, social intention of an utterance (as seen in the indirect sentence "Can you pass the salt?") and the locutionary act of an utterance, which includes the ostensible surface-level meaning of the words. DAs are commonly thought of as describing illocutionary actions in talk. Example DAs used in computational systems include *yes-no question*, *statement*, *backchannel*, and *opinion* (Jurafsky et al., 1998).

Winograd and Flores (1986) were some of the first to conceptualize DAs with state transitions as a model for conversation. Similarly, contemporary unsupervised DA models often use a hidden Markov model (HMM) to structure a generative process of utterance sequences (Ritter et al., 2010). It is commonly assumed that each hidden state corresponds to a DA, but different approaches use different representations for states.

One common representation of a state is a multinomial distribution over words, from which words related to DAs are generated. Often, this generative process includes domain- or content-related language models that are independent of states and used to filter out words unrelated to DAs (Lee et al., 2013; Ritter et al., 2010). However, these language models have some limitations. For instance, Lee et al. (2013) rely on domain labels for learning domain-specific language models, which may require human annotation, whereas our model learns them without labels. Ritter et al. (2010) learn conversation-specific language models to filter out content words. We take a different approach, simultaneously learning content-related topics underlying the entire corpus and filtering out these content words. Although most models incorporate a general language model to separate out common words (Lee et al., 2013; Paul, 2012; Ritter et al., 2010), we do not learn it because we assume that common words are relevant to DAs.

Word embedding vector representations have also been researched as the outputs of latent states. For example, Brychcín and Král (2017) represent an utterance as a weighted sum of word vectors from GloVe[1]. Each utterance vector is generated from a Gaussian distribution that parameterizes a latent state. This model has been shown to capture

[1] https://nlp.stanford.edu/projects/glove/

DAs effectively for short utterances.

DAs are not completely determined by preceding DAs (Levinson, 1983), and this difficulty can be overcome partly by modeling speaker style, as there is evidence that each speaker has preferences for certain DAs (Appling et al., 2013). Joty et al. (2011) model speakers as outputs generated by an HMM, but this structure makes it hard to adjust the contribution of speaker preferences and may overestimate the influence of speakers. We model speaker preferences more directly such that the preceding DA and the speaker together determine an utterance's probability distribution over DAs.

One reason for the nondeterministic nature of DAs is that one utterance can involve more than one DA (Levinson, 1983); this suggests that one language model per DA may not be enough. Paul (2012) represents latent states as mixtures of topics, but there is no one-to-one relationship between states and DAs. Joty et al. (2011) assume that words are drawn individually from a fixed number of language models specific to each DA. However, we observe that one sentence usually performs a consistent finer-grained act, so we constrain each sentence in an utterance to one language model. Thus, utterances, which may consist of multiple sentences, are represented as a mixture of finer-grained sentence-level DAs.

Word order in an utterance may play an important role in determining a DA, as in the difference between "I am correct" and "am I correct". Ezen-Can and Boyer (2015) compute the similarity between utterances based on word order using a Markov random field and cluster similar utterances to identify DAs. This model, however, does not consider transitions between clusters.

Online conversations often have asynchronous, multi-level structure (e.g., nested replies). In Joty et al. (2011)'s model, individual reply structure paths from the first utterance to terminal utterances are teased apart into separate sequential conversations by duplicating utterances. However, this method counts the same utterance multiple times and requires an aggregation method for making a final decision of the DA for each utterance. We address multi-level structure without duplicating utterances.

The properties of the models explained so far are summarized in Table 1.

The relative importance of each structural component in a model may not be identical across all corpora. Differences, especially as they are attributed to meaningful contextual variables, can be interesting both practically and theoretically. One contribution of our work is considering how differences in these kinds of contextual variables lead to meaningful differences in the utility of our different modeling assumtions. More typical work in the field has emphasized methodological concerns such as minimization of parameter tuning, for example, by using a hierarchical Dirichlet process to determine the number of latent DAs automatically (Lee et al., 2013; Ritter et al., 2010) or by simply assuming that a word is equally likely to be DA-related or general (Paul, 2012). While these efforts are useful, especially when maximizing the likelihood of the data, searching for the optimal values of parameters for DA recognition may allow us to better understand the contribution of each model component depending on the characteristics of the dialogue, which in turn can inform future model design.

	Sp	Tr	LM	ML	M
Brychcín and Král (2017)	N	Y	-	N	N
Ezen-Can and Boyer (2015)	N	N	-	N	N
Lee et al. (2013)	N	Y	GD	N	N
Paul (2012)	N	Y	G	N	Y
Joty et al. (2011)	Y	Y	U	Y	Y
Ritter et al. (2010)	N	Y	GD	N	N
Our model	Y	Y	D	Y	Y

Table 1: Properties of baseline models. **(Columns)** Sp: speaker preferences, Tr: DA transitions, LM: language models unrelated to DAs (G: general background, D: domain-specific, U: unspecified), ML: multi-level structure support, M: mixture of language models for DAs.

3 Model

Our model, CSM (content word filtering and speaker preferences model), is based on an HMM combined with components for content word filtering and speaker preferences. In the model, each latent state represents an utterance-level DA as a mixture of *foreground topics*, each of which represents a sentence-level DA. Each sentence in an utterance is assigned one foreground topic. To filter content words, there is a set of *background topics* shared across conversations, and each conversation is assigned a background topic that underlies the whole conversation.

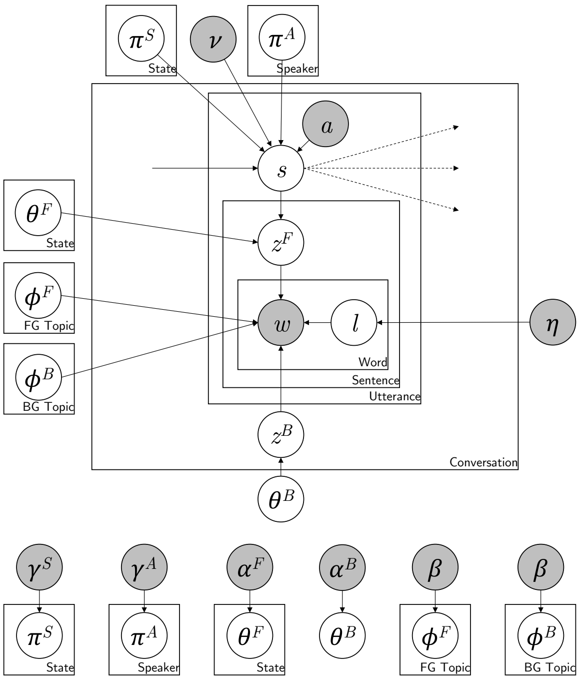


Figure 1: Graphical representation. Shaded nodes represent observable variables.

A transition between states is defined on every parent-child utterance pair, supporting multi-level structure. The state of an utterance is dependent on both its parent's state and its speaker. Speakers are specific to each conversation, i.e., a speaker participating in multiple conversations is treated as different speakers for different conversations. The graphical representation of CSM is in Figure 1.

The formal generative process of conversations is as follows:

- For each speaker a, draw a preference distribution over states $\pi_a^A \sim \text{Dir}(\gamma^A)$.
- For each state s
  - ▹ Draw a transition probability distribution over states $\pi_s^S \sim \text{Dir}(\gamma^S)$.
  - ▹ Draw a probability distribution over foreground topics $\theta_s^F \sim \text{Dir}(\alpha^F)$.
- For each foreground topic t, draw a probability distribution over words $\phi_t^F \sim \text{Dir}(\beta)$.
- For each background topic t, draw a probability distribution over words $\phi_t^B \sim \text{Dir}(\beta)$.
- For the corpus, draw a distribution over background topics $\theta^B \sim \text{Dir}(\alpha^B)$.
- For each conversation
  - ▹ Draw a background topic $z^B \sim \text{Cat}(\theta^B)$.
  - ▹ For each utterance u, with its speaker a_u, its parent p, and the parent's state s_p,
    - ♦ Draw a state $s_u \sim \text{Cat}(\nu\pi_{s_p}^S + (1-\nu)\pi_{a_u}^A)$.
    - ♦ For each sentence
      - □ Draw a foreground topic $z^F \sim \text{Cat}(\theta_{s_u}^F)$.
      - □ For each word
        - · Draw an indicator of "foreground" or "background" $l \sim \text{Cat}((\eta, 1-\eta))$.
        - · If l is "foreground", draw a word $w \sim \text{Cat}(\phi_{z^F}^F)$.
        - · If l is "background", draw a word $w \sim \text{Cat}(\phi_{z^B}^B)$.

According to this model, content words are separated out into background topics in several ways. A background topic does not transition as frequently as foreground topics within a conversation. Accordingly, words that are consistently used across utterances in a conversation are likely to be clustered into the background topic z^B, whereas words whose use is sensitive to the previous state and the speaker are likely to be clustered into foreground topics z^F. However, common function words, such as pronouns, prepositions, and punctuations, may also be separated out. Hence, η, the probability of a word being foreground, adjusts the degree of filtering. The higher the η value, the more words are likely to be generated from a foreground topic, and thus the more function words are included in foreground topics, leaving background topics with content words. Hence, we may set η high if we believe function words play an important role in DAs in a corpus and low otherwise. Note that $\eta = 0.5$ is equivalent to the assumption of existing models that a word is equally likely to be foreground or background (Lee et al., 2013; Paul, 2012). Background topics capture content words underlying the corpus, as they are shared across conversations.

Speaker preferences are captured as a probability distribution over DAs (π^A), which, along with the preceding state, affects the probability of the current state. ν adjusts the contribution of the speaker's preferences; the higher ν, the weaker the contribution. So, we may set ν low if the role or conversational style of each speaker is believed to be invariant and each speaker is expected to conduct specific DAs. If there is not enough such evidence and the conversation is driven without specific roles of the speakers, then we may set ν high. We find that corpora have different optimal values

N_{ij}^{SS}	Transition from state i to state j
N_{ij}^{AS}	Assignment of speaker i to state j
N_{ij}^{SF}	Assignment of state i to foreground topic j
N_{j}^{B}	Assignment to background topic j
N_{ij}^{FW}	Assignment of foreground topic i to word j
N_{ij}^{BW}	Assignment of background topic i to word j

Table 2: Descriptions of counter matrices.

of ν depending on the conversational characteristics.

We use collapsed Gibbs sampling for inference to integrate out π^S, π^A, θ^F, θ^B, ϕ^F, and ϕ^B. Given conversation text with speakers for each utterance, along with the hyperparameters, ν, and η, the Gibbs sampler estimates the following variables using counter matrices explained in Table 2:

$$\pi_{ij}^{S} = \frac{N_{ij}^{SS} + \gamma^S}{\sum_{j'}(N_{ij'}^{SS} + \gamma^S)}, \pi_{ij}^{A} = \frac{N_{ij}^{AS} + \gamma^A}{\sum_{j'}(N_{ij'}^{AS} + \gamma^A)}$$

$$\theta_{ij}^{F} = \frac{N_{ij}^{SF} + \alpha^F}{\sum_{j'}(N_{ij'}^{SF} + \alpha^F)}, \theta_{j}^{B} = \frac{N_{j}^{B} + \alpha^B}{\sum_{j'}(N_{j'}^{B} + \alpha^B)}$$

$$\phi_{ij}^{F} = \frac{N_{ij}^{FW} + \beta}{\sum_{j'}(N_{ij'}^{FW} + \beta)}, \phi_{ij}^{B} = \frac{N_{ij}^{BW} + \beta}{\sum_{j'}(N_{ij'}^{BW} + \beta)}.$$

We may use slice sampling (Neal, 2003) to estimate ν and η too, but the estimated values of ν and η may not be optimal for DA recognition. We can also obtain state assignments for utterances by taking a sample from the Gibbs sampler. Detailed derivation for Gibbs sampling and the code are available online[2].

4 Evaluation

This section describes our evaluation method and settings.

4.1 Task and Metrics

We evaluate our model in terms of accuracy in utterance-level DA recognition. Since the output of the model is assignments to discovered states for utterances, not pre-determined DA labels, we use a clustering evaluation method, as adopted by previous work on unsupervised DA modeling. Specifically, we use homogeneity, completeness, and v-measure as metrics (Rosenberg and Hirschberg, 2007). Homogeneity represents the degree to which utterances assigned to the same cluster by the model share the same DA in the labeled corpus. Completeness represents the degree to which utterances that have the same DA according to the gold standard are assigned to the same cluster. V-measure is the harmonic mean of homogeneity and completeness. These metrics are easy to interpret and have been demonstrated to be invariant to dataset size and number of clusters. This enables a meaningful comparison of accuracy across different corpora.

	CNET	NPS
# conversations	310	15
# utterances	1,332	10,567
# DAs	12	15
# domains	24	-
Median # utterances/conversation	3	706
Median # words/utterance	51	2
Median # speakers/conversation	2	94

Table 3: Corpora statistics.

CNET	NPS
Question-Question	Accept
Question-Add	Bye
Question-Confirmation	Clarify
Question-Correction	Continuer
Answer-Answer	Emotion
Answer-Add	Emphasis
Answer-Confirmation	Greet
Answer-Correction	Reject
Answer-Objection	Statement
Resolution	System
Reproduction	yAnswer
Other	nAnswer
	whQuestion
	ynQuestion
	Other

Table 4: Dialogue act tags in the corpora.

4.2 Corpora and Preprocessing

We evaluate on two corpora: CNET and NPS Chat (see Table 3 for statistics).

CNET (Kim et al., 2010) is a set of post threads from the Operating System, Software, Hardware, and Web Development sub-forums of CNET. This corpus is tagged with 12 DAs, including *Question-Question*, *Question-Confirmation*, *Answer-Add*, *Resolution*, and *Other* (Table 4). Note that question- and answer-related DAs are two-level. Most posts are tagged with one DA; in case a post is tagged with multiple DAs, we choose the first DA in the meta-data[3]. Each post is considered an

[2] https://github.com/yohanjo/Dialogue-Acts

[3] Some tagging systems, such as the DAMSL-style, break down an utterance that has multiple DAs.

utterance and each thread as a conversation. Each thread has only a few posts (median 3) and involves a few speakers (median 2). Since there are many URLs, email addresses, and numbers in text, we replace them with special tokens using regular expressions, and tokenize with the Stanford PTBTokenizer included in Stanford Parser 3.7.0[4].

NPS Chat (Forsythand and Martell, 2007) is a set of conversations from various online chat services. This corpus is tagged with 15 DAs, including *Emotion*, *System*, and *whQuestion* (Table 4). Every turn is tagged with a DA and considered an utterance. Each conversation is long (median 706 utterances) and involves many speakers (median 94). This corpus has already been tokenized, so we only replace usernames with a special token. Conversations in NPS have no reply structure, but we build in multi-level structure, simply treating an utterance that mentions another user as a child of the nearest utterance of the mentioned user. We compare the DA accuracy of the multi-level structure and the original linear structure in Section 5.

4.3 Models and Parameters

We set the numbers of states and background topics to the numbers of DAs and domains, respectively, if these numbers are available. For NPS, we search for the optimal number of background topics between 1 and 2, because there are only a few conversations. The optimal number of foreground topics is chosen among multiples of five between the number of states and four times the number of states, and the weights for state transition (ν) and foreground topics (η) are chosen among multiples of 0.1. For Dirichlet hyperparameters, we use $\alpha^F = 0.1, \gamma^A = 0.1, \beta = 0.001$ to induce sparsity, and $\gamma^S = 1, \alpha^B = 1$ for the uniform distribution over all configurations.

We randomly split each corpus into five groups and use three groups for training, one for parameter tuning, and one for testing. We run 5-fold cross-validation and report the average optimal parameter values and accuracy across the folds. The number of sampling iterations was chosen such that the log-likelihood of the data has converged. For each fold, we take 10 samples during inference on the test data with interval of 10 iterations and compute the mean and standard deviation of the 50 samples from all folds.

We compare our model with the three most recent unsupervised models we surveyed. The baseline models and settings are as follows.

Gaussian mixture HMM (Brychcín and Král, 2017), based on an HMM, has a characteristic output representation: utterance vectors. These vectors are generated from Gaussian distributions instead of using language models as in most existing models. After following their same preprocessing steps, we trained a model on the training data, chose the optimal word vector dimensionality on the validation data (among 50, 100, 200, and 300, as used in the original model), and performed inference on the test data. We used the original source code from the authors for training and modified the code for inference.

MRF-based clustering (Ezen-Can and Boyer, 2015) considers word order within an utterance to calculate similarity between utterances using an MRF. Then k-medoids clustering is conducted based on the similarity scores, resulting in clusters that represent DAs. The similarity score between two utterances is asymmetric, so we took the average value of each direction and inversed it to obtain the distance between two utterances. We trained a model on the training data, chose the optimal parameter values ($\lambda_i, \lambda_t, \alpha_d$ in the original paper) on the validation data, and assigned clusters to the test data. We implemented the algorithm since the original code was not available.

HDP-HMM (Lee et al., 2013) is based on an HMM, and each word comes from either the state-specific, general background, or domain-specific language model. HDP-HMM automatically decides the number of states using a hierarchical Dirichlet process, but we manually set the number of DAs in our experiment, assuming that we know the number of DAs of interest. We trained a model on the training data and performed inference on the test data; the validation data was not used since there are no parameters to tune. We used the original source code from the authors for training and modified the code for inference.

5 Results

Accuracy of DA recognition in terms of homogeneity, completeness, and v-measure on both corpora is summarized in Table 5. We also tested the following configurations:

- **CSM + Domain** uses true domain labels when learning background topics by force-

[4] https://nlp.stanford.edu/software/lex-parser.html

	CNET			NPS		
Model	H	C	V	H	C	V
Brychcín and Král (2017)	.13± .00	.09± .00	.10± .00	.24± .10	**.33± .06**	.28± .08
Ezen-Can and Boyer (2015)	.03± .00	.37± .00	.05± .00	.26± .00	.33± .00	.28± .00
Lee et al. (2013)	.09± .03	.16± .03	.11± .03	**.36± .02**	.28± .02	.31± .02
CSM	.24± .03	**.38± .04**	**.29± .03**	.35± .04	.31± .04	**.33± .04**
CSM + Domain	**.27± .02**	.33± .11	.29± .05		N/A	
CSM - Speaker	.24± .03	**.38± .04**	**.29± .03**	.21± .03	.19± .05	.20± .04
CSM - Multi-level	.23± .04	.33± .06	.27± .04	.35± .02	.30± .04	.32± .03
CSM - Background Topics	.15± .03	.11± .02	.12± .02	.35± .04	.31± .04	**.33± .04**

Table 5: Accuracy of DA recognition (the higher the better). Smaller numbers are population standard deviations. **(Columns)** H: homogeneity, C: completeness, V: v-measure. Optimal parameter values for CSM: # foreground topics=34, $\eta = .86$, $\nu = 1.00$ for CNET and # foreground topics=35, $\eta = 1.00$, $\nu = 0.58$ for NPS.

fully assigning a conversation the background topic corresponding to the true label.

- **CSM - Speaker** does not use speaker preferences by setting $\nu = 1$.
- **CSM - Multi-level** ignores multi-level structure; that is, utterances in each conversation are ordered by time.
- **CSM - Background Topics** uses only one background topic.

Overall, our model performs significantly better than the baselines for CNET and marginally better for NPS. The baseline models show a large variance in performance depending on the characteristics of the corpus. In contrast, our model has a low variance between the corpora, because the content word filtering, distinction between utterance-level and sentence-level DAs, and speaker preferences complement one another to adapt to different corpora. For example, content word filtering and DA level distinction play more significant roles than speaker preferences on CNET, whereas their effects are reversed on NPS. The details will be described later with qualitative analyses.

There may be several reasons for the poor performance of the baseline models on CNET. First, in our model, each utterance-level DA (latent state) is a probability distribution over sentence-level DAs (foreground topics), which better captures multiple sentence-level DAs in long utterances as in CNET. The utterances in CNET are long and may be too complex for the baseline models, which use a simpler representation for utterance-level DAs. Another reason for the low performance could be that the baseline models do not filter out content words as our model does.

BT0	drive partition drives partitions c
BT1	router wireless network connected connection
BT2	vista camera canon windows scanner
BT3	drive ipod touch data recovery
BT4	speakers firewall sound still no
BT5	/ \blaster dos drive
BT6	windows cd i xp boot
BT7	page xp sp3 ! content
BT8	ram mhz 1gb 512mb screen
BT9	his rupesh to company he
BT10	xp drive drivers new hard
BT11	tv port cpu motherboard grounded
BT12	file files copy external mac
BT13	“ password flash ##NUMBER## ?
BT14	fan fans cpu case air
BT15	ram card 2.4 graphics nvidia
BT16	registry file shutdown machines screen
BT17	div site % ie6 firefox
BT18	printer sound would card contact
BT19	hosting web hostgator they host
BT20	ubuntu linux memory boot reader
BT21	mac compression archive format trash
BT22	bluetooth router wireless laptop 802.11
BT23	email address account mail bounce

Table 6: Background topics learned from CNET. **(Columns)** Left: topic index, right: top 5 words.

In the remainder of this section, we describe our qualitative analysis on the results. All examples shown in the analysis are from the result with the optimal parameter values for the first fold.

Filtering content words Our model effectively separates content words from DA-related words without using the domain label of each conversation. As an example, the background topics learned by our model from CNET are shown in Table 6. These topics are clearly related to the subjects of the forum, rather than reflecting DAs, and the topics are distinctive from one another and cohesive in themselves.

The main purpose of learning background topics is to filter out content words and retain DA-related words as foreground. The learned background topics serve this purpose well, as these topics increase v-measure by 0.17 (CSM vs. CSM - Background Topics). It is also promising that the background topics learned without domain labels perform as well as when they are learned with domain labels (CSM vs. CSM + Domain), because domain labels may not always be available.

Function words play an important role in DAs in CNET as indicated by the high optimal value of $\eta = 0.86$ (the probability of a word being foreground). The higher η means more function words are included in foreground topics, leaving background topics with content words (Section 3). The high η is evidence contrary to the common practice of designating a general background topic to filter out common words and assuming that a word is equally likely to be foreground or background (Lee et al., 2013; Paul, 2012).

The effectiveness of our method of separating background topics turns out to diminish when there are no consistent conversational topics within and across conversations as in NPS. Our model learns not to use background topics ($\eta = 1$) for NPS, because background topics may filter out function words and DA-related words that occur more consistently throughout a conversation than content words do.

Environments (FT20)	. i a ##NUMBER## and have -rrb- xp -lrb- : windows my is the dell vista
Error msgs (FT12)	. the # * messages / : it log
Asking (FT19)	any help you ? ! . appreciated i suggestions
Thanking (FT17)	thanks . for the ! in advance help your all response
Problem (FT8)	: \file is the c corrupted following missing or error
Wishes (FT14)	. bob good luck
Reference (FT5)	##URL##
Praise (FT1)	. thank you ˜ sovereign , and are excellent recommendations
Explanation (FT10)	the . to , i and a it you is that of

(a) Foreground topics learned from CNET.

Wh question (FT7)	##USERNAME## ? how you are u good is round where who . ??
Wh question (FT27)	##USERNAME## ? you i u what how , ok 'm for up do have
YN question (FT1)	chat any wanna / me pm to ? anyone f guys m want here
Greeting (FT5)	##USERNAME## hi hey :) hello wb ! ... hiya ty
Laughing (FT0)	##USERNAME## lol lmao yes ! hey up !!!! ?
Laughing (FT12)	lol ##USERNAME## haha ! brb omg nite hiyas hb :p !!! . ha lmfao
Emotion (FT30)	ok ! im lol my its in " ... oh always
System logs (FT25)	part join

(b) Foreground topics learned from NPS.

Table 7: Foreground topics learned from the corpora. **(Columns)** Left: interpretation (topic index), right: top words truncated for clarity.

Mixture of foreground topics As a consequence of filtering out content words, the foreground topics reflect various acts in conversation. Some of the learned foreground topics from CNET are shown in Table 7a. These topics capture important sentence-level DAs that constitute utterance-level DAs that are assigned to each post in CNET. For example, *Question-Question* is an utterance-level DA that often starts a conversation, and conducting this DA typically includes multiple finer-grained acts, such as explaining the environment and situation, asking a question, and thanking, as shown in the post:

> I am currently running Windows XP Media Edition on a 500G hard drive. (FT20) / I want to move my XP to it's own partition, move all of my files(music, games, work) to another, and then install the Windows 7 beta on another partition. (FT10) / I don't know if this is possible or not, but I have access to Partition Magic 8, and am wondering if I can do it with that or not. (FT10) / I am not worried about installing 7 on another partition, but am not sure if I can move my files onto a separate one while keeping XP intact. (FT10) / Any help is great, thank you. (FT17)

Likewise, the *Answer-Answer* DA includes finer acts such as wishes or URLs, as in the posts:

> Simple - Download and install the Vista Rebel XT drivers from canon usa.com. (FT10) / Once installed...........go to camera menu and switch the communication to Print/PTP. (FT10) / Don't forget to switch it back if you're connecting to an XP machine. (FT10) / Good Luck (FT14)

> http://forums.microsoft.com/MSDN/ShowPost.aspx?PostID=1996406&amp;SiteID=1 (FT5)

When a problem is resolved, the *Resolution* DA may be performed with thanking and praising:

> Excellent summary Thank you. (FT1) / Sounds like at some point it's worth us making the transition to a CMS... (FT10)

FT10 covers explanations and statements, as well as long sentences. The distinction between two levels DAs is effective for CNET, as our model beats the baselines significantly.

The foreground topics learned from NPS also reflect DAs in the corpus (Table 7b). The distinction between utterance-level and sentence-level DAs is not beneficial for NPS because each utterance is short and usually conducts only one DA. As a consequence, the model has difficulty grouping foreground topics (i.e., sentence-level DAs) that are related to one another into the same utterance-level DAs (i.e., states); for CNET, on the other hand, foreground topics that co-occur in the same utterance tend to cluster to the same state.

The DAs of some foreground topics not shown in Table 7 are difficult to interpret, and those topics possibly capture aspects of sentences other than DAs. However, they do not have undue influence in our model.

Speaker preferences Speaker preferences substantially increase the v-measure by 0.13 for NPS (CSM vs. CSM - Speaker). Notably, speaker preferences complement the mixture of sentence-level DAs, which is not good at clustering related sentence-level DAs into the same utterance-level DA for short utterances. More specifically, each speaker is modeled to have sparse preferences for utterance-level DAs (i.e., states), so foreground topics used by the same speaker, often representing the same utterance-level DA, tend to cluster to the same state.

Speaker preferences also capture the characteristic styles of some speakers. Among speakers who are found to have sparse preferences by our model, some actively express reactions and often mark laughter (FT12). Others frequently agree (FT0), greet everyone (FT5), or have many questions (FT7, FT27). Accordingly, the model finds a relatively high optimal weight for speaker preferences in NPS ($\nu = 0.58$).

In contrast, CNET benefits little from speaker preferences ($\nu = 1$), partly because there is not enough information about each speaker in such short conversations. Speakers also show little preference for DAs as defined in the corpus. For instance, while a conversation initiator tends to ask questions in successive posts, these questions are annotated as different DAs (e.g., *Question-Question*, *Question-Add*, *Question-Confirmation*, etc.) depending on the position of the post within the conversation.

Multi-level structure Our model's ability to account for multi-level structure improves the accuracy of DA recognition for both corpora (CSM vs. CSM - Multi-level). For NPS, where multi-level structure is not explicit, this improvement comes from simple heuristics for inferring multi-level structure based on user mentions.

Sentence length and foreground topics In our model, all words in the same sentence are assigned to the same foreground topic, just as many existing models assign one utterance one topic. Topic assignment is based on similarity of words in a sentence to other sentences in that topic, and short sentences often find similar sentences more easily than long sentences do. Therefore, learned topics tend to be characteristic of short sentences that are similar enough to form the separate topics, and as a result, long sentences may be assigned the same topic regardless of the DA actually performed.

6 Conclusion

We have presented an unsupervised model of DAs in conversation that separates out content words to better capture DA-related words and that incorporates speaker preferences. Our model also uses a mixture of sentence-level DAs for utterance-level DAs and supports multi-level thread structure. We find that different characteristics of conversation require different modeling assumptions for DA recognition. Unlike the baseline models, which show a large variance in performance across corpora, our model is robust for both corpora used in the evaluation due to the model components complementing one another. Specifically, content word filtering is found to be effective when each conversation has a consistent conversational topic, and the separation between sentence-level and utterance-level DAs is beneficial for long utterances. Speaker preferences are found to be helpful when speakers have characteristic styles of conversation. These findings in addition to the fact that many function words are not filtered out as background may help inform future model design.

Acknowledgments

This research was supported by the Kwanjeong Educational Foundation, NIH grant R01HL122639, and NSF grant IIS-1546393.

References

D Scott Appling, Erica J Briscoe, Heather Hayes, and Rudolph L Mappus. 2013. Towards automated personality identification using speech acts. In *AAAI Workshop - Technical Report*.

John L Austin. 1975. *How to Do Things with Words*. Harvard University Press.

Lee Becker, Wayne H Ward, Sarel van Vuuren, and Martha Palmer. 2011. DISCUSS: a dialogue move taxonomy layered over semantic representations. *Proceedings of the Ninth International Conference on Computational Semantics*, pages 310–314.

Tomáš Brychcín and Pavel Král. 2017. Unsupervised dialogue act induction using gaussian mixtures. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 485–490, Valencia, Spain. Association for Computational Linguistics.

Gijsbert Erkens and Jeroen Janssen. 2008. Automatic coding of dialogue acts in collaboration protocols. *International Journal of Computer-Supported Collaborative Learning*, 3(4):447–470.

Aysu Ezen-Can and Kristy Elizabeth Boyer. 2015. Understanding Student Language: An Unsupervised Dialogue Act Classification Approach. *JEDM - Journal of Educational Data Mining*, 7(1):51–78.

Eric N Forsythand and Craig H Martell. 2007. Lexical and Discourse Analysis of Online Chat Dialog. In *International Conference on Semantic Computing (ICSC 2007)*, pages 19–26.

Marsal Gavaldà. 2004. *Soup: A Parser for Real-World Spontaneous Speech*. Springer Netherlands, Dordrecht.

Shafiq Joty, Giuseppe Carenini, and Chin-Yew Lin. 2011. Unsupervised Modeling of Dialog Acts in Asynchronous Conversations. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, pages 1807–1813. AAAI Press.

Daniel Jurafsky, Rebecca Bates, N. Coccaro, R. Martin, M. Meteer, K. Ries, E. Shriberg, A. Stolcke, Paul Taylor, and C. Van Ess-Dykema. 1998. Automatic detection of discourse structure for speech recognition and understanding. *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 88–95.

Su Nam Kim, Li Wang, and Timothy Baldwin. 2010. Tagging and Linking Web Forum Posts. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 192–202, Uppsala, Sweden. Association for Computational Linguistics.

Donghyeon Lee, Minwoo Jeong, Kyungduk Kim, Seonghan Ryu, and Gary Geunbae Lee. 2013. Unsupervised Spoken Language Understanding for a Multi-Domain Dialog System. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(11):2451–2464.

Stephen C Levinson. 1983. Conversational structure. In *Pragmatics*, chapter 6, pages 284–333. Cambridge University Press.

Diane J Litman and Katherine Forbes-Riley. 2006. Correlations between dialogue acts and learning in spoken tutoring dialogues. *Natural Language Engineering*, 12(02):161–176.

Max Louwerse, Art Graesser, Andrew Olney, and the Tutoring Research Group. 2002. Good Computational Manners: Mixed-Initiative Dialog in Conversational Agents. *Etiquette for Human-Computer Work: Papers from the AAAI Fall Symposium*, pages 71–76.

J R Martin and David Rose. 2003. *Negotiation: interacting in dialogue*. New Century Series. Bloomsbury Academic.

Amita Misra and Marilyn Walker. 2013. Topic Independent Identification of Agreement and Disagreement in Social Media Dialogue. *Proceedings of the SIGDIAL 2013 Conference*, (August):41–50.

Radford M Neal. 2003. Slice sampling. *The Annals of Statistics*, 31(3):705–767.

James O'Shea, Zuhair Bandar, and Keeley Crockett. 2012. *A Multi-classifier Approach to Dialogue Act Classification Using Function Words*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Michael J. Paul. 2012. Mixed membership markov models for unsupervised conversation modeling. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 94–104, Jeju Island, Korea. Association for Computational Linguistics.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised Modeling of Twitter Conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180, Los Angeles, California. Association for Computational Linguistics.

Andrew Rosenberg and Julia Hirschberg. 2007. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic. Association for Computational Linguistics.

Jack Sidnell. 2011. *Conversation Analysis: An Introduction*. Language in Society. Wiley.

David R. Traum. 1999. *Speech Acts for Dialogue Agents*. Springer Netherlands, Dordrecht.

Soroush Vosoughi and Deb Roy. 2016. Tweet Acts : A Speech Act Classifier for Twitter. *Proceedings of the 10th AAAI Conference on Weblogs and Social Media*, (ICWSM):1–4.

Terry Winograd and Fernando Flores. 1986. *Understanding Computers and Cognition: A New Foundation for Design*. Language and being. Ablex Publishing Corporation.

Britta Wrede and Elizabeth Shriberg. 2003. Relationship between dialogue acts and hot spots in meetings. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 180–185. IEEE.

Towards Implicit Content-Introducing for Generative Short-Text Conversation Systems

Lili Yao[1], **Yaoyuan Zhang**[1], **Yansong Feng**[1], **Dongyan Zhao**[1,2] and **Rui Yan**[1,2] *
[1]Institute of Computer Science and Technology, Peking University, Beijing, China
[2]Beijing Institute of Big Data Research, Beijing, China
{yaolili,zhang_yaoyuan,fengyansong,zhaody,ruiyan}@pku.edu.cn

Abstract

The study on human-computer conversation systems is a hot research topic nowadays. One of the prevailing methods to build the system is using the generative Sequence-to-Sequence (Seq2Seq) model through neural networks. However, the standard Seq2Seq model is prone to generate trivial responses. In this paper, we aim to generate a more meaningful and informative reply when answering a given question. We propose an implicit content-introducing method which incorporates additional information into the Seq2Seq model in a flexible way. Specifically, we fuse the general decoding and the auxiliary cue word information through our proposed hierarchical gated fusion unit. Experiments on real-life data demonstrate that our model consistently outperforms a set of competitive baselines in terms of BLEU scores and human evaluation.

1 Introduction

To establish a conversation system with adequate artificial intelligence is a long-cherished goal for researchers and practitioners. In particular, automatic conversation systems in open domains are attracting increasing attention due to its wide applications, such as virtual assistants and chatbots. In open domains, researchers mainly focus on data-driven approaches, since the diversity and uncertainty make it impossible to prepare the interaction logic and domain knowledge. Basically, there are two mainstream ways to build an open-domain conversation system: 1) to search pre-established database for candidate responses by query retrieval (Isbell et al., 2000; Wang et al., 2013; Yan et al., 2016; Song et al., 2016), and 2) to generate a new, tailored utterance given the user-issued query (Shang et al., 2015; Vinyals and Le, 2015; Serban et al., 2016; Mou et al., 2016; Song et al., 2016). In these studies, generation-based conversation systems have shown impressive potential. Especially, the Sequence-to-Sequence (Seq2Seq) model (Sutskever et al., 2014) based on neural networks has been extensively used in practice; the idea is to encode a query as a vector and to decode the vector into a reply. Inspired by (Mou et al., 2016), we mainly focus on the generative short-text conversation without context information.

Despite this, the performance of Seq2Seq generation-based conversation systems is far from satisfactory because its generation process is not controllable; it responses to a query according to the pattern learned from the training corpus. As a result, the system is likely to generate an unexpected reply even with little semantics, e.g, “I don’t know” and “Okay” due to the high frequency of these patterns in training data (Li et al., 2016a; Mou et al., 2016). To address this issue, Li et al. (2016a) proposed to increase diversity in the Seq2Seq model so that more informative utterances have a chance to stand out. Mou et al. (2016) provided a content-introducing approach that generates a reply based on a predicted word. The word is usually enlightening and drives the generated response to be more meaningful. However, this method is to some extent rigid; it requires the predicted word to explicitly occur in the generated utterance. As shown in Table 1, sometimes, it is better to generate a semantic related sentence based on the cue word rather than including it in the reply directly.

As for such content-introducing method, there are two aspects that need to be taken into consid-

[*] Corresponding author: ruiyan@pku.edu.cn

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2190–2199
Copenhagen, Denmark, September 7–11, 2017.

Query	你不觉得好丑吗(Don't you think it is ugly?)
Cue Word	审美(Aesthetics)
Reply	好恶心啊! (It's disgusting!)
Query	先放个大招(Let me use my ultimate power.)
Cue Word	技能(Skill)
Reply	新技能？(New skill?)

Table 1: The content-introducing conversation examples.

eration. 1) How to add the additional cue words during the generation process? One of the prevailing methods is modifying the neural cell with various gating mechanisms (Wen et al., 2015a,b; Xu et al., 2016). However, we need careful operation to ensure the neuron works as expected. 2) How to display the cue words in replies? As mentioned above, the explicit content-introducing approach in (Mou et al., 2016) does not fit well with all situations.

In this paper, we present an implicit content-introducing method for generative conversation systems, which incorporates cue words using our proposed hierarchical gated fusion unit (HGFU) in a flexible way. Our main contributions are as follows:

- We propose the cue word GRU, another neural cell, to deal with the auxiliary information. Compared with other gating methods, our cue word GRU is more flexible.
- We focus on the implicit content-introducing method during generation: the information of the cue word will be fused into the generation process but not necessarily occur explicitly. In this way, we change the "hard" content-introducing method into a new "soft" schema.

The rest of paper is organized as follows. We start by introducing the technical background. In Section 3, we describe our proposed method. In Section 4, we illustrate the experimental setup and evaluations against a variety of baselines. Section 5 briefly reviews related work. Finally, we conclude our paper in Section 6.

2 Technical Background

2.1 Seq2Seq Model and Attention Mechanism

Seq2Seq model was first introduced in statistical machine translation; the idea is to encode a source sentence as a vector by a recurrent neural network (RNN) and to decode the vector to a target sentence by another RNN. Now, the conversational generation is treated as a monolingual translation task (Ritter et al., 2011; Shang et al., 2015). Given a query $Q = (x_1, ..., x_n)$, the encoder represents it as a context vector C and then the decoder generates a response $R = (y_1, ..., y_m)$ word by word by maximizing the generation probability of R conditioned on Q. The objective function of Seq2Seq can be written as:

$$\begin{aligned} & p(y_1, ..., y_m | x_1, ..., x_n) \\ =& p(y_1|C) \prod_{t=2}^{T} p(y_t|C, y_1, ..., y_{t-1}) \end{aligned} \tag{1}$$

To be specific, the encoder RNN calculates the context vector by:

$$h_t = f(x_t, h_{t-1}); C = h_T \tag{2}$$

where h_t is the hidden state of encoder RNN at time t and f is a non-linear transformation which can be a long-short term memory unit (LSTM) (Hochreiter and Schmidhuber, 1997) or a gated recurrent unit (GRU) (Cho et al., 2014). In this work, we implement f using GRU.

The decoder RNN generates each reply word conditioned on the context vector C. The probability distribution p_t of candidate words at every time step t is calculated as:

$$s_t = f(y_{t-1}, s_{t-1}, C); p_t = \mathit{softmax}(s_t, y_{t-1}) \tag{3}$$

where s_t is the hidden state of decoder RNN at time t and y_{t-1} is the generated word in the reply at time $t-1$.

Attention mechanisms (Bahdanau et al., 2014) have been proved effective to improve the generation quality. In Seq2Seq with attention, each y_i corresponds to a context vector C_i; it is weighted average of all hidden states of the encoder. Formally, C_i is defined as $C_i = \sum_{j=1}^{T} \alpha_{ij} h_j$, where α_{ij} is given by:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T} exp(e_{ik})}; e_{ij} = \eta(s_{i-1}, h_j) \tag{4}$$

where η is usually implemented as a multi-layer perceptron (MLP) with tanh as an activation function.

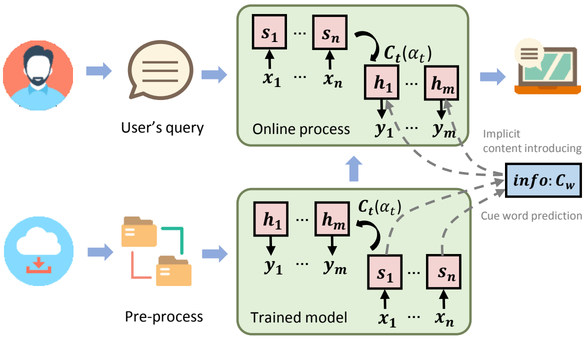


Figure 1: The architecture of our system. Based on the constructed corpus, we train our implicit content-introducing conversation system. Given a user-issued query, we first predict the cue word. Then, we incorporate the cue word into decoding process to generate a meaningful response.

2.2 Pointwise Mutual Information

Pointwise mutual information (PMI) (Church and Hanks, 1990) is a measure of association ratio based on the information theoretic concept of mutual information. Given a pair of outcomes x and y belonging to discrete random variables $\mathcal{X}$ and $\mathcal{Y}$, the PMI quantifies the discrepancy between the probability of their coincidence based on their joint distribution and their individual distributions. Mathematically:

$$\mathrm{PMI}(x,y) = \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} \tag{5}$$

This quantity is zero if x and y are independent, positive if they are positively correlated, and negative if they are negatively correlated.

3 Implicit Content-Introducing Conversation System

Figure 1 provides an overview of our system architecture. We crawl conversational data from social media which are publicly available. After filtering and cleaning procedures, we establish the conversational parallel dataset, which consists of a large number of aligned $\langle query - reply \rangle$ pairs. Based on the entire set, we first predict the cue word for the given query in Subsection 3.1. Next, we propose the new implicit content-introducing process, which explores when to incorporate the predicted cue word in Subsection 3.2 and how to apply such information in Subsection 3.3.

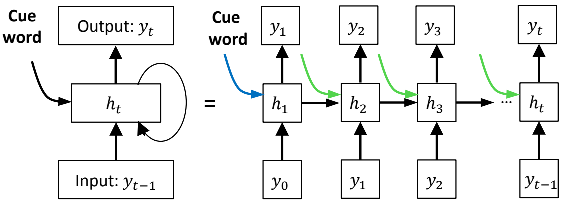


Figure 2: The information fusion patterns. The local information initialization is presented by the blue arrowhead and the global information inception includes both the blue arrowhead and the green arrowhead.

3.1 Cue Word Prediction

In computational linguistics, PMI has been used for finding collocations and associations between words. As mentioned in Mou et al. (2016), it is an appropriate statistic for cue words prediction, which is also adopted in this paper to predict a cue word C_w for the given query. Formally, given a query word w_q and a reply word w_r, the PMI is computed as:

$$\mathrm{PMI}(w_q, w_r) = \log \frac{p(w_q|w_r)}{p(w_q)} \tag{6}$$

Then, we choose the cue word C_w with highest PMI score against the query words $w_{q1}, ..., w_{qn}$ during the prediction, i.e., $C_w = \mathrm{argmax}_{w_r}\, \mathrm{PMI}(w_{q1}, ..., w_{qn}, w_r)$, where

$$\mathrm{PMI}(w_{q1}, ..., w_{qn}, w_r) \approx \log \frac{\prod_i p(w_{qi}|w_r)}{\prod_i p(w_{qi})}$$
$$= \sum_i \log \frac{p(w_{qi}|w_r)}{p(w_{qi})} = \sum_i \mathrm{PMI}(w_{qi}, w_r) \tag{7}$$

The approximation is based on the independence assumptions of both the prior distribution $p(w_{qi})$ and posterior distributions $p(w_{qi}|w_r)$. Even the two assumptions may not be true, we use them in a pragmatic way so that the word-level P-MI is additive for a whole utterance. PMI penalizes a common word by dividing its prior probability; hence, it prefers a word which is most "mutually informative" with the query.

3.2 Information Fusion Patterns

To implant the specific information in conversation system, we consider two types of information fusion patterns, namely 1) Local information initialization 2) Global information inception.

Local information initialization. In the local pattern, we fuse the cue word C_w as the auxiliary

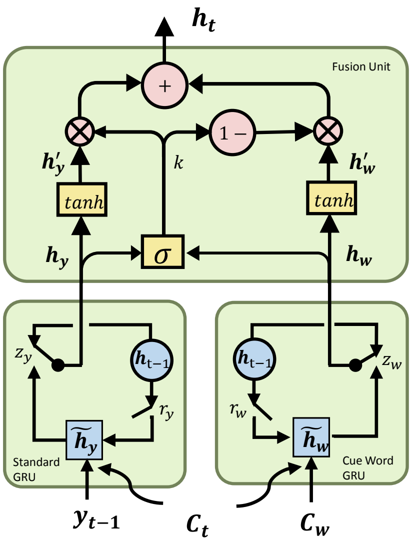


Figure 3: The structure of a HGFU. The bottom of two GRUs deal with corresponding input source, i.e., the last generated word y_{t-1} and the cue word C_w. After that, fusion unit combines the output of two GRUs to compute current hidden state h_t.

.

information only in the beginning of decoding. We describe this kind of pattern by the blue arrowhead in Figure 2. Recurrent neural networks(RNNs) such as gated recurrent units (GRUs) have the ability to keep the information from the beginning to the end to some extent. Therefore, the cue word added on the first step of the neural networks can still influence the generation of the later steps.

Global information inception. However, we observe that, although the network is capable of deciding what to keep in the cell state to affect the later generation, the influence of the added information in the beginning of decoding is becoming weaker and weaker over time. Therefore, to provide the model a broader and more flexible space for learning, we propose a global information inception pattern, which fuses the cue word C_w as the auxiliary information at every step of decoding. This process is presented by both the blue arrowhead and the green arrowheads in Figure 2.

3.3 Hierarchical Gated Fusion Unit

In this subsection, we propose our Hierarchical Gated Fusion Unit (HGFU), which incorporates cue words into the generation process and relaxes the constraint from the "hard" content-introducing method into a new "soft" schema. Figure 3 provides an overview of the structure of a HGFU. As seen, the framework consists of three components: the standard GRU, the cue word GRU, and the fusion unit. Among them, standard GRU and cue word GRU take the last generated word y_{t-1} and cue word C_w respectively as the decoder GRU's input; the fusion unit combines the hidden states of both GRUs to predict the next word y_t. In the following, we will illustrate these components in detail.

3.3.1 Standard GRU

We adopt the standard gated recurrent unit (GRU) with the attention mechanism at the decoder part. Let h_{t-1} be the last hidden state, y_{t-1} be the embedding of the last generated word, and C_t be the current attention-based context. The current hidden state of the general decoding, h_y, is defined as:

$$\begin{aligned} r_y &= \sigma(W_r y_{t-1} + U_r h_{t-1} + U_{cr} C_t + b_r) \\ z_y &= \sigma(W_z y_{t-1} + U_z h_{t-1} + U_{cz} C_t + b_z) \\ \widetilde{h_y} &= \tanh(W_h y_{t-1} + U_h(r_y \circ h_{t-1}) + U_{ch} C_t + b_h) \\ h_y &= (1 - z_y) \circ h_{t-1} + z_y \circ \widetilde{h_y} \end{aligned} \tag{8}$$

where W's $\in \mathbb{R}^{dim \times E}$ and U's $\in \mathbb{R}^{dim \times dim}$ are weight matrices; b's $\in \mathbb{R}^{dim}$ are bias terms; E denotes the word embedding dimensionality and dim denotes the number of hidden state units. This general decoding process is presented by the "Standard GRU" in Figure 3.

3.3.2 Cue word GRU

To generate more meaningful and informative replies, we introduce cue words as the additional information during generation. Naturally, the key point lies in how to incorporate such information. One of the prevailing methods is modifying the neural cell by various gating mechanisms. However, these approaches are designed specially for a specific scenario, and not effective as expected when they are employed to other tasks. To tackle this issue, we propose the cue word GRU, another independent neural cell, to deal with the auxiliary information. Since this neural cell can be replaced easily by other units, it greatly improves the flexibility and reusability.

Given the last hidden state h_{t-1} , the additional cue word C_w and the current attention-based context C_t, the new hidden state of the auxiliary decoding h_w is computed by following equations:

$$\begin{aligned} r_w &= \sigma(W_r C_w + U_r h_{t-1} + U_{cr} C_t + b_r) \\ z_w &= \sigma(W_z C_w + U_z h_{t-1} + U_{cz} C_t + b_z) \\ \widetilde{h_w} &= \tanh(W_h C_w + U_h(r_w \circ h_{t-1}) + U_{ch} C_t + b_h) \\ h_w &= (1 - z_w) \circ h_{t-1} + z_w \circ \widetilde{h_w} \end{aligned} \tag{9}$$

where W's and U's are weights and b's are bias terms like those in the standard GRU. Note that the standard GRU does not share parameter matrixes with the cue word GRU. The "Cue word GRU" in Figure 3 describes the auxiliary decoding process.

3.3.3 Fusion unit

To combine both the general decoding information and the auxiliary decoding information, we apply the fusion unit (Arevalo et al., 2017) integrating the hidden states of both standard GRU, i.e., h_y, and the cue word GRU, i.e., h_w, to compute the current hidden state h_t. The equations are as follows:

$$\begin{aligned} h_y^{'} &= \tanh(W_1 h_y) \\ h_w^{'} &= \tanh(W_2 h_w) \\ k &= \sigma(W_k[h_y^{'}, h_w^{'}]) \\ h_t &= k \circ h_y + (1 - k) \circ h_w \\ \theta &= \{W_1, W_2, W_k\} \end{aligned} \tag{10}$$

with θ the parameters to be learned. From the equations above we can see that, the gate neuron k controls the contribution of the information calculated from h_y and h_w to the overall output of the unit.

3.4 Model Training

When training on the aligned corpus, we randomly sample a noun in the reply as the cue word. The objective function was the cross entropy error between the generated word distribution p_t and the actual word distribution y_t in the training corpus.

4 Experiments

In this section, we compare our method with the-state-of-art response generation models based on a huge conversation resource. The objectives of our experiments are to 1) evaluate the effectiveness of our proposed HGFU model, and 2) explore how cue words affect the process of reply generation.

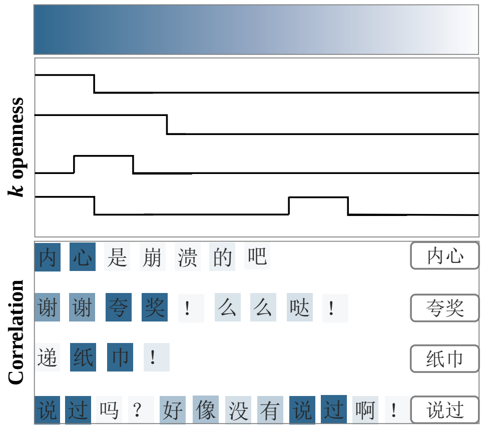


Figure 4: Heat map and the k gate openness. Bottom: The correlation between the generated reply words and the cue word. Top: The openness of k gate in fusion unit.

4.1 Experimental setup

We evaluated our model on a massive Chinese dataset of human conversation crawled from the Baidu Tieba[1] forum. There are 500,000 $\langle query - reply \rangle$ pairs for training, 2,000 for validation, and another unseen 27,871 samples for testing. In total, we kept about 63,000 distinct words.

In our experiments, the encoder, the standard decoder and the cue word decoder have 1,000 hidden units; the word embedding dimensionality is 610 which were initialized randomly and learned during training. We applied AdaDelta with a mini-batch size of 80 for optimization. These values were mostly chosen empirically. In order to prevent overfitting, early stopping was implemented using a held-out validation set.

4.2 Comparison Methods

In this paper, we conduct extensive experiments to compare our proposed method against several representative baselines. All the methods actually are implemented in two ways to utilize the cue word, which are local information initialization and global information inception.

rGRU: Through a specially designed Recall gate (Xu et al., 2016), domain knowledge was transformed into the extra global memory of a deep neural network.

SCGRU: In SCGRU (Wen et al., 2015b), an additional control cell was introduced to gate the dia-

[1] http://tieba.baidu.com

Query (**Cue word**)	班主任还拍了我超级丑的照片已被笑死.(上镜) The teacher took a photo of me; it was really ugly and people laughed at me. (**Photogenic**)	Related Criterion	Labels
Reply1	谁的照片？ Whose photo?	Logic Consistency	Unsuitable
Reply2	什么时候拍的？ When did he took the photo?	Implicit Relevance	Neutral
Reply3	抱抱。Give you a hug.	Implicit Relevance	Neutral
Reply4	我拍照也都是巨丑的！ My photos are also ugly!	——	Suitable

Table 2: An example query, corresponding cue word in **bold** and its candidate replies with human annotation. The query states that people laughed at the author's photo, it is unsuitable to ask the ownership of this photo in Reply1. Generally, Reply2 and Reply3 apply to this scenario, but they do not reflect semantic relevance with the cue word. Reply4 talks about the respondent's situation and related to "Photogenic", thus it is a suitable response.

Method		BLEU-1	BLEU-2	BLEU-3	Human score
Local	rGRU	1.087	0.419	0.249	
	SCGRU	2.135	0.622	0.255	
	SLGD	1.678	0.508	0.209	
	FGRU	2.262	0.598	0.208	
	HGFU	1.861	0.545	0.209	
Global	rGRU	1.793	0.676	0.277	0.542
	SCGRU	3.637	0.981	0.369	0.73
	SLGD	4.146	1.059	0.367	0.71
	FGRU	4.197	1.013	0.282	0.677
	HGFU	**4.893**	**1.225**	**0.393**	**0.942**

Table 4: Performance of evaluated methods.

logue act (DA) features during the generation process.

SLGD: We implemented the Stochastic Language Generation in Dialogue (SLGD) method (Wen et al., 2015a), which added additional features in each gate of the neural cell.

FGRU: To explore more fusion strategies, intuitively, we fused the cue word and hidden states by vector concatenation during the decoding process.

Note that rGRU and SCGRU incorporate additional information by gating mechanisms, while SLGD and FGRU fuse the information into each gate of the neural cell directly.

4.3 Experiment Evaluation

Objective metrics. To evaluate the performance of different methods for the conversation generation task, we leverage BLEU (Papineni et al., 2002) as the automatic evaluation metric, which is originally designed for machine translation and evaluates the output by using n-gram matching between the output and the reference. Here, we use BLEU-1, BLEU-2 and BLEU-3 in our experiments.

Subjective metrics. Since automatic metrics may not consistently agree with human perception (Stent et al., 2005), human testing is essential to assess subjective quality. Hence, we randomly sampled 150 queries in the test set, then we invited five annotators to offer a judgment. For fairness, all of our human evaluation was conducted in a random, blind fashion, i.e., replies obtained from the five evaluated models are pooled and randomly permuted for each annotator. Three levels are assigned to a reply with scores from 0 to 2: 0 = Unsuitable reply, 2 = Suitable reply, and 1 = Neutral reply.

To make the annotation task operable, the suitability of the generated reply is judged not only based on *Grammar and Fluency, Logic Consistency and Semantic Relevance* following (Shang et al., 2015), but also *Implicit Relevance*, i.e., the generated reply should be semantically relevant to the predicted cue word, no matter the cue word explicitly appears in the reply or not. If any of the first three criteria is contradicted, the reply should be labeled as "Unsuitable". Only the replies conforming to all requirements are labeled as "Suitable". Table 2 shows an example of the annotation results of a query and its replies. The first reply is labeled as "Unsuitable" because of the logic consistency. Reply2 and Reply3 are not semantically related to the cue word, and is therefore annotated as "Neutral".

4.4 Overall Performance

The overall results against all baseline methods are listed in Table 4. Our proposed HGFU model in global schema obviously shows better performance than the baseline methods; it obtains the

	Chinese Sentence	English Tranlation
Query	写的真心棒！(**夸奖**)	What a nice written! (**Appreciation**)
Reply	谢谢夸奖！么么哒！	Thanks for your appreciation! Love you!
Query	还是无法淡定。(**内心**)	Still cannot calm down. (**Heart**)
Reply	内心是崩溃的吧。	Your heart must be broken.
Query	我先去哭一会。(**纸巾**)	I am going to cry for a while. (**Tissue**)
Reply	递纸巾！	Offer you a tissue!
Query	当初你们不是说过他是诺维斯基吗？(**说过**)	Didn't you say that he was $Nowitzki^{\dagger}$? (**Say**)
Reply	说过吗？好像没有说过啊！？	Did I say it? I don't seem to say it!?

Table 3: The explicit introducing-content cases of our HGFU model. The predicted cue word in **bold** explicitly occurs in the generated reply. $Nowitzki^{\dagger}$ is a NBA basketball player.

highest BLEU scores as well as the highest human score.

In terms of automatic evaluations, the global-based methods perform much better than a set of local-based methods, which demonstrates the effectiveness of global information inception. As mentioned above, the global schema provides the model a broader and more flexible space for learning, which is benefit for information fusion. When it comes to human scores (For the sake of convenience, we only conducted human evaluation in global schema), there are similar conclusions to BLEU results.

From Table 4, we can see that the performance of rGRU is not as good as the other systems, while SCGRU outperforms the others in the local pattern and shows comparative performance in the global schema. These two methods both augment the standard neural network with specially designed gate to control the cue word, but the results vary greatly. It is the limitation of gating mechanisms that is lacking in adaptiveness. Besides, SLGD adding cue word term in each gate of the neural cell has the similar result as FGRU method, which concatenates cue word with hidden state. Basically, our proposed HGFU has a significant improvement against the baseline systems. The most probable credits come from the cue word GRU: we apply the extra GRU unit to control the auxiliary information instead of fusion in the standard GRU, which is more flexible.

Till now, we have elaborated the overall performance of all methods. Next we will come to a closer look at some representative cases of our HGFU model for further analysis and discussions.

4.5 Analysis and Case Studies

Given a query and the cue word, our HGFU model generates a meaningful and informative response. In Table 3, the predicted cue word occurs in the generated response and we treat this kind of generation as the explicit introducing-content. However, we do not strictly restrict tothis. As shown in Table 5,our HGFUmodel also generates the replies without containing the cue word, while the responsesare still somehow related to the cue word and the query. This reflects our expectation: the information of the cue word will be fused into the generation process but not necessarily occur explicitly. It provesthe characteristics of our proposed new "soft" schema, whichare more flexible, extensible, and controllable.

We further analyze these explicit cases using a heat map as shown in Figure 4. We use various shades of blue to present the extent of correlation between the cue word and the generated reply. The darker the blue is, the higher correlation they have. For the added information in the reply (Here is exactly the cue word in darkblue), its position and occurrence times are not fixed, which are autonomously controlled by our model.

Besides, the rectangular pulse is also a significant presentation of this correlation, which indicates how the k gate in fusion unit balance the influence of h_y and h_w. When in the high level of the rectangular pulse, k "opens" the switch of h_w to generate the current word; when in the low level, the fusion unit mainly takes h_y for generation. We observe that the switch corresponds with the heat map: the generated word is more correlated with the cue word when the switch is open.

	Chinese Sentence	English Tranlation
Query	在微博看到这样一个评论真的是完全无法反驳。(**观点**)	I saw such a comment in the microblog which cannot be refused completely. (**View**)
Reply	路人表示不服啊！	As an outsider, I am not convinced!
Query	怎么突然就下雨了？(**委屈**)	Why is it raining suddenly? (**Grievance**)
Reply	好伤心啊。	So sad.
Query	泰民这张也是做了很久桌面。(**屏保**)	This photo of $Taemin^{\dagger}$ was also taken as a desktop for a long while. (**Screenshot**)
Reply	锁屏吗?	As the lockscreen?
Query	混脸熟求勾搭！(**小新**)	Make acquaintance and seek chances for further relations! (**Freshman**)
Reply	同新人！求认识。	I am also the new! Nice to meet you.

Table 5: The implicit introducing-content cases of our HGFU model. The cue word in **bold** is not contained in the reply, while the response is still related to the cue word. $Taemin^{\dagger}$ is a Korean singer.

5 Related work

5.1 Conversation Systems

Automatic human-computer conversation has attractedincreasing attention over the past few years. At the very beginning, people start the research using hand-crafted rules and templates (Walker et al., 2001; Misu and Kawahara, 2007; Williams et al., 2013). These approaches require no data or little data for trainingbuthuge manual effort to build the model, which is very time-consuming. For now, buildinga conversation systemmainly falls into two categories: retrieval-based and generation-based. As information retrieval techniques are developing fast, Leuski et al. (2009) build systems to select the most suitable response from the query-reply pairs using a statistical language model in cross-lingual information retrieval. Yan et al. (2016) propose a retrieval-based conversation system with the deep learning-to-respond schema through a deep neural network framework driven by web data. Recently, generation-based conversation systems have shownimpressive potential. Shang et al. (2015) generate replies for short-text conversation by Seq2Seq-basedneural networks with local and global attentions.

5.2 Content Introducing

In vertical domains, Wen et al. (2015b) apply an additional control cell to gate the dialogue act (DA) features during the generation process to ensure the generated repliesexpressthe intended meaning. Also, the Stochastic Language Generation in Dialogue method (Wen et al., 2015a) adds additional features in each gate of the neural cell. Xu et al. (2016) introduce a new trainable gate to recall the global domain memory to enhance the ability of modeling the sequence semantics. Different from the above work, our paper addresses the problem of content introducing in the open-domain generative conversation systems.

In open domains, Xing et al. (2016) incorporate topic information into Seq2Seq framework to generate informative and interesting responses. To provide informative clues for content introducing, Li et al. (2016b) detect entities from previous utterances and search for more related entities in a large knowledge graph. A very recent study similar to ours is Mou et al. (2016), where the predicted word explicitly occurs in the generated utterance. Unlike the existing work, we explore an implicit content-introducing method for neural conversation systems, which utilizes the additional cue word in a "soft" manner to generate a more meaningful response given a user-issued query.

6 Conclusion

In this paper, we explore an implicit content-introducing method for generative short-text conversation system. Given a user-issued query, our proposed HGFU incorporates an additional cue word in a "soft" manner to generate a more meaningful response. The HGFU model consists of three components: the standard GRU, the cue word GRU and the fusion unit. The standard GRU operates a general decoding process, and the cue word GRU imitates this process but treats the predicted cue word as the current input. As for the fusion unit, it combines both the hidden states of the standard GRU and the cue word GRU to generate the current output word. The experimental results demonstrate the effectiveness of our approach.

Acknowledgments

This work was supported by the National Hi-Tech R&D Program of China No. 2015AA015403; the National Science Foundation of China No. 61672058.

References

John Arevalo, Thamar Solorio, Manuel Montes-y Gómez, and Fabio A González. 2017. Gated multimodal units for information fusion. *arXiv preprint arXiv:1702.01992*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

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

Charles Lee Isbell, Michael Kearns, Dave Kormann, Satinder Singh, and Peter Stone. 2000. Cobot in lambdamoo: A social statistics agent. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 36–41. AAAI Press.

Anton Leuski, Ronakkumar Patel, David Traum, and Brandon Kennedy. 2009. Building effective question answering characters. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, pages 18–27. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119. Association for Computational Linguistics.

Xiang Li, Lili Mou, Rui Yan, and Ming Zhang. 2016b. Stalematebreaker: A proactive content-introducing approach to automatic human-computer conversation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2845–2851.

Teruhisa Misu and Tatsuya Kawahara. 2007. Speech-based interactive information guidance system using question-answering technique. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–145. IEEE.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 3349–3358.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, pages 583–593. Association for Computational Linguistics.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3776–3783. AAAI Press.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1577–1586.

Yiping Song, Rui Yan, Xiang Li, Dongyan Zhao, and Ming Zhang. 2016. Two are better than one: An ensemble of retrieval-and generation-based dialog systems. *arXiv preprint arXiv:1610.07149*.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 341–351. Springer.

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

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *Deep Learning Workshop of the 32nd International Conference on Machine Learning*.

Marilyn A Walker, Rebecca Passonneau, and Julie E Boland. 2001. Quantitative and qualitative evaluation of darpa communicator spoken dialogue systems. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 515–522.

Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 935–945. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gasic, Dongho Kim, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 275–284.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2016. Topic augmented neural response generation with a joint attention mechanism. *arXiv preprint arXiv:1606.08340*.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2016. Incorporating loose-structured knowledge into lstm with recall gate for conversation modeling. *CoRR*, abs/1605.05110.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 55–64. ACM.

Affordable On-line Dialogue Policy Learning

Cheng Chang*, **Runzhe Yang***, **Lu Chen, Xiang Zhou and Kai Yu**
Key Lab. of Shanghai Education Commission for Intelligent Interaction and Cognitive Eng.
SpeechLab, Department of Computer Science and Engineering
Brain Science and Technology Research Center
Shanghai Jiao Tong University, Shanghai, China
{cheng.chang,yang_runzhe,chenlusz,owenzx,kai.yu}@sjtu.edu.cn

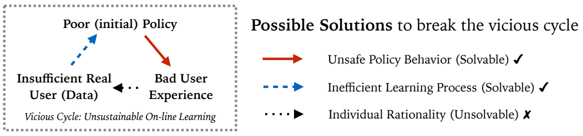
Abstract

The key to building an evolvable dialogue system in real-world scenarios is to ensure an affordable on-line dialogue policy learning, which requires the on-line learning process to be safe, efficient and economical. But in reality, due to the scarcity of real interaction data, the dialogue system usually grows slowly. Besides, the poor initial dialogue policy easily leads to bad user experience and incurs a failure of attracting users to contribute training data, so that the learning process is unsustainable. To accurately depict this, two quantitative metrics are proposed to assess safety and efficiency issues. For solving the unsustainable learning problem, we proposed a complete *companion teaching* framework incorporating the guidance from the human teacher. Since the human teaching is expensive, we compared various *teaching schemes* answering the question *how* and *when* to teach, to economically utilize teaching budget, so that make the online learning process affordable.

1 Introduction

A *task-oriented* dialogue system is designed for interacting with humans users to accomplish several predefined domains or tasks (Young et al., 2013; Daubigney et al., 2012). *Dialogue Manager* is the core component in a typical dialogue system, which controls the flow of dialogue by a *state tracker* and a *policy module* (Levin et al., 1997). The state tracker tracks the internal state of the system while the policy module decides the response to the user according to the status of states (Sun et al., 2014a; Thomson and Young, 2010). The approaches of constructing a policy module can be divided into two categories: rule-based and statistical. Rule-based policies are usually hand-crafted by domain experts which means they are inconvenient and difficult to be optimized (Williams and Young, 2007; Wang and Lemon, 2013). In recent mainstream statistical studies, *Partially Observable Markov Decision Process* (POMDP) framework has been applied to model dialogue management with unobservable states, where policy training can be formulated as a *Reinforcement Learning* (RL) problem, which enables the policy to be optimized automatically (Kaelbling et al., 1998; Arnold, 1998; Young et al., 2013).

Though RL-based approaches have the potential to improve themselves as they interact more with human users and achieve better performance than rule-based approaches, they are rarely used in real-world applications, especially in on-line scenarios, since the training process is unsustainable.


Poor (initial) Policy
Insufficient Real User (Data)
Bad User Experience
Vicious Cycle: Unsustainable On-line Learning
Possible Solutions to break the vicious cycle
Unsafe Policy Behavior (Solvable) ✓
Inefficient Learning Process (Solvable) ✓
Individual Rationality (Unsolvable) ✗

The main causes of unsustainable on-line dialogue policy learning are two-fold:

- *Safety issue:* the initial policy trained from scratch may lead to terrible user experience at the early training period, thus fail to attract sufficient users for more dialogues to do further policy training.
- *Efficiency issue:* if the progress of policy learning is not so efficient, it will exhaust users' patience before the policy reaches a desirable performance level.

[*] Both authors contributed equally to this work.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2200–2209
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Prior works have mainly focused on improving *efficiency*, such as Gaussian Processes RL (Gašić et al., 2010), deep RL (Fatemi et al., 2016), etc. For deep RL approaches, recent researches on the *student-teacher RL framework* have shown prominent acceleration to policy learning process (Torrey and Taylor, 2013; Williams and Zweig, 2016; Amir et al., 2016). In such framework, the *teacher agent* instructs the *student agent* by providing suggestions on what actions should be taken next (Clouse, 1996).

For the *safety* issue, Chen et al. (2017) developed several teaching strategies answering "how" the human teacher guide the learning process.

However, those previous teaching methods exclude "when" to teach from concern. They simply exhaust all the budget continuously from the beginning, which is wasteful and causes a heavy workload of the human teacher. An *affordable* dialogue policy learning with human teaching should require a lighter workload and economically utilize teaching budget.

Furthermore, as for safety and efficiency evaluation, previous works have been observing the training curves and testing curves to tell which one is better, or evaluate policy performance after certain dialogues of training, which are subjective and error prone (Chen et al., 2015a; Su et al., 2016; Chen et al., 2017).

Our contribution is to address the above problems. We propose a complete framework of companion teaching, and develop various *teaching schemes* which combine different *teaching strategies* and *teaching heuristics* together, to answer the questions of "how" and "when" to teach to achieve affordable dialogue policy learning (section 2). Specifically, a novel *failure prognosis* based teaching heuristic is proposed, where MultiTask Learning (MTL) is utilized to predict the dialogue success reward (section 3). To avoid the drawbacks of traditional subjective measurements, we propose two evaluation metrics, called *Risk Index* (RI) and *Hitting Time* (HT), to quantify the safety and efficiency of on-line policy learning respectively (section 4). Simulation experiments showed, with the proposed companion teaching schemes, sustainable and affordable on-line dialogue policy learning has been achieved (section 5).

2 Companion Teaching Framework

The *companion teaching* framework is an on-line policy training framework with three intelligent participants: machine dialogue manager, human user, and human teacher (Chen et al., 2017). Under this framework, the human teacher is able to accompany the dialogue manager to guide policy learning with a limited teaching budget. By investigating the real work mode in a call center, this framework makes a reasonable assumption that human teacher has access to the extracted dialogue states from the dialogue state tracker as well as the system's dialogue act, and can also reply in the same format.

However, there are two major problems in the previous framework. First, the system will judge whether a dialogue session succeeds by several simple rules and then determine whether to feed a success reward signal to dialogue manager for reinforcement learning. Actually, the success feedback made by the system lacks flexibility and credibility, and it could mislead the policy learning. A more suitable judge should be the user or the human teacher. Second, the previous framework only answers in which way the human teacher can guide the online dialogue policy learning, but another essential question, *when* should the human teacher give guidance, remains undiscussed.

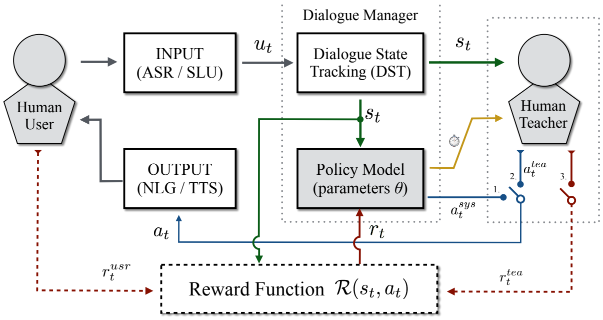


Figure 1: Companion Teaching Framework for On-line Policy Learning

In this paper, we proposed a complete framework of *companion teaching*, depicted as Figure 1. At each turn, the input module receives a speech input from the human user, then produces possible utterances u_t of the speech in text. After that, the dialogue state tracker extracts the dialogue state s_t from possible utterances. This dialogue state will be shared with policy model and human teacher if needed. When the final response a_t, has been determined, the output module will translate this dialogue act to the natural language and reply to the human user. The success signal will be fed back by the user or the human teacher as an important part of system reward, at the end of each session. The human teacher can take the initiative or be activated by *student initiated heuristic* to give the dialogue guidance with strategies corresponding to different configurations of switches in the illustration. We call the combination of strategy and heuristic as *teaching scheme*.

2.1 Teaching Strategies

The teacher can choose among three teaching strategies corresponding to different configurations of switches in a wiring diagram as Figure 1 shows: The left switch is a Single-Pole, Double-Throw (SPDT) switch, which controls whether the answer is made by the system (connected to 1) or given by the teacher as an example (connected to 2). The right switch is a simple on-off switch, which represents whether there is an extra reward signal from the teacher (ON) or not (OFF). The strategy related to the right switch is called *teaching via Critic Advice* (CA), also known as turn-level reward shaping (Thomaz and Breazeal, 2008; Judah et al., 2010). When the switch at position 3 is turned on, the teacher will give the policy model extra turn-level reward to distinguish the student's actions between good and bad actions. Besides, the left switch corresponds to *teaching via Example Action* (EA), which means the teacher gives example action for the student to take according to the student's state.

The other strategy is proposed by Chen et al. (2017), which take the advantages of both EA and CA, named *teaching via Example Action with Predicted Critique* (EAPC). With this strategy, the human teacher gives example actions, meanwhile, a weak action predictor is trained using this teaching information to provide the extra reward even in teacher's absence.

2.2 Teaching Heuristics

The strategies only answer how the human teacher can offer companion teaching to the system. However, the timing of teaching should not be ignored for the sake of utilizing the limited teaching budget better. Exhausting all the budget at early training stage, named *Early teaching heuristic* (Early), is simple and straightforward but wastes teaching opportunities on unnecessary cases. Thus, it is imperative to design some effective heuristics to instruct when the teacher should give a hand to the student.

In addition to early teaching, the teaching heuristics can be broadly divided into two categories: *teacher-initiated* heuristics and *student-initiated* heuristics (Amir et al., 2016). However, the teacher-initiated approaches require the constant long-term attention of the teacher to monitor the dialogue process (Torrey and Taylor, 2013; Amir et al., 2016), which is costly and impractical for real applications. Therefore, in this paper, we only discuss student-initiated heuristics, shown as the line with a stopwatch in Figure 1, which means that the student agent decides when to ask for the teacher's help.

Previous works have presented several effective heuristics based on *state importance*, $I(s)$, which is determined by the Q-values of the RL agent:

$$I(s) = max_a Q_{(s,a)} - min_a Q_{(s,a)}$$

Torrey and Taylor (2013) proposed *State Importance based Teaching heuristic* (SIT) which make the student ask for advice only when the current state is important:

$$I(s) > t_{si}, \tag{1}$$

where t_{si} is a fixed threshold for importance. And Clouse (1996) proposed an *State Uncertainty based Teaching heuristic* (SUT) which ask for advice when the student is uncertain about which action to take:

$$I(s) < t_{su}, \tag{2}$$

where t_{su} is a given threshold for uncertainty.

Though teaching effort can be conserved by only applying to those important or uncertain states, it may end up wasting advice if the dialogue is likely to be successful without teaching. In this paper, we propose a novel *Failure Prognosis based Teaching heuristic* (FPT) for on-line policy learning to reduce that unnecessary advice. The details are given in section 3. For comparison, we will also investigate *Random teaching heuristic* (Rand) which means the student seek for advice with a fixed probability p_r.

3 Failure Prognosis Based Teaching Heuristic

To make better use of teaching advice, we propose to use an on-line turn-level *task success predictor* to predict whether the ongoing dialogue will end in success and ask for advice only when the current prediction is a failure. The proposed approach utilizes MultiTask Learning (MTL) for the policy model to estimate future dialogue success reward and is compatible with various RL algorithms. In this paper, we implement the policy model with a Deep Q-Network (DQN), in which a neural network function approximator, named *Q-network*, is used to estimate the action-value function (Mnih et al., 2013).

3.1 Multitask Deep Q-Network

The goal of the policy model is to interact with human user by choosing actions in each turn to maximize future rewards. We define the dialogue state shared by dialogue state tracker in the t-th turn as s_t, the action taken by policy model under current policy π_θ with parameters θ in the t-th turn as a_t, and $a_t \sim \pi_\theta(\cdot|s_t)$. In an ideal dialogue environment, once the policy model emit an action a_t, the human user will give an explicit feedback, like a normal response or a feedback of whether the dialogue is successful, which will be converted to a reward signal r_t delivering to the policy model immediately, and then the policy model will transit to next state s_{t+1}. The reward r_t is composed of two parts:

$$r_t = r_t^{turn} + r_t^{\mathtt{succ}},$$

where r_t^{turn} is the turn penalty reward and $r_t^{\mathtt{succ}}$ is the dialogue success reward. Typically, r_t^{turn} is fixed for each turn as a negative constant R^{turn}, while $r_t^{\mathtt{succ}}$ equals to a predefined positive constant $R^{\mathtt{succ}}$ only when the dialogue terminates and receives a successful user feedback otherwise zero.

In DQN algorithm, all these transitions (s_t, a_t, r_t, s_{t+1}) will be stored in a replay memory $\mathcal{D}$. And the objective is to optimize MSE between *Q-network* $Q(s, a; \theta)$ and *Q-learning target* Q_e. The loss function $L(\theta)$ is defined as:

$$L(\theta) = \mathbb{E}_{s,a\sim\pi_\theta}\left[(Q_e - Q(s, a; \theta))^2\right]. \qquad (3)$$

During the training period, Q_e is estimated with old fixed parameter θ^- and sampled transitions $e \sim \mathcal{D}$:

$$Q_e = r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta^-), \qquad (4)$$

where γ is the discount factor.

The reward $Q(s, a)$ estimated by original Q-learning algorithm is essentially a combination of future turn penalty reward $Q^{\mathtt{turn}}(s, a)$ and future dialogue success reward $Q^{\mathtt{succ}}(s, a)$. For a task-oriented dialogue system, the prediction of $Q^{\mathtt{succ}}(s, a)$ is much more important because it reflects the possibility of the dialogue to be successful. If these two rewards are estimated separately, the objective of $Q^{\mathtt{succ}}(s, a)$ can be optimized explicitly, and we can get more insights into the estimated future. We found that in practice, optimizing these two objectives with *MultiTask Learning* (MTL) converges faster and more stable compared with two separate models, the reason of which may lie in that MTL can learn different related tasks in parallel using shared representations, which will be helpful for each task to be learned better (Caruana, 1997). The structure of proposed MTL-DQN is depicted in Figure 2.

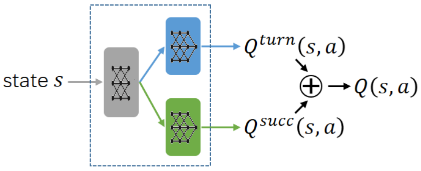


Figure 2: MTL-DQN structure

3.2 Failure Prognosis

In the proposed multitask DQN, we define *on-line task success predictor* $T(s_t)$ as:

$$T(s_t) = Q^{\mathtt{succ}}(s_t, a_t),$$

where a_t is the action taken under state s_t. It is reasonable to assume that the dialogue is going to fail if $T(s_t)$ is relatively small. Based on the task success predictor, we propose a novel student-initiated heuristic, named *Failure Prognosis based Teaching heuristic* (FPT).

The key to the proposed heuristic is to define *failure prognosis* quantitatively. A straight way is to set a ratio threshold α, and consider it to be *failure prognosis* when $T(s_t) < \alpha R^{\mathtt{succ}}$. However, this assumes that the numerical scale of $Q^{\mathtt{succ}}$ is consistent through the training period, which is not always the case. And the student's noisy estimation of $Q^{\mathtt{succ}}$ at early training period will make the learning process unstable. To smooth the teaching, we consider using a turn-level sliding window near the current state to calculate an average value as the replacement of the fixed R^{succ}. So in the t-th turn, the *failure prognosis* for the student to be *true* is equivalent to:

$$T(s_t) < \alpha \frac{1}{w} \sum_{j=t-w}^{t-1} T(s_j), \tag{5}$$

where w is the size of the sliding window.

4 Quantitative Measurements for Safety and Efficiency

The performance of different teaching strategies and heuristics should be measured in both the *safety* and *efficiency* dimension. However, the measurements in these two dimensions are subjective and error prone in the previous work (Chen et al., 2017). Especially for assessing the degree of safety of various teaching strategies and heuristics, we simply obverse the training curves so that we cannot tell of two interleaving curves which training process is safer. Thus, it is imperative to set up some quantitive measurements for both *safety* and *efficiency* evaluations. In this paper, we propose two scalar criteria: *Risk Index* (RI) and *Hitting Time* (HT).

4.1 Risk Index

The *Risk Index* is a nonnegative index designed to indicate how risky the training process could be for evaluating the safety issue during the whole online dialogue policy learning process. Because we expect that the system satisfies the quality-of-service requirement in the early training period, specifically, we hope it can keep a relatively acceptable success rate. It is straightforward to set a success rate *threshold* for the training process. In a real application scenario, this threshold can be obtained by an appropriate user study.

If the success rate over a training process keeps above this threshold all the time, we will think this training process is absolutely safe. Therefore, its RI should equal to zero.

On the other hand, if the success rate over a training process rises and falls and sometimes is below the threshold, it is risky. The riskiness consists of two parts:

- **Disruptiveness**: Sometimes the success rate during a certain period will fall much lower than the threshold, which could be very disruptive. To quantify the disruptiveness, we consider the function

  $$\texttt{dis}(t) = \texttt{threshold} - \%\texttt{succ}(t)$$

  over the training process. The higher the value of $\texttt{dis}(t')$ is, the riskier the training process could be during the period of a certain length centered with time t'.

- **Persistence**: Another thing we should take into account is the duration of the time at high risk. Let $\delta_{\texttt{risk}}(t)$ be the indicator of whether $\texttt{threshold} \geq \%\texttt{succ}(t)$. Then the persistence can be quantified as total risky time

  $$\texttt{per}(T) = \int_{t=0}^{T} \delta_{\texttt{risk}}(t) dt$$

  The longer the danger persists over the training process, the value of persistence of the training process will be, and the riskier it is.

Our *Risk Index* integrate these two contents of riskiness. That is, a nonnegative scalar

$$\texttt{RI} = \int_{t=0}^{T} \texttt{dis}(t)\delta_{\texttt{risk}}(t) dt,$$

which measures the integrated riskiness for the online training process of total length T. The RI also has an intuitive interpretation as the area of the region which is below the threshold line and above training curve. Straightforwardly, high RI indicates poor safety.

4.2 Hitting Time

To measure the efficiency, we proposed the *Hitting Time* in order to show how fast the system learns and reaches the satisfactory performance.

The difficulty of designing such a criterion lies in the dramatic and undamped fluctuation of the test curves, which is inherent in the instability dialogue task. Therefore, many popular criteria for the evaluating dynamic performance in control theory, such as “settling time” and “rise time”, cannot be applied to evaluate efficiency here.

We use *Hitting Time* to evaluate efficiency over the fluctuant testing curve first by fitting it to the empirical learning curve

$$f(t) = a - b \cdot e^{-(t/c)^2},$$

where the parameter a is the stationary performance which is predicted as the asymptotic goal of the system, b relates to the initial performance, and c relates to the climbing speed. This empirical model forces our fitted learning curve be an "S" shape curve satisfying constraints $f'(0) = 0$ and $f''(c/2) = 0$. Then we observe when this fitted learning curve hits the target performance τ and this time (measured in sessions) is *Hitting Time*. It can be calculated analytically as follow

$$\mathrm{HT} = c\sqrt{\ln\left(\frac{b}{a-\tau}\right)}.$$

Ideally, the ultimate success rate a should be very close under different settings because of the sufficient training. However, if the success rate keeps very poor during the given sessions, the fitted a will be very low, and even less than the target satisfactory performance τ. In this situation, a is meaningless, and HT becomes a complex number. And this indicates the real hitting time is far larger than given number of sessions T. We will note the HT in this case as **ULT** (Unacceptably Large Time).

In this way, we overcome the fluctuation and make the HT tell us how much time the system takes to hit and surpass the target success rate.

5 Experiments and Results

Three objectives are set for our experiments: (1) Observing the effect of multitask DQN; (2) Contrasting the performances of different teaching schemes (strategies and heuristics) under the companion teaching framework; (3) Observing the safety and efficiency issues under sparse user feedback scenarios.

5.1 Experimental Setup

Our experiments are conducted with the Dialogue State Tracking Challenge 2 (DSTC2) dataset, which is on restaurant information domain (Henderson and Thomson, 2014). The human user is emulated by an agenda-based user simulator with error model (Schatzmann et al., 2007), while the human teacher is emulated by a pre-trained policy model with success rate of about 0.78 through multitask DQN approach without teaching. A rule-based tracker is used for dialogue state tracking (Sun et al., 2014b). The semantic parser is implemented according to an SVM-based method proposed by Henderson et al. (2012). The natural language generator is implemented and modified based on an RNNLG toolkit (Wen et al., 2016, 2015a,c,b).

Early	Rand	SIT	SUT	FPT
None	$p_r = 0.6$	$t_{si} = 5$	$t_{su} = 10$	$\alpha = 1.2$ $w = 25$

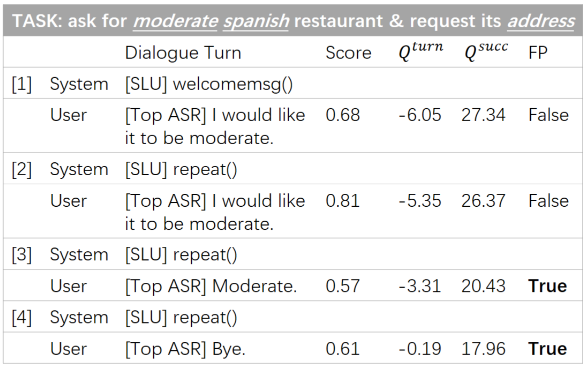
Table 1: Experimental configurations of teaching heuristics introduced in section 2.2 and 3.2.

In our experiments, all dialogues are limited to twenty turns. The "dialogue success" is judged by the user simulator according to whether all user goals are satisfied. And for policy learning, we set a small per-turn penalty of one to encourage short interactions, i.e. $R^{\mathtt{turn}} = -1$, and a large dialogue success reward of thirty to appeal to successful interactions, i.e. $R^{\mathtt{succ}} = 30$, and the discount factor γ is set to one. Table 1 summarizes the heuristics studied in our experiments, together with corresponding configurations which are chosen empirically.

5.2 Observing the Effect of MTL-DQN

The MTL-DQN described in section 3.1 can estimate the prediction of Q^{turn} and Q^{succ} respectively. In our experiments, it was implemented with one shared hidden layer and two dependent hidden layers for two different tasks using MXNet (Chen et al., 2015b).

Figure 3 shows a typical failure in dialogue policy training. The policy showed in the example hasn't been trained well, and it tends to ask the user to repeat over and over again when the confidence score of the user utterance is not high, which causes the user to terminate the dialogue impatiently.



TASK: ask for *moderate spanish* restaurant & request its *address*

		Dialogue Turn	Score	Q^{turn}	Q^{succ}	FP
[1]	System	[SLU] welcomemsg()				
	User	[Top ASR] I would like it to be moderate.	0.68	-6.05	27.34	False
[2]	System	[SLU] repeat()				
	User	[Top ASR] I would like it to be moderate.	0.81	-5.35	26.37	False
[3]	System	[SLU] repeat()				
	User	[Top ASR] Moderate.	0.57	-3.31	20.43	**True**
[4]	System	[SLU] repeat()				
	User	[Top ASR] Bye.	0.61	-0.19	17.96	**True**

Figure 3: An example of failed dialogue while training without teaching. The labels "Score" and "FP" represent for the confidence score of user utterance and the value of *failure prognosis* of the current turn respectively.

This kind of failure can be predicted and corrected in advance. By equation 5, the third turn will be estimated to be *failure prognosis*, which can be a sign for the teacher to intervene and correct the following actions to avoid dialogue failure. Besides, the explicit separate estimation of Q^{turn} and Q^{succ} provides a better understanding of the state of the current turn. For example, although the first turn and second turn have similar Q-values ($Q^{turn}+Q^{succ}$), the latter turn is predicted with less future turns and less possibility to lead to dialogue success. See appendix A for additional successful example.

5.3 Comparing Different Teaching Schemes

Our proposed complete companion teaching framework allows us to teach dialogue systems with different teaching schemes, which consists various strategies and heuristics. In our experiments, we compared 18 schemes consisting of three teaching strategies (CA, EA and EAPC), and six teaching heuristics (Early, Rand, SIT, SUT, FPT and SUT&FPT). The *SUT&FPT* heuristic means the student only ask for advice when equation 2 and 5 are both satisfied. For comparison, we use *No Teaching* (NoTeaching) as the baseline.

To verify the effects of different companion *teaching schemes*, we conduct a set of experiments to see their performances on safety and efficiency dimensions. During training, the teacher can only teach for a limited budget of 1000 turns. All the training curves shown in this paper are moving average curves with a window of size 250 dialogues and over eight runs with an endurable standard error.

5.3.1 Safety Evaluation

To compare effects of different teaching schemes on safety dimension, we use the *Risk Index* (RI) in section 4.1 to quantitatively measure each training process. We set the empirical safety threshold as 65% here. The results are shown in Table 2.

As RIs implies, schemes composed with EAPC strategy is much safer than those composed with other strategies. As for teaching heuristics, FPT, SUT and SUT&FPT are three relatively safer heuristic accompanying different strategies. One exception is that Early teaching looks more suitable for CA. A possible explanation is that when the teacher gives critique earlier, the student will mind its behavior earlier so that increase safety. Figure 4 shows the training curves of on-line learning process under EAPC with various heuristics. Among all 18 teaching schemes, EAPC+SUT is the safest teaching scheme which reduces about 78% risk of no-teaching learning.

	CA	EA	EAPC
Early	**98.5**	110.6	56.1
Rand	193.4	102.4	65.5
FPT	154.4	**86.2**	**53.6**
SIT	230.8	121.7	66.0
SUT	183.5	**95.8**	**44.5***
SUT&FPT	**131.6**	101.8	54.6
NoTeaching	202.9		

Table 2: RIs of learning processes under different teaching schemes. The least risky teaching scheme is annotated with *. For comparing different teaching heuristics with fixed teaching strategy, the smallest RIs in each column are bold and underlined, the 2^{nd} smallest ones are bold only, and the 3^{rd} smallest ones are underlined only. See abbreviations of schemes in section 2.1 and 2.2.

5.3.2 Efficiency Evaluation

We use *Hitting Time* (HT) in section 4.2 to measure the efficiency of learning process under different teaching schemes. The empirical satisfactory target success rate for the student is 70% in our experimental settings.

	CA	EA	EAPC
Early	3390.9	3479.4	4354.7
Rand	3669.0	3518.5	2979.2
FPT	**3089.4**	**2921.1**	**2798.4**
SIT	3576.4	4339.7	3768.7
SUT	3230.4	**2954.5**	3300.2
SUT&FPT	**2890.7**	3393.0	**2702.2***
NoTeaching	3204.1		

Table 3: HTs of test curves of different teaching schemes. The most efficient teaching scheme is annotated with *. For comparing different teaching heuristics with fixed strategy on efficiency issue, the smallest HTs in each column are bold and underlined, and the 2^{nd} smallest bold only. See abbreviations of schemes in section 2.1 and 2.2.

Table 3 contains all HTs of learning process under 18 teaching schemes. Intuitively, The number in the table reflect the number of sessions at which the model achieves target success rate. As it shows, not any teaching scheme will improve the learning efficiency. If the teacher intervenes at an improper time, it will distract system or confuse system even with a right guidance. But teaching when a potential failure exists (F-

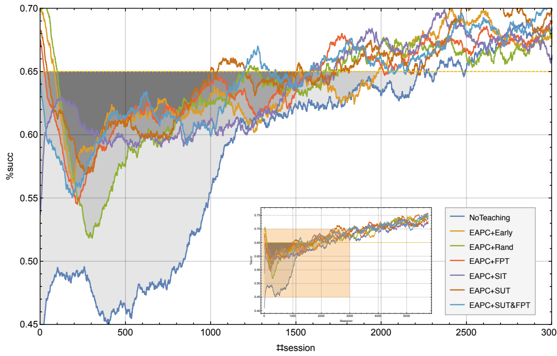


Figure 4: On-line learning process under different teaching schemes (EAPC + different heuristics). The yellow dashed line indicates safe success rate threshold. The area in gray indicates how risky a training process is. See abbreviations of schemes in section 2.1 and 2.2.

PT) is always good for improving learning efficiency. EAPC+SUT&FPT is the teaching scheme that leads to the most efficient learning process in our experiments. Figure 6 gives some example test curves and fitted empirical learning curves of learning process under EAPC with various heuristics.

5.3.3 Teacher's Workload

We also observe teacher's workload of all the teaching schemes since economically utilizing teaching budget is one of our goals.

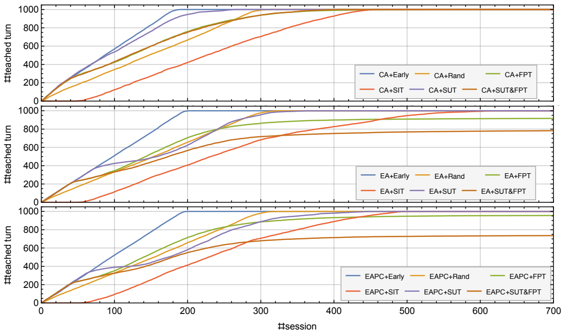


Figure 5: Cumulative usage of teaching budget. The total teaching budget is 1000 for every teaching scheme. See abbreviations of schemes in section 2.1 and 2.2.

Figure 5 illustrates the cumulative usage of teaching budget of 18 teaching schemes. It shows that early teaching is the most costly teaching heuristic so that the teaching budget is soon used up. SIT looks a bit lazy at the beginning and consumes teaching budget slowly. When the teaching strategy is EA or EAPC, FPT-based schemes do not use up full teaching budget in our experiments. Combine SUT and FPT, the workload is relatively lighter than that of teaching in other heuristics. And through proper teaching schemes, we can make better use of the teaching budget and reduce teacher's workload.

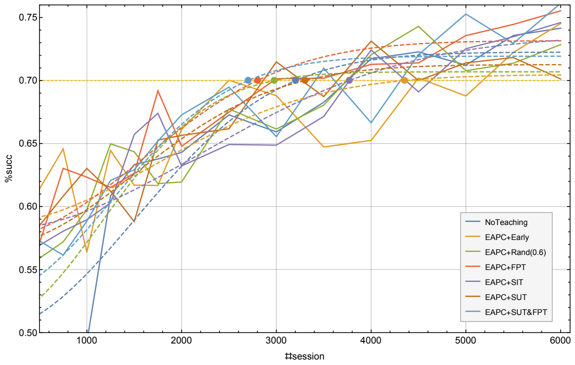


Figure 6: Test curves and fitted empirical learning curves of learning process with different teaching schemes (EAPC+different heuristic). See abbreviations of schemes in section 2.1 and 2.2.

5.4 Safety and Efficiency Issues under Sparse User Feedback Scenarios

In real application scenarios, the user rarely provides feedback at the end of the dialogue, so that safety and efficiency issues are even more serious. To observe the effectiveness of different teaching schemes under sparse user feedback, we conducted experiments with sparse user feedback.

The user feedback rate is set to 30% empirically and experiments are carried out under teaching schemes consisting of EAPC strategies and different heuristics, since EAPC is much safer and more efficient than other teaching strategies.

	RIs	**HTs**
NoTeaching	608.2	ULT
Early	223.0	6881.8
Rand	226.6	ULT
FPT	**171.5**	**6753.0**
SIT	308.8	7868.4
SUT	183.3	**5876.9**
SUT&FPT	**155.4**	8420.9

Table 4: RIs & HTs of learning processes under EAPC strategy and different heuristics when user feedback rate is 30%. See abbreviations of schemes in section 2.1 and 2.2.

Table 4 records the RIs and HTs of those different learning process when user feedback is sparse.

We can see that when the user feedback rate drops from 100% to 30%, the RIs and HTs increase dramatically. The NoTeaching baseline is very risky and inefficient (its hitting time is even unpredictable within 10000 sessions learning). However, with teaching scheme such as EAPC+FPT, both safety and efficiency can be improved a lot.

6 Conclusions and Future Work

This paper addressed the *safety* and *efficiency* issues of sustainable on-line dialogue policy learning with different teaching schemes, which answer both "how" and "when" to teach, within the complete companion teaching framework. To evaluate the policy learning process precisely, we proposed two measurements, Risk Index (RI) and Hitting Time (HT), to quantify the degree of safety and efficiency. Particularly, through multitask learning, we managed to optimize the predicted remaining turns and dialogue success reward explicitly, based on which we developed a novel *Failure Prognosis based Teaching* (FPT) heuristic to better utilize the fixed teaching budget and make the teaching affordable.

Experiments showed that different teaching schemes have different effects on safety and efficiency dimension. And they also require different workload of the teacher. Among 18 compared teaching schemes, FPT-based heuristics combined with EAPC strategy achieved promising performance on RI and HT, and required relatively slight workload. This result indicates a proper teaching scheme under the companion teaching framework is able to guarantee a sustainable and affordable on-line dialogue policy learning process.

There are several directions for our future work. We expect to deploy our proposed framework in real-world scenarios collaborating with real human teachers to verify the results presented in this paper and discover more potential challenges of on-line dialogue system development. Furthermore, the current study is focused on dialogue success rate, which is a simplification of the human satisfaction evaluation. So future work is needed to take more qualities into consideration to achieve better user experience.

Acknowledgments

This work was supported by the Shanghai Sailing Program No. 16YF1405300, the China NSFC projects (No. 61573241 and No. 61603252) and the Interdisciplinary Program (14JCZ03) of Shanghai Jiao Tong University in China. Experiments have been carried out on the PI supercomputer at Shanghai Jiao Tong University.

References

Ofra Amir, Ece Kamar, Andrey Kolobov, and Barbara Grosz. 2016. Interactive teaching strategies for agent training. In *IJCAI*. International Joint Conferences on Artificial Intelligence.

Benedict Arnold. 1998. Reinforcement learning: An introduction (adaptive computation and machine learning). *IEEE Transactions on Neural Networks*, 9(5):1054.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Lu Chen, Pei Hao Su, and Milica Gasic. 2015a. Hyperparameter optimisation of gaussian process reinforcement learning for statistical dialogue management. In *Meeting of the Special Interest Group on Discourse and Dialogue*, pages 407–411.

Lu Chen, Runzhe Yang, Cheng Chang, Zihao Ye, Xiang Zhou, and Kai Yu. 2017. On-line dialogue policy learning with companion teaching. In *EACL*.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015b. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *Statistics*.

Jeffery Allen Clouse. 1996. On integrating apprentice learning and reinforcement learning. University of Massachusetts.

L Daubigney, M Geist, S Chandramohan, and O Pietquin. 2012. A comprehensive reinforcement learning framework for dialogue management optimization. *IEEE Journal of Selected Topics in Signal Processing*, 6(8):891–902.

Mehdi Fatemi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman. 2016. Policy networks with two-stage training for dialogue systems. *arXiv.org*.

Milica Gašić, Filip Jurčíček, Simon Keizer, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. Gaussian processes for fast policy optimisation of pomdp-based dialogue managers. pages 201–204.

Matthew Henderson, Milica Gai, Blaise Thomson, and Pirros Tsiakoulis. 2012. Discriminative spoken language understanding using word confusion networks. In *Spoken Language Technology Workshop*, pages 176–181.

Matthew Henderson and Blaise Thomson. 2014. The second dialog state tracking challenge. In *SIGDIAL*, volume 263, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kshitij Judah, Saikat Roy, Alan Fern, and Thomas G Dietterich. 2010. Reinforcement learning via practice and critique advice. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 481–486.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134.

E Levin, R Pieraccini, and W Eckert. 1997. Learning dialogue strategies within the markov decision process framework. In *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pages 72–79.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *Computer Science*.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *NAACL*, pages 149–152, Morristown, NJ, USA. Association for Computational Linguistics.

Pei Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-barahona, Stefan Ultes, David Vandyke, Tsung Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. In *Meeting of the Association for Computational Linguistics*, pages 2431–2441.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. 2014a. The sjtu system for dialog state tracking challenge 2. In *Meeting of the Special Interest Group on Discourse and Dialogue*, pages 318–326.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. 2014b. The sjtu system for dialog state tracking challenge 2. In *SIGDIAL*, pages 318–326, Stroudsburg, PA, USA. Association for Computational Linguistics.

A. L. Thomaz and C. Breazeal. 2008. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6):716–737.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech and Language*, 24(4):562–588.

Lisa Torrey and Matthew Taylor. 2013. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1053–1060. International Foundation for Autonomous Agents and Multiagent Systems.

Zhuoran Wang and Oliver Lemon. 2013. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *The Sigdial Meeting on Discourse and Dialogue*.

Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic Language Generation in Dialogue using Recurrent Neural Networks with Convolutional Sentence Reranking. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Toward multi-domain language generation using recurrent neural networks. *NIPS Workshop on Machine Learning for Spoken Language Understanding and Interaction*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *Proceedings of the 2016 Conference on North American Chapter of the Association for Computational Linguistics (NAACL)*. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015c. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422.

Jason D Williams and Geoffrey Zweig. 2016. End-to-end LSTM-based dialog control optimized with supervised and reinforcement learning. *CoRR*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Generating High-Quality and Informative Conversation Responses with Sequence-to-Sequence Models

Louis Shao[1]*, **Stephan Gouws**[3]*, **Denny Britz**[2]†, **Anna Goldie**[2], **Brian Strope**[1], **Ray Kurzweil**[1]
{overmind,sgouws,dennybritz,agoldie,bps,raykurzweil}@google.com
[1]Google Research and [2]Google Brain
Mountain View, CA, USA
and
[3]Google Brain
London, UK

Abstract

Sequence-to-sequence models have been applied to the conversation response generation problem where the source sequence is the conversation history and the target sequence is the response. Unlike translation, conversation responding is inherently creative. The generation of long, informative, coherent, and diverse responses remains a hard task. In this work, we focus on the single turn setting. We add self-attention to the decoder to maintain coherence in longer responses, and we propose a practical approach, called the glimpse-model, for scaling to large datasets. We introduce a stochastic beam-search algorithm with segment-by-segment reranking which lets us inject diversity earlier in the generation process. We trained on a combined data set of over 2.3B conversation messages mined from the web. In human evaluation studies, our method produces longer responses overall, with a higher proportion rated as acceptable and excellent as length increases, compared to baseline sequence-to-sequence models with explicit length-promotion. A back-off strategy produces better responses overall, in the full spectrum of lengths.

1 Introduction

Building computer systems capable of general-purpose conversation is a challenging problem. However, it is a necessary step toward building intelligent agents that can interact with humans via natural language, and for eventually passing the Turing test. The sequence-to-sequence (*seq2seq*) model has proven very popular as a purely data-driven approach in domains that can be cast as learning to map to and from variable-length sequences, with state-of-the art results in many domains, including machine translation (Cho et al., 2014; Sutskever et al., 2014; Wu et al., 2016). Neural conversation models are the latest development in the domain of conversation modeling, with the promise of training computers to converse in an end-to-end fashion (Vinyals and Le, 2015; Shang et al., 2015; Sordoni et al., 2015; Wen et al., 2016). Despite promising results, there are still many challenges with this approach. In particular, these models produce short, generic responses that lack diversity (Sordoni et al., 2015; Li et al., 2015). Even when longer responses are explicitly encouraged (e.g. via length normalization), they tend to be incoherent (*"The sun is in the center of the sun."*), redundant (*"i like cake and cake"*), or contradictory (*"I don't own a gun, but I do own a gun."*).

In this paper, we provide two methods to address these issues with minimal modifications to the standard seq2seq model. First, we present a *glimpse model* that only trains on fixed-length segments of the target-side at a time, allowing us to scale up training to larger data sets. Second, we introduce a *segment-based stochastic decoding technique* which injects diversity earlier in the generated responses. Together, we find that these two methods lead to both longer responses and higher ratings, compared to a baseline seq2seq model with explicit length and diversity-promoting heuristics integrated into the generation procedure (see Table 1 for examples generated using our model).

In Section 2, we present a high-level overview of these two techniques. We then discuss each

[*] Both authors contributed equally to this work.

[†] Work done as a member of the Google Brain Residency program (g.co/brainresidency).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2210–2219
Copenhagen, Denmark, September 7–11, 2017.

technique in more detail in Sections 3 and 4. Finally, we report small and large-scale experimental evaluations of the proposed techniques in Section 5.

2 Overview and Motivation

A major difference between translation and responding to conversations is that, in the former, the high-level semantic content to generate in the target sequence $\mathbf{y}$ is completely given by the source sequence, *i.e.*, given the source $\mathbf{x}$, there is low conditional entropy in the target distribution $P(\mathbf{y}|\mathbf{x})$. In the seq2seq approach, the decoder network therefore only has to keep track of where it is in the output, and the content to generate can be transformed from the relevant parts in the source via the attention mechanism (Bahdanau et al., 2014). In contrast, in conversation response generation, the prompt turn may be short and general (e.g., *"what do you have planned tonight"*), while an appropriate response may be long and informative.

The standard seq2seq model struggles with generating long responses, since the decoder has to keep track of everything output so far in its fixed-length hidden state vector, which leads to incoherent or even contradictory outputs. To combat this, we propose to integrate *target-side attention* into the decoder network, so it can keep track of what has been output so far. This frees up capacity in the hidden state for modeling the higher-level semantics required during the generation of coherent longer responses. We were able to achieve small perplexity gains using this idea on the small OpenSubtitles 2009 data set (Tiedemann, 2009). However, we found it to be too memory-intensive when scaling up to larger data sets.

As a trade-off, we propose a technique (called the 'glimpse model') which interpolates between source-side-only attention on the encoder, and source and target-side attention on the encoder and decoder, respectively. Our solution simply trains the decoder on fixed-length *glimpses* from the target side, while having both the source sequence and the part of the target sequence before the glimpse on the encoder, thereby sharing the attention mechanism on the encoder. This can be implemented as a simple data-preprocessing technique with an unmodified standard seq2seq implementation, and allows us to scale training to very large data sets without running into any memory issues. See Figure 1 for a graphical overview, where we illustrate this idea with a glimpse-model of length 3.

Given such a trained model, the next challenge is how to generate long, coherent, and diverse responses with the model. As observed in the previous section and in other work, standard maximum a posteriori (MAP) decoding using beam search often yields short, uninformative, and high-frequency responses. One approach to produce longer outputs is to employ length-promoting heuristics (such as length-normalization (Wu et al., 2016)) during decoding. We find this increases the length of the outputs, however often at the expense of coherence. Another approach to explicitly create variation in the generated responses is to rerank the N-best MAP-decoded list of responses from the model using diversity-promoting heuristics (Li et al., 2015) or a backward RNN (Wen et al., 2015). We find this works for shorter responses, but not for long responses, primarily for two reasons: First, the method relies on the MAP-decoding to produce the N-best list, and as mentioned above, MAP-decoding prefers short, generic responses. Second, it is too late to delay reranking in the beam search until the whole sequence has been generated, since beam-search decoding tends to yield beams with low diversity per given prompt, even when the number of beams is high. Instead, our solution is to break up the reranking over shorter segments, and to rerank segment-by-segment, thereby injecting diversity earlier during the decoding process, where it has the most impact on the resulting diversity of the generated beams.

To further improve variation in the generated responses, we replace the deterministic MAP-decoding of the beam search procedure with sampling. If a model successfully captures the distribution of responses given targets, one can expect simple greedy sampling to produce reasonable responses. However, due to model underfitting, the learned distributions are often not sharp enough, causing step-by-step sampling to accumulate errors along the way, manifesting as incoherent outputs. We find that integrating sampling into the beam-search procedure yields responses that are more coherent and with more variation overall.

In summary, the contributions of this work are the following:

1. We propose to integrate target-side attention in neural conversation models, and provide a

PROMPT:	**RESPONSE FROM BASELINE (A) vs OUR METHOD (B)**
what do you have planned tonight	**A:** I have no idea what I want to do with my life. I have no idea what I want to (Bad, Mediocre, Bad) **B:** I was planning on doing a few of these, but I'm not sure what to choose. I need to find a place to camp, and I don't have a lot of time to make a plan. (Excellent, Good, Mediocre)
I'd like to read a good science fiction novel. Haven't read scifi in a while.	**A:** If you like sci fi you should check out The Name of the Wind by Patrick Rothfuss (Excellent, Excellent, Excellent) **B:** If you want to know what sci-fi is, read the first book by Neil Gaiman. It's a sci-fi sci-fi sci-fi novel. It is really good and I recommend it to all of you. It has a great premise. (Acceptable, Excellent, Acceptable)
Got any big weekend plans?	**A:** I have plans for the weekend but I don't know if I will be able to make it to the (Bad, Mediocre, Bad) **B:** I don't really have much to do on Saturday. But I plan on going to the movies at the mall and going to the show. (Good, Good, Bad)
Sometimes flying messes my ears, because of the air pressure. Especially if I have a cold or something, flying gets me messed up.	**A:** I don't know why, but when I was a kid I used to do this all the time when I was younger. (Good, Bad, Bad) **B:** I have no idea why, but I feel exactly the same way. I don't know why, but my brain is so sensitive to air that it makes my nose twitch. (Excellent, Mediocre, Good)
Yeah. I have this friend, and we get bored just hanging out here sometimes, so we take little trips to places. We went to Seattle a couple of months ago.	**A:** What kind of things do you like to do? (Excellent, Excellent, Excellent) **B:** That sounds like a great time. I have a lot of friends that go to Seattle, so I was wondering if you could share any of the locations? (Excellent, Excellent, Excellent)
have you seen Suicide Squad?	**A:** I have not, but I will check it out. (Good, Good, Good) **B:** I saw it in a theater once and it was great. It was very... I don't know, a little dark. I like the psychological effects and the way it portrays the characters in the movie. (Excellent, Excellent, Excellent)
Actually I like the snow too, except for today. It's all slushy and it's all over the sidewalks and it's gross.	**A:** I hate the snow. I hate the snow. I hate the snow. I hate the snow. I hate the snow. I hate the snow. I hate the snow. I hate the snow. (Acceptable, Good, Good) **B:** I'm in the same boat as you. I hate it when it's snowy and I don't like to shovel it all day. :(So this is a perfect excuse to not get out of bed and go somewhere else. (Excellent, Good, Acceptable)
What do you hope to see in the future?	**A:** I want to be able to go back in time to when I was born. I want to be (Mediocre, Bad, Mediocre) **B:** I want to see an actual movie where the characters have the ability to hide the fact that they are actually in the movie. (Good, Acceptable, Acceptable)
what dangers could happen with the singularity?	**A:** I don't think it would be that big of a deal. (Acceptable, Mediocre, Acceptable) **B:** If you look at it from the perspective of an alien, you can't say that there aren't any alternative ways to deal with it. (Acceptable, Mediocre, Acceptable)
I think you're a little crazy.	**A:** I don't think you know what that word means. (Acceptable, Good, Good) **B:** I am, actually. I don't think I'm crazy though. I certainly am. (Acceptable, Mediocre, Acceptable)

Table 1: Some example responses generated from the baseline (**A**) compared to our model (**B**), with human ratings in parentheses. These were selected for quality. See the appendix for an unbiased sample.

practical approach, referred to as the **glimpse model**, which scales well and is easy to implement on top of the standard sequence-to-sequence model.

2. We introduce a **stochastic beam-search procedure** with segment-by-segment reranking which improves the diversity of the generated responses.

3. We present large-scale experiments with human evaluations showing the proposed techniques improve over strong baselines.

4. We release our collection of context-free conversation prompts used in our evaluations as a benchmark for future open-domain conversation response research.

3 Seq2Seq Model with Attention on Target

We discuss conversation response generation in the sequence-to-sequence problem setting. In this setting, there is a source sequence $\mathbf{x} = (x_1, x_2, ..., x_M)$, and a target sequence $\mathbf{y} = (y_0, y_1, y_2, ..., y_N)$. We assume y_0 is always the start-of-sequence token and y_N is the end-of-sequence token. In a typical sequence-to-sequence model, the encoder gets its input from the source sequence $\mathbf{x}$ and the decoder models the conditional language model $P(\mathbf{y}|\mathbf{x})$ of the target sequence $\mathbf{y}$, given $\mathbf{x}$.

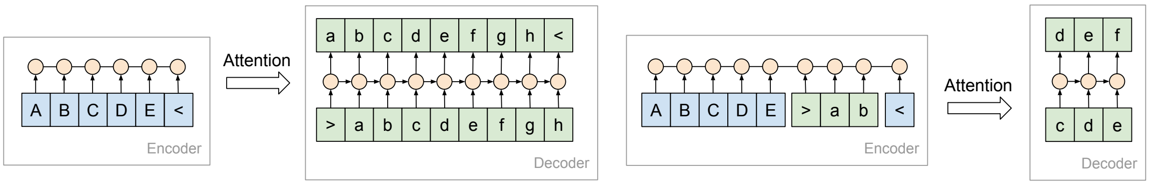
Seq2seq models with attention (Bahdanau et al., 2014) parameterize the per-symbol conditional probability as:

$$P\left(y_i|\mathbf{y}_{[0:i-1]};\mathbf{x}\right) = \mathrm{DecoderRNN}\,(\\ y_{i-1}, h_{i-1}, \mathrm{Attention}\left(h_{i-1}, \mathbf{x}\right)) \quad (1)$$

for $1 \leq i \leq N$, where DecoderRNN() is a recurrent neural network that map the sequence of decoder symbols into fixed-length vectors, and Attention() is a function that yields a fixed-size vector summary of the encoder symbols $\mathbf{x}$ (the 'focus') most relevant to predicting y_i, given the previous recurrent state of the network h_{i-1} (the 'context'). The full conditional probability follows from the product rule, as:

$$P\left(\mathbf{y}|\mathbf{x}\right) = \prod_{i=1}^{N} P\left(y_i|\mathbf{y}_{[0:i-1]};\mathbf{x}\right) \quad (2)$$

We propose to implement **target-side attention** by augmenting the attention mechanism to include the part of the target sequence already generated, *i.e.*, we include $\mathbf{y}_{[0:i-2]}$ in the arguments to the attention function: Attention$(h_{i-1}, \mathbf{y}_{[0:i-2]}, \mathbf{x})$. We implemented this in TensorFlow (Abadi et al., 2015) using 3 LSTM layers on both the encoder and the decoder, with 1024 units per layer. We experimented on the OpenSubtitles 2009 data set, and obtained a small perplexity gain from the target-side attention: 24.6 without versus 24.2 with. However, OpenSubtitles is a small data set,



(a) The vanilla sequence-to-sequence model.

(b) Length-3 Target-glimpse Model

Figure 1: The vanilla seq2seq with attention on the left, and our proposed target-glimpse model on the right. The symbol ">" and "<" are start-of-sequence and end-of-sequence, respectively.

and the majority of its response sequences are shorter than 10 tokens. This may prevent us from seeing bigger gains, since our method is designed to help with longer outputs. In order to train on the much larger Reddit data set, we implemented this method on top of the GNMT model (Wu et al., 2016). Unfortunately, we met with frequent out-of-memory issues, as the 8-layer GNMT model is already very memory-intensive, and adding target-side attention made it even more so. Ideally, we would like to retain the model's capacity in order to train a rich response model, and therefore a more efficient approach is necessary.

To this end, we propose the **target-glimpse model** which has a fixed-length decoder. The target-glimpse model is implemented as a standard sequence-to-sequence model with attention, where the decoder has a fixed length K. During training, we split the target sequence into non-overlapping, contiguous segments (*glimpses*) with fixed length K, starting from the beginning. We then train on each of these glimpses, one at a time on the decoder, while putting all target-side symbols before the glimpse on the encoder. For example, if a sequence $\mathbf{y}$ is split into two glimpses $\mathbf{y}_1$ and $\mathbf{y}_2$, each with length K ($\mathbf{y}_2$ may be shorter than K), then we will train the model with two examples, $(\mathbf{x} \rightarrow \mathbf{y}_1)$, and $(\mathbf{x}, \mathbf{y}_1 \rightarrow \mathbf{y}_2)$. Each time the concatenated sequence on the left of the arrow is put on the encoder and the sequence on the right is put on the decoder. Figure 1(b) illustrates the training of $(\mathbf{x}, \mathbf{y}_1 \rightarrow \mathbf{y}_2)$ when $K = 3$. In our implementation, we always put the source-side end-of-sequence token at the end of the whole encoder sequence, and we split the glimpses according to the decoder time steps. For example, if the sequence $\mathbf{y}$ is $y_0, y_1, y_2, ..., y_{10}$, and $K = 3$, the first example will have y_0, y_1, y_2 on the input layer of the decoder, and y_1, y_2, y_3 on the output layer of the decoder. The second example has y_3, y_4, y_5 as input of the decoder and y_4, y_5, y_6 as the output of the decoder, and so on. In our experiments, we use $K = 10$.

While decoding each glimpse, the decoder therefore attends to both the source sequence and the part of the target sequence that precedes the glimpse, thereby benefiting from the GNMT encoder's bidirectional RNN. Through generalization, the decoder should learn to decode a glimpse of length K in any arbitrary position of the target sequence (which we will exploit in our decoding technique discussed in Section 4). One drawback of this model, however, is that the context inputs to the attention mechanism only include the words that have been generated so far in this glimpse, rather than the words from the full target side. The workaround that we use is to simply connect the last hidden state of the GNMT-encoder to the initial hidden state of the decoder[1], thereby giving the decoder access to all previous symbols regardless of the starting position of the glimpse.

4 Stochastic Decoding with Segment-by-Segment Reranking

We now turn our attention from training to inference (decoding). Our strategy is to perform reranking with a normalized score at the segment level, where we generate the candidate segments using a trained glimpse-model and using a stochastic beam search procedure, which we discuss next. The full decoding algorithm proceeds segment by segment.

The standard beam search algorithm generates symbols step-by-step by keeping a set of the B highest-scoring beams generated so far at each step[2]. The algorithm adds all possible single-token extensions to every existing beam, and then selects

[1] This is the default in standard seq2seq models, but not in the GNMT model.

[2] Beams are also called 'hypotheses', and B is referred to as the 'beam width'.

the top B beams. In our stochastic beam search algorithm, we replace this deterministic top-B selection by a stochastic sampling operation in order to encourage variation. Further, to discourage a single beam from dominating the search and decreasing the final response diversity, we perform a two-step sampling procedure: 1) For each single-token extension of an individual beam we don't enumerate all possibilities, but instead sample a fixed number of D candidate tokens to be added to the beam. This yields a total of $B \times D$ beams, each with one additional symbol. 2) We then compute the accumulated conditional log-probabilities for each beam (normalized across all $B \times D$ beams), and treat these as the logits for sub-sampling B beams for the next step. We repeat this procedure until we reach the desired segment-length H, or until a segment ends with the end-of-sequence token.

For a given source sequence, we can use this stochastic beam search algorithm to generate B candidate H-length segments as the beginning of the target sequence. We then perform a reranking step (described below), and keep one of these. The concatenation of the source and the first target segment is then used as the input for generating the next B candidate segments. The algorithm continues until the segment selected ends with an end-of-sequence token.

This algorithm behaves similarly to standard beam search when the categorical distribution used during the process is sharp ('peaked'), since the samples are likely to be the top categories (words) . However, when the distribution is smooth, many of the choices are likely. In conversation response generation we are dealing with a conditional probability model with high entropy, so this is what often happens in practice.

For the reranking, we normalize the scores using random prompts. In particular, suppose $\mathbf{y}_k = y_1, ..., y_{k-1}$ is a candidate segment, and $(\mathbf{x}, \mathbf{y}_{1:k-1})$ is the input to the stochastic beam search. The normalized score is then computed as follows:

$$S\left(\mathbf{y}_k|\mathbf{x}, \mathbf{y}_{1:k-1}\right) = \frac{P\left(\mathbf{y}_k|\mathbf{x}, \mathbf{y}_{1:k-1}\right)}{\sum_{\mathbf{x}' \in \Phi} P\left(\mathbf{y}_k|\mathbf{x}', \mathbf{y}_{1:k-1}\right)} \tag{3}$$

In this equation, the set Φ is a collection of randomly sampled source sequences (prompts). In our experiments, we randomly select Q prompts from the context-free evaluation set (introduced in the Experiments section).

It is worth noting that when Φ is an unbiased sample from $P(\mathbf{x})$, the summation in the denominator is a Monte-Carlo approximation of $P(\mathbf{y}_k|\mathbf{y}_{1:k-1})$. In the case of reranking whole target sequences $\mathbf{y}$, this becomes the marginal $P(\mathbf{y})$, which corresponds to the same diversity-promoting objective used in (Li et al., 2015). However, we found that our approximation works better in terms of N-choose-1 accuracy (see Section 5.2), which suggests that its value may be closer to the true conditional probability.

In our experiments, we set number of random prompts Q to 15, segment length H to 10, number of beams B to 2, and samples per beam D to 10. We select a small value for B, since we find that larger values makes the algorithm behave more like standard beam search.

5 Experimental Results

In this section we present experimental results for evaluating the target-glimpse model and the stochastic decoding method that we presented. We train the model using the Google neural machine translation model (GNMT, (Wu et al., 2016)), on a data set that combines multiple sources mined from the Web:

1. The full Reddit data[3] that contains 1.7 billion messages (221 million conversations).
2. The 2009 Open Subtitles data (0.5 million conversations, (Tiedemann, 2009)).
3. The Stack Exchange data (0.8 million conversations).
4. Dialogue-like texts that we recognized and extracted from the web (17 million conversations).

For all these data sets, we extract pairs of messages where one can be considered as a response to the other. For example, in the Reddit data set, the messages belonging to the same post are organized as a tree. A child node is a message that replies to its parent. This may not necessarily be true as people may be replying to other messages that are also visually close. However, for our current single-turn experiments, we treat these as a single exchange.

[3] Download links are at https://redd.it/3bxlg7

In this setting, the GNMT model trained on prompt-to-response pairs works surprisingly well without modification when generating short responses with beam search. Similar to previous work on neural conversation models, we find that the generated responses are almost always grammatical, and sometimes even interesting. They are also usually on topic. In addition, we found that even greedy sampling from the 8-layer GNMT model produces grammatical responses most of the time, although these responses are more likely to be semantically-broken than responses generated using standard beam search. We would like to leverage the benefits of greedy sampling, because the induced variation generates more surprises and may potentially help improve user-engagement, and we found that our proposed segment-based beam sampling procedure accomplishes this to some extent.

5.1 Evaluation Metric

It is difficult to come up with an objective evaluation metric for conversation response generation that can be computed automatically. The conditional distribution $P(\mathbf{y}|\mathbf{x})$ is supposed to have high entropy in order to be interesting (many possible valid responses to a given prompt). Therefore BLEU scores used in translation are not a good fit (also see (Liu et al., 2016)). Other than looking at the evaluation set perplexity, we use two metrics, the **N-choose-1 accuracy** and **5-scale side-by-side human evaluation**. In the N-choose-K metric, we use the model as a retriever. Given a prompt, we ask the model to rank N candidate responses, where one is the ground truth and the other $N-1$ are random responses from the same data set. We then calculate the N-choose-K accuracy as the proportion of trials where the true response is in the top K. The prompts used for evaluation are selected randomly from the same data set. This metric isn't necessarily correlated well with the true response quality, but provides a useful first diagnostic for faster experimental iteration. It takes about a day to train a small model on a single GPU that reaches 2-choose-1 accuracies of around 70% or 80%, but it is much harder to make progress on the 50-choose-1 accuracy. As a reference, human performance on the 10-choose-1 task is around 45% accuracy.

In the 5-scale human evaluation, we use a collection of 200 context-free prompts[4]. These prompts are collected from the following sources, and filtered to prompts that are context-free (*i.e.* do not depend on previous turns in the conversation), general enough, and by eliminating near duplicates:

1. The questions and statements that users asked an internal testing bot.
2. The Fisher corpus (David et al., 2004).
3. User inputs to the Jabberwacky chatbot[5].

These can be either generic or specific. Some example prompts from this collection are shown in Table 1. These prompts are open-domain (not about any specific topic), and include a wide range of topics. Many require some creativity for answering, such as *"Tell me a story about a bear."* Our evaluation set is therefore not from the same distribution as our training set. However, since our goal is to produce good general conversation responses, we found it to be a good general purpose evaluation set.

The evaluation itself is done by human raters. They are well-trained for the purpose of ensuring rating quality, and they are native English speakers. The A 5-scale rating is produced for each prompt-response pair: *Excellent*, *Good*, *Acceptable*, *Mediocre*, and *Bad*. For example, the instructions for rating *Excellent* is "On topic, interesting, shows understanding, moves the conversation forward. It answers the question." The instruction for *Acceptable* is "On topic but with flaws that make it seem like it didnt come from a human. It implies an answer." The instruction for *Bad* is "A completely off-topic statement or question, nonsensical, or grammatically broken. It does not provide an answer."

In our experiments, we perform the evaluations side-by-side, each time using responses generated from two methods. Every prompt-response pair is rated by three raters. We rate 200 pairs in total for every method, garnering 600 ratings overall. After the evaluation, we report aggregated results from each method individually.

5.2 Motivating Experiments

To see whether generating long responses is indeed a challenging problem, we trained the plain

[4] This list will be released to the community.

[5] http://www.jabberwacky.com/

seq2seq with the GNMT model where the encoder holds the source sequence and the decoder holds the target sequence. We experimented with the standard beam search and the beam search with length normalization $\alpha = 0.8$ similar to (Wu et al., 2016). With this length normalization the generated responses are indeed longer. However, they are more often semantically incoherent. It produces *"I have no idea what you are talking about."* more often, similarly observed in (Li et al., 2016). The human evaluation results are summarized in Figure 2(b). Methods that generate longer responses have more *Bad* and less *Excellent* / *Good* ratings.

We also performed the N-choose-1 evaluation on the baseline model using different normalization schemes. The results are shown in Table 2(a). *No Normalization* means that we use $P(\mathbf{y}|\mathbf{x})$ for scoring, *Normalize by Marginal* uses $P(\mathbf{y}|\mathbf{x})/P(\mathbf{y})$, as suggested in (Li et al., 2015), and *Normalize by Random Prompts* is our scoring objective described in Section 4. The significant boost when using both normalization schemes indicates that the conditional log probability predicted by the model may be biased towards the language model probability of $P(\mathbf{y})$. After adding the normalization, the score may be closer to the true conditional log probability.

Overall, this *reranking* evaluation indicates that our heuristic is preferred to scoring using the marginal. However, it is unfortunately hard to directly make use of this score during beam search decoding (*i.e., generation*), since the resulting sequences are usually ungrammatical, as also observed by (Li et al., 2015). This is the motivation for using a segment-by-segment reranking procedure, as described in Section 4.

5.3 Large-Scale Experiments

For our large-scale experiments, we train our target-glimpse model on the full combined data set. Figure 2(d) shows the training progress curve. In this figure, we also include the curve for $K = 1$, that is, the glimpse model with decoder-length 1. It is clear enough that this model progresses much slower, so we terminated it early. However, it is surprising that the glimpse model with $K = 10$ progresses faster than the baseline model with only source-side attention, because the model is trained on examples with decoder-length fixed at 10, while the average response length is 38 in our data set. This means it takes on average 3.8x training steps for the glimpse model to train on the same number of raw training-pairs as the baseline model. Despite this, the faster progress indicates that target-side attention indeed helps the model generalize better.

The human evaluation results shown in Figure 2 compare our proposed method with the baseline seq2seq model. For this, we trained a length-10 target-glimpse model and decoded with stochastic beam-search using segment-by-segment reranking. In our experiments, we were unable to generate better long, coherent responses using the whole-sequence level reranking method from (Li et al., 2015) compared to using standard beam search with length-normalization[6]. We therefore choose the latter as our baseline, because it is the only method which generates responses that are long enough that we can compare to.

Figure 2 shows that our proposed method generates more long responses overall. One third of all responses are longer than 100 characters, while the baseline model produces only a negligible fraction. Although we do not employ any length-promoting objectives in our method, length-normalization is used for the baseline. For responses generated by our method, the proportion of *Acceptable* and *Excellent* responses remains constant or even increases as the responses grow longer. Conversely, human ratings decline sharply with length for the baseline model.

The percentage of test cases with major agreement is high for both methods. We consider a test to have major agreement if two ratings out of the three are the same. For the baseline method, 80% of the responses have major agreements, and for our method it is 70%.

However, shorter responses have a much smaller search space, and we find that standard beam search tends to generate better ("safer") short responses. To maximize cumulative response quality, we therefore implemented a back-off strategy that combines the strengths of the two methods. We choose to fallback to the baseline model without length normalization when the latter produces a response shorter than 40 characters, otherwise we use the response from our method. This corresponds to the white histogram in Figure 2(b). Compared to the other methods in the fig-

[6] This is because the method reranks the responses in the N-best list resulting from the beam search, which tend to be short with not much variation to begin with.

ure, the combined strategy results in more ratings of *Excellent*, *Good*, *Acceptable*, and *Mediocre*, and fewer *Bad* ratings. With this strategy, among the responses generated for the same 200 prompts, 133 were from the standard beam search and 67 were from our model. Out of the 67 long responses, two thirds were longer than 60 characters and half were longer than 75 characters. To compare the combined model's performance with the baseline, we generated responses from both models using the same 200 prompts. For 20 of the response pairs, human raters had no preference, but for the remaining 180, human raters preferred the combined model's response in 103 cases and the baseline's in only 77, indicating a significant win.

6 Conclusion

The research of building end-to-end systems that can engage in general-purpose conversation is still in its infancy. More significant progress is expected to be made with more advanced neural architectures. However, our results reported in this paper show that minimal modeling change and a slightly more advanced decoding technique, combined with training over very large data sets, can still lead to noticeable improvements in the quality of responses generated using neural conversation models. Overall, we found using fixed-lengths in the decoder to make it easier to train on large data sets, as well as to allow us to improve the diversity and coherence of the generated responses earlier during generation, when it has most impact. While the focus of this work has been on conversation modeling, we expect some of these results to carry over to other sequence-to-sequence settings, such as machine translation or image-captioning.

Acknowledgments

We would like to thank Quoc Le, Oriol Vinyals and Jakob Uszkoreit for many helpful discussions, and Scott Benson, Fuchun Peng for collecting the context-free prompt set, and Amin Ahmad for setting up the human evaluation, and Rami Eid, Daniel Cer for collecting training data sets, and Yonghui Wu, Zhifeng Chen, Mike Schuster for help on training the GNMT model.

References

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

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

Kyunghyun Cho, Bart Van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734.

Christopher Cieri David, David Miller, and Kevin Walker. 2004. The fisher corpus: a resource for the next generations of speech-to-text. In *in Proceedings 4th International Conference on Language Resources and Evaluation*. pages 69–71.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* .

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *CoRR* abs/1606.01541.

Chia-Wei Liu, Ryan Lowe, Iulian Vlad Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*. ACL, Austin, Texas, page 21222132. https://aclweb.org/anthology/D16-1230.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364* .

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714* .

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pages 3104–3112.

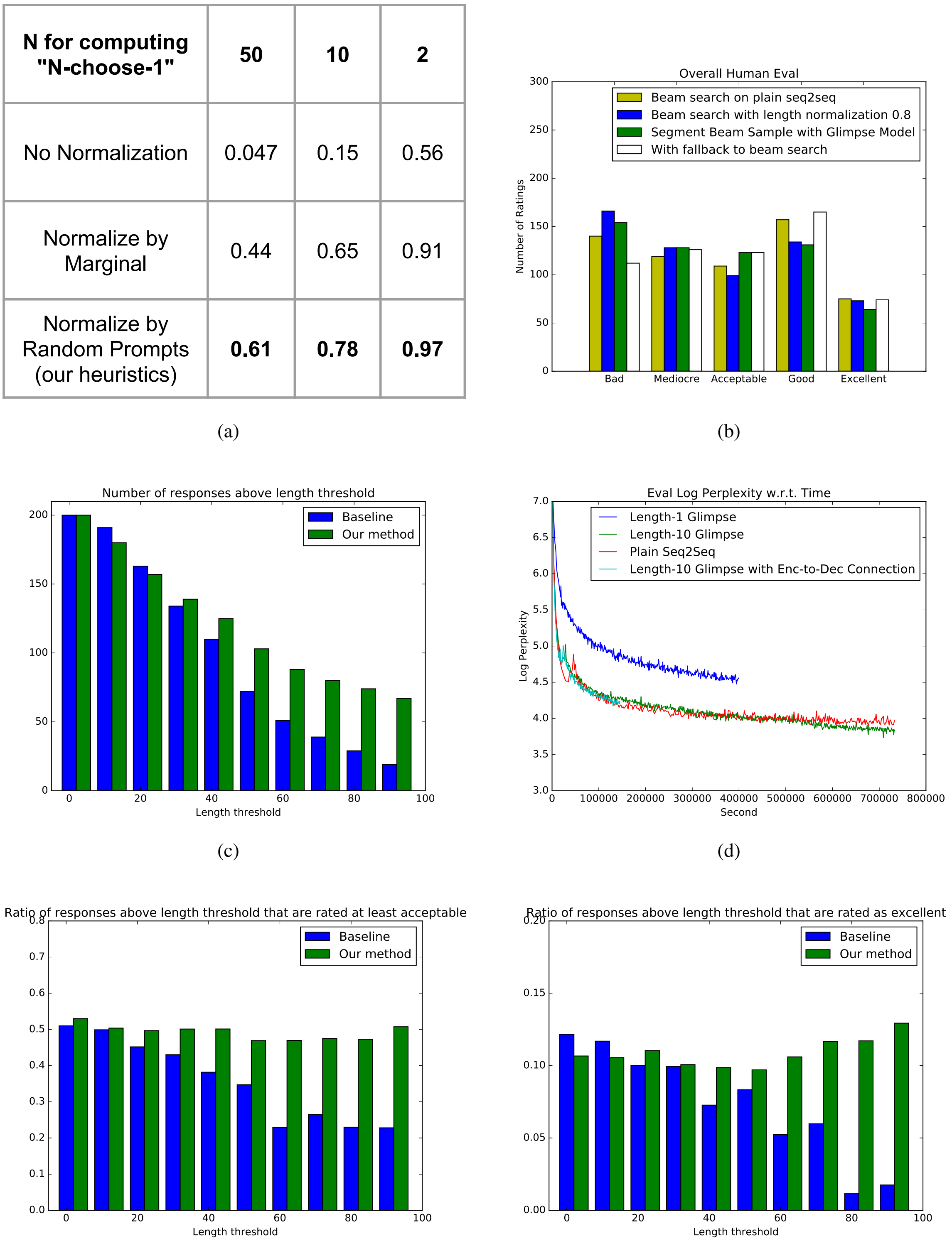
Jörg Tiedemann. 2009. News from OPUS - A Collection of Multilingual Parallel Corpora with Tools and Interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing (vol V)*, John Benjamins, Amsterdam/Philadelphia, pages 237–248.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* .

Tsung-Hsien Wen, Milica Gasic, Dongho Kim, Nikola Mrksic, Pei-hao Su, David Vandyke, and Steve J. Young. 2015. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. *CoRR* abs/1508.01755.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. Conditional generation and snapshot learning in neural dialogue systems. In *EMNLP*. ACL, Austin, Texas, pages 2153–2162. https://aclweb.org/anthology/D16-1233.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation .



N for computing "N-choose-1"	50	10	2
No Normalization	0.047	0.15	0.56
Normalize by Marginal	0.44	0.65	0.91
Normalize by Random Prompts (our heuristics)	**0.61**	**0.78**	**0.97**

(a) (b)

(c) (d)

(e) (f)

Figure 2: (a) N-choose-1 evaluation on the baseline model. (d) Training progress of different models on the full combined data set. *Length-1* and *Length-10* are the target-glimpse models we propose, and *Plain Seq2seq* is the baseline model we described. (b)(c)(e)(f): Human evaluation results on the conversation data. (b) The histogram of 5 ratings per method. (c) The length thresholds (horizontal axis) and the number of responses generated that are above the length threshold (vertical axis); (e) The proportion of responses above the length-threshold that are judged at least *Acceptable*; (f) The proportion of responses above the length-threshold that are judged as *Excellent*. The length thresholds are all measured in number of characters.

Bootstrapping incremental dialogue systems from minimal data: the generalisation power of dialogue grammars

Arash Eshghi
Interaction Lab
Heriot-Watt University
a.eshghi@hw.ac.uk

Igor Shalyminov
Interaction Lab
Heriot-Watt University
is33@hw.ac.uk

Oliver Lemon
Interaction Lab
Heriot-Watt University
o.lemon@hw.ac.uk

Abstract

We investigate an end-to-end method for automatically inducing task-based dialogue systems from small amounts of unannotated dialogue data. It combines an incremental semantic grammar - Dynamic Syntax and Type Theory with Records (DS-TTR) - with Reinforcement Learning (RL), where language generation and dialogue management are a joint decision problem. The systems thus produced are *incremental*: dialogues are processed word-by-word, shown previously to be essential in supporting natural, spontaneous dialogue. We hypothesised that the rich linguistic knowledge within the grammar should enable a combinatorially large number of dialogue variations to be processed, even when trained on very few dialogues. Our experiments show that our model can process 74% of the Facebook AI bAbI dataset even when trained on only 0.13% of the data (5 dialogues). It can *in addition* process 65% of bAbI+, a corpus[1] we created by systematically adding incremental dialogue phenomena such as restarts and self-corrections to bAbI. We compare our model with a state-of-the-art retrieval model, MEMN2N (Bordes et al., 2017). We find that, in terms of semantic accuracy, MEMN2N shows very poor robustness to the bAbI+ transformations even when trained on the full bAbI dataset.

1 Introduction

There are currently several key problems for the practical data-driven (rather than hand-crafted) development of task-oriented dialogue systems, among them: (1) large amounts of dialogue data are needed, i.e. thousands of examples in a domain; (2) this data is usually required to be annotated with task-specific semantic/pragmatic information for the domain (e.g. various dialogue act schemes); and (3) the resulting systems are generally turn-based, and so do not support natural spontaneous dialogue which is processed *incrementally*, word-by-word, with many characteristic phenomena that arise from this incrementality.

In overcoming issue (2), a recent advance made in research on (non-task) chat dialogues has been the development of so-called "end-to-end" systems, in which all components are trained from textual dialogue examples, e.g. (Sordoni et al., 2015; Vinyals and Le, 2015). However, as Bordes and Weston (2017) argued, these end-to-end methods may not transfer well to task-based settings (where the user is trying to achieve a domain goal, such as booking a flight or finding a restaurant, resulting in an API call). Bordes and Weston (2017) then presented an end-to-end method using Memory Networks (MEMN2NS), which achieves 100% performance on a test-set of 1000 dialogues, after being trained on 1000 dialogues. This method processes dialogues turn-by-turn, and so does not have the advantages of more natural incremental systems (Aist et al., 2007; Skantze and Hjalmarsson, 2010); nor does it really perform language generation, rather it's based on a retrieval model that selects from a set of candidate system responses seen in the data.

This paper investigates an approach to these challenges - dubbed BABBLE - using an incremental, semantic parser and generator for dialogue (Eshghi et al., 2011; Eshghi, 2015), based around the Dynamic Syntax grammar formalism (DS, Kempson et al. (2001); Cann et al. (2005)).

Our advance in this paper, for end-to-end systems, is therefore twofold: (a) the BABBLE method

[1] Dataset available at https://bit.ly/babi_plus

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2220–2230
Copenhagen, Denmark, September 7–11, 2017.

overcomes the requirement for large amounts of dialogue data (i.e. 1000s of dialogues in a domain); (b) resulting systems are word-by-word incremental, in both parsing, generation and dialogue management. We show that using only 5 example dialogues from the bAbI, Task 1 dataset (i.e. 0.13% of the training data used by (Bordes et al., 2017)) BABBLE can automatically induce dialogue systems which process 74% of the bAbI testset in an incremental manner. We then introduce an extended incremental version of the bAbI dataset, which we call bAbI+ (see section 4.1), which adds some characteristic incremental phenomena - such as mid-utterance self-corrections - to the bAbI dialogues (this new dataset is freely available). Using this, we demonstrate that the BABBLE system can in addition generalise to, and process 65% of the bAbI+ dataset, still when trained *only on 5 dialogues from bAbI*. We compare this method to (Bordes et al., 2017)'s MEMN2N, which, in terms of semantic accuracy (reflected in how well `api-calls` are predicted at the end of bAbI Task 1), shows very poor robustness to the bAbI+ transformations, even when it is trained on *the full bAbI dataset*.

This overall method is portable to other task-based domains. Furthermore, as we use a semantic parser, the semantic/contextual representations of the dialogue can be used directly for large-scale inference, required in more complex tasks (e.g. interactive QA and search).

1.1 Dimensions of Pragmatic Synonymy

There are two important dimensions along which dialogues can vary, but nevertheless, lead to identical contexts: interactional, and lexical. Interactional synonymy is analogous to syntactic synonymy - when two distinct sentences are parsed to identical logical forms - except that it occurs not only at the level of a single sentence, but at the dialogue or discourse level. Fig. 1 shows examples of interactional variants that lead to very similar final contexts, in this case, that the user wants to buy an LG phone. These dialogues can be said to be *pragmatically synonymous* for this domain. Arguably, a good computational model of dialogue processing, and interactional dynamics should be able to capture this synonymy.

Lexical synonymy relations, on the other hand, hold among utterances, or dialogues, when different words (or sequences of words) express meanings that are sufficiently similar in a particular domain. What is striking about lexical synonymy relations is that unlike syntactic/interactional ones, they can often break down when one moves to another domain: lexical synonymy relations are domain specific.

Eshghi & Lemon (2014) developed a method similar in spirit to Kwiatkowski et al. (2013) for capturing lexical synonymy relations by creating clusters of semantic representations based on observations that they give rise to similar or identical extra-linguistic actions observed within a domain (e.g. a data-base query, a flight booking, or any API call). Distributional methods could also be used for this purpose (see e.g. Lewis & Steedman (2013)). In general, this kind of clustering is achieved when the domain-general semantics resulting from semantic parsing is grounded in a particular domain.

We note that while interactional synonymy relations in dialogue should be accounted for by semantic grammars or formal models of dialogue structure (such as DS-TTR (Eshghi et al., 2012), or KoS (Ginzburg, 2012)), lexical synonymy relations have to be learned from data.

2 Why a grammar-based approach?

Recent end-to-end data-driven machine learning approaches treat dialogue as a sequence-to-sequence generation problem, and train their models from large datasets e.g. (Sordoni et al., 2015; Wen et al., 2016b,a; Vinyals and Le, 2015). The systems resulting from these types of approach are in principle able to handle variations/patterns that they have encountered (sufficiently often) in the training data, but not beyond.

This large-data constraint is problematic for developers but is also strange when we consider the structural knowledge that we have about language and dialogue that can be encoded in grammars and computational models of interaction. Indeed, it is often stated that for humans to learn how to perform adequately in a domain, one example is enough from which to learn (e.g. Li et. al (2006)).

Furthermore, as these systems do not parse to logical forms, they do not allow for explicit inference, which further limits their application.

We therefore develop a method combining learning from data with an incremental semantic grammar of dialogue that is able to generalise from small number of observations in a domain –

Interactional Variation	USR: I would like an LG laptop sorry uhm phone SYS: okay.	USR: I would like a phone by LG. SYS: sorry a what? USR: a phone by LG. SYS: okay.	SYS: what would you like? USR: an LG phone SYS: okay.
	SYS: what would you like? USR: a phone SYS: by which brand? USR: LG SYS: okay	SYS: you'd like a ...? USR: a phone SYS: by what brand? USR: LG. SYS: okay	SYS: so would you like a computer? USR: no, a phone. SYS: okay. by which brand? USR: LG. SYS: okay.

Figure 1: Some Interactional Variations in a Shopping Domain

in fact even from just a few examples of successful dialogues – to a large range of interactional and syntactic variations, including everyday natural incremental phenomena.

3 Inducing Dialogue Systems

Our overall method involves incrementally parsing dialogues, and encoding the resulting semantics as state vectors in a Markov Decision Process (MDP), which is then used for Reinforcement Learning (RL) of word-level actions for system output (i.e. a combined incremental DM and NLG module for the resulting dialogue system).

3.1 Dynamic Syntax and Type Theory with Records (DS-TTR)

Dynamic Syntax (DS) is an action-based, word-by-word incremental and semantic grammar formalism (Kempson et al., 2001; Cann et al., 2005), especially suited to the highly fragmentary and context-dependent nature of dialogue. In DS, words are conditional actions - semantic updates; and dialogue is modelled as the interactive and incremental construction of contextual and semantic representations (Eshghi et al., 2015) - see Fig. 2. The contextual representations afforded by DS are of the fine-grained semantic content that is jointly negotiated/agreed upon by the interlocutors, as a result of processing questions and answers, clarification interaction, acceptances, self-/other-corrections, restarts, and other characteristic incremental phenomena in dialogue - see 3 for a sketch of how self-corrections and restarts are processed via a backtrack and search mechanism over the parse search graph (see Hough (2011); Hough and Purver (2014); Eshghi et al. (2015) for details of the model, and how this parse search graph is effectively the context of the conversation). Generation/linearisation in DS is defined using trial-and-error parsing (see Section 3.2, with the provision of *a generation goal*, viz. the semantics of the utterance to be generated. Generation thus proceeds, just as with parsing, on a word-by-word basis (see Purver et al. (2014); Hough (2015) for details). The upshot of this is that using DS, we can not only track the semantic content of some current turn as it is being constructed (parsed or generated) word-by-word, but also the context of the conversation as whole, with the latter also encoding the grounded/agreed content of the conversation (see e.g. Fig. 4, and see Eshghi et al. (2015); Purver et al. (2010) for details). Crucially for our model below, the inherent incrementality of DS-TTR together with the word-level, as well as cross-turn, parsing constraints it provides, enables the word-by-word exploration of the space of grammatical dialogues, and the semantic and contextual representations that result from them.

Type Theory with Records (TTR) is an extension of standard type theory shown to be useful in semantics and dialogue modelling (Cooper, 2005; Ginzburg, 2012). To accommodate dialogue processing, and allow for richer representations of the dialogue context recent work has integrated DS and the TTR framework to replace the logical formalism in which meanings are expressed (Purver et al., 2010, 2011; Eshghi et al., 2012). In TTR, logical forms are specified as *record types* (RTs), sequences of *fields* of the form $[\, l : T \,]$ containing a label l and a type T. RTs can be witnessed (i.e. judged as true) by *records* of that type, where a record is a sequence of label-value pairs $[\, l = v \,]$, and $[\, l = v \,]$ is of type $[\, l : T \,]$ just in case v is of type T (see Fig. 2 for example record types).

Importantly for us here, the standard *subtype* relation $\sqsubseteq$ can be defined for record types: $R_1 \sqsubseteq R_2$ if for all fields $[\, l : T_2 \,]$ in R_2, R_1 contains $[\, l : T_1 \,]$ where $T_1 \sqsubseteq T_2$. A record type can thus be indefinitely extended, and is therefore always underspecified by definition. This allows for incrementally growing meanings to be expressed in a natural way as more words are parsed or generated in

$$\begin{bmatrix} event & : & e_s \\ p1_{=today(event)} & : & t \end{bmatrix} \mapsto \begin{bmatrix} event_{=arrive} & : & e_s \\ p1_{=today(event)} & : & t \\ p2_{=pres(event)} & : & t \\ x_{=robin} & : & e \\ p3_{=subj(event,x)} & : & t \end{bmatrix} \mapsto \begin{bmatrix} event_{=arrive} & : & e_s \\ p1_{=today(event)} & : & t \\ p2_{=pres(event)} & : & t \\ x_{=robin} & : & e \\ p3_{=subj(event,x)} & : & t \\ x1 & : & e \\ p3_{=from(event,x1)} & : & t \end{bmatrix} \mapsto \begin{bmatrix} event_{=arrive} & : & e_s \\ p1_{=today(event)} & : & t \\ p2_{=pres(event)} & : & t \\ x_{=robin} & : & e \\ p_{=subj(event,x)} & : & t \\ x1_{=Sweden} & : & e \\ p3_{=from(event,x1)} & : & t \end{bmatrix}$$

"A: Today" ↦ *"..Robin arrives"* ↦ *"B: from?"* ↦ *"A: Sweden"*

Figure 2: Incremental parsing using DS-TTR

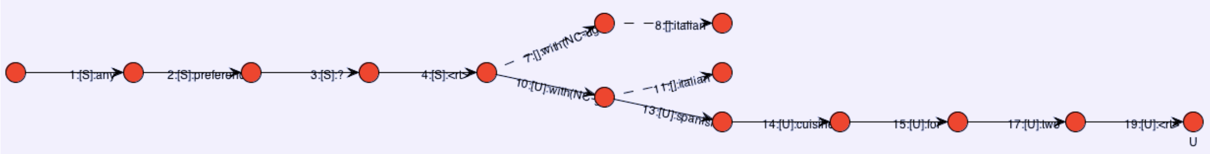


Figure 3: DS-TTR: Incremental Parsing of self-corrections and restarts

turn. In addition, as will become clear below, this subtype checking operation is the key mechanism used in our system below for feature checking.

3.2 Overall Method: BABBLE

In this section we describe our method for combining incremental dialogue parsing with Reinforcement Learning for Dialogue Management (DM) and Natural Language Generation (NLG) where these are treated as a joint decision/optimisation problem.

We start with two resources: a) a DS-TTR parser *DS* (either learned from data (Eshghi et al., 2013a), or constructed by hand), for incremental language processing, but also, more generally, for tracking the context of the dialogue using Eshghi et al.'s model of feedback (Eshghi et al., 2015; Eshghi, 2015; Eshghi et al., 2011); b) a set D of transcribed successful dialogues in the target domain.

We perform the following steps overall to induce a fully incremental dialogue system from D:

1. Automatically induce the MDP state space, S, and the dialogue goal, G_D, from D;
2. Automatically define the state encoding function $F : C \rightarrow S$; where $s \in S$ is a (binary) state vector, designed to extract from the current context of the dialogue, the semantic features observed in the example dialogues D; and $c \in C$ is a DS context, viz. a pair of TTR Record Types: $\langle c_p, c_g \rangle$, where c_p is the content of the current, *PENDING* clause as it is being constructed, but not necessarily fully grounded yet; and c_g is the content already jointly built and *GROUNDED* by the interlocutors (loosely following the DGB model of (Ginzburg, 2012)).
3. Define the MDP action set as the *DS* lexicon L (i.e. actions are words);
4. Define the reward function R as reaching G_D, while minimising dialogue length.

We then solve the generated MDP using Reinforcement Learning, with a standard Q-learning method, implemented using BURLAP (McGlashan, 2016): train a policy $\pi : S \rightarrow L$, where L is the DS Lexicon, and S the state space induced using F. The system is trained in interaction with a (semantic) simulated user, also automatically built from the dialogue data and described in the next section.

The state encoding function, F As shown in figure 4 the MDP state is a binary vector of size $2 \times |\Phi|$, i.e. twice the number of the RT features. The first half of the state vector contains the grounded features (i.e. agreed by the participants) ϕ_i, while the second half contains the current semantics being incrementally built in the current dialogue utterance. Formally: $s = \langle F_1(c_p), \ldots, F_m(c_p), F_1(c_g), \ldots, F_m(c_g) \rangle$; where $F_i(c) = 1$ if $c \sqsubseteq \phi_i$, and 0 otherwise. (Recall that $\sqsubseteq$ is the RT subtype relation).

3.2.1 Semantic User Simulation

The simulator is in charge of two key tasks during training: (1) generating user turns in the right dialogue contexts; and (2) word-by-word monitoring of the utterance so far generated by the sys-

Grounded Semantics

$$\left[\begin{array}{lcl} x2 & : & e \\ e2_{=like} & : & es \\ x1_{=USR} & : & e \\ p2_{=pres(e2)} & : & t \\ p5_{=subj(e2,x1)} & : & t \\ p4_{=obj(e2,x2)} & : & t \\ p11_{=phone(x2)} & : & t \end{array}\right]$$

Current Turn Semantics

$$\left[\begin{array}{lcl} x2 & : & e \\ e2_{=like} & : & es \\ x1_{=USR} & : & e \\ p2_{=pres(e2)} & : & t \\ p5_{=subj(e2,x1)} & : & t \\ p4_{=obj(e2,x2)} & : & t \\ p11_{=phone(x2)} & : & t \\ x3 & : & e \\ p10_{=by(x2,x3)} & : & t \\ p9_{=brand(x3)} & : & t \\ p10_{=question(x3)} & : & t \end{array}\right]$$

Dialogue so far

SYS: What would you like?
USR: a phone
SYS: by which brand?

RT Feature:

$$\left[\begin{array}{lcl} x10 & : & e \\ p15_{=brand(x10)} & : & t \end{array}\right]\left[\begin{array}{lcl} e3_{=like} & : & es \\ p2_{=pres(e3)} & : & t \end{array}\right]\left[\begin{array}{lcl} x10 & : & e \\ x8 & : & e \\ p14_{=by(x8,x10)} & : & t \end{array}\right]\left[\begin{array}{lcl} e3_{=like} & : & es \\ x5_{=usr} & : & e \\ p7_{=subj(e3,x5)} & : & t \end{array}\right]\left[\begin{array}{lcl} x8 & : & e \\ e3_{=like} & : & es \\ p6_{=obj(e3,x8)} & : & t \end{array}\right]$$

$$\text{State: } \left\langle \begin{array}{lccccc} & F_1\downarrow & F_2\downarrow & F_3\downarrow & F_4\downarrow & F_5\downarrow \\ \text{Current Turn:} & 1, & 1, & 1, & 1, & 1, \\ \text{Grounded:} & 0, & 1, & 0, & 1, & 1 \end{array} \right\rangle$$

Figure 4: Semantics to MDP state encoding with RT features

tem during exploration (i.e. babbling grammatical word sequences) by the system. To exploit (and evaluate) the full generalisation properties of the *DS* dialogue model, both (1) and (2) use the full machinery of the *DS* parser, as well the state encoding function F, described above. They are thus performed based on the *semantic context* of the dialogue so far, as tracked by *DS* (rather than, e.g. being based on string or template matching). Since this includes not just the semantic features of the current turn, but also of *the history of the conversation*, our simulator respects the turn orderings encountered in the data, i.e. it is sensitive to the order in which information is gathered from the user.

The rules required for (1) & (2) are extracted *automatically* from the raw dialogue data, D, using *DS* and F. The dialogues in D are parsed and encoded using F incrementally. For (1), all the states that trigger the user into action, $s_i = F(c)$ – where c is a DS context – immediately prior to any user turn are recorded, and mapped to what the user ends up saying in those contexts - for more than one training dialogue there may be more than one candidate (in the same context/state). The rules thus extracted will be of the form:
$s_{trig} \rightarrow \{u_1, \ldots, u_n\}$, where u_i are user turns.

Now note that the s_i's prior to the user turns also immediately follow system turns. And thus to perform (2), i.e. to monitor the system's behaviour during training, we only need to check further that the current state resulting from processing a word generated by the system, subsumes - is extendible to - one of the s_i. We perform this through a simple bitmask operation (recall that the states are binary). The simulation can thus semantically identify erroneous/out-of-domain actions (words) by the system. It would then terminate the learning episode and penalise the system immediately, aiding speed of training significantly.

4 Evaluation

We have so far induced two prototype dialogue systems, one in an 'electronics shopping' domain (see Kalatzis et al. (2016) and Fig. 1) and another in a 'restaurant-search' domain, showing that fully incremental dialogue systems can be automatically induced from small amounts of unannotated dialogue transcripts (Kalatzis et al., 2016; Eshghi et al., 2017) - in this case both systems were bootstrapped from a *single* successful example dialogue. We are in the process of evaluating these systems with real users.

In this paper, however, our focus is not on building dialogue systems *per se*, but on: **(1)** studying and quantifying the interactional and structural generalisation power of the DS-TTR grammar formalism (see Section 2), and that of symbolic, grammar-based approaches to language processing more generally. We focus here on specific dialogue phenomena, such as mid-sentence self-corrections, hesitations, and restarts (see below); **(2)** doing the same for Bordes and Weston's (2017) state-of-the-art, bottom up response retrieval model, without use of linguistic knowledge of any form; and **(3)** comparing (1) and (2).

In order to test and quantify the interactional and structural generalisation power/robustness of the two models, BABBLE and MEMN2N, we need contrasting dialogue data-sets that control for interactional vs. lexical variations in the input dialogues. Furthermore, to make our results comparable to the existing approach of Bordes and Weston (2017), we need to use the same dataset that they have used. We therefore use Facebook AI Research's bAbI dialogue tasks dataset (Bordes et al., 2017). These are goal-oriented dialogues in the domain of restaurant search. Here we tackle Task 1, where in each dialogue the system asks the user about their preferences for the properties of a restaurant, and each dialogue results in an API call which contains values of each slot obtained. Other than the explicit API call notation, there are no annotations in the data whatsoever.

4.1 The bAbI+ dataset

While containing some lexical variation, the original bAbI dialogues significantly lack interactional variation vital for natural real-life dialogue. In order to obtain such variation while holding lexical variation constant, we created the bAbI+ dataset by systematically transforming the bAbI dialogues.

bAbI+ is an extension of the original bAbI Task 1 dialogues with everyday incremental dialogue phenomena (hesitations, restarts, and corrections – see below). While the original bAbI tasks 2—7 increase the user's goal complexity, modifications introduced in bAbI+ can be thought of as orthogonal to this: we instead increase the complexity of surface forms of dialogue utterances, while keeping every other aspect of the task fixed.

The variations introduced in bAbI+ are: **1. Hesitations**, e.g. as in "we will be `uhm` eight"; **2. Restarts**, e.g. "can you make a restaurant `uhm yeah can you make a restaurant` reservation for four people with french cuisine in a moderate price range"; and **3. Corrections** affecting task-specific information - both short-distance ones correcting one token, e.g. "with french `oh no spanish` food", and long-distance NP/PP-level corrections, e.g. "with french food `uhm sorry with spanish food`".

The phenomena above are mixed in probabilistically from the fixed sets of templates to the original data[2]. The modifications affect a total of **11336** utterances in the **3998** dialogues. Around **21%** of user turns contain corrections, **40%** hesitations, and **5%** restarts (they are not mutually exclusive, so that an utterance can contain up to 3 modifications). Our modifications, with respect to corrections in particular, are more conservative than those observed in real-world data: Hough (2015) reports that self-corrections appear in **20%** of all turns of natural conversations from the British National Corpus, and in **40%** of turns in the Map Task, a corpus of human-human goal-oriented dialogues. Here's part of an example dialogue in the bAbI+ corpus:

sys:	hello what can I help you with today?
usr:	I'd like to book a uhm yeah I'd like to book a table in a expensive price range
sys:	I'm on it. Any preference on a type of cuisine?
usr:	with indian food no sorry with spanish food please

4.2 Memory Network setup

In all the experiments we describe below, we follow Bordes and Weston's setup by using a MEMN2N (we took an open source Tensorflow implementation for bAbI QA tasks and modified it[3] according to their setup – see details below). In order to adapt the data for the MEMN2N, we transform the dialogues into <*story, question, answer*> triplets. The number of triplets for a single dialogue is equal to the number of the system's turns, and in each triplet, the answer is the current system's turn, the question is the user's turn preceding it, and the story is a list of all the previous turns among both sides.

The MEMN2N hyperparameters are set as follows: **1** hop, and **128** as the size of embeddings; we train it for **100** epochs with a learning rate of **0.01** and a batch size of **8** – in this we follow the best bAbI Task 1 setup reported by (Bordes et al., 2017).

4.3 Testing the DS-TTR parser

Dynamic Syntax (DS) lexicons are learnable from data (Eshghi et al., 2013a,b). But since the lexicon was induced from a corpus of child-directed utterances in this prior work, there were some constructions as well as individual words that it did not include[4]. One of the authors therefore extended this induced grammar manually to cover the bAbI dataset, which, despite not being very diverse,

[2] See `https://github.com/ishalyminov/babi_tools`

[3] See `https://github.com/ishalyminov/memn2n`

[4] We are currently looking into applying Eshghi et al.'s (2013a) model to induce DS grammars from larger semantic corpora such as the Groningen Meaning Bank, leading to much more wide-coverage lexicons

contains a wide range of complex grammatical constructions, such as long sequences of prepositional phrases, adjuncts, short answers to yes/no and wh-questions, appositions of NPs, causative verbs etc.

We parsed all dialogues in the bAbI train and test sets, as well as on the bAbI+ corpus word-by-word, including both user and system utterances, in context. The grammar parses 100% of the dialogues, i.e. it does not fail on any word in any of the dialogues. We assess the semantic accuracy of the parser on bAbI & bAbI+ using the dialogue-final `api-calls` in section 4.5 below.

4.4 Experiment 1: Generalisation from small data

We have now set out all we need to perform the first experiment. Our aim here is to assess the generalisation power that results from the grammar and our state encoding method (section 3.1) - we dub our overall model BABBLE - and compare this to the state of the art results of Bordes et al. (2017). The method in Bordes et al. (2017) is not generative, rather it is based on retrieval of system responses, based on the history of the dialogue up to that point. Therefore, for direct comparison, and for simplicity of exposition, we do the same here: we apply the method described for creating a user simulation (section 3.2.1), this time *for the system side*, resulting in a 'system simulation'. We then use this to predict a system response, by parsing and encoding the containing test dialogue up to the point immediately prior to the system turn. This results in a triggering state, s_{trig}, which is then used as the key to look up the system's response from the rules constructed as per section 3.2.1. The returned response is then parsed word-by-word as normal, and this same process continues for the rest of the dialogue. This method uses the full machinery of DS-TTR & our state-encoding method - the BABBLE model - and will thus reflect the generalisation properties that we are interested in.

Cross-Validation Since we are here interested in data efficiency and generalisation we use all the bAbI and bAbI+ data - the train, dev, and test sets - as follows: we train Bordes & Weston's MEMN2N and BABBLE from 1-5 examples selected at random from *the longest dialogues* in bAbI – note **bAbI+ data is never used for training in these experiments**. This process is repeated across 10 folds. The models are then tested on sets of 1000 examples selected at random, in each fold. Both the training and test sets constructed in this way are kept constant in each fold across the BABBLE & MEMN2N models. The test sets are selected either exclusively from bAbI or exclusively from bAbI+.

4.4.1 Results: Predicting system turns

Table 1 shows per utterance accuracies for the BABBLE & MEMN2N models. Per utterance accuracy is the percentage of all system turns in the test dialogues that were correctly predicted. The table shows that BABBLE can generalise to a remarkable 74% of bAbI and 65% of bAbI+ with only 5 input dialogues from bAbI. It also shows that MEMN2NS can also generalise remarkably well. Although as discussed below, this result is misleading on its own as the MEMN2NS are very poor at generating the final `api-calls` correctly on both the bAbI & bAbI+ data, and are thus making too many semantic mistakes.

4.5 Experiment 2: Semantic Accuracy

The results from Experiment 1 on their own can be misleading, as correct prediction of system responses does not in general tell us enough about how well the models are interpreting the dialogues, or whether they are doing this with a sufficient level of granularity. To assess this, in this second experiment, we measure the semantic accuracy of each model by *looking exclusively* at how accurately they predict the final `api-calls` in the bAbI & bAbI+ datasets. For the MEMN2N model, we follow the same overall procedure as in the previous experiment: train on bAbI data, and test on bAbI+.

4.5.1 Results: Prediction of `api-calls`

BABBLE Mere successful parsing of all the dialogues in the bAbI and bAbI+ datasets as shown above doesn't mean that the semantic representations compiled for the dialogues were in fact correct. To measure the semantic accuracy of the DS-TTR parser we programmatically checked that the correct slot values – those in the `api-call` annotations – were in fact present in the semantic representations produced by the parser for each dialogue (see Fig. 2 for example semantic representations). We further checked that there is no other incorrect slot value present in these representations. The results showed that the parser has 100% se-

# of training dialogues:	1	2	3	4	5
BABBLE on bAbI	67.12	73.36	72.63	73.32	74.08
MEMN2N on bAbI	2.77	59.15	70.94	71.68	72.6
BABBLE on bAbI+	59.42	65.27	63.45	64.34	65.2
MEMN2N on bAbI+	0.22	56.75	68.65	71.84	73.2

Table 1: Mean per utterance accuracies (%) for MEMN2N & BABBLE models across the bAbI & bAbI+ datasets (10 folds)

mantic accuracy on both bAbI and bAbI+[5]. This result is not surprising, given that DS-TTR is a general model of incremental language processing, including phenomena such as self-corrections and restarts (see Hough (2015) for details of the model).

MEMN2N Given just 1 to 5 training instances from bAbI as in the previous experiment, the mean `api-call` prediction accuracy of the MEMN2N model is **nearly 0 on both bAbI and bAbI+**. This is not at all unexpected, since prediction of the `api-calls` is a *generative* process, unlike the prediction of system turns which can be done on a retrieval/look-up basis alone. For this, the model needs to observe the different word sequences that might determine each parameter (slot) value, and observe them with sufficient frequency and variation. This is unlike a semantic parser like DS-TTR, that produces semantic representations for the dialogues as a result of the structural, linguistic knowledge that it embodies.

Nevertheless, we were also interested in the general semantic robustness of the MEMN2N model, to the transformations in bAbI+, i.e. how well does the MEMN2N model interpret bAbI+ dialogues, when trained *on the full bAbI dataset*? Does it then learn to generalise to (process) the bAbI+ dialogues with sufficient semantic accuracy?

Table 2 shows that we can fully replicate the results reported in Bordes et al. (2017): the MEMN2N model can predict the `api-calls` with 100% accuracy, when trained on the bAbI train-set and tested on the bAbI test-set. But when this same model is tested on bAbI+, the accuracy drops to a very poor 28%, making any dialogue system built using this model unusable in the face of natural, spontaneous dialogue data. This is further discussed below.

testing configuration	accuracy
MEMN2N on bAbI	100
MEMN2N on bAbI+	28

Table 2: `api-call` prediction accuracies (%) for the MEMN2N model trained on the bAbI trainset

5 Discussion

5.1 BABBLE

The method described above has the following advantages over previous approaches to dialogue system development:

– incremental (and thus more natural) language understanding, dialogue management, and generation;

– “end-to-end" method for task-based systems: no Dialogue Act annotations are required (i.e. reduced development time and effort);

– a complete dialogue system for a new task can be automatically induced, using only ‘raw’ data – i.e. successful dialogue transcripts;

– the MDP state and action spaces are automatically induced, rather than having to be designed by hand (as in prior work);

– wide-coverage, task-based dialogue systems can be built from much smaller amounts of data as shown in section 4 .

This final point bears further examination. As an empirically adequate model of incremental language processing in dialogue, the DS-TTR grammar is required to capture interactional variants such as question-answer pairs, over- and under-answering, self- and other-corrections, clarification, split-utterances, and ellipsis more generally. As we showed in section 4, even if most of these structures are not present in the training exam-

[5] A helpful reviewer points out that the DS-TTR setup is a carefully tuned rule-based system, thus perhaps rendering these results trivial. But we note that the results here are not due to ad-hoc constructions of rules/lexicons, but due to the generality of the grammar model, and its attendant incremental, left-to-right properties; and that the same parser can be used in other domains. Furthermore, the ability to process self-corrections, restarts, etc. “comes for free”, without the need to add or posit new machinery

ple(s), the resulting trained system is able to handle them, thus resulting in a very significant generalisation around the original data.

We also note that since we were in this instance interested in a direct comparison with MEMN2NS over the bAbI & bAbI+ datasets, we didn't exploit the power of Reinforcement Learning and exploration as we described above - as we have done before with other systems (Kalatzis et al., 2016). Therefore the generalisation results we report above for BABBLE follow entirely from the knowledge present within the grammar as a computational model of dialogue processing and contextual update, rather than this having been learned from data. Applying the full RL method described above would have meant that the system would actually discover many interactional and syntactic variations that are not present in bAbI, nor in bAbI+.

5.2 MEMN2N

Even when trained on very few training instances, the MEMN2N model was able to predict system responses remarkably well. But results from Experiment 2 above showed that this was misleading: the MEMN2NS were making a drastic number of semantic mistakes when interpreting the dialogues, both in the bAbI and bAbI+ datasets. Even when trained on the full bAbI data-set, the model performed badly on bAbI+ in terms of semantic accuracy. We diagnose these results as follows:

Problem complexity: The first thing to notice is that in bAbI dialogue Task 1, the responses are highly predictable and stay constant regardless of the actual task details (slot values) up to the point of the final `api-calls`; and further, that the prediction of `api-calls` is a *generative* process, unlike the prediction of the system turns, which is retrieval-based. This, in our view, explains the very large difference in MEMN2N performance across the two prediction tasks.

Model robustness to the bAbI+ transformations:. The variations introduced in bAbI+ are repetitions of both content and non-content words, as well additional incorrect slot values. The model was working in the same setup as BABBLE, therefore none of those variations could be treated as unknown tokens for either system. Although in the case of MEMN2N, some of the mixed-in words never appeared in the training data, and bAbI+ utterances were augmented significantly with those words – so it was interesting to see how such untrained embeddings would affect the latent memory representations inside MEMN2N. The resulting performance suggests that there was no significant impact on MEMN2N from these variations as far as the predicting system responses was concerned. But the incorrect slot values introduced in self-corrections affect the system's task completion performance significantly, only appearing at the point of `api-call` predictions.

We note also that none of our experiments in this paper involved training MEMN2N on bAbI+ data. There is a very interesting question here: is the MEMN2N model in principle able to learn to process the bAbI+ structures if it is in fact trained on it? And how much bAbI+ data would it require to do so? These issues are address in detail in Shalyminov et al. (2017).

6 Conclusions

Our main advances are in a) training end-to-end dialogue systems from small amounts of data, b) incremental processing for wider coverage of more natural everyday dialogues (e.g. containing self-repairs).

We compared our grammar-based approach to dialogue processing (DS-TTR) with a state-of-the-art, end-to-end response retrieval model (MEMN2NS) (Bordes et al., 2017), when training on small amounts of dialogue data.

Our experiments show that our model can process 74% of the Facebook AI bAbI dataset even when trained on only 0.13% of the data (5 dialogues). It can *in addition* process 65% of bAbI+, a corpus we created by systematically adding incremental dialogue phenomena such as restarts and self-corrections to bAbI. We find on the other hand that the MEMN2N model is not robust to the structures we introduced in bAbI+, even when trained on the full bAbI dataset.

Acknowledgements

This research is supported by the EPSRC, under grant number EP/M01553X/1 (BABBLE project[6]).

[6] `https://sites.google.com/site/hwinteractionlab/babble`

References

Gregory Aist, James Allen, Ellen Campana, Carlos Gomez Gallo, Scott Stoness, Mary Swift, and Michael K Tanenhaus. 2007. Incremental dialogue system faster than and preferred to its nonincremental counterpart. In *Annual Conference of the Congitive Science Society*.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. *ICLR* https://openreview.net/pdf?id=S1Bb3D5gg.

Ronnie Cann, Ruth Kempson, and Lutz Marten. 2005. *The Dynamics of Language*. Elsevier, Oxford.

Robin Cooper. 2005. Records and record types in semantic theory. *Journal of Logic and Computation* 15(2):99–112.

A. Eshghi, C. Howes, E. Gregoromichelaki, J. Hough, and M. Purver. 2015. Feedback in conversation as incremental semantic update. In *Proceedings of the 11th International Conference on Computational Semantics (IWCS 2015)*. Association for Computational Linguisitics, London, UK.

A. Eshghi, M. Purver, and Julian Hough. 2011. Dylan: Parser for dynamic syntax. Technical report, Queen Mary University of London.

Arash Eshghi. 2015. DS-TTR: An incremental, semantic, contextual parser for dialogue. In *Proceedings of Semdial 2015 (goDial), the 19th workshop on the semantics and pragmatics of dialogue*.

Arash Eshghi, Julian Hough, and Matthew Purver. 2013a. Incremental grammar induction from child-directed dialogue utterances. In *Proceedings of the 4th Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 94–103.

Arash Eshghi, Julian Hough, Matthew Purver, Ruth Kempson, and Eleni Gregoromichelaki. 2012. Conversational interactions: Capturing dialogue dynamics. In S. Larsson and L. Borin, editors, *From Quantification to Conversation: Festschrift for Robin Cooper on the occasion of his 65th birthday*, College Publications, London, volume 19 of *Tributes*, pages 325–349. http://www.eecs.qmul.ac.uk/ mpurver/papers/eshghi-et-al12cooper.pdf.

Arash Eshghi and Oliver Lemon. 2014. How domain-general can we be? Learning incremental dialogue systems without dialogue acts. In *Proceedings of Semdial 2014 (DialWatt)*.

Arash Eshghi, Matthew Purver, Julian Hough, and Yo Sato. 2013b. Probabilistic grammar induction in an incremental semantic framework. In *CSLP, Lecture Notes in Computer Science*, Springer.

Arash Eshghi, Igor Shalyminov, and Oliver Lemon. 2017. Interactional Dynamics and the Emergence of Language Games. In *Proceedings of the ESSLLI 2017 workshop on Formal approaches to the Dynamics of Linguistic Interaction*. Barcelona.

Jonathan Ginzburg. 2012. *The Interactive Stance: Meaning for Conversation*. Oxford University Press.

Julian Hough. 2011. Incremental semantics driven natural language generation with self-repairing capability. In *Recent Advances in Natural Language Processing (RANLP)*. Hissar, Bulgaria, pages 79–84.

Julian Hough. 2015. *Modelling Incremental Self-Repair Processing in Dialogue*. Ph.D. thesis, Queen Mary University of London.

Julian Hough and Matthew Purver. 2014. Probabilistic type theory for incremental dialogue processing. In *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*. Association for Computational Linguistics, Gothenburg, Sweden, pages 80–88. http://www.aclweb.org/anthology/W14-1410.

Dimitrios Kalatzis, Arash Eshghi, and Oliver Lemon. 2016. Bootstrapping incremental dialogue systems: using linguistic knowledge to learn from minimal data. In *Proceedings of the NIPS 2016 workshop on Learning Methods for Dialogue*. Barcelona.

Ruth Kempson, Wilfried Meyer-Viol, and Dov Gabbay. 2001. *Dynamic Syntax: The Flow of Language Understanding*. Blackwell.

Tom Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics* 1:179–192. http://aclweb.org/anthology//Q/Q13/Q13-1015.pdf.

Fei-Fei Li, Robert Fergus, and Pietro Perona. 2006. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.* 28(4):594–611. https://doi.org/10.1109/TPAMI.2006.79.

James McGlashan. 2016. BURLAP: Brown-UMBC Reinforcement Learning and Planning. *http://burlap.cs.brown.edu/* .

Matthew Purver, Arash Eshghi, and Julian Hough. 2011. Incremental semantic construction in a dialogue system. In J. Bos and S. Pulman, editors, *Proceedings of the 9th International Conference on Computational Semantics*. Oxford, UK, pages 365–369.

Matthew Purver, Eleni Gregoromichelaki, Wilfried Meyer-Viol, and Ronnie Cann. 2010. Splitting the 'I's and crossing the 'You's: Context, speech acts and grammar. In P. Łupkowski and M. Purver, editors, *Aspects of Semantics and Pragmatics of Dialogue. SemDial 2010, 14th Workshop on the Semantics and Pragmatics of Dialogue*. Polish Society for Cognitive Science, Poznań, pages 43–50.

Matthew Purver, Julian Hough, and Eleni Gregoromichelaki. 2014. Dialogue and compound contributions. In S. Bangalore and A. Stent, editors, *Natural Language Generation in Interactive Systems*, Cambridge University Press, pages 63–92. http://www.cambridge.org/us/academic/subjects/engineering/communications-and-signal-processing/natural-language-generation-interactive-systems.

Igor Shalyminov, Arash Eshghi, and Oliver Lemon. 2017. Challenging Neural Dialogue Models with Natural Data: Memory Networks Fail on Incremental Phenomena. In *Proceedings of the 21st Workshop on the Semantics and Pragmatics of Dialogue (SemDial 2017 - SaarDial)*.

Gabriel Skantze and Anna Hjalmarsson. 2010. Towards incremental speech generation in dialogue systems. In *Proceedings of the SIGDIAL 2010 Conference*. Association for Computational Linguistics, Tokyo, Japan, pages 1–8.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv* (1506.06714).

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv* (1506.05869v3).

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016a. Multi-domain neural network language generation for spoken dialogue systems. In *Proceedings of the 2016 Conference on North American Chapter of the Association for Computational Linguistics (NAACL)*. Association for Computational Linguistics.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016b. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint: 1604.04562* .

Composite Task-Completion Dialogue Policy Learning via Hierarchical Deep Reinforcement Learning

Baolin Peng[⋆] Xiujun Li[†] Lihong Li[†] Jianfeng Gao[†]
Asli Celikyilmaz[†] Sungjin Lee[†] Kam-Fai Wong[⋆]
[†]Microsoft Research, Redmond, WA, USA
[⋆]The Chinese University of Hong Kong, Hong Kong
{blpeng, kfwong}@se.cuhk.edu.hk
{xiul,lihongli,jfgao,aslicel,sule}@microsoft.com

Abstract

Building a dialogue agent to fulfill complex tasks, such as travel planning, is challenging because the agent has to learn to *collectively* complete multiple subtasks. For example, the agent needs to reserve a hotel and book a flight so that there leaves enough time for commute between arrival and hotel check-in. This paper addresses this challenge by formulating the task in the mathematical framework of *options* over Markov Decision Processes (MDPs), and proposing a hierarchical deep reinforcement learning approach to learning a dialogue manager that operates at different temporal scales. The dialogue manager consists of: (1) a top-level dialogue policy that selects among subtasks or options, (2) a low-level dialogue policy that selects primitive actions to complete the subtask given by the top-level policy, and (3) a global state tracker that helps ensure all cross-subtask constraints be satisfied. Experiments on a travel planning task with simulated and real users show that our approach leads to significant improvements over three baselines, two based on handcrafted rules and the other based on flat deep reinforcement learning.

1 Introduction

There is a growing demand for intelligent personal assistants, mainly in the form of dialogue agents, that can help users accomplish tasks ranging from meeting scheduling to vacation planning. However, most of the popular agents in today's market, such as Amazon Echo, Apple Siri, Google Home and Microsoft Cortana, can only handle very simple tasks, such as reporting weather and requesting songs. Building a dialogue agent to fulfill complex tasks remains one of the most fundamental challenges for the NLP community and AI in general.

In this paper, we consider an important type of complex tasks, termed *composite task*, which consists of a set of subtasks that need to be fulfilled collectively. For example, in order to make a travel plan, we need to book air tickets, reserve a hotel, rent a car, etc. in a collective way so as to satisfy a set of cross-subtask constraints, which we call *slot constraints*. Examples of slot constraints for travel planning are: hotel check-in time should be later than the flight's arrival time, hotel check-out time may be earlier than the return flight depart time, the number of flight tickets equals to that of hotel check-in people, and so on.

It is common to learn a task-completion dialogue agent using reinforcement learning (RL); see Su et al. (2016); Cuayáhuitl (2017); Williams et al. (2017); Dhingra et al. (2017) and Li et al. (2017a) for a few recent examples. Compared to these dialogue agents developed for individual domains, the composite task presents additional challenges to commonly used, *flat* RL approaches such as DQN (Mnih et al., 2015). The first challenge is reward sparsity. Dialogue policy learning for composite tasks requires exploration in a much larger state-action space, and it often takes many more conversation turns between user and agent to fulfill a task, leading to a much longer trajectory. Thus, the reward signals (usually provided by users at the end of a conversation) are delayed and sparse. As we will show in this paper, typical flat RL methods such as DQN with naive ϵ-greedy exploration is rather inefficient. The second challenge is to satisfy slot constraints across subtasks. This requirement makes most of the existing methods of learning *multi-domain dialogue* agents (Cuayáhuitl, 2009; Gasic et al., 2015b) inapplicable: these methods train a

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2231–2240
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

collection of policies, one for each domain, and there is no cross-domain constraints required to successfully complete a dialogue. The third challenge is improved user experience: we find in our experiments that a flat RL agent tends to switch between different subtasks frequently when conversing with users. Such incoherent conversations lead to poor user experience, and are one of the main reasons that cause a dialogue session to fail.

In this paper, we address the above mentioned challenges by formulating the task using the mathematical framework of *options over MDPs* (Sutton et al., 1999), and proposing a method that combines deep reinforcement learning and hierarchical task decomposition to train a composite task-completion dialogue agent. At the heart of the agent is a dialogue manager, which consists of (1) a top-level dialogue policy that selects subtasks (options), (2) a low-level dialogue policy that selects primitive actions to complete a given subtask, and (3) a global state tracker that helps ensure all cross-subtask constraints be satisfied.

Conceptually, our approach exploits the structural information of composite tasks for efficient exploration. Specifically, in order to mitigate the reward sparsity issue, we equip our agent with an evaluation module (internal critic) that gives intrinsic reward signals, indicating how likely a particular subtask is completed based on its current state generated by the global state tracker. Such intrinsic rewards can be viewed as heuristics that encourage the agent to focus on solving a subtask before moving on to another subtask. Our experiments show that such intrinsic rewards can be used inside a hierarchical RL agent to make exploration more efficient, yielding a significantly reduced state-action space for decision making. Furthermore, it leads to a better user experience, as the resulting conversations switch between subtasks less frequently.

To the best of our knowledge, this is the first work that strives to develop a composite task-completion dialogue agent. Our main contributions are three-fold:

- We formulate the problem in the mathematical framework of options over MDPs.
- We propose a hierarchical deep reinforcement learning approach to efficiently learning the dialogue manager that operates at different temporal scales.
- We validate the effectiveness of the proposed approach in a travel planning task on simulated as well as real users.

2 Related Work

Task-completion dialogue systems have attracted numerous research efforts. Reinforcement learning algorithms hold the promise for dialogue policy optimization over time with experience (Scheffler and Young, 2000; Levin et al., 2000; Young et al., 2013; Williams et al., 2017). Recent advances in deep learning have inspired many deep reinforcement learning based dialogue systems that eliminate the need for feature engineering (Su et al., 2016; Cuayáhuitl, 2017; Williams et al., 2017; Dhingra et al., 2017; Li et al., 2017a).

All the work above focuses on single-domain problems. Extensions to composite-domain dialogue problems are non-trivial due to several reasons: the state and action spaces are much larger, the trajectories are much longer, and in turn reward signals are much more sparse. All these challenges can be addressed by hierarchical reinforcement learning (Sutton et al., 1999, 1998; Singh, 1992; Dietterich, 2000; Barto and Mahadevan, 2003), which decomposes a complicated task into simpler subtasks, possibly in a recursive way. Different frameworks have been proposed, such as Hierarchies of Machines (Parr and Russell, 1997) and MAXQ decomposition (Dietterich, 2000). In this paper, we choose the *options framework* for its conceptual simplicity and generality (Sutton et al., 1998); more details are found in the next section. Our work is also motivated by hierarchical-DQN (Kulkarni et al., 2016) which integrates hierarchical value functions to operate at different temporal scales. The model achieved superior performance on a complicated ATARI game "Montezuma's Revenge" with a hierarchical structure.

A related but different extension to single-domain dialogues is multi-domain dialogues, where each domain is handled by a separate agent (Lison, 2011; Gasic et al., 2015a,b; Cuayáhuitl et al., 2016). In contrast to composite-domain dialogues studied in this paper, a conversation in a multi-domain dialogue normally involves one domain, so completion of a task does *not* require solving sub-tasks in different domains. Consequently, work on multi-domain dialogues focuses on different technical challenges such as transfer learning across different domains (Gasic et al., 2015a) and domain selection (Cuayáhuitl et al., 2016).

3 Dialogue Policy Learning

Our composite task-completion dialogue agent consists of four components: (1) an LSTM-based language understanding module (Hakkani-Tür et al., 2016; Yao et al., 2014) for identifying user intents and extracting associated slots; (2) a state tracker for tracking the dialogue state; (3) a dialogue policy which selects the next action based on the current state; and (4) a model-based natural language generator (Wen et al., 2015) for converting agent actions to natural language responses. Typically, a dialogue manager contains a state tracker and a dialogue policy. In our implementation, we use a *global* state tracker to maintain the dialogue state by accumulating information across all subtasks, thus helping ensure all inter-subtask constraints be satisfied. In the rest of this section, we will describe the dialogue policy in details.

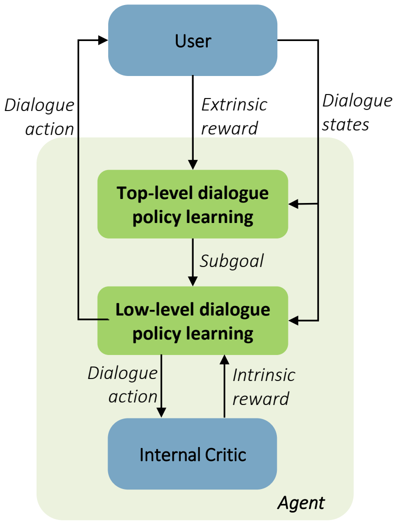


Figure 1: Overview of a composite task-completion dialogue agent.

3.1 Options over MDPs

Consider the following process of completing a composite task (e.g., travel planning). An agent first selects a subtask (e.g., book-flight-ticket), then takes a sequence of actions to gather related information (e.g., departure time, number of tickets, destination, etc.) until all users' requirements are met and the subtask is completed, and finally chooses the next subtask (e.g., reserve-hotel) to complete. The composite task is fulfilled after all its subtasks are completed collectively. The above process has a natural hierarchy: a top-level process selects which subtasks to complete, and a low-level process chooses primitive actions to complete the selected subtask. Such hierarchical decision making processes can be formulated in the *options* framework (Sutton et al., 1999), where options generalize primitive actions to higher-level actions. Different from the traditional MDP setting where an agent can only choose a primitive action at each time step, with options the agent can choose a "multi-step" action which for example could be a sequence of primitive actions for completing a subtask. As pointed out by Sutton et al. (1999), options are closely related to actions in a family of decision problems known as semi-Markov decision processes.

Following Sutton et al. (1999), an option consists of three components: a set of states where the option can be initiated, an intra-option policy that selects primitive actions while the option is in control, and a termination condition that specifies when the option is completed. For a composite task such as travel planning, subtasks like *book-flight-ticket* and *reserve-hotel* can be modeled as options. Consider, for example, the option *book-flight-ticket*: its initiation state set contains states in which the tickets have not been issued or the destination of the trip is long away enough that a flight is needed; it has an intra-option policy for requesting or confirming information regarding departure date and the number of seats, etc.; it also has a termination condition for confirming that all information is gathered and correct so that it is ready to issue the tickets.

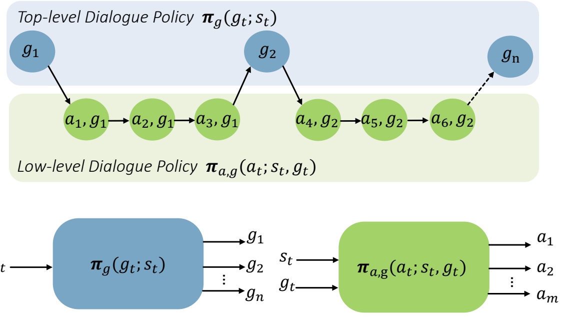

Top-level Dialogue Policy $\boldsymbol{\pi}_g(g_t; s_t)$
g_1
g_2
g_n
a_1, g_1
a_2, g_1
a_3, g_1
a_4, g_2
a_5, g_2
a_6, g_2
Low-level Dialogue Policy $\boldsymbol{\pi}_{a,g}(a_t; s_t, g_t)$
s_t
$\boldsymbol{\pi}_g(g_t; s_t)$
g_1
g_2
g_n
s_t
g_t
$\boldsymbol{\pi}_{a,g}(a_t; s_t, g_t)$
a_1
a_2
a_m

Figure 2: Illustration of a two-level hierarchical dialogue policy learner.

3.2 Hierarchical Policy Learning

The intra-option is a conventional policy over primitive actions, we can consider an inter-option policy over sequences of options in much the same way as we consider the intra-option policy over sequences of actions. We propose a method that combines deep reinforcement learning and hierarchical value functions to learn a composite task-completion dialogue agent as shown in Figure 1. It is a two-level hierarchical reinforcement learning agent that consists of a top-level dialogue policy π_g and a low-level dialogue policy $\pi_{a,g}$, as shown in Figure 2. The top-level policy π_g perceives state s from the environment and selects a subtask $g \in \mathcal{G}$, where $\mathcal{G}$ is the set of all possible subtasks. The low-level policy $\pi_{a,g}$ is shared by all options. It takes as input a state s and a subtask g, and outputs a primitive action $a \in \mathcal{A}$, where $\mathcal{A}$ is the set of primitive actions of all subtasks. The subtask g remains a constant input to $\pi_{a,g}$, until a terminal state is reached to terminate g. The internal critic in the dialogue manager provides intrinsic reward $r_t^i(g_t)$, indicating whether the subtask g_t at hand has been solved; this signal is used to optimize $\pi_{a,g}$. Note that the state s contains global information, in that it keeps track of information for all subtasks.

Naturally, we aim to optimize the low-level policy $\pi_{a,g}$ so that it maximizes the following cumulative intrinsic reward at every step t:

$$\max_{\pi_{a,g}} \mathbb{E}\Big[\sum_{k \geq 0} \gamma^k r_{t+k}^i \Big| s_t = s, g_t = g, a_{t+k} = \pi_{a,g}(s_{t+k})\Big]$$

where r_{t+k}^i denotes the reward provided by the internal critic at step $t+k$. Similarly, we want the top-level policy π_g to optimize the cumulative extrinsic reward at every step t:

$$\max_{\pi_g} \mathbb{E}\Big[\sum_{k \geq 0} \gamma^k r_{t+k}^e \Big| s_t = s, a_{t+k} = \pi_g(s_{t+k})\Big] \,,$$

where r_{t+k}^e is the reward received from the environment at step $t+k$ when a new subtask starts.

Both the top-level and low-level policies can be learned with deep Q-learning methods, like DQN. Specifically, the top-level dialogue policy estimates the optimal Q-function that satisfies the following:

$$\begin{aligned} Q_1^*(s,g) = \mathbb{E}\Big[& \sum_{k=0}^{N-1} \gamma^k r_{t+k}^e + \\ & \gamma^N \cdot \max\nolimits_{g'} Q_1^*(s_{t+N}, g') | s_t = s, g_t = g\Big], \quad (1) \end{aligned}$$

where N is the number of steps that the low-level dialogue policy (intra-option policy) needs to accomplish the subtask. g' is the agent's next subtask in state s_{t+N}. Similarly, the low-level dialogue policy estimates the Q-function that satisfies the following:

$$\begin{aligned} Q_2^*(s,a,g) = \mathbb{E}\Big[& r_t^i + \\ & \gamma \cdot \max_{a_{t+1}} Q_2^*(s_{t+1}, a_{t+1}, g) | s_t = s, g_t = g\Big] \,. \end{aligned}$$

Both $Q_1^*(s,g)$ and $Q_2^*(s,a,g)$ are represented by neural networks, $Q_1(s,g;\theta_1)$ and $Q_2(s,a,g;\theta_2)$, parameterized by θ_1 and θ_2, respectively.

The top-level dialogue policy tries to minimize the following loss function at each iteration i:

$$\begin{aligned} \mathcal{L}_1(\theta_{1,i}) = \mathbb{E}_{(s,g,r^e,s') \sim \mathcal{D}_1}[(y_i - Q_1(s,g;\theta_{1,i}))^2] \\ y_i = r^e + \gamma^N \max_{g'} Q_1(s', g', \theta_{1,i-1}) \,, \end{aligned}$$

where, as in Equation (1), $r^e = \sum_{k=0}^{N-1} \gamma^k r_{t+k}^e$ is the discounted sum of reward collected when subgoal g is being completed, and N is the number of steps g is completed.

The low-level dialogue policy minimizes the following loss at each iteration i using:

$$\begin{aligned} \mathcal{L}_2(\theta_{2,i}) &= \mathbb{E}_{(s,g,a,r^i,s') \sim \mathcal{D}_2}[(y_i - \\ & \qquad Q_2(s,g,a;\theta_{2,i}))^2] \\ y_i &= r^i + \gamma \max_{a'} Q_2(s', g, a', \theta_{2,i-1}) \,. \end{aligned}$$

We use SGD to minimize the above loss functions. The gradient for the top-level dialogue policy yields:

$$\begin{aligned} \nabla_{\theta_{1,i}} L_1(\theta_{1,i}) = \mathbb{E}_{(s,g,r^e,s') \sim \mathcal{D}_1}[(r^e + \\ \gamma^N \max_{g'} Q_2(s', g', \theta_{1,i-1}) - Q_1(s,g,\theta_{1,i})) \\ \nabla_{\theta_{1,i}} Q_1(s,g,\theta_{1,i})] \end{aligned} \tag{2}$$

The gradient for the low-level dialogue policy yields:

$$\begin{aligned} \nabla_{\theta_{2,i}} L_2(\theta_{2,i}) = \mathbb{E}_{(s,g,a,r^i,s') \sim \mathcal{D}_2}[(r^i + \\ \gamma \max_{a'} Q_2(s', g, a', \theta_{2,i-1}) - Q_2(s,g,a,\theta_{2,i})) \\ \nabla_{\theta_{2,i}} Q_2(s,g,a,\theta_{2,i})] \end{aligned} \tag{3}$$

Following previous studies, we apply two most commonly used performance boosting methods: target networks and experience replay. Experience replay tuples (s, g, r^e, s') and (s, g, a, r^i, s'), are sampled from the experience replay buffers $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively. A detailed summary of the learning algorithm for the hierarchical dialogue policy is provided in Appendix B.

4 Experiments and Results

To evaluate the proposed method, we conduct experiments on the composite task-completion dialogue task of travel planning.

4.1 Dataset

In the study, we made use of a human-human conversation data derived from a publicly available multi-domain dialogue corpus[1] (El Asri et al., 2017), which was collected using the Wizard-of-Oz approach. We made a few changes to the schema of the data set for the composite task-completion dialogue setting. Specifically, we added inter-subtask constraints as well as user preferences (soft constraints). The data was mainly used to create simulated users, as will be explained below shortly.

4.2 Baseline Agents

We benchmark the proposed *HRL agent* against three baseline agents:

- A *Rule Agent* uses a sophisticated hand-crafted dialogue policy, which requests and informs a hand-picked subset of necessary slots, and then confirms with the user about the reserved tickets.
- A *Rule+ Agent* requests and informs all the slots in a pre-defined order exhaustedly, and then confirms with the user about the reserved tickets. The average turn of this agent is longer than that of the *Rule* agent.
- A *flat RL Agent* is trained with a standard flat deep reinforcement learning method (DQN) which learns a flat dialogue policy using extrinsic rewards only.

4.3 User Simulator

Training reinforcement learners is challenging because they need an environment to interact with. In the dialogue research community, it is common to use simulated users as shown in Figure 3 for this purpose (Schatzmann et al., 2007; Asri et al., 2016). In this work, we adapted the publicly-available user simulator, developed by Li et al. (2016), to the composite task-completion dialogue setting using the human-human conversation data described in Section 4.1.[2] During training, the simulator provides the agent with an (extrinsic) reward signal at the end of the dialogue. A dialogue is considered to be successful only when a travel plan is made successfully, and the information provided by the agent satisfies user's constraints. At the end of each dialogue, the agent receives a positive reward of $2{*}max\_turn$ ($max\_turn = 60$ in our experiments) for success, or a negative reward of $-max\_turn$ for failure. Furthermore, at each turn, the agent receives a reward of -1 so that shorter dialogue sessions are encouraged.

User Goal A user goal is represented by a set of slots, indicating the user's request, requirement and preference. For example, an *inform slot*, such as dst_city="Honolulu", indicates a user requirement, and a *request slot*, such as price="?", indicates a user asking the agent for the information.

In our experiment, we compiled a list of user goals using the slots collected from the human-human conversation data set described in Section 4.1, as follows. We first extracted all the slots that appear in dialogue sessions. If a slot has multiple values, like "or_city=[San Francisco, San Jose]", we consider it as a user preference (soft constraint) which the user may later revise its value to explore different options in the course of the dialogue. If a slot has only one value, we treat it as a user requirement (hard constraint), which is unlikely negotiable. If a slot is with value "?", we treat it as a user request. We removed those slots from user goals if their values do not exist in our database. The compiled set of user goals contains 759 entries, each containing slots from at least two subtasks: *book-flight-ticket* and *reserve-hotel*.

User Type To compare different agents' ability to adapt to user preferences, we also constructed three additional user goal sets, representing three different types of (simulated) users, respectively:

- *Type A*: All the informed slots in a user goal have a single value. These users have hard constraints for both the flight and hotel, and have no preference on which subtask to accomplish first.
- *Type B*: At least one of informed slots in the *book-flight-ticket* subtask can have multiple values, and the user (simulator) prefers to start with the *book-flight-ticket* subtask. If the user receives "no ticket available" from the agent during the conversation, she is willing to explore alternative slot values.

[1] https://datasets.maluuba.com/Frames

[2] A detailed description of the user simulator is presented in Appendix A.

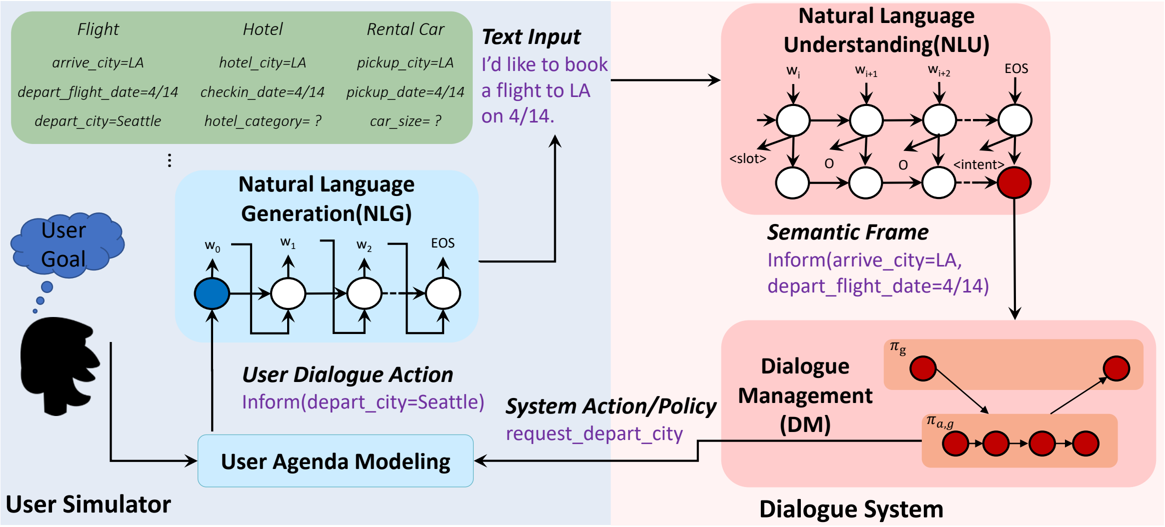


Figure 3: Illustration of the Composite Task-Completion dialogue System

- *Type C*: Similar to *Type B*, at least one of informed slots of the *reserve-hotel* subtask in a user goal can have multiple values. The user prefers to start with the *reserve-hotel* subtask. If the user receives a "no room available" response from the agent, she is willing to explore alternative slot values.

4.4 Implementation

For the RL agent, we set the size of hidden layer to 80. For the HRL agent, both top-level and low-level dialogue policies had a hidden layer size of 80. RMSprop was applied to optimize the parameters. We set batch size to 16. During training, we used the ϵ-greedy strategy for exploration. For each simulation epoch, we simulated 100 dialogues and stored these state transition tuples in an experience replay buffer. At the end of each simulation epoch, the model was updated with all the transition tuples in the buffer in a batch manner.

The experience replay strategy is critical to the success of deep reinforcement learning. In our experiments, at the beginning, we used a rule-based agent to run N ($N = 100$) dialogues to populate the experience replay buffer, which was an implicit way of imitation learning to initialize the RL agent. Then, the RL agent accumulated all the state transition tuples and flushes the replay buffer only when the current RL agent reached a success rate threshold no worse than that of the *Rule* agent.

This strategy was motivated by the following observation. The initial performance of an RL agent was often not strong enough to result in dialogue sessions with a reasonable success rate. With such data, it was easy for the agent to learn the locally optimal policy that "failed fast"; that is, the policy would finish the dialogue immediately, so that the agent could suffer the least amount of per-turn penalty. Therefore, we provided some rule-based examples that succeeded reasonably often, and did not flush the buffer until the performance of the RL agent reached an acceptable level. Generally, one can set the threshold to be the success rate of the *Rule* agent. To make a fair comparison, for the same type of users, we used the same *Rule* agent to initialize both the RL agent and the HRL agent.

4.5 Simulated User Evaluation

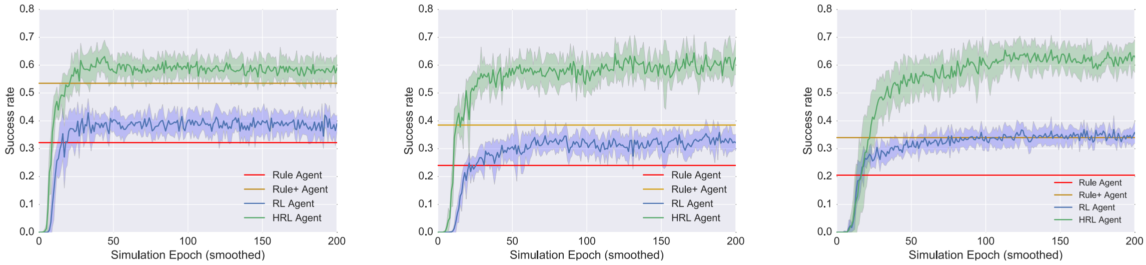
On the composite task-completion dialogue task, we compared the HRL agent with the baseline agents in terms of three metrics: success rate[3], average rewards, and the average number of turns per dialogue session.

Figure 4 shows the learning curves of all four agents trained on different types of users. Each learning curve was averaged over 10 runs. Table 1 shows the performance on test data. For all types of users, the HRL-based agent yielded more robust dialogue policies outperforming the hand-crafted rule-based agents and flat RL-based agent measured on success rate. It also needed fewer turns per dialogue session to accomplish a task than the rule-based agents and flat RL agent. The results

[3] Success rate is the fraction of dialogues where the tasks are successfully accomplished within the maximum turns.

	Type A			Type B			Type C		
Agent	Succ.	Turn	Reward	Succ.	Turn	Reward	Succ.	Turn	Reward
Rule	.322	46.2	-24.0	.240	54.2	-42.9	.205	54.3	-49.3
Rule+	*.535*	*82.0*	*-3.7*	*.385*	*110.5*	*-44.95*	*.340*	*108.1*	*-51.85*
RL	.437	45.6	-3.3	.340	52.2	-23.8	.348	49.5	-21.1
HRL	**.632**	**43.0**	**33.2**	**.600**	**44.5**	**26.7**	**.622**	**42.7**	**31.7**

Table 1: Performance of three agents on different User Types. Tested on 2000 dialogues using the best model during training. Succ.: success rate, Turn: average turns, Reward: average reward.



(a) Success Rate of User Type A

(b) Success Rate of User Type B

(c) Success Rate of User Type C

Figure 4: Learning curves of dialogue policies for different User Types under simulation

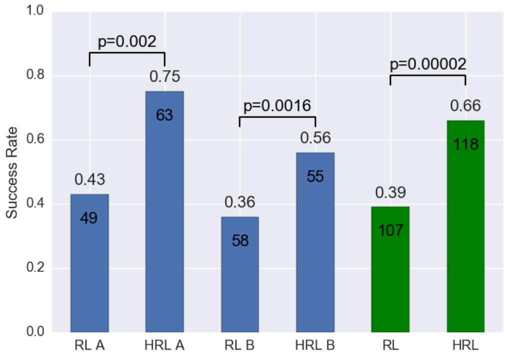


Figure 5: Performance of HRL agent versus RL agent tested with real users: success rate, number of tested dialogues and p-values are indicated on each bar; the rightmost green ones are for total (difference in mean is significant with $p < 0.01$).

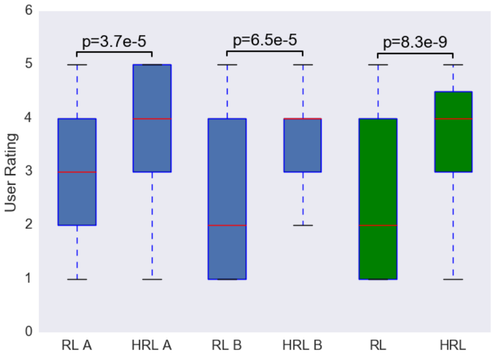


Figure 6: Distribution of user ratings for HRL agent versus RL agent, and total.

across all three types of simulated users suggest the following conclusions.

First, he HRL agent significantly outperformed the RL agent. This, to a large degree, was attributed to the use of the hierarchical structure of the proposed agent. Specifically, the top-level dialogue policy selected a subtask for the agent to focus on, one at a time, thus dividing a complex task into a sequence of simpler subtasks. The selected subtasks, combined with the use of intrinsic rewards, alleviated the sparse reward and long-horizon issues, and helped the agent explore more efficiently in the state-action space. As a result, as shown in Figure 4 and Table 1, the performance of the HRL agent on types B and C users (who may need to go back to revise some slots during the dialogue) does not drop much compared to type A users, despite the increased search space in the former. Additionally, we observed a large drop in the performance of the RL Agent due to the increased complexity of the task, which required more dialogue turns and posed a challenge for temporal credit assignment.

Second, the HRL agent learned much faster than the RL agent. The HRL agent could reach the same level of performance with a smaller number of simulation examples than the RL agent,

Table 2: Sample dialogue by RL and HRL agents with real user: Left column shows the dialogue with the RL agent; Right column shows the dialogue with the HRL agent; bolded slots are the joint constraints between two subtasks.

<table>
<tr><th colspan="2">User Goal</th></tr>
<tr><td><i>reserve-hotel</i> subtask:
{
“request_slots”: {
“hotel_price”: “?”
“hotel_date_checkout”: “?”
“hotel_name”: “?”
},
“inform_slots”: {
<b>“hotel_city”: “Cancun”</b>,
<b>“hotel_numberofpeople”: “3”</b>,
<b>“hotel_date_checkin”: “09/20”</b>
}
}</td><td><i>book-flight-ticket</i> subtask:
{
“request_slots”: {
“price”: “?”
“return_time_dep”: “?”
“return_date_dep”: “?”
“depart_time_dep”: “?”
“seat”: “?”
}
“inform_slots”: {
“or_city”: “Campinas”,
<b>“dst_city”: “Cancun”</b>,
<b>“numberofpeople”: “3”</b>,
<b>“depart_date: ”9/20”</b>
}
}</td></tr>
<tr><th>RL agent</th><th>HRL agent</th></tr>
<tr><td><i>Agent</i>: Welcome to our travel agent. May I know your original city, destination city and departure date?
<i>User</i>: I depart from Campinas to Cancun and prefer to leave on 09-20.
<i>Agent</i>: There is a return flight on 9/26 and hotel checkout date is on 9/26 too.
<i>User</i>: My wife and son will travel with me, three people in total.
<i>Agent</i>: There is a hotel called Hotel Tropic available for you.
<i>User</i>: What depart time is available?
<i>Agent</i>: Departure time is at 10:00AM.
<i>User</i>: What is the departure time of return flight?
<i>Agent</i>: It is 4:00PM.
<i>User</i>: Which seat is available?
<i>Agent</i>: Only Business class available.
<i>User</i>: What’s the price for the flight?
<i>Agent</i>: Total is 1399 dollars for flight.
<i>User</i>: What’s the price for the hotel?
<i>Agent</i>: It is 1091 dollars for hotel.
<i>User</i>: Could you please help me book the tickets?
<i>Agent</i>: Departure flight on 9/20 and hotel checkin date on 9/20 are available.
<i>User</i>: Could you please help me book the tickets?
<i>Agent</i>: Okay. Tickets are issued.
<i>User</i>: Thanks!
<i>Agent</i>: Thanks!</td><td><i>Agent</i>: Welcome to our travel agent. May I know your original city, destination city and departure date?
<i>User</i>: I depart from Campinas to Cancun and prefer to leave on 09-20.
<i>Agent</i>: Only Business class seat is available.
<i>User</i>: What is the depart time?
<i>Agent</i>: Departure time is at 10:00AM.
<i>User</i>: My wife and son will travel with me, three people in total.
<i>Agent</i>: There is one return flight departed at 4:00PM.
<i>User</i>: What is the date of the return flight?
<i>Agent</i>: It is on 9/26 and hotel checkout date is also 9/26.
<i>User</i>: What’s the price for the flight?
<i>Agent</i>: Total is 1399 dollars for flight.
<i>User</i>: Which hotel is available?
<i>Agent</i>: A hotel called Hotel Tropic is available for you.
<i>User</i>: What’s the price for the hotel?
<i>Agent</i>: It is 1091 dollars for hotel.
<i>User</i>: Could you please help me book the tickets?
<i>Agent</i>: Okay. Tickets are issued.
<i>User</i>: Thanks!
<i>Agent</i>: Thanks!</td></tr>
<tr><td><b>Success</b>: but the agent does not behave naturally where is an inter-subtask transition from <i>book-flight-ticket</i> to <i>reserve-hotel</i>.</td><td><b>Success</b></td></tr>
</table>

demonstrating that the hierarchical dialogue policies were more sample-efficient than flat RL policy and could significantly reduce the sample complexity on complex tasks.

Finally, we also found that the *Rule+* and *flat* RL agents had comparable success rates, as shown in Figure 4. However, a closer look at the correlation between success rate and the average number of turns in Table 1 suggests that the *Rule+* agent required more turns which adversely affects its success, whilst the *flat* RL agent achieves similar success with much less number of turns in all the user types. It suffices to say that our hierarchical RL agent outperforms all in terms of success rate as depicted in Figure 4.

4.6 Human Evaluation

We further evaluated the agents, which were trained on simulated users, against real human users, recruited from the authors’ affiliation. We conducted the study using the HRL and RL agents, each tested against two types of users: *Type A* users who had no preference for subtask, and *Type B* users who preferred to complete the *book-flight-ticket* subtask first. Note that *Type C* users were symmetric to Type B ones, so were not included in the study. We compared two (agent, user type) pairs: {RL A, HRL A} and {RL B, HRL B}; in other words, four agents were trained against their specific user types. In each dialogue session, one of the agents was randomly picked to converse with a user. The user was presented with a user goal sampled from our corpus, and was instructed to converse with the agent to complete the task. If one of the slots in the goal had multiple values, the user had multiple choices for this slot and might revise the slot value when the agent replied with a message like "No ticket is available" during the conversation. At the end of each session, the user was asked to give a rating on a scale from 1 to 5 based on the naturalness and coherence of the dialogue. (1 is the worst rating, and 5 the best). We collected a total of 225 dialogue sessions from 12 human users.

Figure 5 presents the performance of these agents against real users in terms of success rate. Figure 6 shows the comparison in user rating. For all the cases, the HRL agent was consistently better than the RL agent in terms of success rate and user rating. Table 2 shows a sample dialogue session. We see that the HRL agent produced a more coherent conversation, as it switched among subtasks much less frequently than the *flat* RL agent.

5 Discussion and Conclusions

This paper considers composite task-completion dialogues, where a set of subtasks need to be fulfilled collectively for the entire dialogue to be successful. We formulate the policy learning problem using the options framework, and take a hierarchical deep RL approach to optimizing the policy. Our experiments, both on simulated and real users, show that the hierarchical RL agent significantly outperforms a flat RL agent and rule-based agents. The hierarchical structure of the agent also improves the coherence of the dialogue flow.

The promising results suggest several directions for future research. First, the hierarchical RL approach demonstrates strong adaptation ability to tailor the dialogue policy to different types of users. This motivates us to systematically investigate its use for dialogue *personalization*. Second, our hierarchical RL agent is implemented using a two-level dialogue policy. But more complex tasks might require multiple levels of hierarchy. Thus, it is valuable to extend our approach to handle such deep hierarchies, where a subtask can invoke another subtask and so on, taking full advantage of the options framework. Finally, designing task hierarchies requires substantial domain knowledge and is time-consuming. This challenge calls for future work on automatic learning of hierarchies for complex dialogue tasks.

Acknowledgments

Baolin Peng was in part supported by General Research Fund of Hong Kong (14232816). We would like thank anonymous reviewers for their insightful comments.

References

Layla El Asri, Jing He, and Kaheer Suleman. 2016. A sequence-to-sequence model for user simulation in spoken dialogue systems. In *Interspeech 2016*, pages 1151–1155. ISCA.

Andrew G. Barto and Sridhar Mahadevan. 2003. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(1-2):41–77.

Heriberto Cuayáhuitl. 2009. Hierarchical reinforcement learning for spoken dialogue systems.

Heriberto Cuayáhuitl. 2017. SimpleDS: A simple deep reinforcement learning dialogue system. In *Dialogues with Social Robots*, pages 109–118. Springer.

Heriberto Cuayáhuitl, Seunghak Yu, Ashley Williamson, and Jacob Carse. 2016. Deep reinforcement learning for multi-domain dialogue systems. *arXiv preprint arXiv:1611.08675*.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2017. End-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Thomas G. Dietterich. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. *arXiv preprint arXiv:1704.00057*.

Milica Gasic, Dongho Kim, Pirros Tsiakoulis, and Steve J. Young. 2015a. Distributed dialogue policies for multi-domain statistical dialogue management. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, pages 5371–5375. IEEE.

Milica Gasic, Nikola Mrksic, Pei-hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2015b. Policy committee for adaptation in multidomain spoken dialogue systems. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2015, Scottsdale, AZ, USA, December 13-17, 2015*, pages 806–812. IEEE.

Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. 2016. Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association*.

Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3675–3683.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Trans. Speech and Audio Processing*, 8(1):11–23.

Xiujun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao. 2017a. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017b. Investigation of language understanding impact for reinforcement learning based dialogue systems. *arXiv preprint arXiv:1703.07055*.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*.

Pierre Lison. 2011. Multi-policy dialogue management. In *Proceedings of the SIGDIAL 2011*, pages 294–300. The Association for Computer Linguistics.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Ronald Parr and Stuart J. Russell. 1997. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems 10, [NIPS Conference, Denver, Colorado, USA, 1997]*, pages 1043–1049. The MIT Press.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve J. Young. 2007. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, April 22-27, 2007, Rochester, New York, USA*, pages 149–152. The Association for Computational Linguistics.

Jost Schatzmann and Steve Young. 2009. The hidden agenda user simulation model. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):733–747.

Konrad Scheffler and Steve J. Young. 2000. Probabilistic simulation of human-machine dialogues. In *IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP 2000, 5-9 June, 2000, Hilton Hotel and Convention Center, Istanbul, Turkey*, pages 1217–1220. IEEE.

Satinder P. Singh. 1992. Reinforcement learning with a hierarchy of abstract models. In *Proceedings of the 10th National Conference on Artificial Intelligence. San Jose, CA, July 12-16, 1992.*, pages 202–207. AAAI Press / The MIT Press.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689*.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. 1998. Intra-option learning about temporally abstract actions. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, pages 556–564.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. 1999. Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Peihao Su, David Vandyke, and Steve J. Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1711–1721.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: Practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. 2014. Spoken language understanding using long short-term memory neural networks. In *2014 IEEE Spoken Language Technology Workshop, SLT 2014, South Lake Tahoe, NV, USA, December 7-10, 2014*, pages 189–194.

Steve J. Young, Milica Gasic, Blaise Thomson, and Jason D. Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Why We Need New Evaluation Metrics for NLG

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry and **Verena Rieser**
School of Mathematical and Computer Sciences
Heriot-Watt University, Edinburgh
j.novikova, o.dusek, ac293, v.t.rieser@hw.ac.uk

Abstract

The majority of NLG evaluation relies on automatic metrics, such as BLEU. In this paper, we motivate the need for novel, system- and data-independent automatic evaluation methods: We investigate a wide range of metrics, including state-of-the-art word-based and novel grammar-based ones, and demonstrate that they only weakly reflect human judgements of system outputs as generated by data-driven, end-to-end NLG. We also show that metric performance is data- and system-specific. Nevertheless, our results also suggest that automatic metrics perform reliably at system-level and can support system development by finding cases where a system performs poorly.

1 Introduction

Automatic evaluation measures, such as BLEU (Papineni et al., 2002), are used with increasing frequency to evaluate Natural Language Generation (NLG) systems: Up to 60% of NLG research published between 2012–2015 relies on automatic metrics (Gkatzia and Mahamood, 2015). Automatic evaluation is popular because it is cheaper and faster to run than human evaluation, and it is needed for automatic benchmarking and tuning of algorithms. The use of such metrics is, however, only sensible if they are known to be sufficiently correlated with human preferences. This is rarely the case, as shown by various studies in NLG (Stent et al., 2005; Belz and Reiter, 2006; Reiter and Belz, 2009), as well as in related fields, such as dialogue systems (Liu et al., 2016), machine translation (MT) (Callison-Burch et al., 2006), and image captioning (Elliott and Keller, 2014; Kilickaya et al., 2017). This paper follows on from the above previous work and presents another evaluation study into automatic metrics with the aim to firmly establish the need for new metrics. We consider this paper to be the most complete study to date, across metrics, systems, datasets and domains, focusing on recent advances in data-driven NLG. In contrast to previous work, we are the first to:

- Target end-to-end data-driven NLG, where we compare 3 different approaches. In contrast to NLG methods evaluated in previous work, our systems can produce ungrammatical output by (a) generating word-by-word, and (b) learning from noisy data.
- Compare a large number of 21 automated metrics, including novel grammar-based ones.
- Report results on two different domains and three different datasets, which allows us to draw more general conclusions.
- Conduct a detailed error analysis, which suggests that, while metrics can be reasonable indicators at the system-level, they are not reliable at the sentence-level.
- Make all associated code and data publicly available, including detailed analysis results.[1]

2 End-to-End NLG Systems

In this paper, we focus on recent end-to-end, data-driven NLG methods, which jointly learn sentence planning and surface realisation from non-aligned data (Dušek and Jurčíček, 2015; Wen et al., 2015; Mei et al., 2016; Wen et al., 2016; Sharma et al., 2016; Dušek and Jurčíček, 2016, Lampouras and Vlachos, 2016). These approaches do not require costly semantic alignment between Meaning Representations (MR) and human references (also referred to as "ground truth" or "targets"), but are

[1] Available for download at: https://github.com/jeknov/EMNLP_17_submission

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2241–2252
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

System	Dataset			Total
	BAGEL	SFREST	SFHOTEL	
LOLS	202	581	398	1,181
RNNLG	-	600	477	1,077
TGEN	202	-	-	202
Total	404	1,181	875	2,460

Table 1: Number of NLG system outputs from different datasets and systems used in this study.

based on parallel datasets, which can be collected in sufficient quality and quantity using effective crowdsourcing techniques, e.g. (Novikova et al., 2016), and as such, enable rapid development of NLG components in new domains. In particular, we compare the performance of the following systems:

- **RNNLG:**[2] The system by Wen et al. (2015) uses a Long Short-term Memory (LSTM) network to jointly address sentence planning and surface realisation. It augments each LSTM cell with a gate that conditions it on the input MR, which allows it to keep track of MR contents generated so far.
- **TGEN:**[3] The system by Dušek and Jurčíček (2015) learns to incrementally generate deep-syntax dependency trees of candidate sentence plans (i.e. which MR elements to mention and the overall sentence structure). Surface realisation is performed using a separate, domain-independent rule-based module.
- **LOLS:**[4] The system by Lampouras and Vlachos (2016) learns sentence planning and surface realisation using Locally Optimal Learning to Search (LOLS), an imitation learning framework which learns using BLEU and ROUGE as non-decomposable loss functions.

3 Datasets

We consider the following crowdsourced datasets, which target utterance generation for spoken dialogue systems. Table 1 shows the number of system outputs for each dataset. Each data instance consists of one MR and one or more natural language references as produced by humans, such as the following example, taken from the BAGEL dataset:[5]

MR: inform(name=X, area=X, pricerange=moderate, type=restaurant)
Reference: *"X is a moderately priced restaurant in X."*

- **SFHOTEL & SFREST** (Wen et al., 2015) provide information about hotels and restaurants in San Francisco. There are 8 system dialogue act types, such as *inform*, *confirm*, *goodbye* etc. Each domain contains 12 attributes, where some are common to both domains, such as *name, type, pricerange, address, area,* etc., and the others are domain-specific, e.g. *food* and *kids-allowed* for restaurants; *hasinternet* and *dogs-allowed* for hotels. For each domain, around 5K human references were collected with 2.3K unique human utterances for SFHOTEL and 1.6K for SFREST. The number of unique system outputs produced is 1181 for SFREST and 875 for SFHOTEL.
- **BAGEL** (Mairesse et al., 2010) provides information about restaurants in Cambridge. The dataset contains 202 aligned pairs of MRs and 2 corresponding references each. The domain is a subset of SFREST, including only the *inform* act and 8 attributes.

4 Metrics

4.1 Word-based Metrics (WBMs)

NLG evaluation has borrowed a number of automatic metrics from related fields, such as MT, summarisation or image captioning, which compare output texts generated by systems to ground-truth references produced by humans. We refer to this group as word-based metrics. In general, the higher these scores are, the better or more similar to the human references the output is.[6] The following order reflects the degree these metrics move from simple n-gram overlap to also considering term frequency (TF-IDF) weighting and semantically similar words.

- **Word-overlap Metrics (WOMs):** We consider frequently used metrics, including TER (Snover et al., 2006), BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), NIST (Doddington, 2002), LEPOR (Han et al., 2012), CIDER (Vedantam et al., 2015), and METEOR (Lavie and Agarwal, 2007).
- **Semantic Similarity (SIM):** We calculate the Semantic Text Similarity measure designed by Han et al. (2013). This measure is based on distributional similarity and Latent Semantic Analysis

[2] https://github.com/shawnwun/RNNLG

[3] https://github.com/UFAL-DSG/tgen

[4] https://github.com/glampouras/JLOLS_NLG

[5] Note that we use lexicalised versions of SFHOTEL and SFREST and a partially lexicalised version of BAGEL, where proper names and place names are replaced by placeholders ("X"), in correspondence with the outputs generated by the systems, as provided by the system authors.

[6] Except for TER whose scale is reversed.

(LSA) and is further complemented with semantic relations extracted from WordNet.

4.2 Grammar-based metrics (GBMs)

Grammar-based measures have been explored in related fields, such as MT (Giménez and Màrquez, 2008) or grammatical error correction (Napoles et al., 2016), and, in contrast to WBMs, do not rely on ground-truth references. To our knowledge, we are the first to consider GBMs for sentence-level NLG evaluation. We focus on two important properties of texts here – readability and grammaticality:

- **Readability** quantifies the difficulty with which a reader understands a text, as used for e.g. evaluating summarisation (Kan et al., 2001) or text simplification (Francois and Bernhard, 2014). We measure readability by the Flesch Reading Ease score (RE) (Flesch, 1979), which calculates a ratio between the number of characters per sentence, the number of words per sentence, and the number of syllables per word. Higher RE score indicates a less complex utterance that is easier to read and understand. We also consider related measures, such as characters per utterance (**len**) and per word (**cpw**), words per sentence (**wps**), syllables per sentence (**sps**) and per word (**spw**), as well as polysyllabic words per utterance (**pol**) and per word (**ppw**). The higher these scores, the more complex the utterance.

- **Grammaticality:** In contrast to previous NLG methods, our corpus-based end-to-end systems can produce ungrammatical output by (a) generating word-by-word, and (b) learning from noisy data. As a first approximation of grammaticality, we measure the number of misspellings (**msp**) and the parsing score as returned by the Stanford parser (**prs**). The lower the **msp**, the more grammatically correct an utterance is. The Stanford parser score is not designed to measure grammaticality, however, it will generally prefer a grammatical parse to a non-grammatical one.[7] Thus, lower parser scores indicate less grammatically-correct utterances. In future work, we aim to use specifically designed grammar-scoring functions, e.g. (Napoles et al., 2016), once they become publicly available.

5 Human Data Collection

To collect human rankings, we presented the MR together with 2 utterances generated by different systems side-by-side to crowdworkers, which were asked to score each utterance on a 6-point Likert scale for:

- **Informativeness:** *Does the utterance provide all the useful information from the meaning representation?*
- **Naturalness:** *Could the utterance have been produced by a native speaker?*
- **Quality:** *How do you judge the overall quality of the utterance in terms of its grammatical correctness and fluency?*

Each system output (see Table 1) was scored by 3 different crowdworkers. To reduce participants' bias, the order of appearance of utterances produced by each system was randomised and crowdworkers were restricted to evaluate a maximum of 20 utterances. The crowdworkers were selected from English-speaking countries only, based on their IP addresses, and asked to confirm that English was their native language.

To assess the reliability of ratings, we calculated the intra-class correlation coefficient (ICC), which measures inter-observer reliability on ordinal data for more than two raters (Landis and Koch, 1977). The overall ICC across all three datasets is 0.45 ($p < 0.001$), which corresponds to a moderate agreement. In general, we find consistent differences in inter-annotator agreement per system and dataset, with lower agreements for LOLS than for RNNLG and TGEN. Agreement is highest for the SFHOTEL dataset, followed by SFREST and BAGEL (details provided in supplementary material).

6 System Evaluation

Table 2 summarises the individual systems' overall corpus-level performance in terms of automatic and human scores (details are provided in the supplementary material).

All WOMs produce similar results, with SIM showing different results for the restaurant domain (BAGEL and SFREST). Most GBMs show the same trend (with different levels of statistical significance), but RE is showing inverse results. System performance is dataset-specific: For WBMs, the LOLS system consistently produces better results on BAGEL compared to TGEN, while for SFREST and SFHOTEL, LOLS is outperformed by RNNLG in

[7] http://nlp.stanford.edu/software/parser-faq.shtml

metric	Bagel		SFHotel		SFRest	
	TGen	LOLS	RNNLG	LOLS	RNNLG	LOLS
WOMs		More overlap	More overlap*		More overlap*	
sim	More similar		More similar*			More similar
GBMs	Better grammar(*)		Better grammar(*)		Better grammar	
re		More complex*		More complex*		More complex*
inform	4.77(Sd=1.09)	**4.91**(Sd=1.23)	**5.47***(Sd=0.81)	5.27(Sd=1.02)	**5.29***(Sd=0.94)	5.16(Sd=1.07)
natural	**4.76**(Sd=1.26)	4.67(Sd=1.25)	**4.99***(Sd=1.13)	4.62(Sd=1.28)	**4.86** (Sd=1.13)	4.74(Sd=1.23)
quality	**4.77**(Sd=1.19)	4.54(Sd=1.28)	**4.54** (Sd=1.18)	4.53(Sd=1.26)	4.51 (Sd=1.14)	**4.58**(Sd=1.33)

Table 2: System performance per dataset (summarised over metrics), where "*" denotes $p < 0.05$ for all the metrics and "(*)" shows significance on $p < 0.05$ level for the majority of the metrics.

terms of WBMs. We observe that human *informativeness* ratings follow the same pattern as WBMs, while the average similarity score (sim) seems to be related to human *quality* ratings.

Looking at GBMs, we observe that they seem to be related to *naturalness* and *quality* ratings. Less complex utterances, as measured by readability (re) and word length (cpw), have higher *naturalness* ratings. More complex utterances, as measured in terms of their length (len), number of words (wps), syllables (sps, spw) and polysyllables (pol, ppw), have lower *quality* evaluation. Utterances measured as more grammatical are on average evaluated higher in terms of *naturalness*.

These initial results suggest a relation between automatic metrics and human ratings at system level. However, average scores can be misleading, as they do not identify worst-case scenarios. This leads us to inspect the correlation of human and automatic metrics for each MR-system output pair at utterance level.

7 Relation of Human and Automatic Metrics

7.1 Human Correlation Analysis

We calculate the correlation between automatic metrics and human ratings using the Spearman coefficient (ρ). We split the data per dataset and system in order to make valid pairwise comparisons. To handle outliers within human ratings, we use the median score of the three human raters.[8] Following Kilickaya et al. (2017), we use the Williams' test (Williams, 1959) to determine significant differences between correlations. Table 3 summarises the utterance-level correlation results between automatic metrics and human ratings, listing the best (i.e. highest absolute ρ) results for each type of metric (details provided in supplementary material). Our results suggest that:

- In sum, no metric produces an even moderate correlation with human ratings, independently of dataset, system, or aspect of human rating. This contrasts with our initially promising results on the system level (see Section 6) and will be further discussed in Section 8. Note that similar inconsistencies between document- and sentence-level evaluation results are observed in MT (Specia et al., 2010).
- Similar to our results in Section 6, we find that WBMs show better correlations to human ratings of *informativeness* (which reflects content selection), whereas GBMs show better correlations to *quality* and *naturalness*.
- Human ratings for *informativeness*, *naturalness* and *quality* are highly correlated with each other, with the highest correlation between the latter two ($\rho = 0.81$) reflecting that they both target surface realisation.
- All WBMs produce similar results (see Figure 1 and 2): They are strongly correlated with each other, and most of them produce correlations with human ratings which are *not* significantly different from each other. GBMs, on the other hand, show greater diversity.
- Correlation results are system- and dataset-specific (details provided in supplementary material). We observe the highest correlation for TGen on Bagel (Figures 1 and 2) and LOLS on SFRest, whereas RNNLG often shows low correlation between metrics and human ratings. This lets us conclude that WBMs and GBMs are sensitive to different systems and datasets.
- The highest positive correlation is observed between the number of words (wps) and *informative-*

[8] As an alternative to using the median human judgment for each item, a more effective way to use all the human judgments could be to use Hovy et al. (2013)'s MACE tool for inferring the reliability of judges.

		Bagel		SFHotel		SFRest	
		TGen	LOLS	RNNLG	LOLS	RNNLG	LOLS
Best WBM	inform.	0.30* (BLEU-1)	0.20* (ROUGE)	0.09 (BLEU-1)	0.14* (LEPOR)	0.13* (SIM)	0.28* (LEPOR)
	natural.	-0.19* (TER)	-0.19* (TER)	0.10* (METEOR)	-0.20* (TER)	0.17* (ROUGE)	0.19* (METEOR)
	quality	-0.16* (TER)	0.16* (METEOR)	0.10* (METEOR)	-0.12* (TER)	0.09* (METEOR)	0.18* (LEPOR)
Best GBM	inform.	0.33* (wps)	0.16* (ppw)	-0.09 (ppw)	0.13* (cpw)	0.11* (len)	0.21* (len)
	natural.	-0.25* (len)	-0.28* (wps)	-0.17* (len)	-0.18* (sps)	-0.19* (wps)	-0.21* (sps)
	quality	-0.19* (cpw)	0.31* (prs)	-0.16* (ppw)	-0.17* (spw)	0.11* (prs)	-0.16* (sps)

Table 3: Highest absolute Spearman correlation between metrics and human ratings, with "*" denoting $p < 0.05$ (metric with the highest absolute value of ρ given in brackets).

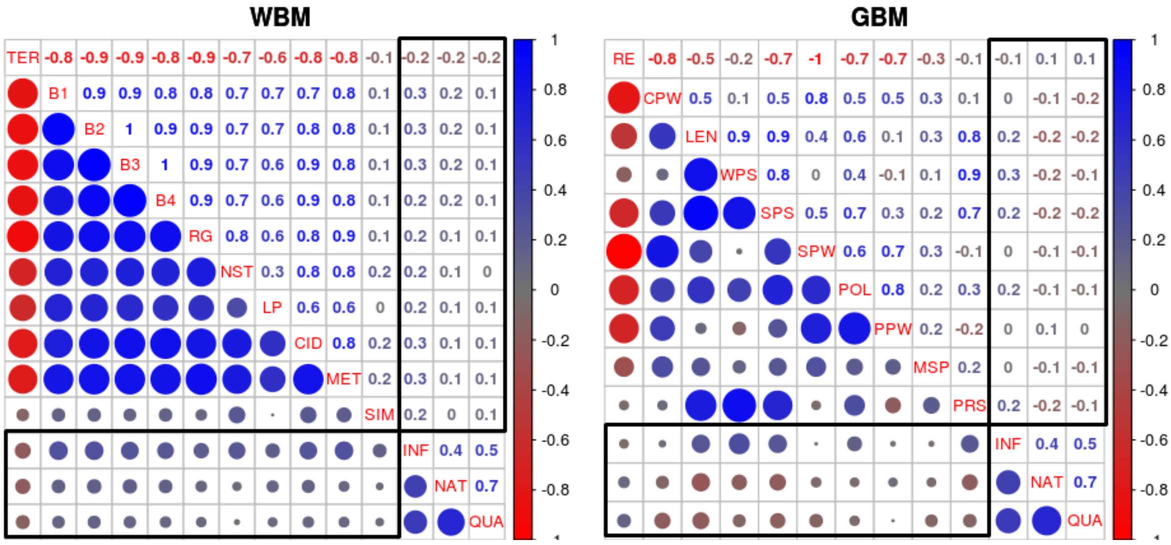


Figure 1: Spearman correlation results for TGen on Bagel. Bordered area shows correlations between human ratings and automatic metrics, the rest shows correlations among the metrics. Blue colour of circles indicates positive correlation, while red indicates negative correlation. The size of circles denotes the correlation strength.

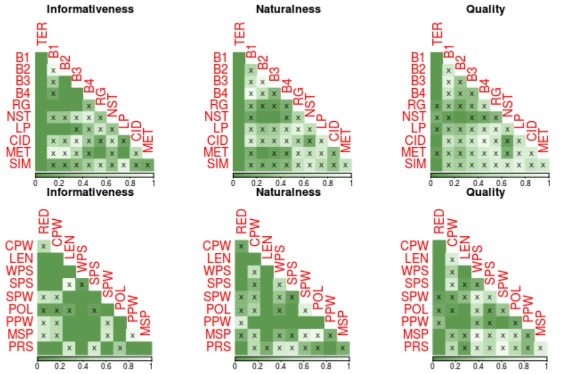


Figure 2: Williams test results: X represents a *non*-significant difference between correlations ($p < 0.05$; top: WBMs, bottom: GBMs).

ness for the TGen system on Bagel ($\rho = 0.33$, $p < 0.01$, see Figure 1). However, the wps metric (amongst most others) is not robust across systems and datasets: Its correlation on other datasets is very weak, ($\rho \leq .18$) and its correlation with informativeness ratings of LOLS outputs is insignificant.

- As a sanity check, we also measure a random score $[0.0, 1.0]$ which proves to have a close-to-zero correlation with human ratings (highest $\rho = 0.09$).

7.2 Accuracy of Relative Rankings

We now evaluate a more coarse measure, namely the metrics' ability to predict relative human ratings. That is, we compute the score of each metric for two system output sentences corresponding to the same MR. The prediction of a metric is correct if it orders the sentences in the same way as median human ratings (note that ties are allowed). Following previous work (Vedantam et al., 2015; Kilickaya et al., 2017), we mainly concentrate on WBMs. Results summarised in Table 4 show that most metrics' performance is not significantly different from that of a random score (Wilcoxon signed rank test). While the random score fluctuates between 25.4–44.5% prediction accuracy, the metrics achieve an accuracy of between 30.6–49.8%. Again, the performance of the metrics is dataset-specific: Metrics perform best on BAGEL data; for SFHOTEL, metrics show mixed performance while for SFREST, metrics perform worst.

		informat.	**naturalness**	**quality**
BAGEL	raw data	TER, BLEU1-4, ROUGE, NIST, LEPOR, CIDEr, METEOR, SIM	TER, BLEU1-4, ROUGE, NIST, LEPOR, CIDEr, METEOR, SIM	TER, BLEU1-4, ROUGE, NIST, LEPOR, CIDEr, METEOR, SIM
SFHOTEL	raw data	TER, BLEU1-4, ROUGE, LEPOR, CIDEr, METEOR, SIM	METEOR	N/A
SFREST	raw data	SIM	LEPOR	N/A
	quant. data	TER, BLEU1-4, ROUGE, NIST, LEPOR, CIDEr, METEOR SIM	N/A	N/A

Table 4: Metrics predicting relative human rating with significantly higher accuracy than a random baseline.

Discussion: Our data differs from the one used in previous work (Vedantam et al., 2015; Kilickaya et al., 2017), which uses explicit relative rankings ("*Which output do you prefer?*"), whereas we compare two Likert-scale ratings. As such, we have 3 possible outcomes (allowing ties). This way, we can account for equally valid system outputs, which is one of the main drawbacks of forced-choice approaches (Hodosh and Hockenmaier, 2016). Our results are akin to previous work: Kilickaya et al. (2017) report results between 60-74% accuracy for binary classification on machine-machine data, which is comparable to our results for 3-way classification.

Still, we observe a mismatch between the ordinal human ratings and the continuous metrics. For example, humans might rate system A and system B both as a 6, whereas BLEU, for example, might assign 0.98 and 1.0 respectively, meaning that BLEU will declare system B as the winner. In order to account for this mismatch, we quantise our metric data to the same scale as the median scores from our human ratings.[9] Applied to SFREST, where we previously got our worst results, we can see an improvement for predicting *informativeness*, where all WBMs now perform significantly better than the random baseline (see Table 4). In the future, we will investigate related discriminative approaches, e.g. (Hodosh and Hockenmaier, 2016; Kannan and Vinyals, 2017), where the task is simplified to distinguishing correct from incorrect output.

8 Error Analysis

In this section, we attempt to uncover why automatic metrics perform so poorly.

8.1 Scales

We first explore the hypothesis that metrics are good in distinguishing extreme cases, i.e. system outputs which are rated as clearly good or bad by the human judges, but do not perform well for utterances rated in the middle of the Likert scale, as suggested by Kilickaya et al. (2017). We 'bin' our data into three groups: *bad*, which comprises low ratings ($\leq$2); *good*, comprising high ratings ($\geq$5); and finally a group comprising *average* ratings.

We find that utterances with low human ratings of *informativeness* and *naturalness* correlate significantly better ($p < 0.05$) with automatic metrics than those with average and good human ratings. For example, as shown in Figure 3, the correlation between WBMs and human ratings for utterances with low *informativeness* scores ranges between $0.3 \leq \rho \leq 0.5$ (moderate correlation), while the highest correlation for utterances of average and high informativeness barely reaches $\rho \leq 0.2$ (very weak correlation). The same pattern can be observed for correlations with *quality* and *naturalness* ratings.

This discrepancy in correlation results between low and other user ratings, together with the fact that the majority of system outputs are rated "good" for informativeness (79%), naturalness (64%) and quality (58%), whereas low ratings do not exceed 7% in total, could explain why the overall correlations are low (Section 7) despite the observed trends in relationship between average system-level performance scores (Section 6). It also explains why the RNNLG system, which contains very few instances of low user ratings, shows poor correlation between human ratings and automatic metrics.

[9] Note that this mismatch can also be accounted for by continuous rating scales, as suggested by Belz and Kow (2011).

No.	system	MR	system output	human reference	WOMs	SIM	humans
1	LOLS	inform(name = the donatello, hasinternet = yes)	well there is a hotel with internet access called the donatello	the donatello has internet	1.4	5	6
2	LOLS	inform_nomatch(area = embarcadero, kidsallowed= yes, pricerange = expensive)	i but i but i but i but i but i but i but i but i	unfortunately i could not find any expensive restaurants in embarcadero that allow kids.	1.1	1	1
3	TGen	inform(name = X, area = riverside, eattype = restaurant, food = fastfood, pricerange = cheap)	x is a restaurant on the riverside called located at the riverside and at is	x is a cheap fastfood restaurant located near the riverside	2.4	4	1
4	RNNLG	inform_nomatch(kidsallowed = yes, food = moroccan)	i am sorry, i did not find any restaurants that allows kids and serve moroccan.	sorry, there are no restaurants allowing kids and serving moroccan food	1.85	4	5

Table 5: Example pairs of MRs and system outputs from our data, contrasting the average of word-overlap metrics (normalised in the 1-6 range) and semantic similarity (SIM) with human ratings (median of all measures).

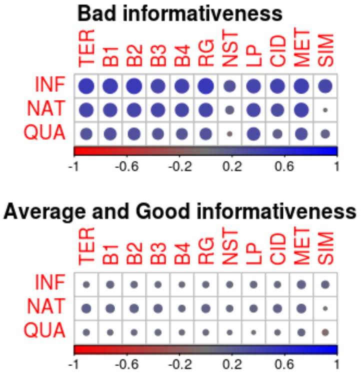


Figure 3: Correlation between automatic metrics (WBMs) and human ratings for utterances of bad informativeness (top), and average and good informativeness (bottom).

8.2 Impact of Target Data

Characteristics of Data: In Section 7.1, we observed that datasets have a significant impact on how well automatic metrics reflect human ratings. A closer inspection shows that BAGEL data differs significantly from SFREST and SFHOTEL, both in terms of grammatical and MR properties. BAGEL has significantly shorter references both in terms of number of characters and words compared to the other two datasets. Although being shorter, the words in BAGEL references are significantly more often polysyllabic. Furthermore, BAGEL only consists of utterances generated from *inform* MRs, while SFREST and SFHOTEL also have less complex MR types, such as *confirm*, *goodbye*, etc. Utterances produced from *inform* MRs are significantly longer and have a significantly higher correlation with human ratings of *informativeness* and *naturalness* than *non-inform* utterance types. In other words, BAGEL is the most complex dataset to generate from. Even though it is more complex, metrics perform most reliably on BAGEL here (note that the correlation is still only weak). One possible explanation is that BAGEL only contains two human references per MR, whereas SFHOTEL and SFREST both contain 5.35 references per MR on average. Having more references means that WBMs naturally will return higher scores (‘anything goes’). This problem could possibly be solved by weighting multiple references according to their quality, as suggested by (Galley et al., 2015), or following a reference-less approach (Specia et al., 2010).

Quality of Data: Our corpora contain crowdsourced human references that have grammatical errors, e.g. “*Fifth Floor does not allow childs*” (SFREST reference). Corpus-based methods may pick up these errors, and word-based metrics will rate these system utterances as correct, whereas we can expect human judges to be sensitive to ungrammatical utterances. Note that the parsing score (while being a crude approximation of grammaticality) achieves one of our highest correlation results against human ratings, with $|\rho| = .31$. Grammatical errors raise questions about the quality of the training data, especially when being crowdsourced. For example, Belz and Reiter (2006) find that human experts assign low rankings to their original corpus text. Again, weighting (Galley et al., 2015) or reference-less approaches (Specia et al., 2010) might remedy this issue.

8.3 Example-based Analysis

As shown in previous sections, word-based metrics moderately agree with humans on bad quality output, but cannot distinguish output of good or medium quality. Table 5 provides examples from

Study	Dimension of human ratings		Domain
	Sentence Planning	Surface Realisation	
this paper	weak positive ($\rho = 0.33$, WPS)	weak negative ($\rho = 0. - 31$, parser)	NLG, restaurant/hotel search
(Reiter and Belz, 2009)	none	strong positive (Pearson's $r = 0.96$, NIST)	NLG, weather forecast
(Stent et al., 2005)	weak positive ($\rho = 0.47$, LSA)	negative ($\rho = -0.56$, NIST)	paraphrasing of news
(Liu et al., 2016)	weak positive ($\rho = 0.35$, BLEU-4)	N/A	dialogue/Twitter pairs
(Elliott and Keller, 2014)	positive ($\rho = 0.53$, METEOR)	N/A	image caption
(Kilickaya et al., 2017)	positive ($\rho = 0.64$, SPICE)	N/A	image caption
(Cahill, 2009)	N/A	negative ($\rho = -0.64$, ROUGE)	NLG, German news texts
(Espinosa et al., 2010)	weak positive ($\rho = 0.43$, TER)	positive ($\rho = 0.62$, BLEU-4)	NLG, news texts

Table 6: Best correlation results achieved by our and previous work. Dimensions targeted towards Sentence Planning include 'accuracy', 'adequacy', 'correctness', 'informativeness'. Dimensions for Surface Realisation include 'clarity', 'fluency', 'naturalness'.

our three systems.[10] Again, we observe different behaviour between WOMs and SIM scores. In Example 1, LOLS generates a grammatically correct English sentence, which represents the meaning of the MR well, and, as a result, this utterance received high human ratings (median = 6) for *informativeness, naturalness* and *quality*. However, WOMs rate this utterance low, i.e. scores of BLEU1-4, NIST, LEPOR, CIDEr, ROUGE and METEOR normalised into the 1-6 range all stay below 1.5. This is because the system-generated utterance has low overlap with the human/corpus references. Note that the SIM score is high (5), as it ignores human references and computes distributional semantic similarity between the MR and the system output. Examples 2 and 3 show outputs which receive low scores from both automatic metrics and humans. WOMs score these system outputs low due to little or no overlap with human references, whereas humans are sensitive to ungrammatical output and missing information (the former is partially captured by GBMs). Examples 2 and 3 also illustrate inconsistencies in human ratings since system output 2 is clearly worse than output 3 and both are rated by human with a median score of 1. Example 4 shows an output of the RNNLG system which is semantically very similar to the reference (SIM=4) and rated high by humans, but WOMs fail to capture this similarity. GBMs show more accurate results for this utterance, with mean of readability scores 4 and parsing score 3.5.

9 Related Work

Table 6 summarises results published by previous studies in related fields which investigate the relation between human scores and automatic metrics. These studies mainly considered WBMs, while we are the first study to consider GBMs. Some studies ask users to provide separate ratings for surface realisation (e.g. asking about 'clarity' or 'fluency'), whereas other studies focus only on sentence planning (e.g. 'accuracy', 'adequacy', or 'correctness'). In general, correlations reported by previous work range from weak to strong. The results confirm that metrics can be reliable indicators at system-level (Reiter and Belz, 2009), while they perform less reliably at sentence-level (Stent et al., 2005). Also, the results show that the metrics capture realization better than sentence planning. There is a general trend showing that best-performing metrics tend to be the more complex ones, combining word-overlap, semantic similarity and term frequency weighting. Note, however, that the majority of previous works do not report whether any of the metric correlations are significantly different from each other.

10 Conclusions

This paper shows that state-of-the-art automatic evaluation metrics for NLG systems do not sufficiently reflect human ratings, which stresses the need for human evaluations. This result is opposed to the current trend of relying on automatic evaluation identified in (Gkatzia and Mahamood, 2015).

A detailed error analysis suggests that automatic metrics are particularly weak in distinguishing outputs of medium and good quality, which can be partially attributed to the fact that human judgements and metrics are given on different scales. We also show that metric performance is data- and system-specific.

Nevertheless, our results also suggest that automatic metrics can be useful for error analysis by helping to find cases where the system is performing poorly. In addition, we find reliable results on

[10] Please note that WBMs tend to match against the reference that is closest to the generated output. Therefore, we only include the closest match in Table 5 for simplicity.

system-level, which suggests that metrics can be useful for system development.

11 Future Directions

Word-based metrics make two strong assumptions: They treat human-generated references as a gold standard, which is *correct* and *complete*. We argue that these assumptions are invalid for corpus-based NLG, especially when using crowdsourced datasets. Grammar-based metrics, on the other hand, do not rely on human-generated references and are not influenced by their quality. However, these metrics can be easily manipulated with grammatically correct and easily readable output that is unrelated to the input. We have experimented with combining WBMs and GBMs using ensemble-based learning. However, while our model achieved high correlation with humans within a single domain, its cross-domain performance is insufficient.

Our paper clearly demonstrates the need for more advanced metrics, as used in related fields, including: assessing output quality within the dialogue context, e.g. (Dušek and Jurčíček, 2016); extrinsic evaluation metrics, such as NLG's contribution to task success, e.g. (Rieser et al., 2014; Gkatzia et al., 2016; Hastie et al., 2016); building discriminative models, e.g. (Hodosh and Hockenmaier, 2016), (Kannan and Vinyals, 2017); or reference-less quality prediction as used in MT, e.g. (Specia et al., 2010). We see our paper as a first step towards reference-less evaluation for NLG by introducing grammar-based metrics. In current work (Dušek et al., 2017), we investigate a reference-less quality estimation approach based on recurrent neural networks, which predicts a quality score for a NLG system output by comparing it to the source meaning representation only.

Finally, note that the datasets considered in this study are fairly small (between 404 and 2.3k human references per domain). To remedy this, systems train on de-lexicalised versions (Wen et al., 2015), which bears the danger of ungrammatical lexicalisation (Sharma et al., 2016) and a possible overlap between testing and training set (Lampouras and Vlachos, 2016). There are ongoing efforts to release larger and more diverse data sets, e.g. (Novikova et al., 2016, 2017).

Acknowledgements

This research received funding from the EPSRC projects DILiGENt (EP/M005429/1) and MaDrIgAL (EP/N017536/1). The Titan Xp used for this research was donated by the NVIDIA Corporation.

References

Anja Belz and Eric Kow. 2011. Discrete vs. continuous rating scales for language evaluation in NLP. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers – Volume 2*. Association for Computational Linguistics, Portland, OR, USA, pages 230–235. http://aclweb.org/anthology/P11-2040.

Anja Belz and Ehud Reiter. 2006. Comparing automatic and human evaluation of NLG systems. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*. Trento, Italy, pages 313–320. http://aclweb.org/anthology/E06-1040.

Aoife Cahill. 2009. Correlating human and automatic evaluation of a German surface realiser. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Association for Computational Linguistics, Suntec, Singapore, pages 97–100. https://aclweb.org/anthology/P09-2025.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*. Trento, Italy, pages 249–256. http://aclweb.org/anthology/E06-1032.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pages 138–145. http://dl.acm.org/citation.cfm?id=1289189.1289273.

Ondrej Dušek, Jekaterina Novikova, and Verena Rieser. 2017. Referenceless quality estimation for natural language generation. In *Proceedings of the 1st Workshop on Learning to Generate Natural Language*.

Ondřej Dušek and Filip Jurčíček. 2015. Training a natural language generator from unaligned data. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China, pages 451–461. http://aclweb.org/anthology/P15-1044.

Ondřej Dušek and Filip Jurčíček. 2016. A context-aware natural language generator for dialogue systems. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Los Angeles, CA, USA, pages 185–190. arXiv:1608.07076. http://aclweb.org/anthology/W16-3622.

Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Berlin, Germany, pages 45–51. arXiv:1606.05491. http://aclweb.org/anthology/P16-2008.

Desmond Elliott and Frank Keller. 2014. Comparing automatic evaluation measures for image description. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, MD, USA, pages 452–457. http://aclweb.org/anthology/P14-2074.

Dominic Espinosa, Rajakrishnan Rajkumar, Michael White, and Shoshana Berleant. 2010. Further meta-evaluation of broad-coverage surface realization. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 564–574. http://aclweb.org/anthology/D10-1055.

Rudolf Franz Flesch. 1979. *How to write plain English: A book for lawyers and consumers*. Harper-Collins.

Thomas Francois and Delphine Bernhard, editors. 2014. *Recent Advances in Automatic Readability Assessment and Text Simplification*, volume 165:2 of *International Journal of Applied Linguistics*. John Benjamins. http://doi.org/10.1075/itl.165.2.

Michel Galley, Chris Brockett, Alessandro Sordoni, Yangfeng Ji, Michael Auli, Chris Quirk, Margaret Mitchell, Jianfeng Gao, and Bill Dolan. 2015. deltaBLEU: A discriminative metric for generation tasks with intrinsically diverse targets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, pages 445–450. http://aclweb.org/anthology/P15-2073.

Jesús Giménez and Lluís Màrquez. 2008. A smorgasbord of features for automatic MT evaluation. In *Proceedings of the Third Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Columbus, OH, USA, pages 195–198. http://aclweb.org/anthology/W08-0332.

Dimitra Gkatzia, Oliver Lemon, and Verena Rieser. 2016. Natural language generation enhances human decision-making with uncertain information. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Berlin, Germany, pages 264–268. arXiv:1606.03254. http://aclweb.org/anthology/P16-2043.

Dimitra Gkatzia and Saad Mahamood. 2015. A snapshot of NLG evaluation practices 2005–2014. In *Proceedings of the 15th European Workshop on Natural Language Generation (ENLG)*. Association for Computational Linguistics, Brighton, UK, pages 57–60. https://doi.org/10.18653/v1/W15-4708.

Aaron L. F. Han, Derek F. Wong, and Lidia S. Chao. 2012. LEPOR: A robust evaluation metric for machine translation with augmented factors. In *Proceedings of COLING 2012: Posters*. The COLING 2012 Organizing Committee, Mumbai, India, pages 441–450. http://aclweb.org/anthology/C12-2044.

Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. UMBC_EBIQUITY-CORE: Semantic textual similarity systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (*SEM)*. Atlanta, Georgia, volume 1, pages 44–52. http://aclweb.org/anthology/S13-1005.

Helen Hastie, Heriberto Cuayahuitl, Nina Dethlefs, Simon Keizer, and Xingkun Liu. 2016. Why bother? Is evaluation of NLG in an end-to-end Spoken Dialogue System worth it? In *Proceedings of the International Workshop on Spoken Dialogue Systems (IWSDS)*. Saariselkä, Finland.

Micah Hodosh and Julia Hockenmaier. 2016. Focused evaluation for image description with binary forced-choice tasks. In *Proceedings of the 5th Workshop on Vision and Language*. Berlin, Germany, pages 19–28. http://aclweb.org/anthology/W16-3203.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard H. Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of NAACL-HLT*. Atlanta, GA, USA, pages 1120–1130. http://aclweb.org/anthology/N13-1132.

Min-Yen Kan, Kathleen R. McKeown, and Judith L. Klavans. 2001. Applying natural language generation to indicative summarization. In *Proceedings of the 8th European Workshop on Natural Language Generation*. Association for Computational Linguistics, Toulouse, France, pages 1–9. https://doi.org/10.3115/1117840.1117853.

Anjuli Kannan and Oriol Vinyals. 2017. Adversarial evaluation of dialogue models. *CoRR* abs/1701.08198. https://arxiv.org/abs/1701.08198.

Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis, and Erkut Erdem. 2017. Re-evaluating automatic metrics for image captioning. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Valencia, Spain. arXiv:1612.07600. https://arxiv.org/abs/1612.07600.

Gerasimos Lampouras and Andreas Vlachos. 2016. Imitation learning for language generation from unaligned data. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, pages 1101–1112. http://aclweb.org/anthology/C16-1105.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics* 33(1):159–174. https://doi.org/10.2307/2529310.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Prague, Czech Republic, pages 228–231. http://aclweb.org/anthology/W07-0734.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*. Barcelona, Spain, pages 74–81. http://aclweb.org/anthology/W04-1013.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, TX, USA, pages 2122–2132. arXiv:1603.08023. http://aclweb.org/anthology/D16-1230.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 1552–1561. http://aclweb.org/anthology/P10-1157.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? Selective generation using LSTMs with coarse-to-fine alignment. In *Proceedings of NAACL-HLT 2016*. San Diego, CA, USA. arXiv:1509.00838. http://aclweb.org/anthology/N16-1086.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2016. There's no comparison: Reference-less evaluation metrics in grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, TX, USA, pages 2109–2115. arXiv:1610.02124. http://aclweb.org/anthology/D16-1228.

Jekaterina Novikova, Ondrej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Saarbrücken, Germany. ArXiv:1706.09254. https://arxiv.org/abs/1706.09254.

Jekaterina Novikova, Oliver Lemon, and Verena Rieser. 2016. Crowd-sourcing NLG data: Pictures elicit better data. In *Proceedings of the 9th International Natural Language Generation Conference*. Edinburgh, UK, pages 265–273. arXiv:1608.00339. http://aclweb.org/anthology/W16-2302.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, PA, USA, pages 311–318. http://aclweb.org/anthology/P02-1040.

Ehud Reiter and Anja Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics* 35(4):529–558. https://doi.org/10.1162/coli.2009.35.4.35405.

Verena Rieser, Oliver Lemon, and Simon Keizer. 2014. Natural language generation as incremental planning under uncertainty: Adaptive information presentation for statistical dialogue systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22(5):979–993. https://doi.org/10.1109/TASL.2014.2315271.

Shikhar Sharma, Jing He, Kaheer Suleman, Hannes Schulz, and Philip Bachman. 2016. Natural language generation in dialogue using lexicalized and delexicalized data. *CoRR* abs/1606.03632. http://arxiv.org/abs/1606.03632.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation of the Americas*. Cambridge, MA, USA, pages 223–231. http://mt-archive.info/AMTA-2006-Snover.pdf.

Lucia Specia, Dhwaj Raj, and Marco Turchi. 2010. Machine translation evaluation versus quality estimation. *Machine translation* 24(1):39–50. https://doi.org/10.1007/s10590-010-9077-2.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *Computational Linguistics and Intelligent Text Processing: 6th International Conference, CICLing 2005, Mexico City, Mexico, February 13-19, 2005. Proceedings*. Springer, Berlin/Heidelberg, pages 341–351. https://doi.org/10.1007/978-3-540-30586-6_38.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *Proceedings of the 2015 IEEE Conference on*

Computer Vision and Pattern Recognition (CVPR). Boston, MA, USA, pages 4566–4575. https://doi.org/10.1109/CVPR.2015.7299087.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina Maria Rojas-Barahona, Pei-hao Su, David Vandyke, and Steve J. Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, CA, USA, pages 120–129. arXiv:1603.01232. http://aclweb.org/anthology/N16-1015.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 1711–1721. http://aclweb.org/anthology/D15-1199.

Evan James Williams. 1959. *Regression analysis*. John Wiley & Sons, New York, NY, USA.

Challenges in Data-to-Document Generation

Sam Wiseman and **Stuart M. Shieber** and **Alexander M. Rush**
School of Engineering and Applied Sciences
Harvard University
Cambridge, MA, USA
{swiseman,shieber,srush}@seas.harvard.edu

Abstract

Recent neural models have shown significant progress on the problem of generating short descriptive texts conditioned on a small number of database records. In this work, we suggest a slightly more difficult data-to-text generation task, and investigate how effective current approaches are on this task. In particular, we introduce a new, large-scale corpus of data records paired with descriptive documents, propose a series of extractive evaluation methods for analyzing performance, and obtain baseline results using current neural generation methods. Experiments show that these models produce fluent text, but fail to convincingly approximate human-generated documents. Moreover, even templated baselines exceed the performance of these neural models on some metrics, though copy- and reconstruction-based extensions lead to noticeable improvements.

1 Introduction

Over the past several years, neural text generation systems have shown impressive performance on tasks such as machine translation and summarization. As neural systems begin to move toward generating longer outputs in response to longer and more complicated inputs, however, the generated texts begin to display reference errors, intersentence incoherence, and a lack of fidelity to the source material. The goal of this paper is to suggest a particular, long-form generation task in which these challenges may be fruitfully explored, to provide a publically available dataset for this task, to suggest some automatic evaluation metrics, and finally to establish how current, neural text generation methods perform on this task.

A classic problem in natural-language generation (NLG) (Kukich, 1983; McKeown, 1992; Reiter and Dale, 1997) involves taking structured data, such as a table, as input, and producing text that adequately and fluently describes this data as output. Unlike machine translation, which aims for a complete transduction of the sentence to be translated, this form of NLG is typically taken to require addressing (at least) two separate challenges: *what to say*, the selection of an appropriate subset of the input data to discuss, and *how to say it*, the surface realization of a generation (Reiter and Dale, 1997; Jurafsky and Martin, 2014). Traditionally, these two challenges have been modularized and handled separately by generation systems. However, neural generation systems, which are typically trained end-to-end as conditional language models (Mikolov et al., 2010; Sutskever et al., 2011, 2014), blur this distinction.

In this context, we believe the problem of generating multi-sentence summaries of tables or database records to be a reasonable next-problem for neural techniques to tackle as they begin to consider more difficult NLG tasks. In particular, we would like this generation task to have the following two properties: (1) it is relatively easy to obtain fairly clean summaries and their corresponding databases for dataset construction, and (2) the summaries should be primarily focused on conveying the information in the database. This latter property ensures that the task is somewhat congenial to a standard encoder-decoder approach, and, more importantly, that it is reasonable to *evaluate* generations in terms of their fidelity to the database.

One task that meets these criteria is that of generating summaries of sports games from associated box-score data, and there is indeed a long history of NLG work that generates sports game

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2253–2263
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

summaries (Robin, 1994; Tanaka-Ishii et al., 1998; Barzilay and Lapata, 2005). To this end, we make the following contributions:

- We introduce a new large-scale corpus consisting of textual descriptions of basketball games paired with extensive statistical tables. This dataset is sufficiently large that fully data-driven approaches might be sufficient.
- We introduce a series of extractive evaluation models to automatically evaluate output generation performance, exploiting the fact that post-hoc information extraction is significantly easier than generation itself.
- We apply a series of state-of-the-art neural methods, as well as a simple templated generation system, to our data-to-document generation task in order to establish baselines and study their generations.

Our experiments indicate that neural systems are quite good at producing fluent outputs and generally score well on standard word-match metrics, but perform quite poorly at content selection and at capturing long-term structure. While the use of copy-based models and additional reconstruction terms in the training loss can lead to improvements in BLEU and in our proposed extractive evaluations, current models are still quite far from producing human-level output, and are significantly worse than templated systems in terms of content selection and realization. Overall, we believe this problem of data-to-document generation highlights important remaining challenges in neural generation systems, and the use of extractive evaluation reveals significant issues hidden by standard automatic metrics.

2 Data-to-Text Datasets

We consider the problem of generating descriptive text from database records. Following the notation in Liang et al. (2009), let $\boldsymbol{s} = \{r_j\}_{j=1}^{J}$ be a set of records, where for each $r \in \boldsymbol{s}$ we define $r.t \in \mathcal{T}$ to be the *type* of r, and we assume each r to be a binarized relation, where $r.e$ and $r.m$ are a record's entity and value, respectively. For example, a database recording statistics for a basketball game might have a record r such that $r.t$ = POINTS, $r.e$ = RUSSELL WESTBROOK, and $r.m$ = 50. In this case, $r.e$ gives the player in question, and $r.m$ gives the number of points the player scored. From these records, we are interested in generating descriptive text, $\hat{y}_{1:T} = \hat{y}_1, \ldots, \hat{y}_T$ of T words such that $\hat{y}_{1:T}$ is an adequate and fluent summary of $\boldsymbol{s}$. A dataset for training data-to-document systems typically consists of $(\boldsymbol{s}, y_{1:T})$ pairs, where $y_{1:T}$ is a document consisting of a gold (i.e., human generated) summary for database $\boldsymbol{s}$.

Several benchmark datasets have been used in recent years for the text generation task, the most popular of these being WEATHERGOV (Liang et al., 2009) and ROBOCUP (Chen and Mooney, 2008). Recently, neural generation systems have show strong results on these datasets, with the system of Mei et al. (2016) achieving BLEU scores in the 60s and 70s on WEATHERGOV, and BLEU scores of almost 30 even on the smaller ROBOCUP dataset. These results are quite promising, and suggest that neural models are a good fit for text generation. However, the statistics of these datasets, shown in Table 1, indicate that these datasets use relatively simple language and record structure. Furthermore, there is reason to believe that WEATHERGOV is at least partially machine-generated (Reiter, 2017). More recently, Lebret et al. (2016) introduced the WIKIBIO dataset, which is at least an order of magnitude larger in terms of number of tokens and record types. However, as shown in Table 1, this dataset too only contains short (single-sentence) generations, and relatively few records per generation. As such, we believe that early success on these datasets is not yet sufficient for testing the desired linguistic capabilities of text generation at a document-scale.

With this challenge in mind, we introduce a new dataset for data-to-document text generation, available at `https://github.com/harvardnlp/boxscore-data`. The dataset is intended to be comparable to WEATHERGOV in terms of token count, but to have significantly longer target texts, a larger vocabulary space, and to require more difficult content selection.

The dataset consists of two sources of articles summarizing NBA basketball games, paired with their corresponding box- and line-score tables. The data statistics of these two sources, ROTOWIRE and SBNATION, are also shown in Table 1. The first dataset, ROTOWIRE, uses professionally written, medium length game summaries targeted at fantasy basketball fans. The writing is colloquial, but relatively well structured, and targets an audience primarily interested in game

TEAM	WIN	LOSS	PTS	FG_PCT	RB	AS . . .
Heat	11	12	103	49	47	27
Hawks	7	15	95	43	33	20

PLAYER	AS	RB	PT	FG	FGA	CITY . . .
Tyler Johnson	5	2	27	8	16	Miami
Dwight Howard	4	17	23	9	11	Atlanta
Paul Millsap	2	9	21	8	12	Atlanta
Goran Dragic	4	2	21	8	17	Miami
Wayne Ellington	2	3	19	7	15	Miami
Dennis Schroder	7	4	17	8	15	Atlanta
Rodney McGruder	5	5	11	3	8	Miami
Thabo Sefolosha	5	5	10	5	11	Atlanta
Kyle Korver	5	3	9	3	9	Atlanta
. . .						

The Atlanta Hawks defeated the Miami Heat , 103 - 95 , at Philips Arena on Wednesday . Atlanta was in desperate need of a win and they were able to take care of a shorthanded Miami team here . Defense was key for the Hawks , as they held the Heat to 42 percent shooting and forced them to commit 16 turnovers . Atlanta also dominated in the paint , winning the rebounding battle , 47 - 34 , and outscoring them in the paint 58 - 26.The Hawks shot 49 percent from the field and assisted on 27 of their 43 made baskets . This was a near wire - to - wire win for the Hawks , as Miami held just one lead in the first five minutes . Miami (7 - 15) are as beat - up as anyone right now and it 's taking a toll on the heavily used starters . Hassan Whiteside really struggled in this game , as he amassed eight points , 12 rebounds and one blocks on 4 - of - 12 shooting ...

Figure 1: An example data-record and document pair from the ROTOWIRE dataset. We show a subset of the game's records (there are 628 in total), and a selection from the gold document. The document mentions only a select subset of the records, but may express them in a complicated manner. In addition to capturing the writing style, a generation system should select similar record content, express it clearly, and order it appropriately.

	RC	WG	WB	RW	SBN
Vocab	409	394	400K	11.3K	68.6K
Tokens	11K	0.9M	19M	1.6M	8.8M
Examples	1.9K	22.1K	728K	4.9K	10.9K
Avg Len	5.7	28.7	26.1	337.1	805.4
Rec. Types	4	10	1.7K	39	39
Avg Records	2.2	191	19.7	628	628

Table 1: Vocabulary size, number of total tokens, number of distinct examples, average generation length, total number of record types, and average number of records per example for the ROBOCUP (RC), WEATHERGOV (WG), WIKIBIO (WB), ROTOWIRE (RW), and SBNATION (SBN) datasets.

statistics. The second dataset, SBNATION, uses fan-written summaries targeted at other fans. This dataset is significantly larger, but also much more challenging, as the language is very informal, and often tangential to the statistics themselves. We show some sample text from ROTOWIRE in Figure 1. Our primary focus will be on the ROTOWIRE data.

3 Evaluating Document Generation

We begin by discussing the evaluation of generated documents, since both the task we introduce and the evaluation methods we propose are motivated by some of the shortcomings of current approaches to evaluation. Text generation systems are typically evaluated using a combination of automatic measures, such as BLEU (Papineni et al., 2002), and human evaluation. While BLEU is perhaps a reasonably effective way of evaluating short-form text generation, we found it to be unsatisfactory for document generation. In particular, we note that it primarily rewards fluent text generation, rather than generations that capture the most important information in the database, or that report the information in a particularly coherent way. While human evaluation, on the other hand, is likely ultimately necessary for evaluating generations (Liu et al., 2016; Wu et al., 2016), it is much less convenient than using automatic metrics. Furthermore, we believe that current text generations are sufficiently bad in sufficiently obvious ways that automatic metrics can still be of use in evaluation, and we are not yet at the point of needing to rely solely on human evaluators.

3.1 Extractive Evaluation

To address this evaluation challenge, we begin with the intuition that assessing document quality is easier than document generation. In particular, it is much easier to automatically extract information from documents than to generate documents that accurately convey desired information. As such, simple, high-precision information extraction models can serve as the basis for assessing and better understanding the quality of automatic generations. We emphasize that such an evaluation scheme is most appropriate when evaluating generations (such as basketball game summaries) that are primarily intended to summarize information. While many generation problems do not fall into this category, we believe this to be an interesting category, and one worth focusing on *because* it is amenable to this sort of evaluation.

To see how a simple information extraction system might work, consider the document in Figure 1. We may first extract candidate entity (player, team, and city) and value (number and certain string) pairs $r.e, r.m$ that appear in the text, and then predict the type $r.t$ (or none) of each candidate pair. For example, we might extract the entity-value pair ("Miami Heat", "95") from the first sentence in Figure 1, and then predict that the *type* of this pair is POINTS, giving us an extracted record r such that $(r.e, r.m, r.t) = (\text{MIAMI HEAT}, 95, \text{POINTS})$. Indeed, many relation extraction systems reduce relation extraction to multi-class classification precisely in this way (Zhang, 2004; Zhou et al., 2008; Zeng et al., 2014; dos Santos et al., 2015).

More concretely, given a document $\hat{y}_{1:T}$, we consider all pairs of word-spans in each sentence that represent possible entities e and values m. We then model $p(r.t \mid e, m; \boldsymbol{\theta})$ for each pair, using $r.t = \epsilon$ to indicate unrelated pairs. We use architectures similar to those discussed in Collobert et al. (2011) and dos Santos et al. (2015) to parameterize this probability; full details are given in the Appendix.

Importantly, we note that the $(\boldsymbol{s}, y_{1:T})$ pairs typically used for training data-to-document systems are also sufficient for training the information extraction model presented above, since we can obtain (partial) supervision by simply checking whether a candidate record lexically matches a record in $\boldsymbol{s}$.[1] However, since there may be multiple records $r \in \boldsymbol{s}$ with the same e and m but with different types $r.t$, we will not always be able to determine the type of a given entity-value pair found in the text. We therefore train our classifier to minimize a latent-variable loss: for all document spans e and m, with observed types $t(e,m) = \{r.t : r \in \boldsymbol{s}, r.e = e, r.m = m\}$ (possibly $\{\epsilon\}$), we minimize

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{e,m} \log \sum_{t' \in t(e,m)} p(r.t = t' \mid e, m; \boldsymbol{\theta}).$$

We find that this simple system trained in this way is quite accurate at predicting relations. On the ROTOWIRE data it achieves over 90% accuracy on held-out data, and recalls approximately 60% of the relations licensed by the records.

3.2 Comparing Generations

With a sufficiently precise relation extraction system, we can begin to evaluate how well an automatic generation $\hat{y}_{1:T}$ has captured the information in a set of records $\boldsymbol{s}$. In particular, since the predictions of a precise information extraction system serve to align entity-mention pairs in the text with database records, this alignment can be used both to evaluate a generation's content selection ("what the generation says"), as well as content placement ("how the generation says it").

We consider in particular three induced metrics:

- Content Selection (CS): precision and recall of unique relations r extracted from $\hat{y}_{1:T}$ that are also extracted from $y_{1:T}$. This measures how well the generated document matches the gold document in terms of selecting which records to generate.
- Relation Generation (RG): precision and number of unique relations r extracted from $\hat{y}_{1:T}$ that also appear in $\boldsymbol{s}$. This measures how well the system is able to generate text containing factual (i.e., correct) records.
- Content Ordering (CO): normalized Damerau-Levenshtein Distance (Brill and Moore, 2000)[2] between the sequences of records extracted from $y_{1:T}$ and that extracted from $\hat{y}_{1:T}$. This measures how well the system orders the records it chooses to discuss.

We note that CS primarily targets the "what to say" aspect of evaluation, CO targets the "how to say it" aspect, and RG targets both.

We conclude this section by contrasting the automatic evaluation we have proposed with recently proposed *adversarial evaluation* approaches, which also advocate automatic metrics backed by classification (Bowman et al., 2016; Kannan and Vinyals, 2016; Li et al., 2017). Unlike adversarial evaluation, which uses a black-box classifier to determine the quality of a generation, our metrics are defined with respect to the

[1] Alternative approaches explicitly align the document with the table for this task (Liang et al., 2009).

[2] DLD is a variant of Levenshtein distance that allows transpositions of elements; it is useful in comparing the ordering of sequences that may not be permutations of the same set (which is a requirement for measures like Kendall's Tau).

predictions of an information extraction system. Accordingly, our metrics are quite interpretable, since by construction it is always possible to determine which fact (i.e., entity-value pair) in the generation is determined by the extractor to not match the database or the gold generation.

4 Neural Data-to-Document Models

In this section we briefly describe the neural generation methods we apply to the proposed task. As a base model we utilize the now standard attention-based encoder-decoder model (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015). We also experiment with several recent extensions to this model, including copy-based generation, and training with a source reconstruction term in the loss (in addition to the standard per-target-word loss).

Base Model For our base model, we map each record $r \in \boldsymbol{s}$ into a vector $\tilde{\boldsymbol{r}}$ by first embedding $r.t$ (e.g., POINTS), $r.e$ (e.g., RUSSELL WESTBROOK), and $r.m$ (e.g., 50), and then applying a 1-layer MLP (similar to Yang et al. (2016)).[3] Our source data-records are then represented as $\tilde{\boldsymbol{s}} = \{\tilde{\boldsymbol{r}}_j\}_{j=1}^{J}$. Given $\tilde{\boldsymbol{s}}$, we use an LSTM decoder with attention and input-feeding, in the style of Luong et al. (2015), to compute the probability of each target word, conditioned on the previous words and on $\boldsymbol{s}$. The model is trained end-to-end to minimize the negative log-likelihood of the words in the gold text $y_{1:T}$ given corresponding source material $\boldsymbol{s}$.

Copying There has been a surge of recent work involving augmenting encoder-decoder models to copy words directly from the source material on which they condition (Gu et al., 2016; Gülçehre et al., 2016; Merity et al., 2016; Jia and Liang, 2016; Yang et al., 2016). These models typically introduce an additional binary variable z_t into the per-timestep target word distribution, which indicates whether the target word $\hat{y}_t$ is copied from the source or generated:

$$p(\hat{y}_t \mid \hat{y}_{1:t-1}, \boldsymbol{s}) = \sum_{z \in \{0,1\}} p(\hat{y}_t, z_t = z \mid \hat{y}_{1:t-1}, \boldsymbol{s}).$$

In our case, we assume that target words are copied from the *value* portion of a record r; that is, a copy implies $\hat{y}_t = r.m$ for some r and t.

Joint Copy Model The models of Gu et al. (2016) and Yang et al. (2016) parameterize the *joint* distribution table over $\hat{y}_t$ and z_t directly:

$$p(\hat{y}_t, z_t \mid \hat{y}_{1:t-1}, \boldsymbol{s}) \propto \begin{cases} \mathrm{copy}(\hat{y}_t, \hat{y}_{1:t-1}, \boldsymbol{s}) & z_t = 1, \hat{y}_t \in \boldsymbol{s} \\ 0 & z_t = 1, \hat{y}_t \notin \boldsymbol{s} \\ \mathrm{gen}(\hat{y}_t, \hat{y}_{1:t-1}, \boldsymbol{s}) & z_t = 0, \end{cases}$$

where copy and gen are functions parameterized in terms of the decoder RNN's hidden state that assign scores to words, and where the notation $\hat{y}_t \in \boldsymbol{s}$ indicates that $\hat{y}_t$ is equal to $r.m$ for some $r \in \boldsymbol{s}$.

Conditional Copy Model Gülçehre et al. (2016), on the other hand, decompose the joint probability as:

$$p(\hat{y}_t, z_t \mid \hat{y}_{1:t-1}, \boldsymbol{s}) = \begin{cases} p_{\mathrm{copy}}(\hat{y}_t \mid z_t, \hat{y}_{1:t-1}, \boldsymbol{s})\, p(z_t \mid \hat{y}_{1:t-1}, \boldsymbol{s}) & z_t{=}1 \\ p_{\mathrm{gen}}(\hat{y}_t \mid z_t, \hat{y}_{1:t-1}, \boldsymbol{s})\, p(z_t \mid \hat{y}_{1:t-1}, \boldsymbol{s}) & z_t{=}0, \end{cases}$$

where an MLP is used to model $p(z_t \mid \hat{y}_{1:t-1}, \boldsymbol{s})$.

Models with copy-decoders may be trained to minimize the negative log marginal probability, marginalizing out the latent-variable z_t (Gu et al., 2016; Yang et al., 2016; Merity et al., 2016). However, if it is known which target words y_t are copied, it is possible to train with a loss that does not marginalize out the latent z_t. Gülçehre et al. (2016), for instance, assume that any target word y_t that also appears in the source is copied, and train to minimize the negative joint log-likelihood of the y_t and z_t.

In applying such a loss in our case, we again note that there may be multiple records r such that $r.m$ appears in $\hat{y}_{1:T}$. Accordingly, we slightly modify the p_{copy} portion of the loss of Gülçehre et al. (2016) to sum over all matched records. In particular, we model the probability of relations $r \in \boldsymbol{s}$ such that $r.m = y_t$ and $r.e$ is in the same sentence as $r.m$. Letting $r(y_t) = \{r \in \boldsymbol{s} : r.m = y_t, \mathrm{same{-}sentence}(r.e, r.m)\}$, we have:

$$p_{\mathrm{copy}}(y_t \mid z_t, y_{1:t-1}, \boldsymbol{s}) = \sum_{r \in r(y_t)} p(r \mid z_t, y_{1:t-1}, \boldsymbol{s}).$$

We note here that the key distinction for our purposes between the Joint Copy model and the Conditional Copy model is that the latter *conditions* on whether there is a copy or not, and so in p_{copy} the

[3] We also include an additional feature for whether the player is on the home- or away-team.

source records compete only with each other. In the Joint Copy model, however, the source records also compete with words that cannot be copied. As a result, training the Conditional Copy model with the supervised loss of Gülçehre et al. (2016) can be seen as training with a word-level reconstruction loss, where the decoder is trained to choose the record in $\boldsymbol{s}$ that gives rise to y_t.

Reconstruction Losses Reconstruction-based techniques can also be applied at the document- or sentence-level during training. One simple approach to this problem is to utilize the hidden states of the decoder to try to reconstruct the database. A fully differentiable approach using the decoder hidden states has recently been successfully applied to neural machine translation by Tu et al. (2017). Unlike copying, this method is applied only at training, and attempts to learn decoder hidden states with broader coverage of the input data.

In adopting this reconstruction approach we segment the decoder hidden states $\boldsymbol{h}_t$ into $\lceil \frac{T}{B} \rceil$ contiguous blocks of size at most B. Denoting a single one of these hidden state blocks as $\boldsymbol{b}_i$, we attempt to predict each field value in some record $r \in \boldsymbol{s}$ from $\boldsymbol{b}_i$. We define $p(r.e, r.m \mid \boldsymbol{b}_i)$, the probability of the entity and value in record r given $\boldsymbol{b}_i$, to be $\mathrm{softmax}(f(\boldsymbol{b}_i))$, where f is a parameterized function of $\boldsymbol{b}_i$, which in our experiments utilize a convolutional layer followed by an MLP; full details are given in the Appendix. We further extend this idea and predict K records in $\boldsymbol{s}$ from $\boldsymbol{b}_i$, rather than one. We can train with the following reconstruction loss for a particular $\boldsymbol{b}_i$:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= -\sum_{k=1}^{K} \min_{r \in \boldsymbol{s}} \log p_k(r \mid \boldsymbol{b}_i; \boldsymbol{\theta}) \\ &= -\sum_{k=1}^{K} \min_{r \in \boldsymbol{s}} \sum_{x \in \{e,m,t\}} \log p_k(r.x \mid \boldsymbol{b}_i; \boldsymbol{\theta}),\end{aligned}$$

where p_k is the k'th predicted distribution over records, and where we have modeled each component of r independently. This loss attempts to make the *most* probable record in $\boldsymbol{s}$ given $\boldsymbol{b}_i$ more probable. We found that augmenting the above loss with a term that penalizes the total variation distance (TVD) between the p_k to be helpful.[4] Both $\mathcal{L}(\boldsymbol{\theta})$ and the TVD term are simply added to the standard negative log-likelihood objective at training time.

5 Experimental Methods

In this section we highlight a few important details of our models and methods; full details are in the Appendix. For our ROTOWIRE models, the record encoder produces $\tilde{\boldsymbol{r}}_j$ in $\mathbb{R}^{600}$, and we use a 2-layer LSTM decoder with hidden states of the same size as the $\tilde{\boldsymbol{r}}_j$, and dot-product attention and input-feeding in the style of Luong et al. (2015). Unlike past work, we use two identically structured attention layers, one to compute the standard generation probabilities (gen or p_{gen}), and one to produce the scores used in copy or p_{copy}.

We train the generation models using SGD and truncated BPTT (Elman, 1990; Mikolov et al., 2010), as in language modeling. That is, we split each $y_{1:T}$ into contiguous blocks of length 100, and backprop both the gradients with respect to the current block as well as with respect to the encoder parameters for each block.

Our extractive evaluator consists of an ensemble of 3 single-layer convolutional and 3 single-layer bidirectional LSTM models. The convolutional models concatenate convolutions with kernel widths 2, 3, and 5, and 200 feature maps in the style of (Kim, 2014). Both models are trained with SGD.

Templatized Generator In addition to neural baselines, we also use a problem-specific, template-based generator. The template-based generator first emits a sentence about the teams playing in the game, using a templatized sentence taken from the training set:

> The `<team1>` (`<wins1>`-`<losses1>`) defeated the `<team2>` (`<wins2>`-`<losses2>`) `<pts1>`-`<pts2>`.

Then, 6 player-specific sentences of the following form are emitted (again adapting a simple sentence from the training set):

> `<player>` scored `<pts>` points (`<fgm>`-`<fga>` FG, `<tpm>`-`<tpa>` 3PT, `<ftm>`-`<fta>` FT) to go with `<reb>` rebounds.

[4] Penalizing the TVD between the p_k might be useful if, for instance, K is too large, and only a smaller number of records can be predicted from $\boldsymbol{b}_i$. We also experimented with *encouraging*, rather than penalizing the TVD between the p_k, which might make sense if we were worried about ensuring the p_k captured different records.

The 6 highest-scoring players in the game are used to fill in the above template. Finally, a typical end sentence is emitted:

> The `<team1>`' next game will be at home against the Dallas Mavericks, while the `<team2>` will travel to play the Bulls.

Code implementing all models can be found at `https://github.com/harvardnlp/data2text`. Our encoder-decoder models are based on OpenNMT (Klein et al., 2017).

6 Results

We found that all models performed quite poorly on the SBNATION data, with the best model achieving a validation perplexity of 33.34 and a BLEU score of 1.78. This poor performance is presumably attributable to the noisy quality of the SBNATION data, and the fact that many documents in the dataset focus on information not in the box- and line-scores. Accordingly, we focus on ROTOWIRE in what follows.

The main results for the ROTOWIRE dataset are shown in Table 2, which shows the performance of the models in Section 4 in terms of the metrics defined in Section 3.2, as well as in terms of perplexity and BLEU.

6.1 Discussion

There are several interesting relationships in the development portion of Table 2. First we note that the Template model scores very poorly on BLEU, but does quite well on the extractive metrics, providing an upper-bound for how domain knowledge could help content selection and generation. All the neural models make significant improvements terms of BLEU score, with the conditional copying with beam search performing the best, even though all the neural models achieve roughly the same perplexity.

The extractive metrics provide further insight into the behavior of the models. We first note that on the gold documents $y_{1:T}$, the extractive model reaches 92% precision. Using the Joint Copy model, generation only has a record generation (RG) precision of 47% indicating that relationships are often generated incorrectly. The best Conditional Copy system improves this value to 71%, a significant improvement and potentially the cause of the improved BLEU score, but still far below gold.

> The Utah Jazz (38 - 26) defeated the Houston Rockets (38 - 26) 117 - 91 on Wednesday at Energy Solutions Arena in Salt Lake City . The Jazz got out to a quick start in this one , out - scoring the Rockets 31 - 15 in the first quarter alone . Along with the quick start , the Rockets were the superior shooters in this game , going 54 percent from the field and 43 percent from the three - point line , while the Jazz went 38 percent from the floor and a meager 19 percent from deep . The Rockets were able to out - rebound the Rockets 49 - 49 , giving them just enough of an advantage to secure the victory in front of their home crowd . The Jazz were led by the duo of Derrick Favors and James Harden . Favors went 2 - for - 6 from the field and 0 - for - 1 from the three - point line to score a game - high of 15 points , while also adding four rebounds and four assists ....

Figure 2: Example document generated by the Conditional Copy system with a beam of size 5. Text that accurately reflects a record in the associated box- or line-score is highlighted in blue, and erroneous text is highlighted in red.

Notably, content selection (CS) and content ordering (CO) seem to have no correlation at all with BLEU. There is some improvement with CS for the conditional model or reconstruction loss, but not much change as we move to beam search. CO actually gets worse as beam search is utilized, possibly a side effect of generating more records (RG#). The fact that these scores are much worse than the simple templated model indicates that further research is needed into better copying alone for content selection and better long term content ordering models.

Test results are consistent with development results, indicating that the Conditional Copy model is most effective at BLEU, RG, and CS, and that reconstruction is quite helpful for improving the joint model.

6.2 Human Evaluation

We also undertook two human evaluation studies, using Amazon Mechanical Turk. The first study attempted to determine whether generations considered to be more precise by our metrics were also considered more precise by human raters. To accomplish this, raters were presented with a particular NBA game's box score and line score, as well as with (randomly selected) sentences from summaries generated by our different models for those games. Raters were then asked to count how many facts in each sentence were supported by records in the box or line scores, and how many were contradicted. We randomly selected 20 distinct games to present to raters, and a total of 20 generated sentences per game were evaluated by raters. The left two columns of Table 3 contain the

		Development						
		RG		CS		CO	PPL	BLEU
Beam	Model	P%	#	P%	R%	DLD%		
	Gold	91.77	12.84	100	100	100	1.00	100
	Template	99.35	49.7	18.28	65.52	12.2	N/A	6.87
B=1	Joint Copy	47.55	7.53	20.53	22.49	8.28	7.46	10.41
	Joint Copy + Rec	57.81	8.31	23.65	23.30	9.02	7.25	10.00
	Joint Copy + Rec + TVD	60.69	8.95	23.63	24.10	8.84	7.22	12.78
	Conditional Copy	68.94	9.09	25.15	22.94	9.00	7.44	13.31
B=5	Joint Copy	47.00	10.67	16.52	26.08	7.28	7.46	10.23
	Joint Copy + Rec	62.11	10.90	21.36	26.26	9.07	7.25	10.85
	Joint Copy + Rec + TVD	57.51	11.41	18.28	25.27	8.05	7.22	12.04
	Conditional Copy	71.07	12.61	21.90	27.27	8.70	7.44	14.46
		Test						
	Template	99.30	49.61	18.50	64.70	8.04	N/A	6.78
	Joint Copy + Rec (B=5)	61.23	11.02	21.56	26.45	9.06	7.47	10.88
	Joint Copy + Rec + TVD (B=1)	60.27	9.18	23.11	23.69	8.48	7.42	12.96
	Conditional Copy (B=5)	71.82	12.82	22.17	27.16	8.68	7.67	14.49

Table 2: Performance of induced metrics on gold and system outputs of RotoWire development and test data. Columns indicate Record Generation (RG) precision and count, Content Selection (CS) precision and recall, Count Ordering (CO) in normalized Damerau-Levenshtein distance, perplexity, and BLEU. These first three metrics are described in Section 3.2. Models compare Joint and Conditional Copy also with addition Reconstruction loss and Total Variation Distance extensions (described in Section 4).

average numbers of supporting and contradicting facts per sentence as determined by the raters, for each model. We see that these results are generally in line with the RG and CS metrics, with the Conditional Copy model having the highest number of supporting facts, and the reconstruction terms significantly improving the Joint Copy models.

Using a Tukey HSD post-hoc analysis of an ANOVA with the number of contradicting facts as the dependent variable and the generating model and rater id as independent variables, we found significant ($p < 0.01$) pairwise differences in contradictory facts between the gold generations and all models except "Copy+Rec+TVD," as well as a significant difference between "Copy+Rec+TVD" and "Copy". We similarly found a significant pairwise difference between "Copy+Rec+TVD" and "Copy" for number of supporting facts.

Our second study attempted to determine whether generated summaries differed in terms of how natural their *ordering* of records (as captured, for instance, by the DLD metric) is. To test this, we presented raters with random summaries generated by our models and asked them to rate the naturalness of the ordering of facts in the summaries on a 1-7 Likert scale. 30 random summaries were used in this experiment, each rated 3 times by distinct raters. The average Likert ratings are shown in the rightmost column of Table 3.

	# Supp.	# Cont.	Order Rat.
Gold	2.04	0.70	5.19
Joint Copy	1.65	2.31	3.90
Joint Copy + Rec	2.33	1.83	4.43
Joint Copy + Rec +TVD	2.43	1.16	4.18
Conditional Copy	3.05	1.48	4.03

Table 3: Average rater judgment of number of box score fields supporting (left column) or contradicting (middle column) a generated sentence, and average rater Likert rating for the naturalness of a summary's ordering (right column). All generations use B=1.

While it is encouraging that the gold summaries received a higher average score than the generated summaries (and that the reconstruction term again improved the Joint Copy model), a Tukey HSD analysis similar to the one presented above revealed no significant pairwise differences.

6.3 Qualitative Example

Figure 2 shows a document generated by the Conditional Copy model, using a beam of size 5. This particular generation evidently has several nice properties: it nicely learns the colloquial style of the text, correctly using idioms such as "19 percent from deep." It is also partially accurate in its use of the records; we highlight in blue when it generates text that is licensed by a record in the associated box- and line-scores.

At the same time, the generation also contains major logical errors. First, there are basic copying mistakes, such as flipping the teams' win/loss records. The system also makes obvious semantic errors; for instance, it generates the phrase "the Rockets were able to out-rebound the Rockets." Finally, we see the model hallucinates factual statements, such as "in front of their home crowd," which is presumably likely according to the language model, but ultimately incorrect (and not supported by anything in the box- or line-scores). In practice, our proposed extractive evaluation will pick up on many errors in this passage. For instance, "four assists" is an RG error, repeating the Rockets' rebounds could manifest in a lower CO score, and incorrectly indicating the win/loss records is a CS error.

7 Related Work

In this section we note additional related work not noted throughout. Natural language generation has been studied for decades (Kukich, 1983; McKeown, 1992; Reiter and Dale, 1997), and generating summaries of sports games has been a topic of interest for almost as long (Robin, 1994; Tanaka-Ishii et al., 1998; Barzilay and Lapata, 2005).

Historically, research has focused on both content selection ("what to say") (Kukich, 1983; McKeown, 1992; Reiter and Dale, 1997; Duboue and McKeown, 2003; Barzilay and Lapata, 2005), and surface realization ("how to say it") (Goldberg et al., 1994; Reiter et al., 2005) with earlier work using (hand-built) grammars, and later work using SMT-like approaches (Wong and Mooney, 2007) or generating from PCFGs (Belz, 2008) or other formalisms (Soricut and Marcu, 2006; White et al., 2007). In the late 2000s and early 2010s, a number of systems were proposed that did both (Liang et al., 2009; Angeli et al., 2010; Kim and Mooney, 2010; Lu and Ng, 2011; Konstas and Lapata, 2013).

Within the world of neural text generation, some recent work has focused on conditioning language models on tables (Yang et al., 2016), and generating short biographies from Wikipedia Tables (Lebret et al., 2016; Chisholm et al., 2017). Mei et al. (2016) use a neural encoder-decoder approach on standard record-based generation datasets, obtaining impressive results, and motivating the need for more challenging NLG problems.

8 Conclusion and Future Work

This work explores the challenges facing neural data-to-document generation by introducing a new dataset, and proposing various metrics for automatically evaluating content selection, generation, and ordering. We see that recent ideas in copying and reconstruction lead to improvements on this task, but that there is a significant gap even between these neural models and templated systems. We hope to motivate researchers to focus further on generation problems that are relevant both to content selection and surface realization, but may not be reflected clearly in the model's perplexity.

Future work on this task might include approaches that process or attend to the source records in a more sophisticated way, generation models that attempt to incorporate semantic or reference-related constraints, and approaches to conditioning on facts or records that are not as explicit in the box- and line-scores.

Acknowledgments

We gratefully acknowledge the support of a Google Research Award.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Regina Barzilay and Mirella Lapata. 2005. Collective content selection for concept-to-text generation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 331–338. Association for Computational Linguistics.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(04):431–455.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *CoNLL*, pages 10–21.

Eric Brill and Robert C Moore. 2000. An improved error model for noisy channel spelling correction. In

Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, pages 286–293. Association for Computational Linguistics.

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135. ACM.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. *CoRR*, abs/1702.06235.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

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

Pablo A Duboue and Kathleen R McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *EMNLP*, pages 121–128. Association for Computational Linguistics.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Eli Goldberg, Norbert Driedger, and Richard I Kittredge. 1994. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*.

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *ACL*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *ACL Volume 1: Long Papers*.

Dan Jurafsky and James H Martin. 2014. *Speech and language processing*, volume 3. Pearson London.

Anjuli Kannan and Oriol Vinyals. 2016. Adversarial evaluation of dialogue models. In *NIPS 2016 Workshop on Adversarial Training*.

Joohyun Kim and Raymond J Mooney. 2010. Generative alignment and semantic parsing for learning from ambiguous supervision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 543–551. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *CoRR*, abs/1701.02810.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *J. Artif. Intell. Res.(JAIR)*, 48:305–346.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *ACL*, pages 145–150.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *EMNLP*, pages 1203–1213.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *CoRR*, abs/1701.06547.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACL*, pages 91–99. Association for Computational Linguistics.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*, pages 2122–2132.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1611–1622. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pages 1412–1421.

Kathleen McKeown. 1992. *Text generation - using discourse strategies and focus constraints to generate natural language text*. Studies in natural language processing. Cambridge University Press.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL HLT*, pages 720–730.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

T. Mikolov, M. Karafit, L. Burget, J. Cernock, and S. Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ehud Reiter. 2017. You need to understand your corpora! the weathergov example. https://ehudreiter.com/2017/05/09/weathergov/.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167(1-2):137–169.

Jacques Robin. 1994. *Revision-based generation of Natural Language Summaries providing historical Background*. Ph.D. thesis, Citeseer.

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

Radu Soricut and Daniel Marcu. 2006. Stochastic language generation using widl-expressions and its application in machine translation and summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 1105–1112. Association for Computational Linguistics.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1017–1024.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

Kumiko Tanaka-Ishii, Kôiti Hasida, and Itsuki Noda. 1998. Reactive content selection in the generation of real-time soccer commentary. In *COLING-ACL*, pages 1282–1288.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *AAAI*, pages 3097–3103.

Michael White, Rajakrishnan Rajkumar, and Scott Martin. 2007. Towards broad coverage surface realization with ccg. In *Proceedings of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+ MT)*, pages 267–276.

Yuk Wah Wong and Raymond J Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *HLT-NAACL*, pages 172–179.

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

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-aware language models. *CoRR*, abs/1611.01628.

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

Zhu Zhang. 2004. Weakly-supervised relation classification for information extraction. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 581–588. ACM.

GuoDong Zhou, JunHui Li, LongHua Qian, and Qiaoming Zhu. 2008. Semi-supervised learning for relation extraction. In *Third International Joint Conference on Natural Language Processing*, page 32.

All that is English may be Hindi: Enhancing language identification through automatic ranking of likeliness of word borrowing in social media

[1]Jasabanta Patro, [2]Bidisha Samanta, [2]Saurabh Singh,
[2]Abhipsa Basu,[2]Prithwish Mukherjee, [3]Monojit Choudhury, [4]Animesh Mukherjee
[1,2,4] Indian Institute of Technology Kharagpur, India – 721302
[6]Microsoft Research India, Bangalore – 560001
[1]jasabantapatro@iitkgp.ac.in, [3]monojitc@microsoft.com, [4]animeshm@cse.iitkgp.ernet.in

Abstract

In this paper, we present a set of computational methods to identify the likeliness of a word being borrowed, based on the signals from social media. In terms of Spearman's correlation values, our methods perform more than two times better (~ 0.62) in predicting the borrowing likeliness compared to the best performing baseline (~ 0.26) reported in literature. Based on this likeliness estimate we asked annotators to re-annotate the language tags of foreign words in predominantly native contexts. In 88% of cases the annotators felt that the foreign language tag should be replaced by native language tag, thus indicating a huge scope for improvement of automatic language identification systems.

1 Introduction

In social media communication, multilingual people often switch between languages, a phenomenon known as *code-switching* or *code-mixing* (Auer, 1984). This makes *language identification and tagging*, which is perhaps a prerequisite for almost all other language processing tasks that follow, a challenging problem (Barman et al., 2014). In code-mixing people are subconsciously aware of the foreign origin of the code-mixed word or the phrase. A related but linguistically and cognitively distinct phenomenon is *lexical borrowing* (or simply, *borrowing*), where a word or phrase from a foreign language say L_2 is used as a part of the vocabulary of native language say L_1. For instance, in Dutch, the English word "sale" is now used more frequently than the Dutch equivalent "uitverkoop". Some English words like "shop" are even inflected in Dutch as "shoppen" and heavily used. While it is difficult in general to ascertain whether a foreign word or phrase used in an utterance is borrowed or just an instance of code-mixing (Bali et al., 2014), one tell tale sign is that only proficient multilinguals can code-mix, while even monolingual speakers can use borrowed words because, by definition, these are part of the vocabulary of a language. In other words, just because an English speaker understands and uses the word "tortilla" does not imply that she can speak or understand Spanish. A borrowed word from L_2 initially appears frequently in speech, then gradually in print media like newspaper and finally it loses its origin's identity and is used in L_1 resulting in an inclusion in the dictionary of L_1 (Myers-Scotton, 2002; Thomason, 2003). Borrowed words often take several years before they formally become part of L_1 dictionary. This motivates our research question "is early-stage automatic identification of likely to be borrowed words possible?". This is known to be a hard problem because (i) it is a socio-linguistic phenomenon closely related to acceptability and frequency, (ii) borrowing is a dynamic process; new borrowed words enter the lexicon of a language as old words, both native and borrowed, might slowly fade away from usage, and (iii) it is a population level phenomenon that necessitates data from a large portion of the population unlike standard natural language corpora that typically comes from a very small set of authors. Automatic identification of borrowed words in social media content (SMC) can improve language tagging by recommending the tagger to tag the language of the borrowed words as L_1 instead

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2264–2274
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

of L_2. The above reasons motivate us to resort to the social media (in particular, Twitter), where a large population of bilingual/multilingual speakers are known to often tweet in code-mixed colloquial languages (Carter et al., 2013; Solorio et al., 2014; Vyas et al., 2014; Jurgens et al., 2017; Rijhwani et al., 2017). We designed our methodology to work for any pair of languages L_1 and L_2 subject to the availability of sufficient SMC. In the current study, we consider Hindi as L_1 and English as L_2.

The main stages of our research are as follows:

Metrics to quantify the likeliness of borrowing from social media signals: We define three novel and closely similar metrics that serve as social signals indicating the likeliness of borrowing. We compare the likeliness of borrowing as predicted by our model and a baseline model with that from the ground truth obtained from human judges.

Ground truth generation: We launch an extensive survey among 58 human judges of various age groups and various educational backgrounds to collect responses indicating if each of the candidate foreign word is likely borrowed.

Application: We randomly selected some words that have a high, low and medium borrowing likeliness as predicted by our metrics. Further, we randomly selected one tweet for each of the chosen words. The chosen words in almost all of these tweets have L_2 as their language tag while a majority of the surrounding words have a tag L_1. We asked expert annotators to re-evaluate the language tags of the chosen words and indicate if they would prefer to switch this tag from L_2 from L_1.

Finally, our key results are outlined below:

1. We obtained the Spearman's rank correlation between the ground-truth ranking and the ranking based on our metrics as ~ 0.62 for all the three variants which is more than double the value (~ 0.26) if we use the most competitive baseline (Bali et al., 2014) available in the literature.

2. Interestingly, the responses of the judges in the age group below 30 seem to correspond even better with our metrics. Since language change is brought about mostly by the younger population, this might possibly mean that our metrics are able to capture the early signals of borrowing.

3. Those users that mix languages the least in their tweets present the best signals of borrowing in case they do mix the languages (correlation of our metrics estimated from the tweets of these users with that of the ground truth is ~ 0.65).

4. Finally, we obtain an excellent re-annotation accuracy of 88% for the words falling in the surely borrowed category as predicted by our metrics.

2 Related work

In linguistics *code-mixing* and *borrowing* have been studied under the broader scope of language change and evolution. Linguists have for a long time focused on the sociological and the conversational necessity of borrowing and mixing in multilingual communities (see Auer (1984) and Muysken (1996) for a review). In particular, Sankoff et al. (1990) describes the complexity of choosing features that are indicative of borrowing. This work further showed that it is not always true that only highly frequent words are borrowed; nonce words could also be borrowed along with the frequent words. More recently, (Nzai et al., 2014) analyzed the formal conversation of Spanish-English multilingual people and found that code mixing/borrowing is not only restricted to daily speech but is also prevalent in formal conversations. (Hadei, 2016) showed that phonological integration could be evaluated to understand the phenomenon of word borrowing. Along similar lines, (Sebonde, 2014) showed morphological and syntactic features could be good indicators for numerical borrowings. (Senaratne, 2013) reported that in many languages English words are likely to be borrowed in both formal and semi-formal text.

Mixing in computer mediated communication and social media: (Sotillo, 2012) investigated various types of code-mixing in a corpora of 880 SMS text messages. The author observed that most often mixing takes place at the beginning of a sentence as well as through simple insertions. Similar observations about chat and email messages have been reported in (Bock, 2013; Negrón, 2009). However, studies of code-mixing with Chinese-English bilinguals from Hong Kong (Li, 2009) and Macao (San, 2009) brings forth results that contrast the aforementioned findings and indicate that in these societies code-mixing is driven more by linguistic than social motivations.

Recently, the advent of social media has immensely propelled the research on code-mixing and borrowing as a dynamic social phenomena. (Hidayat, 2012) noted that in Facebook, users mostly preferred inter-sentential mixing and showed that 45% of the mixing originated from real lexical needs, 40% was used for conversations on a particular topic and the rest 5% for content clarification. In contrast, (Das and Gambäck, 2014) showed that in case of Facebook messages, intra-sentential mixing accounted for more than half of the cases while inter-sentential mixing accounted only for about one-third of the cases. In fact, in the First Workshop on *Computational Approaches to Code Switching* a shared task on code-mixing in tweets was launched and four different code-mixed corpora were collected from Twitter as a part of the shared task (Solorio et al., 2014). Language identification task has also been handled for English-Hindi and English-Bengali code-mixed tweets in (Das and Gambäck, 2013). Part-of-speech tagging have been recently done for code-mixed English-Hindi tweets (Solorio and Liu, 2008; Vyas et al., 2014).

Diachronic studies: As an aside, it is interesting to note that the availability of huge volumes of timestamped data (tweet streams, digitized books) is now making it possible to study various linguistic phenomena quantitatively over different timescales. For instance, (Sagi et al., 2009) uses latent semantic analysis for detection and tracking of changes in word meaning, whereas (Fremann and Lapata, 2016) presents a Bayesian approach for the same problem. (Peirsman et al., 2010) presents a distributed model for automatic identification of lexical variation between language varieties. (Bamman and Crane, 2011) discusses a method for automatically identifying word sense variation in a dated collection of historical books . (Mitra et al., 2014) presents a computational method for automatic identification of change in word senses across different timescales. (Cook et al., 2014) presents a method for novel sense identification of words over time.

Despite these diverse and rich research agendas in the field of code-switching and lexical dynamics, there has not been much attempt to quantify the likeliness of borrowing of foreign words in a language. The only work that makes an attempt in this direction is (Bali et al., 2014), which is described in detail in Sec 3.1. One of the primary challenges faced by any quantitative research on lexical borrowing is that borrowing is a social phenomenon, and therefore, it is difficult to identify suitable indicators of such a lexical diffusion process unless one has access to a large population-level data. In this work, we show for the first time how certain simple and closely related signals encoding the language usage of social media users can help us construe appropriate metrics to quantify the likeliness of borrowing of a foreign word.

3 Methodology

In this section, we present the baseline metric and propose three new metrics that quantify the likeliness of borrowing.

3.1 Baseline metric

Baseline metric – We consider the $log(\frac{F_{L_2}}{F_{L_1}})$ value proposed in (Bali et al., 2014) as the baseline metric. Here F_{L_2} denotes the frequency of the L_1 transliterated form of the word w in the standard L_1 newspaper corpus. F_{L_1}, on the other hand, denotes the frequency of the L_1 translation of the word w in the same newspaper corpus. For our experiments discussed in the later sections, both the transliteration and the translation of the words have been done by a set of volunteers who are native L_1 speakers. The authors in (Bali et al., 2014) claim that the more positive the value of this metric is for a word w, the higher is the likeliness of its being borrowed. The more negative the value is, the higher are the chances that the word w is an instance of code-mixing.
Ranking – Based on the values obtained from the above metric for a set of target words, we rank these words; words with high positive values feature at the top of the rank list and words with high negative values feature at the bottom of the list. For two words having the same $log(\frac{F_{L_2}}{F_{L_1}})$ value, we resolve the conflict by assigning each of these the average of their two rank positions. In a subsequent section, we shall compare this rank list with the one obtained from the ground truth responses.

3.2 Proposed metric

In this section, we present three novel and closely related metrics based on the language usage patterns of the users of social media (in specific, Twitter). In order to define our metrics, we need all the words to be language tagged. The different tags that a word can have are: L_1, L_2, *NE* (Named Entity) and *Others*. Based on the word level tag, we also create a tweet level tag as follows:

1. L_1: Almost every word ($> 90\%$) in the tweet is tagged as L_1.
2. L_2: Almost every word ($> 90\%$) in the tweet is tagged as L_2.

3. CML_1: Code-mixed tweet but majority (i.e., $> 50\%$) of the words are tagged as L_1.
4. CML_2: Code-mixed tweet but majority (i.e., $> 50\%$) of the words are tagged as L_2.
5. *CMEQ*: Code-mixed tweet having very similar number of words tagged as L_1 and L_2 respectively.
6. *Code Switched*: There is a trail of L_1 words followed by a trail of L_2 words or vice versa.

Using the above classification, we define the following metrics:

Unique User Ratio (UUR) – The Unique User Ratio for word usage across languages is defined as follows:

$$UUR(w) = \frac{U_{L_1} + U_{CML_1}}{U_{L_2}} \tag{1}$$

where U_{L_1} (U_{L_2}, U_{CML_1}) is the number of unique users who have used the word w in a L_1 (L_2, CML_1) tweet at least once.

Unique Tweet Ratio (UTR) – The Unique Tweet Ratio for word usage across languages is defined as follows:

$$UTR(w) = \frac{T_{L_1} + T_{CML_1}}{T_{L_2}} \tag{2}$$

where T_{L_1} (T_{L_2}, T_{CML_1}) is the total number of L_1 (L_2, CML_1) tweets which contain the word w.

Unique Phrase Ratio (UPR) – The Unique Phrase Ratio for word usage across languages is defined as follows:

$$UPR(w) = \frac{P_{L_1}}{P_{L_2}} \tag{3}$$

where P_{L_1} (P_{L_2}) is the number of L_1 (L_2) phrases which contain the word w. Note that unlike the definitions of UUR and UTR that exploit the word level language tags, the definition of UPR exploits the phrase level language tags.

Ranking – We prepare a separate rank list of the target words based on each of the three proposed metrics – UUR, UTR and UPR. The higher the value of each of this metric the higher is the likeliness of the word w to be borrowed and higher up it is in the rank list. In a subsequent section, we shall compare these rank lists with the one prepared from the ground truth responses.

4 Experiments

In this section we discuss the dataset for our experiments, the evaluation criteria and the ground truth preparation scheme.

4.1 Datasets and preprocessing

In this study, we consider code-mixed tweets gathered from Hindi-English bilingual Twitter users in order to study the effectiveness of our proposed metrics. The native language L_1 is Hindi and the foreign language L_2 is English. To bootstrap the data collection process, we used the language tagged tweets presented in (Rudra et al., 2016). In addition to this, we also crawled tweets (between Nov 2015 and Jan 2016) related to 28 hashtags representing different Indian contexts covering important topics such as sports, religion, movies, politics etc. This process results in a set of 811981 tweets. We language-tag (see details later in this section) each tweet so crawled and find that there are 3577 users who use mixed language for tweeting. Next, we systematically crawled the time lines of these 3577 users between Feb 2016 and March 2016 to gather more mixed language tweets. Using this two step process we collected a total of 1550714 distinct tweets. From this data, we filtered out tweets that are not written in romanized script, tweets having only URLs and tweets having empty content. Post filtering we obtained 725173, tweets which we use for the rest of the analysis. The datasets can be downloaded from `http://cnerg.org/borrow`

Language tagging: We tag each word in a tweet with the language of its origin using the method outlined in (Gella et al., 2013). *Hi* represents a predominantly Hindi tweet, *En* represents a predominantly English tweet, *CMH* (*CME*) represents code-mixed tweets with more Hindi (English) words, *CMEQ* represents code-mixed tweets with almost equal number of Hindi and English words and *CS* represents code-switched tweets (the number and % of tweets in each of the above six categories are presented in the supplementary material). Like the word level, the tagger also provides a phrase level language tag. Once again, the different tags that an entire phrase can have are: *En*, *Hi* and *Oth* (Other). The metrics defined in the previous section are computed using these language tags.

Newspaper dataset for the baseline: As we had discussed in the previous section, for the construction of the baseline ranking we need to resort to counting the frequency of the foreign words (i.e., English words) and their Hindi translations in a newspaper corpus as has been outlined in (Bali et al., 2014). For this purpose, we use the FIRE dataset built from the Hindi Jagaran newspaper corpus[1] which is written in Devanagari script.

4.2 Target word selection

We first compute the most frequent foreign (i.e., English words) in our tweet corpus. Since we are interested in the frequency of the English word only when it appears as a foreign word we do not consider the (i) *Hi* tweets since they do not have any foreign word, (ii) *En* tweets since here the English words are not foreign words and the (iii) code-switched tweets. Based on the frequency of usage of English as a foreign word, we select the top 1000 English words. Removal of stop words and text normalization leaves beyond 230 nouns (see supplementary material for the list of words).

Final selection of target words: In language processing, context plays an important role in understanding different properties of a word. For our study, we also attempt to use the language tags as features of the context words for a given *target word*. Our hypothesis here is that there should exist classes of words that have similar context features and the likelihood of being borrowed in each class should be different. For example, when an English word is surrounded by mostly Hindi words it seems to be more likely borrowed. We present two examples in the box below to illustrate this.

Example I:
@****** Welcome. ***Film*** *jaroor dekhna. Nahi to injection ready hai.*
Example II:
@*****_Trust @*****
Kuch to ache se karo sirji....
Har jagah bhaagte rehna is not a good ***thing***.

In *Example I* the English word “film” is surrounded by mostly Hindi words. On the other hand, in *Example II* the English word “thing” is surrounded mostly by English words. Note that the word “film” is very commonly used by Hindi monolingual speakers and is therefore highly likely to have been borrowed unlike the English word “thing” which is arguably an instance of mixing. This socio-linguistic difference seems to be very appropriately captured by the language tag of the surrounding words of these two words in the respective tweets. Based on this hypothesis we arrange the 230 words into contextually similar groups (see supplementary material for the grouping details). Finally, using the baseline metric $log(\frac{F_E}{F_H})$ (E: English, H: Hindi), we proportionately choose words from these groups as follows:

Words with very high or very low values of $log(\frac{F_E}{F_H})$ ($hlws$) – we select words having the highest and the lowest values of $log(\frac{F_E}{F_H})$ from each of the context groups. This constitutes a set of 30 words. Note that these words are baseline-biased and therefore the metric should be able to discriminate them well.

Words with medium values of $log(\frac{F_E}{F_H})$ (mws) – we selected 27 words having not so high and not so low $log(\frac{F_E}{F_H})$ at uniformly at random.

Full set of words ($full$) – Thus, in total we selected 57 target words for the purpose of our evaluation. We present these words in the box below.

Baseline-biased words – ’thing’, ’way’, ’woman’, ’press’, ’wrong’, ’well’, ’matter’, ’reason’, ’question’, ’guy’, ’moment’, ’week’, ’luck’, ’president’, ’body’, ’job’, ’car’, ’god’, ’gift’, ’status’, ’university’, ’lyrics’, ’road’, ’politics’, ’parliament’, ’review’, ’scene’, ’seat’, ’film’, ’degree’
Randomly selected words – ’people’, ’play’, ’house’, ’service’, ’rest’, ’boy’, ’month’, ’money’, ’cool’, ’development’, ’group’, ’friend’, ’day’, ’performance’, ’school’, ’blue’, ’room’, ’interview’, ’share’, ’request’, ’traffic’, ’college’, ’star’, ’class’, ’superstar’, ’petrol’, ’uncle’

4.3 Evaluation criteria

We present a four step approach for evaluation as follows. We measure (i) how well the UUR, UTR and UPR based ranking of the $hlws$ set, the mws set and the $full$ set correlate with the ground truth ranking (discussed in the next section) in comparison to the rank given by the baseline metric, (ii) how well the different rank ranges obtained from our metric align with the ground truth as compared to the baseline metric, (iii) whether there are some systematic effects of the age group of the survey participants on the rank correspondence, (iv) how our metrics if computed from the tweets of users who (a) rarely mix languages, (b) almost always mix languages and (c) are in between (a) and (b), align with the ground truth.

Rank correlation: We measure the standard Spearman’s rank correlation (ρ) (Zar, 1972) pairwise between rank lists generated by (i) UUR (ii) UTR (iii) UPR (iv) baseline metric and the ground truth.

We shall describe the next four measurements

[1] Jagaran corpus: http://fire.irsi.res.in/fire/static/data

taking UUR as the running example. The same can be extended verbatim for the other two similar metrics.

Rank ranges: We split each of the three rank lists (UUR, ground truth and baseline) into five different equal-sized ranges as follows – (i) surely borrowed (SB) containing top 20% words from each list, (ii) likely borrowed (LB) containing the next 20% words from each list, (iii) borderline (BL) constituting the subsequent 20% words from each list, (iv) likely mixed (LM) comprising the next 20% words from each list and (v) surely mixed (SM) having the last 20% words from each rank list. Therefore, we have three sets of five buckets, one set each for UUR, the ground truth and the baseline based rank list.

Next we calculate the bucket-wise correspondence between (i) the UUR and the ground truth set and (ii) the baseline and the ground truth set in terms of standard *precision* and *recall* measures. For our purpose, we adapt these measures as follows.

G: ground truth bucket set, B_b: baseline bucket set, U_b: UUR bucket set;

$BS \in \{B_b, U_b\}$, T (type of bucket) = {SB, LB, BL, LM, SM};

b_t = words in type t bucket from BS, g_t = words in type t bucket from G, $t \in T$;

tp_t (no. of true positives) = $|b_t \cap g_t|$, fp_t (no. of false positives) = $|b_t - g_t|$, tn_t (no. of true negatives) = $|g_t - b_t|$;

Bucket-wise *precision* and *recall* therefore:

$precision(b_t) = \frac{tp_t}{fp_t + tp_t}$; $recall(b_t) = \frac{tp_t}{tn_t + tp_t}$

For a given set, we obtained the overall *macro precision* (*recall*) by averaging the *precision* (*recall*) values over the five buckets. For a given set, we also obtained the overall *micro precision* by first adding the true positives across all the buckets and then normalizing by the sum of the true and the false positives over all the buckets. We take an equivalent approach for obtaining the *micro recall*.

Age group effect: Here we construct two ground truth rank lists one using the responses of the participants with age below 30 (young population) and the other using the responses of the rest of the participants (elderly population). Next we repeat the above two evaluations considering each of the new ground truth rank lists.

Extent of language mixing: Here we divide all the 3577 users into three categories – (i) High (users who have more than 20% of tweets as code-mixed), (ii) Mid (users who have 7–20% of their tweets as code-mixed, and (iii) Low (users who have less than 7% of their tweets as code-mixed). We create three UUR based rank lists for each of these three user categories and respectively compare them with the ground truth rank list.

4.4 Ground truth preparation

Since it is very difficult to obtain a suitable ground truth to validate the effectiveness of our proposed ranking scheme, we launched an online survey to collect human judgment for each of the 57 *target words*.

Online survey We conducted the online survey[2] among 58 volunteers majority of whom were either native language(Hindi) speakers or had very high proficiency in reading and writing in that language. The participants were selected from different age groups and different educational backgrounds. Every participant was asked to respond to a multiple choice question about each of the 57 *target words*. Therefore, for every single *target word*, 58 responses were gathered. The multiple choice question had the following three options and the participants were asked to select the one they preferred the most and found more natural – (i) a Hindi sentence with the *target word* as the only English word, (ii) the same Hindi sentence in (i) but with the *target word* replaced by its Hindi translation and (iii) none of the above two options. There were no time restrictions imposed while gathering the responses, i.e., the volunteers theoretically had unlimited time to decide their responses for each *target word*.

Language preference factor For each *target word*, we compute a *language preference factor* (LPF) defined as ($Count_{En} - Count_{Hi}$), where $Count_{Hi}$ refers to the number of survey participants who preferred the sentence containing the Hindi translation of the *target word* while $Count_{En}$ refers to the number of survey participants who preferred the sentence containing the *target word* itself. More positive values of LPF denotes higher usage of *target word* as compared to its Hindi translation and therefore higher likeliness of the word being borrowed.

Ground truth rank list generation We generate the ground truth rank list based on the LPF score of a *target word*. The word with the highest value

[2] Survey portal: https://goo.gl/forms/L0kJm8BNMhRj0jA53

Rank-List$_1$	Rank-List$_2$	$\rho - hlws$	$\rho - mws$	$\rho - full$
UUR	Ground truth	**0.67**	**0.64**	**0.62**
UTR	Ground truth	**0.66**	**0.63**	**0.63**
UPR	Ground truth	**0.66**	**0.64**	**0.62**
Baseline	Ground truth	0.49	0.14	0.26

Table 1: Spearman's rank correlation coefficient (ρ) among the different rank lists. Best result is marked in bold.

of LPF appears at the top of the ground truth rank list and so on in that order. Tie breaking between *target words* having equal LPF values is done by assigning average rank to each of these words.

Age group based rank list: As discussed in the previous section, we prepare the age group based rank lists by first splitting the responses of the survey participants in two groups based on their age – (i) young population (age < 30) and (ii) elderly population (age $\geq$ 30). For each group we then construct a separate LPF based ranking of the *target words*.

5 Results

5.1 Correlation among rank lists

The Spearman's rank correlation coefficient (ρ) of the rank lists for the $hlws$ set, the mws set and the $full$ set according to the baseline metric, UUR, UTR and UPR with respect to the ground truth metric LPF are noted in table 1. We observe that for the $full$ set, the ρ between the rank lists obtained from all the three metrics UUR, UTR, and UPR with respect to the ground truth is ~ 0.62 which is more than double the ρ (~ 0.26) between the baseline and the ground truth rank list. This clearly shows that the proposed metrics are able to identify the likeliness of borrowing quite accurately and far better than the baseline. Further, a remarkable observation is that our metrics outperform the baseline metric even for the $hlws$ set that is baseline-biased. Likewise, for the mws set, our metrics outperform the baseline indicating a superior recall on arbitrary words. The ranking of the $full$ set of words obtained from the ground truth, the baseline and the UUR metric is available in the supplementary material.

We present the subsequent results for the $full$ set and the UUR metric. The results obtained for the other two metrics UTR and UPR are very similar and therefore not shown.

Bucket type	Ground truth bucket	Baseline bucket	UUR bucket
SB	11	11	11
LB	11	11	11
BL	12	12	12
LM	11	11	11
SM	12	12	12

Table 2: Number of words falling in each bucket of three bucket sets.

Bucket type	*prec./rec.* Baseline	*prec./rec.* UUR
SB	**0.27**	**0.27**
LB	0.09	**0.18**
BL	0.08	**0.33**
LM	0.18	**0.36**
SM	0.33	**0.50**

Table 3: Bucket-wise *precision/recall*. Best results are marked in bold.

5.2 Rank list alignment across rank ranges

The number of *target words* falling in each bucket across the three rank lists are the same and are noted in table 2. Thus, the *precision* and *recall* as per the definition are also the same. The bucket-wise *precision/recall* for the baseline and UUR with respect to the ground truth are noted in table 3. We observe that while in the SB bucket both the baseline and UUR perform equally well, for all the other buckets UUR massively outperforms the baseline. This implies that for the case where the likeliness of borrowing is the strongest, the baseline does as good as UUR. However, as one moves down the rank list, UUR turns out to be a considerably better predictor than the baseline. The overall *macro* and *micro precision/recall* as shown in table 4 further strengthens our observation that UUR is a better metric than the baseline.

5.3 Age group based analysis

As already discussed earlier, we split the ground truth responses based on the age group of the survey participants. We split the responses into two groups – (i) young population (age < 30) and (ii) elderly population (age ≥ 30) so that there are almost equal number of responses in both the groups (see supplementary material for the exact distribution).

Rank correlation: The Spearman's rank correlation of UUR and the baseline rank lists with these

Measure	Baseline	UUR
Macro prec./rec.	0.19	**0.33**
Micro prec./rec.	0.19	**0.33**

Table 4: Overall *macro* and *micro precision/recall*. Best results are marked in bold.

Rank-List$_1$	Rank-List$_2$	ρ
Baseline	Ground-truth-Young	0.26
UUR	Ground-truth-Young	**0.62**
Baseline	Ground-truth-Elder	0.27
UUR	Ground-truth-Elder	**0.53**

Table 5: Spearman's rank correlation across the two age groups. Best results are marked in bold.

Bucket type	Young-Baseline *p*/*r*	Young-UUR *p*/*r*	Elder-Baseline *p*	Elder-UUR *p*	Elder-baseline *r*	Elder-UUR *r*
SB	**0.27**	**0.27**	0.27	**0.36**	0.25	**0.33**
LB	0.09	**0.18**	0.09	**0.18**	0.08	**0.17**
BL	0.08	**0.33**	**0.16**	0.08	**0.28**	0.14
LM	0.18	**0.36**	0.18	**0.45**	0.14	**0.35**
SM	0.33	**0.5**	**0.41**	0.25	**0.41**	0.25

Table 6: Bucket-wise *precision* (*p*)/*recall* (*r*) for UUR and the $baseline$ metrics for the two new ground truths. Best results are marked in bold.

Measure	Young-Baseline	Young-UUR	Elder-Baseline	Elder-UUR
Mac. pre.	0.19	**0.33**	0.22	**0.27**
Mac. rec.	0.19	**0.33**	0.23	**0.25**
Mic. pre./rec.	0.19	**0.33**	0.23	**0.26**

Table 7: Overall *macro* and *micro precision* and *recall* for the two new ground truths. Best results are marked in bold.

Bucket	Number of users	ρ
High	302	0.52
Mid	744	0.60
Low	2531	**0.65**

Table 8: Spearman's correlation between UUR and the ground truth in the different user buckets. Best results are marked in bold.

two ground truth rank lists are shown in table 5. Interestingly, the correlation between UUR rank list and the young population ground truth is better than the elderly population ground truth. This possibly indicates that UUR is able to predict recent borrowings more accurately. However, note that the UUR rank list has a much higher correlation with both the ground truth rank lists as compared to the baseline rank list.

Rank ranges: Table 6 shows the bucket-wise *precision* and *recall* for UUR and the baseline metrics with respect to two new ground truths. For the young population once again the number of words in each bucket for all the three sets is the same thus making the values of the *precision* and the *recall* same. In fact, the *precision*/*recall* for this ground truth is exactly same as in the case of the original ground truth.

In contrast, when we consider the ground truth based on the responses of the elderly population, the number of words across the different buckets are different across the three sets. In this case, we observe that the *precision*/*recall* values are better for the UUR metric in SB, LB and LM buckets.

Finally, the overall *macro* and *micro precision* and *recall* for both the age groups are noted in table 7. Once again, for both the young and the elderly population based ground truths, the *macro* and *micro precision* and *recall* values for the UUR metric are higher compared to that of the baseline.

5.4 Extent of language mixing

As mentioned earlier, we divide the set of 3577 users into three categories. The Spearman's correlation between UUR and the ground truth for each of these buckets are given in table 8. As we can see, for Low bucket the ρ value is maximum. This points to the fact that the signals of borrowing is strongest from the users who rarely mix languages.

6 Re-annotation results

In order to conduct the re-annotation experiments we performed the following. To begin with, we ranked all the 230 English nouns in non-increasing order of their UUR values. We then randomly selected 20 words each having (i) high UUR (top 20%) values (call TOP), (ii) low UUR (bottom 20%) values (call BOT), and (iii) middle UUR (middle 20%) values (call MID). This makes a total of 60 words. Using this word list we extracted one tweet each that contained the (foreign) word from this list along with all other words in the tweet tagged in Hindi (H_{all}). We similarly prepared another such list of 60 words and extracted one tweet each in which most of the other words were tagged in Hindi (H_{most}).

We presented the selected words and the corresponding tweets to a set of annotators and asked them to annotate these selected words once again. Over all the words, we calculated the mean ($\mu_{E\rightarrow H}$) and the standard deviation ($\sigma_{E\rightarrow H}$) of the fraction of cases where the annotators altered the tag of the selected word from English to Hindi. The average inter-annotator agreement (Fleiss, 1971) for our experiments was found to be as high as 0.64. For the words in the TOP list, the fraction of times the tag is altered is **0.91** (**0.85**) with an inter-annotator agreement of 0.84 (0.80) for the

Word-rank	Context	$\mu_{E \to H}$	$\sigma_{E \to H}$
TOP	H_{all}	**0.91**	0.15
TOP	H_{most}	**0.85**	0.23
MID	H_{all}	0.58	0.28
MID	H_{most}	0.61	0.34
BOT	H_{all}	0.13	0.18
BOT	H_{most}	0.16	0.21

Table 9: Re-annotation results

H_{all} (H_{most}) category. In other words, on average, in as high as 88% of cases the annotators altered the tags of the words that are highly likely to be borrowed (i.e., TOP) in a largely Hindi context (i.e., H_{all} or H_{most}). Table 9 shows the fractional changes for all the other possible cases. An interesting observation is that the annotators rarely flipped the tags for the words in the BOT list (i.e., the sure mixing cases) in either of the H_{all} or the H_{most} contexts. These results strongly support the inclusion of our metric in the design of future automatic language tagging tasks.

7 Discussion and conclusion

In this paper, we proposed a few new metrics for estimating the likliness of borrowing that rely on signals from large scale social media data. Our best metric is two-fold better than the previous metric in terms of the accuracy of the prediction. There are some interesting linguistic aspects of borrowing as well as certain assumptions regarding the social media users that have important repercussions on this work and its potential extensions, which are discussed in this section.

Types of borrowing: Linguists define broadly three forms of borrowing, (i) *cultural*, (ii) *core*, and (iii) *therapeutic* borrowings. In *cultural borrowing*, a foreign word gets borrowed into native language to fill a lexical gap. This is because there is no equivalent native language word present to represent the same foreign word concept. For instance, the English word 'computer' has been borrowed in many Indian languages since it does not have a corresponding term in those languages[3]. In *core borrowing*, on the other hand, a foreign word replaces its native language translation in the native language vocabulary. This occurs due to overwhelming use of the foreign word over native language translation as a matter of prestige, ease of use etc. For example, the English word 'school' has become much more prevalent than its corresponding Hindi translation '*vidyalaya*' among the native Hindi speakers. Finally, therapeutic borrowing refers to borrowing of words to avoid taboo and homonomy in the native language. In this paper, although we did not perform any category based studies, most of our focus was on core borrowing.

Language of social media users: We assumed that if a user is predominantly using Hindi words in a tweet then the chances of him/her being a native Hindi speaker should be high, since, while the number of English native speakers in India is 0.02%, the number of Hindi native speakers is 41.03%[4]. This assumption has also been made in earlier studies (Rudra et al., 2016). Note that even if a user is not a native Hindi speaker but a proficient (or semi-proficient) Hindi speaker, the main results of our analysis should hold. For instance, consider two foreign words 'a' and 'b'. If 'a' is frequently borrowed in the native language, then the proficient speaker would also tend to borrow 'a' similar to a native speaker. Even if due to lack of adequate native vocabulary, the non-native speaker borrows the word 'b' in some cases, these spurious signals should get eliminated since we are making an aggregate level statistics over a large population.

Future directions: It would be interesting to understand and develop theoretical justification for the metrics. Further, it would be useful to study and classify various other linguistic phenomena closely related to core borrowing, such as: (i) *loanword*, where a form of a foreign word and its meaning or one component of its meaning gets borrowed, (ii) *calques*, where a foreign word or idiom is translated into existing words of native language, and (iii) *semantic loan*, where the word in the native language already exists but an additional meaning is borrowed from another language and added to existing meaning of the word.

Finally, we would also like to incorporate our findings into other standard tasks of multilingual IR and multilingual speech synthesis (for example to render the appropriate native accent to the borrowed word).

[3] Words for "computer" were coined in many Indian languages through morphological derivation from the term for "compute", however, none of these words are used in either formal or informal contexts.

[4] http://bit.ly/2ufOvea

References

P. Auer. 1984. *Bilingual Conversation*. John Benjamins.

K. Bali, J. Sharma, M. Choudhury, and Y. Vyas. 2014. "i am borrowing ya mixing?" an analysis of english-hindi code mixing in facebook. In *First workshop on Computational approaches to code-switching, EMNLP*, page 116.

David Bamman and Gregory Crane. 2011. Measuring historical word sense variation. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, pages 1–10. ACM.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. *EMNLP 2014*, 13.

Z. Bock. 2013. Cyber socialising: Emerging genres and registers of intimacy among young south african students. *Language Matters: Studies in the Languages of Africa*, 44(2):68–91.

S. Carter, W. Weerkamp, and M. Tsagkias. 2013. Micro-blog language identification: Overcoming the limitations of short, unedited and idiomatic text. *Language Resources and Evaluation*, 47(1):195–215.

Paul Cook, Jey Han Lau, Diana McCarthy, and Timothy Baldwin. 2014. Novel word-sense identification. In *COLING*, pages 1624–1635.

A. Das and B. Gambäck. 2013. code-mixing in social media text: The last language identification frontier? *TAL*, 54(3):41–64.

A. Das and B. Gambäck. 2014. Identifying languages at the word level in code-mixed indian social media text. In *ICON*, pages 169–178.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Lea Frermann and Mirella Lapata. 2016. A bayesian model of diachronic meaning change. *Transactions of the Association for Computational Linguistics*, 4:31–45.

S. Gella, J. Sharma, and K. Bali. 2013. Query word labeling and back transliteration for indian languages: Shared task system description. *FIRE Working Notes*, 3.

M. Hadei. 2016. Single word insertions as code-switching or established borrowing? *International Journal of Linguistics*, 8(1):14–25.

T. Hidayat. 2012. An analysis of code switching used by facebookers (a case study in a social network site). In *BA thesis*. English Education Study Program, College of Teaching and Education (STKIP), Bandung, Indonesia.

David Jurgens, Yulia Tsvetkov, and Dan Jurafsky. 2017. Incorporating dialectal variability for socially equitable language identification. *ACL 2017*.

D. C. S. Li. 2009. Cantonese-english code-switching research in hong-kong: a y2k review. *World Englishes*, 19(3):305–322.

Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee, and Pawan Goyal. 2014. That's sick dude!: Automatic identification of word sense change across different timescales. *arXiv preprint arXiv:1405.4392*.

P. Muysken. 1996. Code-switching and grammatical theory. In *One speaker, two languages: Cross-disciplinary perspectives on code-switching*, pages 177–198. Cambridge University Press.

C. Myers-Scotton. 2002. *Contact linguistics: Bilingual encounters and grammatical outcomes*. Oxford University Press.

R. G. Negrón. 2009. Spanish-english code-switching in email communication. *Language@Internet*, 6(3).

V. E. Nzai, Y-L. Feng, M. R. Medina-Jiménez, and J. Ekiaka-Oblazamengo. 2014. Understanding code-switching & word borrowing from a pluralistic approach of multilingualism.

Yves Peirsman, Dirk Geeraerts, and Dirk Speelman. 2010. The automatic identification of lexical variation between language varieties. *Natural Language Engineering*, 16(4):469–491.

Shruti Rijhwani, Royal Sequiera, Monojit Choudhury, Kalika Bali, and Chandra Shekhar Maddila. 2017. Estimating code-switching on twitter with a novel generalized word-level language detection technique. *ACL 2017*.

Koustav Rudra, Shruti Rijhwani, Rafiya Begum, Kalika Bali, Monojit Choudhury, and Niloy Ganguly. 2016. Understanding language preference for expression of opinion and sentiment: What do hindi-english speakers do on twitter? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1131–1141.

Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2009. Semantic density analysis: Comparing word meaning across time and phonetic space. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 104–111. Association for Computational Linguistics.

H. K. San. 2009. Chinese-english code-switching in blogs by macao young people. In *MSc. thesis*. University of Edinburgh.

D. Sankoff, S. Poplack, and S. Vanniarajan. 1990. The case of the nonce loan in tamil. *Language variation and change*, 2(01):71–101.

R. Y. Sebonde. 2014. Code-switching or lexical borrowing: Numerals in chasu language of rural tanzania. *Journal of Arts and Humanities*, 3(3):67.

C D. W. Senaratne. 2013. Borrowings or code mixes: The presence of lone english nouns in mixed discourse.

T. Solorio, E. Blair, S. Maharjan, S. Bethard, M. Diab, M. Gohneim, A. Hawwari, F. AlGhamdi, J. Hirschberg, A. Chang, and P. Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *EMNLP First Workshop on Computational Approaches to Code Switching*, pages 62–72.

T. Solorio and Y. Liu. 2008. Part-of-speech tagging for english-spanish code-switched text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1051–1060. Association for Computational Linguistics.

S. Sotillo. 2012. Ehhhh utede hacen plane sin mi???:@ im feeling left out :(form, function and type of code switching in sms texting. In *ICAME*, pages 309–310.

S. G. Thomason. 2003. Contact as a source of language change. *The handbook of historical linguistics*, pages 687–712.

Y. Vyas, S. Gella, J. Sharma, K. Bali, and M. Choudhury. 2014. POS tagging of english-hindi code-mixed social media content. In *EMNLP*, volume 14, pages 974–979.

J. H. Zar. 1972. Significance testing of the spearman's rank correlation coefficient. *Journal of the American Statistical Association*, 67(339):578–580.

Multi-View Unsupervised User Feature Embedding for Social Media-based Substance Use Prediction

Tao Ding
Department of Information Systems
University of Maryland, Baltimore County
taoding01@umbc.edu

Warren K. Bickel
Addiction Recovery Research Center
Virginia Tech Carilion Research Institute
wkbickel@vtc.vt.edu

Shimei Pan
Department of Information Systems
University of Maryland, Baltimore County
shimei@umbc.edu

Abstract

In this paper, we demonstrate how the state-of-the-art machine learning and text mining techniques can be used to build effective social media-based substance use detection systems. Since a substance use ground truth is difficult to obtain on a large scale, to maximize system performance, we explore different unsupervised feature learning methods to take advantage of a large amount of unsupervised social media data. We also demonstrate the benefit of using multi-view unsupervised feature learning to combine heterogeneous user information such as Facebook "likes" and "status updates" to enhance system performance. Based on our evaluation, our best models achieved 86% AUC for predicting tobacco use, 81% for alcohol use and 84% for illicit drug use, all of which significantly outperformed existing methods. Our investigation has also uncovered interesting relations between a user's social media behavior (e.g., word usage) and substance use.

1 Introduction

A substance use disorder (SUD) is defined as a condition in which recurrent use of substances such as alcohol, drugs and tobacco causes clinically and functionally significant impairment in an individual's daily life (SAMHSA, 2015). According to the 2014 National Survey on Drug Use and Health, 1 in 10 Americans age 12 and older had a substance use disorder. Substance use also costs Americans more than $700 billion a year in increased health care costs, crimes and lost productivity (NIDA, 2015).

These days, people also spend a significant amount of time on social media such as Twitter, Facebook and Instagram to interact with friends and families, exchange ideas and thoughts, provide status updates and organize events and activities. The ubiquity and widespread use of social media underlines the needs to explore its intersection with substance use and its potential as a scalable and cost-effective solution for screening and preventing substance misuse and abuse.

In this research, we employ the state-of-the-art machine learning and text mining algorithms to build automated substance use prediction systems, which can be used to identify people who are at risk of SUD. Since SUD data are often expensive to obtain at a large scale, to maximize system performance, we focus on methods that employ unsupervised feature learning to take advantage of a large amount of unsupervised social media data. Previous research in Machine Learning, Image Processing, Speech and Natural language Processing has shown that to be able to utilize a large amount of unsupervised data is one of the most reliable ways to achieve good performance (Le et al., 2011; Lee et al., 2009; Le and Mikolov, 2014). Moreover, by analyzing rich human behavior data on social media, we can also gain insight into patterns of use and risk factors associated with substance use. The main contributions of this work include:

1. We have explored a comprehensive set of single-view feature learning methods to take advantage of a large amount of unsupervised social media data. Our results have shown significant improvement over baseline systems that only use supervised training data.

2. We have explored several multi-view learning algorithms to take advantage of heteroge-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2275–2284
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

neous user data such as Facebook “likes” and “status updates”. Our results have demonstrated significant improvement over baselines that only use a single data type.

3. We have uncovered new insight into the relationship between a person’s social media activities and substance use such as the relationship between word usage and SUD.

2 Related Work

Substance use disorder (SUD) encompasses a complex pattern of behaviors. Many studies have been conducted to discover factors interacting with SUD. A growing number of studies have confirmed a strong association between personal traits and substance use. For example, (Campbell et al., 2014) found that smokers have significantly higher *openness to experience* and lower *conscientiousness*, a personality trait related to a tendency to show self-discipline, act dutifully, and aim for achievement. (Cook et al., 1998) examined the links between alcohol consumption and personality and found that alcohol use is correlated positively with sociability and extraversion. (Terracciano et al., 2008) conducted a study involving 1102 participants and found a link between drug use and low *conscientiousness*. (Carroll et al., 2009) revealed risk factors related to addiction such as age, sex, impulsivity, sweet-liking, novelty reactivity, proclivity for exercise, and environmental impoverishment. Additionally, addiction is also linked to environmental and social factors such as neighborhood environment (Crum et al., 1996), family environment (Cadoret et al., 1986; Brent, 1995) and social norms (Botvin, 2000; Oetting and Beauvais, 1987).

Traditionally, in behavior science research, data are collected from surveys or interviews with a limited number of people. The advent of social media makes a large volume of diverse user data available to researchers, which makes it possible to study SUD based on online user behaviors in a natural setting. Typical data from social media include demographics (age, gender etc.), status updates (text posts etc.), social networks (follower and following graph etc.) and likes (thumb up/down etc.). Recently, social media analytics has increasingly become a powerful tool to help understand the traits and behaviors of millions of social media users such as personal traits (Golbeck et al., 2011; Volkova and Bachrach, 2015; Youyou et al., 2015; Kiliç and Pan, 2016), brand preferences (Yang et al., 2015), communities and events (Sayyadi et al., 2009), influenza trend (Aramaki et al., 2011) and crime (Li et al., 2012). So far, however, there has been limited work that directly applies large scale social media analytics to automatically predict SUD. Among the work known to us, (Zhou et al., 2016) identified common drug consumption behaviors with regard to the time of day and week. They also discovered common interests shared by drug users such as celebrities (e.g, Chris Tucker) and comedians (e.g., cheechandchong). In addition, (Kosinski et al., 2013) automatically predicted SUD based on social media likes. Since their dataset is very similar to ours, we will use the Kosinski model as one of our baselines.

3 Dataset

The data for the study was collected from 2007 to 2012 as a part of the myPersonality project (Kosinski et al., 2015). myPersonality was a popular Facebook application that offered to its users psychometric tests and feedback on their scores. The data were gathered with an explicit opt-in consent for reuse for research purposes. Our study uses three separate datasets from myPersonality: Facebook status updates (a.k.a. posts), Facebook likes and SUD status.

The *status update dataset* contains 22 million textual posts authored by 153,000 users. The average posts per user is 143 and the average words per user is 1730. We removed users who only have non-English posts and those who have written less than 500 words. Our final status update dataset includes 106,509 users with 21 million posts. After filtering out low frequency words (those appear less than 50 times in our corpus), the vocabulary size of the status update dataset is 73,935.

The *likes dataset* contains the Facebook likes used to express positive sentiment toward various targets such as products, movies, books, expressions, websites and people (they are called *Like Entities* or LEs). Previous studies have demonstrated that social media likes speak volumes about who we are. In addition to directly signaling interests and preferences, social media likes are indicative of ethnicity, intelligence and personality (Kosinski et al., 2013). The like dataset includes the likes of 11 million Facebook users.

Overall, there are 9.9 million unique LEs and 1.8 billion user-like pairs in this dataset. The average likes per user is 161 and the average Likes each LE received is 182. We filter out users who have a small number of likes as well as LEs receiving a small number of likes. The filtering threshold for users is 50 and is 800 for LEs. After the filtering, our like dataset contains 5,138,857 users and 253,980 unique LEs.

The SUD dataset contains a total of 13,557 participants (Stillwell and Tunney, 2012). Users were asked to answer questions like “Do you smoke?”, with answers “daily or more”, “less than daily” or “never”. They also completed the Cigarette Dependence Scale (CDS-5) (Etter et al., 2003), Alcohol Use Questionnaire(AUQ) (Townshend and Duka, 2005) and the Assessment of Substance Misuse Questionnaire (ASMA) (Willner, 2000). Based on these assessments, the participants were divided into groups for each SUD type. For example, based on the assessment of tobacco use, a person is categorized as “daily or more” (group 3), “less than daily” (group 2), or “never” (group 1). The validity of the grouping was confirmed by the CDS-5 scores of the groups. Similarly, based on the assessment of alcohol use, participants were categorized as “weekly or more” (group 3), “less than once a week” (group 2) or “never” (group 1). Finally, based on the assessment of drug use, a person is assigned to “weekly or more” (group 3), “less than once a week” (group 2), or “never” (group 1). Among all the SUD participants, 37% of them are males and 63% are females. Their average age is 23 years old.

Since the like, status update and SUD datasets are only partially overlapping, their intersections are usually much smaller. Table 1 summarizes the sizes and usage of these datasets. Table 2 shows additional details of the SUD dataset including the distributions of each SUD class.

In summary, among all the datasets we have, the unsupervised like dataset is the largest (5 million+ people). We also have a significant amount of unsupervised status update data (100k+ users). In contrast, the supervised datasets which have the SUD ground truth are pretty small, ranging from 896 for the intersection of the likes, status updates and SUD (LikeStatusSUD in Table 1) to 3508, which is the intersection of the likes and SUD (LikesSUD in Table 1). Thus, the main focuses of this research include (1) employing unsupervised feature learning to take advantage of a large amount of unsupervised data (2) employing multi-view learning to combine heterogeneous user data for better prediction.

4 Single-View Post Embedding (SPE)

The main purpose of this study is to demonstrate the usefulness of employing unsupervised feature learning to derive a feature representation of a user’s Facebook posts to take advantage of a large amount of unsupervised data. Since we only use Facebook status updates (a.k.a. posts) in this study, we call the process Single-view user Post Embedding (SPE).

4.1 SPE Feature Learning Methods

Since each user is associated with a sequence of textual posts, we have explored the following methods to learn a SPE for the user.

Singular Value Decomposition (SVD) is a mathematical technique that is frequently used for dimension reduction (De Lathauwer et al., 2000). Given any $m * n$ matrix A, the algorithm will find matrices U, V and W such that $A = UWV^T$. Here U is an orthonormal $m * n$ matrix, W is a diagonal $n * n$ metrix and V is an orthonormal $n * n$ matrix. Dimensionality reduction is done by computing $R = U * W_r$ where W_r neglects all but the r largest singular values in the diagonal matrix W. In our study, the m is the number of users, n is the number of unique words in the vocabulary. $A_{ij} = k$ where k is how many times $word_j$ appears in $user_i$’s posts.

Latent Dirichlet Allocation (LDA) is a generative graphical model that allows sets of documents to be explained by unobserved latent topics (Blei et al., 2003). For each document, LDA outputs a multinomial distribution over a set of latent topics. For each topic, LDA also outputs a multinomial distribution over the vocabulary.

To learn an SPE for each user based on all his/her posts, we have tried several methods (1) UserLDA: it treats all the posts from each user as one big document and trains an LDA model to drive the topic distribution for this document. The per-document topic distribution is then used as the SPE for this user. (2) PostLDA_Doc: it treats each post as a separate document and trains an LDA model to derive a topic distribution for each post. To derive the SPE for each user, we aggregate

Table 1: Dataset Descriptions

Dataset	users	AvgUserLikes	AvgUserPosts	Usage
Likes	5,138,857	184	NA	Single View Feature Learning
LikesSUD	3,508	267	NA	Single View SUD Prediction
Status Update	106,509	NA	143	Single View Feature Learning
StatusSUD	1,231	NA	195	Single View SUD Prediction
LikeStatus	54,757	232	220	Multi-View Feature Learning
LikeStatusSUD	896	277	219	Multi-View SUD Predication

Table 2: Class Distribution of Different SUD Datasets

Dataset	Tabacco Use			Alcohol Use			Drug Use		
	3	2	1	3	2	1	3	2	1
LikeSUD	498	290	2603	469	1174	1716	171	276	1965
StatusSUD	226	95	880	179	416	596	76	102	671
LikeStatusSUD	147	69	660	123	290	453	262	53	75

all the per-post topic distribution vectors from the same user by averaging them. (3) PostLDA_Word: instead of using the $average$ of post-based topic distribution vectors, we used a word-based aggregation method suggested in (Schwartz et al., 2013):

$$p(topic|user) = \sum_{w \in voc} P(topic|w) * p(w|user)$$

where voc represents the vocabulary, $p(w|user)$ is the probability that word w appears in the posts of $user$ and $p(topic|w)$ is the topic distribution of a word w, which is available internally in an LDA model. For the UserLDA model, all the hyper parameters were set to default values. For all the PostLDA models, since Facebook posts are usually short and have a small number of topics in each post, we set the hyper parameter α to 0.3, as suggested in (Schwartz et al., 2013)

Document Embedding with Distributed Memory (D-DM) Given a document, D-DM simultaneously learns a vector representation for each word and a vector for the entire document (Le and Mikolov, 2014). During training, the document vector and one or more word vectors are aggregated to predict a target word in the context. To learn an SPE for each user, we have explored two methods (1) User-D-DM: it treats all the posts by the same user as one document and trains a document vector to represent the user. (2) Post-D-DM: it treats each post as a document and train a D-DM to learn a vector for each post. To derive the SPE for a user, we aggregate all the post vectors from the same person using "average".

Table 3: SPE: Prediction Results

Methods	Tobacco	Alcohol	Drug
Unigram	0.663	0.672	0.644
LIWC	0.731	0.689	0.758
SVD	0.779	0.724	0.764
UserLDA	0.641	0.603	0.599
PostLDA_Word	0.733	0.617	0.628
PostLDA_Doc	0.768	0.687	0.721
Post-D-DM	0.536	0.622	0.520
User-D-DM	0.775	0.730	0.767
Post-D-DBOW	0.531	0.606	0.526
User-D-DBOW	**0.802**	**0.768**	**0.819**

Document Embedding with Distributed Bag of Words (D-DBOW) D-DBOW learns a global document vector to predict words randomly sampled from the document (Le and Mikolov, 2014). Unlike D-DM, D-DBOW only learns a vector for the entire document. It does not learn vectors for individual words. Neither does it use a local context window since the words for prediction are randomly sampled from the entire document. Similar to D-DM, to derive the SPE for a user, we used two methods (1) User-D-DBOW and (2) Post-D-DBOW.

4.2 SUD Prediction with SPE

In our experiments, to search for the best model, we systematically varied the output SPE dimension from 50, 100, 300, to 500. We used the Gensim implementation of SVD, LDA, D-DM and D-DBOW in our experiments. For D-DM, the context window size was set to 5.

We compared our models with two baselines that use only supervised learning (1) a unigram model which uses unigrams as the predicting features. Since we have a large number of unigrams, we performed supervised feature selection to lower the total number of input features. Finally since all our SUD variables have three values, we employed SVM in 3-way classifications. (2) a LIWC model which uses human engineered LIWC features for SUD prediction. LIWC is a psycholinguistic lexicon (Pennebaker et al., 2015) that has been frequently used in text-based human behavior prediction. Since the number of LIWC features is relatively small, no feature selection was performed. Here, we only used the *Status Update* dataset in Table 1 as the training data for SPE learning and the *StatusSUD* dataset for supervised SUD prediction .

We evaluate the performance of our models using 10-fold cross validation. The evaluation results shown in Table 3 are based on weighted ROC AUC of the best models. Among all the feature learning methods for Facebook status updates, User-D-DBOW performed the best. It significantly outperformed all the baseline systems that only rely on supervised training ($p < 0.01$ based on t-tests). It also significantly outperformed all the traditional feature learning methods such as LDA and SVD ($p < 0.01$ based on t-tests). Moreover, in terms of whether to treat all the posts by the same user as one big document or separate documents, LDA prefers one post one document (models with a "post" prefix) while all the document vector-based methods prefer one user one document (models with a "User" prefix). Moreover, to use post-level LDA to derive the SPE of a user, the document-based aggregation method (PostLDA_Doc) performed better than the word-based method (PostLDA_Word).

5 Single-View Like Embedding (SLE)

In addition to textual posts, each user account is also associated with a set of likes. Since the like dataset is very sparse (e.g., among the millions of unique likes on Facebook, each user only has a small number of likes), we conduct experiments to learn a dense vector representation for all the likes by a user. We call this process Single-view user Like Embedding (SLE).

5.1 SLE Feature Learning Methods

The input to SLE is simply a set of LEs liked by a user. Each LE is represented by its id. To map such a representation to a dense user like vector, we have tried multiple methods.

Singular Value Decomposition (SVD) is similar to the one used in SPE except $A_{ij} = 1$ if $user_i$ likes LE_j. Otherwise, it is 0. Here A is a $m * n$ matrix where m is the number of users and n is the number of unique LEs in the like dataset.

Latent Dirichlet Allocation (LDA). To apply LDA to the like data, each individual LE is treated as a word token and all the LEs liked by the same person form a document. The order of the LEs in the document is random. For each user, LDA outputs a multinomial distribution over a set of latent "Like Topics". For example, a "Like Topic" about "hip hop music" may include famous hip hop songs and musicians.

Autoencoder (AE) is a neural network-based method for self-taught learning (Hinton and Salakhutdinov, 2006). It learns an identity function so that the output is as close to the input as possible. Although an identity function seems a trivial function to learn, by placing additional constraints (e.g,, to make the number of neurons in the hidden layer much smaller than that of the input), we can still force the system to uncover structures in the data. Architecturally, the AE we used has one input layer, one hidden layer and one output layer. For each user, we construct a training instance (X, Y) where the input vector X and output vector Y are the same. The size of X and Y is the total number of unique LEs in our dataset. X_i and Y_i equal to 1 if the user likes LE_i. Otherwise they are 0.

Document Vector with Distributed Memory (D-DM) We also applied D-DM to the like data. Given all the likes of a user, D-DM learns a vector representation for each LE as well as a document vector for all the LEs from the same user. We use the learned document vector as the output SLE.

Document Vector with Distributed Bag of Words (D-DBOW) Similarly, we applied D-DBOW to the like dataset. Since D-DBOW does not use a local context window and the words for prediction are randomly sampled from the entire document, it is more appropriate for the like dataset than D-DM. where the relative positions of LEs do matter.

5.2 SUD Prediction with SLE

Similarly, we systematically varied the output SLE dimension from 50, 100, 300, to 500 in order to search for the best model. We used the Gensim implementation of SVD, LDA, D-DM and D-

Table 4: SLE: Prediction Results

Method	Tobacco	Alcohol	Drug
Unigram	0.687	0.651	0.673
Kosinski⋆	0.730*	0.700*	0.650*
AE	0.678	0.648	0.672
SVD	0.757	0.756	0.753
LDA	0.723	0.737	0.704
D-DM	0.688	0.713	0.687
D-DBOW	**0.787**	**0.795**	**0.791**

⋆:2-way classification, 3-way for the others

DBOW in our experiments. For D-DM, the context window size was set to 20. We used Keras with Theano backend to implement AE.

We used SVM to perform 3-way classification. We compared our results with a unigram baseline. We also compared our results with that of the Kosinski model reported in (Kosinski et al., 2013). The Kosinski model was trained on the same Facebook like dataset. However, its results were based on two-way classification, a simpler task than 3-way classification. All the results are based on weighted ROC AUC.

As shown in Table 4, among all the SLE methods, the D-DBOW model performed the best. It significantly outperformed the unigram baseline that does not use any unsupervised data ($p < 0.01$ based on t-tests). It also significantly outperformed all the traditional feature learning method such as SVD and LDA (The Kosinski model used SVD for feature learning) ($p < 0.01$ based on t-tests). Between the two document vector-based methods, D-DBOW outperformed D-DM. We think this is due to the fact that D-DBOW does not use local context window, thus is not sensitive to the positions of LEs in a document. Since LE positions are randomly decided in our like data, D-DBOW seems to be a better fit for this dataset.

6 Multi-View User Embedding (MUE)

The main purpose of this study is to demonstrate the usefulness of combining heterogeneous user data such as likes and posts to learn a dense vector representation for each user. Since we employ unsupervised multi-view feature learning to combine these data, we call this process Multi-view User Embedding (MUE).

6.1 MUE Feature Learning Methods

We have explored two multi-view feature learning algorithms: Canonical Correlation Analysis (CCA) and Deep Canonical Correlation Analysis (DCCA).

Canonical Correlation Analysis (CCA) CCA is a statistical method for exploring the relationships between two multivariate sets of variables (vectors) (Hardoon et al., 2004). Given two vectors X and Y, CCA tries to find aX, bY that are maximally correlated:

$$(a^*, b^*) = \arg\max_{a,b} corr(a^{'}X, b^{'}Y) \qquad (1)$$

$$= \arg\max_{a,b} \frac{a^{'} \sum_{XY} b}{\sqrt{a^{'} \sum_{XX} ab^{'} \sum_{YY} b}} \qquad (2)$$

where (X, Y) denote random vectors with covariances $\sum_{XX} = Cov(X, X)$, $\sum_{YY} = Cov(Y, Y)$ and cross-covariance $\sum_{XY} = Cov(X, Y)$. CCA has been used frequently in unsupervised data analysis (Sargin et al., 2006; Chaudhuri et al., 2009; Kumar and Daumé, 2011; Sharma et al., 2012).

Deep Canonical Correlation Analysis (DCCA) DCCA aims to lean highly correlated deep architectures, which can be a non-linear extension of CCA (Andrew et al., 2013). The intuition is to find a maximally correlated representation of the two views by passing them through multiple stacked layers of nonlinear transformation (Andrew et al., 2013). Typically, there are three steps to train DCCA: (1) using a denoising autoencoder to pretrain each single view. In our experiments, we pretrain each single view using SPE or SLE. (2) computing the gradient of the correlation of top-level representation. (3) tuning parameters using back propagation to optimize the total correlation. Previously, DCCA was found to be more effective than CCA in image processing (Andrew et al., 2013).

6.2 SUD Prediction With MUE

The input to MUE are the two single views obtained earlier (i.e. SPE or SLE). Here, we choose the outputs from D-DBOW since it consistently outperformed all the other methods in learning SPEs and SLEs. We have run CCA and DCCA in two settings (1) balanced setting in which the SPE and SLE dimensions are always the same (2) imbalanced setting in which the dimension of SPE may be different from that of SLE. Since we varied the output dimensions of SPE and SLE from 50, 100, 300, to 500 systematically, the input dimension to MUE under the balanced setting are 100, 200, 600 and 1000. When running CCA

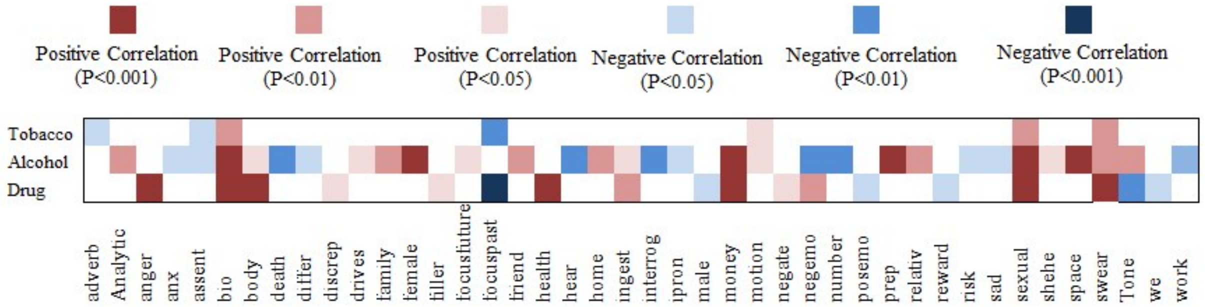


Figure 1: LIWC Features that are Most Significantly Correlated with Substance Use.

and DCCA under the imbalanced setting, we only chose the best SPE (with 50 dimensions) and the best SLE (with 300 dimensions). We also varied the number of MUE output dimensions systematically from 20,50,100,200,300,400,500 to 1000 (up to the total input MUE dimensions). We used the *LikeStatus* dataset in Table 1 as the training data for multi-view unsupervised feature learning. For MUE-based supervised SUD predication, we used the *LikeStatusSUD data*. In our experiments, we use a variant of CCA called *wGCCA* implemented by (Benton et al., 2016) where we set the weights for both views to be equal [1]. We used the DCCA implementation by (Andrew et al., 2013) which uses Keras and Theano as the deep learning platform [2]. We also varied the number of hidden layers from 1 to 3 to tune the performance.

We compared our multi-view learning results with 3 baselines: BestSPE and BestSLE are the best single view models. We also used a 3rd baseline called *Unigram_combine*, which simply concatenates all the post and like unigrams together and then applies supervised feature selection before uses the remaining features in a SVM-based classification. As shown in Table 5, both wGCCA and DCCA significantly outperformed the unigram-based baseline ($p < 0.01$ based on t-test). The difference between the best multi-view models (wGCCA_balanced for Alcohol and drug, wGCCA_imbalanced for illicit drugs) and the best single view models are also significant ($p < 0.01$). wGCCA also performed significantly better than DCCA on our tasks ($p < 0.01$ based on t-tests).

Table 5: MUE: Prediction Results

	Tobacco	Alcohol	Drug
BestSPE	0.802	0.768	0.819
BestSLE	0.787	0.795	0.791
Unigram_combine	0.685	0.669	0.662
wGCCA_balanced	0.848	**0.811**	**0.844**
wGCCA_imbalanced	**0.855**	0.799	0.832
DCCA_balanced	0.774	0.778	0.742
DCCA_imbalanced	0.760	0.781	0.737

7 Social Media and Substance Use

In addition to building models that predict SUD, we are also interested in understanding the relationship between a person's social media activities and substance use behavior. Since many of the SPEs and SLEs are not easily interpretable, in this section, we focus on the LIWC features from status updates and the LDA topics from both Likes and status updates. Since the SUD ground truth is an ordinal variable and the LIWC/LDA features are numerical, we used Spearman's rank correlation analysis to identify features that are most significantly correlated with SUD. Figure 1 shows the LIWC features that are significantly correlated with at least one type of SUD ($p < 0.05$). The color red represents a positive correlation while blue represents a negative correlation. In addition, the saturation of the color indicates the significance of the correlation. The darker the color is, the more significant the correlation is.

As shown in Figure 1, swear words such as "fuck" and "shit", sexual words such as "horny" and "sex", words related to biological process such as "blood" and "pain" are positively correlated with all three types of SUD. In addition, words related to money such as "cash", words related to body such as "hands" and "legs", words related to ingestion such as "eat" and "drink" are positively correlated with both alcohol and drug use; words related to motion such as "car" and "go" are positively correlated with both alcohol and tobacco use. In addition, female references such as "girl" and "woman", prepositions, space reference words such as "up" and "down" are positively cor-

[1] https://github.com/abenton/wgcca
[2] https://github.com/VahidooX/DeepCCA

Table 6: Topics Most Significantly Correlated with Substance Use.

	Significance	Topic
Tobacco		
Posts	+	*(T1) fuck, shit, ass, fucking, bitch, face, don't, kick, damn, man, lol, hell...*
	-	*(T2) paper, book, writing, read, class, essay, english, finished, reading, time, page ...*
Likes	+	*(T3) Tool, Misfits, A Perfect Circle, Rob Zombie ...*
	-	*(T4) The Twilight Saga, Forever 21, Twilight, Victoria's Secret, Katy Perry*
Alcohol		
Posts	+	*(T5) tonight, night, free, party, tickets, bar, saturday, friday, dj, drink, club, show, beer, ladies...*
	–	*(T6) class, history, paper, math, science, writing, essay, finished, study, test, final, exam ...*
Likes	++	*(T7) V For Vendetta, Boondock Saints, Pan's Labyrinth ...*
	–	*(T8) Cookie Monster, Squirt, Last Day of School, Hunger Games Official Page, Wonka ...*
Drug		
Posts	++	*(T9) fuck, shit, ass, fucking, bitch, face, don't, kick, damn, man, lol, hell...*
	-	*(T10) dinner, nice, shopping, christmas, home, weekend, lunch, family, house,love,wine :-)...*
Likes	+	*(T11) Radiohead,The Cure, Depeche Mode, The Smiths, Arctic Monkeys ...*
	-	*(T12) Music, Movies, Traveling, Photography, Dancing ...*

related with alcohol use, while words related to anger such as "hate" and "kill", words related to health such as "clinic" and "pill" are positively correlated with drug use.

In terms of LIWC features that are negatively correlated with SUD, words associated with the past such as "did" and "ago" are negatively correlated with both tobacco and drug use; assent words such as "ok", "yes" and "agree" are negatively correlated to both alcohol and tobacco use. In addition, male references such as "boy" and "man", words related to reward such as "prize" and "benefit", words related to positive emotions such as "nice" and "sweet", first person pronouns (plural) such as "we" and "our" are negatively correlated to drug use. Moreover, impersonal pronouns such as "it", differentiation words such as "but" and "else", and work-related words such as "job" and "work" are negatively correlated with alcohol use. Surprisingly, risk related words such as "danger", words related to sadness, death and negative emotions are also negatively correlated with alcohol use.

There are a few surprising correlations in our results. For example, female references such as "girl" and "woman" are positively related to alcohol use while male references such as "man" and "boy" are negatively related to drug use. To interpret this, previous research has shown (Schwartz et al., 2013) that female references actually are used more often by male authors and vice versa. Thus, our findings suggest that males are more likely to use alcohol while females are less likely to use illicit drugs.

We have also used Spearman's correlation analysis to identify SUD-related "Like Topics" and "Status update Topics" learned by LDA. Since the number of significant topics is quite large, in Table 6, we only show a few samples. Based on a user's status updates, "swear topics" (T1, T9) are positively correlated with both tobacco and drug use, which is consistent with our LIWC findings. The "night life topic" (T5) is positively related to alcohol use. In addition, school related topics (T2, T6) are negatively correlated with tobacco and alcohol use. Positive family-related activities (T10) are negatively correlated with drug use. In addition, based on the LDA topics learned from "likes", a preference for rock music (T3,T11) is positive correlated with tobacco and drug use. A preference for movies such as "V For Vendetta" and "Boondock Saints" (T7) is positively correlated with alcohol use, while having a hobby (T12), liking cartoons and shows favored by kids (T8) or liking movies and brands favored by girls (T4) are negatively correlated with drug, alcohol and tobacco use.

8 Discussion and Future Work

Currently, our multi-view unsupervised features learning methods only learn from the intersection of the like and status update data, which is much smaller than either the like or the status update data. Similarly, MUE-based supervised prediction used only the intersection of all three datasets which is very small (only contains 896 users).

Thus, it would be useful if a future multi-view feature learning algorithm is capable of using all the available data (e.g., the union of all the supervised and unsupervised training data). Moreover, our best SPE model only has 50 dimensions while our best SLE model has 300 dimensions. This might be because the supervised training data used by SPE is almost three times smaller than that used by SLE . But surprisingly, SPE-based models performed better than SLE-based models. We expect that with more training data, the performance of SPE-based methods can be further improved.

9 Conclusion

We believe social media is a promising platform for both studying SUD-related human behaviors as well as engaging the public for substance abuse prevention and screening. In this study, we have focused on four main tasks (1) employing unsupervised features learning to take advantage of a large amount of unsupervised social media data (2) employing multi-view feature learning to combine heterogeneous user information such as "likes" and "status updates" to learn a comprehensive user representation (3) building SUD prediction models based on learned user features (4) employing correlation analysis to obtain human-interpretable results. Our investigation has not only produced models with the state-of-the-art prediction performance (e.g., for all three types of SUD, our models achieved over 80% prediction accuracy based on AUC), but also demonstrated the benefits of incorporating unsupervised heterogeneous user data for SUD prediction.

References

Galen Andrew, Raman Arora, Jeff A Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *ICML (3)*. pages 1247–1255.

Eiji Aramaki, Sachiko Maskawa, and Mizuki Morita. 2011. Twitter catches the flu: detecting influenza epidemics using Twitter. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 1568–1576.

Adrian Benton, Raman Arora, and Mark Dredze. 2016. Learning multiview embeddings of Twitter users. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. volume 2, pages 14–19.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Gilbert J Botvin. 2000. Preventing drug abuse in schools: Social and competence enhancement approaches targeting individual-level etiologic factors. *Addictive behaviors* 25(6):887–897.

David A Brent. 1995. Risk factors for adolescent suicide and suicidal behavior: mental and substance abuse disorders, family environmental factors, and life stress. *Suicide and Life-Threatening Behavior* 25(s1):52–63.

Remi J Cadoret, Ed Troughton, Thomas W O'Gorman, and Ellen Heywood. 1986. An adoption study of genetic and environmental factors in drug abuse. *Archives of general psychiatry* 43(12):1131–1136.

S Campbell, L Henry, J Hammelman, and M Pignatore. 2014. Personality and smoking behaviour of non-smokers, previous smokers, and habitual smokers. *J Addict Research & Therapy* 5:191.

Marilyn E Carroll, Justin J Anker, and Jennifer L Perry. 2009. Modeling risk factors for nicotine and other drug abuse in the preclinical laboratory. *Drug and alcohol dependence* 104:S70–S78.

Kamalika Chaudhuri, Sham M Kakade, Karen Livescu, and Karthik Sridharan. 2009. Multi-view clustering via canonical correlation analysis. In *Proceedings of the 26th annual international conference on machine learning*. ACM, pages 129–136.

Mark Cook, Alison Young, Dean Taylor, and Anthony P Bedford. 1998. Personality correlates of alcohol consumption. *Personality and Individual Differences* 24(5):641–647.

Rosa M Crum, Marsha Lillie-Blanton, and James C Anthony. 1996. Neighborhood environment and opportunity to use cocaine and other drugs in late childhood and early adolescence. *Drug and alcohol dependence* 43(3):155–161.

Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. 2000. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications* 21(4):1253–1278.

Jean-Francois Etter, Jacques Le Houezec, and Thomas V Perneger. 2003. A self-administered questionnaire to measure dependence on cigarettes: the cigarette dependence scale. *Neuropsychopharmacology* 28(2):359.

Jennifer Golbeck, Cristina Robles, and Karen Turner. 2011. Predicting personality with social media. In *CHI'11 extended abstracts on human factors in computing systems*. ACM, pages 253–262.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural computation* 16(12):2639–2664.

G. Hinton and R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science* 313(5786):504–507.

Işil Doğa Yakut Kiliç and Shimei Pan. 2016. Analyzing and preventing bias in text-based personal trait prediction algorithms. In *Tools with Artificial Intelligence (ICTAI), 2016 IEEE 28th International Conference on*. IEEE, pages 1060–1067.

Michal Kosinski, Sandra C Matz, Samuel D Gosling, Vesselin Popov, and David Stillwell. 2015. Facebook as a research tool for the social sciences: Opportunities, challenges, ethical considerations, and practical guidelines. *American Psychologist* 70(6):543.

Michal Kosinski, David Stillwell, and Thore Graepel. 2013. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences* 110(15):5802–5805.

Abhishek Kumar and Hal Daumé. 2011. A co-training approach for multi-view spectral clustering. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. pages 393–400.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*. volume 14, pages 1188–1196.

Quoc V Le, Will Y Zou, Serena Y Yeung, and Andrew Y Ng. 2011. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, pages 3361–3368.

Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng. 2009. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*. pages 1096–1104.

Rui Li, Kin Hou Lei, Ravi Khadiwala, and Kevin Chen-Chuan Chang. 2012. Tedas: A Twitter-based event detection and analysis system. In *2012 IEEE 28th International Conference on Data Engineering*. IEEE, pages 1273–1276.

NIDA. 2015. Drugs, brains, and behavior: The science of addiction. https://www.drugabuse.gov/publications/drugs-brains-behavior-science-addiction/introduction.

ER Oetting and Fred Beauvais. 1987. Common elements in youth drug abuse: Peer clusters and other psychosocial factors. *Journal of Drug Issues* 17(2):133–151.

James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of liwc2015. Technical report.

SAMHSA. 2015. Substance use disorders. https://www.samhsa.gov/disorders/substance-use.

Mehmet Emre Sargin, Engin Erzin, Yücel Yemez, and A Murat Tekalp. 2006. Multimodal speaker identification using canonical correlation analysis. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. IEEE, volume 1, pages I–I.

Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. 2009. Event detection and tracking in social streams. In *ICWSM*.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, et al. 2013. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS one* 8(9):e73791.

Abhishek Sharma, Abhishek Kumar, Hal Daume, and David W Jacobs. 2012. Generalized multiview analysis: A discriminative latent space. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, pages 2160–2167.

David J Stillwell and Richard J Tunney. 2012. Effects of measurement methods on the relationship between smoking and delay reward discounting. *Addiction* 107(5):1003–1012.

Antonio Terracciano, Corinna E Löckenhoff, Rosa M Crum, O Joseph Bienvenu, and Paul T Costa. 2008. Five-factor model personality profiles of drug users. *Bmc Psychiatry* 8(1):22.

Julia M Townshend and Theodora Duka. 2005. Binge drinking, cognitive performance and mood in a population of young social drinkers. *Alcoholism: Clinical and Experimental Research* 29(3):317–325.

Svitlana Volkova and Yoram Bachrach. 2015. On predicting sociodemographic traits and emotions from communications in social networks and their implications to online self-disclosure. *Cyberpsychology, Behavior, and Social Networking* 18(12):726–736.

Paul Willner. 2000. Further validation and development of a screening instrument for the assessment of substance misuse in adolescents. *Addiction* 95(11):1691–1698.

Chao Yang, Shimei Pan, Jalal Mahmud, Huahai Yang, and Padmini Srinivasan. 2015. Using personal traits for brand preference prediction. In *EMNLP*. pages 86–96.

Wu Youyou, Michal Kosinski, and David Stillwell. 2015. Computer-based personality judgments are more accurate than those made by humans. *Proceedings of the National Academy of Sciences* 112(4):1036–1040.

Yiheng Zhou, Numair Sani, Chia-Kuei Lee, and Jiebo Luo. 2016. Understanding illicit drug use behaviors by mining social media. *arXiv preprint arXiv:1604.07096* .

Demographic-Aware Word Associations

Aparna Garimella, Carmen Banea, and Rada Mihalcea
University of Michigan
Ann Arbor, MI
{gaparna,carmennb,mihalcea}@umich.edu

Abstract

Variations of word associations across different groups of people can provide insights into people's psychologies and their world views. To capture these variations, we introduce the task of demographic-aware word associations. We build a new gold standard dataset consisting of word association responses for approximately 300 stimulus words, collected from more than 800 respondents of different gender (male/female) and from different locations (India/United States), and show that there are significant variations in the word associations made by these groups. We also introduce a new demographic-aware word association model based on a neural net skip-gram architecture, and show how computational methods for measuring word associations that specifically account for writer demographics can outperform generic methods that are agnostic to such information.

1 Introduction

Understanding the associations that are formed in the mind is paramount to understanding the way humans acquire language throughout a lifetime of learning (Elman et al., 1997; Rogers and McClelland, 2004). Furthermore, word associations are believed to mirror the mental model of the conceptual connections in a human mind, and constitute a direct path to assessing one's semantic knowledge (Nelson et al., 2004; Mollin, 2009).

Word associations start forming early in life, as language is acquired and one learns based on the environment where concepts lie in relation to each other. For example, we may learn to associate "mother" with "warmth," or "fire" with "burn." Yet, this mental model is not static but highly dynamic, and is shaped by new experiences over a lifetime. For instance, (Tresselt and Mayzner, 1964) showed that word associations change with time, and that for respondents in younger age groups their variability is lower, while for those in older age groups the variability is higher, as their life experiences modify the commonality between respondents from the same group.

Computational linguistics has traditionally taken the "one-size-fits-all" approach, with most models being agnostic to the language of the speakers behind the language. With the introduction and adoption of Web 2.0, there has been an exponential increase in the availability of digital user-centric data in the form of blogs, microblogs and other forms of online participation. Such data often times can be augmented with demographic or other user-focused attributes, whether these are user-provided (e.g., from a user's online profile) or labeled using an automatic system. This enables computational linguists to go beyond generic corpus-based metrics of word associations, and attempt to extract associations that pertain to given demographic groups that would not have been possible without administering time consuming and resource intensive word association surveys.

While current NLP methods generally deal with more advanced tasks (relation extraction, text similarity, etc.), at their very core many of these tasks assume some way of drawing connections (or associations) between words. Therefore, as a step toward demographic-aware NLP, we choose to work on the core task of "word association." The algorithms we introduce can be immediately applied to demographic-aware word similarity, and with some minor changes to demographic-aware text similarity. Future stages could also include demographic-aware labeled associations, and more advanced applications such as information retrieval (which relies heavily on word associations/similarity), demographic-aware keyword extraction, dialogue personalization, and so forth.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2285–2295
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Note that a few other researchers have explored demographic-aware NLP models with promising results, primarily focusing on the use of demographics for various forms of text classification (Hovy, 2015) or sentiment and subjectivity classification (Volkova et al., 2013).

The paper makes several main contributions. First, we create a novel dataset of demographic-aware word associations, consisting of approximately 300 stimulus words along with 800 responses per word collected from a demographically-diverse group of respondents, for a total of 228,800 responses. Removing spam responses resulted in 176,097 responses. Analyses that we perform on this dataset demonstrate that indeed word associations vary across user dimensions.[1] Second, we show that the associations we obtained follow the same pattern as those elicited during traditional classroom surveys. Third, we propose an evaluation metric suited for the free association norms task. Fourth, we introduce a demographic-aware model based on a skip-gram architecture and through several comparative experiments, we show that we are able to surpass the performance attainable on demographic agnostic models.

We specifically focus on two demographic dimensions: location and gender. For location, we consider India and United States (US), choice made primarily because these two countries have a large English-speaking population, represented both on social media and on crowdsourcing platforms.

2 Related Work

Word associations have captured the attention of psychologists since at least the early 1900. In (1910), Kent and Rosanoff proposed the use of a set of 100 emotionally neutral words for word associations surveys. A psycholinguistics study that looked at the impact that the nationality of respondents may have on formed word associations was carried out by Rosenzweig (1961), employing the stimulus word list proposed by Kent and Rosanoff (1910) manually translated into several West European languages. Based on the primary responses coming from native speakers of English, French, German and Italian, which were mapped back into English, the author concludes that the associations formed by speakers of the four languages are very similar, with "almost half the comparisons in any pair of languages yielding agreements," where the most frequent responses are encountered across pairs of languages. Given that the primary responses were compared across languages and people with a relatively common origin (West European), our work seeks to investigate whether similar results are encountered when looking at different locations (namely US versus India). Furthermore, our study is conducted in English from the beginning, to eliminate a third party's subjectivity in mapping primary responses from one language to another.

There have also been attempts in computational linguistics to derive associations not based on survey results (which are static and resource intensive), but based on statistics derived from large corpora (Church et al., 1989; Wettler and Rapp, 1989; Church and Hanks, 1990). Research in semantic similarity can also be used to model associations based on several directions: (1) co-occurrence metrics that rely on large corpora such as PMI (Church and Hanks, 1990), second order PMI (Islam and Inkpen, 2008), or Dice (Dice, 1945); (2) distributional similarity-based measures, that characterize a word by its surrounding context such as LSA (Landauer and Dumais, 1997), ESA (Gabrilovich and Markovitch, 2007), or SSA (Hassan and Mihalcea, 2011); and (3) knowledge-based metrics that rely on resources such as lexica or thesauri (Leacock and Chodorow, 1998; Lesk, 1986; Jarmasz and Szpakowics, 2003; Hughes and Ramag, 2007). However, most of these metrics have so far been applied to model the relatedness between two words, namely given a word pair, to score how similar the two words are; as such, they have not been used to predict free association norms, namely given a word, to attempt to determine the most likely word that a human would associate with that stimulus.

Large word association databases exist, such as the one collected by Deyne et al. (2013), who used a set of 12,000 stimulus words and surveyed 70,000 participants. Yet to our knowledge, no concerted attempt has been made to gather word associations jointly with the demographic characteristics of the people behind them.

While not directly seeking to extract word associations but rather trying to represent language meaning through a locality lens, (Bamman et al., 2014) have proposed using distributed representations to model words employed by social media

[1] This work is not centered around comparing different word forms, as one would encounter for example in British English and American English, but rather around different word associations that people with a particular demographic characteristic are inclined to make, e.g., "health" in India is more strongly associated with "wealth", while in the United States it is more strongly associated with "sick."

users from different US states. They were able to show that the regional meaning of words can successfully be carried by word embeddings, for example the word “wicked” was most similar to the word “evil” in Kansas, while in Massachusetts, it was most similar to “super” (based on the cosine similarity of the words’ vectorial representation). In contrast, our rationale in this article is to explore if word associations can be automatically derived from large corpora annotated with user-centered attributes such as location or gender.

3 Word Associations Dataset

Word association data collection typically consists of providing participants with a list of words, also known in the psycholinguistics literature as *stimulus words*, and asking them to provide the first word that comes to mind in response to each stimulus. For instance, given a stimulus word such as *cat*, one would expect answers such as *dog* or *mouse*. Earlier work on word associations administered the tests in classroom settings, with 100 words per survey, and the results were compiled into tables of norms of word associations (Kent and Rosanoff, 1910; Nelson et al., 2004).

Since our goal is to explore the effect of demographics on word associations, we created a task on Amazon Mechanical Turk (AMT) able to reach a wide and demographically diverse audience. The survey was structured into two sections: the word association part, followed by a demographic survey. Given the online nature of the survey, and since we aimed for a high quality dataset, each participant was presented with a set of 50 stimulus words at a time (instead of 100). The demographic section consisted of seven questions covering gender, age, location, occupation, ethnicity, education, and income.

Stimuli. The stimulus list consists of a set of approximately 300 words. Among these, 99 words are sourced from the word list proposed by Kent and Rosanoff (1910) (*standard* list).[2] The remaining words are identified using the method for finding word-usage differences between two groups introduced in (Garimella et al., 2016), which relies on large collections of texts authored by the two groups to identify words that can be accurately classified by an automatic classifier as belonging to one group versus another. Using their method, we obtain 100 words as the top most different words between US and India (*culture* list), and another set of 100 words as the top most different words between male and female (*gender* list). The reunion of these three lists results in 286 stimulus words for which we collect word associations. Examples are shown in Table 1.

Responses. The task was published separately for respondents from US and India, as AMT has an option of only presenting the survey to people from a preselected geographical location. Six different surveys, each including approximately 50 stimulus words, were administered for each region. The survey was conducted in English for both countries, noting that one of the official languages of India is English (alongside Hindi). Each survey also included four spam-checking questions with previously known answers (e.g., *What is the color of the sky?*, with five options *blue*, *red*, *pink*, *green*, *yellow*), which were used to filter out respondents who were filling out the survey without reading the questions.

For each set, we gathered 400 responses per region, resulting in 800 responses for both US and India. After removing the respondents who did not pass the spam-checking questions, we were left with an average of 752 responses per word, which we then balanced by gender, to retain an equal number of Indian women, Indian men, US women, and US men. This resulted in 492 and 480 responses for the two sets of 50 *standard* stimulus words, 436 and 468 for the *culture* words, and 440 and 432 for the *gender* words. Similar to (Rosenzweig, 1961), all the responses were normalized (i.e. plural was mapped to singular, gerund to infinitive, etc.); in our case we used the Stanford CoreNLP Lemmatizer (Manning et al., 2014), ultimately aggregating the responses into a gold standard.

Table 1 shows the top associations for a few sample stimuli, as collected from India and US, and males and females. Finer-grained qualitative analyses also reveal interesting distinctions. For instance *bath* is overwhelmingly associated by men with *water*, while US women associate it with *bubble*, and Indian women with *soap*. Interestingly, US men seem to provide responses based on collocations, e.g., they answer *Kane* for *citizen* (citizen Kane), *weight* for *heavy* (heavyweight), or *lion* for *mountain* (mountain lion); on the contrary, women more often provide responses that consist of synonym or antonym words, e.g., *person* for *citizen*, *health* for *sick*, or *light* for *heavy*.

For further insight, Table 2 shows the average

[2] Note that this list originally included 100 words. The word “foot” was however misspelled in our survey, and instead we gathered answers for “food.”

	Gender		Location	
Word	Male	Female	India	US
beautiful	girl, woman, pretty	pretty, girl, ugly	girl, nature, flower	pretty, girl, ugly
cheese	pizza, bread, milk	butter, mouse, pizza	pizza, butter, bread	cracker, swiss, cheddar
hard	soft, rock, work	soft, work, rock	work, stone, rock	soft, rock, time
health	good, wealth, care	good, wealth, sick	wealth, good, fitness	good, sick, care
range	distance, gun, shooting	gun, rover, mountain	price, rover, wide	gun, distance, rover
admit	hospital, guilt, card	hospital, confess, one	hospital, card, accept	guilt, one, confess
mix	tape, match, juice	cake, tape, stir	juice, tape, match	stir, tape, cake
organize	clean, arrange, party	clean, arrange, meeting	arrange, meeting, party	clean, sort, neat
stack	pile, book, box	book, pile, hay	book, queue, pile	pile, book, pancake

Table 1: Top three most frequent responses for sample stimulus words.

number of different responses obtained for a given stimulus word, with the lowest variability word, and the highest variability word.[3] The second column lists the correlations between the frequency of the primary response and the number of different responses, as also reported by (Jenkins and Palermo, 1965). This correlation is negative, as the more people agree on the primary response, the fewer overall unique answers for a stimulus word are provided. Additionally, Figure 1 shows the Zipfian distribution of average norm frequency; the most frequent response is given on average by 24% of the respondents, while the third most frequent response is given by 7% of them.

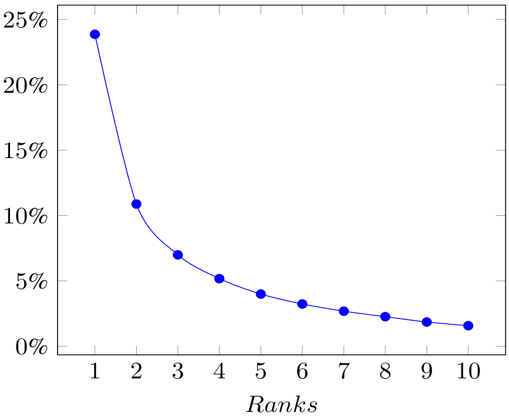


Figure 1: Primary response frequency (in percent) versus rank for the Standard word list.

Demographic	Avg.	Correlation	Lowest Variability	Highest Variability
Standard				
India	60.88	-0.52	stove	city
US	51.19	-0.53	bath	trouble
Male	61.63	-0.45	stove	city
Female	56.75	-0.55	stove	city
All				
India	72.27	-0.59	stove	regardless
US	57.03	-0.56	east	basically
Male	70.33	-0.52	stove	regardless
Female	66.54	-0.59	east	respectively

Table 2: Average number of responses obtained for a given stimulus word, correlation between frequency of primary response and number of different responses, words exhibiting the lowest variability, and words with the highest variability.

Analyses of Demographic Variations. To model norm strength within a given demographic group or across groups, we tabulate how often respondents from a group match the most frequent answer (*Primary*) or one of the most frequent ten answers for that group (*Top10*). That is, given the response for one stimulus word as provided by one held-out survey respondent at a time, we determine whether that response matches the most frequent association of the *remaining* members of the same group (Table 3, *Primary* columns), or one of the top 10 associations pertaining to that same group (Table 3, *Top10* columns). Similarly, we measure the match with the most frequent or the top 10 responses from the other group, as shown in Table 4. As expected, the intra-group similarities are significantly higher than the inter-group similarities, which supports our hypothesis that different groups make different word associations, which tend to be more coherent within a group than across groups. While males and females have similar ranges for their agreement figures, we notice that on average US respondents have stronger intra-group agreements. Note also that inter-group similarities are asymmetrical, as multiple words may have the same association frequency for one group, yet for the complementary group that may not be the case.

[3] In several of our data analyses, in order to allow for a direct comparison with the word list from (Kent and Rosanoff, 1910), in addition to showing statistics for the entire dataset (*All*), we also show statistics separately compiled for the list from (Kent and Rosanoff, 1910) (*Standard*).

As an additional analysis of demographic variations in the responses received, for each respondent, we predict his / her demographic group using a majority vote conducted across all the user's responses using a simple rule-based system that assigns each response to the group having the highest frequency for that particular association. For instance, given the response *sun* obtained from a respondent for the stimulus *yellow*, we assign the respondent to either India or US depending on the highest normalized frequency of the response *sun* for the same stimulus in each of those groups. A similar rule-based assignment is also used for gender. Thus, we compute the response words and their normalized frequencies based on the responses from 80% of the users chosen randomly, and accordingly predict the demographic group for the remaining 20% of the users based on a decision across the entire set of a user's responses. Table 5 shows the results of these predictions, which indicate high location variability (i.e., we can predict with high accuracy the location of a respondent), and medium gender variability.

Demo-	Standard		All	
graphic	Primary	Top10	Primary	Top10
India-India	0.23	0.77	0.18	0.78
US-US	0.29	0.82	0.25	0.81
Male-Male	0.23	0.79	0.19	0.79
Female-Female	0.25	0.80	0.21	0.81

Table 3: Intra-group similarities (the higher the similarity, the more cohesive the group is).

Demo-	Standard		All	
graphic	Primary	Top10	Primary	Top10
India-US	0.18	0.55	0.14	0.50
US-India	0.20	0.60	0.16	0.56
Male-Female	0.22	0.63	0.17	0.59
Female-Male	0.24	0.66	0.19	0.61

Table 4: Inter-group similarities (the higher the similarity, the less distinct the groups are).

Demographic	Standard	All
Gender	0.60	0.56
Location	0.94	0.94

Table 5: Predictions based on similarity to group.

4 Computational Models of Word Associations

We first introduce a new model for measuring word associations that leverages a shallow neural net architecture to embed demographically-enriched words. We then compare the performance of the predicted associations to those resulting from other approaches, including traditional corpus-based measures such as mutual information or vector-space models, as well as a recent distributed learning model with word embeddings. For each of these methods, we predict, evaluate, and compare generic associations (devoid of any demographic information), as well as demographic-aware associations.

4.1 Composite Skip-gram Models

We introduce a new word association model, which relies on the skip-gram neural net architecture (Mikolov et al., 2013), and leverages its efficiency and ability to deal with less frequent words.

The skip-gram model tries to predict the context given a word, that is, for each word w_i in the input sequence $w_1, \ldots, w_T$, the model tries to predict w_{i-2}, w_{i-1}, w_{i+1} and w_{i+2}, assuming, for example, a sliding window of five words. Mathematically, the model maximizes the objective function

$$J = \frac{1}{T} \sum_{i=1}^{T} \sum_{j=-c, j \neq 0}^{c} \log P(w_{i+j}|w_i) \qquad (1)$$

where T is the number of tokens in the data set, c is the number of context words on each side of the target word w_i and $P(w_{i+j}|w_i)$ is the probability to observe word w_{i+j} in the context of word w_i.

To make this model demographic-aware, we propose two variations, which we refer to as composite skip-gram models ($C - SGM$). In the first one ($EMB1$), the target word w_i is tagged with a demographic label L (e.g., gender). For example, for the target word "formula$^{L=\text{female}}$" we try to predict a high probability for "baby" and "milk" occurring in the neighboring context. The underlying reasoning is that tagged words that appear in similar contexts will be nudged toward each other, while those that do not, will further distance themselves. This allows discrepancies to emerge between how the words are embedded given a demographic dimension.

In the second variation ($EMB2$), we also include the demographic label in the context. That is, for each skip-gram $(c_{i,\text{left}}, w_i, c_{i,\text{right}})$ we generate three skip-grams

$$\begin{array}{lll} (c^{\text{label}}_{i,\text{left}}, & w_i, & c_{i,\text{right}}) \\ (c_{i,\text{left}}, & w^{\text{label}}_i, & c_{i,\text{right}}) \\ (c_{i,\text{left}}, & w_i, & c^{\text{label}}_{i,\text{right}}) \end{array} \qquad (2)$$

The two models seek to capture different scenarios. In the first model, where we only add the demographic label to the target word, the embedding of the labeled word is optimized with respect to the generic embedding of the context. In the second model, the optimization is rather symmetric, allowing tagged and generic embeddings to influence each other. Thus, the optimization function seeks to predict both tagged and untagged words in the vicinity given a target word, instead of only focusing on predicting untagged words like EMB1. The embeddings resulting from such a model should allow for more accurate representations across the tagged and untagged vocabulary, where for example the word "mother" uttered by a female would be close to the word "mother" (regardless of author gender). In both scenarios, the embeddings space accommodates both tagged and untagged words at the same time, being very computationally robust, and allowing comparisons across the tagged version of words, as well as between generic words and their tagged surrogates. For both variations, we compute the cosine similarity between the stimulus word and each of the vocabulary words (whether generic or demographic-enhanced), and retain the closest unique candidates (after dropping their demographic tag).

4.2 Other Word Association Models

Mutual Information (MI). We implement the information theoretic measure proposed by Church and Hill (1990). It is defined as follows:

$$I(x, y) = log_2 \frac{P(x, y)}{P(x)P(y)} \tag{3}$$

This measure compares the probability of observing words x and y together (the joint probability) with the probabilities of observing x and y independently. The joint probability, $P(x, y)$, is generally estimated by counting the number of times x is followed by y in a window of w words, and normalizing this count with the size of the corpus. We follow Church and Hill and set the window size w to five, as it is large enough to capture verb-argument constraints, and not so large to restrict to strict adjacency. For a given stimulus word, (1) we use the entire corpus and compute the generic MI word association with the rest of the vocabulary, and get the top associations according to their MI scores; and (2) we use the section of the corpus obtained for a given demographic, and determine the top demographic-aware MI word associations.

Vector-Space Model (VSM). We also implement the traditional vector-space model, where each word is represented by a $tf.idf$ weighted vector inside the term-document matrix (representing term occurrences inside the documents in the corpus), with a length equal to the number of documents D in the corpus (Salton and McGill, 1986). For a given stimulus word, cosine similarities are computed with all the remaining word vectors in the vocabulary, and those words having the highest similarity are considered as the top responses. Similar to MI, we use all the documents in the corpus to produce generic word associations, while only those documents pertaining to a specific demographic value are utilized to derive demographic-aware associations.

Skip-gram Language Model. We also use the distributional representation technique of word embeddings ($SGLM$) proposed by Bamman et al. (2014). Specifically, information about the speaker (geography, in their case) is used while learning the vector-space representations of word meanings from textual data that is supplemented with metadata about the authors. In addition to the *global* embedding matrix W_{main} that contains low-dimensional representations for every word in the vocabulary (Mikolov et al., 2013), this approach has an additional $|C|$ matrices $\{W_c\}$ of the same size as W_{main}, where $|C|$ denotes the number of values the demographic variable has in the data (e.g., if gender is the demographic variable, $C = \{female, male\}$ and $|C| = 2$). Each of these $|C|$ matrices captures the effect that each demographic variable value has on each word in the vocabulary. To index the embedding of a stimulus word $w \in \mathbb{R}^{|V| \times k}$, the hidden layer h is computed as the sum of the matrix multiplications with each of the independent embeddings:

$$h = w^T W_{main} + \Sigma_{c \in C} w^T W_c \tag{4}$$

It then predicts the value of the context word y using another parameter matrix $X \in \mathbb{R}^{|V| \times k}$ based on a softmax function $o = softmax(Xh)$, where $o \in \mathbb{R}^{|V| \times k}$. Backpropagation using (input x, output y) word tuples learns the values of the various embedding matrices W and parameter matrix X, which maximize the likelihood of context words y conditioned on the stimulus word x.

We use this approach in its original implementation provided by (Bamman et al., 2014) to compute the word embedding vectors for all the words in the vocabulary. Given a stimulus word, the closest vocabulary words with the highest cosine similarity are retained as the top association predictions for the given stimulus word.

5 Experiments

All our models require textual data with demographic information. We introduce below the data we used and the metrics we adopted for evaluation.

Data. Given the requirement of having gender and location information associated with the data, we resort to blogs, and collect from Google Blogger[4] a large set of blog posts authored between 1999 and 2016. Table 6 shows the breakdown of the raw blog counts per demographic category. From these, we retain only those posts with non-empty content, and preprocess the data by removing HTML tags, converting all the tokens to their lemmatized forms,[5] and discarding those lemmas with a frequency less than 10, in order to avoid misspellings and other noise characteristic to social media content.

Demo-	Raw		Balanced		
graphic	Profiles	Posts	Profiles	Posts	Tokens
India	1,520	339,624	1,520	34,987	16,884K
US	3,273	825,093	1,520	32,782	11,706K
Male	2,031	597,935	1,818	44,299	21,971K
Female	1,818	321,779	1,818	45,980	17,070K

Table 6: Raw and balanced blog dataset statistics.

From the above pool of blog posts, we create two datasets with complementary demographic classes (1) location: India-US and (2) gender: male-female.

We process each of these datasets so that they are profile-balanced with no peaks for any specific years, by applying several heuristics: **(1)** Compute the minimum number of users n over all the classes (e.g., Indian and US authors in the case of the location dataset). **(2)** From each class, select the top n users based on the number of years they were blogging and the number of posts they wrote.[6] This ensures that the maximum amount of data will be available for the selected users. **(3)** For each of these n users, pick at most 50 posts in a round-robin fashion from the years in which they blogged. **(4)** Let M be the total number of posts collected in this manner from all the classes. In order to avoid having most of the posts coming from a small number of years, set a cutoff X as a fraction of M. For each year, a maximum of X posts will be chosen from the set of M posts ($X = 0.1M$). **(5)** To ensure that all the users get to contribute posts, and that the contribution of prolific writers is kept in check, maintain user participation scores:

$$p(user) = \frac{\text{posts collected from user}}{\text{total number of posts collected}} \quad (5)$$

These scores are updated after every year is processed, as explained further. **(6)** Sort the years in increasing order of number of posts and iterate through them; identify the lowest number of posts contributed by the least prolific writer, then collect the minimum number of posts from all users who published in that year in a round-robin manner. Then, select additional posts from users in increasing order of participation scores, until the number of posts for the year reaches the cutoff X. **(7)** After each year, update the user participation scores. Table 6 shows the number of users and posts retained after balancing. This particular composition is used in our *location* data set (consisting of India and US posts) and *gender* data set (consisting of females and males posts).

Metrics. Given that the word association task is relatively similar to the lexical substitution task, in terms of open vocabulary and lack of a "right" answer, we decided to borrow the *best* and out-of-ten (*oo10*) evaluation metrics traditionally used for the latter (McCarthy and Navigli, 2009), yet corrected for weight (Jabbari et al., 2010). Briefly, these measures take the best (or top ten) responses from a system, and compare them against the gold standard, while accounting for the frequencies of the responses in the gold standard. In addition, since Figure 1 shows that the top three ranking norms are provided as answers by approximately 42% of the respondents, with the remaining norms following a long Zipfian distribution in terms of frequency of appearance, we also compute out-of-three (*oo3*), which represents a more focused approximation of our ability to predict human associations (note that out-of-ten covers 62% of the responses). Several recent papers on word associations evaluated their models indirectly via Pearson or Spearman correlation performance on a word similarity task (Chaudhari et al., 2011; Deyne et al., 2016); we choose instead to evaluate word associations directly, by using metrics that more closely align with the evaluations performed in the field of psychology where the best output of a system is compared against the most frequent human response (Bel-Enguix, 2014; Mohammad, 2011).

For a given stimulus word w with human responses H_w, suppose a system returns a set of answers S_w. We estimate how well this system

[4] www.blogger.com

[5] We normalize the word forms using the Stanford CoreNLP lemmatizer (Manning et al., 2014).

[6] Prolific users will be chosen first. For a class with exactly n users, all users will be chosen.

can find a *best* substitute for w using Equation 6, where the function $freq_w(s)$ returns the count of a system response s in H_w, and $maxfreq_w$ returns the maximum count of any response in H_w.

$$best(w) = \frac{\Sigma_{s \in S_w} freq_w(s)}{maxfreq_w \times |S_w|} \qquad (6)$$

$$oo\mathbf{n}(w) = \frac{\Sigma_{s \in S_w^n} freq_w(s)}{|H_w|} \qquad (7)$$

Equation 7 measures the coverage of a system by allowing it to offer a set S_w^n of n responses for w, where each response s is weighted by its frequency $freq_w(s)$ in H_w.

6 Evaluations and Discussions

We conduct evaluations using all the word association models described in Section 4. The results using the *best*, *out-of-three*, and *out-of-ten* evaluation metrics are listed in Table 7. For all the embeddings experiments, we use 300 latent dimensions. The Gen variation uses the demographic-blind dataset, whereas the DA variation uses the demographic-aware dataset.[7]

The MI and VSM models do not perform well in the word association prediction task, whether considering the generic or the demographic-aware data. We should emphasize, however, that the generic version of these models is able to consider co-occurrences across the entire generic datasets, while the demographic-aware co-occurrences can only be computed from the section of the dataset that matches a particular demographic; as such, these latter models are placed at a disadvantage.

Perhaps not surprisingly, the neural network skip-gram-based architectures, whether SGLM or our C-SGM, always achieve better results when compared to MI or VSM. The demographic-aware variation proposed by (Bamman et al., 2014) uses an extended skip-gram architecture that encodes a generic embedding, and several demographic-based filters per class, which in our case translates into three matrices of 300 dimensions each, the first for the generic words, and the subsequent ones for skews to be applied to the generic words in order to render the embedding through the lens of a given demographic. $SGLM - Gen$ in our case are the predictions based on the generic matrix, while $SGLM - DA$ are the predictions modified along the lines of a particular demographic.

Our composite skip-gram models encode a single matrix that contains a mix of demographic-aware and generic words expressed as 300 latent dimensions. For both gender and location, our gender-aware models ($EMB1$ and $EMB2$) surpass the SGLM gender-aware model. Surprisingly, while SGLM was never meant to be generic, the predictions based on its generic embedding matrix prove to be a difficult baseline to surpass, similar to C-SGM generic. Nonetheless, the composite skip-gram models ($EMB1$ and $EMB2$) do achieve best and second best rankings in the vast majority of cases (when compared to the best among all the other methods), with $EMB1$ being the more robust variation performing well both for gender and for location. Focusing on the performance of $EMB1$, the highest gains are observed for India-based predictions, for best (from 0.05 to 0.08) and out-of-three (from 0.07 to 0.12); for male-based predictions increasing from 0.11 to 0.13 for best, and from 0.17 to 0.20 for out-of-three; and for female-based predictions, increasing from 0.13 to 0.14 for best, and from 0.17 to 0.20 for out-of-three. US-based associations are the hardest to predict, probably because of the diverse makeup of society; additional evaluations are needed to pinpoint the exact cause.

To determine how susceptible the embedding model is to skewed, but larger training data, we also run a separate experiment on the entire raw set of blogs we collected (described on the left of Table 6), where we re-generate the $EMB1$ and $EMB2$ models. While the entire dataset is significantly larger than the balanced set, it is also significantly skewed: the data in the India:US dataset was skewed in a proportion of 1:0.48 tokens, while for Female:Male the proportion was 1:0.41 tokens. As was the case for the balanced dataset, the $EMB1$ model is still the most robust (see the bottom section in Table 7), and it achieves significant gains when compared to its balanced counterpart, in particular for *best* (for the US demographic from 0.03 to 0.13, and for India from 0.08 to 0.11), and for *out-of-three* (for US from 0.07 to 0.15, and for India from 0.12 to 0.17), which suggests that as an avenue for future research, we can explore the use of significantly larger even if unbalanced datasets to train our models.

7 Conclusion

In this paper, we introduced the task of demographic-aware word associations. To understand the various ways in which people associate words, we collected a new large demographics-

[7] To place the results in this table in perspective, it is important to note that results for this task are traditionally low. Given that the most frequent response is selected on average by 24% of respondents (see Figure 1), we can see that even for humans, the highest score would be around 0.24.

Method	Type	best		oo3		oo10		best		oo3		oo10	
		IN	US	IN	US	IN	US	M	F	M	F	M	F
MI	Gen	0.00	0.00	0.01	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	DA	0.00	0.00	0.01	0.00	0.02	0.02	0.00	0.01	0.01	0.01	0.01	0.02
VSM	Gen	0.00	0.00	0.01	0.01	0.03	0.03	0.00	0.00	0.02	0.02	0.05	0.05
	DA	0.00	0.00	0.02	0.01	0.04	0.02	0.00	0.01	0.02	0.01	0.04	0.06
SGLM	Gen	0.02	0.02	0.03	0.03	0.06	0.05	**0.13**	0.13	**0.18**	*0.18*	0.20	0.21
	DA	0.05	0.01	0.07	0.02	0.11	0.03	0.10	0.13	0.16	*0.18*	0.18	0.20
C-SGM	Gen	0.05	**0.04**	0.07	**0.07**	0.11	**0.10**	*0.11*	0.13	*0.17*	0.17	0.20	0.21
	EMB1	*0.08*	*0.03*	*0.12*	**0.07**	*0.18*	**0.10**	**0.13**	*0.14*	**0.20**	**0.20**	**0.25**	**0.26**
	EMB2	**0.09**	0.02	**0.14**	*0.04*	**0.19**	*0.06*	0.10	**0.16**	*0.17*	**0.20**	*0.23*	*0.25*
C-SGM-raw	EMB1	0.11	0.13	0.17	0.15	0.21	0.17	0.09	0.16	0.17	0.18	0.21	0.23
	EMB2	0.10	0.08	0.15	0.12	0.19	0.15	0.09	0.14	0.15	0.16	0.18	0.20

Table 7: Best, out-of-three (oo3), and out-of-ten (oo10) scores across the various methods. IN: India, US: United States, M: Male, F: Female. The numbers in bold mark the highest scores, those in italics, the second highest.

enhanced dataset of approximately 300 stimulus words and their associated norms compiled from 800 respondents for a total of 176,097 non-spam responses, and show that for people of different demographics, associations do differ with gender and location.

We proposed a new demographic-aware word association method based on composite skip-gram models that are able to jointly embed generic and gender tagged words. We showed that this method improves over its generic counterpart, and also outperforms previously proposed models of word association, thus demonstrating that it is useful to account for the demographics of the people behind the language when performing the task of automatic word association. We regard this as a first step toward demographic-aware NLP, and in future work we plan to address other more advanced NLP tasks while accounting for demographics.

The word association dataset introduced in this paper is publicly available from http://lit.eecs.umich.edu/downloads.html.

Acknowledgements

This material is based in part upon work supported by the Michigan Institute for Data Science, by the National Science Foundation (grant #1344257), and by the John Templeton Foundation (grant #48503). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the Michigan Institute for Data Science, the National Science Foundation, or the John Templeton Foundation.

References

David Bamman, Chris Dyer, and Noah A. Smith. 2014. Distributed representations of geographically situated language. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (ACL 2014)*. pages 828–834.

Gemma Bel-Enguix. 2014. Retrieving word associations with a simple neighborhood algorithm in a graph-based resource. In *Proceedings of the 4th Workshop on Cognitive Aspects of the Lexicon*. Dublin, Ireland, pages 60–63.

Dipak L. Chaudhari, Om P. Damani, and Srivatsan Laxman. 2011. Lexical co-occurrence, statistical significance, and word association. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*. Edinburgh, Scotland, UK, pages 1058–1068.

Kenneth Church, William Gale, Patrick Hanks, and Donald Hindle. 1989. Parsing, word associations and typical predicate-argument relations. In *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, pages 75–81.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics* 16(1):22–29.

Simon De Deyne, Daniel J. Navarro, and Gert Storms. 2013. Better explanations of lexical and semantic cognition using networks derived from continued rather than single-word associations. *Behavior Research Methods* 45(2):480–498.

Simon De Deyne, Amy Perfors, and Daniel J Navarro. 2016. Predicting human similarity judgments with distributional models: The value of word associations. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers (COLING 2016)*. Osaka, Japan, pages 1861–1870.

Lee R. Dice. 1945. Measures of the amount of ecologic association between species. *Ecology* 26(3):297–302.

Jeffrey L. Elman, Elizabeth A. Bates, Mark H. Johnson, Annette Karmiloff-Smith, Domenico Parisi, and Kim Plunkett. 1997. *Rethinking innateness: a connectionist perspective on development*. The MIT Press.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proceedings of the 20th AAAI International Conference on Artificial Intelligence (AAAI 2007)*. Hyderabad, India, pages 1606–1611.

Aparna Garimella, Rada Mihalcea, and James Pennebaker. 2016. Identifying cross-cultural differences in word usage. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers (COLING 2016)*. Osaka, Japan, pages 674–683.

Samer Hassan and Rada Mihalcea. 2011. Measuring semantic relatedness using salient encyclopedic concepts. *Artificial Intelligence, Special Issue* xx(xx).

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL 2015)*. Beijing, China, pages 752–762.

Thad Hughes and Daniel Ramag. 2007. Lexical semantic relatedness with random graph walks. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP 2007)*. Association for Computational Linguistics, Prague, Czech Republic, pages 581–589.

Aminul Islam and Diana Inkpen. 2008. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Transactions on Knowledge Discovery from Data* 2(2):10:1–10:25.

Sanaz Jabbari, Mark Hepple, and Louise Guthrie. 2010. Evaluation metrics for the lexical substitution task. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2010)*. pages 289–292.

Mario Jarmasz and Stan Szpakowics. 2003. Rogets thesaurus and semantic similarity. In *Proceedings of Recent Advances in Natural Language Processing (RANLP 2003)*. Borovetz, Bulgaria, pages 111–120.

James J. Jenkins and David S. Palermo. 1965. Further data on changes in word-association norms. *Journal of Personality and Social Psychology* 1(4):303–309.

Grace H. Kent and Aaron J. Rosanoff. 1910. A study of association in insanity. *American Journal of Psychiatry* 67(1):37–96.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review* 104(2):211–240.

Claudia Leacock and Martin Chodorow. 1998. Combining local context and WordNet similarity for word sense identification. In *WordNet: An Electronic Lexical Database*, pages 305–332.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries. In *Proceedings of the 5th annual international conference on Systems documentation (SIGDOC 1986)*. Toronto, Ontario, pages 24–26.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics System Demonstrations (ACL 2014)*. pages 55–60.

Diana McCarthy and Roberto Navigli. 2009. The english lexical substitution task. *Language Resources and Evaluation* 43(2):139–159.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Neural Information Processing Systems Conference (NIPS 2013)*. pages 3111–3119.

Saif Mohammad. 2011. Colourful language: Measuring word-colour associations. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*. pages 97–106.

Sandra Mollin. 2009. Combining corpus linguistic and psychological data on word co-occurrences: Corpus collocates versus word associations. *Corpus Linguistics and Linguistic Theory* 5(2):175–200.

Douglas L. Nelson, McEvoy Cathy L., and Schreiber Thomas A. 2004. The University of South Florida free association, rhyme, and word fragment norms. *Behavior research methods, instruments, & computers : a journal of the Psychonomic Society, Inc* 36(3):402–407.

Timothy R. Rogers and James L. McClelland. 2004. *Semantic cognition: a parallel distributed processing approach*. The MIT Press.

Mark R. Rosenzweig. 1961. Comparisons among word-association responses in English, French, German, and Italian. *The American Journal of Psychology* 74(3):347–360.

Gerard Salton and Michael J. McGill. 1986. *Introduction to modern information retrieval*. McGraw-Hill, Inc., New York, NY, USA.

Margaret E. Tresselt and Mark S. Mayzner. 1964. The Kent-Rosanoff word association: Word association norms as a function of age. *Psychonomic Science* 1(1-12):65–66.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*. Seattle, WA, USA, October, pages 1815–1827.

Manfred Wettler and Reinhard Rapp. 1989. A connectionist system to simulate lexical decisions in information retrieval. *Connectionism in Perspective. Amsterdam: Elsevier* pages 463–469.

A Factored Neural Network Model for Characterizing Online Discussions in Vector Space

Hao Cheng **Hao Fang** **Mari Ostendorf**
University of Washington
{chenghao,hfang,ostendorf}@uw.edu

Abstract

We develop a novel factored neural model that learns comment embeddings in an unsupervised way leveraging the structure of distributional context in online discussion forums. The model links different context with related language factors in the embedding space, providing a way to interpret the factored embeddings. Evaluated on a community endorsement prediction task using a large collection of topic-varying Reddit discussions, the factored embeddings consistently achieve improvement over other text representations. Qualitative analysis shows that the model captures community style and topic, as well as response trigger patterns.

1 Introduction

Massive user-generated content on social media has drawn interests in predicting community reactions in the form of virality (Guerini et al., 2011), popularity (Suh et al., 2010; Hong et al., 2011; Lakkaraju et al., 2013; Tan et al., 2014), community endorsement (Jaech et al., 2015; Fang et al., 2016), persuasive impact (Althoff et al., 2014; Tan et al., 2016; Wei et al., 2016), etc. Many of these studies have analyzed content-agnostic factors such as submission timing and author social status, as well as language factors that underlie the textual content, e.g., the topic and idiosyncrasies of the community. In particular, there is an increasing amount of work on online discussion forums such as Reddit that exploits the conversational and community-centric nature of the user-generated content (Lakkaraju et al., 2013; Althoff et al., 2014; Jaech et al., 2015; Tan et al., 2016; Wei et al., 2016; He et al., 2016a; Fang et al., 2016), which contrasts with Twitter where the author's social status seems to play a larger role in popularity. This paper focuses on Reddit, using the karma score[1] as a readily available measure of community endorsement.

Some of the prior work on Reddit investigates specific linguistic phenomena (e.g. politeness, topic relevance, community style matching) using feature engineering to understand their role in predicting community reactions (Althoff et al., 2014; Jaech et al., 2015). In contrast, this paper explores methods for unsupervised text embedding learning using a model structured so as to provide some interpretability of the results when used in comment endorsement prediction. The model aims to characterize the interdependence of comment on its global context and subsequent responses that is characteristic of multi-party discussions. Specifically, we propose a factored neural model with separate mechanisms for representing global context, comment content and response generation. By factoring the model, we hope unsupervised learning will pick up different components of interactive language in the resulting embeddings, which will improve prediction of community reactions.

Distributed representations of text, or text embeddings, have achieved great success in many language processing applications, using both supervised and unsupervised methods. Unsupervised learning, in particular, has been successful at different levels, including words (Mikolov et al., 2013b), sentences (Kiros et al., 2015), and documents (Deerwester et al., 1990; Le and Mikolov, 2014). Studies have also shown that the learned embedding captures both syntactic and semantic functions of words (Mikolov et al., 2013a; Pennington et al., 2014; Levy and Goldberg, 2014; Faruqui et al., 2015a). At the same time, em-

[1] karma = #up-votes - #down-votes. See https://goo.gl/TnXgCr.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2296–2306
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

beddings are often viewed as uninterpretable – it is difficult to align embedding dimensions to existing semantic or syntactic classes. This concern has triggered attempts in developing more interpretable embedding models (Faruqui et al., 2015b), which is also a goal of our work. We leverage the fact that the structure of the distributional context impacts what is learned in an unsupervised way and include multiple objectives for separating different types of context.

Here, we are interested in linking two types of context with corresponding language factors learned in the embedding space that may impact comment reception. First, conformity to the topic and the language use of the community tends to make the content better accepted (Lakkaraju et al., 2013; Tan et al., 2014; Tran and Ostendorf, 2016). Those global *modes* typically influence the author's generation of local *content*. Second, characteristics of a comment can influence the *responses* it triggers. Clearly, questions and statements will elicit different responses, and comments directed at a particular discussion participant may prompt that individual to respond. Of more interest here are aspects of comments that might elicit minimal response or responses with different sentiments, which are relevant for eventual endorsement.

The primary contribution of this work is the development of a factored neural model to jointly learn these aspects of multi-party discussions from a large collection of Reddit comments in an unsupervised fashion. Extending the recent neural attention model (Bahdanau et al., 2015), the proposed model can interpret the learned latent global modes as community-related topic and style. A comment-response generation model component captures aspects of the comment that are response triggers. The multi-factored comment embedding is evaluated on the task of predicting the comment endorsement for three online communities different in topic trends and writing style. The representation of textual information using our approach consistently outperforms multiple document embedding baselines, and analyses of the global modes and response trigger subvectors show that the model learns common communication strategies in discussion forums.

2 Model Description

To characterize different aspects of language use in a comment, the proposed model factorizes a comment embedding into two sub-vectors, *i.e.* a *local mode* vector and a *content* vector. The local mode vector, computed as a mixture of global mode vectors, exploits the global context of a comment. In Reddit discussions that we use, the global mode represents the topic and language idiosyncracies (style) of a particular subreddit. More specific information communicated in the comment is captured in the content vector. The generation process of a comment is modeled through a recurrent neural network (RNN) language model (LM) conditioned on local mode and content vectors, while the global mode vectors are jointly learned during the training. Moreover, a residual learning architecture (He et al., 2016b) is used to extend the RNN LM for separating the information flow of the mode and the content vectors.

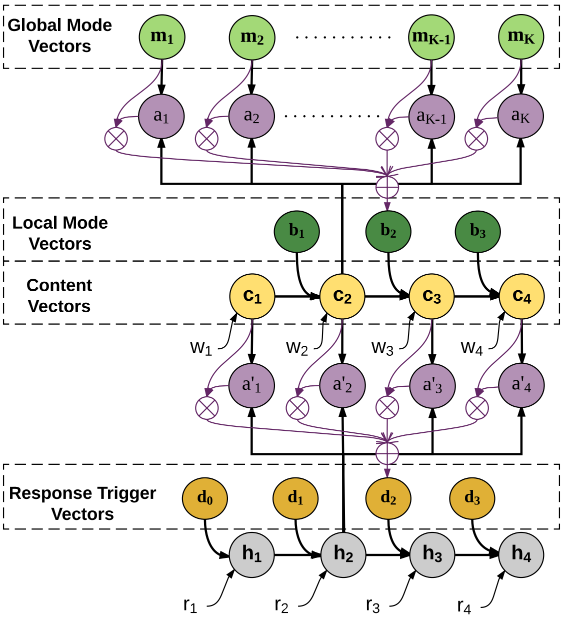


Figure 1: The structure of the full model omitting output layers, illustrating the computation of attention weights for $\mathbf{b}_2$ and $\mathbf{d}_3$ in a comment $w_{1:4}$ with its response $r_{1:4}$. Purple circles a_k and a'_j represent scalars computed in (1) and (6), respectively. $\otimes$ and $\oplus$ are scaling and element-wise addition operators, respectively. Black arrowed lines are connections carrying weight matrices.

In addition to the global context, the full model further exploits direct responses to the comment in order to learn better comment embeddings. This is achieved by modeling the generation of comment responses through another RNN LM conditioned on *response trigger* vectors. The response trigger vectors are computed as mixtures of content vectors, with the idea that they will characterize aspects of the comment that incent others to respond, whether that be information or framing.

The full model is illustrated in Fig. 1. While the end goal is a joint framework, the model is described in the following two sub-sections in terms of two components: i) mode vectors for capturing global context, and ii) response trigger vectors for exploiting comment responses.

2.1 Mode Vectors

Using an RNN LM shown in the upper part of Fig. 1, we model the generation process of a word sequence by predicting the next word conditioned on the global context as well as the local content. The global context is encoded in the local mode vector, computed as a mixture of global mode vectors with mixture weights inferred based on content vectors. The local mode vector indicates where the comment fits in terms of what people in this subreddit generally say. It changes dynamically with the content vector as the comment generation progresses, considering the possibility of topic shifts or different broad categories of discussion participants.

Suppose there is a set of K latent global modes with distributed representations $\mathbf{m}_{1:K} \in \mathbb{R}^n$. For the t-th word w_t in a sequence, a local mode vector $\mathbf{b}_t \in \mathbb{R}^n$ is computed as

$$\mathbf{b}_t = \textstyle\sum_{k=1}^{K} a(\mathbf{c}_t, \mathbf{m}_k) \otimes \mathbf{m}_k,$$

where $\mathbf{c}_t \in \mathbb{R}^n$ is the content vector for the current partial sequence $w_{1:t}$, $\otimes$ multiplies a vector by a scalar, and the function $a(\mathbf{c}_t, \mathbf{m}_k)$ outputs a scalar *association probability* for the current content vector $\mathbf{c}_t$ and a mode vector $\mathbf{m}_k$. The association function $a(\mathbf{c}, \mathbf{m}_k)$ is defined as

$$a(\mathbf{c}, \mathbf{m}_k) = \frac{\exp(\mathbf{v}^T \tanh(\mathbf{U}\,[\mathbf{c}; \mathbf{m}_k]))}{\sum_{i=1}^{K} \exp(\mathbf{v}^T \tanh(\mathbf{U}\,[\mathbf{c}; \mathbf{m}_i]))}, \quad (1)$$

where $\mathbf{U} \in \mathbb{R}^{n \times 2n}$ and $\mathbf{v} \in \mathbb{R}^n$ are parameters characterizing the similarity between $\mathbf{m}_k$ and $\mathbf{c}$.

The computation of the association probability is the well-known attention mechanism (Bahdanau et al., 2015). However, unlike the original attention RNN model where the attended vector is concatenated with the input vector to augment the input to the recurrent layer, we adopt a residual learning approach (He et al., 2016b) to learn content vectors. For the t-th word w_t in a sequence, the content vector $\mathbf{c}_t$ under the original attention RNN model is computed as

$$\mathbf{c}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{G}\mathbf{b}_{t-1}, \mathbf{c}_{t-1}), \quad (2)$$

where $\mathbf{x}_t \in \mathbb{R}^d$ is the word embedding for w_t, $\mathbf{b}_{t-1} \in \mathbb{R}^n$ and $\mathbf{c}_{t-1} \in \mathbb{R}^n$ are previous local mode and content vectors, respectively, $\mathbf{W} \in \mathbb{R}^{n \times d}$ and $\mathbf{G} \in \mathbb{R}^{n \times n}$ are weight matrices transforming the input to the recurrent layer, and $f(\cdot, \cdot)$ is the recurrent layer activation function. To address the vanishing gradient issue in RNNs, we use the gated recurrent unit (Cho et al., 2014) for the RNN layer, *i.e.*

$$f(\mathbf{p}, \mathbf{q}) = (\mathbf{1} - \mathbf{u}) \odot \tanh(\mathbf{p} + \mathbf{R}[\mathbf{r} \odot \mathbf{q}]) + \mathbf{u} \odot \mathbf{q},$$

where $\odot$ is the element-wise multiplication, $\mathbf{R}$ is the recurrent weight matrix, and $\mathbf{u}$ and $\mathbf{r}$ are the update and reset gates, respectively. In this paper, we compute the content vector $\mathbf{c}_t$ as follows:

$$\mathbf{c}_t = f(\mathbf{W}\mathbf{x}_t, \mathbf{G}\mathbf{b}_{t-1} + \mathbf{c}_{t-1}). \quad (3)$$

Comparing (2) and (3), it can be seen that we first aggregate the local mode vector $\mathbf{b}_{t-1}$ and the content vector $\mathbf{c}_{t-1}$ and treat the resulting vector $\mathbf{G}\mathbf{b}_{t-1} + \mathbf{c}_{t-1}$ as the memory of the recurrent layer. The resulting hidden state vectors from the recurrent layer are content vectors $\mathbf{c}_t$'s. The use of residual learning is motivated by the following considerations. The local mode vector $\mathbf{b}_{t-1}$ can be seen as a non-linear transformation of $\mathbf{c}_{t-1}$ into a global mode space parameterized by $\mathbf{m}_{1:K}$. If the global information carried in $\mathbf{b}_{t-1}$ is residual for generating the following word in the comment, the model only needs to exploit the information in local content $\mathbf{c}_{t-1}$ and learns to zero out the local mode vector $\mathbf{b}_{t-1}$, *i.e.* $\mathbf{G} = 0$. He et al. (2016b) show that the residual learning usually leads to a more well-conditioned model which promises better generalization ability.

Finally, the RNN LM estimates the probability of the $(t+1)$-th word w_{t+1} based on the current local mode vector $\mathbf{b}_t$ and content vector $\mathbf{c}_t$, *i.e.*

$$\Pr(w_{t+1}|w_{1:t}) = \mathrm{softmax}(\mathbf{Q}(\mathbf{G}\mathbf{b}_t + \mathbf{c}_t)), \quad (4)$$

where $\mathbf{Q} \in \mathbb{R}^{V \times n}$ is the weight matrix, and V is the vocabulary size. Note that the model jointly learns all parameters in the RNN together with the mode vectors $\mathbf{m}_{1:K}$. This differs our model from the context-dependent RNN LM (Mikolov and Zweig, 2012), which is conditioned on a context vector inferred from a pre-trained topic model.

2.2 Response Trigger Vectors

Another important aspect of comments in online discussions is how other participants react to the content. In order to exploit those characteristics, we use comment-reply pairs in online discussions and build this component upon the encoder-decoder framework with the attention mechanism (Bahdanau et al., 2015), which is illustrated in the lower part of Fig. 1. The decoder is essentially another RNN LM conditioned on response trigger vectors aiming at distilling relevant parts of the comment which other people are responding to.

Let r_j denote the j-th word in a reply to a comment $w_1, \cdots, w_T$. The decoder RNN LM computes a hidden vector $\mathbf{h}_j \in \mathbb{R}^n$ for r_j as follows,

$$\mathbf{h}_j = f(\mathbf{W}^\dagger \mathbf{x}_j + \mathbf{G}^\dagger \mathbf{d}_{j-1}, \mathbf{h}_{j-1}), \tag{5}$$

where $\mathbf{W}^\dagger \in \mathbb{R}^{n \times d}$ and $\mathbf{G}^\dagger \in \mathbb{R}^{n \times n}$ are weight matrices, $\mathbf{x}_j$ is r_j's word embeddings from a shared embedding dictionary as used by the encoder RNN LM in Subsection 2.1, and $\mathbf{d}_{j-1} \in \mathbb{R}^n$ and $\mathbf{h}_{j-1} \in \mathbb{R}^n$ are the response trigger vector and hidden vector at the previous time step, respectively. The initial hidden vector $\mathbf{h}_0$ is set to be the last content vector $\mathbf{c}_T$. With the comment's content vectors $\mathbf{c}_1, \cdots, \mathbf{c}_T$ obtained from the encoder RNN LM in Subsection 2.1, a response trigger vector $\mathbf{d}_j$ is computed as the mixture:

$$\mathbf{d}_j = \textstyle\sum_{t=1}^T a'(\mathbf{h}_j, \mathbf{c}_t) \cdot \mathbf{c}_t, \tag{6}$$

where $a'(\mathbf{h}_j, \mathbf{c}_t)$ is a similar function to $a(\mathbf{c}_t, \mathbf{m}_k)$ defined in (1) with different parameters. Similar to the encoder RNN LM, the decoder RNN LM estimates the probability of the $(j+1)$-th word r_{j+1} in the reply based on the hidden vector $\mathbf{h}_j$ and the response trigger vector $\mathbf{d}_j$, *i.e.*

$$\Pr(r_{j+1}|r_{1:j}) = \mathrm{softmax}(\mathbf{Q}^\dagger\, [\mathbf{h}_j; \mathbf{d}_j]),$$

where $\mathbf{Q}^\dagger \in \mathbb{R}^{V \times 2n}$ is the weight matrix.

Note the decoder RNN only aims at providing additional supervision signals in training the encoder RNN through a response generation task. At test time, we do not use the responses therefore do not need to run the decoder RNN LM.

3 Model Learning

The full model is trained by maximizing the log-likelihood of the data, *i.e.*

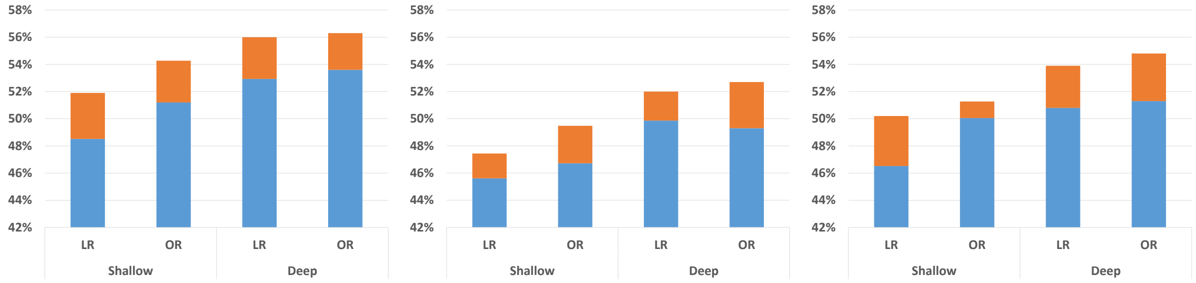
$$\textstyle\sum_i \log \Pr(w^{(i)}_{1:T^{(i)}}) + \alpha \log \Pr(r^{(i)}_{1:J^{(i)}} | w^{(i)}_{1:T^{(i)}}),$$

where the two terms correspond to the log-likelihood of the encoder RNN LM and the decoder RNN LM, respectively, and α is the hyper parameter which weights the importance of the second term. In our experiments, we let $\alpha = 0.1$. During the training, each comment-reply pair is used as a training sample. Considering that comments may receive a huge number of replies, we keep up to 5 replies for each comment. Due to memory limitations associated with the RNN, we use only the first 50 words of comments and the first 20 words of replies. If a comment has no reply, a special token is used. All weights are randomly initialized according to $\mathcal{N}(0, 0.01)$. The model is optimized using Adam (Kingma and Ba, 2015) with an initial learning rate 0.01. Once the validation log-likelihood decreases for the first time, we halve the learning rate at each epoch. The training process is terminated when the validation log-likelihood decreases for the second time. In our experiments, we learn word embeddings of dimension $d = 256$ from scratch. The number of modes K is set to 16. A single-layer RNN is used, with the dimension n of hidden layers set to 64.

4 Data and Task

In this paper, we work with Reddit discussion threads, taking advantage of their conversational and community-centric nature as well as the available karma scores. Each thread starts from a post and grows with comments to the post or other comments within the thread, presented as a tree structure. Posts and comments can be voted up or down by readers depending on whether they agree or disagree with the opinion, find it amusing vs. offensive, etc. A *karma* score is computed as the difference between up-votes and down-votes, which has been used as a proxy of community endorsement for a Reddit comment. Three popular subreddits with different topics and styles are studied[2] `AskWomen` (814K comments), `AskMen` (1,057K comments), and `Politics` (2,180K comments). For each subreddit, we randomly split comments by threads into training, validation, and test data, with a 3:1:1 ratio. The vocabulary of each subreddit is built on the training set. After removing singletons, the vocabulary sizes are 45K, 52K, and 60K for `AskWomen`, `AskMen`, and `Politics`, respectively.

[2] Comment IDs and labels used in this paper is at `https://github.com/hao-cheng/factored_neural`.



(a) `AskWomen` (b) `AskMen` (c) `Politics`

Figure 2: Averaged F1 scores of different classifiers. Blue bars show the performance using no comment embeddings. Orange bars show the absolute improvement by using factored comment embeddings.

Task: Considering the heavy-tailed Zipfian distribution of karma scores, regression with a mean squared error objective may not be informative because low-karma comments dominate the overall objective. Following (Fang et al., 2016), we quantize comment karma scores into 8 discrete levels and design a task consisting of 7 *binary* classification subtasks which individually predict whether a comment's karma is *at least* level-l for each level $l = 1, \cdots, 7$. This task is sensitive to the order of quantized karma scores, e.g., for the level-6 subtask, predicting a comment as level-5 or level-7 would lead to different evaluation results such as recall, which is not the case for a standard multi-class classification task. Additionally, compared to a standard multi-class classification task, these subtasks alleviate the unbalanced data issue, although higher levels are still more skewed.

Evaluation metric: For each level-l binary classification subtask, we compute the F1 score by treating comments at levels lower than l as negative samples and others as positive samples. Note that we only compute F1 scores for $l \in \{1, \ldots, 7\}$ since no comment is at a level lower than 0. The averaged F1 scores is used as an indicator of the overall prediction performance.

5 Experiments

In this section, we evaluate the effectiveness of the factored comment embeddings on the quantized karma prediction task. We use the concatenation of the local mode vector and the content vector at the last time step as the factored comment embedding. First, we study the overall prediction performance of four different classifiers under two settings, i.e., using factored comment embeddings or not. Then we compare the factored comment embeddings inferred from the full model and its two variants with other kinds of text features using the best type of classifiers. Finally, we carry out error analysis on prediction results of the best classifiers using the factored comment embeddings.

Range	Description
0/1	Whether the comment author is the user who initiated the thread.
$\mathbb{Z}_{\geq 0}$	Number of comments made by the author.
	Number of replies to the comment.
	Number of earlier comments.
	Number of later comments.
	Number of sibling comments.
	Number of comments in the subtree rooted from the comment.
	Height of the subtree rooted from the comment.
	Depth of the comment in the tree rooted from the original post.
$\mathbb{R}_{\geq 0}$	Relative comment time (in hours) with respect to the original post.
	Relative comment time (in hours) with respect to the parent comment.
	Normalized† number of replies to the comment.
	Normalized† number of comments in the subtree rooted from the comment.

Table 1: Content-agnostic features. † means two kinds of normalization are used: 1) zero-mean normalization; 2) divided by the squared-root-rank of the feature value in the thread.

	`AskWomen`	`AskMen`	`Politics`
Baseline	53.6%	49.3%	51.3%
BoW	53.1%	50.9%	51.8%
LDA	55.3%	51.1%	52.5%
Doc2Vec	55.2%	51.7%	53.0%
Factored$\backslash$M	54.2%	51.8%	52.9%
Factored$\backslash$R	55.1%	51.9%	53.4%
Factored	**56.3%**	**52.7%**	**54.8%**

Table 2: Averaged F1 scores of DeepOR classifiers using different text features. Baseline results do not use any text features.

5.1 Classifiers

The following four types of classifiers are studied:

- **ShallowLR**: A standard multi-class logistic regression model;
- **ShallowOR**: An ordinal regression model (Rennie and Srebro, 2005), which can exploit the or-

der information of the quantized karma labels;

- **DeepLR**: A feed-forward neural network using the logistic regression objective;
- **DeepOR**: A feed-forward neural network using the ordinal regression objective.

These classifiers have different objectives and model complexities, allowing us to study the robustness of the learned comment embeddings. The factored comment embeddings are inferred from the proposed models trained on the same training data but independently with these classifiers.

As baselines, we train the classifiers using only content-agnostic features, as shown in Table 1, which have strong correlations with community endorsement (Jaech et al., 2015; Fang et al., 2016). In our pilot work, we experimented with several groups of features from (Jaech et al., 2015) to find the content-agnostic features used in our paper. Since Jaech et al. (2015) work on a different task (ranking comments in a short time window), many of the useful content-agnostic features from (Jaech et al., 2015), including k-index, do not give additional improvement over the selected configuration for the karma prediction task.

All classifiers are trained on the training data for each subreddit independently, with hyperparameter tuned on the validation data. The penultimate weights are regularized using L_2 and the regularization parameters are selected in $\{0.0, 0.001, 0.01, 0.1, 1.0\}$. The number of hidden layers for deep classifiers are chosen from $\{1, 2, 3\}$, and the number of hidden neurons is selected from $\{32, 48, 64\}$.

We report the prediction performance on the test data, as shown in Fig. 2. We observe that using comment embeddings consistently improves the performance of these classifiers. While ShallowOR significantly outperforms ShallowLR, indicating the usefulness of exploiting the order information in quantized karma labels, the difference is much smaller for deep classifiers. Also, deep classifiers consistently outperforms their shallow counterparts.

5.2 Text Features

We compare the factored comment embeddings with the following text features:

- **BoW**: A sparse bag-of-word representation;
- **LDA**: A vector of topic probabilities inferred from the topic modeling (Blei et al., 2003);
- **Doc2Vec**: Embeddings inferred from the paragraph vector model (Le and Mikolov, 2014).

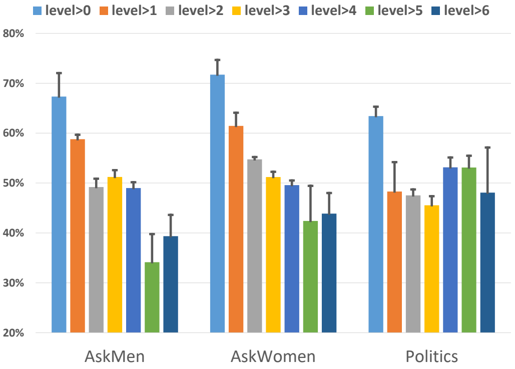
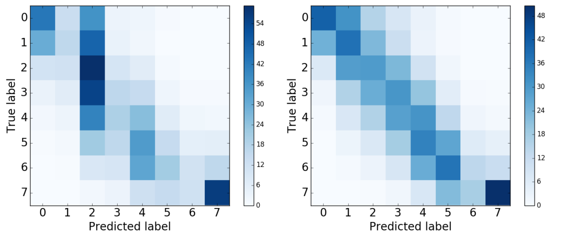


Figure 3: F1 scores of the DeepOR classifier for individual subtasks. Error bars indicate the improvement of using the factored comment embeddings over the classifier using no text features.

For these models, which do not use RNNs, all words in a comment are used. We use the gensim implementations (Řehůřek and Sojka, 2010) for both LDA and Doc2Vec. For LDA, the number of topic is selected in $\{16, 32, 64\}$, and 32 works the best on the validation set for all subreddits. For Doc2Vec, we select the embedding dimension from $\{32, 64, 128\}$, and 64 works the best on the validation set for all subreddits. We train the Doc2Vec for 20 epochs, and the learning rate is initialized as 0.025 and decreased by 0.001 at each epoch.

In addition to the factored comment embeddings obtained from our full model, we study two variants of the full model: 1) a model trained without the mode vector component (**Factored**\M), which is a normal sequence-to-sequence attention model (Bahdanau et al., 2015), and 2) a model trained without the response trigger vector component (**Factored**\R). All textual representations are used together with the baseline content-agnostic features described previously.

Since the DeepLR and the DeepOR perform best across all subreddits and they have similar trends, we report results of the DeepOR in Tabel 2. Among all text features, the BoW has the worst averaged F1 scores and even hurts the performance for AskWomen, probably due to the data sparsity problem. Both the LDA and the Doc2Vec outperform the BoW. The Doc2Vec performs slightly better on AskMen and Politics, which might be attributed to the relative larger training data size. The factored comment embeddings derived from the full model consistently achieve better averaged F1 scores. It can be observed that the



(a) w/o comment embeddings (b) w/ comment embeddings

Figure 4: The confusion matrices for the DeepOR classifier on `Politics`. The color of cell on the i-th row and the j-th column indicates the percentage of comments with quantized karma level i that are classified as j, and each row is normalized.

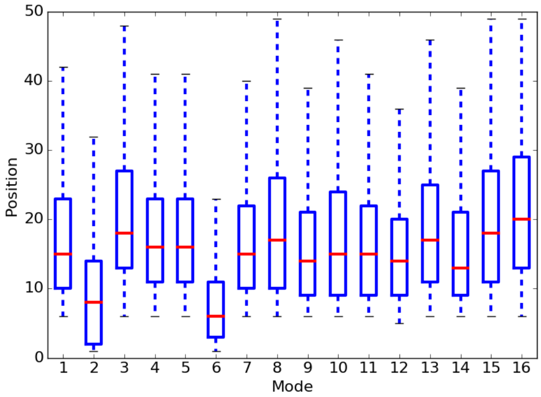


Figure 5: The box plot of strongest association positions for each global mode in `Politics`.

two variants of the full model mostly lead to similar performance as the Doc2Vec, though the Factored\R embeddings usually have higher averaged F1 scores than the Factored\M embeddings. These results suggest advantages of jointly modeling two components, which may drive the model to discover more latent factors and patterns in the data that could be useful for downstream tasks.

5.3 Error Analysis

In this subsection, we focus on analyzing how factored comment embeddings improve the prediction results of the DeepOR classifiers. The F1 scores for individual subtasks are shown in Fig. 3. Note that the higher the level is, the more skewed the task is, *i.e.* a lower positive ratio. As expected, comments with the lowest endorsement level are easier to classify. Adding comment embeddings primarily boosts the performance of the classifier on the high-endorsement tasks (level $> 5, 6$) and the low-endorsement tasks (level $> 0, 1$).

Confusion matrices for the DeepOR classifier with and without factored comment embeddings are shown in Fig. 4 for `Politics`. Using the additional comment embeddings leads to a higher concentration of cell weights near the diagonals, corresponding to errors that mainly confuse neighboring levels. Without any text features, the classifier seems to only distinguish four levels. We observe similar trends on `AskWomen` and `AskMen`.

6 Qualitative Analysis

In this section, we conduct analysis to better understand what the factored model is learning, again using the `Politics` subreddit. First, we analyze latent global modes learned from the full model. For each global mode, we extract comments with top association scores. Note that the model assumes a locally coherent mixture of global modes and updates the mixture for each observed word. Thus, each comment receives a sequence of association probabilities over the global modes. The association score β_k between a comment $w_{1:T}$ and `Mode-`k is then computed as $\beta_k = \max_{t\in\{1,\cdots,T\}} a(\mathbf{c}_t, \mathbf{m}_k)$ for $k \in \{1, \cdots, K\}$, where $a(\mathbf{c}_t, \mathbf{m}_k)$ is defined in (1). In Table 3, we show examples from the most coherent modes out of the 16 learned modes. Some modes seem to be capturing style (modes 2, 6, and 10), while others are related to topics (modes 7 and 16). `Mode-2` captures the style of starting with rhetorical question to express negative sentiment and disagreement. Many comments in `Mode-6` begin with words of drawing attention such as “bull” and “psst”. `Mode-10` tends to be associated with comments telling a story about a closely related person. Many comments in `Mode-7` discuss low salaries, whereas `Mode-16` comments discuss politicians or ideology of the Republican.

The characteristics of examples in modes 2 and 6 suggested that modes might have a location dependency, so we looked at word positions with the strongest association of each mode, *i.e.* $\mathrm{argmax}_{t\in\{1,\cdots,T\}} a(\mathbf{c}_t, \mathbf{m}_k)$. For each `Mode-`$k$, we only keep comments with association score higher than $\mathrm{mean}(\beta_k) + \mathrm{std}(\beta_k)$. Fig. 5 shows the box plot of locations where the strongest association happens. It can be seen that modes 2 and 6 usually have the strongest association at the beginning of a comment. For modes 3, 8, 15 and 16, the strongest associations occur over a wider span in comments. In addition to the interpretability of

Mode-2	• Oh come on! Really? One can't make that trip and spend maybe half and save the other for milk, bread and things that do spoil? … • Remind me. How many filibusters did Harry Reid conduct this year? … • Feckless tyrant? How did you do that with your brain? … • Seriously? You have to be registered to vote. … • Holy f*, seriously? This is some heavy duty shit. …
Mode-6	• Bull. Plenty of individuals influence policy by never missing a single chance to vote, no matter how minor the election. … • Bull. Conservatives hate Obamacare so much because if their constituents got mental health treatment, they'd stop voting Republican. • Utter bull s*. Where was the compromise from Obama and the Dems when they pushed through Obamacare without ONE Republican vote. … • psst…it's college • psst- he's "black" - meaning that one of his ancestors is black (as if it's pollutant of some sort).
Mode-7	• …, I used to work 55+ hours a week, salaried, lower quartile salary to boost. … • Or possibly that the standard of living between unemployment and the "jobs" that are out there is really insignificant. … • Where on earth is 7.25 a living wage? If by some miracle you get 40 hours a week that's only $1,160 before taxes. … • If you have to work 40 hours a week to pay your bills that means you are controlled in your fight for survival. … • …Working 15 hours a week for extra pocket money when you're a teen is easy. Working 50 hours a week at fast food to cover rent, food, …
Mode-10	• …Had a guy stalk a trans friend of mine for months trying to terrorize her because … • A co-worker of mine got audited by the IRS because … • …Some conservative friends of mind wanted to meet up at a coffee house with shittier coffee because the other one was too "liberal". … • …Friend of mine works with mentally unstable and aggressive people as part of some social service. … • …A student of mine asked our own AP about an atheist group and he just flat out said "You kidding me?" …
Mode-16	• …These same people will continue to listen to the bullshit that is the Republican Party. And when that happens, they have this twisted reality … • …After spending almost my entire life in Texas and as a Born gain evangelical conservative Republican, I learned my lessons about how completely dishonest and corrupted that entire culture is the hard way. …I will never gain ever vote for or support any kind of conservative. … • …has been our greatest embarrassment, but what makes matter even worse is the support he has for re-election. I would not be surprised … • Well, it is entirely possible that …the underlying cause of Limbaugh's attack was that this guy was playing the type of dirty politics … • …this was more of a referendum on the GOP leadership in Congress by Republican voters, because let's face it, they haven't done anything.…

Table 3: Examples of comments associated with the learned global modes for `Politics`.

the learned modes as one can get from LDA, these observations suggest that our model may further capture word location effects which may help predicting community endorsement.

Next, we analyze the response characteristics by examining the response trigger vectors associated with the onset of comment responses, which is a special start-of-reply token. These response trigger vectors are clustered into 8 classes via k-means and visualized in Fig. 6 using t-SNE (van der Maaten and Hinton, 2008). For each cluster, we study the karma distribution, as well as comments together with the first reply. Related data statistics and examples are shown in Fig. A-4 and Tables A-2&A-3 in the supplementary materials. The horizontal dimension seems to be associated with how many replies a comment elicits. The vertical dimension is less interpretable but most clusters have identifiable traits. The far left classes (Class-1&4) both have few replies and low karma, often two-party exchanges where Class-4 has more negative sentiment. Class-2 comments tend to involve complements, whereas comments in Class-3 usually trigger a reply with *but*-clause for a contrast and disagreement intent. Comments in Class-5 mostly receive responses expanding on the original comments. Class-6 has a lot of sarcastic and cynical comments and replies. Comments in Class-7 are mostly anomalous since their first responses were usually [deleted]. It seems there are multiple response trigger factors in the proposed embedding model, some may reflect dialog acts and others sentiment, any of which may be helpful in predicting community endorsement.

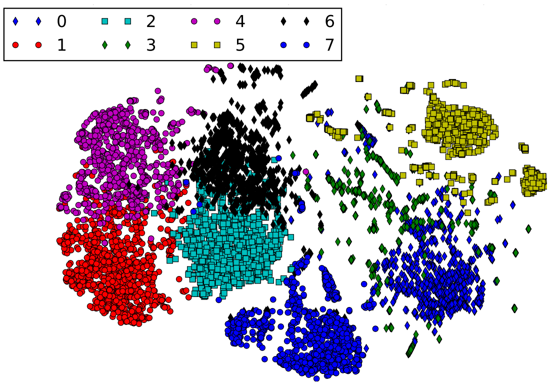


Figure 6: t-SNE visualization of response trigger vectors clustered using k-means.

7 Related Work

The skip-thought vector method (Kiros et al., 2015) is most closely related to our work in terms of utilizing context for unsupervised sequence modeling under the sequence-to-sequence framework (Sutskever et al., 2014). A key difference is the context being exploited. The skip-thought vector method uses surrounding sentences by abstracting the skip-gram structure (Mikolov et al., 2013a) from word to sequence. In our model, we exploit two types of context that are unique in online discussions: 1) the global context such as community topic and style which is learned in the *mode* vectors, and 2) the responses to a comment modeled as the *response trigger* vectors. Moreover, we augment our model with the attention mechanism (Bahdanau et al., 2015) to push the model to distill the relevant information from context.

The neural attention mechanism has been used for a variety of natural language processing tasks, e.g., machine translation (Bahdanau et al., 2015), question answering (Sukhbaatar et al., 2015), constituency parsing (Vinyals et al., 2015), social media opinion mining (Yang and Eisenstein, 2017). and dependency parsing (Cheng et al., 2016). The attention mechanism developed in this paper for exploiting global modes differs from previous work in that the global modes being attended over are *latent* rather than explicitly observed, and in that they are learned jointly with the full model.

Predicting the community endorsement has been studied by using either hand-crafted features (Jaech et al., 2015) or neural models (Fang et al., 2016; Zayats and Ostendorf, 2017), but all of them focus on supervised learning. Unsupervised learning strategies have been explored for characterizing different factors in language. A hierarchical Dirichlet process model was originally proposed for topic variations but has been extended to characterize multiple factors in (Huang and Renals, 2008). While much of the Dirichlet modeling work uses multinomial distributions, a loglinear version is introduced in (Eisenstein et al., 2011). Multi-dimensional structure latent factors in text are modeled by extending the sparsity-promoting topic model in (Paul and Dredze, 2012). Our model instead uses a neural network to characterize latent language factors, where the learned latent language factors could have a dependency on word positions.

8 Conclusion

This paper introduces a new factored neural model for unsupervised learning of comment embeddings leveraging two different types of context in online discussions. By extending the attention mechanism and using residual learning, our method is able to jointly model global context, comment content and response generation. Quantitative experiments on three different subreddits show that the factored embeddings achieve consistent improvement in predicting quantized karma scores over other standard document embedding methods. Analyses on the learned global modes show community-related style and topic characteristics are captured in our model. Also, we observe that response trigger vectors characterize certain aspects of comments that elicit different response patterns.

A potential future direction is to explore whether the comment embeddings derived from the unsupervised factored neural model can be useful across multiple tasks. Recently, a dataset with dialogue act annotations on Reddit discussions is published and can be used for a dialogue act prediction task (Zhang et al., 2017). Identifying or ranking persuasive arguments in the `ChangeMyView` subreddit (as studied in (Tan et al., 2016; Wei et al., 2016)) and asking for favors in the `RandomActsOfPizza` subreddit (used in (Althoff et al., 2014)) are also interesting for future work.

Acknowledgments

This paper is based on work supported by the DARPA DEFT Program. Views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Tim Althoff, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2014. How to ask for a favor: A case study of the success of altruistic request. In *Proc. Int. AAAI Conf. Web and Social Media (ICWSM)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. Int. Conf. Learning Representations (ICLR)*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Machine Learning Research*, 3:993–1022.

Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional attention with agreement for dependency parsing. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 2204–2214.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahadanau, Fethhi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 1724–1734.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *J. American Society for Information Science*, 41(6):391–407.

Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. 2011. Sparse additive generative models of text. In *Proc. Int. Conf. Machine Learning (ICML)*.

Hao Fang, Hao Cheng, and Mari Ostendorf. 2016. Learning latent local conversation modes for predicting community endorsement in online discussions. In *Proc. Int. Workshop Natural Language Process. for Social Media*.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, , and Noah A. Smith. 2015a. Retrofitting word vectors to semantic lexicons. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics (NAACL)*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015b. Sparse overcomplete word vector representations. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*.

Marco Guerini, Carlo Strapparava, and Gozde Ozba. 2011. Exploring text virality in social networks. In *Proc. Int. AAAI Conf. Web and Social Media (ICWSM)*.

Ji He, Mari Ostendorf, Xiaodong He, Jiansu Chen, Jianfeng Gao, Lihong Li, and Li Deng. 2016a. Deep reinforcement learning with a combinatorial action space for predicting popular Reddit threads. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 195–200.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Deep residual learning for image reconigtion. In *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Liangjie Hong, Ovidiu Dan, and Brian D. Davison. 2011. Predicting popular messages in Twitter. In *Proc. WWW*.

Songfang Huang and Steve Renals. 2008. Modeling topic and role information in meetings using the hierarchical Dirichlet process. In *Machine Learning for Multimodal Interaction V*, Springer Lecture Notes in Computer Science.

Aaron Jaech, Vicky Zayats, Hao Fang, Mari Ostendorf, and Hannaneh Hajishirzi. 2015. Talking to the crowd: What do people react to in online discussions? In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 2026–2031.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learning Representations (ICLR)*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 3294–3302.

Himabindu Lakkaraju, Julian McAuley, and Jure Leskovec. 2013. What's in a name? Understanding the interplay between titles, content, and communities in social media. In *Proc. Int. AAAI Conf. Web and Social Media (ICWSM)*.

Quo Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. Int. Conf. Machine Learning (ICML)*, pages 3104–3112.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 302–308.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *J. Machine Learning Research*, 9.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proc. Workshop at Int. Conf. Learning Representations*.

Tomas Mikolov, Wen-Tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proc. Conf. North American Chapter Assoc. for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 746–751.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *Proc. IEEE Spoken Language Technologies Workshop*.

Michael J. Paul and Mark Dredze. 2012. Factorial lda: Sparse multi-dimensional text models. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proc. LREC Workshop New Challenges for NLP Frameworks*.

Jason D. M. Rennie and Nathan Srebro. 2005. Loss functions for preference levels: regression with discrete ordered labels. In *Proc. Int. Joint Conf. Artificial Intelligence*.

Bongwon Suh, Lichan Hong, Peter Pirolli, and Ed H. Chi. 2010. Want to be retweeted? large scale analytics on factors impacting retweet in Twitter network. In *Proc. IEEE Second Intern. Conf. Social Computing*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 2431–2439.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 3104–3112.

Chenhao Tan, Lillian Lee, and Bo Pang. 2014. The effect of wording on message propagation: Topic- and author-controlled natural experiments on Twitter. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*.

Chenhao Tan, Vlad Niculae, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *Proc. WWW*.

Trang Tran and Mari Ostendorf. 2016. Characterizing the language of online communities and its relation to community reception. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*, pages 1030–1035.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proc. Annu. Conf. Neural Inform. Process. Syst. (NIPS)*, pages 2755–2763.

Zhongyu Wei, Yang Liu, and Yi Li. 2016. Ranking argumentative comments in the online forum. In *Proc. Annu. Meeting Assoc. for Computational Linguistics (ACL)*, pages 195–200.

Yi Yang and Jacob Eisenstein. 2017. Overcoming language variation in sentiment analysis with social attention. *Trans. Assoc. for Computational Linguistics (TACL)*.

Vicky Zayats and Mari Ostendorf. 2017. Conversation modeling on reddit using a graph-structured LSTM. *arXiv:1704.02080 [cs.CL]*.

Amy Zhang, Bryan Culbertson, and Praveen Paritosh. 2017. Characterizing online discussion using coarse discourse sequences. In *Proc. Int. AAAI Conf. Web and Social Media (ICWSM)*.

Dimensions of Interpersonal Relationships: Corpus and Experiments

Farzana Rashid and **Eduardo Blanco**
Human Intelligence and Language Technologies Lab
University of North Texas
Denton, TX, 76203
farzanarashid@my.unt.edu, eduardo.blanco@unt.edu

Abstract

This paper presents a corpus and experiments to determine dimensions of interpersonal relationships. We define a set of dimensions heavily inspired by work in social science. We create a corpus by retrieving pairs of people, and then annotating dimensions for their relationships. A corpus analysis shows that dimensions can be annotated reliably. Experimental results show that given a pair of people, values to dimensions can be assigned automatically.

1 Introduction

The task of information extraction (IE) consists in creating structured representations from unstructured text. These representations usually consist of relations explicitly stated in text, and involve two or more arguments. For example, IE systems would extract SPOUSE(*John*, *Mary*) or MARRIED(*John*, *Mary*, *1994*) from *John and Mary have been married since 1994*. IE systems have a long history, and became popular after evaluations such as MUC (Grishman and Sundheim, 1996) and ACE (Doddington et al., 2004).

Traditional IE systems are supervised and extract relations defined before training takes place (Peng and McCallum, 2004). More recently, open IE systems have been proposed to extract *all* relations explicitly stated in text in an unsupervised manner and without defining relations a priori (Mausam et al., 2012). Regarding interpersonal relations—relations that take as arguments two people—both IE approaches extract relations such as RELATIVE, FRIEND and COMMUNICATES_WITH. Open IE systems are domain independent and would extract, in principle, relations such as CLASSMATES and ADVISOR from students' diaries or biographies of scientists.

While useful for applications such as question answering (Yao and Van Durme, 2014), these dyadic relations only provide a generic understanding of the relationship between two people. For example, COMMUNICATES_WITH may relate people who have an *intense* or *superficial* relationship (e.g., engaged couples talking about wedding plans vs. home owners discussing remodels with contractors), *pleasure*- or *task-oriented* relationships (e.g., friends planning a backpacking trip vs. software developers discussing the next delivery), and may be *spatially near* or *distant* (e.g., inviduals having an in-person meeting vs. those exchanging emails or talking on the phone).

These elemental properties of interpersonal relationships are called *dimensions* in social science, and have been studied for decades (Wish et al., 1976). In those studies, the goal is to understand how relationships are *perceived* by people, not to extract them. Note that unlike interpersonal relationships, their dimensions are usually implicitly stated in text, thus extracting them is challenging. Also, extracting dimensions of interpersonal relationships requires text understanding beyond the event in which two people participate. As shown above, two people who communicate may have different dimension values depending on what they talk about or the communication device.

In this paper, we target dimensions of interpersonal relationships that characterize the nature of relationships beyond a name per relationship. The main contributions are: (a) set of dimensions of interpersonal relationships, including dimensions from previous work outside computational linguistics and novel ones; (b) corpus consisting of pairs of people and values for the dimensions of their relationships (Cohen's kappa: 0.68); (c) detailed corpus analysis; and (d) experimental results showing that the dimensions can be extracted automatically from text.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2307–2316
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Dimension	Other descriptors and aliases	Ref.
Cooperative vs. Competitive	friendly vs. hostile, promotive vs. contrient	[1]
Equal vs. Hierarchical	autonomy vs. control, submission vs. dominance	[1]
Intense vs. Superficial	important vs. insignificant, influential vs. trivial	[1]
Pleasure vs. Task Oriented	personal vs. impersonal, emotionally involved vs. detached	[1]
Active vs. Passive	direct vs. indirect, unequivocal vs. equivocal	[2]
Intimate vs. Unintimate	emotionally close vs. distant, indifferent, random	[3]
Temporary vs. Enduring	momentary vs. lasting, provisional vs. permanent	[4]
Concurrent vs. Nonconcurrent	convergent vs. divergent, synchronous vs. asynchronous	New
Spatially Near vs. Distant	nearby vs. faraway, attached vs. detached	New

Table 1: Dimensions of interpersonal relationships targeted in this paper. [1] stands for (Wish et al., 1976), [2] for (Kelley, 2013), [3] for (Adamopoulos, 2012), and [4] for (Deutsch, 2011). *New* indicates a dimension discovered after analyzing several examples and pilot annotations.

2 Related Work

Extracting relations between entities such as people, organizations and locations is the core goal of the task of information extraction. A few competitions have served as evaluation benchmarks (Grishman and Sundheim, 1996; Doddington et al., 2004; Kulick et al., 2014; Surdeanu and Heng, 2014), and include interpersonal relationships such as BUSINESS, SPOUSE and CHILDREN. Aguilar et al. (2014) compare several evaluations, and automated approaches to relation extraction—also referred to as link prediction and knowledge base completion—include (Yu and Lam, 2010; Nguyen et al., 2016; West et al., 2014). Open information extraction (Wu and Weld, 2010; Angeli et al., 2015) has emerged as an unsupervised domain-independent approach to extract relations. Regardless of details, all these previous efforts extract explicit relations, and do not attempt to characterize instances of relations with dimensions.

Besides extracting relations per se, there have been efforts within computational linguistics involving interpersonal relationships. Voskarides et al. (2015) extract human-readable descriptions of relations in a knowledge graph by ranking sentences that justify the relations. Iyyer et al. (2016) propose an unsupervised algorithm to extract relationship trajectories of fictional characters, i.e., how interpersonal relationships evolve over time in fictional stories. Bracewell et al. (2012) introduce 9 social acts (e.g., agreement, undermining) designed to characterize relationships between individuals exhibiting adversarial and collegial behavior (similar to our cooperative vs. competitive dimension).

Researchers have studied from a computational perspective how people communicate with each other. For example, Danescu-Niculescu-Mizil et al. (2012) study how power differences affects language style in online communities, and Prabhakaran and Rambow (2014) present a classifier to detect power relationships in email threads. Similarly, Gilbert (2012) explores how people in hierarchical relationships communicate through email, and Bramsen et al. (2011) focus on identifying power relationships in social networks. Politeness in online forums has also been studied (Danescu-Niculescu-Mizil et al., 2013). While power (similar to our equal vs. hierarchical dimension, Section 3) and politeness could be considered dimensions, these works exploit structural and linguistic features derived from communications between two individuals. Unlike all of them, we extract 9 dimensions of interpersonal relationships from sentences describing an event involving two people, and without needing language samples from them.

3 Dimensions of Interpersonal Relationships

Dimensions of interpersonal relationships have been studied for decades outside of computational linguistics, mostly in psychology and social science in general (Wish et al., 1976). The set of dimensions is by no means agreed upon, and neither is the terminology to refer to what apparently is the same dimension. For example, the terms *dominance*, *submission*, *potency*, *autonomy* and *control* are used to describe the distribution of power in a relationship (Deutsch, 2011).

The dimensions we work with in this paper are primarily borrowed from previous works in social science, although we add two new dimensions. Interestingly, the previous works which define these dimensions do so from a theoretical point of view or after conducting experiments with subjects to reveal how they perceive interpersonal relationships. The latter was done using multidimensional scaling analysis after subjects compared 25 relationships, e.g., between a parent and child, between business partners (Wish et al., 1976).

Table 1 presents the nine dimensions targeted in this paper along with the original references and aliases found in the literature. Social scientists have proposed additional dimensions, e.g., voluntary vs. involuntary, public vs. private and licit vs. illicit (Deutsch, 2011), or self-benefiting vs. service-oriented (Adamopoulos, 2012). We discarded these additional dimensions because we discovered that they are not applicable to most pairs of people we work with (Section 4).

We provide below brief descriptions of the nine dimensions of interpersonal relationships we work with. Note that these dimensions are not completely independent, for example, *enduring* relationships are usually *intense* and *intimate*, and *intense* and *pleasure-oriented* relationship are almost always *intimate*. Section 4.3 presents examples, and Section 4.4 discusses inter-dimensional correlations and inter-annotator agreement. We point the reader to the references in Table 1 for additional details, rationales, and examples.

Cooperative vs. Competitive A relationship is *cooperative* if both people (a) have a common interest or goal, (b) like each other, (c) benefit from the relationship, or (d) think alike or have similar views. Otherwise, the relationship is *competitive*.

Equal vs. Hierarchical A relationship is *equal* if both people (a) have the same social status, (b) are at the same level in the power structure, (c) share similar responsibilities, or (d) have the same role. Otherwise, the relationship is *hierarchical*.

Intense vs. Superficial A relationship is *intense* if both people interact with each other frequently, i.e., they are involved repeatedly. Otherwise, the relationship is *superficial*.

Pleasure vs. Task Oriented A relationship is *pleasure oriented* if both people interact socially and their relationship is not bound by professional rules or regulations. Otherwise, the relationship is *task oriented*.

Active vs. Passive A relationship is *active* if both people are involved in a shared activity or event that grants the relationship. Otherwise, the relationship is *passive*. For example, individuals commuting to work in the same car have an *active* relationship, but those who happen to take the same subway line to work have a *passive* relationship.

Intimate vs. Unintimate A relationship is *intimate* if both people are emotionally close and warm to each other. Otherwise, the relationship in *unintimate*.

Temporary vs. Enduring A relationship is *temporary* if it lasts less than a day. A relationship is *enduring* if it lasts over a month. Otherwise (if it lasts more than a day and less than a month), this dimension is undefined.

Concurrent vs. Nonconcurrent A relationship is *concurrent* if both people are involved in an event or action at the same time. Otherwise, the relationship is *nonconcurrent*.

Spatially Near vs. Distant A relationship is *spatially near* (or *near* for short) if both people are at the same location during the event that grants the relationship. Otherwise, the relationship is *spatially distant* (or *distant*).

4 Building a Corpus of Dimensions of Interpersonal Relationships

Existing corpora annotating relations (Section 2) only consider selected interpersonal relationships and do not target dimensions. Our goal is to target dimensions of interpersonal relationships between any two individuals, from weak links (e.g., journalists interviewing celebrities) to strong ties (e.g., close friends). Thus, we create a corpus[1] by first retrieving pairs of people, and then annotating dimensions for their relationships.

We decided to add our annotations to OntoNotes (Hovy et al., 2006). Doing so has several advantages. First, OntoNotes contains texts from several domains and genres (e.g., conversational telephone speech, weblogs, broadcast), thus we not only work with newspaper articles. Second, OntoNotes includes part-of-speech tags, named entities and coreference chains, three annotation layers that allow us to streamline the corpus creation process.

4.1 Retrieving Pairs of People

We retrieve pairs of people within each sentence in OntoNotes following four steps:

[1] Available at `http://hilt.cse.unt.edu/`

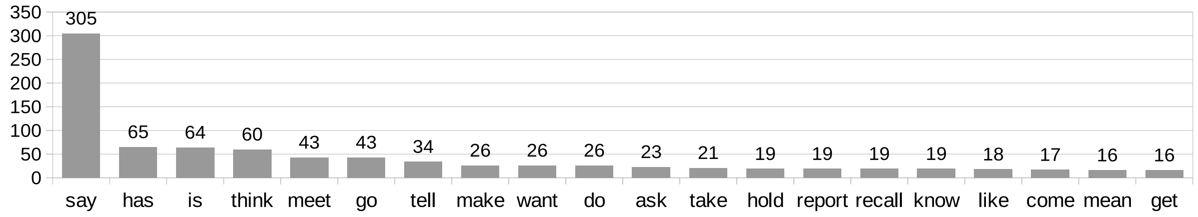


Figure 1: Frequencies of the top 20 most frequent verbs after retrieving pairs of people (Section 4.1). We discard verbs with frequency <4, and randomly select up to 26 pairs per verb for a total of 1,048 pairs.

1. Collect all instances of personal pronouns (part-of-speech tag PRP) *I*, *he* and *she*.
2. Collect all named entities PERSON.
3. Keep one mention per coreference chain, giving priority to named entities over pronouns.
4. Generate combinations of 2 elements from the union of the pronouns and named entities subject to the following constraints: at least
   (a) one is a PERSON named entity, and
   (b) one is the *nsubj* (nominal subject syntactic dependency) of a verb.

   The elements of the pair that satisfy restrictions (4a) and (4b) need not be the same.

Note that removing duplicate mentions (Step 3) does not reduce the number of relationships targeted, it simply avoids duplicate pairs. Also, the only syntactic constraint is that one person in the pair must be the nominal subject of a verb. Thus, we work with relationships between individuals from different clauses, and connected with a variety of syntactic paths (see Examples in Table 2).

The total number of pairs generated skipping Steps 3 and 4 would be 4,886. After removing duplicate mentions (Step 3), the number is reduced to 3,481; restrictions (4a) and (4b) further reduce the number to 3,143 and 2,696 respectively. Executing all steps yields 2,364 pairs.

Figure 1 presents verb frequencies for the top 20 most frequent verbs in the 2,364 pairs. In order to reduce the annotation effort and account for a variety of verbs, we set to annotate 1,000 pairs. After trying several thresholds, we retrieved 1,048 pairs by selecting pairs from verbs that occur at least 4 times, and randomly selecting up to 26 pairs per verb (most verbs occur less than 26 times).

4.2 Annotating Dimensions of Interpersonal Relationships

After generating pairs, annotators determine values for each dimension of interpersonal relationships. The annotation interface shows the sentence from which the pair was generated, and the previous and next sentence to provide some context. The pair of people of interest were highlighted, but no additional information was shown (e.g., the verb of which one person is the subject).

Annotators assign a value to each dimension based on the relationship between the two individuals at the time the verbal event of which one of the individuals is the subject takes place. They were trained to take into account context (previous and next sentences), and to interpret the text as they normally would. Therefore, they assign values using world knowledge that may not be explicitly stated in the text. For example, two people talking on the phone would have a *spatially distant* relationship because (most likely) they are not next to each other while talking. Annotating the changes over time of the dimensions is outside the scope of this paper.

During the first batch of annotations, we discovered that for a given pair of people, dimensions sometimes cannot be determined because (a) there is not enough evidence in the text provided (i.e., sentence from which the pair was generated, previous and next sentences) or (b) the pair is invalid and assigning dimensions is nonsensical. We use `0` label in the former case, and `inv` in the latter. For example, in the sentence *[He]*$_y$ *[criticized]*$_{verb}$ *[Ken Starr]*$_x$, the value for the dimension *spatially near* (vs. *distant*) was marked `0` as there is not enough information to determine whether *He* and *Ken Starr* are at the same location when *criticized* took place. The most frequent example of `inv` is when *God* is marked as a PERSON named entity in the gold annotations in OntoNotes.

In the rest of the paper, we refer to dimensions by the first descriptor in Table 1, and use `1` if the first descriptor of a dimension is true, and `-1` if the second descriptor is true. For example label `-1` applied to dimension *temporary* means that the realtionship is *enduring*.

	Sentence	Cooperative	Equal	Intense	Pleasure Oriented	Active	Intimate	Temporary	Concurrent	Spatially Near
1	[Cheney]$_x$ [got]$_{verb}$ a telephone call from his democratic counterpart [Joseph Lieberman]$_y$ wishing him a speedy recovery.	1	1	-1	-1	1	-1	1	1	-1
2	[...] [I]$_x$ [interviewed]$_{verb}$ one of the nation's top jockies [Shane Sellers]$_y$ about the battle he waged everyday to control his weight.	1	-1	-1	-1	1	-1	1	1	1
3	[I]$_x$ have always remembered the encouragement which Mr. [Yu Youren]$_y$ [gave]$_{verb}$ me as a young reporter. He said that to be a fearless champion of social justice, as is expected of a journalist, the [...]	1	-1	1	-1	1	1	-1	1	1
4	One day, Dingxiang took the opportunity to again urge him to change his ways [...]. After this, [Zhang Sheng]$_x$ [threw]$_{verb}$ out [Dingxiang]$_y$, sold off the family possessions, and spent his days living a life of dissipation.	-1	1	-1	1	1	-1	-1	1	1

Table 2: Annotation examples for pairs of people (x, y). We refer to dimensions by their first descriptor (Section 3); `1` (`-1`) indicates that the first (second) descriptor is true, and `0` that the value is unknown.

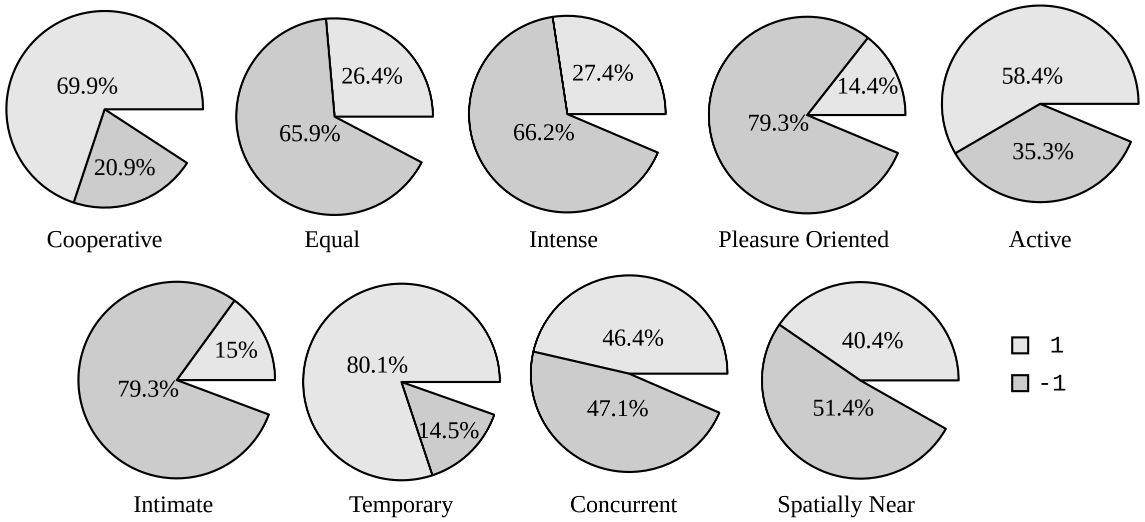


Figure 2: Label distribution per dimension of interpersonal relationships. The missing portion of each pie chart corresponds to labels `0` and `inv`, which always amount to less than 5% each.

4.3 Annotation Examples

We present annotation examples from our corpus in Table 2, including context if it is relevant. We acknowledge that some annotations are ambiguous, and discuss label distributions and inter-annotator agreement in Section 4.4.

Sentences (1) and (2) encode a COMMUNICATION relationships between two individuals, and both are *cooperative*, *superficial*, *work oriented*, *active*, *unintimate*, *temporary*, and *concurrent*. The values for two dimensions, however, are different. Two counterparts (Sentence 1) are at the same level in the power structure (*equal*), but interviewer and interviewee are not (Sentence 2). Similarly, talking on the phone entails that the individuals are *spatially distant* (Sentence 1), but interviewing (most likely) means that they were *spatially near* (Sentence 2). One could argue that `0` would be a better label for *spatially near* in Sentence (2), but annotators interpreted that *interviewed* refers to an in-person interview.

Sentence (3) in context describes one person (Yu Youren) encouraging another one (I). Annotators indicate that this relationship, unlike the ones in Sentences (1) and (2), is *intense* (frequent interaction), *intimate* (emotionally close), and *enduring* (lasting over a month). These values are not explicitly stated, but they are understood given the long-lasting impact *Yu Youren* had on *I*.

Finally, Sentence (4) exemplifies a *competitive* relationship. The context describes a struggling relationship between *Dingxiang* and *Zhang Sheng*. When the latter *threw* the former out, the rela-

	Cooperative	Equal	Intense	Pleasure Or.	Active	Intimate	Temporary	Concurrent
Cooperative	–							
Equal	-.06	–						
Intense	.24	.11	–					
Pleasure Or.	.14	.29	.39	–				
Active	.18	.22	.43	.25	–			
Intimate	.18	.17	.59	.62	.31	–		
Temporary	-.17	.10	-.56	-.43	-.27	-.61	–	
Concurrent	.17	.22	.31	.21	.74	.26	-.17	–
Spat. Near	.21	.18	.30	.25	.68	.28	.19	.89

Table 3: Pearson correlations between dimensions of interpersonal relationships in our corpus.

Dimension	Agreement	κ coefficient
Cooperative	86%	0.74
Equal	81%	0.63
Intense	84%	0.73
Pleasure Oriented	86%	0.70
Active	82%	0.59
Intimate	83%	0.68
Temporary	76%	0.61
Concurrent	84%	0.72
Spatially Near	80%	0.67
All	82%	0.68

Table 4: Inter-annotator agreement per dimension of interpersonal relationships. κ values in the 0.60–0.80 range are considered *substantial*, over 0.80 would be *perfect* (Landis and Koch, 1977).

tionship was *superficial* and *unintimate*, but (most likely) existed for longer than a month (*enduring*).

4.4 Corpus Analysis

Label Distribution. Figure 2 shows the percentage of label `1` (first descriptor) and `-1` (second descriptor) per dimension in our corpus. The percentage of Label `0` ranges from 0.86% (*temporary*) to 4.3% (cooperative) depending on the dimension, and the percentage of `inv` is 4.6% (not shown in Figure 2). Importantly, annotators assigned a useful value (either `1` or `-1`) to most pairs of people (>90%) for all dimensions.

The distributions of `1` and `-1` clearly show that most dimensions are biased towards one label. For example, few relationships are *pleasure oriented* (14.4%) or *enduring* (14.5%). The exceptions are *concurrent vs. nonconcurrent*, with roughly the same percentages (46.4% and 47.1%), *spatially near vs. distant* (40.4% and 51.4%) and *active vs. passive* (58.4% and 35.3%). These distributions are not a representative sample of all interpersonal relationships, we would expect many *pleasure oriented* and *intimate* relationship if we work with personal diaries instead of OntoNotes.

Inter-Dimension Correlations. While the dimensions we work with have a long tradition in social science (Section 2), to the best of our knowledge, they have not been extensively annotated in text before. Table 3 shows inter-dimensional correlations for all pairs of dimensions in our corpus. Not surprisingly, some dimensions correlate with each other. For example, *enduring* relationships tend to also be *intense* (0.56) and *intimate* (0.61), and *concurrent* relationships tend to be *active* (0.74). The highest correlation is between *concurrent* and *spatially near* (0.89), indicating that if two people participate in a common event at the same time, usually they are at the same location (see counterexample in Table 2, Sentence 1). Note, however, that most correlations are low, and some dimensions (e.g., *cooperative*, *equal*) have low correlations (<0.30) with all dimensions.

Inter-Annotator Agreement. The annotations were done by two graduate students. They started annotating small batches of pairs of people, and discussed disagreements with each other. After several iterations, they annotated independently 10% of all pairs of people generated. Table 4 depicts the inter-annotator agreements obtained. Overall Cohen's kappa coefficient is 0.68, and the coefficients range between 0.59 to 0.74 depending on the dimension. Note that kappa coefficients in the range 0.60–0.80 are considered *substantial*, and over 0.80 would be *perfect* (Landis and Koch, 1977). Given these high agreements, the rest of pairs were annotated once.

5 Experiments and Results

We conduct experiments using standard supervised machine learning. Each pair of people become an instance, and we split instances into training (80%) and test (20%). As a learning algorithm, we use SVM with RBF kernel as implemented in scikit-learn (Pedregosa et al., 2011).

We report results in the test set after tuning the SVM parameters (C and γ) using 10-fold cross-validation with the training set. More specifically, we train one classifier per dimension, and experiment with all instances but the ones annotated `inv`. Thus, each classifier predicts 3 labels: `1` (the first descriptor applies), `-1` (the second descriptor applies), and `0` (neither descriptor applies).

	Feature	Description
Verb	word, tag	Word form and part-of-speech tag of the verb
	dep_out	Outgoing syntactic dependency from verb
	deps_in	Flags indicating incoming syntactic dependencies to the verb
	lex_name	Name of the WordNet lexical file of the verb
	token_before	Word form and part-of-speech tag of the token before the verb
	token_after	Word form and part-of-speech tag of the token after the verb
Person	words, tags	Concatenation of word forms and part-of-speech tags
	type	Flag indicating whether the person is a pronoun or named entity
	dep_out	Outgoing syntactic dependency
	distance_verb	Number of tokens between the person and the verb
	first_token	Word form and part-of-speech tag of the first token in the person
	last_token	Word form and part-of-speech tag of the last token in the person
	token_before	Word form and part-of-speech tag of the token before the person
	token_after	Word form and part-of-speech tag of the token after the person
Person$_x$_Person$_y$	direction	Flag indicating whether *x* occurs before or after *y*
	type	Flag indicating whether *x* and *y* are PERSON NEs, or either one is a pronoun

Table 5: Feature set used to determine dimensions of interpersonal relationships between pairs of people (*x*, *y*). *Verb* features are extracted from the verb of which either *x* or *y* is the subject, *Person* features are extracted from *x* and *y* independently, and *Persons* features are extracted from *x* and *y*.

5.1 Feature Set

The features we work with are summarized in Table 5. Most features are standard and have been used before to extract relations from text (Section 2). Following the notation in Table 2, we refer to the pair of people as *x* and *y*.

Verb features capture information about the verb to which *x* or *y* attach. We include words and part-of-speech tags (verb, and tokens before and after), the name of the WordNet lexical file to which the verb belongs, and dependencies.

Person features are extracted from *x* and *y* independently, and consists mostly of words and part-of-speech tags. We also include a flag indicating whether the person is a pronoun or named entity (type feature), and the number of tokens between the person and the verb (distance_verb).

Person$_x$_Person$_y$ features capture information of both *x* and *y*. They capture (a) whether *x* occurs before or after *y* in the sentence, and (b) whether they are both named entities or one is a pronoun and the other one a named entity (type feature).

5.2 Results

We present overall results (averages of the classifiers for each dimension) using the majority baseline and with several feature combinations in Table 6. Then, we present detailed results per dimension with the best feature combination in Table 7. We only present results obtained in the test set.

Overall Results and Feature Ablation (Table 6). The majority baseline obtains 0.53 average F-measure. Recall that we build a classifier per dimension, thus the combination of the nine majority-baseline classifiers predict two labels: `1` (0.55 F-measure) and `-1` (0.64 F-measure).

Features	Label	P	R	F
Majority Baseline	`1`	0.43	0.76	0.55
	`0`	0.00	0.00	0.00
	`-1`	0.57	0.59	0.64
	Avg.	0.45	0.65	0.53
Verb	`1`	0.62	0.72	0.64
	`0`	0.00	0.00	0.00
	`-1`	0.71	0.78	0.65
	Avg.	0.62	0.71	0.65
Verb, Person$_x$	`1`	0.68	0.75	0.69
	`0`	0.00	0.00	0.00
	`-1`	0.78	0.79	0.78
	Avg.	0.70	0.74	0.71
Verb, Person$_y$	`1`	0.64	0.72	0.67
	`0`	0.00	0.00	0.00
	`-1`	0.74	0.76	0.74
	Avg.	0.66	0.70	0.67
Verb, Person$_x$, Person$_y$, Person$_x$_Person$_y$	`1`	0.69	0.74	0.70
	`0`	0.00	0.00	0.00
	`-1`	0.77	0.80	0.76
	Avg.	**0.71**	**0.76**	**0.72**

Table 6: Results obtained for all dimensions with several combinations of features.

Models trained with any combination of features outperform the majority baseline, but they never learn to predict label `0`. Since `0` occurs between 0.86% and 4.3% depending on the dimension (Section 4.4), this limitation does not affect overall performance substantially.

Dimension	`1` (1st descriptor)			`0` (unknown)			`-1` (2nd descriptor)			All		
	P	R	F	P	R	F	P	R	F	P	R	F
Cooperative	0.73	0.96	0.83	0.00	0.00	0.00	0.60	0.19	0.29	0.66	0.72	0.65
Equal	0.56	0.10	0.17	0.00	0.00	0.00	0.74	0.97	0.84	0.68	0.74	0.66
Intense	0.39	0.30	0.34	0.00	0.00	0.00	0.78	0.85	0.82	0.67	0.71	0.69
Pleasure	0.40	0.28	0.33	0.00	0.00	0.00	0.87	0.93	0.90	0.79	0.82	0.80
Active	0.69	0.85	0.76	0.00	0.00	0.00	0.68	0.51	0.58	0.67	0.69	0.67
Intimate	0.44	0.17	0.24	0.00	0.00	0.00	0.88	0.98	0.93	0.81	0.86	0.83
Temporary	0.85	0.96	0.91	0.00	0.00	0.00	0.33	0.10	0.16	0.77	0.83	0.79
Concurrent	0.72	0.80	0.76	0.00	0.00	0.00	0.77	0.75	0.76	0.71	0.74	0.73
Spat. Near	0.66	0.68	0.67	0.00	0.00	0.00	0.73	0.79	0.76	0.66	0.70	0.68
Average	0.69	0.74	0.70	0.00	0.00	0.00	0.77	0.80	0.76	0.71	0.76	0.72

Table 7: Results obtained for each dimension with the best combination of features for all dimensions (*Verb* + *Person*$_x$ + *Person*$_y$ + *Person*$_x$_*Person*$_y$, boldfaced in Table 6)

Verb features alone yield a 0.65 average F-measure (`1`: 0.64, `-1`: 0.65). Adding features derived from *x* (*Verb* + *Person*$_x$) improves performance (0.71 average F-measure), and adding features derived from *y* (*Verb* + *Person*$_y$) slightly improves performance (0.67 average F-measure). In both cases, `-1` is predicted more accurately than `1` (0.78 vs. 0.69 and 0.74 vs. 0.67).

Finally, adding all features (*Verb* + *Person*$_x$ + *Person*$_y$ + *Person*$_x$_*Person*$_y$) yields the best results (average F-measure: 0.72), although by a minimal margin with respect to *Verb* + *Person*$_x$.

Detailed Results per Dimension. Table 7 presents results per dimension with the best overall combination of features (*Verb* + *Person*$_x$ + *Person*$_y$ + *Person*$_x$_*Person*$_y$).

All dimensions obtain overall F-measures between 0.65 and 0.83 (last column). Results per label are heavily biased towards the most frequent label per dimension (Figure 2), although it is the case that the models we experiment with predict both `1` and `-1` for all dimensions. As stated above, none of them predict `0`, but this limitation does not substantially penalize overall performance because of the low frequency of this label.

The model obtains the same F-measures for `1` and `-1` with *concurrent* dimension (0.76), and the labels of this dimension are virtually distributed uniformly (46.4% vs. 47.1%, Figure 2). Similarly, F-measures for `1` and `-1` with *spatially near* and *active* dimensions are similar (0.67 vs. 0.76 and 0.76 vs. 0.58), and the labels are distributed relatively evenly in our corpus (40.4% vs 51.4% and. 58.4% vs. 35.3%).

Finally, F-measures per label with other dimension are biased towards the most frequent label. For example, only 15% of all pairs of people have an *enduring* relationship (Figure 2), and the F-measure for `1` with *temporary* dimension is much higher (0.91) that for `-1` (0.16).

6 Conclusions

We have presented a set of nine dimensions of interpersonal relationships, including dimensions with a long tradition in social science and new ones. These dimensions allow us to differentiate core characteristics of the relationship between two individuals. For example, people that communicate may be *spatially near* or *spatially distant* (asking questions in class vs. chatting online), and have a pleasure-oriented or work-oriented relationship (somebody wishing good luck to a friend vs. interviewer and interviewee).

Our annotations show that assigning values to dimensions can be done reliably (Cohen's kappa: 0.68), and that useful values (`1` and `-1` labels) are assigned to dimensions in most pairs of people (>90%). Experimental results following a standard supervised machine learning approach show that assigning values to dimensions can be automated (0.72 overall F-measure), and that results per label and dimensions are biased towards the most frequent label.

We believe that extracting dimensions of interpersonal relationships complements previous efforts that extract relationships. Our future plans include studying values of dimensions for selected relationships (e.g., COWORKER), and investigating changes on the dimensions of the relationship over time. The latter would allow us to, for example, analyze how the relationship between two individuals changes over time, and determine which events make a relationship go from *superficial* to *intense* and vice versa.

References

John Adamopoulos. 2012. The emergence of social meaning: A theory of action construal. In *Handbook of Social Resource Theory: Theoretical Extensions, Empirical Insights, and Social Applications*, pages 255–272. Springer New York, New York, NY.

Jacqueline Aguilar, Charley Beller, Paul McNamee, Benjamin Van Durme, Stephanie Strassel, Zhiyi Song, and Joe Ellis. 2014. A comparison of the events and relations across ace, ere, tac-kbp, and framenet annotation standards. In *Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 45–53, Baltimore, Maryland, USA. Association for Computational Linguistics.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

David B. Bracewell, Marc T. Tomlinson, Mary Brunson, Jesse Plymale, Jiajun Bracewell, and Daniel Boerger. 2012. Annotation of adversarial and collegial social actions in discourse. In *Proceedings of the Sixth Linguistic Annotation Workshop*, LAW VI '12, pages 184–192, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philip Bramsen, Martha Escobar-Molano, Ami Patel, and Rafael Alonso. 2011. Extracting social power relationships from natural language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 773–782, Portland, Oregon, USA. Association for Computational Linguistics.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 699–708, New York, NY, USA. ACM.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 250–259, Sofia, Bulgaria. Association for Computational Linguistics.

Morton Deutsch. 2011. Interdependence and psychological orientation. In *Conflict, Interdependence, and Justice*, pages 247–271. Springer.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ace) program tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC-2004)*, Lisbon, Portugal. European Language Resources Association (ELRA). ACL Anthology Identifier: L04-1011.

Eric Gilbert. 2012. Phrases that signal workplace hierarchy. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1037–1046, New York, NY, USA. ACM.

Ralph Grishman and Beth Sundheim. 1996. Message understanding conference-6: A brief history. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 1*, COLING '96, pages 466–471, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% Solution. In *NAACL '06: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers on XX*, pages 57–60, Morristown, NJ, USA. Association for Computational Linguistics.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1534–1544, San Diego, California. Association for Computational Linguistics.

Harold H Kelley. 2013. *Personal relationships: Their structures and processes*. Psychology Press.

Seth Kulick, Ann Bies, and Justin Mott. 2014. Inter-annotator agreement for ere annotation. In *Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 21–25, Baltimore, Maryland, USA. Association for Computational Linguistics.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1).

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 523–534, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 460–466, San Diego, California. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Fuchun Peng and Andrew McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In *HLT-NAACL 2004: Main Proceedings*, pages 329–336, Boston, Massachusetts, USA. Association for Computational Linguistics.

Vinodkumar Prabhakaran and Owen Rambow. 2014. Predicting power relations between participants in written dialog from a single thread. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 339–344, Baltimore, Maryland. Association for Computational Linguistics.

Mihai Surdeanu and Ji Heng. 2014. Overview of the english slot filling track at the tac2014 knowledge base population evaluation. In *Proceedings of the TAC-KBP 2014 Workshop*.

Nikos Voskarides, Edgar Meij, Manos Tsagkias, Maarten de Rijke, and Wouter Weerkamp. 2015. Learning to explain entity relationships in knowledge graphs. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 564–574, Beijing, China. Association for Computational Linguistics.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 515–526, New York, NY, USA. ACM.

Myron Wish, Morton Deutsch, and Susan J Kaplan. 1976. Perceived dimensions of interpersonal relations. *Journal of Personality and Social Psychology*, 33(4):409.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 118–127, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966, Baltimore, Maryland. Association for Computational Linguistics.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1399–1407, Stroudsburg, PA, USA. Association for Computational Linguistics.

Argument Mining on Twitter: Arguments, Facts and Sources

Mihai Dusmanu and **Elena Cabrio** and **Serena Villata**
Université Côte dAzur, CNRS, Inria, I3S, France

Abstract

Social media collect and spread on the Web personal opinions, facts, fake news and all kind of information users may be interested in. Applying argument mining methods to such heterogeneous data sources is a challenging open research issue, in particular considering the peculiarities of the language used to write textual messages on social media. In addition, new issues emerge when dealing with arguments posted on such platforms, such as the need to make a distinction between personal opinions and actual facts, and to detect the source disseminating information about such facts to allow for provenance verification. In this paper, we apply supervised classification to identify arguments on Twitter, and we present two new tasks for argument mining, namely facts recognition and source identification. We study the feasibility of the approaches proposed to address these tasks on a set of tweets related to the Grexit and Brexit news topics.

1 Introduction

Argument mining aims at automatically extracting natural language arguments and their relations from a variety of textual corpora, with the final goal of providing machine-processable structured data for computational models of arguments and reasoning engines (Peldszus and Stede, 2013; Lippi and Torroni, 2016). Several approaches have been proposed so far to tackle the two main tasks identified in the field: *i) arguments extraction*, i.e., to detect arguments within the input natural language texts and the further detection of their boundaries, and *ii) relations prediction*, i.e., to predict what are the relations holding between the arguments identified in the first task[1]. Social media platforms like Twitter[2] and newspapers blogs allow users to post their own viewpoints on a certain topic, or to disseminate news read on newspapers. Being these texts short, without standard spelling and with specific conventions (e.g., hashtags, emoticons), they represent an open challenge for standard argument mining approaches (Snajder, 2017). The nature and peculiarity of social media data rise also the need of defining new tasks in the argument mining domain (Addawood and Bashir, 2016; Llewellyn et al., 2014).

In this paper, we tackle the first standard task in argument mining, addressing the research question: *how to mine arguments from Twitter?* Going a step further, we address also the following subquestions that arise in the context of social media: *i)* how to distinguish factual arguments from opinions? *ii)* how to automatically detect the source of factual arguments? To answer these questions, we extend and annotate a dataset of tweets extracted from the streams about the Grexit and the Brexit news. To address the first task of argument detection, we apply supervised classification to separate *argument-tweets* from non-argumentative ones. By considering only argument-tweets, in the second step we apply again a supervised classifier to recognize tweets reporting factual information from those containing opinions only. Finally, we detect, for all those arguments recognized as factual in the previous step, what is the source of such information (e.g., the CNN), relying on the type of the Named Entities recognized in the tweets. The last two steps represent new tasks in the argument

[1] We refer the reader interested in more details on argument mining to (Peldszus and Stede, 2013; Lippi and Torroni, 2016) as survey papers, and to the proceedings of the Argument Mining workshop series (`https://argmining2017.wordpress.com/`).

[2] `www.twitter.com`

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2317–2322
Copenhagen, Denmark, September 7–11, 2017.

mining research field, of particular importance in social media applications.

2 Mining arguments on Twitter

In this section, we describe the approaches we have developed to address the following tasks: *i)* Argument detection, *ii)* Factual vs opinion classification, and *iii)* Source identification, on social media data. Our experimental setting - whose goal is to investigate the tasks' feasibility on such peculiar data - considers a dataset of tweets related to the political debates on whether or not Great Britain and Greece had to leave the European Union (i.e. #Brexit and #Grexit threads in Twitter).

2.1 Experimental setting

Dataset.[3] The only available resource of annotated tweets for argument mining is DART (Bosc et al., 2016a). From the highly heterogeneous topics contained in such resource (i.e. the letter to Iran written by 47 U.S. senators; the referendum for or against Greece leaving the EU; the release of Apple iWatch; the airing of the 4th episode of the 5th season of the TV series Game of Thrones), and considering the fact that tweets discussing a political topic generally have a more developed argumentative structure than tweets commenting on a product release, we decided to select for our experiments the subset of the DART dataset on the thread *#Grexit* (987 tweets). Then, following the same methodology described in (Bosc et al., 2016a), we have extended such dataset collecting 900 tweets from the thread on *#Brexit*. From the original thread, we filtered away retweets, accounts with a bot probability >0.5 (Davis et al., 2016), and almost identical tweets (Jaccard distance, empirically evaluated threshold). Given that tweets in DART are already annotated for task 1 (argument/non-argument, see Section 2.2), two annotators carried out the same task on the newly extracted data. Moreover, the same annotators annotated both datasets (Grexit/Brexit) for the other two tasks of our experiments, i.e. *i)* given the argument tweets, annotation of tweets as either containing factual information or opinions (see Section 2.3), and *ii)* given factual argument tweets, annotate their source when explicitly cited (see Section 2.4). Tables 1, 2 and 3 contain statistical information on the datasets.

Inter annotator agreement (IAA) (Carletta, 1996) between the two annotators has been calculated for the three annotation tasks, resulting in κ=0.767 on the first task (calculated on 100 tweets), κ=0.727 on the second task (on 80 tweets), and Dice=0.84 (Dice, 1945)[4] on the third task (on the whole dataset). More specifically, to compute IAA, we sampled the data applying the same strategy: for the first task, we randomly selected 10% of the tweets of the Grexit dataset (our training set); for task 2, again we randomly selected 10% of the tweets annotated as argument in the previous annotation step; for task 3, given the small size of the dataset, both annotators annotated the whole corpus.

dataset	# argument	# non-arg	total
Brexit	713	187	900
Grexit	746	241	987
total	1459	428	1887

Table 1: Dataset for task 1: argument detection

dataset	# factual arg.	# opinion	total
Brexit	138	575	713
Grexit	230	516	746
total	368	1091	1459

Table 2: Dataset for task 2: factual arguments vs opinions classification

dataset	# arg. with source cit.	# arg. without source cit.	total
Brexit	40	98	138
Grexit	79	151	230
total	119	249	368

Table 3: Dataset for task 3: source identification

Classification algorithms. We tested Logistic Regression (LR) and Random Forest (RF) classification algorithms, relying on the *scikit-learn* tool suite[5]. For the learning methods, we have used a Grid Search (exhaustive) through a set of predefined hyper-parameters to find the best performing ones (the goal of our work is not to optimize

[3] Annotated data are available upon request to the authors.

[4] Dice is used instead of κ to account for partial agreement on the set of sources detected in the tweets.

[5] http://scikit-learn.org/

the classification performance but to provide a preliminary investigation on new tasks in argument mining over Twitter data). We extract argument-level features from the dataset of tweets (following (Wang and Cardie, 2014)), that we group into the following categories:

- *Lexical (L):* unigram, bigram, WordNet verb synsets;
- *Twitter-specific (T):* punctuation, emoticons;
- *Syntactic/Semantic (S):* we have two versions of dependency relations as features, one being the original form, the other generalizing a word to its POS tag in turn. We also use the syntactic tree of the tweets as feature. We apply the Stanford parser (Manning et al., 2014) to obtain parse trees and dependency relations;
- *Sentiment (SE):* we extract the sentiment from the tweets with the Alchemy API[6], the sentiment analysis feature of IBM's Semantic Text Analysis API. It returns a polarity label (positive, negative or neutral) and a polarity score between -1 (totally negative) and 1 (totally positive).

As baselines we consider both LR and RF algorithms with a set of basic features (i.e., lexical).

2.2 Task 1: Argument detection

The task consists in classifying a tweet as being an argument or not. We consider as arguments all those text snippets providing a portion of a standard argument structure, i.e., opinions under the form of claims, facts mirroring the data in the Toulmin model of argument (Toulmin, 2003), or persuasive claims, following the definition of argument tweet provided in (Bosc et al., 2016a,b). Our dataset contains 746 argument tweets and 241 non-argument tweets for Grexit (that we use as training set), and 713 argument tweets and 187 non-argument tweets for Brexit (the test set). Below we report an example of argument tweet (a), and of a non-argument tweet (b).

(a) *Junker asks "who does he think I am". I suspect elected PM Tsipras thinks Junker is an unelected Eurocrat. #justsaying #democracy #grexit*

(b) *#USAvJPN #independenceday #JustinBieberBestIdol Macri #ConEsteFrioYo happy 4th of july #Grefenderum Wireless Festival*

We cast the argument detection task as a binary classification task, and we apply the supervised algorithms described in Section 2.1. Table 4 reports on the obtained results with the different configurations, while Table 5 reports on the results obtained by the best configuration, i.e., LR + All features, per each category.

Approach	Precision	Recall	F1
RF+L	0.76	0.69	0.71
LR+L	0.76	0.71	0.73
LR+all features	0.80	0.77	**0.78**

Table 4: Results obtained on the test set for the argument detection task (L=lexical features)

Category	P	R	F1	#arguments per category
non-arg	0.46	0.60	0.52	187
arg	0.89	0.82	0.85	713
avg/total	0.80	0.77	**0.78**	900

Table 5: Results obtained by the best model on each category of the test set for the argument detection task

Most of the miss-classified tweets are either ironical, e.g.:

If #Greece had a euro for every time someone mentioned #Grexit and #Greferendum they would probably have enough for a bailout. #GreekCrisis

that was wrongly classified as argument, or contain reported speech, e.g.:

Jeremy Warner: Unintentionally, the Greeks have done themselves a favour. Soon, they will be out of the euro `http://t.co/YmqXi361Gj` *#Grexit*

that was wrongly classified as non argument. Our results are comparable to those reported in (Bosc et al., 2016b) (they trained a supervised classifier on the tweets of all topics in the DART dataset but the iWatch, used as test set). Better performances obtained in our setting are most likely due to a better feature selection, and to the fact that in our case

[6] https://www.ibm.com/watson/alchemy-api.html

the topics in the training and test sets are more homogeneous.

2.3 Task 2: Factual vs opinion classification

This task consists in classifying argument-tweets as containing factual information or being opinion-based (Park et al., 2015). Our interest focuses in particular on factual argument-tweets, as we are interested then in the automated identification of their sources. This would allow then to rank factual tweet-arguments depending on the reliability or expertise of their source for subsequent tasks as fact checking. Given the huge amount of work in the literature devoted to opinion extraction, we do not address any further analysis on opinion-based arguments here, referring the interested reader to (Liu, 2012).

An argument is annotated as *factual* if it contains a piece of information which can be proved to be true (see example (a) below), or if it contains "reported speech" (see example (b) below). All the other argument tweets are considered as "opinion" (see example (c) below).

(a) *72% of people who identified as "English" supported #Brexit (while no majority among those identifying as "British")* https://t.co/MuUXqncUBe

(b) *#Hollande urges #UK to start #Brexit talks as soon as possible.* https://t.co/d12TV8JqYD.

(c) *Trump is going to sell us back to England. #Brexit #RNCinCLE*

Our dataset contains 230 factual argument tweets and 516 opinion argument tweets for Grexit (training set), and 138 factual argument tweets and 575 opinion argument tweets for Brexit (test set).

To address the task of factual vs opinion arguments classification, we apply the supervised classification algorithms described in Section 2.1. Tweets from Grexit dataset are used as training set, and those from Brexit dataset as test set. Table 6 reports on the obtained results, while Table 7 reports on the results obtained by the best configuration, i.e. LR + All features, per each category.

Most of the miss-classified tweets contain reported opinions/reported speech and are wrongly classified by the algorithm as opinion - such behaviour could be expected given that sentiment features play a major role in these cases, e.g.,

Approach	Precision	Recall	F1
RF+L	0.75	0.68	0.71
LR+L	0.75	0.75	0.75
LR+all features	0.81	0.79	**0.80**

Table 6: Results obtained on the test set for the factual vs opinion argument classification task (L=lexical features)

Category	P	R	F1	#arguments per category
fact	0.49	0.50	0.50	138
opinion	0.88	0.87	0.88	575
avg/total	0.81	0.79	**0.80**	713

Table 7: Results obtained by the best model on each category of the test set for the factual vs opinion argument classification task

Thomas Piketty accuses Germany of forgetting history as it lectures Greece http://t.co/B0UqPn0i6T *#grexit*

Again, the other main reason for miss-classification is sarcasm/irony contained in the tweets, e.g.,

So for Tsipras, no vote means back to the table, for Varoufakis, meant Grexit?

that was wrongly classified as fact.

2.4 Task 3: Source identification

Since factual arguments (as defined above) are generally reported by news agencies and individuals, the third task we address - and that can be of a value in the context of social media - is the recognition of the information source that disseminates the news reported in a tweet (when explicitly mentioned). For instance, in:

The Guardian: Greek crisis: European leaders scramble for response to referendum no vote. http://t.co/cUNiyLGfg3

the source of information is The Guardian newspaper. Such annotation is useful to rank factual tweet-arguments depending on the reliability or expertise of their source in news summarization or fact-checking applications, for example.

Our dataset contains 79 factual argument tweets where the source is explicitly cited for Grexit (training set), and 40 factual argument tweets where the source is explicitly cited for Brexit (test set). Given the small size of the available annotated dataset, to address this task we implemented a simple string matching algorithm that relies on a gazetteer containing a set of Twitter usernames and hashtags extracted from the training data, and a list of very common news agencies (e.g. BBC, CNN, CNBC). If no matches are found, the algorithm extracts the NEs from the tweets through (Nooralahzadeh et al., 2016)'s system, and applies the following two heuristics: *i)* if a NE is of type `dbo:Organisation` or `dbo:Person`, it considers such NE as the source; *ii)* it searches in the abstract of the DBpedia[7] page linked to that NE if the words "news", "newspaper" or "magazine" appear (if found, such entity is considered as the source). In the example above, the following NEs have been detected in the tweet: "The Guardian" (linked to the DBpedia resource `http://dbpedia.org/page/The_Guardian`) and "Greek crisis" (linked to `http://dbpedia.org/page/Greek_government-debt_crisis`). Applying the mentioned heuristics, the first NE is considered as the source. Table 8 reports on the obtained results. As baseline, we use a method that considers all the NEs detected in the tweet as sources.

Approach	Precision	Recall	F1
Baseline	0.26	0.48	0.33
Matching+heurist.	0.69	0.64	**0.67**

Table 8: Results obtained on the test set for the source identification task

Most of the errors of the algorithm are due to information sources not recognized as NEs (in particular, when the source is a Twitter user), or NEs that are linked to the wrong DBpedia page. However, in order to draw more interesting conclusions on the most suitable methods to address this task, we would need the increase the size of the dataset.

3 Discussion and Future work

This paper investigated argument mining tasks on Twitter data. The main contribution is twofold: first, we propose one of the very few approaches of argument mining on Twitter, and second, we propose and evaluate two new tasks for argument mining, i.e., facts recognition and source identification. These tasks are particularly relevant when applied to social media data, in line with the open popular challenges of fact-checking and source verification to which these results contribute.

The issue of argument detection on Twitter has already been addressed in the literature. Bosc et al. (2016a,b) address a binary classification task (argument-tweet vs. non argument), as first step of their pipeline. Goudas et al. (2015) experiments machine learning techniques over a dataset in Greek extracted from social media. They first detect argumentative sentences, and second identify premises and claims. However, none of them is neither interested in distinguishing facts from opinions nor to identify the arguments' sources. An argumentation-based approach is applied to Twitter data to extract opinions in (Grosse et al., 2015), with the aim of detecting conflicting elements in an opinion tree to avoid potentially inconsistent information. Both the goal and the adopted methodology are different from ours.

Being it a work in progress, several open issues have to be considered as future research. Among them, we are currently extending the dataset of annotated tweets both in terms of annotated tweets per topic, and in terms of addressed topics (e.g. Brexit after the referendum, Trump), in order to have more instances of facts and sources. On such extended dataset, we plan to run experiments using the three modules of the system as a pipeline.

Moreover, we plan to extend our pipeline by considering also the links provided in the tweets to verify their sources, i.e., if a tweet claims to report information from the CNN but the link actually redirects towards an advertisement website the source is not indubitably the CNN.

References

Aseel Addawood and Masooda Bashir. 2016. "what is your evidence?" A study of controversial topics on social media. In *Proceedings of the Third Workshop on Argument Mining, hosted by the 54th Annual Meeting of the Association for Computational Linguistics, ArgMining@ACL 2016, August 12, Berlin, Germany*. http://aclweb.org/anthology/W/W16/W16-2801.pdf.

Tom Bosc, Elena Cabrio, and Serena Villata. 2016a. DART: a dataset of arguments and their relations on twitter. In Nicoletta Calzolari, Khalid

[7] `http://www.dbpedia.org`

Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2016/summaries/611.html.

Tom Bosc, Elena Cabrio, and Serena Villata. 2016b. Tweeties squabbling: Positive and negative results in applying argument mining on social media. In Pietro Baroni, Thomas F. Gordon, Tatjana Scheffler, and Manfred Stede, editors, *Computational Models of Argument - Proceedings of COMMA 2016, Potsdam, Germany, 12-16 September, 2016.*. IOS Press, volume 287 of *Frontiers in Artificial Intelligence and Applications*, pages 21–32. https://doi.org/10.3233/978-1-61499-686-6-21.

Jean Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. *Comput. Linguist.* 22(2):249–254. http://dl.acm.org/citation.cfm?id=230386.230390.

Clayton Allen Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. 2016. Botornot: A system to evaluate social bots. In *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, WWW '16 Companion, pages 273–274. https://doi.org/10.1145/2872518.2889302.

L. R. Dice. 1945. Measures of the amount of ecologic association between species. *Journal of Ecology* 26:297–302.

Theodosios Goudas, Christos Louizos, Georgios Petasis, and Vangelis Karkaletsis. 2015. Argument extraction from news, blogs, and the social web. *International Journal on Artificial Intelligence Tools* 24(5). https://doi.org/10.1142/S0218213015400242.

Kathrin Grosse, María Paula González, Carlos Iván Chesñevar, and Ana Gabriela Maguitman. 2015. Integrating argumentation and sentiment analysis for mining opinions from twitter. *AI Commun.* 28(3):387–401. https://doi.org/10.3233/AIC-140627.

Marco Lippi and Paolo Torroni. 2016. Argumentation mining: State of the art and emerging trends. *ACM Trans. Internet Techn.* 16(2):10:1–10:25. http://doi.acm.org/10.1145/2850417.

B. Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis digital library of engineering and computer science. Morgan & Claypool. https://books.google.fr/books?id=Gt8g72e6MuEC.

Clare Llewellyn, Claire Grover, Jon Oberlander, and Ewan Klein. 2014. Re-using an argument corpus to aid in the curation of social media collections. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014.*. pages 462–468. http://www.lrec-conf.org/proceedings/lrec2014/summaries/845.html.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Farhad Nooralahzadeh, Cédric Lopez, Elena Cabrio, Fabien L. Gandon, and Frédérique Segond. 2016. Adapting semantic spreading activation to entity linking in text. In *Natural Language Processing and Information Systems - 21st International Conference on Applications of Natural Language to Information Systems, NLDB 2016, Salford, UK, June 22-24, 2016, Proceedings*. pages 74–90. http://dx.doi.org/10.1007/978-3-319-41754-7_7.

Joonsuk Park, Cheryl Blake, and Claire Cardie. 2015. Toward machine-assisted participation in erulemaking: an argumentation model of evaluability. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law, ICAIL 2015, San Diego, CA, USA, June 8-12, 2015*. pages 206–210. https://doi.org/10.1145/2746090.2746118.

Andreas Peldszus and Manfred Stede. 2013. From argument diagrams to argumentation mining in texts: A survey. *IJCINI* 7(1):1–31. https://doi.org/10.4018/jcini.2013010101.

Jan Snajder. 2017. Social media argumentation mining: The quest for deliberateness in raucousness. *CoRR* abs/1701.00168. http://arxiv.org/abs/1701.00168.

S.E. Toulmin. 2003. *The Uses of Argument*. Cambridge University Press. https://books.google.fr/books?id=53whAwAAQBAJ.

Lu Wang and Claire Cardie. 2014. Improving agreement and disagreement identification in online discussions with A socially-tuned sentiment lexicon. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, WASSA@ACL 2014, June 27, 2014, Baltimore, Maryland, USA*. pages 97–106. http://aclweb.org/anthology/W/W14/W14-2617.pdf.

Distinguishing Japanese Non-standard Usages from Standard Ones

Tatsuya Aoki[†] **Ryohei Sasano**[‡] **Hiroya Takamura**[†] **Manabu Okumura**[†]
[†]Department of Information and Communications Engineering, Tokyo Institute of Technology
[‡]Graduate School of Informatics, Nagoya University
{aoki,takamura,oku}@lr.pi.titech.ac.jp
sasano@i.nagoya-u.ac.jp

Abstract

We focus on non-standard usages of common words on social media. In the context of social media, words sometimes have other usages that are totally different from their original. In this study, we attempt to distinguish non-standard usages on social media from standard ones in an unsupervised manner. Our basic idea is that non-standardness can be measured by the inconsistency between the expected meaning of the target word and the given context. For this purpose, we use context embeddings derived from word embeddings. Our experimental results show that the model leveraging the context embedding outperforms other methods and provide us with findings, for example, on how to construct context embeddings and which corpus to use.

1 Introduction

On social media such as Twitter, we often find posts that are difficult to interpret without prior knowledge on non-standard usage of words. For example, consider the following Japanese sentence[1]:

鯖の	負担が	増える
mackerel-POSS	load-NOM	increase-PRS

"*The load on a mackerel increases*",

which does not make sense given the standard usages for the words in the sentence. But here, *mackerel* is a non-standard usage that means *computer server*. The entire sentence should be interpreted as *"The load on the computer server increases"*. The Japanese word "鯖 (*saba*)" (i.e., mackerel) is used to mean computer server by Japanese computer geeks because *saba* happens to have a pronunciation that is similar to *sābā* (i.e., computer server). When a word is used in a meaning that is different from its dictionary meaning, we call such a usage *non-standard*.[2]

Non-standard usages can be found in many languages (Sboev, 2016). For example, the word "catfish" means a ray-finned fish as in a standard dictionary, but on social media, it can mean a person who pretends to be someone else in order to create a fake identity. Such non-standard usages would be an obstacle to a variety of language processings including machine translation; Google Translate cannot correctly interpret examples such as this. Humans, however, would be able to notice non-standard usages from the inconsistency between the expected word meaning and the context.

The purpose of this work is to develop a method for distinguishing non-standard usages of Japanese words from standard ones. Since it is impractical to construct a large labeled data set for each word, we focus on unsupervised approaches. The main idea in our method is that the difference between the target word's embedding learned from a general corpus and the embedding predicted from the given context would be a good indicator of the degree of non-standardness.

2 Data

We created a dataset for evaluating our method. First, we selected 40 words that have non-standard usages, including computer terms, company/service names, and other Internet slang. Ten

[1] This is interlinear-gloss text representation. POSS, NOM, PRS respectively represent the possessive case, nominative case, and present tense. The third line is the standard translation of the Japanese sentence.

[2] Although some non-standard usages are metaphoric, such as *sunshine* in "*you are my sunshine*", our definition of non-standard usages covers a wider variety of usages, as in the example of "mackerel".

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2323–2328
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Category	Usage	
	standard	non-standard
Computer terms	416	234
Company/Service names	440	252
Other Internet slang	817	814
Total	1,673	1,300

Table 1: Statistics of the dataset.

of the 40 words were computer terms, another 10 were company/service names, and the remaining 20 were other Internet slang. For each of the 40 target words, we found 100 tweets that contained the target word. Here, we used Twitter as the source for examples since there are many non-standard usages on it. To segment tweets into words, we used the Japanese morphological analyzer MeCab[3] with the standard IPA standard dictionary.[4]

Next, we asked two human annotators to judge whether the usage of the target word in each tweet is standard, non-standard, a named entity, or undecidable. We excluded tweets which at least one annotator judged as undecidable (96 tweets).[5] Cohen's kappa of the annotations for the remaining 3,904 tweets was 0.808. We further excluded tweets which at least one annotator judged as containing a named entity (772 tweets) in order to focus the dataset on our main purpose.[6]

Finally, to create a final dataset, we selected from the remaining 3,132 tweets the 2,973 tweets that are judged as standard by both annotators or as non-standard by them. The selected 2,973 tweets are equivalent to 94.9% of the entire set of tweets, which suggests that human can reliably distinguish non-standard usages from standard ones. The statistics of the final dataset are shown in Table 2.

3 Methodology

Our basic idea for distinguishing word usages is that if a word is used in a non-standard manner, the context words around it will tend to differ from standard context words. To implement this idea, we employed word embeddings. Below, we review the Skip-gram model used for obtaining the word embeddings in Section 3.1 and present our method in Section 3.2.

3.1 Skip-gram

Skip-gram (Mikolov et al., 2013) is widely used for obtaining word embeddings. Given a sequence of words w_1, w_2, ..., w_T as training data, Skip-gram maximizes the likelihood $\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq i\leq m, i\neq 0}\log p(w_{t+i}|w_t)$, where m is the window size. w_{t+i} is a context word nearby w_t. $p(w_k|w_t)$ is given by

$$p(w_k|w_t) = \frac{\exp(v_{w_t}^{IN}\cdot v_{w_k}^{OUT})}{\sum_{w=1}^{W}\exp(v_{w_t}^{IN}\cdot v_{w}^{OUT})},$$

where W is the vocabulary size of the training data. Skip-gram learns a model predicting context words using word embeddings v^{IN} and v^{OUT}, which are called *input* embedding and *output* embedding respectively.

The embeddings are learned in such a way that $v_{w_t}^{IN}\cdot v_{w_k}^{OUT} - \log\sum_{w=1}^{W}\exp(v_{w_t}^{IN}\cdot v_{w}^{OUT})$ increases if word w_t occurs near w_k in the training corpus. As a result, $v_{w_t}^{IN}\cdot v_{w_k}^{OUT}$ tends to be large for such words and small for word pairs that do not co-occur in the training corpus. We exploited this tendency for recognizing non-standard usages; if the dot-product between the embeddings of the target word and the context words is small, it should indicate a non-standard usage, on the condition that the embeddings have been learned on a general balanced corpus where words correspond to their standard meanings in most cases.

v^{IN} is widely used as a *word embedding* in many studies, while v^{OUT} has not been in the limelight; only a few researchers have examined the effectiveness of v^{OUT} (Mitra et al., 2016; Press and Wolf, 2017). In recent studies, embeddings v^{IN} are usually used for measuring the similarity between words. However, given the characteristics described in the previous paragraph and SGNS's equivalence with shifted positive pointwise mutual information (Levy and Goldberg, 2014), if we want to measure to what extent word w_t tends to co-occur with w_k in the training data, then we should use the similarity of $v_{w_t}^{IN}\cdot v_{w_k}^{OUT}$, instead of $v_{w_t}^{IN}\cdot v_{w_k}^{IN}$.

In this study, we show the importance of using v^{OUT} in a task where we need to see if a word matches its context.

[3] http://taku910.github.io/mecab/

[4] https://ja.osdn.net/projects/ipadic/

[5] The undecidable tweets are meaningless expressions such as the emoticon "(*´茸｀*)", which includes the word "茸".

[6] Most of the discarded target words are in a named entity, such as the word "尻" in "利尻島". These expressions are different from our definition of non-standard usage.

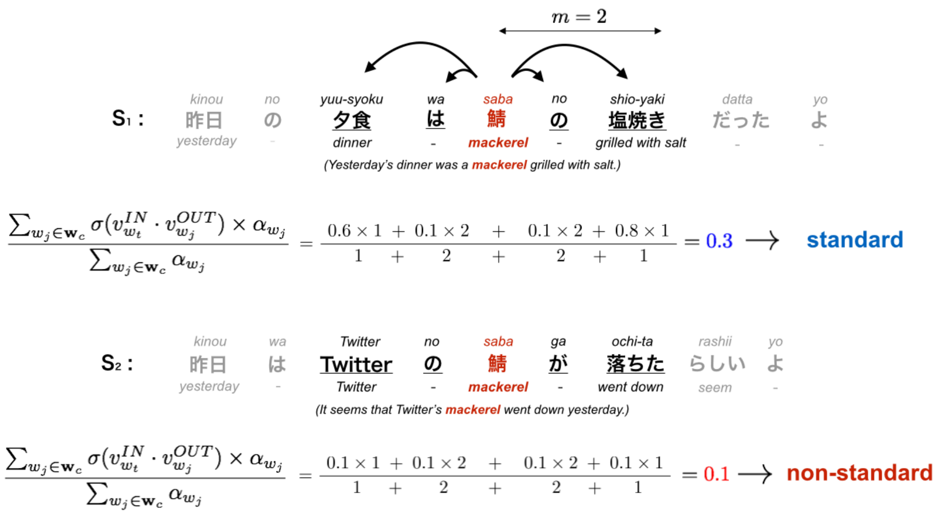

m = 2
S₁ :
kinou 昨日 yesterday
no の -
yuu-syoku 夕食 dinner
wa は -
saba 鯖 mackerel
no の -
shio-yaki 塩焼き grilled with salt
datta だった -
yo よ -
(Yesterday's dinner was a mackerel grilled with salt.)
$\frac{\sum_{w_j \in \mathbf{w}_c} \sigma(v^{IN}_{w_t} \cdot v^{OUT}_{w_j}) \times \alpha_{w_j}}{\sum_{w_j \in \mathbf{w}_c} \alpha_{w_j}} = \frac{0.6 \times 1 + 0.1 \times 2 + 0.1 \times 2 + 0.8 \times 1}{1 + 2 + 2 + 1} = 0.3 \longrightarrow$ standard
S₂ :
kinou 昨日 yesterday
wa は -
Twitter Twitter Twitter
no の -
saba 鯖 mackerel
ga が -
ochi-ta 落ちた went down
rashii らしい seem
yo よ -
(It seems that Twitter's mackerel went down yesterday.)
$\frac{\sum_{w_j \in \mathbf{w}_c} \sigma(v^{IN}_{w_t} \cdot v^{OUT}_{w_j}) \times \alpha_{w_j}}{\sum_{w_j \in \mathbf{w}_c} \alpha_{w_j}} = \frac{0.1 \times 1 + 0.1 \times 2 + 0.1 \times 2 + 0.1 \times 1}{1 + 2 + 2 + 1} = 0.1 \longrightarrow$ non-standard

Figure 1: Overview of our method. Our model exploits context embedding and weighting.

3.2 Distinguishing Non-standard Usages from Standard Ones

Following the idea described in Section 3.1, we propose a method for distinguishing non-standard usages from standard ones by leveraging word embeddings. An overview of our method is shown in Figure 1. We use Skip-gram with Negative Sampling (SGNS) (Mikolov et al., 2013) for obtaining the word embeddings.

Given a target word w_t and its context $\mathbf{w}_c$ as input, we calculate the following weighted average of scaled dot-products as a measure of standardness:

$$\frac{\sum_{w_j \in \mathbf{w}_c} \sigma(v^{IN}_{w_t} \cdot v^{OUT}_{w_j}) \times \alpha_{w_j}}{\sum_{w_j \in \mathbf{w}_c} \alpha_{w_j}}, \tag{1}$$

where $v^{IN}_{w_t}$ is the input embedding for the target word w_t and $v^{OUT}_{w_j}$ is the output embedding for the context word w_j. α_{w_j} is a non-negative weight for the word w_j, and σ is the sigmoid function used for scaling dot-products into a range from 0 to 1. Although the values of α_{w_j} are arbitrary, we will use the values given by the training algorithm used in `word2vec`[7] and `gensim` (Řehůřek and Sojka, 2010), popular tools for obtaining word embeddings. In their training of word embeddings, context words that are closer to the target word are weighted higher.[8] We therefore set α_{w_j} to be $m + 1 - d_{w_j}$, where m is the window size and d_{w_j} is an integer that represents the distance between w_j and the target word. Hence, this is a decaying weighting. In contrast, with uniform weights, we set α_{w_j} to be 1 for all w_j in the context. We call the score of Equation (1) *standardness*. If the standardness is low, our method regards the instance as non-standard; otherwise, our method regards it as standard. We should note again that, in our method, word embeddings should be learned on a general balanced corpus that is different from the domain of the target instances.

4 Experiment

4.1 Methods for Comparative Evaluation

Our model has three characteristics: (input and output) word embeddings, decaying weights, and a general balanced corpus. We evaluated each of these characteristics in a task distinguishing non-standard usages from standard ones.

First, we verified the effectiveness of the input and output embeddings. We tested a method in which only input embeddings are used to calculate the similarity: the cosine similarity between $v^{IN}_{w_t}$ and $v^{IN}_{w_j}$ instead of $\sigma(v^{IN}_{w_t} \cdot v^{OUT}_{w_j})$, which is a similar framework to that of previous work (Neelakantan et al., 2014; Gharbieh et al., 2016). We then tested a method based on the positive pointwise mutual information (PPMI) (Levy et al., 2015; Hamilton et al., 2016). Here, suppose that *M* is a matrix in which each element is a PPMI of words w_i and w_j. $v^{IN}_{w_t} \cdot v^{OUT}_{w_j}$ in Equation (1) is replaced with the (t, j)-element of the low-rank approximation of *M* obtained through singular value decomposition (SVD). We refer to this model as SVD.

[7] `code.google.com/archive/p/word2vec/`

[8] This weighting scheme is mentioned in (Levy et al., 2015).

Corpus	Description	#word	#token
BCCWJ[9]	Japanese Balanced Corpus	131,913	1.1b
Web[10]	Sentences randomly picked from the Web	336,048	6.0b
Wikipedia[11]	Japanese Wikipedia	1,081,154	8.9b
Newspaper[12]	Japanese Newspapers	1,204,914	15.0b

Table 2: Description of corpora.

Next, we replaced the decaying weights α with uniform weights to examine the impact of decaying weights.

Finally, we conducted experiments with different training corpora to examine the impact of the balanced corpus. We used four corpora as training data for obtaining word embeddings. These corpora are described in Table 2.

4.2 Experimental Settings

In the training of the word embeddings, we set the window size to 5, and the dimensions of the word embeddings to 300. We regarded the words with frequency counts of 5 or less in the training data as unknown words and replaced those words with "<unk>". We used gensim (Řehůřek and Sojka, 2010) as an implementation of SGNS, where we set the number of negative samples to 10. We used the code provided by Levy et al. (2015) as the SVD implementation. For the evaluations, we ranked test instances in ascending order of standardness score and evaluated the ranking in terms of the area under the ROC curve (AUC) (Davis and Goadrich, 2006).

4.3 Results

Table 3 shows the AUC for each model.[13] First, we examined the impact of the choice of training corpus for obtaining word embeddings. The models with BCCWJ are constantly better than those with other corpora, although BCCWJ is smaller than the others (Table 2). This result suggests that use of a balanced corpus is crucial in our method for this task.

corpus	SGNS IN-OUT		SGNS IN-IN		SVD	
	decay	uni	decay	uni	decay	uni
BCCWJ	**.875**	**.870**	.846	.837	.821	.813
Web	.846	.842	.817	.807	.771	.765
Wikipedia	.827	.821	.824	.805	.739	.732
Newspaper	.844	.839	.825	.810	.770	.764

Table 3: Area under the ROC curve (AUC) in usage classification task for each model.

Next, we examined the impact of context embeddings. Table 3 shows that our model (SGNS IN-OUT) with BCCWJ achieved the best AUCs (.875 and .870), better than the AUCs of SGNS IN-IN with BCCWJ (.846 and .837). This result suggests that input embeddings should be used in combination with output embeddings for the task of judging whether a word matches its context or not. Table 3 also shows that SGNS-based models are better than SVD-based models.

As we discussed in Section 3.2, we used two weighting schemes for each model. Although the AUC of each decaying weight model is larger than that of the corresponding uniform weight model, the differences were not statistically significant.

5 Related Work

The previous studies focused on distinguishing non-standard usages that are multi-word expressions or idiomatic expressions (Kiela and Clark, 2013; Salehi et al., 2015; Li and Sporleder, 2010). The task of this research is similar to new sense detection (Cook et al., 2014). Our research target includes jargon, whose actual meaning is difficult to infer without specific knowledge about its usage (Huang and Riloff, 2010). Recent studies in computational linguistics have used word embeddings and other techniques to capture various semantic changes in words, such as diachronic changes, geographical variations, and sentiment changes (Mitra et al., 2014; Kulkarni et al., 2015; Fermann and Lapata, 2016; Eisenstein et al., 2010; Hamilton et al., 2016; Yang and Eisenstein, 2016).

A few researchers have exploited output embeddings for natural language applications such as document ranking (Mitra et al., 2016) and improving language models (Press and Wolf, 2017).

[9] The Balanced Corpus of Contemporary Written Japanese (Maekawa et al., 2010).

[10] Japanese sentences are collected using the method described in (Kawahara and Kurohashi, 2006).

[11] We downloaded Japanese Wikipedia articles in July 2016 from https://dumps.wikimedia.org/jawiki/.

[12] We used editions of the Mainichi Shimbun, Nihon Keizai Shimbun, and Yomiuri Shimbun published from 1994 to 2004.

[13] Although we also conducted experiments with a sigmoid function for the SGNS IN-IN model and with the cosine similarity for the SVD model, their accuracies were worse than those in Table 3.

6 Conclusion

We presented a model that uses context embeddings to distinguish Japanese non-standard usages from standard ones on social media. Our experimental results show that our model is better than the other models tested. They indicate the importance of context embeddings. To sum up, to distinguish non-standard usage, (1) using a balanced corpus as training data for obtaining word embeddings is crucial, (2) exploiting context embeddings derived from *input* and *output* word embeddings of SGNS achieves the best AUC, and (3) decaying weights have little impact on performance.

We are interested in expanding our method for detecting words that have non-standard usages. We are also interested in finding the meanings of the detected non-standard usages.

References

Paul Cook, Jey Han Lau, Diana McCarthy, and Timothy Baldwin. 2014. Novel word-sense identification. In *Proceedings of COLING '14*, pages 1624–1635.

Jesse Davis and Mark Goadrich. 2006. The relationship between precision-recall and roc curves. In *Proceedings of ICML '06*, pages 233–240.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of EMNLP '10*, pages 1277–1287.

Lea Frermann and Mirella Lapata. 2016. A bayesian model of diachronic meaning change. *Transactions of the Association for Computational Linguistics*, 4:31–45.

Waseem Gharbieh, Bhavsar Virendra, and Paul Cook. 2016. A word embedding approach to identifying verb-noun idiomatic combinations. In *Proceedings of the 12th Workshop on Multiword Expressions*, pages 112–118.

William L. Hamilton, Kevin Clark, Jure Leskovec, and Dan Jurafsky. 2016. Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of EMNLP '16*, pages 595–605.

Ruihong Huang and Ellen Riloff. 2010. Inducing domain-specific semantic class taggers from (almost) nothing. In *Proceedings of ACL '10*, pages 275–285.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case frame compilation from the web using highperformance computing. In *Proceedings of LREC '06*, pages 1344–1347.

Douwe Kiela and Stephen Clark. 2013. Detecting compositionality of multi-word expressions using nearest neighbours in vector space models. In *Proceedings of EMNLP '13*, pages 1427–1432.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of WWW '15*, pages 625–635.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of NIPS '14*, pages 2177–2185.

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

Linlin Li and Caroline Sporleder. 2010. Linguistic cues for distinguishing literal and non-literal usages. In *Proceedings of COLING '10*, pages 683–691.

Kikuo Maekawa, Makoto Yamazaki, Takehiko Maruyama, Masaya Yamaguchi, Hideki Ogura, Wakako Kashino, Toshinobu Ogiso, Hanae Koiso, and Yasuharu Den. 2010. Design, compilation, and preliminary analyses of balanced corpus of contemporary written Japanese. In *Proceedings of LREC '10*, pages 1483–1486.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS '13*, pages 3111–3119.

Bhaskar Mitra, Eric T. Nalisnick, Nick Craswell, and Rich Caruana. 2016. A dual embedding space model for document ranking. *arXiv: 1602.01137*.

Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee, and Pawan Goyal. 2014. That's sick dude!: Automatic identification of word sense change across different timescales. In *Proceedings of ACL '14*, pages 1020–1029.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient nonparametric estimation of multiple embeddings per word in vector space. In *Proceedings of EMNLP '14*, pages 1059–1069.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of EACL '17*, pages 157–163.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of Workshop on NewChallenges for NLP Frameworks*, pages 45–50.

Bahar Salehi, Paul Cook, and Timothy Baldwin. 2015. A word embedding approach to predicting the compositionality of multiword expressions. In *Proceedings of NAACL-HLT '15*, pages 977–983.

Aleksandr Sboev. 2016. The sources of new words and expressions in the Chinese internet language and the ways by which they enter the internet language. In *Proceedings of PACLIC '16*, pages 355–361.

Yi Yang and Jacob Eisenstein. 2016. Overcoming language variation in sentiment analysis with social attention. *arXiv:1511.06052*.

Connotation Frames of Power and Agency in Modern Films

Maarten Sap **Marcella Cindy Prasetio** **Ari Holtzman**
Hannah Rashkin **Yejin Choi**
Paul G. Allen School of Computer Science & Engineering
University of Washington, Seattle, USA
{msap,mcp21,ahai,hrashkin,yejin}@cs.washington.edu

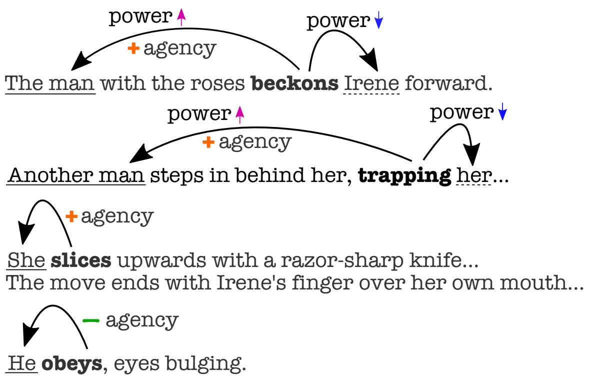
Abstract

The framing of an action influences how we perceive its actor. We introduce connotation frames of *power* and *agency*, a pragmatic formalism organized using frame semantic representations, to model how different levels of power and agency are implicitly projected on actors through their actions. We use the new power and agency frames to measure the subtle, but prevalent, gender bias in the portrayal of modern film characters and provide insights that deviate from the well-known Bechdel test. Our contributions include an extended lexicon of connotation frames along with a web interface that provides a comprehensive analysis through the lens of connotation frames.

1 Introduction

A viewer's impression of a movie character is influenced by how they are written and portrayed, which can in turn influence how people form stereotypes on gender norms (Behm-Morawitz and Mastro, 2008). A character's actions can be projected with varying levels of power and agency, depending on the specific verbs used. For instance, somebody who "accepts" things is implied to be a passive decision-maker (or of lower agency) than somebody who "assesses" things. While not explicitly stated, these connotative meanings projected by different verbs can influence the assumptions the audience makes about the people being described. These assumptions can have negative consequences if they reinforce negative stereotypes (Walton and Spencer, 2009).

To formalize this implicit information about people projected by actions, we introduce *power* and *agency* connotation frames, two new types of predicate-specific connotative relationships as an extension to Rashkin et al. (2016)'s connotation frame lexicon. For instance, in Figure 1, the verb "beckoning" implies that its theme (Irene) has less power than its agent (the man). In the third line, Irene displays strong agency when she "slices" in self-defense. In contrast, when the man "obeys", the man has low implied agency.



The man with the roses **beckons** Irene forward.
Another man steps in behind her, **trapping** her...
She **slices** upwards with a razor-sharp knife...
The move ends with Irene's finger over her own mouth...
He **obeys**, eyes bulging.

Figure 1: An excerpt from a box-office hit, *Sherlock Holmes* (2009). **Bolded** words are the predicates, solid underlined phrases are the agent of the verb, and dash underlined words are the theme. The full example with additional nuanced discussion is available in Figure 6 in the appendix.

Using the new connotation lexicon, we present a quantitative study to reveal the subtle, but prevalent gender[1] bias in modern films. Going beyond the surface level analysis such as screen time or number of female characters (Google, 2017), our study aims for a more focused and precise analysis of power differentials between fictional men and women.

In summary, our major contributions include the creation and release of a lexicon with two new connotative dimensions: *power* and *agency* and an

[1] We acknowledge that gender lies on a spectrum, and reducing it to a male-female binary is simplistic, however our data limits a more complex understanding of gender.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2329–2334
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

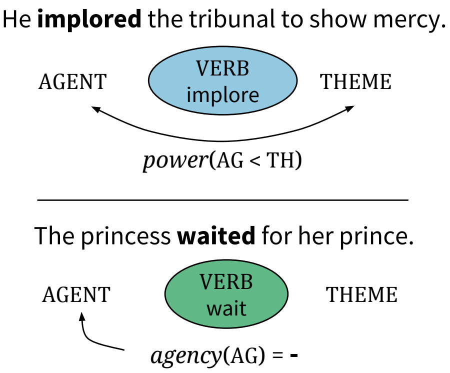

He **implored** the tribunal to show mercy.
AGENT
VERB
implore
THEME
power(AG < TH)
The princess **waited** for her prince.
AGENT
VERB
wait
THEME
agency(AG) = -

Figure 2: The formal notation of the connotation frames of power and agency. The first example shows the relative power differential implied by the verb **"implored"**, i.e., the agent ("he") is in a position of less power than the theme ("the tribunal"). In contrast, "He **demanded** the tribunal show mercy" implies that the agent has authority over the theme. The second example shows the low level of agency implied by the verb **"waited"**.

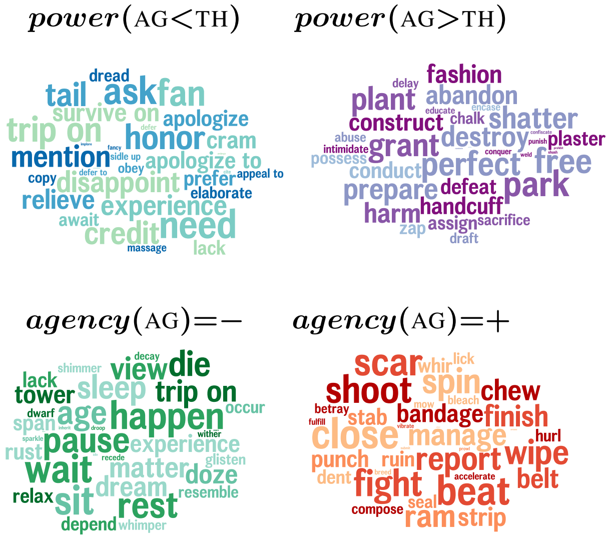


Figure 3: Sample verbs in the connotation frames with high annotator agreement. Size is indicative of verb frequency in our corpus (bigger = more frequent), color differences are only for legibility.

interactive demo website of our findings (see Figure 5 in the appendix for a screenshot).[2] Furthermore, as will be seen in Section 4.1, connotation frames offer new insights that complement and deviate from the well-known Bechdel test (Bechdel, 1986). In particular, we find that high-agency women through the lens of connotation frames are rare in modern films. It is, in part, because some movies (e.g., Snow White) accidentally pass the Bechdel test and also because even movies with strong female characters are not entirely free from the deeply ingrained biases in social norms.

2 Connotation Frames of Power and Agency

We create two new connotation relations, *power* and *agency* (examples in Figure 3), as an expansion of the existing connotation frame lexicons.[3] Three AMT crowdworkers annotated the verbs with placeholders to avoid gender bias in the context (e.g., `X` *rescued* `Y`; an example task is shown in the appendix in Figure 7). We define the annotated constructs as follows:

Power Differentials Many verbs imply the authority levels of the agent and theme relative to one another. For example, if the agent "dominates" the theme (denoted as $power$(AG>TH)), then the agent is implied to have a level of control over the theme. Alternatively, if the agent "honors" the theme (denoted as $power$(AG<TH)), the writer implies that the theme is more important or authoritative. We used AMT crowdsourcing to label 1700 transitive verbs for power differentials. With three annotators per verb, the inter-annotator agreement is 0.34 (Krippendorff's α).

Agency The agency attributed to the agent of the verb denotes whether the action being described implies that the agent is powerful, decisive, and capable of pushing forward their own storyline. For example, a person who is described as "experiencing" things does not seem as active and decisive as someone who is described as "determining" things. AMT workers labeled 2000 transitive verbs for implying high/moderate/low agency (inter-annotator agreement of 0.27). We denote high agency as $agency$(AG)=+, and low agency as $agency$(AG)=−.

Pairwise agreements on a hard constraint are 56% and 51% for power and agency, respectively. Despite this, agreements reach 96% and 94% when moderate labels are counted as agreeing with either high or low labels, showing that annotators rarely strongly disagree with one another. Some contributing factors in the lower KA scores include the subtlety of choosing between neutral

[2] `http://homes.cs.washington.edu/~msap/movie-bias/`.

[3] The lexicons and a demo are available at `http://homes.cs.washington.edu/~msap/movie-bias/`.

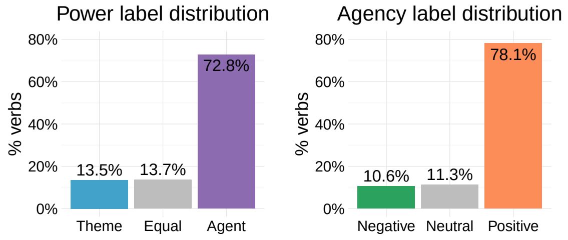


Figure 4: Label distributions for *power* and *agency* based on the crowdsourced annotations.

and positive/negative as well as the skews in the distributions of labels (i.e. more positive than negative labels, see Figure 4). Note that a similar difference between KA scores and soft percent agreement was found in our previous connotation frame work (Rashkin et al., 2016).

3 Bias in Movie Scripts

We use 772 movie scripts from (Gorinski and Lapata, 2015) as a test bed to validate our new connotation frames. Scripts have distinct structure, which allows us to easily parse narrations, dialogues and character names.

We automatically extract 21K male/female characters, using a name-gender list[4] along with gender specific lexicons (e.g., "actor"/"actresses", "duke"/"duchess") to automatically assign gender based on their first three narrations. To identify verbs with characters as their agent, we dependency parse the narratives using the SpaCy[5] parser. Power and agency label distributions in our corpus are consistent with the annotation distribution (Figure 4), and there is little variance across movies (see Figure 8 in the appendix).

In our dataset, there are nearly twice as many men as there are women (34.6% women), in line with previous findings by Smith et al. (2015) and Radford and Gallé (2015). Women are also less present on screen and speak less in movies (Google, 2017). We control for that disparity in all subsequent analyses by including the number of words for each character (standardized) as a covariate. Findings in all the following sections hold when controlling for movie genre (as retrieved from `IMDB.com`), as well as when controlling for effects from individual movies.

Frame	β	gender
$\textit{agency}(\text{AG})=+$	-0.951	**M****
$\textit{power}(\text{AG}>\text{TH})$	-0.468	**M****
$\textit{agency}(\text{AG})=-$	0.277	**F****
$\textit{power}(\text{AG}<\text{TH})$	*not sig.*	

Table 1: Power and agency connotation frames for male and female narratives, controlled for length of narrative text. β represents the change in log-odds of a character being male/female were the corresponding frame to change by one unit. Significant results (** : p<.001) are in bold. "Male" was coded as 0, "Female" as 1.

3.1 Bias in Narratives

Narratives describe what characters are *doing*. We investigate how they vary in terms of *power* and *agency*, using our connotation frames. We measure how each standardized frame metric is associated with the gender of the character through a logistic regression, controlling for the total number of words that the character said, and correcting for multiple comparisons using Holm's correction (Holm, 1979).

Listed in Table 1, our results show that male characters are portrayed with higher level of agency compared to women. Men are also portrayed to have more authority than women as they are more often the agent of powerful verbs.

This suggests that screenwriters tend to have female characters contribute more to the aesthetic of the movie through low-agency verbs, rather than the plot, which is reminiscent of existing gender bias tests for movies (Yehl, 2013).

3.2 Bias in Character Expression

To further our validation of the new connotative dimensions, we look at how characters *express* themselves in movies. Using our connotation frames and LIWC (Tausczik and Pennebaker, 2016), we compile metrics for every character's dialogue. As in subsection 3.1, metrics were standardized for better β interpretability. LIWC results that are not discussed below can be found in the appendix (Tables 4 and 5).

From Table 2, it seems male characters display more power and authority through their speech than their female counterparts do. Specifically, women are written to use more hedges (*# Hedges*) whereas men are written to use more imperative sentences (*# Imperative Sent.*), a finding that re-

[4] `http://www.cs.cmu.edu/Groups/AI/util/areas/nlp/corpora/names/0.html`

[5] `https://spacy.io/`

Frame/metric	β	gender
$agency$(AG)$=-$	0.968	**F****
$agency$(AG)$=+$	-2.177	**M****
$power$(AG$>$TH)	-0.542	**M****
$power$(AG$<$TH)	0.236	**F****
# Imperative Sent.	-0.232	**M****
# Hedges	0.165	**F****
I	0.835	**F****
they	-0.160	**M****
we	-0.361	**M****
you	0.405	**F****
assent	0.202	**F****
space	-1.136	**M****
discrep	0.423	**F****
inhib	-0.171	**M****

Table 2: Gender association with our connotation frames and a subset of LIWC metrics for characters' dialogue, controlled for number of words spoken. All results are significant (** : p<.001).

flects real-world dialogues (Prabhakaran et al., 2014). The usage of imperatives tends to convey power and dominance according to the findings of Bramsen et al. (2011). Along with the fact that female characters tend to agree (*assent*) more than male characters, this corroborates the finding in subsection 3.1 that male characters are generally given more power and agency. Furthermore, male characters use inhibitory language more (*inhib*), which contains words pertaining to blocking or allowing, suggesting that these characters are in positions of power.

Further evidence of power imbalances is found through function words. Women tend to use *I* and *you* pronouns more, whereas men use *we* and *they* pronouns more, echoing real life (Schwartz et al., 2013). Kacewicz et al. (2014) found an association between using "I" pronouns and being lower status, and conversely between "we" pronouns and being higher status. This corroborates the theory that women in movies are generally portrayed with a lower status than men.

Men in movies tend to mention more physical actions (*space* category) whereas women tend talk about what "could" be but isn't (*discrep*; e.g.,"should", "would"). This evokes more commanding connotations for male characters and subordinate connotations for female ones, reinforcing gender stereotypes.

These findings, rooted in previous research, confirm that our connotation frames capture existing bias in how male and female characters display different levels of power and agency in their dialogue.

Metric/Frame		β	P/F
F *dial.*	# Words	10.02	pass**
F *dial.*	$agency$(AG)$=+$	-9.65	fail**
F *dial.*	$power$(AG$>$TH)	2.05	pass*
F *narr.*	$power$(AG$>$TH)	-1.19	fail*

Table 3: Significant correlates of passing the Bechdel test. F: metric for female characters, computed on the dialogues (*dial.*) or on the narratives (*narr.*). * : p<.05; ** : p<.001.

4 Power, Agency and the Bechdel test

A movie passes the Bechdel test (Bechdel, 1986) if it (1) has two (named) female characters, (2) who talk to each other, (3) about something other than a man. While this is a low bar, a surprising number of movies fail at least one of the three criteria. In particular, as many as 42% of the movies in our dataset fail the test according to an online database of the Bechdel scores.[6]

4.1 Beyond the Bechdel Test

We provide comparative insights between the analysis based on connotation frames and the Bechdel test. First, we aggregate our connotation frames, both on dialogue and narration, into movie-level averages per gender. Then, we add features capturing presence of female/male characters (e.g., # F/M words, # F/M characters). Table 3 shows the correlation between passing the Bechdel test and our movie-level connotation frame features using a multivariate logistic regression.

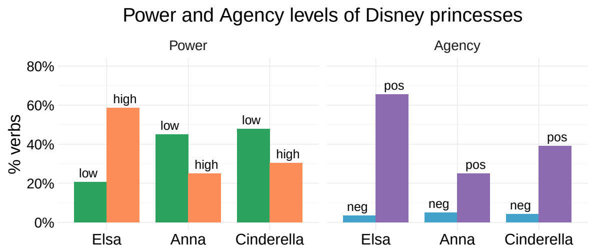
As expected, a movie with more female speaking time is more likely to pass the Bechdel test since it mostly captures female representation. We also find that female characters using agent-empowering verbs, which tend to be more assertive, slightly increases the odds of passing the Bechdel test. Female speakers who use empowering verbs, regardless of the verb's agent, tend to go against the gender-norms of hedging and being less assertive (as we showed in subsection 3.2).

Unexpectedly, movies where women talk with high agency are much less likely to pass the

[6] Available at http://bechdeltest.com. We use this site to obtain ratings for 324 of the movies in our corpus.

Bechdel test. Perhaps these movies typically only show scenes of women interacting in a male-dominated setting. Similarly, the use of more agent-empowering verbs in female narratives decrease the odds of passing the Bechdel test. Chances of two powerful women talking to each other might be lower because movies are less likely to have a lot of powerful women.[7]

Power and Agency of Princesses We further provide a qualitative analysis using Wikipedia plot summaries for movies that are not in our script dataset. Bechdel-passing movies with female protagonists, such as *Frozen* (2013) or *Cinderella* (1950), still perpetuate negative female stereotypes. In *Frozen*, Elsa is portrayed as the only high agency, high power woman, as seen below.[8] Anna and Cinderella, despite also being protagonists, display significantly less power and agency.



The Bechdel test is limited, either by being too inclusive of movies who portray women in non-authoritative, passive positions or by excluding movies that have strong women with agency, who just happen not to talk to each other about something besides men. Our extensions to the connotation frame lexicon provide finer grained information about how women are portrayed through their expression and their actions, which can act complementary to measures of their presence.

5 Related work

There is much prior research focus on bias in social media (Garcia et al., 2014; Prabhakaran et al., 2014; Ratkiewicz et al., 2011; Yano et al., 2010; Srivastava and Sahami, 2009), complementing our investigation on movies. Fast et al. (2016) examine the stereotypes present in fan-fiction using a lexicon-based strategy that focus on commonly gender-biased attributes (e.g., *emotional* for women) rather than the overall power dynamics of the story. In a similar vein, Ramakrishna et al. (2015) learn word-level “gender ladenness” features by looking at the neighbors of 925 manually annotated words.

There exist various sets of high-level criteria to assess gender bias of character portrayal in fiction (Yehl, 2013; Romano, 2013; Powers, 2016). Agarwal et al. (2015), in particular, automate the Bechdel test using social network features, finding that women are less central to the plot in movies that fail it. We compare our linguistic analysis of power and agency with the Bechdel test, demonstrating the need for more fine-grained analysis of how gender is depicted in movies.

Close in spirit to our investigation, Schofield and Mehr (2016) train a number of classifiers over movie scripts for determining the gender of individual (and pairs) of speakers as well as the expected length of their relationships. In contrast, we focus on understanding how the gender of a given character implicitly relates to features that track their control over their own path (agency) and the world around them (power).

6 Conclusion

We created and released new connotation frames of *power* and *agency*, allowing for more nuanced writing analysis than previously possible. We validate our new frames through a case study on movie scripts. Specifically, we analyze differences in power and agency for male and female characters, and compare these dimensions to the Bechdel test. Our connotation frames confirm evidence of imbalances in gender portrayal in movies.

Acknowledgments

We would like to thank Li Zilles for collecting the dataset, and Sydney Rubin and Laura Vianna for their input on the gender analyses. We would also like to thank anonymous reviewers for providing insightful feedback. This material is based upon work supported in part by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1256082, in part by NSF grants IIS-1524371, IIS-1714566, and gifts by Google and Facebook.

[7] Similar observations may have inspired the Mako Mori test (Romano, 2013), which looks at whether there’s a female character with a story arc that doesn’t support a man’s.

[8] Note that plot summaries are more biased toward active verbs, which explains the low negative agency for all characters.

References

Apoorv Agarwal, Jiehan Zheng, Shruti Kamath, Sriramkumar Balasubramanian, and Shirin Ann Dey. 2015. Key female characters in film have more to talk about besides men: Automating the bechdel test. In *HLT-NAACL*.

Alison Bechdel. 1986. *Dykes to watch out for*. Firebrand Books.

Elizabeth Behm-Morawitz and Dana E Mastro. 2008. Mean girls? the influence of gender portrayals in teen movies on emerging adults' gender-based attitudes and beliefs. *Journalism & Mass Communication Quarterly*, 85(1):131–146.

Philip Bramsen, Martha Escobar-Molano, Ami Patel, and Rafael Alonso. 2011. Extracting social power relationships from natural language. In *ACL*.

Ethan Fast, Tina Vachovsky, and Michael S Bernstein. 2016. Shirtless and dangerous: Quantifying linguistic signals of gender bias in an online fiction writing community. In *ICWSM*.

David Garcia, Ingmar Weber, and Venkata Rama Kiran Garimella. 2014. Gender asymmetries in reality and fiction: The bechdel test of social media. In *ICWSM*.

Google. 2017. Using technology to address gender bias in film. https://www.google.com/about/main/gender-equality-films/index.html.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *NAACL*.

Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70.

Ewa Kacewicz, James W Pennebaker, Matthew Davis, Moongee Jeon, and Arthur C Graesser. 2014. Pronoun use reflects standings in social hierarchies. *Journal of Language and Social Psychology*, 33(2):125–143.

Kelsey Powers. 2016. The furiosa test developed from female presence in mad max: Fury road. http://www.calvin.edu/chimes/2015/11/09/the-furiosa-test-developed-from-female-presence-in-mad-max-fury-road.

Vinodkumar Prabhakaran, Emily E Reid, and Owen Rambow. 2014. Gender and power: How gender and gender environment affect manifestations of power. In *EMNLP*.

Will Radford and Matthias Gallé. 2015. Roles for the boys?: Mining cast lists for gender and role distributions over time. In *WWW*.

Anil Ramakrishna, Nikolaos Malandrakis, Elizabeth Staruk, and Shrikanth S Narayanan. 2015. A quantitative analysis of gender differences in movies using psycholinguistic normatives. In *EMNLP*.

Hannah Rashkin, Sameer Singh, and Yejin Choi. 2016. Connotation frames: A Data-Driven investigation. In *ACL*.

Jacob Ratkiewicz, Michael Conover, Mark R Meiss, Bruno Gonçalves, Alessandro Flammini, and Filippo Menczer. 2011. Detecting and tracking political abuse in social media. *ICWSM*.

Aja Romano. 2013. The mako mori test: 'pacific rim' inspires a bechdel test alternative. https://www.dailydot.com/parsec/fandom/mako-mori-test-bechdel-pacific-rim/.

Alexandra Schofield and Leo Mehr. 2016. Gender-distinguishing features in film dialogue. In *CLfL@ NAACL-HLT*.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin E P Seligman, and Lyle H Ungar. 2013. Personality, gender, and age in the language of social media: the open-vocabulary approach. *PLoS One*, 8(9):e73791.

Stacy L Smith, Marc Choueiti, Katherine Pieper, Traci Gillig, Carmen Lee, and Dylan DeLuca. 2015. Inequality in 700 popular films: Examining portrayals of gender, race & lgbt status from 2007 to 2014. *Media, Diversity & Social Change Initiative, USC Annenberg School for Communication and Journalism*.

Ashok N Srivastava and Mehran Sahami. 2009. *Text mining: Classification, clustering, and applications*. CRC Press.

Yla R Tausczik and James W Pennebaker. 2016. The psychological meaning of words: LIWC and computerized text analysis methods. *J. Lang. Soc. Psychol.*

Gregory M Walton and Steven J Spencer. 2009. Latent ability: Grades and test scores systematically underestimate the intellectual ability of negatively stereotyped students. *Psychological Science*, 20(9):1132–1139.

Tae Yano, Philip Resnik, and Noah A Smith. 2010. Shedding (a thousand points of) light on biased language. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

Joshua Yehl. 2013. Kelly sue deconnick talks captain marvel, pretty deadly, and the sexy lamp test. http://www.ign.com/articles/2013/06/20/kelly-sue-deconnick-talks-captain-marvel-pretty-deadly-and-the-sexy-lamp-test.

Controlling Human Perception of Basic User Traits

Daniel Preoţiuc-Pietro and **Sharath Chandra Guntuku** and **Lyle Ungar**
Positive Psychology Center
University of Pennsylvania
{danielpr@sas,sharathg@sas,ungar@cis}.upenn.edu

Abstract

Much of our online communication is text-mediated and, lately, more common with automated agents. Unlike interacting with humans, these agents currently do not tailor their language to the type of person they are communicating to. In this pilot study, we measure the extent to which human perception of basic user trait information – gender and age – is controllable through text. Using automatic models of gender and age prediction, we estimate which tweets posted by a user are more likely to mis-characterize his traits. We perform multiple controlled crowdsourcing experiments in which we show that we can reduce the human prediction accuracy of gender to almost random – an over 20% drop in accuracy. Our experiments show that it is practically feasible for multiple applications such as text generation, text summarization or machine translation to be tailored to specific traits and perceived as such.

1 Introduction

Advances in Natural Language Processing are leading to a point when text generation methods are deployed at scale. However, in the quest to make these applications more likable, effective and hence more usable, these methods should consider a way to adapt themselves to the person or type of persons they are interacting with (Bates, 1994; Loyall and Bates, 1997) e.g., a student may learn better from a tutoring agent that expresses similar traits to himself (Baylor and Kim, 2004).

In this study, we explore the feasibility of controlling human perception of traits using automated methods. Flekova et al. (2016); Carpenter et al. (2016) are the first to study the difference between user traits and their perception by external raters using tweets from social media. Their focus was on quantifying differences between perception and reality and analyzing text features which lead to mis-perception. This study goes a step further, and using the same experimental design and crowdsourcing, aims to use automatic methods to control human perception of basic user traits – here age and gender – through tweets. To this end, we use gender and age prediction algorithms to select tweets posted by users with a known trait with the goal of increasing or decreasing human rater accuracy in guessing their traits.

Obfuscating gender as identified by an automatic classifier was attempted in (Reddy and Knight, 2016). This problem is related, but very different to ours as we study human perception which is both different (Flekova et al., 2016) and more complex. Reddy and Knight (2016) study a range of lexical substitutions that can be performed in order to decrease the prediction accuracy of a classifier, although acknowledging that these may affect lexical coherence. In this pilot study, we circumvent this problem by using tweets known to have been written by the same person, with the downside of possible topic confounds.

Our experiments show that, for gender, we can decrease the human accuracy in perceiving gender from text by more than 20% as compared to a random selection of their tweets, with accuracy in this case being only slightly higher than chance. Further, this accuracy is even lower when predicting males. For age perception, we show consistent results in altering perception as both younger or older, albeit for relatively smaller age differences.

Applications of our proposed line of research include conversational agents or automated e-mail generation. Personalization was motivated in the context of machine translation (Mirkin et al.,

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2335–2341
Copenhagen, Denmark, September 7–11, 2017.

2015) and recently attempted for gender (Rabinovich et al., 2017), even though the authors do not use humans to evaluate perception of gender. Automatic text personalization to user traits can also go beyond basic demographics to more salient ones such as social status (Preoţiuc-Pietro et al., 2015a,b), political ideology (Preoţiuc-Pietro et al., 2017a) or psychological traits such as personality (Schwartz et al., 2013; Guntuku et al., 2015a,b, 2016, 2017), narcissism (Preotiuc-Pietro et al., 2016), trust or empathy (Abdul-Mageed et al., 2017).

2 Data Set

We study two user traits through two Twitter data sets containing users with known gender and age information. First, for gender, we use a subset of 200 users (100 males, 100 females) of the data set collected by (Burger et al., 2011) and released by (Volkova et al., 2013) which mapped users to their gender by linking their Twitter account to their publicly self-declared gender on related blogs. The age data set consists of 200 users that self-reported their age in a survey and disclosed their public Twitter data that are part of a larger set used in (Flekova et al., 2016). The users are chosen to have an age in the 15–34 year old interval in the year 2015 and we only use tweets posted in 2015 in our analysis. We selected exactly 10 users of each age in this interval, as these are the most frequent ages present in our data set, most language variation happens in this interval and these are the age range which raters can most accurately predict (Nguyen et al., 2014).

We use the Twitter API to download up to 3200 tweets from these users. We pre-process tweets by filtering those not written in English as detected by an automated method (Lui and Baldwin, 2012), removing duplicate tweets (i.e., having the same first 6 tokens) and removing re-tweets as these are not authored by the user. All potentially sensitive or revealing information contained in tweets such as URL's, usernames, @-mentions, phone numbers were removed and replaced with placeholders before shown to annotators. Other than publicly available tweets, no other metadata or information was presented with the task, so raters were not able to map the tweets to actual user identities. The raters were also unaware of the conditions (Random, Opposite, Same, Youngest or Oldest) they were assigned to when performing the ratings. All our experiments received approval from Institutional Review Board (IRB) of the University of Pennsylvania.

We are aware that the proposed long-term applications we envision for this research can have personal impact on users. Hence, we propose following criteria which should be at the core of future research in controlling human perception, which we encourage to be completed over time:

- **Transparency**: data trained to build the personalized models should be transparent to any user. This would allow to observe any possible biases that may exist in the data.
- **Control**: the user interacting with a personalized system should be aware of the type of personalization employed by the agent (e.g. by gender, by which particular age group) and should be able to disable it when desired.

3 Experimental Setup

We use Amazon Mechanical Turk to create crowdsourcing tasks for predicting age and gender from tweets. Each HIT consists of 20 tweets authored by a single user and selected using different methods. The annotators were asked to predict gender (M/F) or age (integer value in 13–90) and rate their confidence of their guess from 1 (not at all confident) to 5 (very confident). We collected 3 annotations for each author and set of tweets.

Participants received a small compensation (.04$) for each rating and could repeat the task as many times as they wished, but never for the same authors and set of tweets. They were also presented with an initial bonus (.25$). For quality control, the participants underwent a short training and qualification questions, their location was limited to the US and they had to spend at least 10 seconds on each HIT before they were allowed to submit their guess.

In order to estimate which tweets are more likely to be written by females or a older user, we use the classifier introduced in (Sap et al., 2014). This is a regularized Linear SVM that obtains state-of-the-art prediction results on user gender (91.9% accuracy) and age ($r = .835$) prediction from social media text. We apply the model to all our tweets and select for each user 20 tweets based on the following criteria:

- **Random**: tweets chosen at random from a user's timeline;
- **Opposite**: for gender, tweets that are predicted

	Single Rating	Majority Vote
Baseline	50%	50%
Opposite	55.74%	61.57%
Random	76.67%	83.99%
Same	91.33%	95.49%

Table 1: Human accuracy in gender perception experiments in the three text selection conditions.

as more likely to be written by someone of the different gender;

- **Same**: for gender, tweets predicted to be written by someone of the same gender as the author.
- **Youngest**: for age, tweets from a user that are predicted as youngest age;
- **Oldest**: for age, tweets from a user that are predicted as oldest age;

The tweets selected based on the automatic prediction are presented in the order of prediction scores e.g. tweets for **Youngest** are sorted with the lowest predicted age being shown at the top of the list. Experiments with random ordering of tweets showed similar results.

4 Results

In this section we analyze the extent to which our experiments manage to alter trait perception, the errors and confidence of the annotations.

4.1 Gender

Overall accuracy results for our gender experiments are presented for both individual ratings and majority vote in Table 1. In all experimental setups, the raters were able to guess gender better than chance, with the majority vote of the three raters higher by a significant margin (5.77% on average) than the individual votes.

Our selection procedure has great impact on rater accuracy. Selecting tweets most likely to be written by the opposite gender – even if they are posted by the same user in reality – has an impact of decreasing the individual rater accuracy by 20.93% to only slightly above random guess (55.75%). For the majority vote ratings, the decrease is 22.42% (paired T-test, $t = 8.06$, $p < 10^{-14}$). On the other hand, selecting the tweets that are most likely to be posted by a user from the same gender as determined by our automatic model has the impact of increasing the individual rater accuracy by 14.66%. The majority vote prediction is increased by a relatively smaller amount (11.5% – paired T-test, $t = 7.09$, $p < 10^{-11}$), which we attribute to the accuracy being very close to oracle performance.

The confusion matrices from the three experiments are presented in Table 2. A couple of patterns stand out: females are easier to be accurately identified in all three experiments and males are more likely to be confused for females than vice-versa. This resulted in raters guessing more users to be females, despite our data set being balanced. Intriguingly, in the **Opposite** experiment, males were more often confused for females than correctly guessed, with females being guessed far more accurately, making the average accuracy better than chance. In the **Same** experiment, females are again easier to guess, with accuracy being very close to perfect. These results show that females are more distinctive in their language use on Twitter and thus are harder to be confused for males. On the other hand, as proven by the **Opposite** experiment, posts written by males can be selected such that they are perceived as written by females.

The inter-annotator agreement is presented in Table 4. Pairwise agreement at a user level is very high for the **Same** setup, decreasing significantly for the **Random** and **Opposite** setups.

The average self-rated confidence in assessments for the three experiments are presented in Table 3. Self-rated confidence mirrors almost perfectly the accuracy scores in all experiments and cases: confidence is higher on average in cases when accuracy is higher. Users are in general more confident when accurately guessing a female, and are least accurate when inaccurately guessing a female. Noteworthy, in the **Opposite** experiment, users who incorrectly guessed males were more confident than when correctly identifying males, which is not the case for females. This further shows that females are use more distinctive language on Twitter, while males could be more easily mistaken for females.

4.2 Age

Overall accuracy results for our age experiments are presented in Table 5. We only report results with a user age computed as the average three guesses. Results with individual ratings are very similar and we omit them for brevity.

The experiments show that our model's selection matches human perception: in the **Younger** experiment, the average predicted age is lower than in the **Random** experiment, which is in turn lower on average than the predicted age in the **Oldest** setup. Further, in the **Younger** experiment, many

(a) Random tweets (Acc. 76.66%).

		Predicted	
		Male	Female
Real	Male	35.99%	14%
	Female	9.33%	40.67%

(b) Opposite Tweets (Acc. 55.73%).

		Predicted	
		Male	Female
Real	Male	23.47%	26.35%
	Female	17.9%	32.26%

(c) Same Tweets (Accuracy 91.33%).

		Predicted	
		Male	Female
Real	Male	43.5%	6.5%
	Female	2.16%	47.83%

Table 2: Normalized confusion matrices of human guesses (**Predicted**) compared to ground truth (**Real**).

(a) Random tweets (Average 3.37).

		Predicted	
		Male	Female
Real	Male	3.22	2.92
	Female	2.78	3.80

(b) Opposite Tweets (Average 3.44).

		Predicted	
		Male	Female
Real	Male	3.24	3.57
	Female	3.14	3.65

(c) Same Tweets (Average 3.85).

		Predicted	
		Male	Female
Real	Male	3.70	2.76
	Female	2.76	4.18

Table 3: Average confidence of human guesses (**Predicted**) depending on ground truth (**Real**).

(a) Gender

	Cohen's κ	Agreement
Opposite	.252	64.3%
Random	.354	68%
Same	.764	88.3%

(b) Age

	St. Dev.	Pearson r
Youngest	3.40	.368
Random	4.06	.170
Oldest	4.89	.341

Table 4: Inter-annotator agreement statistics.

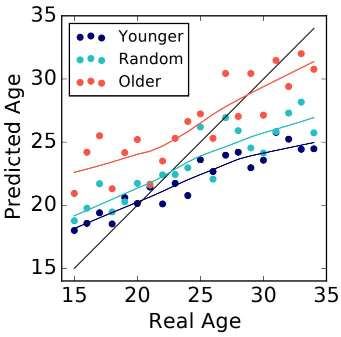


Figure 1: The average predicted ages compared to real age in the three experiments. The black line represents the ideal fit, the colored lines represent a LOESS fit to the data.

more users' age is under-estimated as compared to when predicting average age and in the **Older** experiment, more users' age is over-estimated. We also note that in the **Random** setup, raters tend to under-estimate age (53.5% younger vs. 39.3% older), with the mean being lower than in the data (23.3 vs. 24.5), which aligns with previous research (Nguyen et al., 2014).

Figure 1 plots the average prediction for users by age in the three experiments. Intriguingly, even in the **Younger** setup, users under 18 y.o. are predicted as older, while the groups of users over 20 y.o. are all under-predicted. Notably, the same near-linear pattern largely holds for the other two experiments, with the age cut-off being different (23 for **Random**, 27 for **Oldest**).

The accuracies of the three experiments are very similar regardless if comparing the number of correct guesses or guesses within 1, 3 of 5 years of the actual age. By examining Figure 1, we realize that the set of users accurately predicted shifts from one method to the other. This highlights that, even if controlling age perception is feasible, this is possible only for a difference of a few years.

The inter-annotator agreement is presented in Table 4. First, the average standard deviation across the three guesses for each author shows that **Youngest** setup generates the most similar guesses, which tend to be in the younger age range. In contrast, the **Oldest** setup generates the largest variance in guesses. Average Pearson correlation between the three guesses per author shows that both controlled setups result in higher agreement between raters than the **Random** setup, which shows that users are easier to rank by age based on their extreme language use (**Oldest** or **Youngest**) compared to a random tweet sample.

Finally, the average self-confidence of the ratings is highest in the **Youngest** experiment ($\mu = 3.35$), followed by the **Older** experiment ($\mu = 3.20$) with the **Random** experiment ($\mu = 2.97$) lowest. Further, we checked if there is a relationship between true or predicted age and self-rated confidence. In the **Youngest** experiment, both true age and predicted age are negatively correlated with self-rated confidence (true age: Pearson $r = -0.218$, p-value $< 10^{-8}$, predicted age: Pearson $r = -.246$, p-value $< 10^{-10}$), showing that raters believe their guess is easier when encountering younger users. In the **Random** experiment, a significant correlation exists between self-rated confi-

	Mean	σ	Median	Correct	Younger	Older	$\leq$1 Year	$\leq$3 Years	$\leq$5 Years
Baseline	24.0	0.0	24.0	5.0%	40.0%	45.0%	15.0%	35.0%	55.0%
Youngest	22.0	4.4	21.6	7.6%	62.2%	30.1%	20.7%	44.3%	59.0%
Random	23.3	4.8	22.8	7.1%	53.5%	39.3%	21.3%	42.8%	61.3%
Oldest	26.4	5.7	26.0	7.5%	36.2%	56.2%	22.0%	41.5%	60.6%

Table 5: Age prediction results in the three experimental setups. The predicted user age is the average age of the three human ratings. The **Baseline** represents always selecting the average age in our data set.

Gender			
Opposite (M)		Opposite (F)	
dress	Just saw a **dress** style i can only describe as [...]	**his**	Chinese men amputated **his** own leg: URL
herself	What's USER doing on The Voice and why is she calling **herself** 'James'	**wife**	The Good **Wife** exists to squeeze every last ounce of sincerity and hope out of your soul.
husband	.USER USER Wait ... you're meeting your **husband** in Cleveland?	**haircut**	Right. **Haircut** when I get home.
women's	Forget **women's** rights or voters rights [...]	**burger**	Best **burger** anywhere. [...]
him	Glad I was able to send **him** to the heartbreak hotel...	**of**	I spend 99% **of** my awake time thinking about food I can't eat.

Age			
Youngest (–)		Oldest (+)	
literally	that's **literally** living the dream	**daughter**	[...] USER makes my 2-month-old **daughter** stop fussing :)
so	im laughing **so** hard	**years**	i love [...] regretting everything from like three **years** ago
though	You cute **though**	**via**	Christmas is almost here! Let's party! URL **via** USER
excited	IM SO **EXCITED** URL	**ago**	One year **ago** today URL URL
guys	you **guys** killed it USER	**ok**	I'm **OK** with that. URL

Table 6: Most impactful features in tweet selection and representative tweet.

dence and predicted age only (Pearson $r = -.172$, p-value $< 10^{-5}$), while we found no relationship in the **Oldest** experiment. This indicates that language use at least apparently is more distinguishable for younger users, probably due to specific topics or interests.

5 Qualitative Analysis

Finally, we show in Table 6 the top features that impact selection of tweets in representative setups from this paper together with a representative tweet. The top features are computed by multiplying the regression/classification weight with the user-normalized average frequency of the feature in the displayed tweets. For gender, we use the **Opposite** setup to show words most indicative of females present in tweets selected and written by males and viceversa. Gender specific features are used with different senses than usual ('dress', 'wife', 'women's'), in reference to other persons rather than oneself ('herself', 'his'), or represent stylistic ('of') or topical ('haircut', 'burger') differences. For age, we select the feature most indicative of a younger user in the **Youngest** setup and the ones most indicative of older age in the **Oldest** setup. In this case, most of the top words are stylistic ('literally', 'so', 'though', 'excited, 'guys', 'ok', 'via') with features indicative of older age referencing the past ('years', 'ago') or generally specific of older age ('daughter').

6 Conclusions

We have presented the first study into automatically controlling human perception of written text. Our exploration used gender and age as basic human traits, which most have a good level of knowledge about, to measure the extent to which altering perception in text-mediated communication is feasible. Our results showed that this is possible to some extent, being especially accurate for males. Age experiments demonstrated consistent results across the three experiments, although alteration seems possible only for relatively small age deltas.

In this first experiments on this topic, we chose to perform tweet selection rather than generation, as these methods often generate text that is not semantically and syntactically correct or natural for a reader. In future work, we will experiment with automatically altering or generating text while keeping topic constant, as our current results are in part topically driven. Alterations can be performed through stylistic transformations such as normalization or by using paraphrasing as suggested in (Preoţiuc-Pietro et al., 2016, 2017b).

Text adaptation is especially important for conversational agents that interact only through text. As humans, we automatically perform this adaptation through multiple additional channels: speech tone, frequency, facial expression; which the agent can not alter. In addition to methodology, future work will also need to take into account ethical implications of this personalization.

Acknowledgments

This project/publication was made possible through the support of a grant from Templeton Religion Trust (TRT-0048). The opinions expressed in this publication are those of the author(s) and do not necessarily reflect the views of the Templeton Religion Trust.

References

Muhammad M. Abdul-Mageed, Anneke Buffone, Hao Peng, Giorgi Salvatore, Johannes Eichstaedt, and Lyle Ungar. 2017. Recognizing Pathogenic Empathy in Social Media. In *Proceedings of the Eleventh International AAAI Conference on the Web and Social Media*, ICWSM, pages 448–451.

Joseph Bates. 1994. The Role of Emotion in Believable Agents. *Communications of the ACM*, 37(7):122–125.

Amy L Baylor and Yanghee Kim. 2004. Pedagogical Agent Design: The Impact of Agent Realism, Gender, Ethnicity, and Instructional Role. *Intelligent Tutoring Systems. ITS 2004. Lecture Notes in Computer Science*, 3220:592–603.

D. John Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating Gender on Twitter. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 1301–1309.

Jordan Carpenter, Daniel Preoţiuc-Pietro, Lucie Flekova, Salvatore Giorgi, Courtney Hagan, Margaret Kern, Anneke Buffone, Lyle Ungar, and Martin Seligman. 2016. Real Men don't say 'cute': Using Automatic Language Analysis to Isolate Inaccurate Aspects of Stereotypes. *Social Psychological and Personality Science*, 8:310–322.

Lucie Flekova, Jordan Carpenter, Salvatore Giorgi, Lyle Ungar, and Daniel Preoţiuc-Pietro. 2016. Analyzing Biases in Human Perception of User Age and Gender from Text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL, pages 843–854.

Sharath Chandra Guntuku, Weisi Lin, Jordan Carpenter, Wee Keong Ng, Lyle H Ungar, and Daniel Preoţiuc-Pietro. 2017. Studying personality through the content of posted and liked images on twitter. In *Proceedings of the 2017 ACM on Web Science Conference*, pages 223–227. ACM.

Sharath Chandra Guntuku, Weisi Lin, Michael James Scott, and Gheorghita Ghinea. 2015a. Modelling the influence of personality and culture on affect and enjoyment in multimedia. In *Affective Computing and Intelligent Interaction (ACII), 2015 International Conference on*, pages 236–242. IEEE.

Sharath Chandra Guntuku, Lin Qiu, Sujoy Roy, Weisi Lin, and Vinit Jakhetiya. 2015b. Do others Perceive you as you want them to?: Modeling Personality based on Selfies. In *Proceedings of the 1st International Workshop on Affect & Sentiment in Multimedia*, pages 21–26. ACM MM.

Sharath Chandra Guntuku, Joey T Zhou, Sujoy Roy, Lin Weisi, and Ivor W Tsang. 2016. Who likes What, and Why? Insights into Personality Modeling based on Image 'Likes'. *IEEE Transactions on Affective Computing*, PP:1–14.

A Bryan Loyall and Joseph Bates. 1997. Personality-rich Believable Agents that use Language. In *Proceedings of the First International Conference on Autonomous Agents*, AGENTS, pages 106–113.

Marco Lui and Timothy Baldwin. 2012. Langid.Py: An Off-the-shelf Language Identification Tool. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (System Demonstrations)*, ACL, pages 25–30.

Shachar Mirkin, Scott Nowson, Caroline Brun, and Julien Perez. 2015. Motivating Personality-aware Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 1102–1108.

Dong-Phuong Nguyen, Dolf Trieschnigg, A. Seza Doğruöz, Rilana Gravel, Mariët Theune, Theo Meder, and Franciska de Jong. 2014. Why Gender and Age Prediction from Tweets is Hard: Lessons from a Crowdsourcing Experiment. In *Proceedings of the 25th International Conference on Computational Linguistics*, COLING, pages 1950–1961.

Daniel Preoţiuc-Pietro, Jordan Carpenter, and Lyle Ungar. 2017a. Beyond Binary Labels: Political Ideology Prediction of Twitter Users. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, ACL.

Daniel Preoţiuc-Pietro, Jordan Carpenter, and Lyle Ungar. 2017b. Personality Driven Differences in Paraphrase Preference. In *Proceedings of the Workshop on Natural Language Processing and Computational Social Science (NLP+CSS)*, ACL.

Daniel Preoţiuc-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015a. An Analysis of the User Occupational Class through Twitter Content. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics*, ACL, pages 1754–1764.

Daniel Preoţiuc-Pietro, Svitlana Volkova, Vasileios Lampos, Yoram Bachrach, and Nikolaos Aletras. 2015b. Studying User Income through Language, Behaviour and Affect in Social Media. *PLoS ONE*, 10(9).

Daniel Preoţiuc-Pietro, Wei Xu, and Lyle Ungar. 2016. Discovering User Attribute Stylistic Differences via Paraphrasing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI, pages 3030–3037.

Daniel Preotiuc-Pietro, Jordan Carpenter, Salvatore Giorgi, and Lyle Ungar. 2016. Studying the Dark Triad of Personality using Twitter Behavior. In *Proceedings of the 25th ACM Conference on Information and Knowledge Management*, CIKM.

Ella Rabinovich, Shachar Mirkin, Raj Nath Patel, Lucia Specia, and Shuly Wintner. 2017. Personalized Machine Translation: Preserving Original Author Traits. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, EACL, pages 1074–1084.

Sravana Reddy and Kevin Knight. 2016. Obfuscating Gender in Social Media Writing. In *Workshop on Natural Language Processing and Computational Social Science (NLP for CSS)*, EMNLP, pages 17–26.

Maarten Sap, Gregory Park, Johannes Eichstaedt, Margaret Kern, Lyle Ungar, and H Andrew Schwartz. 2014. Developing Age and Gender Predictive Lexica over Social Media. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 1146–1151.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, and Lyle H Ungar. 2013. Personality, Gender, and Age in the Language of Social Media: The Open-vocabulary Approach. *PLoS One*, 8.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring Demographic Language Variations to Improve Multilingual Sentiment Analysis in Social Media. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 1815–1827.

Topic Signatures in Political Campaign Speeches

Clément Gautrais[1], Peggy Cellier[2], René Quiniou[3], and Alexandre Termier[1]

[1]University of Rennes 1, IRISA, France
[2]INSA Rennes, IRISA, France
[3]Inria Rennes, IRISA, France

Abstract

Highlighting the recurrence of topics usage in candidates speeches is a key feature to identify the main ideas of each candidate during a political campaign. In this paper, we present a method combining standard topic modeling with *signature mining* for analyzing topic recurrence in speeches of Clinton and Trump during the 2016 American presidential campaign. The results show that the method extracts automatically the main ideas of each candidate and, in addition, provides information about the evolution of these topics during the campaign.

1 Introduction

Political discourse analysis (Van Dijk, 1998) is a branch of discourse analysis that aims at expliciting from speeches or debates the salient features of political discourses. From that point of view, a presidential election provides interesting datasets to study. Indeed, it is a major political event in a country and gives rise to many political meetings where candidates discuss personally selected societal problems and detail their own solutions. In that context, the identification of the favourite topics of candidates as well as how they evolve throughout the campaign is a crucial task.

In (Savoy, 2010), the author presents an analysis of the evolution of topics in political speeches by comparing the words that are overused or underused by Obama and McCain during the 2008 US presidential campaign. The dynamics of these particular words usage is analyzed over monthly periods to identify the underlying dynamics of the campaign topics. A limitation of this approach is that the period is fixed (monthly) whereas predictable (vote, debates) or unpredictable (scandals) events usually give the rhythm to a political campaign. Other work used topic modeling methods, such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003), Latent Semantic Anlayis (LSA) (Landauer et al., 1998) or Non-negative Matrix Factorization (NMF) (Lee and Seung, 1999), to study political texts (Prabhakaran et al., 2014; Quinn et al., 2010). For instance in (Quinn et al., 2010), a topic model for legislative speech is defined. However, those works study topics one at a time whereas a set of co-referenced topics is more relevant since it constitutes the core of a candidate's political program. There exist other works about political text analysis (Calvet and Véronis, 2008) but they focus on the use of predefined single words in speeches over time, whereas we aim at finding (without knowing in advance which topic are recurrent) the recurrent topics (multiple topics) usage over time.

In this paper, we propose to identify in political speeches the favourite topics considered by each candidate as well as how and when they evolve throughout the campaign. In our opinion, this gives critical clues to identify and to explain each candidate's main ideas and their evolution. Thus, we describe an approach to extract the *topic signature* of a candidate from her/his political speeches, i.e. the set of topics discussed by some candidate over time. The method associates NMF, a standard topic modeling technique (Lee and Seung, 1999), with *signature mining* (Gautrais et al., 2017) to analyze the speeches of Hillary Clinton and Donald Trump during the 2016 US presidential campaign. The advantages of this approach are twofold. First, the set of campaign speeches is modeled with *topic signatures*, i.e. recurrent topics occurring at a flexible periodicity during the campaign, instead of sets of specific words occurring at a fixed periodicity. The topic signature provides a more abstract view of each candidate's

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2342–2347
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

main ideas and propositions. Second, the *signature mining* technique automatically adapts the periodicity to the campaign rhythms, to provide a better insight of the *campaign dynamics*.

2 Topic Signature Model

To model recurrent topics in a political campaign, we use the signature model (Gautrais et al., 2017). This model was originally developed to capture the recurrent purchase behavior of retail customers. The analogy between politics and retail is that customers' purchases consist of regularly bought products, and, similarly, politicians' speeches contain recurrent topics.

We consider a set of topics ($\mathcal{W}$) and a sequence of speeches (α) such that $\alpha = \langle (t_1, S_1),\ (t_2, S_2) \dots (t_n, S_n) \rangle$ where $\forall i \in [1, n]$, $S_i \subseteq \mathcal{W}$ and t_i gives the timestamp of S_i. For instance, in Figure 1, $\mathcal{W} = \{a, b, c, d, e\}$ and α is a sequence of seven speeches displayed in chronological order. We see that during Speech S3, two topics were addressed: *b* and *d*.

A *k-segmentation* of a sequence of speeches α, $P(\alpha, k) = \langle E_1 \dots E_k \rangle$, is a sequence of k non-overlapping consecutive sub-sequences of α, E_i, called *episodes*, each consisting of consecutive speeches. An example of a 4-segmentation is given in Figure 1, the first episode $E1$ contains 3 speeches ($S1$, $S2$, $S3$), $E2$ contains 2 speeches ($S4$, $S5$), $E3$ contains speech $S6$ and $E4$ contains speech $S7$. This segmentation contains episodes of different sizes, in both number of speeches and time span. This flexibility of the model allows for adapting the episodes size to the sequence rhythm.

A *topic k-signature*, $Rec(\alpha, k)$, is defined as a *maximal* set of recurrent topics in a k-segmentation of α. Roughly, given $P(\alpha, k) = \langle E_1 \dots E_k \rangle$ a k-segmentation of α, we have $Rec(P(\alpha, k)) = \bigcap_{E_i \in P(\alpha,k)} (\bigcup_{S_j \in E_i} S_j)$. In other words, $Rec(P(\alpha, k))$ contains the set of all recurrent topics that are present in each episode of $P(\alpha, k)$. $Rec(\alpha, k)$ is *maximal* means that it is obtained from a k-segmentation of α that maximizes the size of the recurrent topics set: $Rec(\alpha, k) = Rec(P_{max}(\alpha, k))$ with $P_{max}(\alpha, k) = \text{argmax}_{\{P(\alpha,k)\}} |Rec(P(\alpha, k))|$. k gives the number of recurrences of the topic signature in sequence α. Thus, given a number of recurrences k, finding the *topic k-signature* relies on finding the k-segmentation that maximizes the size of the topic set that appears in each episode of that segmentation. For example, in Figure 1, $\{a, b\}$ is a topic 4-signature, indeed $Rec(\alpha, 4) = E1 \cap E2 \cap E3 \cap E4$ $= (S_1 \cup S_2 \cup S_3) \cap (S_4 \cup S_5) \cap (S_6) \cap (S_7) = \{a, b, c, d\} \cap \{a, b\} \cap \{a, b\} \cap \{a, b, c, e\} = \{a, b\}$. There is no largest set of topics that is repeated in each episode of a 4-segmentation of α. As one can see in this example, episodes can be of different sizes, and speeches are grouped into episodes such that the topic signature is the largest.

The signature model contains two types of information. First, the intersection of all E_j contains the topics that are recurrent. In our case, this reveals the topics that one candidate has been speaking about, throughout the campaign speeches. The second information is temporal, through the episode timestamps. These timestamps reveal the rhythm of the topics usage. The signature actually links both information, to give the recurrent topics and their dynamic.

By varying the value of k, one can explore the main topics (if k is large) or the secondary topics, that are still recurrent (when k is low). Therefore, recurrent topics and their dynamics can be analyzed on different time scales. The difference with some previous approaches (Savoy, 2010) is that the size of each episode E_j is not defined in advance. Instead, the signature adapts the segmentation and episode size to reveal the rhythm of the topics usage.

3 Case Study: 2016 US Presidential Campaign

In this section, the topic signatures of Clinton and Trump during the 2016 US presidential campaign are analyzed.

3.1 Dataset

The dataset contains the transcripts of campaign speeches of both candidates Clinton and Trump, from April, 2015 to November, 2016. The speeches have been extracted from the American Presidency Project (APP)[1]. This yielded a total of 164 speeches: 93 for Clinton and 71 for Trump[2].

3.2 Preprocessing

The dataset was preprocessed as follows. First, the sentences that did not correspond to a candi-

[1] http://www.presidency.ucsb.edu/2016_election.php

[2] Including the 3 presidential debates. Speeches of Clinton prior to April 2015 were discarded

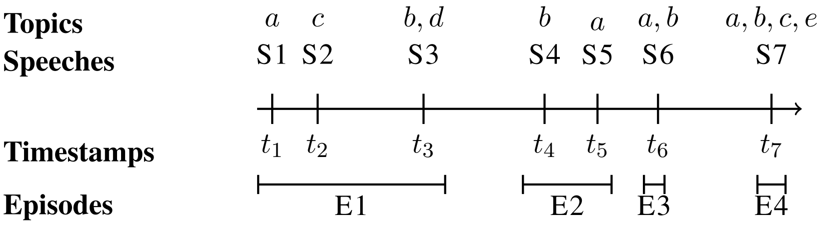

Topics
Speeches
Timestamps
Episodes
a c b,d b a a,b a,b,c,e
S1 S2 S3 S4 S5 S6 S7
t_1 t_2 t_3 t_4 t_5 t_6 t_7
E1 E2 E3 E4

Figure 1: A speech sequence and a 4-segmentation. The recurrent topics are $\{a, b\}$.

date utterance (journalists questions, introduction by another speaker . . .) were removed. Next, the sentences were tokenized and the tokens associated with some Part-of-Speech (POS) tags were kept. Precisely, nouns, adjectives and foreign words were kept while verbs and personal nouns were removed. While removing verbs can lead to a loss of semantic information, we found that it resulted in more interpretable topics. This choice of removing verbs has previously been made for topic modeling in political texts (Zirn and Stuckenschmidt, 2014). Personal nouns were discarded to remove all references to interviewers or other politicians. We considered keeping some proper nouns (the ones of both campaigners and of some other politicians) but it added noise in the topic modeling step, without providing additional relevant information. Finally, remaining tokens were lemmatized and stop words were removed. We used the WordNet lemmatizer (Miller and Fellbaum, 1998) and the list of stop words from the nltk library[3] (Bird et al., 2009). The final dataset contained 6240 different lemma.

3.3 Topic Modeling

Even though words could be analyzed directly (Savoy, 2010), we decided to analyze topics. This choice is mainly guided by the fact that we are looking for recurrent topics, and working directly on words gave uninteresting results, as recurrent words are not directly representative of each candidate ideas. Different topic modeling techniques were tested (Stevens et al., 2012) (LDA (Blei et al., 2003) and NMF (Lee and Seung, 1999)) with different parameters, number of topics and settings (with or without verbs for example). As a result, we concluded that using NMF on count vectors with 15 topics produced the most meaningful, diverse, yet non redundant topics. Some of these topics and their top lemma are shown in Table 1.

Table 1: Some topics found by NMF, and their main lemma.

Topic name	**Main topic lemma**
Economic policy	economy, growth, new, business, income, wage
Woman president and voters	woman, election, president, future, young
Illegal immigration	immigration, illegal, law, border, criminal, visa
Climate change	energy, climate, change, clean, future, important

However, it should be noted that other topic modeling techniques ((Greene and Cross, 2017) for example) could be used, and lead to meaningful results. Indeed, as our method is built on top of topics, any technique that provides good enough topics can be used. Any improvements in the topic model can help to draw more precise conclusions (if cleaner topics are available). This remark is also true regarding our choice of removing verbs and personal nouns.

Within NMF, a speech is represented as a numeric weight vector across all topics. However, the signature model works on symbolic data, which means that a set of representative topics for each speech has to be selected. As we want to discriminate the main topics of a speech from the remaining ones, we applied a clustering on the weight vectors of each speech. Two clusters were looked for, the first containing the highest weights i.e. the cluster of the main topics, and the second containing the secondary or absent topics, with lowest weights. We used the spectral clustering technique (Shi and Malik, 2000) from the scikit-learn library[4] (Pedregosa et al., 2011). We did not used techniques based on the Euclidean distance (such as k-means (MacQueen et al., 1967)) as it is not suited to separate main topics from minor ones. Three main topics emerged per speech on

[3] http://www.nltk.org/

[4] http://scikit-learn.org

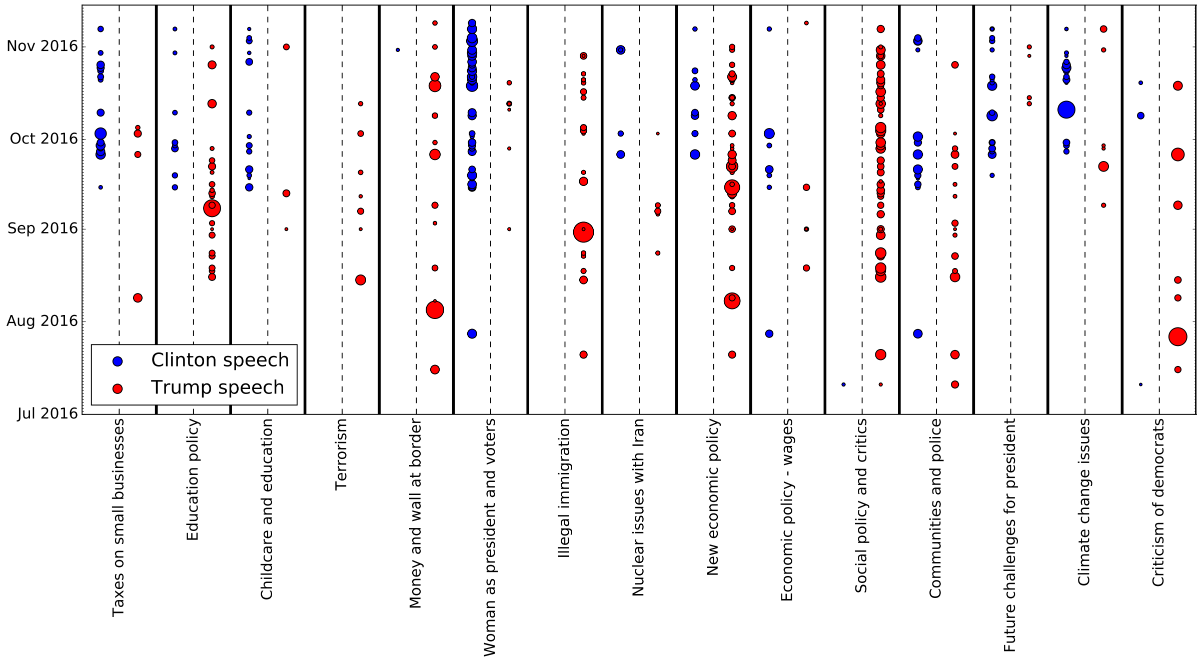


Figure 2: Campaign topics through time for each candidate. Each circle represents the presence of a topic in a speech. The larger the topic, the more present in a speech. Trump speeches are depicted in red, Clinton speeches in blue.

average.

3.4 Topic Signature Extraction

To study the main topics on different time scales, we computed signatures with different values of k. Table 2 displays the results.

3.5 Discussion

About Extracted Topics Figure 2 displays a visualization of all main topics. Only the last months of the campaign are plotted, since both candidates were particularly active during that period and speeches were sparse earlier. The visualization is especially suited to analyze single topics. First, we can see that most topics are discriminative, they appear often in one candidate's speech while being almost absent in the other's. Some topics, like *Communities and police*, are shared but not used on the same time line. Another example is the use of the *Climate change issues* topic. We can see that it is mainly used at the end of the presidential campaign by Clinton.[5]

About Topic Signatures While the previous section shows how individual topics can be analyzed, the signature allows for analyzing the main topics as a whole. Let us look at each candidate's recurring topics in Table 2. The main topics of each candidate are well separated, showing that each candidate has its own targeted voters. Clinton focused on topics related to communities, youth, issues for the next generations, and woman as president. Trump focused on topics such as new economical policies, illegal immigration, new social policies and criticism of the former government.

Table 2: Signature topics in speeches of Clinton (top) and Trump (bottom), for some values of k.

Clinton

No	Recurrences (k)	Signature topics
C1	57	Woman as President
C2	30	C1 + Future challenges for President
C3	16	C2 + Communities and police
C4	12	C3 + Childcare and education

Trump

No	Recurrences (k)	Signature topics
T1	48	Social policy and critics
T2	28	T1 + New economic policy
T3.1	15	T2 + Illegal immigration
T3.2	15	T2 + Education policy
T4.1	9	T3.2 + Illegal immigration (T3.1 + T3.2)
T4.2	9	T3.2 + Money and wall at border

The signature of Clinton is quite simple, as low-

[5] *Climate change issues* became a topic of interest when Clinton attacked Trump on him saying that climate change is a hoax in the first presidential debate (September 26, 2016).

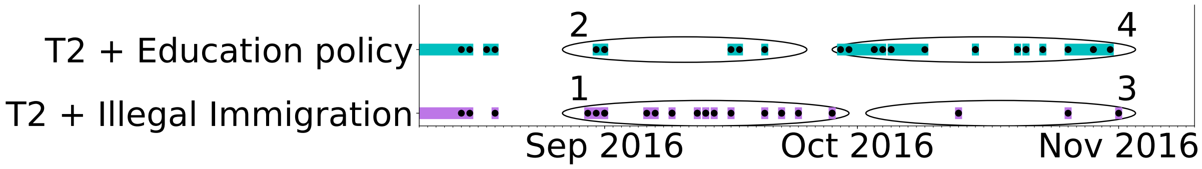


Figure 3: Episodes of two Trump signatures. T3.1: Social policy and critics + New economic policy + *Illegal immigration* ; T3.2: Social policy and critics + New economic policy + *Education policy*. Every rectangle in pink or blue represents an episode, and each black dot represents a speech belonging to an episode. Each numbered ellipse represents a group (annotated by hand) of episodes.

ering the minimal number of occurrences only adds new topics to the signature. This means that Clinton is very stable in her main topics. This observation is also partially true for Trump. Indeed, Trump has sometimes different signatures for a given number of occurrences. For example, with $k = 15$, Trump speeches main topics can include *Illegal immigration* or *Education policy*, but not both together. This is interesting because it shows that Trump is more diverse in his recurrent topics and that some of them rarely occur together.

To further deepen the analysis of the fact that Trump speeches include either *Education policy* or *Illegal Immigration* but rarely both, let us look at the episodes of the related signatures, represented in Figure 3. First, we note that the difference between both signatures episodes began to be apparent by September 2016. Indeed, the signature containing *Illegal immigration* only has three episodes (Group 2), whereas the one with *Education policy* has 11 episodes (Group 1). This large difference shows that, in September, Trump discussed his main topics a lot (*Criticism of former government*, *New social policies* and *New economic policy*) in association with *Education policies*. In October 2016, he switched to *Illegal immigration* while keeping his main topics, as there are 3 episodes for *Education policy* (Group 3) whereas there are 7 episodes for *Illegal immigration* (Group 4). While the fact that Trump stopped talking about *Education policy* at the end of September 2016 is visible in Figure 2, the segmentation performed by the signature brings additional information. Indeed, the signature is changing only one of its topics, so we know that Trump kept talking about his other main topics (*Social policy and critics* and *New economic policy*) while switching from *Education policy* to *Illegal immigration*.

Another important point is that by the beginning of October, when Trump switched from *Education policy* to *Illegal immigration*, the episodes are longer than the remaining ones (Group 4). This means that Trump's main topics are distributed among more speeches than before, which can reflect a change in his strategy. This information is not easily visible in Figure 2, but it is available from a simple analysis of Trump signature.

This case study, based on topic signatures, shows that our method is able to derive each candidate's recurrent topics. Analyzing episodes and related signature topics enables to spot changes in Trump speeches and to explain how some of his recurrent topics are related to each other. This kind of precise analysis is beyond the capabilities of naive regular segmentation techniques.

4 Conclusion

We have presented a new method for analyzing political discourse. It associates standard topic modeling with signature mining and enables the identification of the main topics of politicians during a campaign as well as their dynamics. The 2016 US presidential campaign analysis provides interesting results: though the discourse of H. Clinton was relatively stable, important changes could be identified in the discourse and communication strategy of D. Trump. These specific results on the campaign dynamics were obtained thanks to the temporal flexibility of the model.

In the future, we would like to apply the method to more challenging data, such as political tweets. Preliminary results on the 2016 US campaign tweets show that the topics used by both candidates were different from their speech: the tweets, being shorter than speeches, emphasize more oversimplified criticism of the opponent rather than justified political ideas.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3(Jan):993–1022.

Louis-Jean Calvet and Jean Véronis. 2008. *Les mots de Nicolas Sarkozy*. Seuil París.

Clement Gautrais, René Quiniou, Peggy Cellier, Thomas Guyet, and Alexandre Termier. 2017. Purchase signatures of retail customers. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pages 110–121.

Derek Greene and James P. Cross. 2017. Exploring the political agenda of the european parliament using a dynamic topic modeling approach. *Political Analysis* 25(1):7794. https://doi.org/10.1017/pan.2016.7.

Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes* 25(2-3):259–284.

Daniel D Lee and H Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–791.

James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Oakland, CA, USA., volume 1, pages 281–297.

George Miller and Christiane Fellbaum. 1998. Wordnet: An electronic lexical database.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Vinodkumar Prabhakaran, Ashima Arora, and Owen Rambow. 2014. Staying on topic: An indicator of power in political debates. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, pages 1481–1486.

Kevin M Quinn, Burt L Monroe, Michael Colaresi, Michael H Crespin, and Dragomir R Radev. 2010. How to analyze political attention with minimal assumptions and costs. *American Journal of Political Science* 54(1):209–228.

Jacques Savoy. 2010. Lexical analysis of us political speeches. *Journal of Quantitative Linguistics* 17(2):123–141.

Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence* 22(8):888–905.

Keith Stevens, Philip Kegelmeyer, David Andrzejewski, and David Buttler. 2012. Exploring topic coherence over many models and many topics. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 952–961.

Teun Van Dijk. 1998. What is Political Discourse Analysis? *Belgian Journal of Linguistics* 11:11–52. https://doi.org/10.1075/bjl.11.03dij.

Cäcilia Zirn and Heiner Stuckenschmidt. 2014. Multidimensional topic analysis in political texts. *Data & Knowledge Engineering* 90:38–53.

Assessing Objective Recommendation Quality through Political Forecasting

H. Andrew Schwartz[†] Masoud Rouhizadeh[†,‡] Michael Bishop[‡]
Philip Tetlock[‡] Barbara Mellers[‡] Lyle H. Ungar[⊙]
[†]Computer Science, Stony Brook University
[‡]Psychology, University of Pennsylvania
[⊙]Computer & Information Science, University of Pennsylvania
has@cs.stonybrook.edu, ungar@cis.upenn.edu

Abstract

Recommendations are often rated for their subjective quality, but few researchers have studied quality in terms of objective utility. We explore quality assessment with respect to both subjective (i.e. users' ratings) and objective (i.e., did it influence? did it improve decisions?) metrics in a *massive online geopolitical forecasting system*, ultimately comparing linguistic characteristics of each quality metric. Using a variety of features, we predict all types of quality with better accuracy than the simple yet strong baseline of recommendation length. For example, more complex sentence constructions, as evidenced by subordinate conjunctions, are characteristic of recommendations leading to objective improvements in forecasting. Our analyses also reveal rater biases; for example, forecasters are subjectively biased in favor of recommendations mentioning business deals and material things, even though such recommendations do not indeed prove any more useful objectively.

1 Introduction

Finding good recommendations is an integral part of a modern information-seeking life – from purchasing products based on reviews to finding answers to baffling questions. Following the tradition of sentiment analysis, many have proposed methods to automatically assess the quality of recommendations or comments based on subjective ratings of their usefulness (Liu et al., 2007; Siersdorfer et al., 2010; Becker et al., 2012; Momeni et al., 2013) or of persuasiveness (Wei et al., 2016). However, information thought to be useful does not always prove so, and subjective ratings may be driven by biases. Reviews which convince you to watch a movie or buy a product do not guarantee that you will enjoy the product, and the most convincing or highest rated answers to questions on sites like Stack Overflow or Yahoo Answers are not always the most accurate.

We explore recommendations in a unique dataset, an online forecasting competition, which offers a rare glimpse into both subjective and objective quality. In this competition, the users (forecasters) had a measurable goal — to forecast the outcomes of geopolitical events — and a need to effectively gather information in order to reach this goal. They viewed recommendations (or "tips") from other forecasters, rated them, and potentially updated their own forecast. This data not only allows us to access what information the forecasters *thought* useful based on their ratings, but also what was objectively useful based on (a) the rate at which forecasters change their prediction after viewing tips and, (b) the average improvement (or decrease) in their prediction accuracy after this change.

We seek to tease out subjective biases by distinguishing the linguistic characteristics of recommendations with high subjective ratings from those of objective utility. Since objectively good recommendations tend to get higher subjective assessments, detecting such differences is nontrivial. Past literature has suggested that subjective quality is well predicted by comment length (Agichtein et al., 2008; Beygelzimer et al., 2015); We seek differences beyond this. We build quality predictive models from linguistic features of recommendations — 1 to 3 word sequences, parts-of-speech, and mentions of concepts from a taxonomy — comparing to surface-level features (length and readability). We then explore the language which distinguishes high quality comments from low quality, controlling for comment length,

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2348–2357
Copenhagen, Denmark, September 7–11, 2017.

and ultimately what distinguishes high subjective quality from objective quality in order to reveal subjective biases.

Contributions. We see the key contribution of this paper, perhaps non-conventional for NLP, as presenting an evidence-based suggestion for the field to consider metrics of objective quality beyond that of subjective ratings. To the best of our knowledge this represents the first study of objective comment quality using randomized experimental data. Specific novel contributions include (a) the development of automated assessments fit to objective outcomes, (b) the identification of linguistic features distinguishing high- from low-quality comments, (c) the use of a new, important, real-world domain for NLP – geopolitical information, and (d), most consequentially, the identification of subjective biases manifested in the comments' text itself.

2 Data Set

Data were collected from the **m**assive **o**nline ge**o**political **f**orecasting system (MOOF) described by Mellers et al. (Mellers et al., 2015; Atanasov et al., 2016). Forecasters completed *tasks* where they indicated the probability of discrete outcomes for specific future geopolitical events around the world. For example, forecasters might be asked to forecast the likelihood of a coup in Venezuela within the next 12 months. Respondents indicated a probability for the event and, after the event resolved, they received a score (Brier, 1950) based on how close their probability reflected the outcome of the event.[1]

Recommendations generated from a another *parallel* forecasting system were presented as "tips". In the parallel system, the recommendation writers were asked "Why did you answer the way you did?" when making forecasts, and were given the option to mark their response as potentially being useful to others. Comments marked useful from the *parallel* system were then presented as tips within the main *MOOF* system, where they were evaluated for their subjective and objective usefulness. Below is an example of one such comment:

> *Predicting the foreign ministers will meet and state something along the lines of "hoping for a summit at the appropriate time." ... Predict this because of the cautious language here and elsewhere: http://www.china.org.cn/wap/2015-03/13/... If wrong, will have time to redress the damage done before any possible summit.*

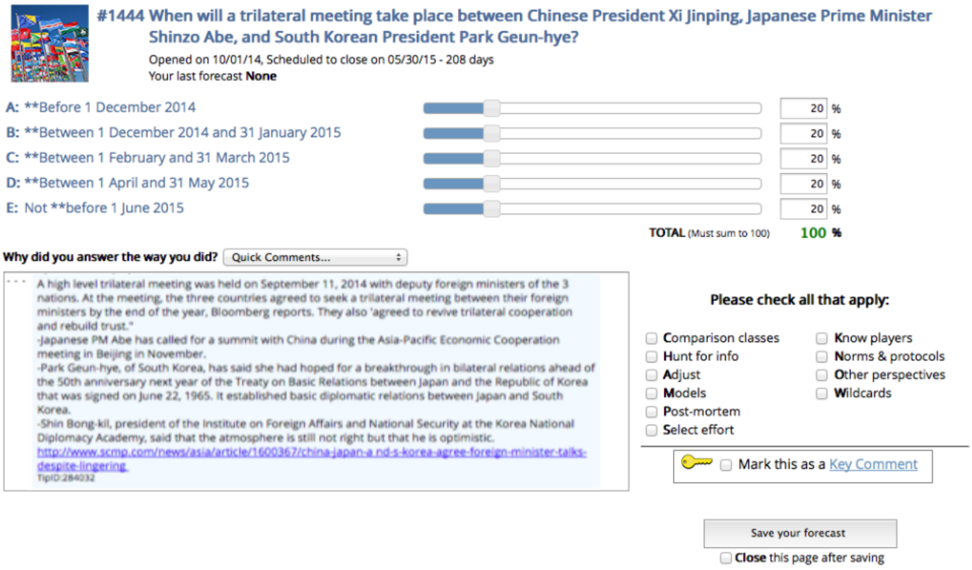
The above forecaster identifies a reason for their decision, a source of information (a Chinese news website), and even a contingency plan if their reasoning doesn't seem to be planning out. Figure 1 shows a representative screen shot from the *MOOF* system.

We focus on one subjective and two objective quality metrics for these recommendations:
rating: subjective ratings on a 5-point scale by the forecasters in the parallel system.
influence: the rate at which MOOF forecasters update their predictions after reading the comment.
benefit: the mean *change* in MOOF forecaster accuracy resulting from updating their predictions after rating the comment. Because there were numerous confounding factors regarding the magnitude of change (i.e. forecaster quality, time until task resolution), we simply encoded the *change* as a binary indicator of whether it was positive or negative.

For the purposes of this study, we consider both *influence* and *benefit* as assessments of objective utility, though they each capture different aspects of utility (behavioral *influence* of others in the case of influence; and specifically positive influence in the case of *benefit*). The *MOOF* forecasters did not know who wrote the tip or any other information about it; their actions were based on the comment's content and not reputation of the comment author. *MOOF forecasters* were also linked directly to the form to update their forecast from the comment to minimize outside influences between the reading of the comment and prediction update. However, as the forecasters were not in a laboratory, other Web browsing behavior in other tabs could not be controlled.

In order to balance quality score reliability with quantity of recommendation data, we restricted the dataset to recommendations with at least 10 words that received at least 3 ratings. This resulted in $8,498$ comments with *ratings* and *influence* scores. Out of those, $4,317$ comments had at

[1] Data, in the form of quality ratings, POS features, and aggregate concept features is released at `http://http://www3.cs.stonybrook.edu/~has/objective_quality/`. Due to IRB privacy considerations we are unable to release the full data set.



#1444 When will a trilateral meeting take place between Chinese President Xi Jinping, Japanese Prime Minister Shinzo Abe, and South Korean President Park Geun-hye?

Opened on 10/01/14, Scheduled to close on 05/30/15 - 208 days
Your last forecast **None**

A: **Before 1 December 2014	20 %
B: **Between 1 December 2014 and 31 January 2015	20 %
C: **Between 1 February and 31 March 2015	20 %
D: **Between 1 April and 31 May 2015	20 %
E: Not **before 1 June 2015	20 %
TOTAL (Must sum to 100)	100 %

Why did you answer the way you did? Quick Comments...

A high level trilateral meeting was held on September 11, 2014 with deputy foreign ministers of the 3 nations. At the meeting, the three countries agreed to seek a trilateral meeting between their foreign ministers by the end of the year, Bloomberg reports. They also 'agreed to revive trilateral cooperation and rebuild trust."
-Japanese PM Abe has called for a summit with China during the Asia-Pacific Economic Cooperation meeting in Beijing in November.
-Park Geun-hye, of South Korea, has said she had hoped for a breakthrough in bilateral relations ahead of the 50th anniversary next year of the Treaty on Basic Relations between Japan and the Republic of Korea that was signed on June 22, 1965. It established basic diplomatic relations between Japan and South Korea.
-Shin Bong-kil, president of the Institute on Foreign Affairs and National Security at the Korea National Diplomacy Academy, said that the atmosphere is still not right but that he is optimistic.
http://www.scmp.com/news/asia/article/1600367/china-japan-a nd-s-korea-agree-foreign-minister-talks-despite-lingering
TipID:284032

Please check all that apply:

☐ Comparison classes
☐ Hunt for info
☐ Adjust
☐ Models
☐ Post-mortem
☐ Select effort
☐ Know players
☐ Norms & protocols
☐ Other perspectives
☐ Wildcards

☐ Mark this as a Key Comment

Figure 1: Screenshot of the forecasting system prompting users to justify their forecasts with recommendations. These recommendations are then passed on to the parallel *MOOF* system as "tips" which are rated and sometimes lead forecasters to change their predictions.

rating v influence	*rating v benefit*	*influence v benefit*
.45*	.01	.06*

Table 1: Correlation (Pearson product-moment coefficient) between all subjective and objective quality metrics: rating: forecaster subjective rating, influence: forecaster update rate, benefit: change in forecast accuracy due to update. *significant at $p < .01$.

least one associated change in forecast, and thus had a score for the final metric, *benefit*.

Table 1 shows correlations (Pearson product-moment coefficients) between all subjective and objective quality metrics. *Rating* and *influence* are fairly predictive of each other ($r = 0.45$, $p < 0.01$), and neither correlates particularly strongly with *benefit*. ($r = 0.01$, $p > .01$ for *rating* v *benefit*, and $r = 0.06$, $p < .01$ for *influence* v *benefit*). This implies that while comments rated for subjective utility are also more likely to influence users, they are not necessary influencing them in a positive way. Further, it is possible (and indeed the case) that the characteristics of comments deemed subjectively useful may differ from those that objectively provide benefit. In Section 4 we consider the content that leads to these differences in the metrics.

3 Regression

Our goal is to predict the quality score of a comment from its content over each of the three quality metrics: *rating*, *influence*, and *benefit*. These 3 types of features were chosen in order to account for qualitatively different types of linguistic attributes. To capture linguistic variance at varying resolutions, we use a combination of open-vocabulary and taxonomic linguistic features:

ngrams: *1 to 3 word sequences*. These ngrams were recorded as binary variables indicating whether each ngram appeared in each comment. We limited ngrams to those mentioned in at least 0.1% of all comments.

parts-of-speech: *POS frequences* The Stanford Part-of-Speech tagger was used to identify parts of speech in each comment. The relative frequency of each tag (i.e., the probability of the tag, given the comment) was recorded.

concepts: *Nominal concepts within a hierarchy.* The WordNet noun ontology (Fellbaum, 1998) was used. For each comment, we tracked all of the hypernyms of each noun within the comment. As features per comment, we use the presence of each hypernym concept, limited to those concepts that appeared in at least 0.1% of all comments.

To control for *task*-specific language, features (n-grams) were restricted to those mentioned in at least 50% of forecasting *tasks*. The *n-grams*, intended to capture fine-grained lexical information, were also restricted to those mentioned in at least 1% of comments, while the *parts-of-speech* and *concepts*, intended to capture more coarse-grained linguistic characteristics, were restricted to those appearing in 5% of comments. These thresholds were chosen such that total number of features was within the same order of magnitude as the number of recommendations with objective scores In the end, there were $1,202$ unique *n-grams*, 32 unique *parts-of-speech*, and 155 unique *concepts*.

3.1 Predictive Modeling

We built predictive models of all three quality metrics based on the linguistic features of the messages. To handle covariance and avoid over-fitting with so many features, we used a series of feature selection and dimensionality reduction techniques fed into a ridge regression to create the models. Specifically, we filtered the features (which were already restricted to those present in at least 50% of the forecasting tasks) to those with a small linear correlation with the outcome, defined as having a family-wise error rate < 60 (Toothaker, 1993). This feature selection was run independently on each type of feature (n-grams, parts-of-speech, and concepts). Similar to correcting for multiple hypothesis tests, family-wise error penalizes groups containing more features (i.e. n-grams) more stringently.

We then used randomized *singular value decomposition* (SVD) (Halko et al., 2011) to reduce the feature set to $^1/_5$ the number of original features. Randomized SVD uses stochastic sampling to more efficiently calculate the principal components (Martinsson et al., 2011).In this context, SVD functions as a type of regularization to reduce variance by removing lesser principal components (and thus helping to avoid overfit). Finally, the resulting dimensions were fit to the given quality metric using L2 penalized linear regression. All feature selection, dimensionality reduction, and L2 parameter tuning were done over a held-out portion of the training set.

3.2 Evaluation

To evaluate our models out-of-sample, we use 10-fold cross-validation over the subjective ratings (*rating*) and update rates (*influence*). In this process, a random selection of 1/10th of the comments are held-out as a test set, while the other 9/10ths are used to train (estimate) the model. This model is then used to predict the quality of the comments in the 1/10th sample and compared to the true quality for those comments (using Pearson correlation in this case). However, many of our scores for change in forecaster accuracy (*benefit*) are based simply on one change and thus quite unreliable. While it is best to include such noisy data when training, it does not provide a very accurate assessment. Therefore, we use dedicated training and test sets, where the test set is a random sample of 500 comments with more than 3 updates and thus a more reliable mean change in forecaster accuracy.

	rating	***influence***	***benefit***
baseline	.59	.24	.02
our model	.76*	.37	.21*

Table 2: Predictive accuracy (out-of-sample Pearson correlation coefficient) of our content-based models across the subjective and objective measures. *baseline*: square-root number of words; *our model*: based on *ngrams*, *parts-of-speech*, and *concepts*. *significant reduction in error over the baseline at $p < .001$.

As a baseline, we use the square-root of the number of words in the comment. This may seem like weak measure of quality, but the history of automatic quality assessment is saturated with findings that length is the best predictor of quality. This holds true for both answers to questions (Agichtein et al., 2008; Surdeanu et al., 2011; Beygelzimer et al., 2015); as well as e-commerce reviews (Cao et al., 2011; Racherla and Friske, 2012). Of course, length is not as shallow as it may seem at first; given no strong incentive for authors to leave long comments, length is likely a proxy for thoroughness of the comment. Still, because we would like to understand the content distinguishing various metrics of quality, we view length as a baseline to move beyond.

Table 2 compares the accuracy of models built on content (*ngrams*, *parts-of-speech*, and *concepts*) to the baseline of length. In all cases, *our models*, based on content, perform significantly better than those based only on length. Further in the case of *benefit* (change in forecaster accuracy), length has virtually no predictive power.

measure	*readability*	*length & readability*
rating	.23	.59
influence	.08	.24

Table 3: Predictive accuracy (out-of-sample Pearson correlation coefficient) of the baseline of length and the baseline of readability and their combination across the subjective and objective measures. *length*: square-root number of words; *readability*: Flesch-Kincaid Scale.

3.3 Relation to Readability

Some previous work used on *readability* measures to evaluate comment quality (e.g. (Agichtein et al., 2008; Hsu et al., 2009)). Readability often refers to the difficulty or complexity scale of a comment, determining the minimum age group able to perceive it. Various readability measures have been suggested in the past such as Flesch-Kincaid Formula (Kincaid et al., 1975), Gunning-Fog Index (Gunning, 1952), and SMOG Grading (Mc Laughlin, 1969). All methods are based on the combination of the count of syllables or words in the comment (as a representation of syntactic complexity), and the number of sentences in the comments (representing the semantic complexity). Such measures of readability are often considered naive and questionable, however, they are commonly used and present a coarse evaluation of the comment's complexity.

We used Flesch-Kincaid scale to measure readability. This scale measures readability by the average number of syllables per word as well as the average number of words per sentence (Doak et al., 1996). We combined the baseline of length and the baseline of readability in order to measure the quality of comments. Table 3 shows results for readability and length plus readability. We find no significant improvement in the baseline and that our model based on content still adds significantly more predictive accuracy.[2]

4 Differential Analysis

The prediction results show promise for automated quality assessment, and that linguistic content can predict quality above-and-beyond length (a proxy for comprehensiveness). Next, we explore what content exactly it is that is predictive of each quality metric, and what content suggests biases distinguishing subjective versus objective quality.

4.1 Method

To identify distinguishing features, we use a series of multivariate linear regressions to find the relationship between each individual linguistic feature and the given quality metric, controlling for comment length – a process known as differential language analysis (Schwartz et al., 2013). Specifically, the individual linguistic feature along with the comment length (logged) are standardized and used as independent variables, and then fit to the standardized form of the given quality metric. The coefficient given is then taken as the standardized effect size of the relationship between that feature and the quality metric, holding length constant (Fox, 1997). In other words, it tells us how much the feature can explain the quality score, beyond what is explainable simply from length.

Using regression to relate variables, although rarely done in Computer Science domains, is standard practice in social and political science (Fox, 1997), though typically not over thousands of potential independent variables as we do here. Therefore, we also correct for multiple hypotheses by using the Benjamini-Hochberg procedure (Benjamini and Hochberg, 1995) over our significance tests.

Differential language analysis allows us to observe and test the unique relationship between each feature and each metric, holding length constant. In addition, we use the difference between standardized metric scores to find the features that distinguish high quality comments in one metric versus another. All methods were implemented within the package, dlatk (Schwartz et al., 2017).

4.2 Quality Comment Features

Figure 2 shows the n-grams most highly correlated with each of our quality metrics. Size indicates correlation strength while color represents overall frequency. Across both subjective ratings and objective update rates, we see discussion of news plays an important role (e.g. "news", "article", and "www.reuters.com"). We do not see the same from comments resulting in positive changes of forecaster accuracy (benefit), which seemed to be distinguished by negation (e.g. "no", "unlikely"). For influence, we see other features indicating probabilistic reasoning (e.g. "%"); the individual

[2] Adding length and readability together to the full model had no benefit.

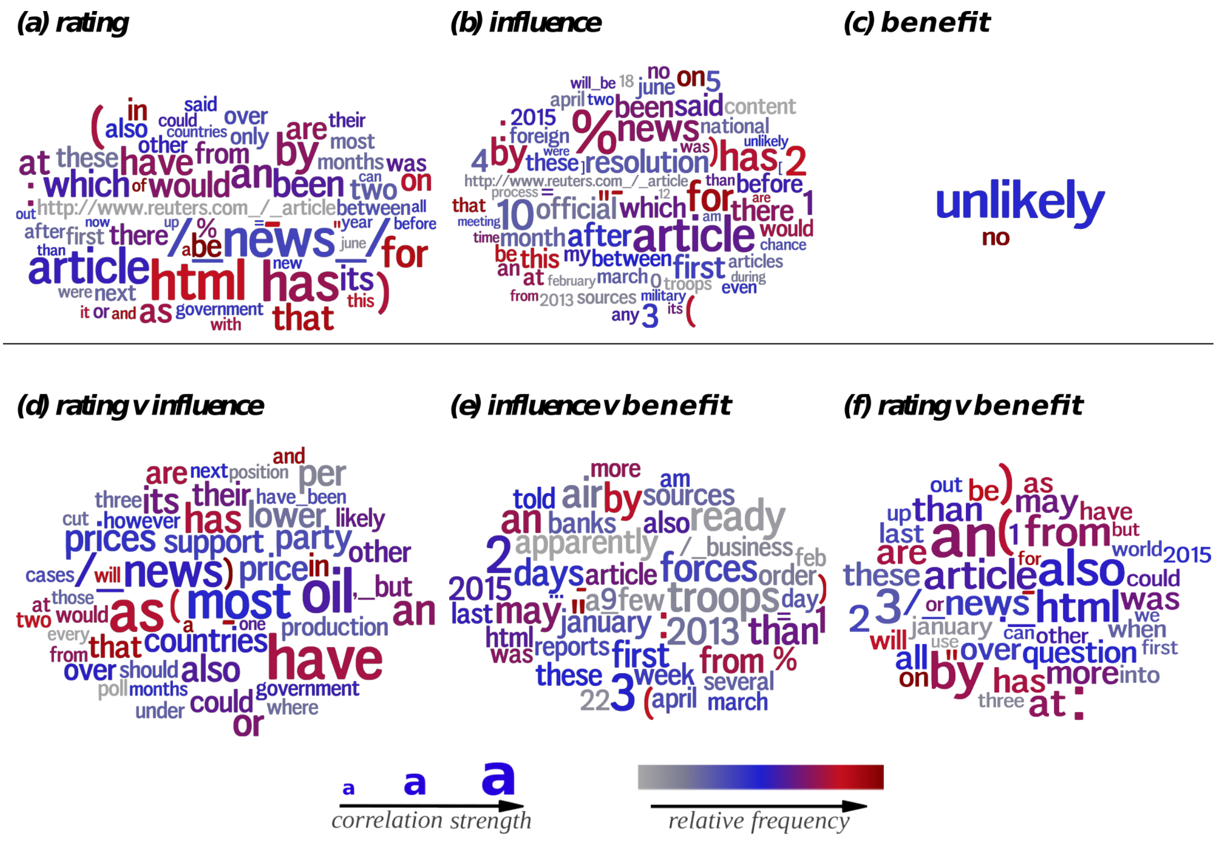


Figure 2: Top: *ngrams* (words and phrases) most distinguishing high quality comments based on (a) subjective ratings, (b) objective forecaster update rates, and (c) objective changes in forecaster accuracy. Bottom: *ngrams* (words and phrases) most distinguishing (d) subjective ratings from objective update rates and (e) objective update rates from objective changes in forecaster accuracy. All correlates are significant at $p < .05$ after a Benjamini-Hochberg false-discovery rate correction.

numbers in influence actually represent numbered lists of signal. These features are more indicative of a comment that convinces one to update their prediction rather than one that highly rated.

We can directly observe the differences in what the metrics capture by looking at the final two visualizations in Figure 2, *rating v influence* and *rating v benefit*. *Rating v influence* tells us which ngrams were predictive of high quality comments that were less likely to result in a forecast update. Discussion of energy topics (e.g. “oil”, “prices”, “cut”, “production”) are predictive of comments subjectively rated higher than their update rates would suggest. Further, discussion of dates (e.g. “january”, “may”, “days”, “2015”, “2013”) seems to predict comments that lead forecasters to update but which do not actually result in better predictions (*influence v benefit*). Discussion of news and articles (“article”, “news”, “html”, “question” appears to predict subjectively top-rated comments from those comments that actually result in better predictions (*influence v benefit*).

Table 4 shows differential language of quality based on part-of-speech. We observe that some patterns from the *ngram* results are generalized. Examples of these patterns include more numbers, parentheses, and quotes in highly rated and influential comments. Other results are somewhat novel, such as the use of more adverbs and subordinate conjunctions (e.g. “though”, “since”, “whereas”) in comments leading to better forecast accuracy, and that both ratings and influence favored quotes.

We notice that including explanation or afterthought (using more parentheses) can predict subjectively high rated comments from the comments leading to updated ratings. The use of quotes and numbers along with mentioning proper nouns can predict influential comments that do not help in better predictions. Including explanation, quotes, and numbers as well as reporting past events seems to predict comment that convinces

	Parts-of-speech				
rating	parentheses	number	line-break	quote	verb, past-tense
influence	line-break	number	parentheses	verb, past-tense	quote
benefit	adverb	sub-conjunction	-	-	-
rating v influence	parentheses	-	-	-	-
influence v benefit	quotes	proper noun	number	-	-
rating v benefit	parentheses	quotes	number	verb, past-tense	-

Table 4: Top: Most distinguishing parts-of-speech correlating with the three metrics of quality: subjective rating, forecaster updates (*influence*), and forecaster accuracy (*benefit*). Bottom: *parts-of-speech* most predictive of differences in quality metric scores (*rating v influence* and *rating v benefit*). All correlates are significant at $p < .05$ after a Benjamini-Hochberg false-discovery rate correction.

one to update their prediction as opposed to helping better forecasting accuracy.

Distinguishing *concepts*, in Table 5 offer a different perspective. Discussions of documents and written material characterize highly rated and influential comments, while changes in accuracy (*benefit*) were characterized by more discussion of abstract attributes or states of being. Highly rated comments were more likely to discuss concepts related to transactions and materials than influential comments, but they are more likely to discuss concepts about creation compared to comments resulting in better predictions.

5 Related Work

While no prior work has focused on recommendation quality in terms of how readers change or improve decisions (i.e. objective metrics), there is an extensive body of literature on the automated analysis of subjective comment quality from which we build. Such work typically uses subjective assessments specific to their application domain (e.g. thumbs up/ thumbs down over YouTube comments, answer ratings in Yahoo Answers, or helpfulness ratings of Amazon product reviews). Below we discuss such work organized into three main categories of subjective comment quality: *comment usefulness*, *the quality of answers in QA platform*, and *comment helpfulness*.

Comment usefulness concerns the acceptance (vs. non-acceptance) of comments by a community (Siersdorfer et al., 2010). Some previous work has focused on the usefulness of YouTube comments. For example, Siersdorfer et. al. (Siersdorfer et al., 2010) have used support vector machines to identify the acceptance of comments by the community in 6 million comments on 67,000 YouTube videos. They showed that community feedback and term weight features can be good predictors of comment acceptance. Other work such as (Momeni et al., 2013) predicted comment usefulness on YouTube and Flickr, and found that comments rated as useful usually include named entities and "insight"-related terms (think, know, consider, etc.), whereas non-useful comments contain emotional and affective expression, and "certainty"-related (always, never, etc.).

Others working on comment quality assessment focus on user-generated answers in social media and QA platforms. (Bian et al., 2008) present a general ranking approach for finding the answers from 1,250 TREC factoid questions containing at least one similar question from Yahoo! Answers. They found that various features including textual (e.g. word overlap, length ratio), and community (e.g. total points, total answers) are important in retrieving factual answers, whereas statistical features (e.g. length, lifetime, popularity) are not very effective. Exploring if additional features could outperform answer length in predicting the best answer, Beygelzimer et al. (2015) considered a wide variety of features including functional, linguistic, questioner and answerer personalization, and "superlative" features, but were unable to overcome the length baseline.

Helpfulness is mainly defined in the context of online reviews and represents the number of users indicating a particular review was helpful. Using structural features like sentence tokens, length,

	Concepts	
rating	*document, written document, papers* – writing that provides information (especially information of an official nature).	*gathering, assemblage* – a group of persons together in one place.
influence	*writing, written material, piece of writing* – the work of a writer; anything expressed in letters of the alphabet (especially when considered from the point of view of style and effect).	*auditory communication* – communication that relies on hearing.
benefit	*attribute* – an abstraction belonging to or characteristic of an entity.	*state* – the way something is with respect to its main attributes.
rating v influence	*transaction, dealing, dealings* – the act of transacting within or between groups (as carrying on commercial activities).	*material, stuff* – the tangible substance that goes into the makeup of a physical object.
rating v benefit	*creation* – an artifact that has been brought into existence by someone.	

Table 5: Top: Select concepts correlating with the three metrics of quality: subjective rating, forecaster updates (*influence*), and forecaster accuracy (*benefit*). Bottom: concepts most predictive of differences in qualtric metric scores (*rating* v *influence* and *rating* v *benefit*). All correlates are significant at $p < .05$ after a Benjamini-Hochberg false-discovery rate correction.

proportion of question sentences along with lexical and syntactic features, Kim et. al. (2006) could achieve rank correlations of up to 0.66 with helpfulness votes of Amazon reviews. Ghose and Ipeirotis (2011) analyzed length, readability, and subjective and objective information on Amazon.com reviews finding that reviews with objective, and highly subjective sentences are rated more helpful. Similar findings were reported by Mudambi and Schuff (2010), finding that review extremity, review depth, and product type affect the perceived helpfulness of the review.

Most studies listed thus far found length of comment to be the dominant predictor, with other features providing minimal benefit. However, a few studies (including our own) have found this baseline can be overcome. For example, Racherla and Friske (Racherla and Friske, 2012) investigated perceived usefulness of consumer reviews on Yelp and found that reputation and expertise were more important than total number of words on perceived usefulness.

All of these prior works focus on assessing subjective aspects of comments (usefulness, quality, and helpfulness); Perhaps the study coming closest in spirit to our own was Ghose et al. Ghose et al. (2007) who quantified quality of reviews by the economic change they produced. However, they still were not dealing with a randomized experiment and so conclusions were correlational and the objective was better sales of the product rather than benefit to the reader (i.e. leading to a better decision).

6 Conclusion

Our results suggest three key findings. First, what one writes in a comment is more important than simply how much one writes; this is true across both subjective and objective outcomes, though length had virtually no predictive ability for improving forecaster accuracy. Second, we found many linguistic features characteristic of quality, many of which seemingly align with attributes of strong forecasters (Mellers et al., 2015). For example, high quality comments contained signals of probabilistic reasoning (e.g. “%”, “unlikely”, numerical parts-of-speech), inductive reasoning (e.g. justifications with “news” and documents), and cognitive flexibility (e.g. subordinate conjunctions which signal more complex sentence constructions used to relate two independent clauses or ideas).

Most importantly, our results suggest a subjective bias: that what people believe to be useful does not always turn out to be truly useful. Subjective ratings were favorably biased towards comments containing energy-related content (e.g. “oil”, “production”, “prices”), news articles, and nouns of creation and materials (as opposed to abstractions or attributes).

The implications of identifying subjective biases in comment quality extend to many domains involving comment ratings. Consumers of comments, typically, desire information that ultimately leads to real utility benefits, and this domain is not the only one where objective quality can be obtained: For example, one could: (1) ask consumers of restaurant reviews to indicate if one convinces them to go to the restaurant and then follow up on their experience, (2) consider evaluating research paper quality – reviewer ratings versus citation count (influence), (3) determine whether the answer to the question about how to drive to conserve fuel lead to the reader actually using less gas? A review being convincing or being ”liked” may correlate with better outcomes, but it is not equivalent.

7 Acknowledgments

Funding This research was supported by the Intelligence Advanced Research Projects Activity (IARPA) via the Department of Interior National Business Center (DoI/NBC) Contract No. D11PC20061. The U.S. government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright annotation thereon. The views and conclusions expressed herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/ NBC, or the U.S. government. This work also supported, in part, from the Templeton Religion Trust, grant TRT-0048.

References

Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. 2008. Finding high-quality content in social media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, pages 183–194.

P. Atanasov, W. Chang, S. Patil, B. Mellers, and P Tetlock. 2016. Accountability and adaptive performance: The long-term view. *Under Review* .

Hila Becker, Dan Iter, Mor Naaman, and Luis Gravano. 2012. Identifying content for planned events across social media sites. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, pages 533–542.

Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)* pages 289–300.

Alina Beygelzimer, Ruggiero Cavallo, and Joel Tetreault. 2015. On yahoo answers, long answers are best. In *Proceedings of CrowdML: The ICML 15 Workshop on Crowdsourcing and Machine Learning*.

Jiang Bian, Yandong Liu, Eugene Agichtein, and Hongyuan Zha. 2008. Finding the right facts in the crowd: factoid question answering over social media. In *Proceedings of the 17th international conference on World Wide Web*. ACM, pages 467–476.

Glenn W Brier. 1950. Verification of forecasts expressed in terms of probability. *Monthly weather review* 78(1):1–3.

Qing Cao, Wenjing Duan, and Qiwei Gan. 2011. Exploring determinants of voting for the “helpfulness” of online user reviews: A text mining approach. *Decision Support Systems* 50(2):511–521.

Cecilia Conrath Doak, Leonard G Doak, and Jane H Root. 1996. Teaching patients with low literacy skills. *AJN The American Journal of Nursing* 96(12):16M.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

John Fox. 1997. *Applied regression analysis, linear models, and related methods.*. Sage Publications, Inc.

Anindya Ghose and Panagiotis G Ipeirotis. 2011. Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics. *Knowledge and Data Engineering, IEEE Transactions on* 23(10):1498–1512.

Anindya Ghose, Panagiotis G Ipeirotis, and Arun Sundararajan. 2007. Opinion mining using econometrics: A case study on reputation systems. In *annual meeting-association for computational linguistics*. volume 45, page 416.

Robert Gunning. 1952. {The Technique of Clear Writing} .

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review* 53(2):217–288.

Chiao-Fang Hsu, Elham Khabiri, and James Caverlee. 2009. Ranking comments on the social web. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*. IEEE, volume 4, pages 90–97.

Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. 2006. Automatically assessing review helpfulness. In *Proceedings of the 2006*

Conference on empirical methods in natural language processing. Association for Computational Linguistics, pages 423–430.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, DTIC Document.

Jingjing Liu, Yunbo Cao, Chin-Yew Lin, Yalou Huang, and Ming Zhou. 2007. Low-quality product review detection in opinion summarization. In *EMNLP-CoNLL*. pages 334–342.

Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. 2011. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis* 30(1):47–68.

G Harry Mc Laughlin. 1969. Smog grading-a new readability formula. *Journal of reading* 12(8):639–646.

Barbara Mellers, Eric Stone, Pavel Atanasov, Nick Rohrbaugh, S Emlen Metz, Lyle Ungar, Michael M Bishop, Michael Horowitz, Ed Merkle, and Philip Tetlock. 2015. The psychology of intelligence analysis: Drivers of prediction accuracy in world politics. *Journal of experimental psychology: applied* 21(1):1.

Elaheh Momeni, Claire Cardie, and Myle Ott. 2013. Properties, prediction, and prevalence of useful user-generated comments for descriptive annotation of social media objects. In *ICWSM-2013: Proceedings of the International AAAI Conference on Weblogs and Social Media*.

Susan M Mudambi and David Schuff. 2010. What makes a helpful review? a study of customer reviews on amazon. com. *MIS quarterly* 34(1):185–200.

Pradeep Racherla and Wesley Friske. 2012. Perceived 'usefulness' of online consumer reviews: An exploratory investigation across three services categories. *Electronic Commerce Research and Applications* 11(6):548–559.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, and Lyle H. Ungar. 2013. Personality, gender, and age in the language of social media: The open-vocabulary approach. *PloS one* 8(9).

H Andrew Schwartz, Salvatore Giorgi, Maarten Sap, Patrick Crutchley, Johannes C Eichstaedt, and Lyle Ungar. 2017. DLATK: Differential Language Analysis ToolKit. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Stefan Siersdorfer, Sergiu Chelaru, Wolfgang Nejdl, and Jose San Pedro. 2010. How useful are your comments?: analyzing and predicting youtube comments and comment ratings. In *Proceedings of the 19th international conference on World wide web*. ACM, pages 891–900.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2011. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics* 37(2):351–383.

Larry E Toothaker. 1993. *Multiple comparison procedures*. 89. Sage.

Zhongyu Wei, Yang Liu, and Yi Li. 2016. Is this post persuasive? ranking argumentative comments in the online forum. In *The 54th Annual Meeting of the Association for Computational Linguistics*. page 195.

Never Abandon Minorities: Exhaustive Extraction of Bursty Phrases on Microblogs Using Set Cover Problem

Masumi Shirakawa[†‡] and **Takahiro Hara**[‡] and **Takuya Maekawa**[‡]
hapicom Inc., Japan[†]
Graduate School of Information Science and Technology, Osaka University, Japan[‡]
shirakawa@hapicom.jp
{hara, maekawa}@ist.osaka-u.ac.jp

Abstract

We propose a language-independent data-driven method to exhaustively extract bursty phrases of arbitrary forms (e.g., phrases other than simple noun phrases) from microblogs. The burst (i.e., the rapid increase of the occurrence) of a phrase causes the burst of overlapping N-grams including incomplete ones. In other words, bursty incomplete N-grams inevitably overlap bursty phrases. Thus, the proposed method performs the extraction of bursty phrases as the set cover problem in which all bursty N-grams are covered by a minimum set of bursty phrases. Experimental results using Japanese Twitter data showed that the proposed method outperformed word-based, noun phrase-based, and segmentation-based methods both in terms of accuracy and coverage.

1 Introduction

Background and motivation. Trends on microblogs reflect manifold real-world events including natural disaster, new product launch, television broadcasting, public speech, airplane accident, scandal and national holiday. To catch real-world events, not a few researchers and practitioners have sought ways to detect trends on microblogs. Trend detection often involves bursty phrase extraction, i.e., extracting phrases of which occurrence rate in microblog texts posted within a certain period of time (and from a certain location) is much higher than that of the normal state. Extracted bursty phrases are directly used as trends as Twitter[1] officially provides, or sent to higher-order processes such as clustering and event labeling.

Bursty phrases on microblogs are likely to be noun phrases, but sometimes deviate from such standards. The title of a movie, song, game or any creation can be an arbitrary form like a long and/or general phrase (e.g., Spielberg's movie "catch me if you can", the Beatles' song "let it be")[2]. A memorable phrase (e.g., Steve Jobs's phrase "stay hungry, stay foolish") can also be a bursty phrase on microblogs. Even numbers and symbols can be potentially bursty phrases (e.g., "1984" can be a novel, "!!!" can be an artist). Any filtering rule such as stop word removal, part-of-speech (POS) tag restrictions, or length limit can mistakenly filter out bursty phrases.

Extracting irregularly-formed bursty phrases as described in the previous paragraph is difficult since no restriction can be used anymore to filter out incomplete N-grams. However, they are rare and little influence the overall accuracy even if they are correctly extracted. Not only that, tackling such difficult and rare cases easily leads to extracting many incomplete N-grams and deteriorating the accuracy. Almost all existing work has therefore ignored difficult and rare cases, and concentrated on extracting simple phrases (e.g., unigrams, bi-grams, tri-grams, or noun phrases identified by POS taggers). By sacrificing minorities, most bursty phrases can be extracted with high accuracy. Thus, irregularly-formed phrases have been abandoned in bursty phrase extraction (and alike in many text analysis tasks).

Contributions. In this work, we aim to accurately and exhaustively extract bursty phrases of arbitrary forms from microblogs. The challenge here is: How do we avoid extracting bursty incomplete N-grams without introducing any filtering rule? To solve this challenging problem, we propose a set cover-based method. We found that the

[1] https://twitter.com/.

[2] We used old but popular titles for illustrative purposes.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2358–2367
Copenhagen, Denmark, September 7–11, 2017.

Table 1: Representative trend detection methods based on bursty phrases.

Method	Unit of process	Measure of burst
(Sayyadi et al., 2009)	Noun phrase	TF, DF, IDF
(O'Connor et al., 2010)	Uni-gram, bi-gram, tri-gram	Burstiness
(Mathioudakis and Koudas, 2010)	Uni-gram	Burstiness
(Weng and Lee, 2011)	Uni-gram	DF-IDF, H-measure (wavelet analysis)
(Metzler et al., 2012)	Uni-gram	Burstiness
(Li et al., 2012)	N-gram of any length	Deviation from Gaussian distribution
(Cui et al., 2012)	Hashtag	Deviation from Gaussian distribution
(Benhardus and Kalita, 2013)-1	Uni-gram, bi-gram, tri-gram	Burstiness
(Benhardus and Kalita, 2013)-2	Uni-gram	TF-IDF, entropy
(Aiello et al., 2013)-1	Uni-gram	Burstiness
(Aiello et al., 2013)-2	Uni-gram, bi-gram, tri-gram	DF-IDF
(Abdelhaq et al., 2013)	Uni-gram	Deviation from Gaussian distribution
(Schubert et al., 2014)	Uni-gram (pair)	Deviation from Gaussian distribution
(Feng et al., 2015)	Hashtag	Deviation from Gaussian distribution

burst (i.e., the rapid increase of occurrence) of a phrase causes the burst of overlapping incomplete N-grams. For example, if phrase "let it be" bursts, the occurrence of some overlapping N-grams such as "let it", "let it be is", and "it be is" inevitably increases, possibly generating bursty incomplete N-grams. Given that bursty incomplete N-grams always accompany overlapping bursty phrases, we can avoid extracting bursty incomplete N-grams using the set cover problem (Chvátal, 1979). The proposed set cover-based method finds a minimum set of bursty phrases that cover all bursty N-grams including incomplete ones. Because the set cover problem is NP-complete, the proposed method approximately solves it by iteratively choosing an N-gram that most covers remaining bursty N-grams.

The advantages of the proposed method are as follows. **1) Exhaustive.** The proposed method can extract bursty (contiguous) phrases of arbitrary forms. In our experiment, the coverage has been shown to be larger than that of word-based, noun phrase-based, and segmentation-based methods. **2) Accurate.** With adequate preprocessing of auto-generated texts, the proposed method achieved 99.3% of precision for top 10 bursty phrases and 97.3% for top 50 bursty phrases, which were even higher than the comparative methods. **3) Language-independent.** Because the proposed method processes texts as a sequence of characters (or words), it works in any languages including those without word boundary such as Japanese. **4) Purely data-driven.** The proposed method only requires raw microblog texts and does not need external resources.

2 Related Work

Much work has been focused on trend detection or event detection from microblogs. Majority of representative trend detection methods (summarized in Table 1) start with extracting bursty phrases, often followed by clustering bursty phrases in order to link them to real-world events. Others first build clusters of words (uni-grams) by using word co-occurrence (Pervin et al., 2013) or topic models (Aiello et al., 2013; Diao et al., 2012; Lau et al., 2012) and then apply burst detection for clusters. Also, there are different approaches such as the document-based approach (Aiello et al., 2013), sketch-based model (Xie et al., 2013) and bursty biterm topic model (Yan et al., 2015).

It is noteworthy that most methods in Table 1 only focus on uni-grams, short N-grams (up to tri-grams), or noun phrases (or rely on hashtags). This is because majority of bursty phrases conform to such simple forms. The rest of bursty phrases are rare but their forms are irregular and difficult to stereotype by using rules. Trying to extract such irregularly-formed phrases easily leads to the deterioration of the precision due to incorrect extraction. Also, the recall can hardly be increased since they are only a small portion of all bursty phrases. To balance the precision and recall of bursty phrase extraction, focusing on simple phrases and ignoring rare cases is a reasonable strategy. In the field of trend detection on microblogs, this ignoring-minority strategy has become a de facto standard. However, it always fails to extract irregularly-formed bursty phrases. In this work, we tackle the challenge of extracting bursty phrases without any restriction of forms.

Among methods in Table 1, Li et al. (Li et al., 2012) have only attempted to extract bursty phrases of arbitrary forms. Their method, Twevent, applies text segmentation (or chunking) before measuring the degree of the burst. Every microblog text is represented as a sequence of word N-grams called segments. N-gram length, Symmetric Conditional Probability (SCP) (da Silva and Lopes, 1999), and semantic resources extracted from Wikipedia are integrated to obtain the best segmentation results. Owing to a good segmentation algorithm, it can potentially detect bursty phrases other than noun phrases, uni-grams, bi-grams, and tri-grams with high accuracy. However, it is still possible to miss irregularly-formed bursty phrases because they are likely to be segmented incorrectly. Our set cover-based method guarantees that all bursty N-grams including irregularly-formed ones must be covered by extracted bursty phrases. Thus, it is unlikely to miss irregularly-formed bursty phrases.

One more thing to note in Table 1 is that how to measure the degree of the burst can be largely classified into a few groups: burstiness, TF-IDF-based measures, and distribution-based methods. The simplest approach is burstiness, which is the ratio of the occurrence rate in target and reference document sets. Reference document set is usually constructed from past microblog texts. TF-IDF-based measures compute term frequency (TF) or document frequency (DF) in the target document set and inverse document frequency (IDF) in the reference document set. Distribution-based methods generally measure how much the observed frequency deviates from the distribution of the normal state using standard scores (z-scores). The Poisson distribution is proper to represent the number of occurrence of phrases, but the Gaussian distribution is often used as its approximation due to computational reasons. We in this work primarily adopt a Gaussian distribution-based approach and use the z-score as the measure of the burst because it reasonably works well for the different magnitude of the number of occurrence.

3 Bursty Phrase Extraction

3.1 Problem Formulation

We formalize the problem of bursty phrase extraction from microblogs. Target document set D_T is a set of microblog messages posted within a certain period of time (e.g., one day, three hours). Reference document set D_R is a set of microblog messages posted before the target time. Both are usually limited to certain locations or languages. Each document is a sequence of characters (or words). The objective here is to extract bursty complete phrases from D_T as much as possible using D_R as the normal state. The output format is a list of bursty N-grams $L = [g_1, g_2, \cdots, g_{|L|}]$ (g_i is an N-gram or a sequence of characters) ranked by the degree of the burst. The accuracy and coverage of top K (K is a user-specified parameter) bursty phrases are important evaluation criteria.

The degree of the burst for N-grams in D_T is defined as the z-score when the Gaussian distribution is estimated from D_R. While most N-grams occur once in a single microblog message, a few N-grams are repeatedly used in it. We thus employ the document frequency-based z-score as the degree of the burst. The z-score of N-gram g is specifically computed as

$$zscore(g) = \frac{df(g) - \mu(g)}{\sigma(g)} \tag{1}$$

where $df(g)$ is the document frequency of g in D_T, and $\mu(g)$ and $\sigma(g)$ are respectively the mean and standard deviation for the document frequency of g estimated from D_R. Given that the Gaussian distribution used here is the approximation of the Poisson distribution, $\sigma(g)$ is approximated by $\sqrt{\mu(g)}$. To smooth $\mu(g)$ when g never occurs in D_R, we add $\delta = 1$ to $\mu(g)$.

3.2 Basic Idea

To exhaustively extract bursty phrases without any restriction of their forms, we have to refrain from using filtering rules such as stop word removal, POS tag restrictions, and length limit. Without filtering rules, there are far more bursty incomplete N-grams than bursty complete phrases. It is challenging to extract bursty phrases including irregularly-formed ones and at the same time to avoid extracting bursty incomplete N-grams. Is there any evident difference between bursty incomplete N-grams and bursty phrases of irregular forms?

We scrutinized Twitter data and found the following fact: Bursty incomplete N-grams always accompany overlapping bursty phrases provided that the definition of the burst is appropriate. We

Algorithm 1: Greedy Set Cover Algorithm for Bursty Phrase Extraction

```
   Input: Target document set $D_T$, reference document set $D_R$
   Output: Ranked list of bursty phrases $L$
1  Initialize $L$;
2  Get a set of valid N-grams $G_V = \{g_i\}$ satisfying $burstiness(g_i) \geq 1 + \epsilon$;
3  Get a set of bursty N-grams $G_B = \{g_i\} \subset G_V$ satisfying $zscore(g_i) \geq \theta$ and $zscore_{tf}(g_i) \geq \theta$;
4  Discard trivial N-grams from $G_V$;
5  while $G_B \neq \emptyset$ do
6      Select $g \in G_V$ that most cover the occurrence of bursty N-grams in $G_B$;
7      Get a set of longer N-grams $G_g \subset G_V$ containing $g$;
8      Determine a set of containment N-grams $G_C \subset G_g$ for $g$;
9      if $G_C = \emptyset$ then
10         Push $g$ into $L$;
11         Negate the occurrence of all N-grams in $G_B$ and $G_V$ where $g$ overlaps;
12         Delete $g_i \in G_B$ if it does not satisfy $zscore_{tf}(g_i) \geq \theta$;
13     else
14         Negate the occurrence of $g$ where containment N-grams $g_i \in G_C$ overlap;
15     end
16 end
17 Rerank $L$ based on the actual z-score;
```

explain this phenomenon in due order. When many microblog users intensively use a certain phrase, it becomes a bursty phrase. Here, the increment of the occurrence of the phrase contributes to the increment of the occurrence of overlapping N-grams. Consequently, (incomplete) N-grams that overlap the phrase can also burst. Thus, bursty incomplete N-grams always have their original bursty phrases that overlap each other.

Based on the phenomenon described in the previous paragraph, bursty incomplete N-grams cease bursting if their original bursty phrases disappear from microblog texts. In other words, all bursty N-grams including incomplete ones can be covered (overlapped) by bursty phrases. Given that bursty phrases cause bursty incomplete N-grams but the reverse was not true, we can formalize the extraction of bursty phrases as the set cover problem (Chvátal, 1979) where a minimum number of bursty phrases are selected to cover all bursty N-grams. When all selected bursty phrases are removed from microblog texts, it is guaranteed that there is no bursty N-gram.

3.3 Proposed Algorithm

Algorithm 1 is a pseudo-code of the greedy set cover algorithm for bursty phrase extraction. As formulated in Section 3.1, the input data is target D_T and reference document sets D_R of microblog messages. The output is a ranked list of bursty phrases $L = [g_1, g_2, \cdots, g_{|L|}]$. Basically, Algorithm 1 iteratively selects an N-gram that most covers the occurrence of bursty N-grams (Line 6) until all bursty N-grams are covered. In the following, we describe points of the Algorithm 1.

3.3.1 Bursty N-grams and Valid N-grams

A set of bursty N-grams G_B in Algorithm 1 (Line 3) corresponds to the universe of the set cover problem which should be all covered. Bursty N-grams satisfy z-score threshold θ both in document frequency-based (Eq. (1)) and term frequency-based z-scores (Eq. (2)).

$$zscore_{tf}(g) = \frac{tf(g) - \mu_{tf}(g)}{\sigma_{tf}(g)} \tag{2}$$

Here, $tf(g)$ is the term frequency of g in D_T, and $\mu_{tf}(g)$ and $\sigma_{tf}(g)$ are respectively the mean and standard deviation for the term frequency of g estimated from D_R. The term frequency-based z-score is used to judge whether a bursty N-gram in G_B still bursts when its occurrences are partly covered. This is required to handle N-grams repeatedly occurring in a single microblog message.

Valid N-grams in G_V (Line 2) are qualified to be bursty phrases to cover G_B. We differentiate bursty N-grams and valid N-grams (specifically, $G_B \subset G_V$) to handle threshold problems.

Namely, it is possible that bursty incomplete N-grams satisfy the z-score threshold but their original bursty phrases do not satisfy the threshold. The criterion of the valid N-gram is defined by using burstiness.

$$burstiness(g) = \frac{df(g)}{df_R(g)} \cdot \frac{|D_R|}{|D_T|} \qquad (3)$$

Here, $df_R(g)$ is the document frequency of N-gram g in D_R. To avoid division by zero, smoothing term $\delta = 1$ is added to $df_R(g)$. When $burstiness(g)$ satisfies threshold $1 + \epsilon$ (i.e., the occurrence of g actually increases), g becomes a valid N-gram. To reduce pointless processes, trivial N-grams that can never be phrases (e.g., starting or ending with spaces, occurring only as a part of a sole longer N-gram) are discarded (Line 4).

3.3.2 Occurrence-based Set Covering

Whereas the standard set cover problem assumes that each item is atomic, i.e., the state of an item is either *not covered* or *covered*, the proposed method manages the state of covering by using all occurrences of bursty N-grams. When a valid N-gram is selected (Line 10), the occurrence of all N-grams in G_B and G_V that the valid N-gram overlaps is negated (Line 11). Here, a bursty N-gram in G_B is completely covered if the term frequency-based z-score computed from remaining occurrences of the N-gram does not satisfy the threshold (Line 12).

Occurrence-based set covering can solve the case when a bursty N-gram is covered by multiple bursty phrases. That is, the bursty N-gram is not completely covered even if one of the bursty phrases is selected. For example, given two bursty phrases “let it be” and “let it go” (a song), incomplete N-gram “let it” ceases bursting only when the occurrence of both phrases is negated. Also, it can handle partially overlapping N-grams (e.g., “let it be” and “be is”) based on the number of overlaps.

3.3.3 Containment N-grams

N-grams that are contained in multiple phrases should be carefully treated in the set cover problem. Shorter N-grams are likely to be contained in more phrases and chosen in the set cover problem even if they are incomplete. For example, when phrases “let it be” and “let it go” burst, shared incomplete N-gram “let it” is likely to cover the occurrence of bursty N-grams more than the two phrases. To prevent selecting shared incomplete N-grams, we determine containment relations between an N-gram and longer N-grams containing it (Lines 7, 8). We define longer N-grams in containment relations as containment N-grams. Containment relations negate the occurrence of the shorter N-gram where containment N-grams overlap (Line 14). Containment relation is inspired by the idea of rectified frequency used in a segmentation-based quality phrase mining method (Liu et al., 2015), though how to rectify the frequency is different. Note that containment relations do not necessarily mean that shorter N-grams are incomplete because both shorter and longer N-grams can be phrases (e.g., “new york” and “new york times”).

How to determine containment relations is designed not to contradict the greedy set cover algorithm. Briefly, containment relations hold when only the containment N-grams among longer N-grams can cover the occurrence of bursty N-grams more than the shorter N-gram owing to the containment relations. That is, we find a stable state of containment relations. To find a stable state, we initially define temporal containment relations and then iteratively find a set of containment N-grams so that containment relations become stable.

Initial containment N-grams are determined using burst context variety, which is an extension of accessor variety (Feng et al., 2004). Accessor variety roughly measures how much an N-gram is likely to be a phrase. It specifically counts the number of distinct characters (or words) before or after the N-gram and employs the minimum one. The drawback of accessor variety is that it handles left and right contexts independently. We thus modify it as context variety in which both left and right contexts are simultaneously counted. Context variety can also be calculated using the set cover problem. In particular, a left or right character (or word) that most covers the occurrence of the N-gram is iteratively selected until all the occurrences are covered. Burst context variety is a further extension of context variety to count the number of contexts only for additional term frequencies given the mean of the term frequency. When the burst context variety of an N-gram is not greater than that of a longer N-gram, we extract the context (i.e., left or right character) and define all longer N-grams having the context as initial containment N-grams.

There are two minute settings for measuring burst context variety. One is that the start and end of every line are all unique and should be counted as distinct contexts. The other is that symbols[3] should be ignored when checking left and right contexts of the N-gram.

3.3.4 Reranking Bursty Phrases

The output L is finally reranked by using actual z-scores (Line 17), which are different from z-scores calculated from raw document frequency. The actual z-score is calculated from the number of occurrences of bursty N-grams that N-gram $g \in L$ actually covered during the set cover process. Since a single valid N-gram usually covers multiple bursty N-grams, the z-score for every covered bursty N-gram is recalculated and the maximum is used as the actual z-score. When the maximum z-score exceeds the original z-score, the original z-score is preserved. Without reranking, incomplete N-grams that covered very few occurrences of bursty N-grams may be overestimated.

4 Evaluation

We evaluated the proposed method using two weeks of Japanese Twitter data.

4.1 Setup

Dataset. We created a dataset using Twitter Streaming API statuses/sample. We chose Japanese as a language because it was one of popular languages used in Twitter and because it has no word boundary and finding phrases is difficult compared to space-delimited languages such as English. We specifically collected 15 days of tweets from September 30 to October 14, 2016[4]. For each day from October 1 to 14, we extracted bursty phrases in reference to the previous day.

To maximally alleviate the influence of auto-generated contents such as tweets posted by bots and spammers, we filtered out them. Detecting bots and spammers (Chu et al., 2010; Subrahmanian et al., 2016) is a nontrivial research task and out of the scope of this paper. In this experiment, we used simple but effective heuristics. First, we only used tweets posted by Twitter official clients[5] because they were mainly used by normal users. Second, we discarded tweets including URLs because most spammers tried to lure users to visit their websites. Third, retweets (actions to propagate someone's tweets) were discarded. The maximum and minimum numbers of remaining tweets per day were 326,002 (Oct. 2) and 253,044 (Oct. 13), respectively. Additionally, we deleted hashtags (starting by #) and mentions (starting by @) from tweets. Note that the degree of the burst for URLs, hashtags, and mentions can be independently measured. We concentrated on extracting bursty phrases from plain texts.

Ground truth. To create the ground truth, we mixed N-grams extracted with all methods and then manually annotated each N-gram by checking its real usage in tweets. While most N-grams were easily judged as complete phrase (i.e., *correct*) or incomplete N-grams (i.e., *incorrect*), a few N-grams were difficult to judge (e.g., a last name that was not frequently used to indicate the person in tweets). We annotated such N-grams as *maybe correct* and regarded as correct in the strict case and incorrect in the loose case when measuring evaluation metrics.

Evaluated methods. In the proposed method (**Proposed**), we processed Japanese texts as sequences of characters. Default threshold parameters θ and ϵ were set to 10 and 0.5, respectively. Comparative methods included the word-based method (**Word**), noun phrase-based method (**NP**), and segmentation-based method (**Segment**) (Li et al., 2012). Because these methods require word breaking, we used MeCab (Kudo et al., 2004) (version 0.996, ipadic as a dictionary), a Japanese morphological analyzer. The word-based method uses all self-sufficient words as candidate phrases. The noun phrase-based method regards concatenated successive nouns as a candidate phrase. In word-based and noun phrase-based methods, the dictionary or vocabulary significantly affects the performance. Thus, we also used neologd[6] (Sato et al., 2017) (version v0.0.5 updated at May 2, 2016), a neologism dictionary extracted from many language resources on the web, as an additional dictionary (**+Dic**). The segmentation-based method detects segments (chunks) from a sequence of words as candidate phrases. The segmentation model was constructed using three

[3] In our experiments, we used isalnum (or iswalnum for wide characters) in C++ standard library to distinguish words and symbols.

[4] We considered that a day changed at 4 am JST.

[5] `https://about.twitter.com/products/list`.

[6] `https://github.com/neologd/mecab-ipadic-neologd`.

Table 2: Average precision of top K bursty phrases (strict case).

Method	Top10	Top20	Top30	Top40	Top50
Word	0.700	0.714	0.731	0.743	0.743
Word+Dic	0.929	0.907	0.895	0.907	0.904
NP	0.936	0.929	0.936	0.927	0.930
NP+Dic	0.971	0.968	0.962	0.955	0.954
Segment	0.921	0.918	0.905	0.918	0.914
Proposed	**0.993**	**0.993**	**0.981**	**0.979**	**0.973**

Table 3: Average precision of top K bursty phrases (loose case).

Method	Top10	Top20	Top30	Top40	Top50
Word	0.764	0.779	0.812	0.821	0.817
Word+Dic	0.950	0.936	0.936	0.945	0.941
NP	0.950	0.943	0.952	0.945	0.946
NP+Dic	0.971	0.979	0.974	0.971	0.973
Segment	0.957	0.954	0.952	0.955	0.953
Proposed	**0.993**	**0.996**	**0.998**	**0.995**	**0.991**

months of tweets (from July 1 to Sept. 30, 2016) and Japanese Wikipedia dump data (as of Oct. 1, 2016).

Evaluation metrics. We employed precision and min-z-score of top K bursty phrases (K was set to 10, 20, 30, 40, or 50) as evaluation metrics. We measured the precision both in strict and loose cases based on the ground truth labels. The min-z-score (the minimum of the z-score, computed from raw document frequency) was introduced to evaluate how much the top K output ranked by z-scores included highly bursty phrases. To increase the min-z-score, all the top K phrases should have high z-scores and hence highly bursty phrases should not be ignored. Thus the min-z-score can evaluate the coverage of extracted bursty phrases using a fixed size of the output. Higher precision and min-z-score indicate that the method can more accurately and exhaustively extract bursty phrases.

4.2 Performance Results: Precision

Tables 2, 3 show the precision of bursty phrase extraction. It was surprisingly that the proposed method achieved higher precision than noun-phrase based methods, which were supposed to be safety by sacrificing irregularly-formed phrases. The precision of the proposed method for top 50 bursty phrases was 97.3% (correct phrases were 681 out of 700) in the strict case and 99.1% (694 out of 700) in the loose case. The precision for top 10 bursty phrases was 99.3% (139 out of 140) even in the strict case. The results demonstrate that the burst information alone can accurately find the boundary of bursty phrases using the set cover problem. Error cases of the proposed method were largely classified into two: base sequences of diversified expressions and phrases with strongly-correlated attached characters.

In comparative methods, the accuracy tended to be high when noun phrases were used and the dictionary was well defined. Especially, the use of the neologism dictionary boosted the precision. The segmentation-based method also marked moderately high precision.

Table 4: Average min-z-score of top K bursty phrases.

Method	Top10	Top20	Top30	Top40	Top50
Word	41.5	25.7	20.4	17.2	15.3
Word+Dic	44.7	29.0	22.5	19.3	16.7
NP	42.6	26.4	20.4	16.7	14.6
NP+Dic	42.7	28.8	21.0	17.6	15.3
Segment	45.0	31.2	24.3	20.5	18.0
Proposed	**50.1**	**33.5**	**24.8**	**21.5**	**19.4**

4.3 Performance Results: Coverage

Table 4 shows the min-z-score of bursty phrase extraction. The proposed method achieved higher min-z-score than the comparative methods. This was because the proposed method extracted bursty phrases regardless of their forms. Noun phrase-based methods tended to miss highly bursty phrases of irregular forms. Therefore the min-z-score of extracted top K bursty phrases became small. Among comparative methods, the segmentation-based method best achieved the min-z-score since it did not restrict the form of phrases. However, it was still possible to miss very irregular phrases due to segmentation mistakes and the min-z-score was less than that of the proposed method. The use of the neologism dictionary increased the min-z-score as well as the precision, indicating that it had no negative effect.

To intuitively assess the coverage of the proposed method, we manually counted the number of bursty phrases that were extracted with the proposed method (when $K = 10$, i.e., 139 correct phrases) but completely missed with the comparative methods. Here, we regarded *completely missed* when neither of the bursty phrase, overlapping N-grams including incomplete ones nor orthographic variants were extracted in top

Table 5: Number of bursty phrases extracted with the proposed method ($K = 10$) but completely missed with comparative methods.

Word	Word+Dic	NP	NP+Dic	Segment
17/139	7/139	10/139	4/139	8/139

Table 6: Average precision of top K bursty phrases with different parameter settings in the proposed method (strict case).

Parameters	Top10	Top20	Top30	Top40	Top50
θ=5, ϵ=0.2	0.993	0.993	0.976	0.977	0.971
θ=5, ϵ=0.5	0.993	0.993	0.976	0.977	0.971
θ=5, ϵ=1.0	0.993	0.993	0.976	0.979	0.971
θ=10, ϵ=0.2	0.993	0.993	0.981	0.979	0.973
θ=10, ϵ=0.5	0.993	0.993	0.981	0.979	0.973
θ=10, ϵ=1.0	0.993	0.993	0.981	0.979	0.973
θ=15, ϵ=0.2	0.993	0.989	0.971	0.971	0.966
θ=15, ϵ=0.5	0.993	0.989	0.971	0.971	0.966
θ=15, ϵ=1.0	0.993	0.989	0.971	0.971	0.966

100 bursty N-grams. Table 5 shows the results. Although the percentages were small, any comparative method completely missed some highly bursty phrases that were extracted with the proposed method. Note that the proposed method did not completely miss top 10 bursty phrases of comparative methods at all since the set cover problem inevitably covered all bursty N-grams.

4.4 Influence of Parameter Settings

We changed threshold parameters θ and ϵ to evaluate their influence on performance. Tables 6, 7 show the performance results with different parameter settings. We confirmed that both parameters, especially ϵ, hardly affected the precision and min-z-score. The results indicate that the threshold parameters can be roughly set based on data.

4.5 Examples

Table 7: Average min-z-score of top K bursty phrases with different parameter settings in the proposed method.

Parameters	Top10	Top20	Top30	Top40	Top50
θ=5, ϵ=0.2	50.6	33.4	24.9	21.4	19.2
θ=5, ϵ=0.5	50.6	33.4	24.8	21.4	19.2
θ=5, ϵ=1.0	50.6	33.4	24.9	21.4	19.3
θ=10, ϵ=0.2	50.1	33.5	24.8	21.5	19.4
θ=10, ϵ=0.5	50.1	33.5	24.8	21.5	19.4
θ=10, ϵ=1.0	50.2	33.5	24.9	21.5	19.5
θ=15, ϵ=0.2	50.2	33.5	25.1	21.8	19.4
θ=15, ϵ=0.5	50.2	33.5	25.1	21.8	19.5
θ=15, ϵ=1.0	50.2	33.5	25.1	21.8	19.6

Table 8: Example of top 10 bursty phrases in Japanese (Oct. 1, 2016).

1.	三代目 (*Sandaime*, generally meaning "third", short name of a music group)
2.	坂口杏里 (*Sakaguchi Anri*, celebrity)
3.	**HE ★ VENS** (idol group)
4.	プリライ (*Pri-rai*, short name of a live concert)
5.	映っちゃった (*Utsucchatta*, generally meaning "it was reflected", short name of a TV program)
6.	単独 (*Tandoku*, generally meaning "singleness", implying a live concert without supporting acts)
7.	しやがれ (*Shiyagare*, generally a phrase used in an imperative sentence, short name of a TV program)
8.	うたプリ (*Utapri*, short name of an anime)
9.	**WORKING** (short name of an anime)
10.	うたぷり (*Utapri*, short name of an anime)

Table 8 shows top 10 bursty phrases (all of them are correct) on Oct. 1. This day contained many irregularly-formed phrases; phrases containing hiragana[7] characters (rank 5, 7, 8, 10), other than noun phrases (rank 5, 7), and containing symbols (rank 3). Especially, 映っちゃった (rank 5) and しやがれ (rank 7) were troublesome since they contain hiragana characters and at the same time they are other than noun phrases. Even with the neologism dictionary, the noun phrase-based method extracted しやがれ but missed 映っちゃった.

To demonstrate the language-independent nature of the proposed method, we also applied it to English with character-level processing[8]. Table 9 shows an example of bursty phrases extracted from English tweets. Whereas an incomplete phrase (rank 2) was extracted due to auto-generated contents that could not be eliminated from data, other top bursty phrases were correctly extracted. The proposed method also extracted a very long Internet meme (rank 5), which burst along with its counterpart "anyone that knows me knows i love _________." (rank 17).

5 Conclusions

We proposed a language-independent data-driven method to accurately and exhaustively extract bursty phrases of arbitrary forms from microblogs. We found that bursty incomplete N-grams always

[7] In Japanese, hiragana is mainly used for auxiliary words and thus difficult to break into words or phrases.

[8] Of course, we can process English in word level.

Table 9: Example of top 10 bursty phrases in English (Oct. 1, 2016, PST). Note that these bursty phrases were generated by processing English tweets in character level.

1.	**LizQuen InSydney Oct30** (maybe an auto-generated phrase used like a tag)
2.	**for The 100 most beautiful faces in 2016** (maybe an auto-generated sequence used to vote a celebrity for the ranking)
3.	**Lamar Jackson** (American football player)
4.	**Clemson** (American football team)
5.	**quote with what you think the answer is and copy this tweet to see what people say about you** (Internet meme)
6.	**Tennessee** (American football team)
7.	**Louisville** (American football team)
8.	**Milner** (American football player)
9.	**FSU** (American football team)
10.	**Louis Walsh** (talent manager)

accompany overlapping bursty phrases by ascertaining the mechanism why incomplete N-grams burst. Based on the findings, the proposed method solves the extraction of bursty phrases as the set cover problem where a minimum set of bursty phrases covers all bursty N-grams including incomplete ones. We confirmed from experimental results that the proposed method outperformed noun phrase-based and segmentation-based methods both in terms of the accuracy and coverage. The source code of the proposed method is publicly available[9].

The future work includes the reduction or estimation of the computation time and memory usage. They increase as the target document set grows or, specifically the number of occurrences of bursty N-grams increases. Also, handling auto-generated contents is an important issue. The proposed method should be used with effective methods of identifying auto-generated contents, bots, and spammers in microblogs.

Acknowledgments

This research is partially supported by the Grant-in-Aid for Scientific Research (A)(2620013) of the Ministry of Education, Culture, Sports, Science and Technology, Japan, and JST, Strategic International Collaborative Research Program, SICORP.

[9] `https://github.com/iwnsew/sc-bursty-phrase`.

References

Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. 2013. EvenTweet: Online Localized Event Detection from Twitter. *Proceedings of the VLDB Endowment*, 6(12):1326–1329.

Luca Maria Aiello, Georgios Petkos, Carlos Martin, David Corney, Symeon Papadopoulos, Ryan Skraba, Ayse Göker, Ioannis Kompatsiaris, and Alejandro Jaimes. 2013. Sensing Trending Topics in Twitter. *IEEE Transactions on Multimedia*, 15(6):1268–1282.

James Benhardus and Jugal Kalita. 2013. Streaming Trend Detection in Twitter. *International Journal of Web Based Communities*, 9(1):122–139.

Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. 2010. Who is Tweeting on Twitter: Human, Bot, or Cyborg? In *Proceedings of Annual Computer Security Applications Conference (ACSAC)*, pages 21–30.

Václav Chvátal. 1979. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3):233–235.

Anqi Cui, Min Zhang, Yiqun Liu, Shaoping Ma, and Kuo Zhang. 2012. Discover Breaking Events with Popular Hashtags in Twitter. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)*, pages 1794–1798.

Qiming Diao, Jing Jiang, Feida Zhu, and Ee-Peng Lim. 2012. Finding Bursty Topics from Microblogs. In *Proceedings of Meeting of the Association for Computational Linguistics (ACL)*, pages 536–544.

Haodi Feng, Kang Chen, Xiaotie Deng, and Weimin Zheng. 2004. Accessor Variety Criteria for Chinese Word Extraction. *Computational Linguistics*, 30(1):75–93.

Wei Feng, Chao Zhang, Wei Zhang, Jiawei Han, Jianyong Wang, Charu Aggarwal, and Jianbin Huang. 2015. STREAMCUBE: Hierarchical Spatio-temporal Hashtag Clustering for Event Exploration over the Twitter Stream. In *Proceedings of IEEE International Conference on Data Engineering (ICDE)*, pages 1561–1572.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying Conditional Random Fields to Japanese Morphological Analysis. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 230–237.

Jey Han Lau, Nigel Collier, and Timothy Baldwin. 2012. On-line Trend Analysis with Topic Models: #twitter trends detection topic model online. In *Proceedings of International Conference on Computational Linguistics (COLING)*, pages 1519–1534.

Chenliang Li, Aixin Sun, and Anwitaman Datta. 2012. Twevent: Segment-based Event Detection from

Tweets. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)*, pages 155–164.

Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren, and Jiawei Han. 2015. Mining Quality Phrases from Massive Text Corpora. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 1729–1744.

Michael Mathioudakis and Nick Koudas. 2010. TwitterMonitor: Trend Detection over the Twitter Stream. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 1155–1158.

Donald Metzler, Congxing Cai, and Eduard Hovy. 2012. Structured Event Retrieval over Microblog Archives. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 646–655.

Brendan O'Connor, Michel Krieger, and David Ahn. 2010. TweetMotif: Exploratory Search and Topic Summarization for Twitter. In *Proceedings of International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 384–385.

Nargis Pervin, Fang Fang, Anindya Datta, Kaushik Dutta, and Debra Vandermeer. 2013. Fast, Scalable, and Context-Sensitive Detection of Trending Topics in Microblog Post Streams. *ACM Transactions on Management Information Systems*, 3(4):19:1–19:24.

Toshinori Sato, Taiichi Hashimoro, and Manabu Okumura. 2017. Implementation of a Word Segmentation Dictionary Called mecab-ipadic-NEologd and Study on How to Use It Effectively for Information Retrieval (in Japanese). In *Proceedings of Meeting of the Association for Natural Language Processing*, pages NLP2017–B6–1.

Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. 2009. Event Detection and Tracking in Social Streams. In *Proceedings of International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 311–314.

Erich Schubert, Michael Weiler, and Hans-Peter Kriegel. 2014. SigniTrend: Scalable Detection of Emerging Topics in Textual Streams by Hashed Significance Thresholds. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 871–880.

Joaquim Ferreira da Silva and Jos'e Gabriel Pereira Lopes. 1999. A Local Maxima Method and a Fair Dispersion Normalization for Extracting Multi-word Units from Corpora. In *Proceedings of Meeting on Mathematics of Language (MOL)*, pages 369–381.

V.S. Subrahmanian, Amos Azaria, Skylar Durst, Vadim Kagan, Aram Galstyan, Kristina Lerman, Linhong Zhu, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. 2016. The DARPA Twitter Bot Challenge. *IEEE Computer*, 49(6):38–46.

Jianshu Weng and Bu-Sung Lee. 2011. Event Detection in Twitter. In *Proceedings of International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 401–408.

Wei Xie, Feida Zhu, Jing Jiang, Ee-Peng Lim, and Ke Wang. 2013. TopicSketch: Real-Time Bursty Topic Detection from Twitter. In *Proceedings of IEEE International Conference on Data Mining (ICDM)*, pages 837–846.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2015. A Probabilistic Model for Bursty Topic Discovery in Microblogs. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pages 353–359.

Maximum Margin Reward Networks for Learning from Explicit and Implicit Supervision

Haoruo Peng[1] **Ming-Wei Chang**[2] **Wen-tau Yih**[2]
[1]University of Illinois, Urbana-Champaign
[2]Microsoft Research, Redmond
[1]hpeng7@illinois.edu [2]{minchang,scottyih}@microsoft.com

Abstract

Neural networks have achieved state-of-the-art performance on several structured-output prediction tasks, trained in a fully supervised fashion. However, annotated examples in structured domains are often costly to obtain, which thus limits the applications of neural networks. In this work, we propose Maximum Margin Reward Networks, a neural network-based framework that aims to learn from both explicit (full structures) and implicit supervision signals (delayed feedback on the correctness of the predicted structure). On named entity recognition and semantic parsing, our model outperforms previous systems on the benchmark datasets, CoNLL-2003 and WebQuestionsSP.

1 Introduction

Structured-output prediction problems, where the goal is to determine values of a set of interdependent variables, are ubiquitous in NLP. Structures of such problems can range from simple sequences like part-of-speech tagging (Ling et al., 2015) and named entity recognition (Lample et al., 2016), to complex syntactic or semantic analysis such as dependency parsing (Dyer et al., 2015) and semantic parsing (Dong and Lapata, 2016). State-of-the-art methods of these tasks are often neural network models trained using fully annotated structures, which can be costly or time-consuming to obtain. Weakly supervised learning settings, where the algorithm assumes only the existence of *implicit* signals on whether a prediction is correct, are thus more appealing in many scenarios.

For example, Figure 1 shows a weakly supervised setting of learning semantic parsers using only question–answer pairs. When the system generates a candidate semantic parse during training, the quality needs to be indirectly measured by comparing the derived answers from the knowledge base and the provided labeled answers.

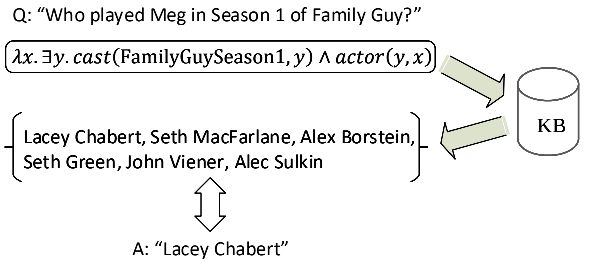

Q: "Who played Meg in Season 1 of Family Guy?"
$\lambda x.\exists y.\,cast(\text{FamilyGuySeason1}, y) \wedge actor(y, x)$
KB
Lacey Chabert, Seth MacFarlane, Alex Borstein, Seth Green, John Viener, Alec Sulkin
A: "Lacey Chabert"

Figure 1: Learning a semantic parser using *implicit* supervision signals (labeled answers). Since there are no gold parses, a model needs to explore different parses, where their quality can only be indirectly verified by comparing retrieved answers and the labeled answers.

This setting of implicit supervision increases the difficulty of learning a neural model, not only because the signals are vague and noisy, but also *delayed*. For instance, among different semantic parses that result in the same answers, typically only few of them correctly represent the meaning of the question. Moreover, the correctness of answers corresponding to a parse can only be evaluated through an external oracle (e.g., executing the query on the knowledge base) *after* the parse is fully constructed. Early model update before the search of a full semantic parse is complete is generally infeasible.[1] It is also not clear how to leverage implicit and explicit signals integrally during learning when both kinds of labels are present.

In this work, we propose Maximum Margin Reward Networks (MMRN), which is a general neural network-based framework that is able to learn from both implicit and explicit supervision signals. By casting structured-output learning as a search problem, the key insight in MMRN is the

[1] Existing weakly supervised methods (Clarke et al., 2010; Artzi and Zettlemoyer, 2013) often leverage domain-specific heuristics, which are not always available.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2368–2378
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

special mechanism of *rewards*. Rewards can be viewed as the training signals that drive the model to explore the search space and to find the correct structure. The explicit supervision signals can be viewed as a source of *immediate* rewards, as we can often instantly know the correctness of the current action. On the other hand, the implicit supervision can be viewed as a source of *delayed* rewards, where the reward of the actions can only be revealed later. We unify these two types of reward signals by using a maximum margin update, inspired by structured SVM (Joachims et al., 2009).

The effectiveness of `MMRN` is demonstrated on three NLP tasks: *named entity recognition*, *entity linking* and *semantic parsing*. `MMRN` outperforms the current best results on CoNLL-2003 named entity recognition dataset (Tjong Kim Sang and De Meulder, 2003), reaching 91.4% F_1, in the close setting where no gazetteer is allowed. It also performs comparably to the existing state-of-the-art systems on entity linking. Models for these two tasks are trained using explicit supervision. For semantic parsing, where only implicit supervision signals are provided, `MMRN` is able to learn from delayed rewards, improving the entity linking component and the overall semantic parsing framework jointly, and outperforms the best published system by 1.4% absolute on the WebQSP dataset (Yih et al., 2016).

In the rest of the paper, we survey the most related work in Sec. 2 and give an in-depth discussion on comparing `MMRN` and other learning frameworks in Sec. 7. We start the description of our method from the search formulation and the state–action spaces in our targeted tasks in Sec. 3, followed by the reward and learning algorithm in Sec. 4 and the detailed neural model design in Sec. 5. Sec. 6 reports the experimental results and Sec. 8 concludes the paper.

2 Related Work

Structured output prediction tasks have been studied extensively in the field of natural language processing (NLP). Many supervised structured learning algorithms has been proposed for capturing the relationships between output variables. These models include structured perceptron (Collins, 2002; Collins and Roark, 2004), conditional random fields (Lafferty et al., 2001), and structured SVM (Taskar et al., 2004; Joachims et al., 2009). Later, the *learning to search* framework is proposed (Daumé and Marcu, 2005; Daumé et al., 2009), which casts the structured prediction task as a general search problem. Most recently, recurrent neural networks such as LSTM models (Hochreiter and Schmidhuber, 1997) have been used as a general tool for structured output models (Vinyals et al., 2015).

Latent structured learning algorithms address the problem of learning from incomplete labeled data (Yu and Joachims, 2009; Quattoni et al., 2007). The main difference compared to our framework is the existence of the external environment when learning from implicit signals.

Upadhyay et al. (2016) first proposed the idea of learning from implicit supervision, and is the most related paper to our work. Compared to their linear algorithm, our framework is more principled and general as we integrate the concept of margin in our method. Furthermore, we also extend the framework using neural models.

3 Search-based Inference

In our framework, predicting the best structured output, *inference*, is formulated as a state/action search problem. Our search space can be described as follows. The initial state, s_0, is the starting point of the search process. We define $\gamma(s)$ as the set of all feasible actions that can be taken at s, and denote $s' = \tau(s,a)$ as the transition function, where s' is the new state after taking action a from s. A path $\mathbf{h}$ is a sequence of state–action pairs, starting with the initial state: $\mathbf{h} = \{(s_0, a_0), \ldots, (s_k, a_k)\}$, where $s_i = \tau(s_{i-1}, a_{i-1}), \forall i = 1, \ldots, k$. We denote $\mathbf{h} \rightsquigarrow \hat{s}$, if $\hat{s} = \tau(s_k, a_k)$, the final state which the path $\mathbf{h}$ leads to. A path essentially is a partial or complete structured prediction. For each input x, we define $\mathcal{H}(x)$ to be the set of all possible paths for the input. We also define $\mathcal{E}(x) = \{\mathbf{h} \mid \mathbf{h} \in \mathcal{H}(x), \mathbf{h} \rightsquigarrow \hat{s}, \gamma(\hat{s}) = \emptyset\}$, which is all possible paths that lead to terminal states.

Given a state s and an action a, the scoring function $f_\theta(s,a)$ measures the quality of an immediate action with respect to the current state, where θ is the model parameters. The score of a path $\mathbf{h}$ is defined as the sum of the scores for state-action pairs in $\mathbf{h}$: $f_\theta(\mathbf{h}) = \sum_{i=0}^{k} f_\theta(s_i, a_i)$. During test time, inference is to find the *best* path in $\mathcal{E}(x)$: $\arg\max_{\mathbf{h}\in\mathcal{E}(x)} f_\theta(\mathbf{h}; x)$. In practice, inference is often approximated by beam search when no efficient algorithm exists.

In the remaining of this section, we describe the states and actions in the targeted tasks in this work: *named entity recognition*, *entity linking* and *semantic parsing*. The the model and learning algorithm will be discussed in Sec. 4 and Sec. 5.

3.1 Named entity recognition

The task of named entity recognition (NER) is to identify entity mentions in a sentence, as well as to assign their types, such as Person or Location. Following the conventional setting, we treat it as a sequence labeling problem using the standard BIOES encoding. For instance, a "B-LOC" tag on a word means that the word is the beginning of a multi-word location entity.

Given a sentence as input, the states represent the tags assigned to the words. Starting from the initial state, s_0, where no tag has been assigned, the search process explores the sequence tagging from the left-to-right order. For each word, the actions are the legitimate tags that can be assigned to it, which depend on previous actions. For example, if the "S-PER" tag ("S" means a single word entity) has been assigned to the previous word, then an action of labeling the current word with either "I-PER" or "E-PER" cannot can be taken. The search reaches a terminal state when all words in the sentence have been tagged.

3.2 Entity linking

The problem of entity linking (EL) is similar to NER, but instead of tagging the mention using a small set of generic entity types, the goal here is to ground the mention to a specific entity, stored in a knowledge base or described by a Wikipedia page. For example, consider the sentence "*nfl news: draft results for giants*" and assume that the mention candidates "*nfl*" and "*giants*" are given. A state reflects how we have assigned the entity labels to these candidates. Following the same left-to-right order and starting from the empty assignment s_0, the first action to take is to assign the entity label to the first candidate "*nfl*". A legitimate action set can be all the entities that have been associated with this mention in the training set (e.g., "*National Football League*" or "*National Fertilizers Limited*"). Once the action is completed, the transition function will bring the focus to the next mention candidate (i.e., "*giants*"). The search reaches a terminal state when all the candidate mentions in the sentence have been linked.

$$\lambda x.\exists y.\, cast(\text{FamilyGuySeason1}, y) \wedge actor(y, x) \wedge character(y, \text{MegGriffin})$$

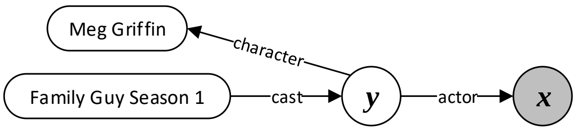


Figure 2: Semantic parses in λ-calculus (top) and query graph (bottom) of the question "*who played meg in season 1 of family guy?*"

3.3 Semantic parsing

Our third targeted task is semantic parsing (SP), which is a task of mapping a text utterance to a formal meaning representation. In this paper, we focus on a specific type of semantic parsing problem that maps a natural language question to a structured query, which is executed on a knowledge base to retrieve the answer to the original question.

Figure 2 shows the semantic parses of an example question "*who played meg in season 1 of family guy*", assuming the knowledge base is Freebase (Bollacker et al., 2008). An entity linking component plays an important role by mapping "*meg*" to `MegGriffin` and "*season 1 of family guy*" to `FamilyGuySeason1`. Predicates like `cast`, `actor` and `character` are also from the knowledge base that define the relationships between these entities and the answer. Together the semantic parse in λ-calculus is shown in the top of Figure 2. Equivalently, the semantic parse can be represented as a *query graph* (Figure 2 bottom), which is used in the STAGG system (Yih et al., 2015). The nodes are either grounded entities or variables, where x is the answer entity. The edges denote the relationship between two entities.

Regardless of the choice of the formal language, the process of constructing the semantic parse is typically formulated as a search problem. A state is essentially a partial or complete semantic parse, and an action is to extend the current semantic parse by adding a new relation or constraint.

Different from previous systems which treat entity linking as a static component, our search space consists of the search space of ***both*** entity linking and semantic parsing. That is, the search space is the union of the search space of entity linking described in Section 3.2 and the search space of the semantic parses, which we describe below. Integrating search spaces allows the model to use implicit signals to update both the semantic parsing and the entity linking systems. To the best of our knowledge, this is the first work that jointly learns the entity linking and semantic parsing systems.

Our search space is defined as follows. Starting from the initial state s_0, the model first explores the entity linking search space. Once the entity linking assignment are assigned (e.g. `FamilyGuySeason1` in Figure 2.) The second phase is then to determine the main relationship between the topic entity and the answer (e.g., the `cast-actor` chain between `FamilyGuySeason1` and x). Constraints (e.g., the `character` is `MegGriffin`) that describe the additional properties that the answer needs to have are added last. In this case, any state that is a legitimate semantic parse (consisting of one topic entity and one main relationship, as well as zero or more constraints) can lead to a terminal state.

4 Maximum Margin Reward Networks

In this section, we introduce the learning framework of `MMRN`, which includes two main components: *reward* and *max-margin loss*. The former is a mechanism for using implicit and explicit supervision signals in a unified way; the latter formally defines the learning objective.

4.1 Reward

The key insight of `MMRN` is that different types of supervision signals can be represented using the appropriate design of the reward function. A reward function is defined over a state–action pair $R(s, a)$, representing the *true* quality of taking action a in the state s. The reward for a path can be formally defined as: $R(\mathbf{h}) = \sum_{i=0}^{k} R(s_i, a_i)$. Intuitively, when the annotated action sequences (explicit supervision signals) exist, the model only needs to learn to imitate the annotated sequence. For instance, when learning NER in the fully supervised setting, the equivalent way of using Hamming distance is to define the reward $R(s, a)$ to be 1 if a matches the annotated sequence at the current state, and 0 otherwise.

In the setting where only implicit supervision is available, the reward function can still be designed to capture the signals. For instance, when only the question–answer pairs exist for learning the semantic parser, the reward can be defined by comparing the answers derived from a candidate parse and the labeled answers. More formally, assume that $s = \tau(s', a)$ is the state after applying action a to state s'. Let $Y(s)$ be the set of predicted answers generated from state s, and $Y(s) = \{\}$ when s is not a legitimate semantic parse. The reward function $R(s', a)$ can be defined by comparing $Y(s)$ and the labeled answers, A, to the input question. While a set similarity function like the Jaccard coefficient can be used as the reward function, we chose the F_1 score in this work as it was used as the evaluation metric in previous work (Berant et al., 2013). Figure 3 shows an example of this reward function.

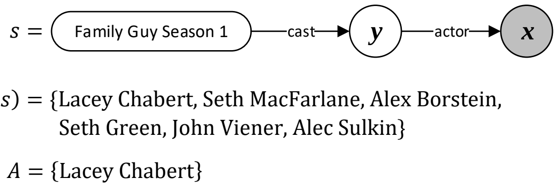

s =
Family Guy Season 1
cast
y
actor
x
Y(s) = {Lacey Chabert, Seth MacFarlane, Alex Borstein, Seth Green, John Viener, Alec Sulkin}
A = {Lacey Chabert}

Figure 3: For the question "*who played meg in season 1 of family guy?*", the candidate semantic parse s lists all the actors in "Family Guy Season 1" ($Y(s)$). By comparing $Y(s)$ to the answer set A, the precision is $\frac{1}{6}$ and the recall is 1. Therefore, the F_1 score used for the reward is $\frac{2}{7}$.

4.2 Max-Margin Loss & Learning Algorithm

The `MMRN` learning algorithm can be viewed as an extension of M^{3}N (Taskar et al., 2004) and Structured SVM (Joachims et al., 2009; Yu and Joachims, 2009). The learning algorithm takes three steps, where the first two involve two different search procedures. The final step is to update the models with respect to the inference results.

Finding the best path The first search step is to find the *best* path $\mathbf{h}^*$ by solving the following optimization problem:

$$\mathbf{h}^* = \arg\max_{\mathbf{h} \in \mathcal{E}(x)} R(\mathbf{h}; y) + \epsilon f_\theta(\mathbf{h}). \qquad (1)$$

The first term defines the path that has the highest reward. Because it is possible that several paths share the same reward, the second term leverages the current model and serves as the tie-breaker, where ϵ is a hyper-parameter that is set to a small positive number in our experiments.

When explicit supervision is available, solving Eq. (1) is trivial – the search simply returns the annotated sequence. In the case of implicit supervision, where true rewards are only revealed for complete action sequences, the search problem becomes difficult as the rewards of early state–action pairs are zeros. In this situation, the search algorithm uses the model score f_θ to guide the search. One possible design is to use beam search for the optimization problem, where the search procedure follows the current model in the early stage (given that $R(\mathbf{h}) = 0$). After generating several complete action sequences, the true reward function is then used to find $\mathbf{h}^*$. The tie-breaker also picks the best sequence when there are multiple sequences that lead to the same reward. Note that $\mathbf{h}^*$ can change between iterations because of the tie-breaker.

Finding the most violated path Once $\mathbf{h}^*$ is found, it is used as our reference path. We would like to update the model so that the scoring function f_θ will behave similarly to the reward R. More formally, we aim to update the model parameters θ to satisfy the following constraint.

$$f_\theta(\mathbf{h}^*) - f_\theta(\mathbf{h}) \geq R(\mathbf{h}^*) - R(\mathbf{h}), \forall \mathbf{h}.$$

The constraint implies that the "best" action sequence should rank higher than any other sequence by a margin computed from rewards as $R(\mathbf{h}^*) - R(\mathbf{h})$. The degree of violation of this constraint, with respect to $\mathbf{h}$, is thus $(R(\mathbf{h}^*) - R(\mathbf{h})) - (f_\theta(\mathbf{h}^*) - f_\theta(\mathbf{h})) = f_\theta(\mathbf{h}) - R(\mathbf{h}) - f_\theta(\mathbf{h}^*) + R(\mathbf{h}^*)$. The max-margin loss is defined accordingly:

$$\mathcal{L}(\mathbf{h}, \mathbf{h}^*) = \max(f_\theta(\mathbf{h}) - R(\mathbf{h}) - f_\theta(\mathbf{h}^*) + R(\mathbf{h}^*), 0)$$

$\mathcal{L}(\mathbf{h}, \mathbf{h}^*)$ is our optimization goal, where we want to update the model by fixing the biggest violation. Note that the associated constraint is only violated when $\mathcal{L}(\mathbf{h}, \mathbf{h}^*)$ is positive. To find the path $\mathbf{h}$ in this step that maximizes the violation is equivalent to maximizing $f_\theta(\mathbf{h}) - R(\mathbf{h})$, given that the rest of the terms are constant with respect to $\mathbf{h}$.

When there exist only explicit supervision signals, our objective function reduces to the one for optimizing structured SVM without regularization. For implicit signals, we find $\mathbf{h}^*$ approximately before we optimize the margin loss. In this case, the search is not exact as the reward signals are delayed. Nevertheless, we found the margin loss worked well empirically, as it kept decreasing in general until being stable.

Algorithm 1 summarizes the learning procedure of MMRN. Search is used in both Line 2 and 3. In Line 4, the algorithm performs a gradient update to modify all the model parameters.

Algorithm 1 Maximum Margin Reward Networks

```
1: for a random labeled data $(x, y)$ do
2:     $\mathbf{h}^* \leftarrow \arg\max_{\mathbf{h} \in \mathcal{E}(x)} R(\mathbf{h}; y) + \epsilon f_\theta(\mathbf{h})$
3:     $\hat{\mathbf{h}} \leftarrow \arg\max_{\mathbf{h} \in \mathcal{E}(x)} f_\theta(\mathbf{h}) - R(\mathbf{h}; y)$
4:     update $\theta$ by minimizing $\mathcal{L}(\hat{\mathbf{h}}, \mathbf{h}^*)$
5: end for
```

4.3 Practical Considerations

Although the learning algorithm of MMRN is simple and general, the quality of the learned model is dictated by the effectiveness of the search procedure. Increasing the beam size generally helps improve the model, but also slows down the training, and has a limited effect when dealing with a large search space. Domain-specific heuristics for pruning search space should thus be used when available. For instance, in the task of semantic parsing, when the reward of a legitimate semantic parse is 0, it implies that none of the derived answers is included in the labeled set of answers. When all the possible follow-up actions can only make the semantic parse stricter (e.g., adding constraints), and result in a subset of the current derived answers, it is clear that the rewards of all these new states are 0 as well. Paths from this state can thus be pruned.

Another strategy for improving search quality is to use *approximated* reward in the early stage of search. Very often the true rewards at this stage are 0, and are not useful to guide the search to find the best path. The approximated reward function can be thought of as estimating whether there exists a high-reward state that is reachable from the current state. The effectiveness of this strategy has been demonstrated successfully by several recent efforts (Mnih et al., 2013; Krishnamurthy et al., 2015; Silver et al., 2016; Narasimhan et al., 2016).

5 Neural Architectures

While the learning algorithm of MMRN described in Sec. 4 is general, the exact model design is task-dependent. In this section, we describe in detail the neural network architectures of the three targeted tasks, *named entity recognition*, *entity linking* and *semantic parsing*.

5.1 Named Entity Recognition

Recall that NER is formulated as a sequence labeling problem, and each action is to label a word with a tag using the BIOES encoding (cf. Sec. 3.1).

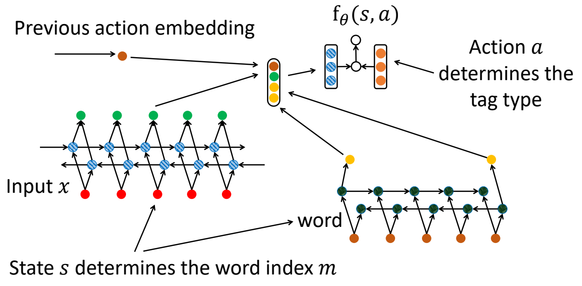

Previous action embedding
$f_\theta(s,a)$
Action a determines the tag type
Input x
word
State s determines the word index m

Figure 4: The action scoring model for NER.

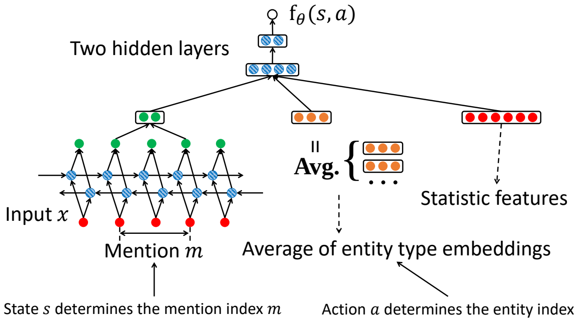

$f_\theta(s,a)$
Two hidden layers
=
Avg.
Statistic features
Input x
Mention m
Average of entity type embeddings
State s determines the mention index m
Action a determines the entity index

Figure 5: The action scoring model for EL.

The model of the action scoring function $f_\theta(s,a)$ is depicted in Figure 4, which is basically the dot product of the *action embedding* and *state embedding*. The action embedding is initialized randomly for each action, but can be fine-tuned during training (i.e. back-propagate the error through the network and update the word/entity type embeddings). The state embedding is the concatenation of bi-LSTM word embeddings of the current word, the character-based word embeddings, and the embedding of the previous action. We also include the orthographic embeddings proposed by Limsopatham and Collier (2016).

5.2 Entity Linking

An action in entity linking is to determine whether a mention should be linked to a particular entity (cf. Sec. 3.2). As shown in Figure 5, we design the scoring function as a feed-forward neural network that takes as input three different input vectors: (1) surface features from hand-crafted mention-entity statistics that are similar to the ones used in (Yang and Chang, 2015); (2) mention context embeddings from a bidirectional LSTM module; (3) entity embeddings constructed from entity type embeddings. All these embeddings, except the feature vectors, are fine-tuned during training.

Some unique properties of our entity linking model are worth noticing. First, we add mention context embeddings from a bidirectional LSTM module as additional input. While using LSTMs is a common practice for sequence labeling, it is not usually used for short-text entity linking. For each mention, we only extract the output from the bi-LSTM module at the start and end tokens of the mention, and concatenate them as the mention context embeddings. Second, we construct entity embeddings using the average of its Freebase (Bollacker et al., 2008) type embeddings[2], initialized using pre-trained embeddings. Adding these two types of embeddings has shown to improve the performance in our experiments.

5.3 Semantic Parsing

Our semantic parsing model follows the STAGG system (Yih et al., 2015), which uses a stagewise search procedure to expand the candidate semantic parses gradually (cf. Sec. 3.3). Compared to the original system, we make two notable changes. First, we use a two-layer feed-forward neural network to replace the original linear ranker that scores the candidate semantic parses. Second, instead of using a separately trained entity linking system, we incorporate our entity linking networks described in Sec. 5.2 as part of the semantic parsing model. The training process will thus fine tune the entity linking component to improve the semantic parsing system.

6 Experiments

It is important to have a general machine learning model working for both implicit and explicit supervision signals. We valid our learning framework when the explicit supervision signals are presented, as well as demonstrate the support of the scenario where supervision signals are mixed.

Specifically, in this section, we report the experimental results of `MMRN` on named entity recognition and entity linking, both using explicit supervision, and on semantic parsing, using implicit supervision. In all our experiments, we tuned hyperparameters on the development set (each task respectively), and then re-trained the models on the combination of the training and development set.

6.1 Named entity recognition

We use the CoNLL-2003 shared task data for the NER experiments, where the standard evaluation

[2] We use only the 358 most frequent Freebase entity types.

System	F_1
Collobert et al. (2011)	89.59
Huang et al. (2015)	90.10
Chiu and Nichols (2015)	90.77
Ratinov and Roth (2009)	90.88
Lample et al. (2016)	90.94
Ma and Hovy (2016)	91.21
MMRN-NER Beam = 5	90.03
MMRN-NER Beam = 20	**91.39**

Table 1: **Explicit Supervision: Named Entity Recognition**. Our MMRN with beam size 20 outperforms current best systems, which are based on neural networks.

	NEEL-Test	TACL
	F_1	F_1
S-MART	77.7	63.6
NTEL	77.9	**68.1**
MMRN-EL	**78.5**	67.5
MMRN-EL - Entity	77.4	66.5
MMRN-EL - LSTM	76.6	66.0

Table 2: **Explicit Supervision: Entity Linking.** Our system trained with MMRN is comparable to the state-of-art NTEL system.

metric is the F_1 score. The pre-trained word embeddings are 100-dimension GloVe vectors trained on 6 billion tokens (Pennington et al., 2014)[3]. The search procedure is conducted using beam search, and the reward function is simply the number of correct tag assignments to the words.

The results are shown in Table 1, compared with recently proposed systems based on neural models. When the beam size is set to 20, MMRN achieves 91.4, which is the best published result so far (without using any gazetteers). Notice that when beam size is 5, the performance drops to 90.03. This demonstrates the importance of search quality when applying MMRN.

6.2 Entity linking

For entity linking, we adopt two publicly available datasets for tweet entity linking: NEEL (Cano et al., 2014)[4] and TACL (Guo et al., 2013; Fang and Chang, 2014; Yang and Chang, 2015; Yang et al., 2016). We follow prior works (Guo et al., 2013; Yang and Chang, 2015) and perform the standard evaluation for an end-to-end entity linking system by computing precision, recall, and F_1 scores, according to the entity references and the system output. An output entity is considered correct if it matches the gold entity and the mention boundary overlaps with the gold mention boundary. Interested readers can refer to (Carmel et al., 2014) for more detail.

We initialize the word embeddings from pre-trained GloVe vectors trained on the twitter corpus, and type embeddings from the pre-trained skip-gram model (Mikolov et al., 2013)[5]. Sizes of both word embeddings are set to 200. Inference is done using a dynamic programming algorithm.

Results of entity linking experiments are presented in Table 2, which are compared with those of S-MART (Yang and Chang, 2015)[6] and NTEL (Yang et al., 2016)[7], two state-of-the-art entity linking systems for short texts. Our MMRN-EL is comparable to the best system. We also conducted two ablation studies by removing the entity type vectors (MMRN-EL - Entity), and by removing the LSTM vectors (MMRN-EL - LSTM). Both show significant performance drops, which validates the importance of these two additional input vectors.

6.3 Semantic parsing

For semantic parsing, we use the dataset WebQSP[8] (Yih et al., 2016) in our experiments. This dataset is a clean and enhanced version of the widely used WebQuestions dataset (Berant et al., 2013), which consists of pairs of questions and answers found in Freebase. Compared to WebQuestions, WebQSP excludes questions with ambiguous intent, and provides verified answers and full semantic parses to the remaining 4,737 questions.

We follow the implicit supervision setting in (Yih et al., 2016), using $3,098$ question–answer pairs for training, and $1,639$ for testing. A subset of 620 pairs from the training set is used for hyper-parameter tuning. Because there can be multiple answers to a question, the quality of a semantic parser is measured using the averaged F_1 score of the predicted answers.

[3] Available at http://nlp.stanford.edu/projects/glove/

[4] NEEL dataset was originally created for an entity linking competition: `http://microposts2016.seas.upenn.edu/challenge.html`

[5] Available at https://code.google.com/archive/p/word2vec/

[6] The winning system of the NEEL challenge.

[7] To have a fair comparison, we compare to the results of NTEL which do not use pretrained user embedding.

[8] Available at http://aka.ms/WebQSP

We experiment with two configurations of incorporating the entity linking component. MMRN-Pipeline trains an MMRN-EL model using the entity linking labels in WebQSP separately. Given a question, the entities in it are first predicted, and used as input to the semantic parsing system. In contrast, MMRN-Joint incorporates the MMRN-EL model in the whole framework. During this joint training process, 15 entity link results are sampled according to the current MMRN-EL model, and passed to the downstream networks. In both cases, we use the previous entity linking model trained on the NEEL dataset to initialize the parameters. As discussed in Sec. 4.1, in this implicit supervision setting, we directly set the (delayed) reward function to be the F_1 score, which can be obtained by comparing the annotated answers with predicted answers.

Table 3 summarizes the results of the MMRN-based semantic parsing systems and other strong baselines. The SP column reports the averaged F_1 scores. Compared to the pipeline approach (MMRN-Pipeline), the joint learning framework (MMRN-Joint) improves significantly, reaching 68.1% F_1. To compare different learning methods, we also apply REINFORCE (Williams, 1992), a popular policy gradient algorithm, to train our joint model using the same setting and reward function.[9] MMRN-Joint outperforms REINFORCE and its variant, REINFORECE+, which re-normalizes the probabilities of the sampled candidate sequences. Its result is also better than the state-of-the-art STAGG system. Note that we use the same architectures and initialization procedures for MMRN-Pipeline/Joint and REINFORCE/REINFORCE+. Therefore, the superior performance of MMRN-Joint shows that the joint learning plays a crucial role in addition to the choices of architecture. Comparing to STAGG, note that Yih et al. (2016) did not jointly train the entity linker and semantic parser together, but they did improve the results by taking the top 10 predictions of their entity linking system for re-ranking parses. Our algorithm further allows to update the entity linker with the labels for semantic parsing and shows superior performance.

Our joint model also improves the entity linking prediction on the questions in WebQSP using the implicit signals (the EL columns in Table 3). The F_1 score of MMRN-Joint on entity linking is 2.4 points higher than the baseline MMRN-Pipeline. Note that the entity linking results of MMRN-Pipeline (line 1) are exactly the results of the entity linking component MMRN-EL. The result is also better than REINFORCE, and comparable to REINFORCE+.

	SP	EL		
	Avg. F_1	P	R	F_1
MMRN-Pipeline	62.5	85.6	77.5	81.3
MMRN-Joint	**68.1**	89.3	78.9	**83.7**
REINFORCE	62.9	87.5	76.6	81.7
REINFORCE+	66.7	91.1	76.9	83.4
STAGG	66.8	–	–	–

Table 3: **Implicit Supervision: Semantic Parsing**. By updating the entity linking and semantic parsing models jointly, MMRN-Joint improves over MMRN-Pipeline by 5 points in F_1 and outperforms REINFORCE+ (SP). It also improves the entity linking result on the WebQSP questions (EL).

Recently Liang et al. (2016) proposed Neural Symbolic Machine (NSM) and reported the best result of 69.0 F1 score on the WebQSP dataset using the weak supervision settings.[10] The NSM architecture for semantic parsing is significantly different from the architecture used in (Yih et al., 2016) and the one used in this paper. In contrast, MMRN is a general learning framework that allows joint training on existing models (i.e. entity linking and semantic parsing modules). This allows MMRN to use the labels of semantic parsing task as implicit supervision signals for the entity linking module. It would be interesting to apply MMRN on the newly proposed architectures as well.

7 Discussion

We discuss several issues that are highly related to MMRN in this section.

Learning to Search There are two main differences between MMRN and search-based algorithms, such as SEARN (Daumé et al., 2009) and DAGGER (Ross et al., 2011). First, both SEARN and DAGGER focus on imitation learning, assuming explicit supervision signals exist. They use a two-step model learning approach:

[9] The REINFORCE algorithm uses warm initialization—the entity linking parameters are initialized using the model trained on the NEEL dataset.

[10] The paper is published after the submission of this paper.

(1) create cost-sensitive examples by listing state–action pairs and their corresponding (estimated) losses; (2) apply cost-aware training algorithms. In contrast, MMRN directly updates the parameters using back-propagation based on search results of each example. Second, SEARN mixes the optimal and current policies during learning, while MMRN performs search twice and simply pushes the current policy towards the optimal one. Recently, Chang et al. (2015) extend this line of work and discuss different roll-in and roll-out strategies during training for structured contextual bandit settings. As MMRN uses two search procedures, there is no need to mix different search policies.

Reinforcement Learning In many reinforcement learning scenarios, the search space is not fully controllable by the agent. For example, a chess playing agent cannot control the move made by its opponent, and has to commit a single move and wait for the opponent. Note that the agent can still think ahead and build a search tree, but only one move can be made in the end. In contrast, in scenarios like semantic parsing, the whole search space is controlled by the agent itself. Therefore, from the initial state, we can explore several search paths and get their real rewards. This may explain why MMRN can be more efficient than REINFORCE, as MMRN can use the reward signals of multiple paths more effectively. In addition, MMRN is not a probabilistic model, so it does not need to handle normalization issues, which often causes large variance in estimating the gradient direction when optimizing the expected reward.

Semantic Parsing MMRN can be applied for many semantic parsing tasks. One key step is to design the right approximated reward for a given task to guide the beam search to nd the reference parses in MMRN, given that the actual reward is often very sparse. In our companion paper, (Iyyer et al., 2017), we used a simple form of approximated reward to get feedback as early as possible during search. In other words, the semantic parse will be executed as soon as the parse is executable (even if the parse is still not completed) *during search*. The execution results will be used to calculate the Jaccard coefficient with respect to the labeled answers as the approximated rewards. The use of approximated reward has been proven to be effective in (Iyyer et al., 2017).

An important research direction for semantic parsing is to reduce the supervision cost. In (Yih et al., 2016), the authors demonstrated that labeling semantic parses is possible and often more effective with a sophisticated labeling interface. However, collecting answers may still be easier or faster for certain problems or annotators. This suggests that we could allow the annotators to choose to label semantic parses or answers in order to minimize the supervision cost. MMRN would be an ideal learning algorithm for this scenario.

8 Conclusion

This paper proposes Maximum Margin Reward Networks, a structured learning framework that can learn from both explicit and implicit supervision signals. In the future, we plan to apply Maximum Margin Reward Networks on other structured learning tasks. Improving MMRN for dealing with large search space is an important future direction as well.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. The first author is partly sponsored by DARPA under agreement number FA8750-13-2-0008. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *EMNLP*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *ICDM*.

Amparo E Cano, Giuseppe Rizzo, Andrea Varga, Matthew Rowe, Milan Stankovic, and Aba-Sah Dadzie. 2014. Making sense of microposts:(# microposts2014) named entity extraction & linking challenge. In *CEUR Workshop*.

David Carmel, Ming-Wei Chang, Evgeniy Gabrilovich, Bo-June Paul Hsu, and Kuansan Wang. 2014. ERD'14: entity recognition and disambiguation challenge. In *ACM SIGIR Forum*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *ICML*.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional LSTM-CNNs. *arXiv preprint arXiv:1511.08308*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *CoNLL*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *SIGDAT*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *ACL*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*.

Hal Daumé and Daniel Marcu. 2005. Learning as search optimization: approximate large margin methods for structured prediction. In *ICML*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *ACL*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *ACL*.

Yuan Fang and Ming-Wei Chang. 2014. Entity linking on microblogs with spatial and temporal signals. *TACL*.

Yuhang Guo, Bing Qin, Ting Liu, and Sheng Li. 2013. Microblog entity linking by leveraging extra posts. In *EMNLP*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

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

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *ACL*.

Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. 2009. Cutting-plane training of structural svms. *Machine Learning*.

Akshay Krishnamurthy, CMU EDU, Hal Daumé III, and UMD EDU. 2015. Learning to search better than your teacher. *arXiv preprint arXiv:1502.02206*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

Nut Limsopatham and Nigel Collier. 2016. Bidirectional LSTM for named entity recognition in twitter messages. In *WNUT*.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNS-CRF. *arXiv preprint arXiv:1603.01354*.

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

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *EMNLP*.

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

Ariadna Quattoni, Sybor Wang, Louis-Philippe Morency, Morency Collins, and Trevor Darrell. 2007. Hidden conditional random fields. *PAMI*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*.

Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004. Max-margin markov networks. In *NIPS*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *NAACL*.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 2016. Learning from explicit and implicit supervision jointly for algebra word problems. In *EMNLP*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *NIPS*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*.

Yi Yang and Ming-Wei Chang. 2015. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. In *ACL*.

Yi Yang, Ming-Wei Chang, and Jacob Eisenstein. 2016. Toward socially-infused information extraction: Embedding authors, mentions, and entities. In *EMNLP*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL*.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *ACL*.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *ICML*.

The Impact of Modeling Overall Argumentation with Tree Kernels

Henning Wachsmuth
Bauhaus-Universität Weimar
Faculty of Media, Webis Group
henning.wachsmuth@uni-weimar.de

Giovanni Da San Martino
Qatar Computing Research Institute
Arabic Language Technologies
gmartino@hbku.edu.qa

Dora Kiesel
Bauhaus-Universität Weimar
Faculty of Media, VR Group
dora.kiesel@uni-weimar.de

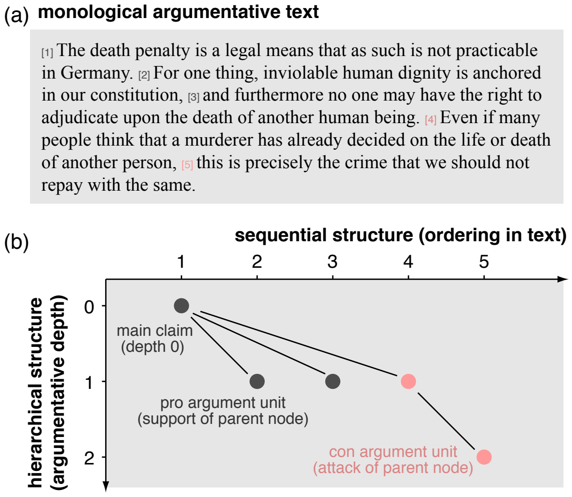
Benno Stein
Bauhaus-Universität Weimar
Faculty of Media, Webis Group
benno.stein@uni-weimar.de

Abstract

Several approaches have been proposed to model either the explicit sequential structure of an argumentative text or its implicit hierarchical structure. So far, the adequacy of these models of overall argumentation remains unclear. This paper asks what type of structure is actually important to tackle downstream tasks in computational argumentation. We analyze patterns in the overall argumentation of texts from three corpora. Then, we adapt the idea of positional tree kernels in order to capture sequential and hierarchical argumentative structure together for the first time. In systematic experiments for three text classification tasks, we find strong evidence for the impact of both types of structure. Our results suggest that either of them is necessary while their combination may be beneficial.

1 Introduction

Argumentation theory has established a number of major argument models focusing on different aspects, such as the roles of an argument's units (Toulmin, 1958), the inference scheme of an argument (Walton et al., 2008), or the support and attack relations between arguments (Freeman, 2011). The common ground of these models is that they conceptualize an argument as a conclusion (in terms of a claim) inferred from a set of pro and con premises (reasons), which in turn may be the conclusions of other arguments. For the overall argumentation of a monological argumentative text such as the one in Figure 1(a), this results in an implicit hierarchical structure with the text's main claim at the lowest depth. In addition, the text has an explicit linguistic structure that can be seen as a regulated sequence of speech acts (van Eemeren and Grootendorst, 2004).


(a) **monological argumentative text**

[1] The death penalty is a legal means that as such is not practicable in Germany. [2] For one thing, inviolable human dignity is anchored in our constitution, [3] and furthermore no one may have the right to adjudicate upon the death of another human being. [4] Even if many people think that a murderer has already decided on the life or death of another person, [5] this is precisely the crime that we should not repay with the same.

(b)
sequential structure (ordering in text)
1
2
3
4
5
hierarchical structure (argumentative depth)
0
1
2
main claim (depth 0)
pro argument unit (support of parent node)
con argument unit (attack of parent node)

Figure 1: (a) Example text with five argument units, taken from the *Arg-Microtexts* corpus introduced in Section 3. (b) Graph visualization of the sequential and hierarchical overall argumentation of the text.

Figure 1(b) illustrates the interplay of the two types of overall structure in form of a tree-like graph.

Natural language processing research has largely adopted the outlined hierarchical models for mining arguments from text (Stab and Gurevych, 2014; Habernal and Gurevych, 2015; Peldszus and Stede, 2016). However, the adequacy of the resulting overall structure for downstream analysis tasks of computational argumentation has rarely been evaluated (see Section 2 for details). In fact, a computational approach that can capture patterns in hierarchical overall argumentation is missing so far. Even more, our previous work indicates that a sequential model of overall structure is preferable for analysis tasks such as stance classification or quality assessment (Wachsmuth and Stein, 2017).

In this paper, we ask and investigate what model of (monological) overall argumentation is important to tackle argumentation-related analysis tasks. To this end, we consider three corpora with fully

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2379–2389
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

annotated argument structure (Section 3). Each corpus allows studying one text classification task, two of which we hypothesize to benefit from modeling argumentation (myside bias, stance), the third not (genre). An empirical analysis of the corpora reveals class-specific patterns of how people argue (Section 4). In order to combine the explicit sequential and the implicit hierarchical structure of an argumentative text for the first time, we then adapt the approach of *route kernels* (Aiolli et al., 2009), modeling overall argumentation in form of a positional tree (Section 5).

On this basis, we design an experiment to evaluate the impact of the different types of argumentative structure (Section 6). In particular, we decompose our approach into four complementary modeling steps, both for a general model of overall argumentation and for the specific models of the given corpora. Using the structure annotated in the corpora, we systematically compare the effectiveness of all eight resulting models and two standard baselines in the three classification tasks.

Our results provide strong evidence that both sequential and hierarchical structure are important. As indicated by related work, sequential structure nearly competes with hierarchical structure, at least based on the specific argument models. Even more impressively, modeling hierarchical structure practically solves the task of identifying argumentation with myside bias, achieving an outstanding accuracy of 97.1%. For stance classification, the combination captured by positional trees works best. In contrast, all types of structure fail in distinguishing genres, suggesting that they indeed capture properties of argumentation. We conclude that the impact of modeling overall structure on downstream analysis tasks is high, while the required type may vary.

Contributions To summarize, the main contributions of this paper are the following:

1. Empirical insights into how people structure argumentative texts in overall terms.
2. The first approach to model and analyze the sequential and hierarchical overall structure of argumentative texts in combination.
3. Evidence that modeling overall structure impacts argumentation-related analysis tasks.

2 Related Work

The study of overall argumentation traces back to Aristotle (2007) who outlined the impact of the sequential arrangement of the different parts of a speech. Conceptually, theory agrees that a monological argumentative text has an implicit tree-like hierarchical structure: Toulmin (1958) defines an argument as a claim supported by data that is reasoned by a warrant, which in turn may have a backing. In addition, a rebuttal may be given showing exceptions to the claim. The role of support and attack relations is investigated by Freeman (2011) who models dialectical arguments that discuss both a proponent's and an opponent's view on the main claim argued for. Walton et al. (2008) put the focus on the inference scheme that describes how an argument's conclusion follows from its premises, which may themselves be conclusions of arguments.

In natural language processing, argumentation research deals with the mining of argument units and their relations from text (Mochales and Moens, 2011). Several corpora with annotated argument structure have been published in the last years. Many of the corpora adapt the hierarchical models from theory (Reed and Rowe, 2004; Habernal and Gurevych, 2015; Peldszus and Stede, 2016) or propose comparable models (Stab and Gurevych, 2014). Since we target monological overall argumentation, we use those that capture the complete structure of texts, as detailed in Section 3. Corpora focusing on dialogical argumentation (Walker et al., 2012), topic-related arguments (Rinott et al., 2015), or sequential structure (Wachsmuth et al., 2014b; Al Khatib et al., 2016) are out of scope.

We do not mine the structure of argumentative texts, but we exploit the previously mined structure to tackle downstream tasks of computational argumentation, namely, to classify the myside bias and stance of texts. For myside bias, Stab and Gurevych (2016) use features derived from discourse structure, whereas Faulkner (2014) and Sobhani et al. (2015) model arguments to classify stance. Ong et al. (2014) and we ourselves (Wachsmuth et al., 2016) do similar to assess the quality of persuasive essays, and Beigman Klebanov et al. (2016) examine how an essay's content and structure influence quality. Other works predict the outcome of legal cases based on the applied types of reasoning (Brüninghaus and Ashley, 2003) or analyze inference schemes for given arguments (Feng and Hirst, 2011). In contrast to the local structure of single arguments employed by all these approaches, we study the impact of the global overall structure of complete monological argumentative texts.

In (Wachsmuth et al., 2017), we point out that the argumentation quality of a text is affected by interactions of its content at different levels of granularity, from single argument units over arguments to overall argumentation. Stede (2016) explores how different depths of overall argumentation can be identified, observing differences across genres. Unlike in our experiments, however, the genres considered there reflect diverging types of argumentation. Related to argumentation, Feng et al. (2014) build upon rhetorical structure theory (Mann and Thompson, 1988) to assess the coherence of texts, while Persing et al. (2010) score the organization of persuasive essays based on sequences of sentence and paragraph functions.

We introduced the first explicit computational model of overall argumentation in (Wachsmuth et al., 2014a). There, we compared the flow of local sentiment in a review to a set of learned flow patterns in order to classify global sentiment. Recently, we generalized the model in order to make flows applicable to any type of information relevant for argumentation-related analysis tasks (Wachsmuth and Stein, 2017). However, flows capture only *sequential* structure, whereas here we also model the *hierarchical* structure of overall argumentation. To this end, we make use of kernel methods.

Kernel methods are a popular approach for learning on structured data, with several applications in natural language processing (Moschitti, 2006b) including argument mining (Rooney et al., 2012). They employ a similarity function defined between any two input objects that are represented in a task-specific implicit feature space. The evaluation of such a kernel function relies on the common features of the input objects (Cristianini and Shawe-Taylor, 2000). The kernel function encodes knowledge of the task in the form of these features.

Several kernel functions have been defined for structured data. To assess the impact of sequential argumentation, we refer to the function of Mooney and Bunescu (2006), which computes common subsequences of two input sequences. For trees, most existing approaches count common subtrees of a certain type (Collins and Duffy, 2001; Moschitti, 2006a; Kimura and Kashima, 2012), but they do not take the ordering of the nodes in the subtrees into account. In contrast, Aiolli et al. (2009) developed a kernel that combines the content of substructures with the relative positions inside trees, called the *route kernel*. Similarly, the tree kernel of Daumé III and Marcu (2004) includes positional information for document compression. For overall argumentation, we decided to use the route kernel in Section 5, as it makes the modeling of the sequential positions of an argument unit in a text straightforward. This allows us to capture both the sequential and the hierarchical structure at the same time. To our knowledge, no work has done this before.[1]

Neural networks denote an alternative for learning on structured data. They become particularly effective when few prior knowledge about what is important to address a task at hand is available, because they can learn any feature representation in principle (Goodfellow et al., 2016). Due to this flexibility, however, large amounts of data are required for training an effective model, making neural networks inadequate for the small datasets that allow studying overall argumentation.

3 Tasks and Datasets

We seek to study the impact of modeling overall argumentation on downstream tasks without the noise from argument mining errors. To this end, we rely on three ground-truth argument corpora. Each corpus is suitable for evaluating one text classification task and comes with a specific model of overall argumentation, as detailed in the following.

Myside Bias on AAE-v2 The *Argument Annotated Essays* corpus was originally been presented by Stab and Gurevych (2014). We use version 2 of the corpus (available on the website of the authors), which consists of 402 persuasive student essays. In each essay, all *main claims*, *claims*, and *premises* are annotated as such. Each claim has a *pro* or *con* stance towards each instance of the main claim, whereas each premise *supports* or *attacks* a claim or another premise. Thereby, argumentation is modeled as one tree structure for each major claim.

Stab and Gurevych (2016) added a *myside bias* class to each essay, reflecting whether its argumentation is one-sided considering only arguments for the own stance (251 cases) or not (151 cases).

Stance on Arg-Microtexts The *Arg-Microtexts* corpus of Peldszus and Stede (2016) contains 112 short argumentative texts. They cover 18 different controversial topics and are annotated according to Freeman (2011): Each argument unit takes the role of the *proponent* or *opponent* of a main claim. What

[1] While extensions of the route kernel idea have been published later on (Aiolli et al., 2011, 2015), we resort to the original version in this paper for simplicity.

	AAE-v2	Arg-Microtexts	Web Discourse
Argument units	6089	576	1149
Avg. units/text	15.1	5.1	3.4
Min. units/text	7	3	0
Max. units/text	28	10	16
Arguments	5687	443	560
Avg. depth	2.8	2.0	0.6
Min. depth	2	1	0
Max. depth	5	4	1
Texts	402	112	340

Table 1: Statistics of the argument units and arguments in the three corpora analyzed in this paper.

the main claim is follows from a tree-like overall structure emerging from four types of relations: *normal* or *example* support from one unit to another, a *rebuttal* of units by other units, and *undercutters* where a relation is attacked by another unit.

For 88 texts, the *stance* towards a specified topic is labeled as *pro* (46) or *con* (42). We use these labels for classification, but we do not access the topic. This way, stance needs to be identified only based on a text itself — a very challenging task.[2]

Genre on Web Discourse Finally, we consider the *Argument Annotated User-Generated Web Discourse* corpus of Habernal and Gurevych (2015). There, 340 texts are annotated according to a modified version of the specific model of Toulmin (1958) where *claims* are supported by *premises* or attacked by *rebuttals*. *Rebuttals* in turn may be attacked by *refutations*. Besides, emotional units not participating in the actual arguments are marked as *pathos*. The support and attack relations build up the overall argumentation of a text.

The corpus composes argumentative texts of four *genres*, namely, 5 *articles*, 216 *comments* to articles, 46 *blog posts*, and 73 *forum posts*. The genre is specified in form of a label for each text. Due to the low number, we ignore the articles below.

To give an idea of the sequential and hierarchical overall structure in each corpus, Table 1 presents statistics of the argument units, the arguments (in terms of relations between two or more units), and the depth of the resulting argumentation.

While the size of the given corpora and the variety of tasks are limited, the only other available corpus with fully annotated argument structure that we are aware of is *AraucariaDB* (Reed and Rowe, 2004). No downstream task can be tackled on AraucariaDB besides inference scheme classification (Feng and Hirst, 2011). As all schemes compose a conclusion and a set of premises (without more specific roles), analyzing overall structure hardly makes sense, which is why we omit the corpus.

4 Insights into Overall Argumentation

Before we approach overall argumentation computationally, this section analyzes the three given corpora empirically to provide insights into how people argue in overall terms. For this, we unify the specific corpus models of overall argumentation outlined above in one general model.

4.1 A Unified View of Overall Argumentation

The texts in all corpora are segmented into argument units, partly with non-argumentative spans in between that we ignore here for lack of relevance. To capture the sequential ordering of the segmentation, we assign a global index to each unit.

As described in Section 3, the specific models of all three corpora in the end consider an argument as a composition of one unit serving as the conclusion with one or more units that support or attack the conclusion (the premises). This composition is defined through multiple relations from one premise to one conclusion each. There is one exception, namely, the undercutter relations in the Arg-Microtexts corpus have a relation as their target. To obtain a unified form in the general model, we modify the undercutters such that they target the premise of the undercutted relation.

In all corpora, a premise may be the conclusion of another argument, while no argument unit serves as a premise in multiple arguments. This leads to a tree structure for each main claim of the associated text. A main claim corresponds to a unit that is not a premise. In AAE-v2 and in Web Discourse, more than one such unit may exist per text.

Depending on the corpus, the distinction of support and attack is encoded through a specified relation type, a unit's stance, or both. We unify these alternatives by modeling the stance of each unit towards its parent in the associated tree. This stance can be derived in all corpora.[3] All other unit and relation types from the specific models are ignored, since there is no clear mapping between them.

[2] We do not include the topic, in order not to conflate the impact of modeling argumentation with the influence of the topic. The corpus is too small to analyze topic differences.

[3] Alternatively, the stance towards the main claim could be modeled. We decided against this alternative to avoid possibly wrong reinterpretations, e.g., it is unclear whether a unit that attacks its parent always supports a unit attacked by the parent.

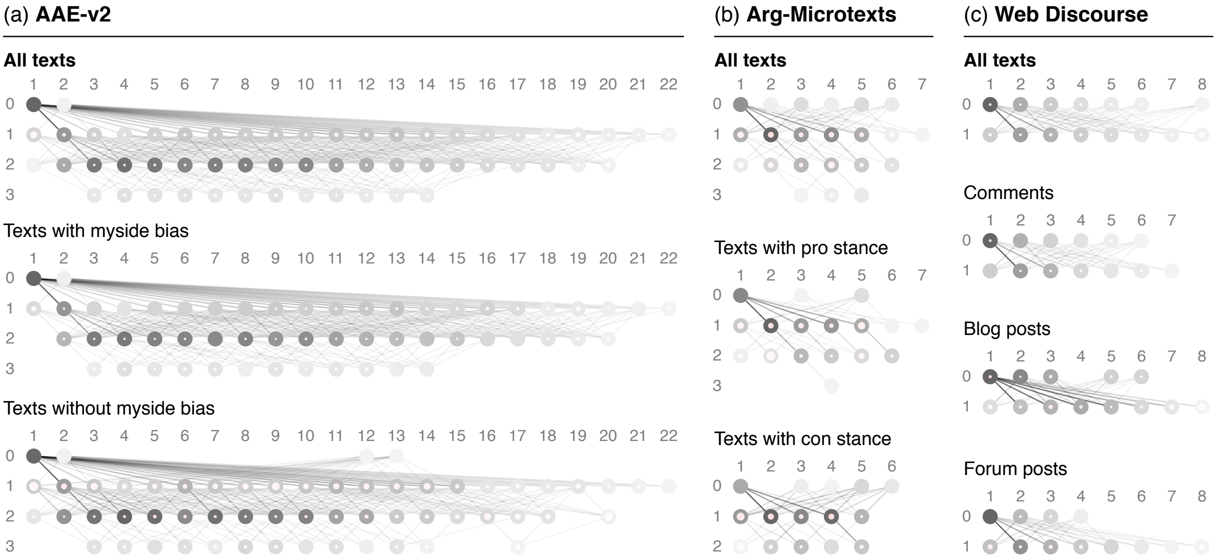


Figure 2: Visualization of the overall argumentation in the three considered corpora based on the introduced general model, averaged over all texts as well over all those texts that belong to a particular class. Left to right: Position in the text. Top to bottom: Depth in graph. Brightness: Inverse relative frequency of each position/depth combination in the corpus. Light red to gray: Proportion of argument units with con stance.

General Model As a result, we model the overall argumentation of an argumentative text as a forest of trees. Each node in a tree corresponds to an argument unit. It has an assigned stance (pro or con) as well as a global index that defines its position in the text. Each edge defines a relation from a premise (the child node) to a conclusion (the parent node). Each main claim defines the root of a tree.

Figure 1(b) has already illustrated an instance of the general model. The general model is slightly less expressive than the specific models. We evaluate in Section 6 to what extent this reduces its use for tackling argumentation-related analysis tasks. The advantage of the general model is that it allows a comparison of patterns of overall argumentation across corpora, as we do in the following.[4]

4.2 Visualization of Argumentation Patterns

Based on the general model, we empirically analyze class-specific patterns of overall argumentation on the three corpora. To this end, we compute one "average graph" for all texts in each complete corpus and one such graph for all texts with a particular class (e.g., for all "no myside bias" texts in case of AAE-v2). In an average graph, each node is labeled with the relative frequency of the associated combination of position and depth in all texts (edges accordingly). We align positions of different texts based on their start node, due to our observation that the first argument unit overproportionally often represents the main claim.[5] In addition the relative frequency, we determine the proportion of con to pro stance for each node.

As we aim to provide intuitive insights into how people argue in overall terms, we discuss the graphs in an informal visual way instead of listing exact numbers.[6] In the visualizations in Figure 2, brightness captures (inverse) frequency, so darker nodes represent more frequent argument units. The diameter of the inner light-red part of each node reflects its proportion of con stance. Nodes with a relative frequency below 0.3% and/or an absolute frequency below 3 are pruned, along with all their associated edges.

AAE-v2 Figure 2(a) stresses that most students state the main claim (depth 0, position 1) in a persuasive essay first. When the first argument unit is a premise of the main claim instead, it often attacks the main claim, as the large light-red proportion of the node at depth 1 and position 1 conveys. While, on average, texts with myside bias do not differ in length from those without, the latter show more con stance, especially at depth 1. Also, argumenta-

[4] Besides, although not in the focus here, we also assume stance to be easier to detect in practice than fine-grained roles.

[5] We also considered using the main claim as the fix point, but the resulting graphs would be much wider than the longest argumentation, which may be misleading.

[6] We provide files with the exact frequencies of all nodes and edges at: http://www.arguana.com/software.html

tion without myside bias shows more variance, as indicated, for instance, by the nodes at depth 0 and position 12 and 13 respectively. In contrast, clear patterns in the sequential ordering of pro and con stance are not recognizable in AAE-v2.

Arg-Microtexts According to the graphs in Figure 2(b), the position of the main claim varies in the microtexts. While the proportion of con stance seems rather similar between pro and con texts, our visualization reveals that their overall structure is "mirror-inverted" to a limited extent: Most pro texts start with the main claim (depth 0, position 1), discuss con stance later (red proportions increase to the right), and deepen the argumentation in a top-down fashion (most edges from top left to bottom right). Vice versa, con texts more often present the main claim later, attack it earlier, and seem to argue more bottom-up. This suggests that both sequential and hierarchical structure play a role here.

Web Discourse The web discourse texts, finally, comprise rather shallow argumentation across all genres. Slight structural differences can be seen, especially, the comments appear a little shorter and richer of pro stance on average. Besides, the blog posts have more con stance later. Still, the darker and thus more frequent nodes are at similar positions in all graphs. So, if at all, differences may be reflected in a sequential model of argumentation, which implicitly covers length. In terms of the hierarchical structure of the frequent nodes, the graphs of all genres are rather indistinguishable.

Altogether, the visualizations give first support for the impact of modeling overall argumentation. In particular, we hypothesize that hierarchical overall structure is decisive for myside bias, whereas a combination of sequential and hierarchical structure helps to distinguish pro-stance from con-stance texts. In contrast, we expect that the impact on classifying genres in the Web Discourse corpus is low.

5 Modeling Overall Argumentation

This section presents our kernel-based approaches for argumentation-related analysis tasks. They rely on a tree representation of overall argumentation.

5.1 Representation of Overall Argumentation

We model the overall structure of an argumentative text in form of a positional tree $T = (V, E)$ that, in principle, equals those exemplified in Figure 1 and analyzed above. Each node $v \in V$ represents an argument unit and each edge $e = (v_1, v_2) \in E$ a relation between two units. Technically, we therefor map the forest of trees representing a text (see Section 4) to a single tree by adding a "virtual" root node v_0 to V that is the parent of all tree roots.

In analysis tasks, we seek to compare sequential and hierarchical structures irrespective of the actual texts and the size of the associated trees. To this end, we represent labels and positions as follows:

Labels The tree kernel approaches in natural language processing discussed in Section 2 include text (usually words) in the leaf nodes. In contrast, we label each node $v \in V$ with the type of the associated argument unit only. Thereby, we almost fully abstract from the content of texts, which benefits the identification of common structures. In case of the general model, the only two labels are *pro* and *con*. In case of the specific models, we combine the role of a unit with the type of the relation the unit is the source of (if any). On Arg-Microtexts, for instance, this creates labels such as *opponent-support* or *opponent-undercutter*.

Positions As we adapt the route kernels of Aiolli et al. (2009) below, we follow their representation of sequential structure with one exception. In particular, the authors assigned an index to each edge that numbers the child nodes of each node ascending from 1. Thereby, they encoded the relative positions of sibling nodes *to each other*. To capture the ordering of argument units in a text from left to right, we also model positions as indices of the edges in E. Unlike Aiolli et al. (2009), however, we use indices decreasing from -1 in the left direction of the parent node and ascending from 1 to the right (derived from the nodes' global indices). While such a simple relabeling allows us to reuse their algorithm for computing kernels, it makes a decisive difference, namely, it encodes the relative positions of child nodes *to their parent*. This in turn implies the sequential structure of the whole tree.

Figure 3(a) exemplifies the tree representation for the argument unit types of the general model, omitting the virtual root v_0 for simplicity. Analogously, the types of the specific models of the three considered corpora could be used.

5.2 Kernel-based Modeling Approaches

Based on the tree representation, we now introduce four approaches for modeling overall argumentation. Figure 3(b) illustrates the kernel representations of each approach. As discussed in Section 2,

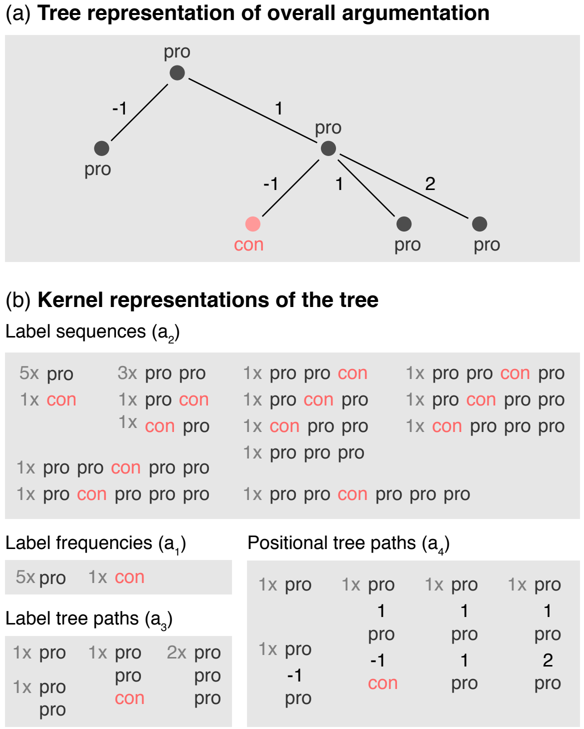


Figure 3: (a) Tree representation exemplified for the general model; node labels are unit types, edge indices relative positions of child nodes. (b) Kernel representations of the tree for all four approaches.

the associated kernel function compares the representations of the trees T, T' of any two texts.

Label Frequencies (a_1) Our simplest model of overall argumentation does not encode structure at all. Instead, it compares only the frequencies of each node label in T and T'. We represent the model with a linear kernel, which in the end corresponds to a standard feature representation.

Label Sequences (a_2) To encode sequential overall structure, we refer to the kernel of Mooney and Bunescu (2006), representing the sequential ordering of node labels in a tree by all contiguous subsequences. The similarity of two trees T and T' follows from the proportion of common subsequences, but longer subsequences are penalized by a decay factor. This approach can be seen as an imitation of our flow model (Wachsmuth and Stein, 2017).[7]

Label Tree Paths (a_3) We capture hierarchical overall structure adapting the non-positional part of the route kernel of Aiolli et al. (2009), *label paths*. A label path $\xi(v_i, v_j)$ denotes the sequence of labels of the nodes in the shortest path between v_i, v_j in a tree (including v_i, v_j). Following Aiolli et al. (2009), we consider only label paths starting at the root $v_i = v_0$, abbreviated here as $\xi(v_j)$. Implicitly, other paths may still be considered through the use of polynomial kernels with degree $d > 1$. As the authors, we compare any two paths with a function δ whose values is 1 when the paths are identical and 0 otherwise. Given two trees $T = (V, E)$ and $T' = (V', E')$, we then define a normalized polynomial kernel $K_\xi(T, T')$ over all label paths as:

$$\left(\sum_{v \in V} \sum_{v' \in V'} \frac{\delta(\xi(v), \xi(v'))}{|V| \cdot |V'|}\right)^d$$

Positional Tree Paths (a_4) In addition to label paths, Aiolli et al. (2009) define a *route* $\pi(v_i, v_j)$ as the sequence of edge indices on the shortest path between any two nodes v_i, v_j in a tree, i.e., the sequence of local positions. As above, they restrict their view to routes starting at the root, which we denote as $\pi(v_j)$, and compare them using δ. To combine positional information with label information, the authors build the product of a kernel based on the label paths and a kernel based on routes. As a result, sequential and hierarchical overall structure are compared at the same time. For overall argumentation, we define the resulting normalized polynomial product kernel $K_{\xi\pi}(T, T')$ as:

$$\left(\sum_{v \in V} \sum_{v' \in V'} \frac{\delta(\xi(v), \xi(v')) \cdot \delta(\pi(v), \pi(v'))}{(|V| \cdot |V'|)^2}\right)^d$$

Each approach, a_1–a_4, can be seen as representing one particular step of modeling overall argumentation; a_4 combines the complementary steps of a_2 and a_3, both of which implicitly include a_1.

6 Evaluation

Finally, we evaluate all four approaches to model overall argumentation from Section 5 on the three tasks associated to the corpora from Section 3.[8]

6.1 Experimental Set-up

Our goal is to assess the theoretical impact of each introduced step of modeling overall argumentation as far as possible. To this end, we conduct a systematic experiment where we use the ground-truth argument structure in each corpus for the associated downstream task based on the following set-up:

[7] We use a sequence kernel instead of flows in order to obtain a uniform setting. In Wachsmuth and Stein (2017), we also analyze flow abstractions (e.g., collapsing sequences of the same label). Here, we resort only to the original sequence.

[8] The Java source code for reproducing the experiment results is available at: http://www.arguana.com/software.html

#	Approach	Myside Bias on AAE-v2		Stance on Arg-Microtexts		Genre on Web Discourse	
		General model	*Specific model*	*General model*	*Specific model*	*General model*	*Specific model*
b_1	POS n-grams	63.3	63.3	58.8	58.8	74.0 (99.9% >a_2)	74.0 (99.9% >a_2)
b_2	Token n-grams	*70.5* (95% >b_1)	*70.5* (95% >b_1)	*65.2* (99% >a_2)	*65.2*	*75.6* (99.9% >a_2)	*75.6* (99.9% >a_2)
a_1	Label frequencies	83.4 (99.9% >b_2)	85.7 (99.9% >b_2)	49.7	54.4	62.6 (95% >a_4)	61.4
a_2	Label sequences	87.9 (99.9% >b_2)	94.7 (99.9% >a_1)	52.2	62.3	*64.5* (95% >a_3)	*64.5* (99.9% >a_3)
a_3	Label tree paths	***97.1*** (99.9% >a_2)	***97.1*** (95% >a_2)	59.8 (95% >a_1)	61.9	58.1	55.5
a_4	Positional tree paths	95.8 (99.9% >a_2)	95.6 (99.9% >a_1)	*66.7* (99% >a_2)	*67.8* (95% >a_1)	53.4	55.2
ba	**Best b_i + Best a_j**	**97.1** (99.9% >a_2)	**97.1** (95% >a_2)	69.8 (99.9% >a_2)	**71.0** (95% >a_1)	75.7 (99.9% >a_2)	**75.9** (99.9% >a_2)
	Majority baseline	62.4	62.4	52.3	52.3	64.5	64.5

Table 2: Accuracy in 10-fold cross-validation (10 repetitions, fairness in training) of all evaluated approaches on each of the three task/corpus combinations, both based on a *general model* of arguments and based on the *specific model* of the respective corpus. The highest value on each corpus is marked in bold; the best b_i and a_j in each column are italicized. In parenthesis: The confidence level in percent at which the respective approach is significantly better than the specified approach and all worse approaches.

Approaches The modeling steps are reflected by the approaches a_1–a_4 from Section 5. For each task, we measure the accuracy of all four approaches. We do this once for our *general model* of overall argumentation from Section 4 and once for the *specific model* annotated in the respective corpus, in order to assess the loss of resorting to our always applicable general model.

Baselines As a basic task-intrinsic measure, we compare a_1–a_4 to the *majority baseline* that always predicts the majority class in the given corpus. In addition, we employ two standard feature types and combine them with a_1–a_4, in order to roughly assess the need for modeling argumentation:

- b_1 *POS n-grams.* The frequency of each part-of-speech 1- to 3-gram found in $\geq$ 5% of all texts. This style feature has been effective in argumentation-related analysis tasks (Persing and Ng, 2015; Wachsmuth et al., 2016).
- b_2 *Token n-grams.* The frequency of each token 1- to 3-gram found in $\geq$ 5% of all texts. This content feature is strong in many text analysis tasks (Joachims, 1998; Pang et al., 2002).

From the tackled tasks, only myside bias has been approached on the given datasets in previous work. While we mention the respective results for completeness below, a comparison is in fact unfair due to our resort to ground-truth argument structure.

Experiments The evaluation of all approaches and baselines was done using the kernel-based machine learning platform *KeLP* (Filice et al., 2015), performing classification with the available implementation of *LibSVM* (Chang and Lin, 2011). As we target the theoretically possible impact of modeling overall argumentation, we tested a number of hyperparameter configurations.[9] We performed 10-fold cross-validation on the complete corpora and repeated each experiment 10 times, with instance shuffling in between. Then, we averaged the accuracy of each configuration over all folds and repetitions. To prevent the classifiers from using knowledge about the class distributions, we used fairness during training, i.e., each class was given an equal weight (Filice et al., 2014). Thus, the majority baseline is not a trivial competitor.

6.2 Results

Table 2 presents the best obtained results of each evaluated approach for each task/corpus combination. To clarify the reliability of the differences between the results, the table includes the confidence level (starting at 95%) at which each approach is significantly better than all weaker approaches according to a two-tailed paired student's t-test.[10]

Myside Bias on AAE-v2 The highest accuracy reported for classifying myside bias is 77.0 (Stab and Gurevych, 2016). While the comparability is limited (see above), we see that label frequencies (a_1) already achieve 83.4 and 85.7 for the general and specific model respectively, outperforming all baselines with 99.9% confidence. Matching the insights from Section 4, the sole proportion of attacks thus seems a good predictor of myside bias.

[9] SVM C parameter: 0.01, 0.1, 1, 10, 100; sequence kernel decay factor: 0, 0.5, 1; polynomial tree kernel degree: 1, 2, 3.

[10] While selecting the best result a posteriori gives an upper bound on the true effectiveness, we do this to assess to what extent each approach captures task-relevant information.

Label sequences (a_2) further improve over a_1, which underlines that also the sequential position of con stance and attack relations has an impact. a_2 is particular strong under the specific model (94.7). Unlike the general model, this model reflects some hierarchical information via the roles of argument units, such as *premise*. a_2 performs only slightly worse than the label tree paths (a_3), indicating that an adequate sequential model can compete with a hierarchical model, as we hypothesized in previous work (Wachsmuth and Stein, 2017).

Nevertheless, a_3 turns out best on AAE-v2, most likely due to its capability to capture the depth at which con stance occurs. Considering that no corpus annotation is perfect, the outstanding accuracy of 97.1 conveys an important finding: Modeling the tree structure of an argumentation basically *solves* the myside bias task without requiring other features. Neither the positional tree paths (a_4) nor the combination with token n-grams (**ba**) can add to that. Also, there is no difference between the general and the specific model, underlining that the unit roles in AAE-v2 are implicitly covered by the hierarchical structure in the general model.

Stance on Arg-Microtexts The accuracy results for the given challenging variant of stance classification (see Section 3) are much lower. Under the general model, the label frequencies (49.7) do not even compete with the majority baseline (52.3). Notable gains are achieved by the label sequences under the specific model (62.3), slightly beating the label tree paths (61.9). Putting them together in the positional tree paths (a_4) yields an accuracy of 66.7 and 67.8 respectively; more than the token n-grams (b_2, 65.2). Combining a_4 and b_2 in **ba** in turn results in the best observed accuracy value (71.0 on the specific model).

We conclude that both sequential and hierarchical overall structure are important for the distinction of pro from con argumentation, supporting our hypothesis from Section 4. They complement content-oriented approaches, such as b_2. Moreover, the fine-grained unit and relation types of the specific model annotated in Arg-Microtexts seem useful, consistently obtaining higher accuracy than the general model. Notice, though, that due to the small size of the corpus, only few reported gains are statistically significant, as shown in Table 2.

Genre on Web Discourse Although Section 4 has made minor structural differences in Web Discourse visible, Table 2 shows that a_1–a_4 all fail in genre classification: None of them beats the majority baseline (64.5), suggesting that no decisive discriminative patterns are learned. Both POS and token n-grams (b_1–b_2) significantly outperform a_1–a_4 at 99.9% confidence. While combining b_2 with a_2 (**ba**) minimally increases accuracy from 75.6 to 75.9, the results reveal that overall argumentation hardly impacts genre — as hypothesized.

7 Conclusion

This paper provides answers to the question of how the overall structure of a monological argumentative text should be modeled in order to tackle downstream tasks of computational argumentation. We have adopted the idea of including positional information in tree kernels in order to capture the explicit sequential and the implicit hierarchical overall structure of the text at the same time. In systematic experiments, we have demonstrated the strong impact of modeling overall argumentation. Most impressively, we have found that hierarchical structure decides about myside bias alone, while the combination of sequential and hierarchical structure has turned out beneficial for classifying stance. The missing impact on genre supports that the presented approaches actually capture argumentation-related properties of a text.

So far, however, we have restricted our view to ground-truth argument structure, leaving the integration of computational argument mining approaches to future work. While the noise from mining errors might qualify some of our findings, we also expect that larger corpora will allow us to discover more reliable and discriminative patterns. After all, our results underline the general importance of modeling overall argumentation.

References

Fabio Aiolli, Giovanni Da San Martino, and Alessandro Sperduti. 2009. Route kernels for trees. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 17–24.

Fabio Aiolli, Giovanni Da San Martino, and Alessandro Sperduti. 2011. Extending tree kernels with topological information. In *Proceedings of the 21th International Conference on Artificial Neural Networks - Volume Part I*, pages 142–149.

Fabio Aiolli, Giovanni Da San Martino, and Alessandro Sperduti. 2015. An efficient topological distance-based tree kernel. *IEEE Transactions on Neural Networks and Learning Systems*, 26(5):1115–1120.

Khalid Al Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, and Benno Stein. 2016. A news editorial corpus for mining argumentation strategies. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3433–3443. The COLING 2016 Organizing Committee.

Aristotle. 2007. *On Rhetoric: A Theory of Civic Discourse* (George A. Kennedy, translator). Clarendon Aristotle series. Oxford University Press.

Beata Beigman Klebanov, Christian Stab, Jill Burstein, Yi Song, Binod Gyawali, and Iryna Gurevych. 2016. Argumentation: Content, structure, and relationship with essay quality. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 70–75. Association for Computational Linguistics.

Stefanie Brüninghaus and Kevin D. Ashley. 2003. Predicting outcomes of case based legal arguments. In *Proceedings of the 9th International Conference on Artificial Intelligence and Law*, pages 233–242.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632. MIT Press.

Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, 1st edition. Cambridge University Press, New York, NY, USA.

Hal Daumé III and Daniel Marcu. 2004. A tree-position kernel for document compression. In *Proceedings of the Fourth Document Understanding Conference*.

Frans H. van Eemeren and Rob Grootendorst. 2004. *A Systematic Theory of Argumentation: The Pragma-Dialectical Approach*. Cambridge University Press, Cambridge, UK.

Adam Robert Faulkner. 2014. *Automated Classification of Argument Stance in Student Essays: A Linguistically Motivated Approach with an Application for Supporting Argument Summarization*. Dissertation, City University of New York.

Vanessa Wei Feng and Graeme Hirst. 2011. Classifying arguments by scheme. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 987–996. Association for Computational Linguistics.

Vanessa Wei Feng, Ziheng Lin, and Graeme Hirst. 2014. The impact of deep hierarchical discourse structures in the evaluation of text coherence. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 940–949. Dublin City University and Association for Computational Linguistics.

Simone Filice, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2014. Effective kernelized online learning in language processing tasks. In *Proceedings of the 36th European Conference on IR Research on Advances in Information Retrieval - Volume 8416*, pages 347–358.

Simone Filice, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2015. KeLP: A kernel-based learning platform for natural language processing. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 19–24. Association for Computational Linguistics and The Asian Federation of Natural Language Processing.

James B. Freeman. 2011. *Argument Structure: Representation and Theory*. Springer.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Ivan Habernal and Iryna Gurevych. 2015. Exploiting debate portals for semi-supervised argumentation mining in user-generated web discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2127–2137. Association for Computational Linguistics.

Thorsten Joachims. 1998. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142.

Daisuke Kimura and Hisashi Kashima. 2012. Fast computation of subpath kernel for trees. In *Proceedings of the 29th International Conference on Machine Learning*.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Raquel Mochales and Marie-Francine Moens. 2011. Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22.

Raymond J. Mooney and Razvan C. Bunescu. 2006. Subsequence kernels for relation extraction. In *Advances in Neural Information Processing Systems 18*, pages 171–178. MIT Press.

Alessandro Moschitti. 2006a. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proceedings of the 17th European Conference on Machine Learning*, pages 318–329.

Alessandro Moschitti. 2006b. Making tree kernels practical for natural language learning. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Nathan Ong, Diane Litman, and Alexandra Brusilovsky. 2014. Ontology-based argument mining and automatic essay scoring. In *Proceedings of the First Workshop on Argumentation Mining*, pages 24–28. Association for Computational Linguistics.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*.

Andreas Peldszus and Manfred Stede. 2016. An annotated corpus of argumentative microtexts. In *Argumentation and Reasoned Action: 1st European Conference on Argumentation*. College Publications.

Isaac Persing, Alan Davis, and Vincent Ng. 2010. Modeling organization in student essays. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 229–239. Association for Computational Linguistics.

Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 543–552. Association for Computational Linguistics.

Chris Reed and Glenn Rowe. 2004. Araucaria: Software for argument analysis, diagramming and representation. *International Journal of AI Tools*, 14:961–980.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, M. Mitesh Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence — An automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 440–450. Association for Computational Linguistics.

Niall Rooney, Hui Wang, and Fiona Browne. 2012. Applying kernel methods to argumentation mining. In *Proceedings of the 25th International FLAIRS Conference*, pages 272–275.

Parinaz Sobhani, Diana Inkpen, and Stan Matwin. 2015. From argumentation mining to stance classification. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 67–77. Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2014. Annotating argument components and relations in persuasive essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1501–1510. Dublin City University and Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2016. Recognizing the absence of opposing arguments in persuasive essays. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 113–118. Association for Computational Linguistics.

Manfred Stede. 2016. Towards assessing depth of argumentation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3308–3317. The COLING 2016 Organizing Committee.

Stephen E. Toulmin. 1958. *The Uses of Argument*. Cambridge University Press.

Henning Wachsmuth, Khalid Al Khatib, and Benno Stein. 2016. Using argument mining to assess the argumentation quality of essays. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1680–1691. The COLING 2016 Organizing Committee.

Henning Wachsmuth, Nona Naderi, Yufang Hou, Yonatan Bilu, Vinodkumar Prabhakaran, Alberdingk Tim Thijm, Graeme Hirst, and Benno Stein. 2017. Computational argumentation quality assessment in natural language. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 176–187. Association for Computational Linguistics.

Henning Wachsmuth and Benno Stein. 2017. A universal model for discourse-level argumentation analysis. *Special Section of the ACM Transactions on Internet Technology: Argumentation in Social Media*, 17(3):28:1–28:24.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, and Gregor Engels. 2014a. Modeling review argumentation for robust sentiment analysis. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 553–564. Dublin City University and Association for Computational Linguistics.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels, and Tsvetomira Palakarska. 2014b. A review corpus for argumentation analysis. In *Proceedings of the 15th International Conference on Intelligent Text Processing and Computational Linguistics, Part II*, pages 115–127.

Marilyn Walker, Jean Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012. A corpus for research on deliberation and debate. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 812–817. European Language Resources Association (ELRA).

Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.

Learning Generic Sentence Representations Using Convolutional Neural Networks

Zhe Gan[†], Yunchen Pu[†], Ricardo Henao[†], Chunyuan Li[†], Xiaodong He[‡], Lawrence Carin[†]
[†]Duke University, [‡]Microsoft Research, Redmond, WA 98052, USA
{zg27, yp42, r.henao, cl319, lcarin}@duke.edu
xiaohe@microsoft.com

Abstract

We propose a new encoder-decoder approach to learn distributed sentence representations that are applicable to multiple purposes. The model is learned by using a convolutional neural network as an encoder to map an input sentence into a continuous vector, and using a long short-term memory recurrent neural network as a decoder. Several tasks are considered, including sentence reconstruction and future sentence prediction. Further, a hierarchical encoder-decoder model is proposed to encode a sentence to predict multiple future sentences. By training our models on a large collection of novels, we obtain a highly generic convolutional sentence encoder that performs well in practice. Experimental results on several benchmark datasets, and across a broad range of applications, demonstrate the superiority of the proposed model over competing methods.

1 Introduction

Learning sentence representations is central to many natural language modeling applications. The aim of a model for this task is to learn fixed-length feature vectors that encode the semantic and syntactic properties of sentences. Deep learning techniques have shown promising performance on sentence modeling, via feedforward neural networks (Huang et al., 2013), recurrent neural networks (RNNs) (Hochreiter and Schmidhuber, 1997), convolutional neural networks (CNNs) (Kalchbrenner et al., 2014; Kim, 2014; Shen et al., 2014), and recursive neural networks (Socher et al., 2013). Most of these models are *task-dependent*: they are trained specifically for a certain task. However, these methods may become inefficient when we need to repeatedly learn sentence representations for a large number of different tasks, because they may require retraining a new model for each individual task. In this paper, in contrast, we are primarily interested in learning *generic* sentence representations that can be used across domains.

Several approaches have been proposed for learning generic sentence embeddings. The paragraph-vector model of Le and Mikolov (2014) incorporates a global context vector into the log-linear neural language model (Mikolov et al., 2013) to learn the sentence representation; however, at prediction time, one needs to perform gradient descent to compute a new vector. The sequence autoencoder of Dai and Le (2015) describes an encoder-decoder model to reconstruct the input sentence, while the skip-thought model of Kiros et al. (2015) extends the encoder-decoder model to reconstruct the surrounding sentences of an input sentence. Both the encoder and decoder of the methods above are modeled as RNNs.

CNNs have recently achieved excellent results in various task-dependent natural language applications as the sentence encoder (Kalchbrenner et al., 2014; Kim, 2014; Hu et al., 2014). This motivates us to propose a CNN encoder for learning generic sentence representations within the framework of encoder-decoder models proposed by Sutskever et al. (2014); Cho et al. (2014). Specifically, a CNN encoder performs convolution and pooling operations on an input sentence, then uses a fully-connected layer to produce a fixed-length encoding of the sentence. This encoding vector is then fed into a long short-term memory (LSTM) recurrent network to produce a target sentence. Depending on the task, we propose three models: (*i*) *CNN-LSTM autoencoder*: this model seeks to reconstruct the original input sentence, by capturing the *intra*-sentence information; (*ii*) *CNN-LSTM future*

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2390–2400
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

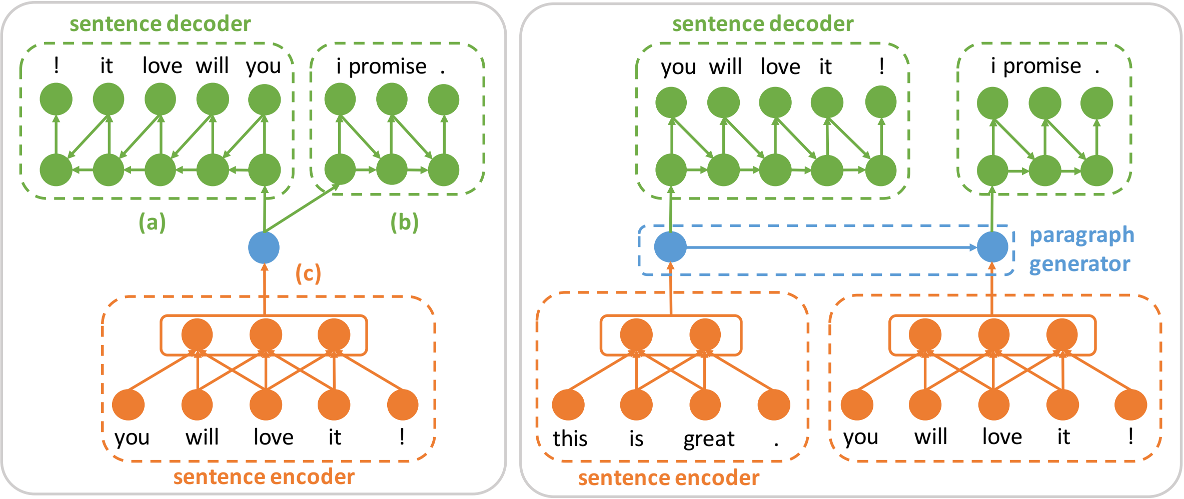


Figure 1: Illustration of the CNN-LSTM encoder-decoder models. The sentence encoder is a CNN, the sentence decoder is an LSTM, and the paragraph generator is another LSTM. (Left) (a)+(c) represents the autoencoder; (b)+(c) represents the future predictor; (a)+(b)+(c) represents the composite model. (Right) hierarchical model. In this example, the input contiguous sentences are: *this is great. you will love it! i promise*.

predictor: this model aims to predict a future sentence, by leveraging *inter*-sentence information; (*iii*) *CNN-LSTM composite model*: in this case, there are two LSTMs, decoding the representation to the input sentence itself and a future sentence. This composite model aims to learn a sentence encoder that captures both *intra*- and *inter*-sentence information.

The proposed CNN-LSTM future predictor model only considers the immediately subsequent sentence as context. In order to capture longer-term dependencies between sentences, we further introduce a hierarchical encoder-decoder model. This model abstracts the RNN language model of Mikolov et al. (2010) to the sentence level. That is, instead of using the current word in a sentence to predict future words (sentence continuation), we encode a sentence to predict multiple future sentences (paragraph continuation). This model is termed *hierarchical CNN-LSTM model*.

As in Kiros et al. (2015), we first train our proposed models on a large collection of novels. We then evaluate the CNN sentence encoder as a generic feature extractor for 8 tasks: semantic relatedness, paraphrase detection, image-sentence ranking and 5 standard classification benchmarks. In these experiments, we train a linear classifier on top of the extracted sentence features, without additional fine-tuning of the CNN. We show that our trained sentence encoder yields generic representations that perform as well as, or better, than those of Kiros et al. (2015); Hill et al. (2016), in all the tasks considered.

Summarizing, the main contribution of this paper is a new class of CNN-LSTM encoder-decoder models that is able to leverage the vast quantity of unlabeled text for learning generic sentence representations. Inspired by the skip-thought model (Kiros et al., 2015), we have further explored different variants: (*i*) CNN is used as the sentence encoder rather than RNN; (*ii*) larger context windows are considered: we propose the hierarchical CNN-LSTM model to encode a sentence for predicting multiple future sentences.

2 Model description

2.1 CNN-LSTM model

Consider the sentence pair (s_x, s_y). The encoder, a CNN, encodes the first sentence s_x into a feature vector $\boldsymbol{z}$, which is then fed into an LSTM decoder that predicts the second sentence s_y. Let $w_x^t \in \{1, \ldots, V\}$ represent the t-th word in sentences s_x, where w_x^t indexes one element in a V-dimensional set (vocabulary); w_y^t is defined similarly w.r.t. s_y. Each word w_x^t is embedded into a k-dimensional vector $\boldsymbol{x}_t = \mathbf{W}_e[w_x^t]$, where $\mathbf{W}_e \in \mathbb{R}^{k \times V}$ is a word embedding matrix (learned), and notation $\mathbf{W}_e[v]$ denotes the v-th column of matrix $\mathbf{W}_e$. Similarly, we let $\boldsymbol{y}_t = \mathbf{W}_e[w_y^t]$.

CNN encoder The CNN architecture in Kim (2014); Collobert et al. (2011) is used for sentence encoding, which consists of a convolution layer and a max-pooling operation over the entire sentence for each feature map. A sentence of length T (padded where necessary) is represented as a matrix $\mathbf{X} \in \mathbb{R}^{k \times T}$, by concatenating its word embeddings as columns, *i.e.*, the t-th column of $\mathbf{X}$ is $\boldsymbol{x}_t$.

A convolution operation involves a filter $\mathbf{W}_c \in \mathbb{R}^{k \times h}$, applied to a window of h words to produce a new feature. According to Collobert et al. (2011), we can induce one feature map $\boldsymbol{c} = f(\mathbf{X} * \mathbf{W}_c + \boldsymbol{b}) \in \mathbb{R}^{T-h+1}$, where $f(\cdot)$ is a nonlinear activation function such as the hyperbolic tangent used in our experiments, $\boldsymbol{b} \in \mathbb{R}^{T-h+1}$ is a bias vector, and $*$ denotes the convolutional operator. Convolving the same filter with the h-gram at every position in the sentence allows the features to be extracted independently of their position in the sentence. We then apply a max-over-time pooling operation (Collobert et al., 2011) to the feature map and take its maximum value, *i.e.*, $\hat{c} = \max\{\boldsymbol{c}\}$, as the feature corresponding to this particular filter. This pooling scheme tries to capture the most important feature, *i.e.*, the one with the highest value, for each feature map, effectively filtering out less informative compositions of words. Further, pooling also guarantees that the extracted features are independent of the length of the input sentence.

The above process describes how one feature is extracted from one filter. In practice, the model uses multiple filters with varying window sizes (Kim, 2014). Each filter can be considered as a linguistic feature detector that learns to recognize a specific class of n-grams (or h-grams, in the above notation). However, since the h-grams are computed in the embedding space, the model naturally handles similar h-grams composed of synonyms. Assume we have m window sizes, and for each window size, we use d filters; then we obtain a md-dimensional vector to represent a sentence.

Compared with the LSTM encoders used in Kiros et al. (2015); Dai and Le (2015); Hill et al. (2016), a CNN encoder may have the following advantages. First, the sparse connectivity of a CNN, which indicates fewer parameters are required, typically improves its statistical efficiency as well as reduces memory requirements (Goodfellow et al., 2016). For example, excluding the number of parameters used in the word embeddings, our trained CNN sentence encoder has 3 million parameters, while the skip-thought vector of Kiros et al. (2015) contains 40 million parameters. Second, a CNN is easy to implement in parallel over the whole sentence, while an LSTM needs sequential computation.

LSTM decoder The CNN encoder maps sentence s_x into a vector $\boldsymbol{z}$. The probability of a length-T sentence s_y given the encoded feature vector $\boldsymbol{z}$ is defined as

$$p(s_y|\boldsymbol{z}) = \prod_{t=1}^{T} p(w_y^t | w_y^0, \dots, w_y^{t-1}, \boldsymbol{z}) \quad (1)$$

where w_y^0 is defined as a special start-of-the-sentence token. All the words in the sentence are sequentially generated using the RNN, until the end-of-the-sentence symbol is generated. Specifically, each conditional $p(w_y^t | w_y^{<t}, \boldsymbol{z})$, where $< t = \{0, \dots, t-1\}$, is specified as softmax$(\mathbf{V}\boldsymbol{h}_t)$, where $\boldsymbol{h}_t$, the hidden units, are recursively updated through $\boldsymbol{h}_t = \mathcal{H}(\boldsymbol{y}_{t-1}, \boldsymbol{h}_{t-1}, \boldsymbol{z})$, and $\boldsymbol{h}_0$ is defined as a zero vector ($\boldsymbol{h}_0$ is not updated during training). $\mathbf{V}$ is a weight matrix used for computing a distribution over words. Bias terms are omitted for simplicity throughout the paper. The transition function $\mathcal{H}(\cdot)$ is implemented with an LSTM (Hochreiter and Schmidhuber, 1997).

Given the sentence pair (s_x, s_y), the objective function is the sum of the log-probabilities of the target sentence conditioned on the encoder representation in (1): $\sum_{t=1}^{T} \log p(w_y^t | w_y^{<t}, \boldsymbol{z})$. The total objective is the above objective summed over all the sentence pairs.

Applications Inspired by Srivastava et al. (2015), we propose three models: (*i*) an autoencoder, (*ii*) a future predictor, and (*iii*) the composite model. These models share the same CNN-LSTM model architecture, but are different in terms of the choices of the target sentence. An illustration of the proposed encoder-decoder models is shown in Figure 1(left).

The autoencoder (i) aims to reconstruct the same sentence as the input. The intuition behind this is that an autoencoder learns to represent the data using features that explain its own important factors of variation, and hence model the internal structure of sentences, effectively capturing the *intra*-sentence information. Another natural task is encoding an input sentence to predict the subsequent sentence. The future predictor (ii) achieves this, effectively capturing the *inter*-sentence information, which has been shown to be useful to learn the semantics of a sentence (Kiros et al., 2015). These two tasks can be combined to create a composite model (*iii*), where the CNN encoder is asked to learn a feature vector that is useful to simultaneously reconstruct the input sentence and predict a future sentence. This composite model encourages the sentence encoder to incorporate contextual information both within and beyond the sentence.

2.2 Hierarchical CNN-LSTM model

The future predictor described in Section 2.1 only considers the immediately subsequent sentence as context. By utilizing a larger surrounding context, it is likely that we can learn even higher-quality sentence representations. Inspired by the standard RNN-based language model (Mikolov et al., 2010) that uses the current word to predict future words, we propose a hierarchical encoder-decoder model that encodes the current sentence to predict multiple future sentences. An illustration of the hierarchical model is shown in Figure 1(right), with details provided in Figure 2.

Our proposed hierarchical model characterizes the hierarchy *word-sentence-paragraph*. A paragraph is modeled as a sequence of sentences, and each sentence is modeled as a sequence of words. Specifically, assume we are given a paragraph $D = (s_1, \ldots, s_L)$, that consists of L sentences. The probability for paragraph D is then defined as

$$p(D) = \prod_{\ell=1}^{L} p(s_\ell | s_{<\ell}) \qquad (2)$$

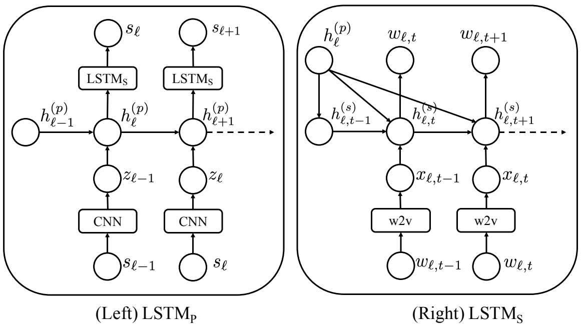
where s_0 is defined as a special start-of-the-paragraph token. As shown in Figure 2(left), each $p(s_\ell | s_{<\ell})$ in (2) is calculated as

$$p(s_\ell | s_{<\ell}) = p(s_\ell | \boldsymbol{h}_\ell^{(p)}) \qquad (3)$$

$$\boldsymbol{h}_\ell^{(p)} = \text{LSTM}_p(\boldsymbol{h}_{\ell-1}^{(p)}, \boldsymbol{z}_{\ell-1}) \qquad (4)$$

$$\boldsymbol{z}_{\ell-1} = \text{CNN}(s_{\ell-1}) \qquad (5)$$

where $\boldsymbol{h}_\ell^{(p)}$ denotes the ℓ-th hidden state of the LSTM paragraph generator, and $\boldsymbol{h}_0^{(p)}$ is fixed as a zero vector. The CNN in (5) is as described in Section 2.1, encoding the sentence $s_{\ell-1}$ into a vector representation $\boldsymbol{z}_{\ell-1}$.



(Left) LSTM$_\text{P}$ (Right) LSTM$_\text{S}$

Figure 2: Detailed illustration of the hierarchical CNN-LSTM model. (Left) LSTM paragraph generator. (Right) LSTM sentence decoder.

Equation (4) serves as the paragraph-level language model (Mikolov et al., 2010), which encodes all the previous sentence representations $\boldsymbol{z}_{<\ell}$ into a vector representation $\boldsymbol{h}_\ell^{(p)}$. This hidden state $\boldsymbol{h}_\ell^{(p)}$ is used to guide the generation of the ℓ-th sentence through the decoder (3), which is defined as

$$p(s_\ell | \boldsymbol{h}_\ell^{(p)}) = \prod_{t=1}^{T_\ell} p(w_{\ell,t} | w_{\ell,<t}, \boldsymbol{h}_\ell^{(p)}) \qquad (6)$$

where $w_{\ell,0}$ is defined as a special start-of-the-sentence token. T_ℓ is the length of sentence ℓ, and $w_{\ell,t}$ denotes the t-th word in sentence ℓ. As shown in Figure 2(right), each $p(w_{\ell,t} | w_{\ell,<t}, \boldsymbol{h}_\ell^{(p)})$ in (6) is calculated as

$$p(w_{\ell,t} | w_{\ell,<t}, \boldsymbol{h}_\ell^{(p)}) = \text{softmax}(\mathbf{V} \boldsymbol{h}_{\ell,t}^{(s)}) \qquad (7)$$

$$\boldsymbol{h}_{\ell,t}^{(s)} = \text{LSTM}_s(\boldsymbol{h}_{\ell,t-1}^{(s)}, \boldsymbol{x}_{\ell,t-1}, \boldsymbol{h}_\ell^{(p)}) \qquad (8)$$

where $\boldsymbol{h}_{\ell,t}^{(s)}$ denotes the t-th hidden state of the LSTM decoder for sentence ℓ, $\boldsymbol{x}_{\ell,t-1}$ denotes the word embedding for $w_{\ell,t-1}$, and $\boldsymbol{h}_{\ell,0}^{(s)}$ is fixed as a zero vector for all $\ell = 1, \ldots, L$. $\mathbf{V}$ is a weight matrix used for computing distribution over words.

3 Related work

Various methods have been proposed for sentence modeling, which generally fall into two categories. The first consists of models trained specifically for a certain task, typically combined with downstream applications. Several models have been proposed along this line, ranging from simple additional composition of the word vectors (Mitchell and Lapata, 2010; Yu and Dredze, 2015; Iyyer et al., 2015) to those based on complex nonlinear functions like recursive neural networks (Socher et al., 2011, 2013), convolutional neural networks (Kalchbrenner et al., 2014; Hu et al., 2014; Johnson and Zhang, 2015; Zhang et al., 2015; Gan et al., 2017), and recurrent neural networks (Tai et al., 2015; Lin et al., 2017).

The other category consists of methods aiming to learn generic sentence representations that can be used across domains. This includes the paragraph vector (Le and Mikolov, 2014), skip-thought vector (Kiros et al., 2015), and the sequential denoising autoencoders (Hill et al., 2016). Hill et al. (2016) also proposed a sentence-level log-linear bag-of-words (BoW) model, where a BoW representation of an input sentence is used to predict adjacent sentences that are also represented as BoW. Most recently, Wieting et al. (2016); Arora et al. (2017); Pagliardini et al. (2017) proposed methods in which sentences are represented as a weighted average of fixed (pre-trained) word vectors. Our model falls into this category, and is most related to Kiros et al. (2015).

However, there are two key aspects that make our model different from Kiros et al. (2015). First, we use CNN as the sentence encoder. The combination of CNN and LSTM has been considered in image captioning (Karpathy and Fei-Fei, 2015), and in some recent work on machine translation (Kalchbrenner and Blunsom, 2013; Meng et al., 2015; Gehring et al., 2016). Our utilization of a CNN is different, and more importantly, the ultimate goal of our model is different. Our work aims to use a CNN to learn generic sentence embeddings.

Second, we use the hierarchical CNN-LSTM model to predict multiple future sentences, rather than the surrounding two sentences as in Kiros et al. (2015). Utilizing a larger context window aids our model to learn better sentence representations, capturing longer-term dependencies between sentences. Similar work to this hierarchical language modeling can be found in Li et al. (2015); Sordoni et al. (2015); Lin et al. (2015); Wang and Cho (2016). Specifically, Li et al. (2015); Sordoni et al. (2015) uses an LSTM for the sentence encoder, while Lin et al. (2015) uses a bag-of-words to represent sentences.

4 Experiments

We first provide qualitative analysis of our CNN encoder, and then present experimental results on 8 tasks: 5 classification benchmarks, paraphrase detection, semantic relatedness and image-sentence ranking. As in Kiros et al. (2015), we evaluate the capabilities of our encoder as a generic feature extractor. To further demonstrate the advantage of our learned generic sentence representations, we also fine-tune our trained sentence encoder on the 5 classification benchmarks. All the CNN-LSTM models are trained using the BookCorpus dataset (Zhu et al., 2015), which consists of 70 million sentences from over 7000 books.

We train four models in total: (*i*) an autoencoder, (*ii*) a future predictor, (*iii*) the composite model, and (*iv*) the hierarchical model. For the CNN encoder, we employ filter windows (h) of sizes $\{3,4,5\}$ with 800 feature maps each, hence each sentence is represented as a 2400-dimensional vector. For both, the LSTM sentence decoder and paragraph generator, we use one hidden layer of 600 units.

The CNN-LSTM models are trained with a vocabulary size of 22,154 words. In order to learn a generic sentence encoder that can encode a large number of possible words, we use two methods of considering words not in the training set. Suppose we have a large pretrained word embedding matrix, such as the publicly available *word2vec* vectors (Mikolov et al., 2013), in which all test words are assumed to reside.

The first method learns a linear mapping between the *word2vec* embedding space $\mathcal{V}_{w2v}$ and the learned word embedding space $\mathcal{V}_{cnn}$ by solving a linear regression problem (Kiros et al., 2015). Thus, any word from $\mathcal{V}_{w2v}$ can be mapped into $\mathcal{V}_{cnn}$ for encoding sentences. The second method fixes the word vectors in $\mathcal{V}_{cnn}$ as the corresponding word vectors in $\mathcal{V}_{w2v}$, and we do not update the word embedding parameters during training. Thus, any word vector from $\mathcal{V}_{w2v}$ can be naturally used to encode sentences. By doing this, our trained sentence encoder can successfully encode 931,331 words.

For training, all weights in the CNN and non-recurrent weights in the LSTM are initialized from a uniform distribution in [-0.01,0.01]. Orthogonal initialization is employed on the recurrent matrices in the LSTM. All bias terms are initialized to zero. The initial forget gate bias for LSTM is set to 3. Gradients are clipped if the norm of the parameter vector exceeds 5 (Sutskever et al., 2014). The Adam algorithm (Kingma and Ba, 2015) with learning rate 2×10^{-4} is utilized for optimization. For all the CNN-LSTM models, we use mini-batches of size 64. For the hierarchical CNN-LSTM model, we use mini-batches of size 8, and each paragraph is composed of 8 sentences. We do not perform any regularization other than dropout (Srivastava et al., 2014). All experiments are implemented

A	you needed me?	this is great.	its lovely to see you.	he had thought he was going crazy.
B	you got me?	this is awesome.	its great to meet you.	i felt like i was going crazy.
C	i got you.	you are awesome.	its great to meet him.	i felt like to say the right thing.
D	i needed you.	you are great.	its lovely to see him.	he had thought to say the right thing.

Table 1: Vector “compositionality” using element-wise addition and subtraction. Let $z(s)$ denote the vector representation z of a given sentence s. We first calculate $z^{\star}$=z(A)-z(B)+z(C). The resulting vector is then sent to the LSTM to generate sentence D.

Query and nearest sentence
johnny nodded his curly head , and then his breath eased into an even rhythm . aiden looked at my face for a second , and then his eyes trailed to my extended hand .
i yelled in frustration , throwing my hands in the air . i stand up , holding my hands in the air .
i loved sydney , but i was feeling all sorts of homesickness . i loved timmy , but i thought i was a self-sufficient person .
“ i brought sad news to mistress betty , ” he said quickly , taking back his hand . “ i really appreciate you taking care of lilly for me , ” he said sincerely , handing me the money .
“ i am going to tell you a secret , ” she said quietly , and he leaned closer . “ you are very beautiful , ” he said , and he leaned in .
she kept glancing out the window at every sound , hoping it was jackson coming back . i kept checking the time every few minutes , hoping it would be five oclock .
leaning forward , he rested his elbows on his knees and let his hands dangle between his legs . stepping forward , i slid my arms around his neck and then pressed my body flush against his .
i take tris ’s hand and lead her to the other side of the car , so we can watch the city disappear behind us . i take emma ’s hand and lead her to the first taxi , everyone else taking the two remaining cars .

Table 2: Query-retrieval examples. In each case (block of rows), the first sentence is a query, while the second sentence is the retrieved result from a random subset of 1 million sentences from the BookCorpus dataset.

in Theano (Bastien et al., 2012), using a NVIDIA GeForce GTX TITAN X GPU with 12GB memory.

4.1 Qualitative analysis

We first demonstrate that the sentence representation learned by our model exhibits a structure that makes it possible to perform analogical reasoning using simple vector arithmetics, as illustrated in Table 1. It demonstrates that the arithmetic operations on the sentence representations correspond to word-level addition and subtractions. For instance, in the 3rd example, our encoder captures that the difference between sentence B and C is *“you"* and *“him"*, so that the former word in sentence A is replaced by the latter (*i.e.*, *“you”*-*“you”*+*“him”*=*“him”*), resulting in sentence D.

Table 2 shows nearest neighbors of sentences from a CNN-LSTM autoencoder trained on the BookCorpus dataset. Nearest neighbors are scored by cosine similarity from a random sample of 1 million sentences from the BookCorpus dataset. As can be seen, our encoder learns to accurately capture semantic and syntax of the sentences.

4.2 Quantitative evaluations

Classification benchmarks We first study the task of sentence classification on 5 datasets: *MR* (Pang and Lee, 2005), *CR* (Hu and Liu, 2004), *SUBJ* (Pang and Lee, 2004), *MPQA* (Wiebe et al., 2005) and *TREC* (Li and Roth, 2002). On all the datasets, we separately train a logistic regression model on top of the extracted sentence features. We restrict our comparison to methods that also aims to learn generic sentence embeddings for fair comparison. We also provide the state-of-the-art results using task-dependent learning methods for reference. Results are summarized in Table 3. Our CNN encoder provides better results than the combine-skip model of Kiros et al. (2015) on all the 5 datasets.

We highlight some observations. First, the autoencoder performs better than the future predictor, indicating that the *intra*-sentence information may be more important for classification than the *inter*-sentence information. Second, the hierarchi-

Method	MR	CR	SUBJ	MPQA	TREC	MSRP(Acc/F1)
ParagraphVec DM (Hill et al., 2016)	61.5	68.6	76.4	78.1	55.8	73.6 / 81.9
SDAE (Hill et al., 2016)	67.6	74.0	89.3	81.3	77.6	76.4 / 83.4
SDAE+emb. (Hill et al., 2016)	74.6	78.0	90.8	86.9	78.4	73.7 / 80.7
FastSent (Hill et al., 2016)	70.8	78.4	88.7	80.6	76.8	72.2 / 80.3
uni-skip (Kiros et al., 2015)	75.5	79.3	92.1	86.9	91.4	73.0 / 81.9
bi-skip (Kiros et al., 2015)	73.9	77.9	92.5	83.3	89.4	71.2 / 81.2
combine-skip (Kiros et al., 2015)	76.5	80.1	93.6	87.1	92.2	73.0 / 82.0
Our Results†						
autoencoder	75.53	78.97	91.97	87.96	89.8	73.61 / 82.14
future predictor	72.56	78.44	90.72	87.48	86.6	71.87 / 81.68
hierarchical model	75.20	77.99	91.66	88.21	90.0	73.96 / 82.54
composite model	76.34	79.93	92.45	88.77	91.4	74.65 / 82.21
combine‡	77.21	80.85	93.11	89.09	91.8	75.52 / 82.62
hierarchical model+emb.	75.30	79.37	91.94	88.48	90.4	74.25 / 82.70
composite model+emb.	77.16	80.64	92.14	88.67	91.2	74.88 / 82.28
combine+emb.‡	**77.77**	**82.05**	**93.63**	**89.36**	**92.6**	**76.45 / 83.76**
Task-dependent methods						
CNN (Kim, 2014)	81.5	85.0	93.4	89.6	93.6	–
AdaSent (Zhao et al., 2015)	83.1	86.3	95.5	93.3	92.4	–
Bi-CNN-MI (Yin and Schütze, 2015)	–	–	–	–	–	78.1/84.4
MPSSM-CNN (He et al., 2015)	–	–	–	–	–	78.6/84.7

Table 3: Classification accuracies on several standard benchmarks. The last column shows results on the task of paraphrase detection, where the evaluation metrics are classification accuracy and F1 score. †The first and second block in our results were obtained using the first and second method of considering words not in the training set, respectively. ‡"combine" means concatenating the feature vectors learned from both the hierarchical model and the composite model.

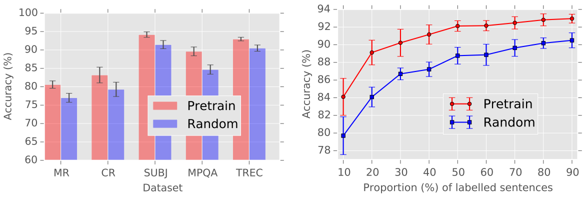


Figure 3: (Left) Effect of pretraining on the 5 classification benchmarks. The error bars are over 10 different runs. (Right) Effect of pretraining on accuracy for the TREC dataset, in terms of change in the size of the labeled training set. The error bars are over 10 different samples of training sets. Pretraining means initializing the CNN parameters from the trained CNN-LSTM composite model.

cal model performs better than the future predictor, demonstrating the importance of capturing long-term dependencies across *multiple* sentences. Our combined model, which concatenates the feature vectors learned from both the hierarchical model and the composite model, performs the best. This may be due to that: (*i*) both *intra-* and long-term *inter*-sentence information are leveraged; (*ii*) it is easier to linearly separate the feature vectors in higher dimensional spaces. Further, using (fixed) pre-trained word embeddings consistently provides better performance than using the learned word embeddings. This may be due to that *word2vec* provides more generic word representations, since it is trained on the large Google News dataset (containing 100 billion words) (Mikolov et al., 2013).

To further demonstrate the advantage of the learned generic representations, we train a CNN classifier (*i.e.*, a CNN encoder with a logistic regression model on top) with two different initialization strategies: random initialization and initialization with trained parameters from the CNN-LSTM composite model. Results are shown in Figure 3(left). The pretraining provides substantial improvements (3.52% on average) over random initialization of CNN parameters. Figure 3(right) shows the effect of pretraining as the number of labeled sentences is varied. For the TREC dataset, the performance improves from 79.7% to 84.1% when only 10% sentences are labeled. As the size of the set of labeled sentences grows, the improvement becomes smaller, as expected. For future work, our CNN-LSTM model can be also used for semi-supervised

	Image Annotation		Image Search	
Method	**R@1**	**Med** r	**R@1**	**Med** r
uni-skip[†]	30.6	3	22.7	4
bi-skip[†]	32.7	3	24.2	4
combine-skip[†]	33.8	3	25.9	4
Our Results				
hierarchical model+emb.	32.7	3	25.3	4
composite model+emb.	33.8	3	25.7	4
combine+emb.	**34.4**	3	**26.6**	4
Task-dependent methods				
DVSA[*]	38.4	1	27.4	3
m-RNN[‡]	41.0	2	29.0	3

Table 4: Results for image-sentence ranking experiments on the COCO dataset. **R@K** denotes Recall@K (higher is better) and **Med** r is the median rank (lower is better). (†) taken from Kiros et al. (2015). ($*$) taken from Karpathy and Fei-Fei (2015). (‡) taken from Mao et al. (2015).

Method	r	ρ	**MSE**
uni-skip[†]	0.8477	0.7780	0.2872
bi-skip[†]	0.8405	0.7696	0.2995
combine-skip[†]	0.8584	0.7916	0.2687
Our Results			
autoencoder	0.8284	0.7577	0.3258
future predictor	0.8132	0.7342	0.3450
hierarchical model	0.8333	0.7646	0.3135
composite model	0.8434	0.7767	0.2972
combine	0.8533	0.7891	0.2791
hierarchical model+emb.	0.8352	0.7588	0.3152
composite model+emb.	0.8500	0.7867	0.2872
combine+emb.	**0.8618**	**0.7983**	**0.2668**
Task-dependent methods			
Bi-LSTM[‡]	0.8567	0.7966	0.2736
Tree-LSTM[‡]	0.8676	0.8083	0.2532

Table 5: Results on the SICK semantic relatedness task. The evaluation metrics are Pearson's r, Spearman's ρ and mean squared error (MSE). (†) taken from Kiros et al. (2015). (‡) taken from Tai et al. (2015).

learning, with the autoencoder on all the data (labeled and unlabled), and the classifier only on the labeled data.

Paraphrase detection Now we consider paraphrase detection on the *MSRP* dataset (Dolan et al., 2004). On this task, one needs to predict whether or not two sentences are paraphrases. The training set consists of 4076 sentence pairs, and the test set has 1725 pairs. As in Tai et al. (2015), given two sentence representations $\boldsymbol{z}_x$ and $\boldsymbol{z}_y$, we first compute their element-wise product $\boldsymbol{z}_x \odot \boldsymbol{z}_y$ and their absolute difference $|\boldsymbol{z}_x - \boldsymbol{z}_y|$, and then concatenate them together. A logistic regression model is trained on top of the concatenated features to predict whether two sentences are paraphrases. We present our results on the last column of Table 3. Our best result is better than the other results that use task-independent methods.

Image-sentence ranking We consider the task of image-sentence ranking, which aims to retrieve items in one modality given a query from the other. We use the COCO dataset (Lin et al., 2014), which contains 123,287 images each with 5 captions. For development and testing we use the same splits as Karpathy and Fei-Fei (2015). The development and test sets each contain 5000 images. We further split them into 5 random sets of 1000 images, and report the average performance over the 5 splits. Performance is evaluated using Recall@K, which measures the average times a correct item is found within the top-K retrieved results. We also report the median rank of the closest ground truth result in the ranked list.

We represent images using 4096-dimensional feature vectors from VggNet (Simonyan and Zisserman, 2015). Each caption is encoded using our trained CNN encoder. The training objective is the same pairwise ranking loss as used in Kiros et al. (2015), which takes the form of $\max(0, \alpha - f(x_n, y_n) + f(x_n, y_m))$, where $f(\cdot,\cdot)$ is the image-sentence score. (x_n, y_n) denotes the related image-sentence pair, and (x_n, y_m) is the randomly sampled unrelated image-sentence pair with $n \neq m$. For image retrieval from sentences, x denotes the caption, y denotes the image, and *vice versa*. The objective is to force the matching score of the related pair (x_n, y_n) to be greater than the unrelated pair (x_n, y_m) by a margin α, which is set to 0.1 in our experiments.

Table 4 shows our results. Consistent with previous experiments, we empirically found that the encoder trained using the fixed word embedding performed better on this task, hence only results using this method are reported. As can be seen, we obtain the same median rank as in Kiros et al. (2015), indicating that our encoder is as competitive as the skip-thought vectors (Kiros et al., 2015). The performance gain between our encoder and the combine-skip model of Kiros et al. (2015) on the R@1 score is significant, which shows that the CNN encoder has more discriminative power on retrieving the most correct item than the skip-thought vector.

Semantic relatedness For our final experiment, we consider the task of semantic relatedness on the *SICK* dataset (Marelli et al., 2014), consisting of 9927 sentence pairs. Given two sentences, our goal is to produce a real-valued score between $[1, 5]$ to indicate how semantically related two sentences are, based on human generated scores. We compute a feature vector representing the pair of sentences in the same way as on the MSRP dataset. We follow the method in Tai et al. (2015), and use the cross-entropy loss for training. Results are summarized in Table 5. Our result is better than the combine-skip model of Kiros et al. (2015). This suggests that CNN also provides competitive performance at matching human relatedness judgements.

5 Conclusion

We presented a new class of CNN-LSTM encoder-decoder models to learn sentence representations from unlabeled text. Our trained convolutional encoder is highly generic, and can be an alternative to the skip-thought vectors of Kiros et al. (2015). Compelling experimental results on several tasks demonstrated the advantages of our approach. In future work, we aim to use more advanced CNN architectures (Conneau et al., 2016) for learning generic sentence embeddings.

Acknowledgments

This research was supported by ARO, DARPA, DOE, NGA, ONR and NSF.

References

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

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian Goodfellow, Arnaud Bergeron, Nicolas Bouchard, David Warde-Farley, and Yoshua Bengio. 2012. Theano: new features and speed improvements. *arXiv:1211.5590*.

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

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. In *JMLR*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for natural language processing. *arXiv:1606.01781*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING*.

Zhe Gan, PD Singh, Ameet Joshi, Xiaodong He, Jianshu Chen, Jianfeng Gao, and Li Deng. 2017. Character-level deep conflation for business data analytics. *arXiv preprint arXiv:1702.02640*.

Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2016. A convolutional encoder model for neural machine translation. *arXiv:1611.02344*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Hua He, Kevin Gimpel, and Jimmy J Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*.

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. In *Neural computation*.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*.

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

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*.

Mohit Iyyer, Varun Manjunatha, Jordan L Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *NAACL HLT*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *CVPR*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

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

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *ACL*.

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

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *EMNLP*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *ICLR*.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-rnn). In *ICLR*.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval-2014*.

Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu. 2015. Encoding source language with convolutional neural network for machine translation. In *ACL*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

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

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

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv:1703.02507*.

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

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

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM*.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Richard Socher, Eric H Huang, Jeffrey Pennington, Andrew Y Ng, and Christopher D Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*.

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

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *CIKM*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. 2015. Unsupervised learning of video representations using lstms. In *ICML*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*.

Tian Wang and Kyunghyun Cho. 2016. Larger-context language modelling with recurrent neural network. In *ACL*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. In *Language resources and evaluation*.

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

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *HLT-NAACL*.

Mo Yu and Mark Dredze. 2015. Learning composition models for phrase embeddings. *TACL*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. *arXiv:1504.05070*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*.

Repeat before Forgetting: Spaced Repetition for Efficient and Effective Training of Neural Networks

Hadi Amiri, Timothy A. Miller, Guergana Savova
Boston Children's Hospital Informatics Program, Harvard Medical School
{firstname.lastname}@childrens.harvard.edu

Abstract

We present a novel approach for training artificial neural networks. Our approach is inspired by broad evidence in psychology that shows human learners can learn efficiently and effectively by increasing intervals of time between subsequent reviews of previously learned materials (spaced repetition). We investigate the analogy between training neural models and findings in psychology about human memory model and develop an efficient and effective algorithm to train neural models. The core part of our algorithm is a cognitively-motivated scheduler according to which training instances and their "reviews" are spaced over time. Our algorithm uses only 34-50% of data per epoch, is 2.9-4.8 times faster than standard training, and outperforms competing state-of-the-art baselines.[1]

1 Introduction

Deep neural models are known to be computationally expensive to train even with fast hardware (Sutskever et al., 2014; Wu et al., 2016). For example, it takes three weeks to train a deep neural machine translation system on 100 Graphics Processing Units (GPUs) (Wu et al., 2016). Furthermore, a large amount of data is usually required to train effective neural models (Goodfellow et al., 2016; Hirschberg and Manning, 2015).

Bengio et al. (2009) and Kumar et al. (2010) developed training paradigms which are inspired by the learning principle that humans can learn more effectively when training starts with *easier* concepts and gradually proceeds with more *difficult* concepts. Since these approaches are motivated by a "starting small" strategy they are called *curriculum* or *self-paced* learning.

In this paper, we present a novel training paradigm which is inspired by the broad evidence in psychology that shows human ability to retain information improves with repeated exposure and exponentially decays with delay since last exposure (Cepeda et al., 2006; Averell and Heathcote, 2011). Spaced repetition was presented in psychology (Dempster, 1989) and forms the building block of many educational devices, including flashcards, in which small pieces of information are repeatedly presented to a learner on a schedule determined by a spaced repetition algorithm. Such algorithms show that human learners can learn efficiently and effectively by increasing intervals of time between subsequent reviews of previously learned materials (Dempster, 1989; Novikoff et al., 2012).

We investigate the analogy between training neural models and findings in psychology about human memory model and develop a spaced repetition algorithm (named Repeat before Forgetting, RbF) to efficiently and effectively train neural models. The core part of our algorithm is a scheduler that ensures a given neural network spends more time working on difficult training instances and less time on easier ones. Our scheduler is inspired by factors that affect human memory retention, namely, *difficulty* of learning materials, *delay* since their last review, and *strength* of memory. The scheduler uses these factors to lengthen or shorten review intervals with respect to individual learners and training instances. We evaluate schedulers based on their scheduling accuracy, i.e., accuracy in estimating network memory retention with respect to previously-seen instances, as well as their effect on the efficiency and effectiveness of downstream neural networks.[2]

[1] Our code is available at scholar.harvard.edu/hadi/RbF/

[2] In this paper, we use the terms memory retention, recall, and learning interchangeably.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2401–2410
Copenhagen, Denmark, September 7–11, 2017.

The contributions of this paper are: (1) we show that memory retention in neural networks is affected by the same (known) factors that affect memory retention in humans, (2) we present a novel training paradigm for neural networks based on spaced repetition, and (3) our approach can be applied without modification to any neural network.

Our best RbF algorithm uses 34-50% of training data per epoch while producing similar results to state-of-the-art systems on three tasks, namely sentiment classification, image categorization, and arithmetic addition.[3] It also runs 2.9-4.8 times faster than standard training, and outperforms competing state-of-the-art baselines.

2 Neural and Brain Memory Models

Research in psychology describes the following memory model for human learning: the probability that a human recalls a previously-seen item (e.g., the Korean translation of a given English word) depends on the *difficulty* of the item, *delay* since last review of the item, and the *strength* of the human memory. The relation between these indicators and memory retention has the following functional form (Reddy et al., 2016; Ebbinghaus, 1913):

$$\Pr(recall) = \exp(-\frac{difficulty \times delay}{strength}). \quad (1)$$

An accurate memory model enables estimating the time by which an item might be forgotten by a learner so that a review can be scheduled for the learner before that time.

We investigate the analogy between the above memory model and memory model of artificial neural networks. Our intuition is that if the probability that a network recalls an item (e.g., correctly predicts its category) depends on the same factors (difficulty of the item, delay since last review of the item, or strength of the network), then we can develop spaced repetition algorithms to efficiently and effectively train neural networks.

2.1 Recall Indicators

We design a set of preliminarily experiments to directly evaluate the effect of the aforementioned factors (recall indicators) on memory retention in neural networks. For this purpose, we use a set of training instances that are partially made available to the network during training. This scheme will allow us to intrinsically examine the effect of recall indicators on memory retention in isolation from external effects such as size of training data, number of training epochs, etc.

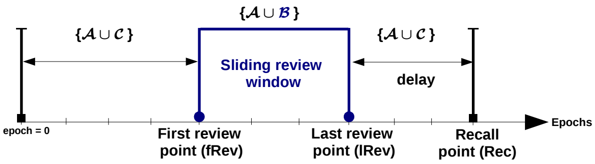

{$\mathcal{A} \cup \mathcal{B}$}
{$\mathcal{A} \cup \mathcal{C}$}
Sliding review window
{$\mathcal{A} \cup \mathcal{C}$}
delay
epoch = 0
First review point (fRev)
Last review point (lRev)
Recall point (Rec)
Epochs

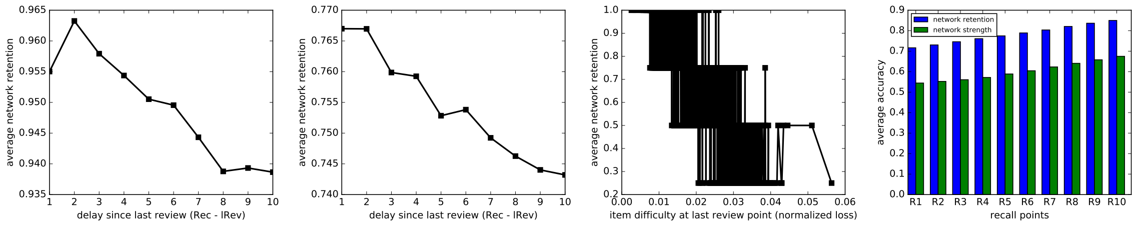
Figure 1: Effect of recall indicators on network retention. Training data is uniformly at random divided into three disjoint sets $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ that respectively contain 80%, 10%, and 10% of the data. Network retention is computed against set $\mathcal{B}$ instances at recall point.

We first define the following concepts to ease understanding the experiments (see Figure 1):

- **First** and **Last review points (fRev and lRev)** of a training instance are the first and last epochs in which the instance is used to train the network respectively,
- **Recall point (Rec)** is the epoch in which *network retention* is computed against some training instances; network retention is the probability that a neural network recalls (i.e. correctly classifies) a previously-seen training instance, and
- **Delay since last review** of a training instance is the difference between the recall point and the last review point of the training instance.

Given training data and a neural network, we uniformly at random divide the data into three disjoint sets: a *base* set $\mathcal{A}$, a *review* set $\mathcal{B}$, and a *replacement* set $\mathcal{C}$ that respectively contain 80%, 10%, and 10% of the data. As depicted in Figure 1, instances of $\mathcal{A}$ are used for training at every epoch, while those in $\mathcal{B}$ and $\mathcal{C}$ are partially used for training. The network initially starts to train with $\{\mathcal{A} \cup \mathcal{C}\}$ instances. Then, starting from the first review point, we inject the review set $\mathcal{B}$ and remove $\mathcal{C}$, training with $\{\mathcal{A} \cup \mathcal{B}\}$ instances at every epoch until the last review point. The network will then continue training with $\{\mathcal{A} \cup \mathcal{C}\}$ instances until the recall point. At this point, network retention is computed against set $\mathcal{B}$ instances, with delay defined as the number of epochs since last review point. The intuition behind using review and replacement sets, $\mathcal{B}$ and $\mathcal{C}$ respectively, is to avoid external effects (e.g.

[3] We obtained similar results on QA tasks (Weston et al., 2016) but they are excluded due to space limit.



(a) Delay full

(b) Delay partial

(c) Item difficulty

(d) Network strength

Figure 2: (a) Delay since last review vs. average network retention (accuracy) on set $\mathcal{B}$ instances at recall point. Recall point is fixed and set to the epoch in which networks obtain their best performance based on rote training. (b) The same as (a) except that recall point is set to the epoch in which networks obtain half of their best performance based on rote training. (c) Item Difficulty (normalized loss at last review point) vs. average network retention at recall point on set $\mathcal{B}$ instances. (d) Network strength (network accuracy on validation data at recall point) vs. average network retention at recall point on set $\mathcal{B}$ instances. Length of sliding window is fixed throughout experiments and set to 5 epochs.

size of data or network generalization and learning capability) for our intrinsic evaluation purpose.

To conduct these experiments, we identify different neural models designed for different tasks.[4] For each network, we fix the recall point to either the epoch in which the network is fully trained (i.e., obtains its best performance based on standard or "rote" training in which all instances are used for training at every iteration), or partially trained (i.e., obtains half of its best performance based on rote training). We report average results across these networks for each experiment.

2.1.1 Delay since Last Review

As aforementioned, delay since last review of a training instance is the difference between the recall point (Rec) and the last review point (lRev) of the training instance. We evaluate the effect of delay on network retention (against set $\mathcal{B}$ instances) by keeping the recall point fixed while moving the sliding window in Figure 1. Figures 2(a) and 2(b) show average network retention across networks for the fully and partially trained recall points respectively. The results show an inverse relationship between network retention and delay since last review in neural networks.

2.1.2 Item Difficulty

We define difficulty of training instances by the loss values generated by a network for the instances. Figure 2(c) shows the difficulty of set $\mathcal{B}$ instances at the last review point against average network retention on these instances at recall point. We normalize loss values to unit vectors (to make them comparable across networks) and then average them across networks for both fully and partially trained recall points. As the results show, network retention decreases as item difficulty increases.

2.1.3 Network Strength

We define strength of a network by its performance on validation data. To understand the effect of network strength on its retention, we use the same experimental setup as before except that we keep the delay (difference between recall point and last review point) fixed while gradually increasing the recall point; this will make the networks stronger by training them for more epochs. Then, at every recall point, we record network retention on set $\mathcal{B}$ instances and network accuracy on validation data. Average results across networks for two sets of 10 consecutive recall points (before fully and partially trained recall points) are shown in Figure 2(d). As the results show, network retention increases as memory strength increases.

The above experiments show that memory retention in neural networks is affected by the same factors that affect memory retention in humans: (a) neural networks forget training examples after a certain period of intervening training data (b): the period of recall is shorter for more difficult examples, and (c): recall improves as networks achieve better overall performance. We conclude that delay since last review, item difficulty (loss values of training instances), and memory strength (network performance on validation data) are key indicators that affect network retention and propose to design spaced repetition algorithms that take such indicators into account in training neural networks.

[4] See section 4, we use Addition and CIFAR10 datasets and their corresponding neural networks for these experiments.

Algorithm 1. Leitner System

Input: $\mathbf{H}$: training data, $\mathbf{V}$: validation data, k : number of iterations, n : number of queues
Output: trained model

```
$Q = [q_0, q_1, \ldots, q_{n-1}]$
$q_0 = [\mathbf{H}]$, $q_i = []$ for $i$ in $[1, n-1]$
For $epoch = 1$ to $k$:
  $current\_batch = []$
  For $i = 0$ to $n-1$:
    If $epoch \% 2^i == 0$:
      $current\_batch = current\_batch + q_i$
  End For
  $pmos, dmos, model = \mathbf{train}(current\_batch, \mathbf{V})$
  $\mathbf{update\_queue}(Q, pmos, dmos)$
End for
return $model$
```

q_0 epochs = $\{1, 2, 3, 4, 5, \ldots\}$
q_1 epochs = $\{2, 4, 6, 8, 10, \ldots\}$
q_2 epochs = $\{4, 8, 12, 16, 20, \ldots\}$
$\ldots$

Figure 3: Leitner System. The $\mathbf{train}(.)$ function trains the network for one epoch using instances in the $current\_batch$, and the $\mathbf{update\_queue}(.)$ function promotes the recalled (correctly classified) instances, $pmos$, to the next queue and demotes the forgotten ones, $dmos$, to q_0.

3 Spaced Repetition

We present two spaced repetition-based algorithms: a modified version of the Leitner system developed in (Reddy et al., 2016) and our Repeat before Forgetting (RbF) model respectively.

3.1 Leitner System

Suppose we have n queues $\{q_0, q_1, \ldots, q_{n-1}\}$. The Leitner system initially places all training instances in the first queue, q_0. As Algorithm 1 shows, at each training iteration, the Leitner scheduler chooses some queues to train a downstream neural network. Only instances in the selected queues will be used for training the network. During training, if an instance from q_i is recalled (e.g. correctly classified) by the network, the instance will be "promoted" to q_{i+1}, otherwise it will be "demoted" to the first queue, q_0.[5]

The Leitner scheduler reviews instances of q_i at every 2^i iterations. Therefore, instance in lower queues (difficult/forgotten instances) are reviewed more frequently than those in higher queues (easy/recalled ones). Figure 3 (bottom) provides examples of queues and their processing epochs. Note that the overhead imposed on training by the Leitner system is $O(|current\_batch|)$ at every epoch for moving instances between queues.

3.2 RbF Model

3.2.1 RbF Memory Models

The challenge in developing memory models is to estimate the time by which a training instance should be reviewed before it is forgotten by the network. Accurate estimation of the review time leads to efficient and effective training. However, a heuristic scheduler such as Leitner system is suboptimal as its hard review schedules (i.e. only 2^i-iteration delays) may lead to early or late reviews.

We develop flexible schedulers that take recall indicators into account in the scheduling process. Our schedulers lengthen or shorten inter-repetition intervals with respect to individual training instances. In particular, we propose using density kernel functions to estimate the latest epoch in which a given training instance can be recalled. We aim to investigate how much improvement (in terms of efficiency and effectiveness) can be achieved using more flexible schedulers that utilize the recall indicators.

We propose considering density kernels as schedulers that favor (i.e., more confidently delay) less difficult training instances in stronger networks. As a kernel we can use any non-increasing function of the following quantity:

$$x_i = \frac{d_i \times t_i}{s_e}, \tag{2}$$

where d_i indicates the loss of network for a training instance $h_i \in \mathbf{H}$, t_i indicates the number of epochs to next review of h_i, and s_e indicates the performance of network— on validation data— at epoch e. We investigate the Gaussian, Laplace, Linear, Cosine, Quadratic, and Secant kernels as described below respectively:

$$f_{gau}(x, \tau) = \exp(-\tau x^2), \tag{3}$$

$$f_{lap}(x, \tau) = \exp(-\tau x), \tag{4}$$

$$f_{lin}(x, \tau) = \begin{cases} 1 - \tau x & x < \frac{1}{\tau} \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

$$f_{cos}(x, \tau) = \begin{cases} \frac{1}{2}\cos(\tau \pi x) + 1 & x < \frac{1}{\tau} \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

$$f_{qua}(x, \tau) = \begin{cases} 1 - \tau x^2 & x^2 < \frac{1}{\tau} \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

$$f_{sec}(x, \tau) = \frac{2}{\exp(-\tau x^2) + \exp(\tau x^2)}, \tag{8}$$

[5] Note that in (Reddy et al., 2016) demoted instances are moved to q_{i-1}. We observed significant improvement in Leitner system by moving such instances to q_0 instead of q_{i-1}.

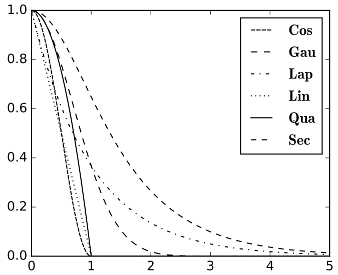


Figure 4: RbF kernel functions with $\tau = 1$.

where τ is a learning parameter. Figure 4 depicts these kernels with $\tau = 1$. As we will discuss in the next section, we use these kernels to optimize delay with respect to item difficulty and network strength for each training instance.

3.2.2 RbF Algorithm

Our Repeat before Forgetting (RbF) model is a spaced repetition algorithm that takes into account the previously validated recall indicators to train neural networks, see Algorithm 2. RbF divides training instances into *current* and *delayed* batches based on their delay values at each iteration. Instances in the current batch are those that RbF is less confident about their recall and therefore are reviewed (used to re-train the network) at current iteration. On the other hand, instances in the delayed batch are those that are likely to be recalled by the network in the future and therefore are not reviewed at current epoch. At each iteration, the RbF scheduler estimates the optimum delay (number of epochs to next review) for each training instance in the current batch. RbF makes such item-specific estimations as follows:

Given the difficulty of a training instance d_i, the memory strength of the neural network at epoch e, s_e, and an RbF memory model f (see section 3.2.1), RbF scheduler estimates the maximum delay $\hat{t_i}$ for the instance such that it can be recalled with a confidence greater than the given threshold $\eta \in (0, 1)$ at time $e + \hat{t_i}$. As described before, d_i and s_e can be represented by the current loss of the network for the instance and the current performance of the network on validation data respectively. Therefore, the maximum delay between the current (epoch e) and next reviews of the instance can be estimated as follows:

$$\hat{t_i} = \arg\min_{t_i} \big(f(x_i, \hat{\tau}) - \eta\big)^2, \quad (9)$$
$$s.t \quad 1 \leq t_i \leq k - e$$

Algorithm 2. RbF Training Model

Input: $\mathbf{H}$: training data, $\mathbf{V}$: validation data, k : number of iterations, f : RbF kernel, η: recall confidence
Output: trained model

```
0  $t_i = 1$ for $h_i \in \mathbf{H}$
1  For $epoch = 1$ to $k$:
2    $current\_batch = \{h_i : t_i <= 1\}$
3    $delayed\_batch = \{h_i : t_i > 1\}$
4    $s_{epoch}, model = \mathbf{train}(current\_batch, \mathbf{V})$
5    $\hat{\tau} = \arg\min_{\tau} \big(f(x_i, \tau) - a_i\big)^2 \quad \forall h_i \in \mathbf{V}, a_i \geq \eta$
6    $\hat{t_i} = \arg\min_{t_i} \big(f(x_i, \hat{\tau}) - \eta\big)^2 \forall h_i \in current\_batch$
7    $t_i = t_i - 1 \quad \forall h_i \in delayed\_bach$
8  End for
9  return $model$
```

Figure 5: RbF training model. The $\mathbf{train}(.)$ function at line 5 trains the network for one epoch using instances in the $current\_batch$. Note that at each iteration $epoch$, x_i is computed using Equation (2) and strength of the current model, s_{epoch}.

where $\hat{\tau}$ is the optimum value for the learning parameter obtained from validation data, see Equation (10). In principle, reviewing instances could be delayed for any number of epochs; in practice however, delay is bounded both below and above (e.g., by queues in the Leitner system). Thus, we assume that, at each epoch e, instances could be delayed for at least one iteration and at most $k - e$ iterations where k is the total number of training epochs. We also note that t_i is a lower bound of the maximum delay as s_e is expected to increase and d_i is expected to decrease as the network trains in next iterations.

Algorithm 2 shows the outline of the proposed RbF model. We estimate the optimum value of τ (line 5 of Algorithm 2) for RbF memory models using validation data. In particular, RbF uses the loss values of validation instances and strength of the network obtained at the previous epoch to estimate network retention for validation instances at the current epoch (therefore $t_i = 1$ for every validation instance). The parameter τ for each memory model is computed as follows:

$$\hat{\tau} = \arg\min_{\tau} \big(f(x_j, \tau) - a_j\big)^2, \forall h_j \in \mathbf{V}, a_j \geq \eta, \quad (10)$$

where $a_j \in (0, 1)$ is the current accuracy of the model for the validation instance h_j. RbF then predicts the delay for current batch instances and reduces the delay for those in the delayed batch by one epoch. The overhead of RbF is $O(|\mathbf{H}|)$ to compute delays and $O(|\mathbf{V}|)$ to compute $\hat{\tau}$. Note that (9) and (10) have closed form solutions.

Dataset	train, dev, test	Network	Task
IMDb	20K, 5K, 25K	MLP/fastext (Joulin et al., 2017), best epoch=8	sentiment analysis
CIFAR10	45K, 5K, 10K	CNN (Chan et al., 2015) best epoch=64	image classification
Addition	40K, 5K, 10K	LSTM (Sutskever et al., 2014) best epoch=32	arithmetic addition

Table 1: Datasets, models, and tasks.

4 Experiments

Table 1 describes the tasks, datasets, and models that we consider in our experiments. It also reports the training epochs for which the models produce their best performance on validation data (based on rote training). We note that the Addition dataset is randomly generated and contains numbers with at most 4 digits.[6]

We consider three schedulers as baselines: a slightly modified version of the Leitner scheduler (Lit) developed in Reddy et al. (2016) for human learners (see Footnote 5), curriculum learning (CL) in which training instances are scheduled with respect to their *easiness* (Jiang et al., 2015), and the uniform scheduler of rote training (Rote) in which all instances are used for training at every epoch. For Lit, we experimented with different queue lengths, $n = \{3, 5, 7\}$, and set $n = 5$ in the experiments as this value led to the best performance of this scheduler across all datasets.

Curriculum learning starts training with *easy* instances and gradually introduces more complex instances for training. Since easiness information is not readily available in most datasets, previous approaches have used heuristic techniques (Spitkovsky et al., 2010; Basu and Christensen, 2013) or optimization algorithms (Jiang et al., 2015, 2014) to quantify easiness of training instances. These approaches consider an instance as easy if its loss is smaller than a threshold (λ). We adopt this technique as follows: at each iteration e, we divide the entire training data into *easy* and *hard* sets using iteration-specific λ_e and the loss values of instances, obtained from the current partially-trained network. All easy instances in conjunction with $\alpha_e \in [0, 1]$ fraction of easiest hard instances (those with smallest loss values greater than λ_e) are used for training at iteration e. We set each λ_e to the average loss of training instances that are correctly classified by the current partially-trained network. Furthermore, at each iteration e, we set $\alpha_e = e/k$ to gradually introduce complex instances at every new iteration.[7] Note that we treat all instances as easy at $e = 0$.

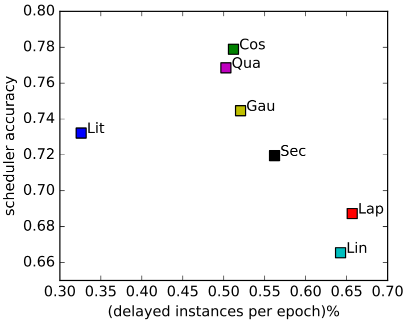


Figure 6: Accuracy of schedulers in predicting network retention. For these experiments recall confidence is set to its default value, $\eta = 0.5$.

Performance values reported in experiments are averaged over 10 runs of systems and the confidence parameter η is always set to 0.5 unless otherwise stated.

4.1 Evaluation of Memory Models

In these experiments, we evaluate memory schedulers with respect to their accuracy in predicting network retention for delayed instances. Since curriculum learning does not estimate delay for training instances, we only consider Leitner and RbF schedulers in these experiments.

For this evaluation, if a scheduler predicts a delay t for a training instance h at epoch e, we evaluate network retention with respect to h at epoch $e + t$. If the network recalls (correctly classifies) the instance at epoch $e + t$, the scheduler has correctly predicted network retention for h, and otherwise, it has made a wrong prediction. We use this binary outcome to evaluate the accuracy of each scheduler. Note that the performance of schedulers on instances that have not been delayed is not a major concern. Although failing to delay an item inversely affects efficiency, it makes the network stronger by providing more instances to train from. Therefore, we consider a good scheduler as the one that accurately delays more items.

Figure 6 depicts the average accuracy of schedulers in predicting networks' retention versus the average fraction of training instances that they delayed per epoch. As the results show, all schedulers

[6] https://github.com/fchollet/keras/blob/master/examples/addition_rnn.py

[7] k is the total number of iterations.

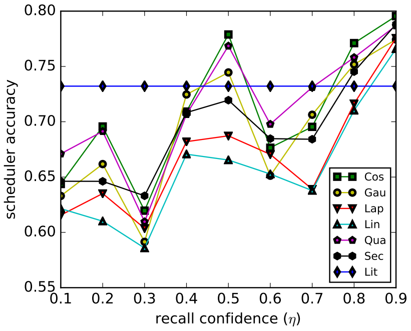


Figure 7: Effect of recall confidence η on the accuracy of different schedulers in predicting network retention (best seen in color.)

delay substantial amount of instances per epoch. In particular, Cos and Qua outperform Lit in both predicting network retention and delaying items, delaying around 50% of training instances per epoch. This is while Gau and Sec show comparable accuracy to Lit but delay more instances. On the other hand, Lap, which has been found effective in Psychology, and Lin are less accurate in predicting network retention. This is because of the tradeoff between delaying more instances and creating stronger networks. Since these schedulers are more flexible in delaying greater amount of instances, they might not provide networks with enough data to fully train.

Figure 7 shows the performance of RbF schedulers with respect to the recall confidence parameter η, see Equation (9). As the results show, schedulers have poor performance with smaller values of η. This is because smaller values of η make schedulers very flexible in delaying instances. However, the performance of schedulers are not dramatically low even with very small ηs. Our further analyses on the delay patterns show that although a smaller η leads to more delayed instances, the delays are significantly shorter. Therefore, most delayed instances will be "reviewed" shortly in next epochs. These bulk reviews make the network stronger and help it to recall most delayed instance in future iterations.

On the other hand, greater ηs lead to more accurate schedulers at the cost of using more training data. In fact, we found that larger ηs do not delay most training instances in the first few iterations. However, once the network obtains a reasonably high performance, schedulers start delaying instances for longer durations. We will further study this effect in the next section.

Model	Accuracy	TIPE	X Faster	Gain
CL	0.868	0.71	0.93	1.40
Lit	0.859	0.67	2.87	1.85
Gau	0.874	0.48	3.02	2.16
Lap	0.857	**0.34**	4.66	**3.15**
Lin	0.864	0.36	**4.78**	3.07
Cos	0.868	0.49	2.90	2.10
Qua	0.871	0.50	2.95	2.08
Sec	0.866	0.44	3.09	2.33
Cos $\eta = 0.9$	**0.880**	0.76	2.36	1.42
Rote	0.887	1.00	1.00	1.00

Table 2: Comparison of schedulers in terms of average network accuracy, average fraction of instances used for training per epoch (TIPE), and the extent to which a model runs faster than Rote training (X Times Faster). Gain column indicates the $\frac{Accuracy}{TIPE}$ improvement over rote training.

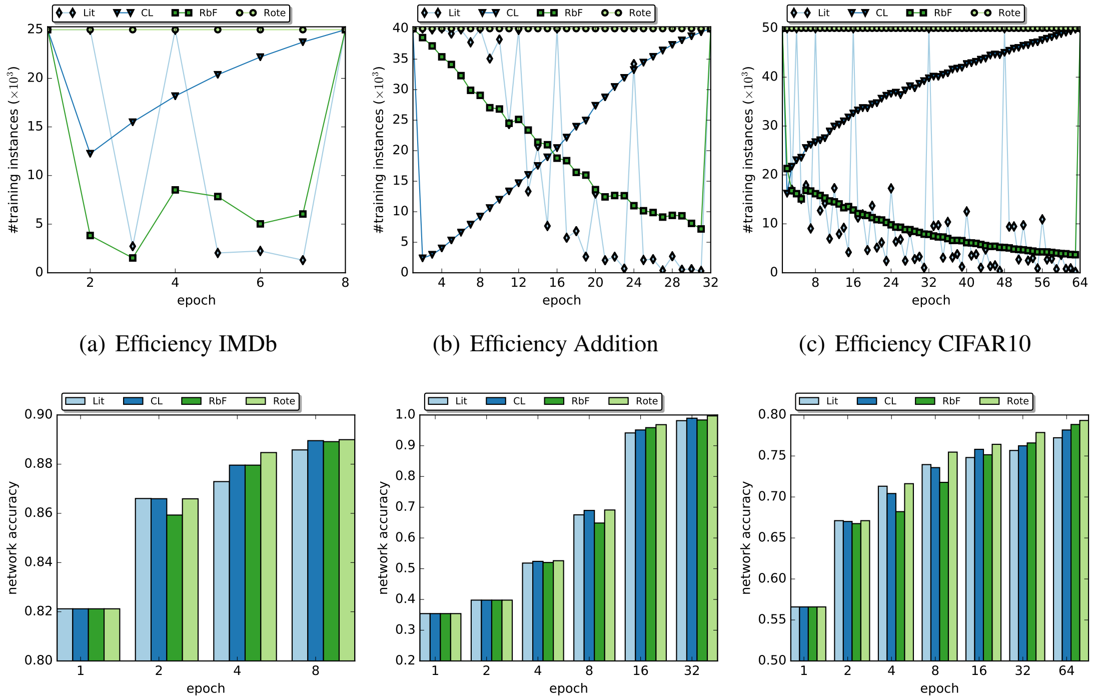
4.2 Efficiency and Effectiveness

We compare RbF against Leitner and curriculum learning in terms of efficiency of training and effectiveness of trained models. We define effectiveness as the accuracy of a trained network on balanced test data, and efficiency as (a): fraction of instances used for training per epoch, and (b): required time for training the networks. For RbF schedulers, we set η to 0.5 and consider the best performing kernel Cosine with $\eta = 0.9$ based on results in Figure 7.

The results in Table 2 show that all training paradigms have comparable effectiveness (Accuracy) to that of rote training (Rote). Our RbF schedulers use less data per epoch (34-50% of data) and run considerably faster than Rote (2.90-4.78 times faster for $\eta = 0.5$). The results also show that Lit is slightly less accurate but runs 2.87 time faster than Rote; note that, as a scheduler, Lit is less accurate than RbF models, see Figures 6 and 7.

In addition, CL leads to comparable performance to RbF but is considerably slower than other schedulers. This is because this scheduler has to identify easier instances and sort the harder ones to sample training data at each iteration. Overall, the performance of Lit, CL, Cos $\eta = .5$ and Cos $\eta = .9$ are only 2.76, 1.90, 1.88, and 0.67 absolute values lower than that of Rote respectively. Considering the achieved efficiency, these differences are negligible (see the overall gain in Table 2).

Figure 8 reports detailed efficiency and effectiveness results across datasets and networks. For clear illustration, we report accuracy at iterations 2^i $\forall i$ in which Lit is trained on the entire data, and consider Cos $\eta = .5$ as RbF scheduler. In terms of efficiency (first row of Figure 8), CL starts with (small set of)



(a) Efficiency IMDb

(b) Efficiency Addition

(c) Efficiency CIFAR10

(d) Effectiveness IMDb

(e) Effectiveness Addition

(f) Effectiveness CIFAR10

Figure 8: Efficiency and Effectiveness of schedulers across three datasets and networks. RbF uses Cos $\eta = .5$ as kernel. CL starts with (small set of) easier instances and gradually incorporate slightly harder instances at each iteration. Lit and RbF start big and gradually delay reviewing easy instances.

easier instances and gradually increases the amount of training data by adding slightly harder instances into its training set. On the other hand, Lit and RbF start big and gradually delay reviewing (easy) instances that the networks have learned. The difference between these two training paradigms is apparent in Figures 8(a)-8(c).

The results also show that the efficiency of a training paradigm depends on the initial effectiveness of the downstream neural network. For CL to be efficient, the neural network need to initially have *low* performance (accuracy) so that the scheduler works on smaller set of easy instances. For example, in case of Addition, Figures 8(b) and 8(e), the initial network accuracy is only 35%, therefore most instances are expected to be initially treated as hard instances and don't be used for training. On the other hand, CL shows a considerably lower efficiency for networks with slightly high initial accuracy, e.g. in case of IMDb or CIFAR10 where the initial network accuracy is above 56%, see Figures 8(a) and 8(d), and 8(c) and 8(f) respectively.

In contrast to CL, Lit and RbF are more efficient when the network has a relatively higher initial performance. A higher initial performance helps the schedulers to more confidently delay "reviewing" most instances and therefore train with a much smaller set of instances. For example, since the initial network accuracy in IMDb or CIFAR10 is above 56%, Lit and RbF are considerably more efficient from the beginning of the training process. However, in case of low initial performance, Lit and RbF tend to avoid delaying instances at lower iterations which leads to poor efficiency at the beginning. This is the case for the Addition dataset in which instances are gradually delayed by these two schedulers even at epoch 8 when the performance of the network reaches above 65%, see Figures 8(e) and 8(b). However, Lit gains its true efficiency after iteration 12, see Figure 8(b), while RbF still gradually improves the efficiency. This might be because of the lower bound delays that RbF estimates, see Equation (9).

Furthermore, the effectiveness results in Figure 8 (bottom) show that all schedulers produce comparable accuracy to the Rote scheduler throughout the training process, not just at specific iterations. This indicates that these training paradigms can much faster achieve the same generalizability as standard training, see Figures 8(b) and 8(e).

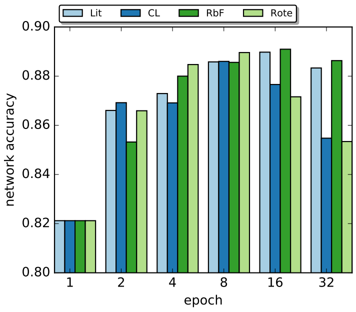


Figure 9: Robustness against overtraining.

4.3 Robustness against Overtraining

We investigate the effect of spaced repetition on overtraining. The optimal number of training epochs required to train `fastText` on the IMDb dataset is 8 epochs (see Table 1). In this experiment, we run `fastText` on IMDb for greater number of iterations to investigate the robustness of different schedulers against overtraining. The results in Figure 9 show that Lit and RbF (Cos $\eta = 0.5$) are more robust against overtraining. In fact, the performance of Lit and RbF further improve at epoch 16 while CL and Rote overfit at epoch 16 (note that CL and Rote also require considerably more amount of time to reach to higher iterations). We attribute the robustness of Lit and RbF to the scheduling mechanism which helps the networks to avoid retraining with easy instances. On the other hand, overtraining affects Lit and RbF at higher training iterations, compare performance of each scheduler at epochs 8 and 32. This might be because these training paradigms overfit the network by paying too much training attention to very hard instances which might introduce noise to the model.

5 Related Work

Ebbinghaus (1913, 2013), and recently Murre and Dros (2015), studied the hypothesis of the exponential nature of forgetting, i.e. how information is lost over time when there is no attempt to retain it. Previous research identified three critical indicators that affect the probability of recall: repeated exposure to learning materials, elapsed time since their last review (Ebbinghaus, 1913; Wixted, 1990; Dempster, 1989), and more recently item difficulty (Reddy et al., 2016). We based our investigation on these findings and validated that these indicators indeed affect memory retention in neural networks. We then developed training paradigms that utilize the above indicators to train networks.

Bengio et al. (2009) and Kumar et al. (2010) also developed cognitively-motivated training paradigms which are inspired by the principle that learning can be more effective when training starts with easier concepts and gradually proceeds with more difficult ones. Our idea is motivated by the spaced repetition principle which indicates learning improves with repeated exposure and decays with delay since last exposure (Ebbinghaus, 1913; Dempster, 1989). Based on this principle, we developed schedulers that space the reviews of training instances over time for efficient and effective training of neural networks.

6 Conclusion and Future Work

We developed a cognitively-motivated training paradigm (scheduler) that space instances over time for efficient and effective training of neural networks. Our scheduler only uses a small fraction of training data per epoch but still effectively train neural networks. It achieves this by estimating the time (number of epochs) by which training could be delayed for each instance. Our work was inspired by three recall indicators that affect memory retention in humans, namely difficulty of learning materials, delay since their last review, and memory strength of the learner, which we validated in the context of neural networks.

There are several avenues for future work including the extent to which our RbF model and its kernels could be combined with curriculum learning or Leitner system to either predict easiness of novel training instances to inform curriculum learning or incorporate Leitner's queueing mechanism to the RbF model. Other directions include extending RbF to dynamically learn the recall confidence parameter with respect to network behavior, or developing more flexible delay functions with theoretical analysis on their lower and upper bounds.

Acknowledgments

We thank Mitra Mohtarami for her constructive feedback during the development of this paper and anonymous reviewers for their thoughtful comments. This work was supported by National Institutes of Health (NIH) grant R01GM114355 from the National Institute of General Medical Sciences (NIGMS). The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

Lee Averell and Andrew Heathcote. 2011. The form of the forgetting curve and the fate of memories. *Journal of Mathematical Psychology*, 55(1):25–35.

Sumit Basu and Janara Christensen. 2013. Teaching classification boundaries to humans. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 109–115. AAAI Press.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Nicholas J Cepeda, Harold Pashler, Edward Vul, John T Wixted, and Doug Rohrer. 2006. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological bulletin*, 132(3):354.

Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. 2015. Pcanet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing*, 24(12):5017–5032.

Frank N Dempster. 1989. Spacing effects and their implications for theory and practice. *Educational Psychology Review*, 1(4):309–330.

Hermann Ebbinghaus. 1913. *Memory: A contribution to experimental psychology*. 3. University Microfilms.

Hermann Ebbinghaus. 2013. Memory: A contribution to experimental psychology. *Annals of neurosciences*, 20(4):155.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Julia Hirschberg and Christopher D Manning. 2015. Advances in natural language processing. *Science*, 349(6245):261–266.

Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. 2014. Self-paced learning with diversity. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2078–2086. Curran Associates, Inc.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G. Hauptmann. 2015. Self-paced curriculum learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2694–2700.

Armand Joulin, Edouard Grave, and Piotr Bojanowski Tomas Mikolov. 2017. Bag of tricks for efficient text classification. *European Chapter of the Association for Computational Linguistics, EACL 2017*, page 427.

M Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197.

Jaap MJ Murre and Joeri Dros. 2015. Replication and analysis of ebbinghaus' forgetting curve. *PloS one*, 10(7):e0120644.

Timothy P Novikoff, Jon M Kleinberg, and Steven H Strogatz. 2012. Education of a model student. *Proceedings of the National Academy of Sciences*, 109(6):1868–1873.

Siddharth Reddy, Igor Labutov, Siddhartha Banerjee, and Thorsten Joachims. 2016. Unbounded human learning: Optimal scheduling for spaced repetition. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1815–1824. ACM.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How less is more in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759. Association for Computational Linguistics.

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

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. *5th International Conference on Learning Representations*.

John T Wixted. 1990. Analyzing the empirical course of forgetting. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 16(5):927.

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

Part-of-Speech Tagging for Twitter with Adversarial Neural Networks

Tao Gui, Qi Zhang[*], Haoran Huang, Minlong Peng, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{tgui16,qz,huanghr15,mlpeng16,xjhuang}@fudan.edu.cn

Abstract

In this work, we study the problem of part-of-speech tagging for Tweets. In contrast to newswire articles, Tweets are usually informal and contain numerous out-of-vocabulary words. Moreover, there is a lack of large scale labeled datasets for this domain. To tackle these challenges, we propose a novel neural network to make use of out-of-domain labeled data, unlabeled in-domain data, and labeled in-domain data. Inspired by adversarial neural networks, the proposed method tries to learn common features through adversarial discriminator. In addition, we hypothesize that domain-specific features of target domain should be preserved in some degree. Hence, the proposed method adopts a sequence-to-sequence autoencoder to perform this task. Experimental results on three different datasets show that our method achieves better performance than state-of-the-art methods.

1 Introduction

During the last decade, social media have become extremely popular, on which billions of user-generated contents are posted every day. Many users have been writing about their thoughts and lives on the go. The massive unstructured data from social media provides valuable information for a variety of applications such as stock prediction (Bollen et al., 2011), public health analysis (Wilson and Brownstein, 2009; Paul and Dredze, 2011), real-time event detection (Sakaki et al., 2010), and so on. The quality of these applications is highly impacted by the performance of natural language processing tasks.

@DORSEY33	lol	aw	i	thought	u
USR	UH	UH	PRP	VBD	PRP

was	talkin	bout	another	time	.	nd	i	dnt
VBD	VBG	IN	DT	NN	.	CC	PRP	VBP

see	u	either	!
VB	PRP	RB	.

Figure 1: An example of tagged Tweet, which contains nonstandard orthography, emoticon, and abbreviation. The tagset is defined similar as that of PTB (Marcus et al., 1993).

Part-of-speech (POS) tagging is one of the most important natural language processing tasks. It has also been widely used in the social media analysis systems (Ritter et al., 2012; Lamb et al., 2013; Kiritchenko et al., 2014). Most state-of-the-art POS tagging approaches are based on supervised methods. Hence, they usually require a large amount of annotated data to train models. Many datasets have been constructed for POS tagging task. Because newswire articles are carefully edited, benchmarks usually use them for annotation (Marcus et al., 1993). However, user-generated contents on social media are usually informal and contain many nonstandard lexical items. Moreover, the difference in domains between training data and evaluation data may heavily impact the performance of approaches based on supervised methods (Caruana and Niculescu-Mizil, 2006). Hence, most POS tagging methods cannot achieve the same performance as reported on newswire domain when applied on Twitter (Owoputi et al., 2013).

To perform the Twitter POS tagging task, some approaches have been proposed to perform the task. Gimpel et al. (2011) manually annotated 1,827 tweets and carefully studied various fea-

[*] Corresponding author.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2411–2420
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

tures. Ritter et al. (2011) also constructed a labeled dataset, which contained 787 tweets, to empirically evaluate the performance of supervised methods on Twitter. Owoputi et al. (2013) incorporated word clusters into the feature sets and further improved the performance. From these works, we can observe that the size of the training data was much smaller than the newswire domain's.

Besides the challenge of lack of training data, the frequent use of out-of-vocabulary words also makes this problem difficult to address. Social media users often use informal ways of expressing their ideas and often spell words phonetically (e.g., "2mor" for "tomorrow"). In addition, they also make extensive use of emoticons and abbreviations (e.g., ":-)" for smiling emotion and "LOL" for laughing out loud). Moreover, new symbols, abbreviations, and words are constantly being created. Figure 1 shows an example of tagged Tweet.

To tackle the challenges posed by the lack of training data and the out-of-vocabulary words, in this paper, we propose a novel recurrent neural network, which we call *Target Preserved Adversarial Neural Network* (TPANN) to perform the task. It can make use of a large quantity of annotated data from other resource-rich domains, unlabeled in-domain data, and a small amount of labeled in-domain data. All of these datasets can be easily obtained. To make use of unlabeled data, motivated by the work of Goodfellow et al. (2014) and Chen et al. (2016), the proposed method extends the bi-directional long short-term memory recurrent neural network (bi-LSTM) with an adversarial predictor. To overcome the defect that adversarial networks can merely learn the common features, we propose to use an autoencoder only acting on target dataset to preserve its own specific features. For tackling the out-of-vocabulary problem, the proposed method also incorporates a character level convolutional neutral network to leverage subword information.

The contributions of this work are as follows:

- We propose to incorporate large scale unlabeled in-domain data, out-of-domain labeled data, and in-domain labeled data for Twitter part-of-speech tagging task.

- We introduce a novel recurrent neural network, which can learn domain-invariant representations through in-domain and out-of-domain data and construct a cross domain POS tagger through the learned representations. The proposed method also tries to preserve the specific features of target domain.

- Experimental results demonstrate that the proposed method can lead to better performance in most of cases on three different datasets.

2 Approach

In this work, we propose a novel recurrent neural network, Target Preserved Adversarial Neural Network (TPANN), to learn common features between resource-rich domain and target domain, simultaneously to preserve target domain-specific features. It extends the bi-directional LSTM with adversarial network and autoencoder. The architecture of TPANN is illustrated in Figure 2. The model consists of four components: *Feature Extractor*, *POS Tagging Classifier*, *Domain Discriminator* and *Target Domain Autoencoder*. In the following sections, we will detail each part of the proposed architecture and training methods.

2.1 Feature Extractor

The feature extractor $\mathcal{F}$ adopts CNN to extract character embedding features, which can tackle the out-of-vocabulary word problem effectively. To incorporate word embedding features, we concatenate word embedding to character embedding as the input of bi-LSTM on the next layer. Utilizing a bi-LSTM to model sentences, $\mathcal{F}$ can extract sequential relations and context information.

We denote the input sentence as $\mathbf{x}$ and the i-th word as x_i. $x_i \in \mathcal{S}(x)$ and $x_i \in \mathcal{T}(x)$ represent input samples are from source domain and target domain, respectively. We denote the parameters of $\mathcal{F}$ as θ_f. Let $\mathcal{V}$ be the vocabulary of words, and $\mathcal{C}$ be the vocabulary of characters. d is the dimensionality of character embedding then $Q \in \mathbb{R}^{d\times|\mathcal{C}|}$ is the representation matrix of vocabulary. We assume that word $x_i \in \mathcal{V}$ is made up of a sequence of characters $\mathbf{C}^i = [c_1, c_2, \dots, c_l]$, where l is the max length of word and every word will be padded to this length. Then $\mathbf{C}^i \in \mathbb{R}^{d\times l}$ would be the inputs of CNN.

We apply a narrow convolution between $\mathbf{C}^i$ and filter $\mathbf{H} \in \mathbb{R}^{d\times k}$, where k is the width of the filter.

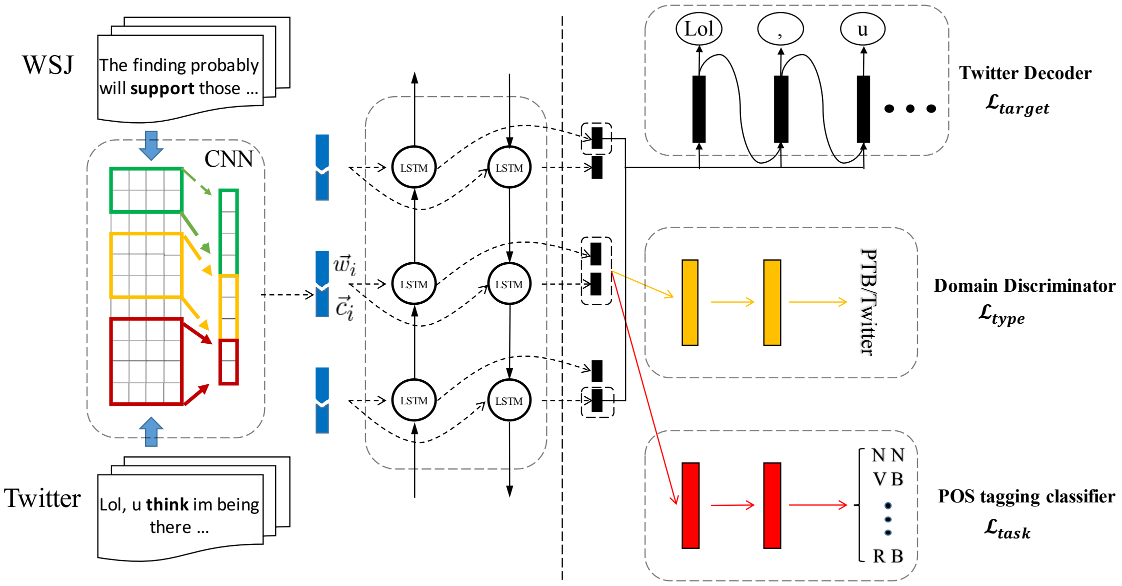


Figure 2: The general architecture of the proposed method.

After that we add a bias and apply nonlinearity to obtain a feature map $\mathbf{m}^i \in \mathbb{R}^{l-k+1}$. Specifically, the j-th element of $\mathbf{m}^i$ is given by:

$$\mathbf{i}^k[j] = \tanh(\langle \mathbf{C}^i[*, j : j+k-1], \mathbf{H} \rangle + b), \quad (1)$$

where $\mathbf{C}^k[*, j : j+k-1]$ is the j-to-$(j+k-1)$-th column of $\mathbf{C}^i$ and $\langle A, B \rangle = \text{Tr}(AB^T)$ is the Frobenius inner product. We then apply a max-over-time pooling operation (Collobert et al., 2011) over the feature map. CNN uses multiple filters with varying widths to obtain the feature vector $\vec{c}_i$ for word x_i. Then, the character-level feature vector $\vec{c}_i$ is concatenated to the word embedding $\vec{w}_i$ to form the input of bi-LSTM on the next layer. The word embedding $\vec{w}$ is pretrained on 30 million tweets. Then, the hidden states $\mathbf{h}$ of bi-LSTM turn into the features that will be transfered to $\mathcal{P}$, $\mathcal{Q}$ and $\mathcal{R}$, i.e. $\mathcal{F}(\mathbf{x}) = \mathbf{h}$.

2.2 POS Tagging Classifier and Domain Discriminator

POS tagging classifier $\mathcal{P}$ and domain discriminator $\mathcal{Q}$ take $\mathcal{F}(x)$ as input. They are standard feed-forward networks with a softmax layer for classification. $\mathcal{P}$ predicts POS tagging label to get classification capacity, and $\mathcal{Q}$ discriminates domain label to make $\mathcal{F}(x)$ domain-invariant.

The POS tagging classifier $\mathcal{P}$ maps the feature vector $\mathcal{F}(x_i)$ to its label. We denote the parameters of this mapping as θ_y. The POS tagging classifier is trained on N_s samples from the source domain with the cross entropy loss:

$$\mathcal{L}_{task} = -\sum_{i=1}^{N_s} y_i * \log \hat{y}_i, \quad (2)$$

where y_i is the one-hot vector of POS tagging label corresponding to $x_i \in \mathcal{S}(x)$, $\hat{y}_i$ is the output of top softmax layer: $\hat{y}_i = \mathcal{P}(\mathcal{F}(x_i))$. During the training time, The parameters θ_f and θ_y are optimized to minimize the classification loss $\mathcal{L}_{task}$. This ensures that $\mathcal{P}(\mathcal{F}(x_i))$ can make accurate prediction on the source domain.

Conversely, domain discriminator maps the same hidden states $\mathbf{h}$ to the domain labels with parameters θ_d. The domain discriminator aims to discriminate the domain label with following loss function:

$$\mathcal{L}_{type} = -\sum_{i=1}^{N_s+N_t} \{d_i \log \hat{d}_i + (1-d_i)\log(1-\hat{d}_i)\}, \quad (3)$$

where d_i is the ground truth domain label for sample i, $\hat{d}_i$ is the output of top layer: $\hat{d}_i = \mathcal{Q}(\mathcal{F}(x_i))$. N_t means N_t samples from the target domain. The domain discriminator is trained towards a saddle point of the loss function through minimizing the loss over θ_d while maximizing the loss over θ_f (Ganin et al., 2016). Optimizing θ_f ensures that the domain discriminator can't discriminate the domain, i.e., the feature extractor finds the common features between the two domains.

2.3 Target Domain Autoencoder

Through training adversarial networks, we can obtain domain-invariant features $\mathbf{h}_{common}$, but it will weaken some domain-specific features which are useful for POS tagging classification. Merely obtaining domain invariant features would therefore limit the classification ability.

Our model tries to tackle this defect by introducing domain-specific autoencoder $\mathcal{R}$, which attempts to reconstruct target domain data. Inspired by (Sutskever et al., 2014) but different from (Dai and Le, 2015), we treat the feature extractor $\mathcal{F}$ as encoder. In addition, we combine the last hidden states of the forward LSTM and backward LSTM in $\mathcal{F}$ as the initial state $h_0(dec)$ of the decoder LSTM. Hence, we don't need to reverse the order of words of the input sentences and the model avoids the difficulty of ”establish communication” between the input and the output (Sutskever et al., 2014).

Similar to (Zhang et al., 2016), we use $h_0(dec)$ and embedding vector of the previous word as the inputs of the decoder, but in a computationally more efficient manner by computing previous word representation. We assume that $(\hat{x}_1, \cdots, \hat{x}_T)$ is the output sequence. z_t is the t-th word representation: $z_t = MLP(h_t)$, and MLP is the multiple perceptron function. Hidden state $h_t = LSTM([h_0(dec) : z_{t-1}], h_{t-1})$, where $[\cdot : \cdot]$ is the concatenation operation. We estimate the conditional probability $p(\hat{x_1}, \cdots, \hat{x_T}|h_0(dec))$ as follows:

$$p(\hat{x_1}, \cdots, \hat{x_T}|h_0(dec)) = \prod_{t=1}^{T} p(\hat{x_t}|h_0(dec), z_1, \cdots, z_{t-1}), \tag{4}$$

where each $p(\hat{x_t}|h_0(dec), z_1, \cdots, z_{t-1})$ distribution is computed with softmax over all the words in the vacabulary.

Our aim is to minimize the following loss function with respect to parameters θ_r:

$$\mathcal{L}_{target} = -\sum_{i=1}^{N_t} x_i * \log \hat{x}_i, \tag{5}$$

where x_i is the one-hot vector of i-th word. This makes $h_0(dec)$ learn an undercomplete and most salient sentence representation of target domain data. When the adversarial networks try to optimize the hidden representation to common representation $\mathbf{h}_{common}$, The target domain autoencoder counteracts a tendency of the adversarial network to erase target domain features by optimizing the common representation to be informative on the target-domain data.

2.4 Training

Our model can be trained end-to-end with standard back-propagation, which we will detail in this section.

Our ultimate training goal is to minimize the total loss function with parameters $\{\theta_f, \theta_y, \theta_r, \theta_d\}$ as follows:

$$\mathcal{L}_{total} = \alpha\mathcal{L}_{task} + \beta\mathcal{L}_{target} + \gamma\mathcal{L}_{type}, \tag{6}$$

where α, β, γ are the weights to balance the effects of $\mathcal{P}$, $\mathcal{R}$ and $\mathcal{Q}$.

For obtaining domain-invariant representation $\mathbf{h}_{common}$, inspired by (Ganin and Lempitsky, 2015), we introduce a special gradient reversal layer (GRL), which does nothing during forward propagation, but negates the gradients if it receives backward propagation, i.e. $g(\mathcal{F}(x)) = \mathcal{F}(x)$ but $\nabla g(\mathcal{F}(x)) = -\lambda\nabla\mathcal{F}(x)$. We insert the GRL between $\mathcal{F}$ and $\mathcal{Q}$, which can run standard Stochastic Gradient Descent with respect to θ_f and θ_d. The parameter $-\lambda$ drives the parameters θ_f not to amplify the dissimilarity of features when minimize $\mathcal{L}_{tpye}$. So by introducing a GRL, $\mathcal{F}$ can drive its parameters θ_f to extract hidden representations that help the POS tagging classification and hamper the domain discrimination.

In order to preserve target domain-specific features, we only optimize the autoencoder on target domain data for reconstruction tasks.

Through above procedures, the model can learn the common features between domains, simultaneously preserve target domain-specific features. Finally, we can update the parameters as follows:

$$\begin{aligned}
\theta_f &= \theta_f - \mu(\alpha\frac{\partial\mathcal{L}^i_{task}}{\partial\theta_f} + \beta\frac{\partial\mathcal{L}^i_{target}}{\partial\theta_f} - \gamma\cdot\lambda\frac{\partial\mathcal{L}^i_{type}}{\partial\theta_f}) \\
\theta_y &= \theta_y - \mu\cdot\alpha\frac{\partial\mathcal{L}^i_{task}}{\partial\theta_y} \\
\theta_r &= \theta_r - \mu\cdot\beta\frac{\partial\mathcal{L}^i_{target}}{\partial\theta_r} \\
\theta_d &= \theta_d - \mu\cdot\gamma\frac{\partial\mathcal{L}^i_{type}}{\partial\theta_d},
\end{aligned} \tag{7}$$

Dataset		# Tokens
WSJ		1,173,766
UNL		1,177,746
RIT-Twitter	RIT-Train	10,652
	RIT-Dev	2,242
	RIT-Test	2,291
NPSCHAT		44,997
ARK-Twitter	OCT27	26,594
	DAILY547	7,707

Table 1: The statistics of the datasets used in our experiments.

where μ is the learning rate. Because the size of the WSJ is more than 100 times that of the labeled Twitter dataset, if we directly train the model with the combined dataset, the final results are much worse than those using two training steps. So, we adopt adversarial training on WSJ and unlabeled Twitter dataset at the first step, then use a small number of in-domain labeled data to fine-tune the parameters with a low learning rate.

3 Experiments

In this section, we will detail the datasets used for experiments and experimental setup.

3.1 Datasets

The methods proposed in this work incorporate out-of-domain labeled data from resource-rich domains, large scale unlabeled in-domain data, and a small number of labeled in-domain data. The datasets used in this work are as follows:

Labeled out-of-domain data. We use a standard benchmark dataset for adversarial POS tagging, namely the Wall Street Journal (WSJ) data from the Penn TreeBank v3 (Marcus et al., 1993), sections 0-24 for the out-of-domain data.

Labeled in-domain data. For training and evaluating POS tagging approaches, we compare the proposed method with other approaches on three benchmarks: RIT-Twitter (Ritter et al., 2011), NPSCHAT (Forsyth, 2007), and ARK-Twitter (Gimpel et al., 2011).

Unlabeled in-domain data. For training the adversarial network, we need to use a dataset that has large scale unlabeled tweets. Hence, in this work, we construct large scale unlabeled data (UNL), from Twitter using its API.

The detailed data statistics of the datasets used in this work are listed in Table 1.

3.2 Experimental Setup

We select both state-of-the-art and classic methods for comparison, as follows:

- **Stanford POS Tagger**: Stanford POS Tagger is a widely used tool for newswire domains (Toutanova et al., 2003). In this work, we train it using two different sets, the WSJ (sections 0-18) and a WSJ, IRC, and Twitter mixed corpus. We use **Stanford-WSJ** and **Stanford-MIX** to represent them, respectively.

- **T-POS**: T-Pos (Ritter et al., 2011) adopts the Conditional Random Fields and clustering algorithm to perform the task. It was trained from a mixture of hand-annotated tweets and existing POS-labeled data.

- **GATE Tagger**: GATE tagger (Derczynski et al., 2013) is based on vote-constrained bootstrapping with unlabeled data. It combines cases where available taggers use different tagsets.

- **ARK Tagger**: ARK tagger (Owoputi et al., 2013) is a system that reports the best accuracy on the RIT dataset. It uses unsupervised word clustering and a variety of lexical features.

- **bi-LSTM**: Bidirectional Long Short-Term Memory (LSTM) networks have been widely used in a variety of sequence labeling tasks (Graves and Schmidhuber, 2005). In this work, we evaluate it at character level, word level, and combining them together. **bi-LSTM (word level)** uses one layer of bi-LSTM to extract word-level features and adopts a random initialization method to transform words to vectors. **bi-LSTM (character level)** represents a method that combines bi-LSTM and CNN-based character embedding, a similar approach with character-aware neural network described in (Kim et al., 2015) to handle the out-of-vocabulary words. **bi-LSTM (word level pretrain)** architecture is the same as that of bi-LSTM(word level) but adopts word2vec tool (Mikolov et al., 2013) to vectorize. **bi-LSTM (combine)** concatenates word to character features.

Methods	RIT-Test	RIT-Dev
Stanford-WSJ (Toutanova et al., 2003)	73.37%	83.29%
Stanford-MIX	83.14%	84.19%
T-POS (Ritter et al., 2011)	84.55%	84.83%
GATE Tagger (Derczynski et al., 2013)	88.69%	89.37%
ARK Tagger (Owoputi et al., 2013)	90.40%	-
bi-LSTM (word level)	75.91%	76.94%
bi-LSTM (word level pretrain)	85.99%	86.93%
bi-LSTM (character level)	82.85%	84.30%
bi-LSTM (combine)	89.48%	89.30%
bi-LSTM (combine + WSJ)	83.54%	83.64%
bi-LSTM (combine + WSJ + adversarial)	83.76%	84.45%
bi-LSTM (combine + WSJ + fine-tune)	89.87%	90.23%
bi-LSTM (combine + WSJ + adversarial + fine-tune)	90.60%	90.73%
TPANN (combine + WSJ + adversarial + fine-tune + autoencoder)	**90.92%**	**91.08%**

Table 2: Token level accuracies of different methods on RIT-Test and RIT-Dev. bi-LSTM(combine) refers to combining word level with character level. bi-LSTM(combine + WSJ) refers to the model trained on WSJ and tested on RIT. bi-LSTM(combine + WSJ + adversarial) refers to adversarial model trained on 1.1 million tokens of labeled WSJ data and the same scale of unlabeled Twitter data, then tested on RIT. Fine-tune means adding RIT-train data to fine-tune.

The hyper-parameters used for our model are as follows. AdaGrad optimizer trained with cross-entropy loss is used with 0.1 as the default learning rate. The dimensionality of word embedding is set to 200. The dimensionality for random initialized character embedding is set to 25. We adopt a bi-LSTM for encoding with each layer consisting of 250 hidden neurons. We set three layers of standard LSTM for decoding. Each LSTM layer consists of 500 hidden neurons. Adam optimizer trained with cross-entropy loss is used to fine-tune with 0.0001 as the default learning rate. Fine-tuning is run for 100 epochs using early stop.

4 Results and Discussion

In this section, we will report experimental results and a detailed analysis of the results for the three different datasets.

4.1 Evaluation on RIT-Twitter

The RIT-Twitter is split into training, development and evaluation sets (RIT-Train, RIT-Dev, RIT-Test). The splitting method is shown in (Derczynski et al., 2013), and the dataset statistics are listed in Table 1. Table 2 shows the results of our method and other approaches on the RIT-Twitter dataset. RIT-Twitter uses the PTB tagset with several Twitter-specific tags: retweets, @usernames, $hashtags$, and urls. Since words in these categories can be tagged almost perfectly using simple regular expressions, similar to (Owoputi et al., 2013), we use regular expressions to tags these words appropriately for all systems.

From the results of the Stanford-WSJ, we can observe that the newswire domain is different from Twitter. Although the token-level accuracy of the Stanford POS Tagger is higher than 97.0% on the PTB dataset, its performance on Twitter drops sharply to 73.37%. By incorporating some in-domain labeled data for training, the accuracy of Stanford POS Tagger can reach up to 83.14%. Taking a variety of linguistic features and many other resources into consideration, the T-POS, GATE tagger, and ARK tagger can achieve better performance.

The second part of Table 2 shows the results of the bi-LSTM based methods, which are trained on the RIT-Train dataset. According to the results of word level, we can see that word2vec can provide valuable information. The pre-trained word vectors in bi-LSTM(word level pretrain) give almost 10% higher accuracy than bi-LSTM(word level).

Comparing the character-level bi-LSTM with word-level bi-LSTM with random initialization, we can observe that the character-level method can achieve better performance than the word-level method. bi-LSTM(combine) combines word with character features, as described in Section 2.1,

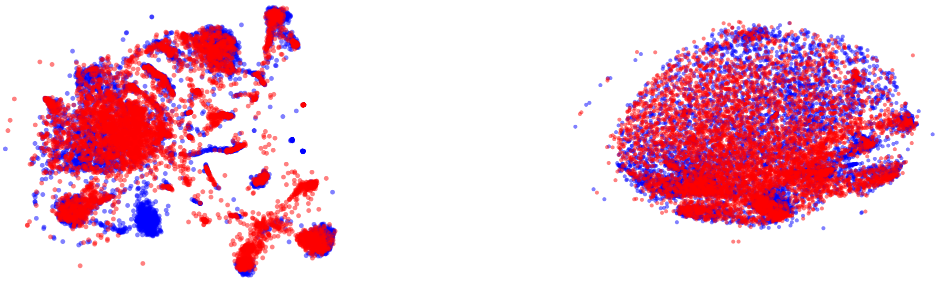


Figure 3: The visualization of bi-LSTM's outputs of the extracted features. The left figure shows the results when no adversary is performed. The right figure shows the results when the adversary procedure is incorporated into training. Blue points correspond to the source PTB domain examples, and red points correspond to the target Twitter domain.

which achieves the best results at 89.48% in the bi-LSTM based baseline systems and shows that the morphological features and pre-trained word vectors are both useful for POS tagging.

The third part of Table 2 shows the results of our methods incorporating out-of-domain labeled data, in-domain unlabeled data, and in-domain labeled data. Putting everything together, our model can achieve 90.92% on this dataset. Compared with the architecture without an adversarial model, our method is almost 1% better. It demonstrates that adversarial networks can significantly help with tasks of this nature. Through introducing the autoencoder in target domain, we can preserve domain-specific features for better performance. Compared with the ARK tagger, which achieves the previous best result on this dataset, our model is also 0.52% better, the error reduction rate is more than 5.5%.

To better understand why adversarial networks can help transfer domains from newswire to Twitter, in this work we also followed the method Ganin and Lempitsky (2015) used to visualize the outputs of LSTM with t-SNE (Van Der Maaten, 2013). Figure 3 shows the visualization results. From the figure, we can see that the adversary in our method makes the two distributions of features much more similar, which means that the outputs of bi-LSTM are domain-invariant. Hence, the PTB training data can provide much more help than directly combining PTB and RIT-Train together.

Methods	**Accuracy**
Forsyth (2007)	90.8%
ARK Tagger	93.4% ± 0.3%
TPANN	**94.1%**

Table 3: Tagging accuracies on NPSChat Corpus.

4.2 Evaluation on NPSChat

IRC, which contains Internet relay room messages from 2006, is a medium of online conversational text. Its content is very similar to tweets. We evaluate the proposed method on the NPSChat corpus (Forsyth, 2007), a PTB-part-of-speech annotated dataset of IRC.

We compared our method with a tagger in the same setup as experiments with (Forsyth, 2007). The training part contains 90% of the data. The testing part contains the other 10%. Table 3 shows the results of the ARK Tagger and our method. We used PTB, unlabeled Twitter, and the training part of NPSChat to train our model. From the results, we can see that our model achieved 94.1% accuracy. This is significantly better than the result Forsyth (2007) reported, which was 90.8%. They trained their tagger with a mix of several POS-annotated corpora (12K from Twitter, 40K from IRC, and 50K from PTB). Our method also outperforms state-of-the-art results 93.4%±0.3%, which was achieved by the ARK Tagger with various external corpus and features, e.g., Brown clustering, PTB, Freebase lists of celebrities, and video games.

Methods	Accuracy
Gimpel et al. (2011) version 0.2	90.8%
ARK Tagger	93.2%
TPANN	92.8%

Table 4: Tagging accuracies on DAILY547.

4.3 Evaluation on ARK-Twitter

ARK-Twitter data contains an entire dataset consisting of a number of tweets sampled from one particular day (October 27, 2010) described in (Gimpel et al., 2011). This part is used for training. They also created another dataset, which consists of 547 tweets, for evaluation (DAILY547). This dataset consists of one random English tweet from every day between January 1, 2011 and June 30, 2012. The distribution of training data may be slightly different from the testing data, for example a substantial fraction of the messages in the training data are about a basketball game. Since ARK-Twitter uses a different tagset with PTB, we manually construct a table to link tags for the two datasets.

Table 4 shows the results of different methods on this dataset. From the results, we can see that our method can achieve a better result than (Gimpel et al., 2011). However, the performance of our method is worse than the ARK Tagger. Through analyzing the errors, we find that 16.7% errors occurr between nouns and proper nouns. Since our method do not include any ontology or knowledge, proper nouns can not be easily detected. However, the ATK Tagge add a token-level name list feature. The name list is useful for proper nouns recognition, which fires on names from many sources, such as Freebase lists of celebrities, the Moby Words list of US Locations, proper names from Mark Kantrowitz's name corpus and so on. So, our model is also competitive when lacking of manual feature knowledge.

5 Related Work

Part-of-Speech tagging is an important preprocessing step and can provide valuable information for various natural language processing tasks. In recent years, deep learning algorithms have been successfully used for POS tagging. A number of approaches have been proposed and have achieved some progress. Santos and Guimaraes (2015) proposed using a character-based convolutional neural network to perform the POS tagging problem. Bi-LSTMs with word, character or unicode byte embedding were also introduced to achieve the POS tagging and named entity recognition tasks (Plank et al., 2016; Chiu and Nichols, 2015; Ma and Hovy, 2016). In this work, we study the problem from a domain adaption perspective. Inspired by these works, we also propose to use character-level methods to handle out-of-vocabulary words and bi-LSTMs to model the sequence relations.

Adversarial networks were successfully used for image generation (Goodfellow et al., 2014; Dosovitskiy et al., 2015; Denton et al., 2015), domain adaption (Tzeng et al., 2014; Ganin et al., 2016), and semi-supervised learning (Denton et al., 2016). The key idea of adversarial networks for domain adaption is to construct invariant features by optimizing the feature extractor as an adversary against the domain classifier (Zhang et al., 2017).

Sequence autoencoder reads the input sequence into a vector and then tries to reconstruct it. Dai and Le (2015) used the model on a number of different tasks and verified its validity. Li et al. (2015) introduced the model to hierarchically build an embedding for a paragraph, showing that the model was able to encode texts to preserve syntactic, semantic, and discourse coherence.

In this work, we incorporate adversarial networks with autoencoder to obtain domain-invariant features and keep domain-specific features. Our model is more suitable for target domain tasks.

6 Conclusion

In this work, we propose a novel adversarial neural network to address the POS tagging problem. Besides learning common representations between source domain and target domain, it can simultaneously preserve specific features of target domain. The proposed method leverages newswire resources and large scale in-domain unlabeled data to help POS tagging classification on Twitter, which has a few of labeled data. We evaluate the proposed method and several state-of-the-art methods on three different corpora. In most of the cases, the proposed method can achieve better performance than previous methods. Experimental results demonstrate that the proposed method can make full use of these resources, which can be easily obtained.

Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011, 61473092, and 61472088) and STCSM (No.16JC1420401).

References

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

Rich Caruana and Alexandru Niculescu-Mizil. 2006. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM.

Xilun Chen, Ben Athiwaratkun, Yu Sun, Kilian Weinberger, and Claire Cardie. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614*.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

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

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087.

Emily Denton, Sam Gross, and Rob Fergus. 2016. Semi-supervised learning with context-conditional generative adversarial networks. *arXiv preprint arXiv:1611.06430*.

Emily L Denton, Soumith Chintala, Rob Fergus, et al. 2015. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *RANLP*, pages 198–206.

Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. 2015. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546.

Eric N Forsyth. 2007. Improving automated lexical and discourse analysis of online chat dialog.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of The 32nd International Conference on Machine Learning*, page 11801189.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. In *AAAI 2016*.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762.

Alex Lamb, Michael J Paul, and Mark Dredze. 2013. Separating fact from fear: Tracking flu infections on twitter. In *HLT-NAACL*, pages 789–795.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

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

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

Michael J Paul and Mark Dredze. 2011. You are what you tweet: Analyzing twitter for public health. *ICWSM*, 20:265–272.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529*.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM.

Cicero Nogueira dos Santos and Victor Guimaraes. 2015. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*.

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

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*.

Laurens Van Der Maaten. 2013. Barnes-hut-sne. *arXiv preprint arXiv:1301.3342*.

Kumanan Wilson and John S Brownstein. 2009. Early detection of disease outbreaks using the internet. *Canadian Medical Association Journal*, 180(8):829–831.

Yizhe Zhang, Zhe Gan, and Lawrence Carin. 2016. Generating text via adversarial training. *NIPS 2016 Workshop on Adversarial Training*.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *arXiv preprint arXiv:1701.00188*.

Investigating Different Syntactic Context Types and Context Representations for Learning Word Embeddings

Bofang Li[1,2]
libofang@ruc.edu.cn

Tao Liu[1,2]
tliu@ruc.edu.cn

Zhe Zhao[1,2]
helloworld@ruc.edu.cn

Buzhou Tang[3]
tangbuzhou@gmail.com

Aleksandr Drozd[4]
alex@smg.is.titech.ac.jp

Anna Rogers[5]
arogers@cs.uml.edu

Xiaoyong Du[1,2]
duyong@ruc.edu.cn

[1] School of Information, Renmin University of China
[2] Key Laboratory of Data Engineering and Knowledge Engineering, MOE
[3] Shenzhen Graduate School, Harbin Institute of Technology
[4] Global Scientific Information and Computing Center, Tokyo Institute of Technology
[5] Department of Computer Science, University of Massachusetts Lowell

Abstract

The number of word embedding models is growing every year. Most of them are based on the co-occurrence information of words and their contexts. However, it is still an open question what is the best definition of context. We provide a systematical investigation of 4 different syntactic context types and context representations for learning word embeddings. Comprehensive experiments are conducted to evaluate their effectiveness on 6 extrinsic and intrinsic tasks. We hope that this paper, along with the published code, would be helpful for choosing the best context type and representation for a given task.

1 Introduction

Recently, there is a growing interest in word embedding models, where words are embedded into low-dimensional (dense) real-valued vectors. The trained word embeddings can be directly used for solving intrinsic tasks like word similarity and word analogy. They are also helpful for solving extrinsic tasks, such as part-of-speech tagging, chunking, named entity recognition (Collobert and Weston, 2008; Collobert et al., 2011) and text classification (Socher et al., 2013; Kim, 2014).

The training objectives of word embedding models are based on the Distributional Hypothesis (Harris, 1954) that can be stated as follows: "words that occur in similar contexts tend to have similar meanings". In most word embedding models, the "context" is defined as the words which precede and follow the target word within some fixed distance (Bengio et al., 2003; Mnih and Hinton, 2007; Mikolov et al., 2013a; Pennington et al., 2014). Among them, Global Vectors (GloVe) proposed by Pennington et al. (2014), Continuous Skip-Gram (CSG) [1] and Continuous Bag-Of-Words (CBOW) proposed by Mikolov et al. (2013b) achieve state-of-the-art results on a range of linguistic tasks, and scale to corpora with billions of words.

The traditional sparse vector-space models have explored many different types of context. Curran (2004); Padó and Lapata (2007); Clark (2012) have discussed a set of context definitions beyond simple linear context. For example, a sentence or document could be used as the boundary instead of window size. Contextual words could be associated with their relative sides (left/right) or positions (+1/-2) to the target word. They could also be associated with part-of-speech or grammatical relation labels. The weight of each contextual word can be explicitly defined. Moreover, words that are connected to target word in dependency parse

[1] Many researches refer to Continuous Skip-Gram as SG. However, in order to distinguish linear (continuous) context and DEPS (dependency-based) context, we refer it as CSG.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2421–2431
Copenhagen, Denmark, September 7–11, 2017.

Basic Model	Context Type Context Representation	Linear	DEPS
generalized Skip-Gram	unbound	CSG (Mikolov et al., 2013a)	this work
	bound	Structured SG (Ling et al., 2015) POSIT (Levy and Goldberg, 2014b)	DEPS (Levy and Goldberg, 2014a)
generalized Bag-Of-Words	unbound	CBOW (Mikolov et al., 2013a)	this work
	bound	CWINDOW (Ling et al., 2015)	this work
original GloVe	unbound	GloVe (Pennington et al., 2014)	this work
	bound	this work	this work

Table 1: Summary of prior research on word embedding models with different syntactic context types and context representations. For linear context, *bound* indicates words associated with positional information. For DEPS context, *bound* indicates words associated with dependency relation.

tree can be considered as context.

Recent word embedding models have also explored some of the above context types. Levy and Goldberg (2014b); Ling et al. (2015)[2] improve CSG and CBOW by introducing position-aware context representation. Levy and Goldberg (2014a) propose dependency-based context (DEPS) for CSG.

However, different types of syntactic context have not been systematically compared for different word embeddings. This paper explores two context types (linear or DEPS) and two context representations (bound or unbound), as shown in Table 1. Three popular word embedding models (CBOW, GloVe, and CSG) are compared on word similarity, word analogy, part-of-speech tagging, chunking, named entity recognition, and text classification tasks.

2 Related Work

Several studies directly compare different word embedding models. Lai et al. (2016) compare 6 word embedding models using different corpora and hyper-parameters. Nayak and Manning (2016) provide a set of evaluations, along with an online tool, for word embedding models. Levy and Goldberg (2014c) show the theoretical equivalence of CSG and PPMI matrix factorization. Levy et al. (2015) further discuss the connections between 4 word embedding models (PPMI, SVD, CSG, GloVe) and re-evaluate them with the same hyper-parameters. Suzuki and Nagata (2015) investigate different configurations of CSG and GloVe and merge them together. Yin and Schutze (2016) propose 4 ensemble methods and show their effectiveness over individual ones.

There is also research evaluating different context types in learning word embeddings. Heylen et al. (2008) compares dependency-based and linear vector space model for finding semantically related nouns in Dutch. Vulic and Korhonen (2016) compare CSG and dependency-based models on various languages. Their results suggest that dependency-based models are better at detecting functional similarity in English, although that does not necessarily hold for other languages. Bansal et al. (2014) show that DEPS context is preferable to linear context on parsing task. Melamud et al. (2016) investigate the performance of CSG, DEPS and a substitute-based word embedding model (Yatbaz et al., 2012)[3], which shows that different types of intrinsic tasks have clear preference for particular types of contexts. On the other hand, for extrinsic tasks, the optimal context types need to be carefully tuned on specific dataset.

The contribution of this study is that in addition to linear and dependency-based context we also consider bound and unbound context representations, as will be described below. Furthermore, we systematically evaluate three word embedding models: CSG, CBOW and GLoVe.

[2] In these two papers, the description of position-aware (bound) context are quite different. However, their ideas are actually identical.

[3] We do not consider this type of context, since in our pilot studies it performed consistently worse than the other two context types. The same observation is also made by Melamud et al. (2016); Vulic and Korhonen (2016).

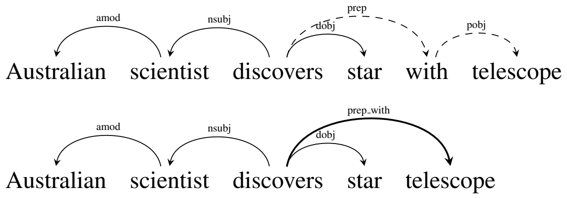

amod
nsubj
prep
dobj
pobj
Australian scientist discovers star with telescope
amod
nsubj
prep_with
dobj
Australian scientist discovers star telescope

Figure 1: Illustration of dependency parse tree.

3 Word Embeddings Models

In this section, we first introduce different contexts in detail, and discuss their strengths and weaknesses. We then show how CSG, CBOW and GloVe can be generalized to use these contexts.

3.1 Context Types

There are many different types of context, both on document and sentence level. For syntactic contexts, the current literature discusses mainly the linear (used in most word embedding models) and dependency-based contexts (DEPS (Levy and Goldberg, 2014a)). Linear context is defined as the positional neighbors of the target word in texts. DEPS context is defined as the syntactic neighbors of the target word based on dependency parse tree, as shown in Figure 1[4].

Compared to the linear context, DEPS context can capture more relevant words that are further away from the target word in the text. For example in Figure 1, linear context does not include the word-context pair "discovers telescope", while DEPS context contains this information. DEPS context can also exclude some uninformative word-context pairs like "with star" and "telescope with".

Note that dependency parsing is time-consuming. Despite its parallelizability, our implementation still takes nearly a month to finish dependency parsing for the Wikipedia corpus on a 32-core machine. it is only fair to compare linear and DEPS context if we ignore the time complexity. it is also worth noting that part-of-speech labels are required when performing dependency parsing.

3.2 Context Representations

In the original CSG, CBOW and GloVe models, contexts are represented by words without any additional information. Ling et al. (2015) modify CSG and CBOW by introducing position-bound words, where each contextual word is associated with their relative position to the target word. This allows CSG and CBOW to distinguish different sequential positions and capture the structural information from the context. We refer to methods that bind positional information with the contextual word as bound (context) representation, as opposed to unbound (context) representation where contextual words are treated the same irrespective of their positions with regards to the target word.

Context Type / Context Representation	Linear	DEPS
unbound	australian, scientist, star, with	scientist, star, telescope
bound	australian/-2, scientist/-1, star/+1, with/+2	scientist/nsubj, star/dobj, telescope/prep_with

Table 2: Illustration of bound and unbound representations under linear and DEPS context types. This example is based on Figure 1, and the target word is "discovers".

The original DEPS uses "bound" representation by default: each word is associated with its dependency relation to the target word. In this paper, we also investigate the simpler context representation where no dependency relation is associated with a word. This enables a fair comparison with conventional models like CSG, CBOW and GloVe, since they do not use bound representation either. An example of different syntactic context types and context representations is shown in Table 2.

Intuitively, bound representation should work better than unbound representation, since it uses information about relative word positions. However, this is not always the case in practice. An obvious drawback is that bound representation is more sparse than unbound representation, especially for DEPS context type. In our data, there were 47 dependency relations in dependency parse tree. Although not every combination of dependency relations and words appear in the word-context pair collection, in practice it still enlarges the contextual words' vocabulary about 5 times.

Both syntactic context types (linear and DEPS) and the choice of context representations (bound and unbound) have a dramatic effect on the word embeddings. Bound linear representation transfers each contextual word into a new one, and the

[4] This example is from Levy and Goldberg (2014a)

	Linear (window size 1)	DEPS
P	(**australian**, scientist) (**scientist**, australian) (**scientist**, discovers) (**discovers**, scientist) (**discovers**, star) ...	(**australian**, scientist) (**scientist**, australian) (**scientist**, discovers) (**discovers**, scientist) (**discovers**, star) (**discovers**, telescope) ...
M	(**australian**, scientist) (**scientist**, australian, discovers) (**discovers**, scientist, star) ...	(**australian**, scientist) (**scientist**, australian, discovers) (**discovers**, scientist, star, telescope) ...
$\overline{M}$	(**australian**, scientist, 1) (**scientist**, australian, 1) (**scientist**, discovers, 1) (**discovers**, scientist, 1) (**discovers**, star, 1) ...	(**australian**, scientist, 1) (**scientist**, australian, 1) (**scientist**, discovers, 1) (**discovers**, scientist, 1) (**discovers**, star, 1) (**discovers**, telescope, 1) ...

Table 3: Illustration of collection P, M and $\overline{M}$ for sentence "australian scientist discovers star with telescope". Unbound representation is used in this example. Words in the collections are **Bold**. .

word-context pairs are changed completely. DEPS, as compared to the linear contexts, increases the likelihood that the contextual words are in a meaningful relation with the target word, although some words captured by DEPS would also be found in the linear contexts if the window is wide enough. For example, in Table 2, "scientist" and "star" are considered as the contextual words of "discovers" in both linear and DEPS context types.

3.3 Generalization

Let P be a collection of word-context pairs. P can be merged based on the words to form a collection M with size of $|V|$, where V is the vocabulary. Each element $(w, c_1, c_2, .., c_{n_w}) \in M$ is word w and its contexts, where n_w is the number of word w's contexts. P can also be merged based on both words and contexts to form a collection $\overline{M}$. Each element $(w, c, \#(w, c)) \in \overline{M}$ is the word w, context c, and the times they appear in collection P. An example of these collections is shown in Table 3.

3.3.1 Generalized Bag-Of-Words

The objective function of Generalized Bag-Of-Words (GBOW) is defined as:

$$\sum_{(w,c_1,..,c_{n_w})\in M} \log p\left(w \middle| \sum_{i=1}^{n_w} \vec{c_i}\right) \tag{1}$$

With negative sampling technique, the log probability is calculated by:

$$\log \sigma\left(\vec{w} \cdot \sum_{i=1}^{n_w} \vec{c_i}\right) - \sum_{k=1}^{K} \log \sigma\left(\vec{w_{N_k}} \cdot \sum_{1=i}^{n_w} \vec{c_i}\right) \tag{2}$$

where σ is the sigmoid function, K is the negative sampling size, $\vec{w}$ and $\vec{c}$ is the vector for word w and c respectively. The negatively sampled word w_{N_k} is randomly selected on the basis of its unigram distribution $(\frac{\#(w)}{\sum_w \#(w)})^{ds}$, where $\#(w)$ is the number of times that word w appears in the corpus, and ds is the distribution smoothing hyper-parameter which is usually defined as 0.75.

Note that with negative sampling technique, both GBOW and original CBOW (Mikolov et al., 2013a) will learn two sets of embeddings (word embeddings and context embeddings). In the original CBOW, the context embeddings can also be considered as word embeddings, since the vocabulary set of words and contexts are the same. However, for bound context, the words (i.e. scientist) and contexts (i.e. scientist/nsubj) are quite different. It is necessary to distinguish conditioned and conditioning variables. For example, in Figure 1, the context "scientist/nsubj" can only be predicted by word "discovers". However, most of the word is connected to several contextual words. Due to this, the sum of contextual word embeddings should be used for predicting the target word.

3.3.2 Generalized Skip-Gram

For generalized Skip-Gram (GSG), the definition is more straightforward and the objective function actually needs no specification (Levy and Goldberg, 2014b). Nonetheless, in order to make it consistent with our GBOW, we also specify the conditioned and conditioning variables in the objective function:

$$\begin{aligned} &\sum_{(w,c)\in P} \log p\left(w|\vec{c}\right) \\ &= \sum_{(w,c)\in P} \left[\log \sigma\left(\vec{w} \cdot \vec{c}\right) - \sum_{k=1}^{K} \log \sigma\left(\vec{w_{N_k}} \cdot \vec{c}\right)\right] \end{aligned} \tag{3}$$

Note that this generalization does not change the nature of the models for linear context. In our pilot experiments on word analogy and word similarity, the performance of both GSG and GBOW is almost identical to their original versions.

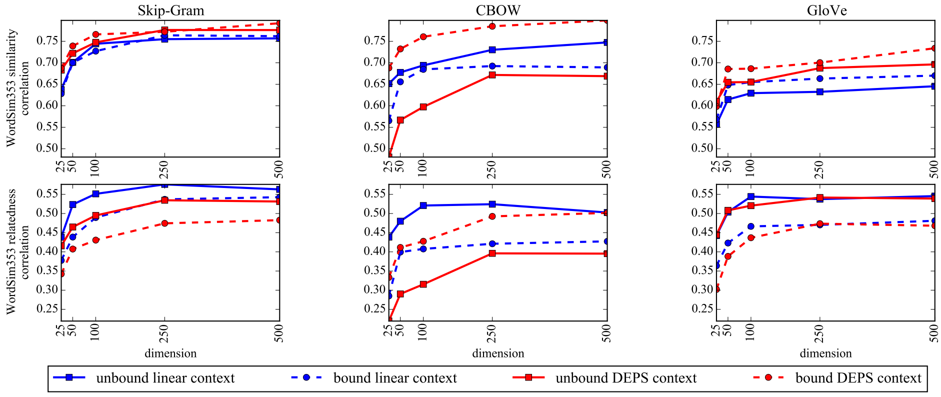


Figure 2: Correlation results for similarity and relatedness categories on WordSim353 (word similarity) dataset.

3.3.3 GloVe

Unlike GSG and GBOW, GloVe explicitly optimizes a log-bilinear regression model based on word co-occurrence matrix. Since GloVe is already a very generalized model, with the previous defined collection $\overline{M}$, the final objective function is written as:

$$\sum_{(w,c)\in\overline{M}} f(\#(w,c))(\vec{w}\cdot\vec{c}+\vec{b_w}+\vec{b_c}-\log\#(w,c)) \quad (4)$$

where $\vec{b_w}$ and $\vec{b_c}$ are biases for word and context. f is a non-decreasing weighting function and ensures that large $\#(w,c)$ is not over-weighted.

Note that the inputs of GSG, GBOW and Glove are the collections P, M and $\overline{M}$ respectively. Once the corpus and hyper-parameters are fixed, these collections (and thus the learned word embeddings) are determined only by the choice of context types and representations.

4 Experiments

We evaluate the effectiveness of different syntactic context types and context representations on word similarity, word analogy, part-of-speech tagging, chunking, named entity recognition, and text classification tasks. In this section we describe our models, and then report and discuss the experimental results on each task.

4.1 Word Embeddings

Previously, the `word2vecf` toolkit [5] (Levy et al., 2015) extended the `word2vec` toolkit [6] (Mikolov et al., 2013b) to accept the input of collection P rather than raw corpus. This makes CSG model accept arbitrary contexts (e.g. DEPS context). However, CBOW and GloVe are not considered in that toolkit. We implement `word2vecPM` toolkit, a further extension of `word2vecf`, which supports generalized CSG, CBOW and GloVe with the input of collection P, M and $\overline{M}$ respectively. For fair comparison, as suggested by Levy et al. (2015), we use the same hyper-parameters [7] for all embedding models. English Wikipedia (August 2013 dump) is used as the training corpus. The Stanford CoreNLP (Manning et al., 2014) is used for dependency parsing. After parsing, tokens are converted to lowercase. Words and contexts that appear fewer than 100 times in the collection P are ignored.

4.2 Word Similarity Task

Word similarity task aims at producing semantic similarity scores of word pairs, which are compared with the human scores using Spearman's correlation. The cosine distance is used for generating similarity scores between two word vectors. We use the WordSim353 (Finkelstein et al., 2001) dataset, divided into similarity and relatedness categories (Zesch et al., 2008; Agirre et al., 2009).

Previous research (Levy and Goldberg, 2014a; Melamud et al., 2016) concluded that compared to linear context, DEPS context can capture more functional similarity (e.g. tiger/cat) rather than topical similarity (relatedness) (e.g. tiger/jungle). However, their experiments do not distinguish the

[5] https://bitbucket.org/yoavgo/word2vecf

[6] http://code.google.com/p/word2vec/

[7] Negative sampling size is set to 5 for SG and 2 for CBOW. Distribution smoothing is set to 0.75. No dynamic context or "dirty" sub-sampling is used. The window size is fixed to 2. The number of iterations is set to 2, 5 and 30 for SG, CBOW and GloVe respectively.

Model	Context Type	Context Representation	Similarity			Relatedness	Similarity+Relatedness	
			WS353	Rare Words	SimLex-999	WS353	MEN	Mech Turk
GSG	linear	unbound	.757	.414	.417	**.563**	**.732**	.632
		bound	.762	.421	**.434**	.543	.695	.608
	dep	unbound	.776	**.422**	.418	.531	.728	**.644**
		bound	**.792**	.413	.421	.483	.674	.643
GBOW	linear	unbound	.747	**.436**	**.439**	**.503**	**.718**	**.644**
		bound	.689	.403	.428	.427	.659	.512
	dep	word	.669	.412	.386	.395	.667	.541
		bound	**.799**	.434	.403	.502	.640	.587
GloVe	linear	unbound	.645	.354	.323	**.545**	.662	.587
		bound	.670	.400	.363	.481	.563	.587
	dep	unbound	.696	.371	.342	.539	**.692**	**.603**
		bound	**.734**	**.409**	**.406**	.468	.541	.557

Table 4: Numerical results on word similarity datasets. Best results in group are marked **Bold**.

Model	Context Type	Context Representation	Google Sem	Google Syn	MSR	Inflectional morphology	Derivational morphology	Encyclopedia	Lexicography
GSG	linear	unbound	.708	.639	.642	.678	.110	.242	.083
		bound	.702	.454	**.653**	.668	.111	.208	**.099**
	dep	unbound	**.716**	**.661**	.644	**.691**	**.122**	**.253**	.095
		bound	.600	.307	.600	.668	.112	.170	**.099**
GBOW	linear	unbound	**.628**	**.566**	**.601**	**.618**	**.096**	.201	.074
		bound	.602	.376	.569	.572	.091	.157	**.081**
	dep	unbound	.573	.553	.520	.496	.094	**.216**	.076
		bound	.495	.248	.516	.563	.086	.126	.078
GloVe	linear	unbound	.471	**.719**	.454	.425	.033	.226	.054
		bound	.502	.218	**.542**	**.559**	**.044**	.129	**.095**
	dep	unbound	**.513**	.700	.525	.491	.043	**.227**	.063
		bound	.402	.121	.525	.446	.033	.093	.083

Table 5: Numerical results on word analogy datasets. Best results in group are marked **Bold**.

effect of different context representations: unbound representation is used for linear context (Mikolov et al., 2013b), while bound representation is used for dependency-based context (Levy and Goldberg, 2014a). Moreover, only CSG model is considered.

We revisit those claims with more systematical experiments. As shown in the top-left sub-figure of Figure 2, DEPS does outperform the linear contextin GSG and GloVe in the similarity section of WordSim353, confirming its ability to capture functional similarity. However, the advantage of DEPS does not fully transfer to GBOW. Although bound DEPS context for GBOW is still the best performer, unbound DEPS context performs the worst, which shows the importance of bound vs unbound representation.

Note that the results are also reversed on WordSim353 relatedness section (the right subfigure of Figure 2), which shows that linear context is more suitable for capturing topical similarity.

Overall, DEPS context type does not get all the credit for capturing functional similarity. Context representations play an important role for word similarity task. it is only safe to say that DEPS context captures functional similarity with the "help" of bound representation. In contrast, linear context type captures topical similarity with the "help" of unbound representation.

However, the above findings come with a major caveat: a lot seems to depend on the particular dataset, in addition to the model and context type. We experimented with MEN dataset (Bruni et al., 2012), Mechanical Turk dataset (Radinsky et al., 2011), Rare Words dataset (Luong et al., 2013), and SimLex-999 dataset (Hill et al., 2016) (Table 4), and we were not able to observe uniform trends even for datasets that are supposed to capture the same relation - like the similarity part of WordSim353, Rare Words and SimLex.

Still, some models do favor a certain context type for both similarity and relatedness: e.g. GBOW favors linear unbound contexts, while GLoVE in most cases prefers DEPS over the linear context. In case of GCG, however, context type needs to be optimized for the particular dataset.

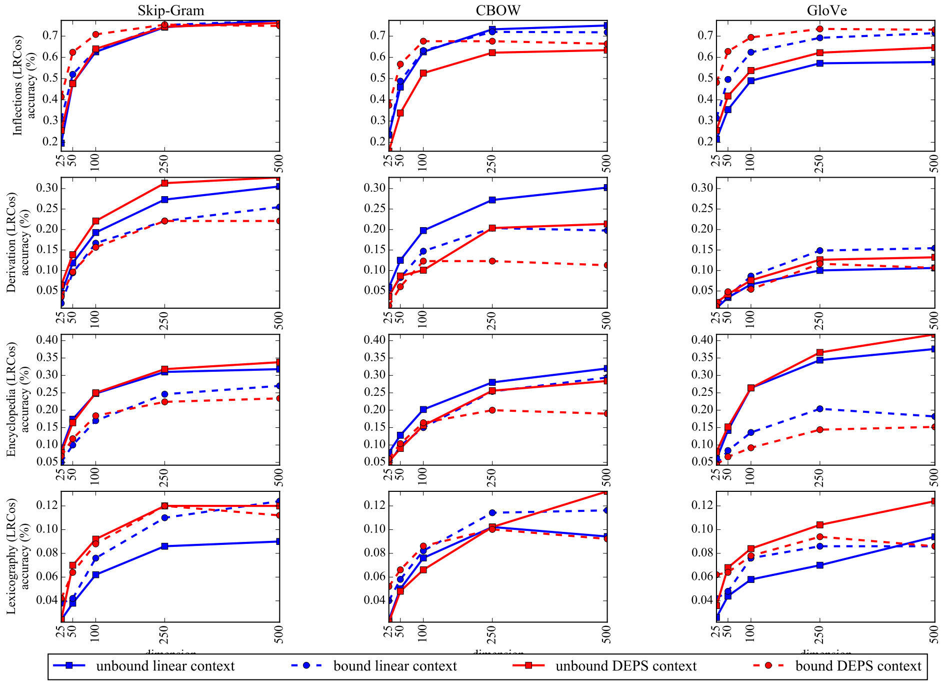


Figure 3: Averaged accuracy results for all Inflections, Derivation, Encyclopedia and Lexicography categories on BATS word analogy dataset.

4.3 Word Analogy Task

Word analogy task aims at answering the questions like "a is to a' as b is to __ ?", such as "London is to Britain as Tokyo is to Japan". We follow the evaluation protocol in Levy and Goldberg (2014b), which answers the questions using LRCos method (Drozd et al., 2016). LRCos shows significant improvement over the traditional vector offset method. We use BATS analogy dataset (Gladkova et al., 2016) in our experiments.

As shown in Figure 3, context representation plays an important role in word analogy task. The choice of context representation (bound or unbound) actually has much larger impact than the choice of context type (linear or DEPS). The results on Encyclopedia category are perhaps the most evident. The performance of unbound linear context and unbound DEPS context is similar. However, for most models and categories, bound representation seems to outperform unbound representation. When bound representation is used, the performance drops around $5 - 15$ percent for DEPS context in terms of accuracy. This is consistent with the findings of Levy and Goldberg (2014a), who report that DEPS context did not work well for the analogy task.

As shown in Table 5, we have also experimented on two much smaller datasets: MSR analogy dataset (Mikolov et al., 2013c), and Google analogy dataset (Mikolov et al., 2013a) (with semantic and syntactic questions). They also show that the choice of context representation has more impact than the choice of context type.

4.4 POS, Chunking and NER Tasks

Although intrinsic evaluations like word similarity and word analogy tasks could provide direct insights about different context types and representations, they have certain methodological problems (Gladkova and Drozd, 2016), and the experimental results above cannot be directly translated to the typical uses of word embeddings in downstream tasks (Schnabel et al., 2015; Linzen, 2016; Chiu et al., 2016). Thus extrinsic tasks should also be considered.

In this subsection, we evaluate the effectiveness of different word embedding models with different contexts on Part-of-Speech Tagging (POS), Chunking[8] and Named Entity Recognition (NER) tasks [9]. For these tasks, a NLP system assigns labels to elements of texts. Note that in practice, one should NOT use DEPS context for POS-tagging and chunking tasks, since their labels are used in

[8] CoNLL 2000 shared task http://www.cnts.ua.ac.be/conll2000/chunking

[9] CoNLL 2003 shared task http://www.cnts.ua.ac.be/conll2003/ner

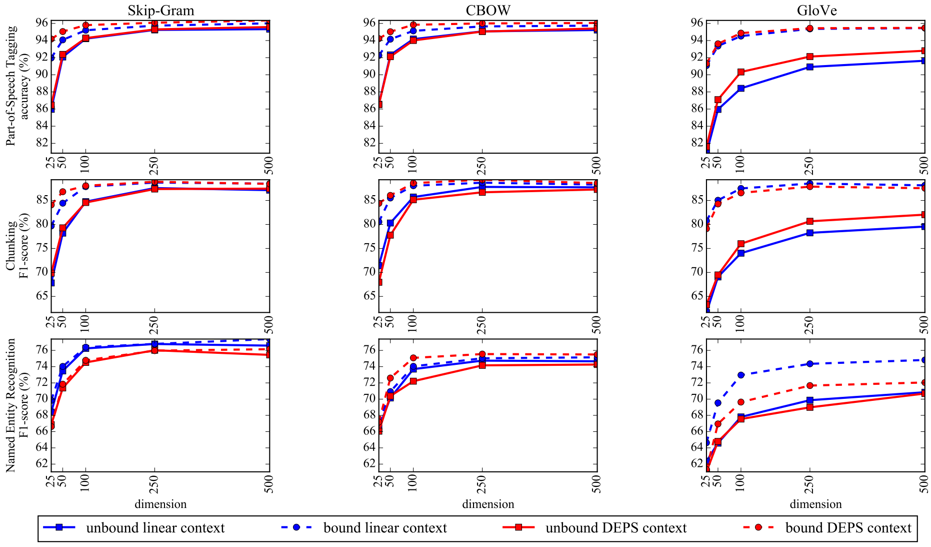


Figure 4: Accuracy or F_1-score results on Part-of-Speech Tagging, Chunking and Named Entity Recognition tasks.

parsing the source corpus.

Following the evaluation protocol used in Kiros et al. (2015), we restrict the predicting model to Logistic Regression Classifier[10]. The classifier's input for predicting the label of word w_i is simply the concatenation of word vectors $\vec{w_{i-2}}$, $\vec{w_{i-1}}$, $\vec{w_i}$, $\vec{w_{i+1}}$, $\vec{w_{i+2}}$. This ensures that the quality of embedding models is directly evaluated, and their strengths and weaknesses are easily observed.

Model	Context Type	Context Representation	POS	Chunking	NER
GSG	linear	unbound	95.3	87.2	76.6
		bound	96.0	**88.5**	**77.4**
	dep	unbound	95.6	87.5	75.5
		bound	**96.3**	**88.5**	76.2
GBOW	linear	unbound	95.2	87.7	74.7
		bound	95.7	88.3	75.2
	dep	unbound	95.4	87.3	74.3
		bound	**96.0**	**88.6**	**75.5**
GloVe	linear	unbound	91.6	79.6	70.8
		bound	**95.5**	**88.2**	**74.8**
	dep	unbound	92.8	82.0	70.7
		bound	**95.5**	87.5	72.0

Table 6: Numerical results on Part-of-Speech Tagging, Chunking and Named Entity Recognition tasks. Best results in group are marked **Bold**.

As shown in Figure 4 and Table 6, GSG, GBOW and GloVe exhibit overall similar trends. When the same context type is used, bound representation outperforms unbound representation on all tasks. Sequence labeling tasks are not sensitive to syntax. For bound representation, the ignorance of syntax becomes beneficial, since it decreases the amount of noise and sparsity.

Moreover, DEPS context type works slightly better than linear context type in most cases. These results suggest that unbound linear context (as in traditional CSG and CBOW) may not be the best choice of input word vectors for sequence labeling. Bound representations should always be used and DEPS context type is also worth considering. Again, similar to the word analogy task, GloVe is more sensitive to different context representations than Skip-Gram and CBOW.

4.5 Text Classification Task

Finally, we evaluate the effectiveness of different word embedding models with different syntactic contexts on text classification task. Text classification is one of the most popular and well-studied tasks in natural language processing. Recently, deep neural networks achieve state-of-the-art results on this task (Socher et al., 2013; Kim, 2014; Dai and Le, 2015). They often need pre-trained word embeddings as inputs to improve their performances. Similarly to the previous evaluation of sequence labeling tasks, instead of building complex deep neural networks, we use a simpler classification method called Neural Bag-of-Words (Li et al., 2017) to directly evaluate the word embeddings: texts are first represented by the sum of their word vectors, then a Logistic Regression Classifier (the same as that in previous subsection)

[10] The implementation by scikit is used http://scikit-learn.org/

Model	Context Type	Context Rep.	Sentence-level MR	CR	Subj	Document-level RT-2k	IMDB
GSG	linear	unbound	**76.1**	78.3	**90.9**	83.5	**85.2**
		bound	75.3	**79.0**	90.4	82.2	**85.2**
	dep	unbound	76.0	77.7	90.7	**84.8**	85.1
		bound	75.0	77.5	90.0	84.7	84.5
GBOW	linear	unbound	74.9	77.9	**90.4**	82.0	**85.0**
		bound	74.1	77.8	90.3	80.7	84.1
	dep	unbound	**75.0**	77.6	90.1	**82.4**	84.9
		bound	73.5	**78.2**	89.9	80.7	83.4
GloVe	linear	unbound	73.4	76.7	89.6	79.2	**83.5**
		bound	73.2	77.5	**90.0**	79.8	83.4
	dep	unbound	**74.0**	**77.7**	89.5	**81.3**	**83.5**
		bound	72.5	76.7	88.8	79.2	**83.5**
random word embeddings			63.9	72.8	79.9	72.2	77.2

Table 7: Accuracy results on 5 text classification datasets. Best results in group are **Bold**

is built upon these text representations for classification.

Different word embedding models are evaluated on 5 text classification datasets. The first 3 datasets are sentence-level: short movie review sentiment (MR) (Pang and Lee, 2005), customer product reviews (CR) (Nakagawa et al., 2010), and subjectivity/objectivity classification (SUBJ) (Pang and Lee, 2004). The other 2 datasets are document-level with multiple sentences: full-length movie review (RT-2k) (Pang and Lee, 2004), and IMDB movie review (IMDB) (Maas et al., 2011)[11].

As shown in Table 7, pre-trained word embeddings outperform random word embeddings by a large margin. This strengthens the previous claim that pre-trained word embeddings are highly useful for text classification (Iyyer et al., 2015; Li et al., 2017). Unlike in the other tasks, in text classification all models exhibit similar performance. Text classification has less focus on syntax and function similarity. Because of that, models with bound representation perform worse than those with unbound representation on almost all datasets except CR. Models with DEPS context type and linear context type are comparable. These observations suggest that simple unbound linear context type (as in traditional CSG and CBOW) is still the best choice of pre-training word embeddings for text classification, which is already used in most studies.

5 Conclusion

This paper provides a first systematical investigation of different syntactic context types (linear vs dependency-based) and different context representations (bound vs unbound) for learning word embeddings. We evaluate GSG, GBOW and GloVe models on intrinsic property analysis tasks (word similarity and word analogy), sequence labeling tasks (POS, Chunking and NER) and text classification task.

We find that most tasks have clear preference for different context types and representations. Context representation plays a more important role than context type for learning word embeddings. Only with the "help" of bound representation does DEPS context capture functional similarity. Word analogies seem to prefer unbound representation, although performance varies by question type No matter which syntactic context type is used, bound representation is essential for sequence labeling tasks, which benefits from its ability of capturing functional similarity. GSG with unbound linear context is still the best choice for text classification task. Linear context is sufficient for capturing topical similarity compared to more labor-intensive DEPS context. Words' position information is generally useless for text classification, which makes bound representation contribute less to this task.

In the spirit of transparent and reproducible experiments, the `word2vecPM` toolkit [12] is published along with this paper. We hope researchers will take advantage of the code for further improvements and applications to other tasks.

Acknowledgments

We thank the authors of the `word2vecf` toolkit and the accompanied evaluation scripts (Levy et al., 2015). Their work systematically investigated the effects of different hyper-parameters on various word embedding models, which directly influenced us a lot.

This work is supported by National Natural Science Foundation of China with grant No. 61472428, the Fundamental Research Funds for the Central Universities, the Research Funds of Renmin University of China No. 14XNLQ06. This work is partially supported by ECNU-RUC-InfoSys Joint Data Science Lab and CCF-Tencent Open Research Fund (RAGR20160102).

[11] Please see Wang and Manning (2012) for more detailed introduction and pre-processing of these datasets.

[12] The current version can be found at `https://github.com/libofang/word2vecPM`. The entire code is also planned to be re-written in Python and integrated into VSMlib (`http://vsm.blackbird.pw/`) for easier use.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *NAACL*, pages 19–27. Association for Computational Linguistics.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *ACL*, pages 809–815.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *ACL*, pages 136–145. Association for Computational Linguistics.

Billy Chiu, Anna Korhonen, and Sampo Pyysalo. 2016. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *ACL*, pages 406–414.

Stephen Clark. 2012. Vector space models of lexical meaning. In *Handbook ofContemporary Semantics second edition.Wiley-Blackwell*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167. ACM.

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

James Richard Curran. 2004. *From distributional to semantic similarity*. Ph.D. thesis, University of Edinburgh, UK.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *NIPS*, pages 3079–3087.

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

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *WWW*, pages 406–414. ACM.

Anna Gladkova and Aleksandr Drozd. 2016. Intrinsic evaluations of word embeddings: What can we do better? In *Proceedings of The 1st Workshop on Evaluating Vector Space Representations for NLP*, pages 36–42, Berlin, Germany. ACL.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. 2016. Analogy-based detection of morphological and semantic relations with word embeddings: What works and what doesnt. In *Proceedings of naacl-hlt*, pages 8–15.

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

Kris Heylen, Yves Peirsman, Dirk Geeraerts, and Dirk Speelman. 2008. Modelling word similarity: an evaluation of automatic synonymy extraction algorithms. In *LREC*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*, pages 1681–1691.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*, pages 3294–3302.

Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. 2016. How to generate a good word embedding. *IEEE Intelligent Systems*, 31:5–14.

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

Omer Levy and Yoav Goldberg. 2014b. Linguistic regularities in sparse and explicit word representations. In *CoNLL*, pages 171–180.

Omer Levy and Yoav Goldberg. 2014c. Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185.

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

Bofang Li, Tao Liu, Zhe Zhao, Puwei Wang, and Xiaoyong Du. 2017. Neural bag-of-n-grams. In *AAAI*, pages 3067–3074.

Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *HLT-NAACL*, pages 1299–1304.

Tal Linzen. 2016. Issues in evaluating semantic spaces using word analogies. *CoRR*, abs/1606.07736.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *ACL*, pages 142–150.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*, pages 55–60.

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

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, volume 13, pages 746–751.

Andriy Mnih and Geoffrey E. Hinton. 2007. Three new graphical models for statistical language modelling. In *ICML*, pages 641–648.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *NAACL*, pages 786–794. Association for Computational Linguistics.

Neha Nayak and Christopher D. Manning. 2016. Evaluating word embeddings using a representative suite of practical tasks. In *ACL workshop*.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33:161–199.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*, pages 271–278. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124. Association for Computational Linguistics.

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

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM.

Tobias Schnabel, Igor Labutov, David M. Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *EMNLP*, pages 649–657.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, volume 1631, page 1642. Citeseer.

Jun Suzuki and Masaaki Nagata. 2015. A unified learning framework of skip-grams and global vectors. In *ACL*, page 186.

Ivan Vulic and Anna Korhonen. 2016. Is "universal syntax" universally useful for learning distributed word representations? In *ACL*, page 518.

Sida I. Wang and Christopher D. Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, pages 90–94.

Mehmet Ali Yatbaz, Enis Sert, and Deniz Yuret. 2012. Learning syntactic categories using paradigmatic representations of word context. In *EMNLP-CoNLL*, pages 940–951.

Wenpeng Yin and Hinrich Schutze. 2016. Learning word meta-embeddings. In *ACL*, pages 327–332.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Using wiktionary for computing semantic relatedness. In *AAAI*, volume 8, pages 861–866.

Does syntax help discourse segmentation? Not so much

Chloé Braud and **Ophélie Lacroix** and **Anders Søgaard**
CoAStaL DIKU
University of Copenhagen
University Park 5, 2100 Copenhagen
chloe.braud@gmail.com lacroix@di.ku.dk soegaard@di.ku.dk

Abstract

Discourse segmentation is the first step in building discourse parsers. Most work on discourse segmentation does not scale to real-world discourse parsing across languages, for two reasons: (i) models rely on constituent trees, and (ii) experiments have relied on gold standard identification of sentence and token boundaries. We therefore investigate to what extent constituents can be replaced with universal dependencies, or left out completely, as well as how state-of-the-art segmenters fare in the absence of sentence boundaries. Our results show that dependency information is less useful than expected, but we provide a fully scalable, robust model that only relies on part-of-speech information, and show that it performs well across languages in the absence of any gold-standard annotation.

1 Introduction

Discourse segmentation is the task of identifying, in a document, the minimal units of text – called Elementary Discourse Units (EDU) (Carlson et al., 2001) – that will be then linked by semantico-pragmatic relations – called discourse relations. Discourse segmentation is the first step when building a discourse parser, and has a large impact on the building of the final structure – predicted segmentation leads to a drop in performance of about 12-14% (Joty et al., 2015).

In this work, we focus on the Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) in which discourse analysis is a tree covering an entire document. Most of the recent discourse parsers have been developed within this framework, making crucial the development of robust RST discourse segmenters. Many corpora have been annotated within this framework for several domains and languages – such as English with the RST Discourse Treebank (RST-DT) (Carlson et al., 2001), but also Spanish (da Cunha et al., 2011), Brazilian Portuguese (Cardoso et al., 2011; Collovini et al., 2007; Pardo and Seno, 2005; Pardo and Nunes, 2004) or German (Stede and Neumann, 2014).

State-of-the-art performance for discourse segmentation on the RST-DT is about 94% in F_1 (Xuan Bach et al., 2012). Most work on discourse parsing has focused on English and on the RST-DT (Ji and Eisenstein, 2014; Feng and Hirst, 2014; Li et al., 2014; Joty et al., 2013), and so discourse segmentation (Xuan Bach et al., 2012; Fisher and Roark, 2007; Subba and Di Eugenio, 2007). And while discourse parsing is a document level task, discourse segmentation is done at the sentence level, assuming that sentence boundaries are known. This prevents from using discourse information for a wider range of downstream tasks.

Moreover, while discourse parsing is a semantic task involving a large range of information, the annotation guidelines reflect that segmentation is merely based on syntax: in practice, an EDU can not overlap sentence boundaries – while some discourse trees can cross the sentence boundaries (van der Vliet and Redeker, 2011) –, and deciding whether a clause is an EDU in the RST-DT strongly depends on its syntactic function – e.g. "Clauses that are subjects or objects of a main verb are not treated as EDUs" (Carlson and Marcu, 2001). Consequently, existing discourse segmenters heavily rely on information derived from constituent trees usually following the Penn Treebank (PTB) (Marcus et al., 1993) guidelines. Nevertheless constituent trees are not easily available for any language. Finally, even for English, using predicted trees leads to a large drop in per-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2432–2442
Copenhagen, Denmark, September 7–11, 2017.

formance for discourse segmentation.

Recently, Braud et al. (2017) proposed the first cross-lingual and cross-domain experiments for discourse segmentation, relying only on words and Part-of-Speech (POS) tags (morpho-syntactic level). However, they focus on document-level discourse segmentation – preventing from a comparison with previous work –, and they did not include any syntactic information. In this paper, we significantly extend their work by investigating the use of syntactic information, reporting results with various sets of features at the sentence level – varying the settings between gold and predicted, and fine-grained vs coarse grained information –, and studying the impact of tokenisation.

Our contributions

- We develop new discourse segmenters that can be used for many languages and domains since they rely on easily available resources;
- We investigate the usefulness of syntactic information when derived from Universal Dependencies (UD) (Nivre et al., 2016) parse trees, compare it to simpler representations and show that accurate POS tags are better than low quality parse trees;
- We compare different settings considering gold and predicted POS tags, tokenization and sentence segmentation.

2 Related work

First discourse segmenters on the RST-DT were based on hand-crafted rules, relying on punctuation, POS tags, discourse cues (e.g. "but", "because", "after") and syntactic information (Le Thanh et al., 2004; Tofiloski et al., 2009). Segmenters based on handwritten rules have also been developed for Brazilian Portuguese (Pardo and Nunes, 2008) (51.3% to 56.8%, depending on the genre), for Spanish (da Cunha et al., 2010, 2012) (80%) and for Dutch (van der Vliet, 2010) (73% with automatic parse, 82% with gold parse).[1]

More recent discourse segmenters on the RST-DT are based on binary classifiers at the word level (Soricut and Marcu, 2003; Fisher and Roark, 2007; Joty et al., 2015), possibly using a neural network architecture (Subba and Di Eugenio, 2007). Joty et al. (2015) also report results for the Instructional corpus (Subba and Di Eugenio, 2009) (F_1 80.9% on 10-fold).

Interestingly, Fisher and Roark (2007) investigate the utility of parse-derived features for the task. More precisely, they compare different sets of features derived from constituent trees, using n-grams or paths in a tree that could be a full constituent tree or a shallow parse (chunks). Their system thus requires chunker or constituent parser. In contrast, we investigate the usefulness of syntactic information derived from dependency parses, and we extend their work in also comparing our results to the use of only POS tags and words.

For English RST-DT, the best discourse segmentation results were presented in Xuan Bach et al. (2012) (F_1 91.0% with automatic parse, 93.7 with gold parse). They cast discourse segmentation as a sequence labeling problem, as also done in (Sporleder and Lapata, 2005; Hernault et al., 2010). More precisely, they develop a base system using CRF on top of which they add a reranking model. Their base system relies on lexico-syntactic features including words, POS tags – from the Penn Treebank (PTB) annotation scheme –, and paths in the constituent trees. The reranking systems then considers subtrees features, corresponding to the boundaries of a candidate EDU and the common boundary of two consecutive candidates EDUs. This post-processing only leads to small improvements: about 1.2% when using gold syntactic information, and only 0.3% with predicted trees.

All these systems rely on a quite large range of lexical and syntactic features (e.g. token, POS tags, chunks, lexicalized production rules, discourse connectives). Sporleder and Lapata (2005) present arguments for a knowledge-lean system that can be used for low-resourced languages. Their system, however, still relies on several tools and gold annotations (e.g. POS tagger, chunker, list of connectives, gold sentences). Moreover, previous work always rely on gold sentence boundaries, and only considers intra-sentential segment boundaries while sentence boundaries are not available for all languages.

Braud et al. (2017) recently proposed the first systems for discourse segmentation of entire documents directly applicable to low-resource languages. Their systems only rely on Universal De-

[1] For German, Sidarenka et al. (2015) propose a segmenter in clauses (that may be EDU or not).

pendencies POS tagging, for which models are available for many languages. As done in that study, we do sequence prediction using a neural network. However, we extend their work significantly in reporting results for intra-sentential segmentation, in comparing more settings concerning the availability of information (tokenisation, POS tags), and in including syntactic information into our systems.

3 Discourse segmentation

3.1 Binary task

Since the EDUs cover the documents entirely, discourse segmentation is generally cast as a binary task at the word level, where the goal is to find which word indicates an EDU boundary: A word is thus either beginning an EDU (label 'B'), or within an EDU (label 'I').

This design choice assumes that EDUs are adjacent spans of text, that is an EDU begins just after the end of the previous EDU. This is not entirely true in RST corpora, where embedded EDUs could break up another EDU, as in Example (1) taken from the RST-DT annotation manual (Carlson and Marcu, 2001). The units 1 and 3 form in fact one EDU, which is acknowledged by the annotation of a pseudo-relation SAME-UNIT between these segments.

(1) [But maintaining the key components (. . .)]$_1$
[– a stable exchange rate and high levels of imports –]$_2$
[will consume enormous amounts (. . .).]$_3$

We follow previous work on treating this as three segments, but note that this may not be the optimal solution. It introduces a bias: while most of the EDUs are full clauses, EDU 1 and 3 are fragments. Other designs are possible, especially a multi-label setting as done in (Afantenos et al., 2010) for a corpus annotated within the Segmented Discourse Representation Theory (Asher and Lascarides, 2003). While it seems relevant to deal with this issue during segmentation rather than using a pseudo-relation, it introduces new issues (i.e. the final structures are not trees anymore). We thus leave this question for future work.

3.2 Sentence *vs* document-level segmentation

Most of the existing work on discourse segmentation always assume a gold segmentation of the sentences: since an EDU boundary never crosses a sentence boundary,[2] these systems only perform intra-sentential segmentation. This is motivated by the quite high performance of sentence segmenters. In our experiments, we report intra-sentential results, in order to compare our systems to previous ones.

However, sentence boundaries are not always available. In a situation where both inter and intra-sentential segmentation is required, there are two alternatives: processing the tasks sequentially or simultaneously. In preliminary experiments we considered using the multilingual system UDPipe [3] (Straka et al., 2016) to segment document into sentences in an effort to use tools available in multiple languages. However, the segmentation is far from perfect: 7.5% of the words marked as beginning a sentence were not an EDU boundary in the RST-DT, thus corresponding to an error.

We thus rather decided to rely on a model performing both inter- and intra-sentential segmentation. We aim at building systems directly segmenting entire documents. Then in order to provide performance of discourse segmenters in a realistic setting, our final systems jointly predict sentence and intra-sentential EDU boundaries.

Finally, for the English RST-DT, we present two performance metrics:

- F_1 for intra-sentential boundaries only (see Section 7.1), in order to be comparable with state-of-the-art systems;
- and F_1 for all EDU boundaries, in order to set up a document-level baseline (see Section 7.2).

For the other languages and domains, since we do not have access to gold sentence boundaries, we only present results at the document level.

4 Approach

4.1 Neural network for sequence prediction

We model the task as a sequence prediction task using a neural network architecture. Our model consists of a stacked k-layer bi-LSTM, a variant of LSTM (Hochreiter and Schmidhuber, 1997) that

[2] With two exceptions in the RST DT, possibly due to errors in the discourse or syntactic annotation (Documents 2343 and 0678). As probably done in previous works, we do not consider these cases as separate sentences, following the discourse annotation.

[3] http://ufal.mff.cuni.cz/udpipe

reads the input in both regular and reversed order. This enables to take into account both left and right context (Graves and Schmidhuber, 2005). This is a crucial property for discourse segmentation, especially with the simplified representations we consider, since the decision depends on the context, e.g. coordinated NPs are not segmented while coordinated VPs are, our model must thus learn to distinguish a VP from a NP without using constituent parses.

The model takes as input a concatenation of randomly initialized and trainable embeddings of words and their morpho-syntactic features (see Section 4.3). The sequence goes through the k-stacked layers, and we output the concatenation of the backward and forward states. At the upper level, we compute the prediction using a Multi-Layer Perceptron. We used the Adam trainer. All other hyper parameters were tuned on development data; see Section 6.2 for a description of hyper-parameter tuning, and our final parameters. [4]

4.2 Tokenization and sentence splitting

In order to evaluate the impact of tokenization on discourse segmentation we propose two settings for English: one for which we evaluate on gold tokens – as done in all previous work –, and another one where tokenization is pre-processed using the UDPipe tokenizer. For the other languages, the task is always evaluated on non-gold tokens.

In the same way, we investigate the impact of gold sentence splitting by considering the traditional setting where discourse segmentation is only intra-sentential (gold sentences) and the more realistic one where we directly segment documents (sentence boundaries are unknown).

4.3 Features

To the best of our knowledge, we are the first to report the scores one can expect when not using syntactic trees and/or cue phrases, that is, only based on words or POS tags. These are interesting results, because they correspond to representations that can be built easily for any new language.

In addition, we investigate the impact of gold *vs* predicted features for discourse segmentation for English, and of automatic pre-processing of the data before feature extraction (tokenization). Until now, only the impact of using predicted constituent trees had been investigated. But since constituent treebanks are not readily available for many languages, we limit ourselves to (predicted) dependency trees.

Focusing on English allows us to set up a baseline using predicted feature information (document level) which could then be evaluated on other languages for which no gold features are available.

We evaluate both the performance when using single features and when combining the features described above, each corresponding to a (randomly intialized) real valued vector. The vectors for each features are concatenated to build a representation of a single word.

Lexical information Our first features are lexical. We use each token as a feature, being represented by a real-valued vector.

Morpho-syntactic features POS tags are also valuable information for the task, for example conjunctions and adverbs may often begin an EDU, because they can correspond to a discourse connective (e.g. “because”, “if”, “and”, “after”).

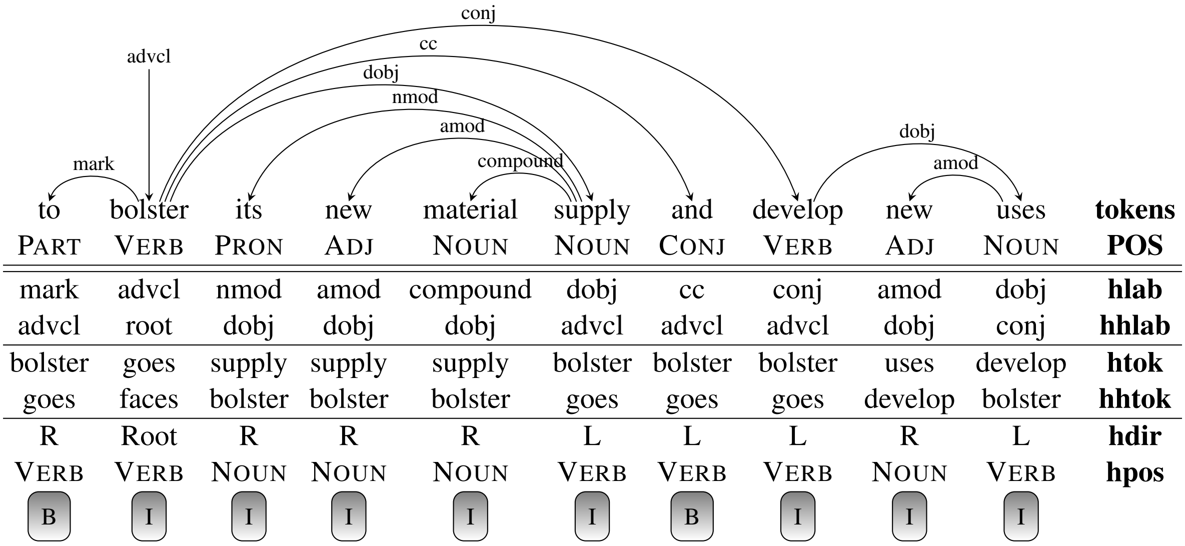
For English, we want to compare between gold and predicted information: gold PTB POS vs predicted PTB POS. For this last setting, we use predicted POS tag features for both training and testing our discourse segmenter in order to minimize the difference between training data and test data. We use our own POS-tagger,[5] which achieves 96.6% accuracy on test data, to predict the POS-tags. The test and development (discourse) data are tagged using a model trained on the entire training set, and the training data are tagged using a 10-fold cross-validation.

We also compare between scarce and available information (predicted setting): PTB POS (fine grained - 45 tags) vs UD POS (coarse grained - 17 tags). For predicting UD POS tags we make use of the UDPipe system (retrained on the v1.3 UD data).

Syntactic information We augment our representation with syntactic information available for many languages: supertags (STAGS) extracted from dependency parsed trees (predicted using UDpipe in the same setting as for POS-tags).

[4] Our system has been implemented with the Dynamic Neural Network Toolkit (Neubig et al., 2017).

[5] Bi-LSTM tagger (keras-based implementation) using non-supervised features about words (e.g. capitalization, suffixes).



to	bolster	its	new	material	supply	and	develop	new	uses	**tokens**
PART	VERB	PRON	ADJ	NOUN	NOUN	CONJ	VERB	ADJ	NOUN	**POS**
mark	advcl	nmod	amod	compound	dobj	cc	conj	amod	dobj	**hlab**
advcl	root	dobj	dobj	dobj	advcl	advcl	advcl	dobj	conj	**hhlab**
bolster	goes	supply	supply	supply	bolster	bolster	bolster	uses	develop	**htok**
goes	faces	bolster	bolster	bolster	goes	goes	goes	develop	bolster	**hhtok**
R	Root	R	R	R	L	L	L	R	L	**hdir**
VERB	VERB	NOUN	NOUN	NOUN	VERB	VERB	VERB	NOUN	VERB	**hpos**
B	I	I	I	I	I	B	I	I	I	

Figure 1: Features extracted from a (part of a) sentence and its predicted UD dependency tree.

Our selection of supertags is first inspired by the work of Ouchi et al. (2016) on supertagging for dependency parsing, and second on our own expertise of discourse segmentation and UD scheme. Actually a large part of EDU boundaries which need syntactic information to be disambiguated are function words such as “to” or “and”. Since the UD scheme favors the attachments via content words rather than function words, the latter are often leaves in the dependency trees. It means that the valuable information for these words comes from their parents, their grand-parents or their siblings. We thus extract the following features for each token:

- *hlab*, the label of its incoming dependency (47 UD labels);
- *hhlab*, the label of its incoming dependency of the token's head (37 UD labels + NONE : 26% nmod, 23% root);
- *hdir*, the direction of its incoming dependency (3 tags : RIGHT, LEFT or ROOT);
- *hpos*, the UD POS-tag of its head (17 UD tags + ROOT : 41% NOUNs, 34% VERBs and 10% PROPNs);
- *htok*, its head token (11.483 different tokens);
- *hhtok*, the head of its head token (8.266 different tokens);
- *sleft*, the POS and incoming label of its left siblings (if it is a coordination or an object) (265 tags);
- *sright*, the POS and incoming label of its right siblings (if it is a coordination or an object) (331 tags).

An example for which supertags help to identify EDU boundaries is presented in Figure 1.

5 Corpora

Corpus	#Doc	#EDU	#Sent	#Word
En-SFU-DT	400	28, 260	16, 827	328, 362
En-DT	385	21, 789	9, 074	210, 584
Pt-DT	330	12, 594	4, 385	136, 346
Es-DT	266	3, 325	1, 816	57, 768
En-Instr-DT	176	5, 754	3, 090	56, 197
De-DT	174	2, 979	1, 805	33, 591

Table 1: Number of documents, EDUs, sentences and words (according to UDPipe).

For English, we use three corpora, allowing us to evaluate how robust is our model across domains. First, we report results on the RST-DT (from now on called En-DT), the most widely used corpus for this task. This corpus is composed of Wall Street Journal articles, it has been annotated over the Penn Treebank. We also report performance on the SFU review corpus[6] (En-SFU-DT)

[6] https://www.sfu.ca/~mtaboada

containing product reviews, and on the instructional corpus (En-Instr-DT) (Subba and Di Eugenio, 2009) built on instruction manuals.[7]

We also evaluate our model across languages. For Spanish, we report performance on the corpus (Es-DT) presented in (da Cunha et al., 2011),[8]. For German, we use the Postdam corpus (De-DT) (Stede, 2004; Stede and Neumann, 2014). For Brazilian Portuguese (Pt-DT), we merged four corpora (Cardoso et al., 2011; Collovini et al., 2007; Pardo and Seno, 2005; Pardo and Nunes, 2003, 2004) as done in (Maziero et al., 2015; Braud et al., 2017).

Table 1 summarizes statistics about the data.

6 Experiments

6.1 Evaluation

For English, on the En-DT, evaluation for discourse segmentation has been done under different conditions.

First, all previous systems were evaluated on the same set of 38 documents that initially contains 991 sentences – and more precisely on each sentence of this set for intra-sentential results. However, Soricut and Marcu (2003) do not consider sentences that are not exactly spanned by a discourse subtree (keeping only 941 sentences in the test set), and Sporleder and Lapata (2005) only keep the sentences that contain intra-sentential EDUs (608 sentences).

Since we want to give results at the document level, – with the sentence boundaries being predicted as the other EDU boundaries –, there is no reason to remove any sentences. We thus keep all the 991 sentences in the test set as done in (Fisher and Roark, 2007; Xuan Bach et al., 2012) at the sentence level, and in (Braud et al., 2017) at the document level.

For the other corpora (see Section 5), we either use the official test set (Es-DT, 84 documents) or build a test set containing 38 documents chosen randomly.

Second, since Soricut and Marcu (2003), the evaluation scores do not include the first boundary of a sentence. Exceptions are (Sporleder and Lapata, 2005), and some results in (Fisher and Roark, 2007) given to compare with the former.

For intra-sentential results, we also ignore the first boundary of each sentence when computing the final score. At the document level, we ignore the first boundary of each document (thus keeping the first boundary of the sentences within the document).

The reported score is the F_1 over the boundaries (the 'B' labels), ignoring the non-boundary words ('I' labels).

6.2 Hyper-parameters

The model has several hyper-parameters, all tuned on the development set over the F_1.

Concerning the dimensions of the input layer d, we tested several values when experimenting on models using only one type of feature (for the POS tags, we only tuned on PTB gold), with $d \in \{50, 100, 200, 300\}$ for the words, and $d \in \{4, 8, 16, 32, 64\}$ for the others.[9] We then keep the best values (300 for words, 64 for the POS tags and 32 for each supertag[10]) for each feature when concatenating.

We also tuned the number of hidden layers $n \in \{1, 2\}$, and the size of the hidden layers $h \in \{50, 100, 200, 400\}$ when experimenting on single features, and used 1 hidden layer of size 200 in our final experiments. Our output layer is of size 32.

The number of iterations i with $1 \leq i \leq 20$ is tuned on the development set for each experiment.

Note that this may not be optimal, as better results could be obtained by tuning all the hyper-parameters for each set of features. But we aim at providing a fair comparison between the models, and thus always keep the same architecture.

7 Results

7.1 Intra-sentential segmentation

Our results for intra-sentential segmentation are summarized in Table 2. Recall that these results are only on the En-DT.

Single features Using only words lead to 81.3% in F_1, which is already high considering that words are generally considered as a too sparse representation especially with a quite small dataset.

[7] We only report fully supervised results, we thus do not consider the GUM corpus and the corpus for Dutch, contrary to (Braud et al., 2017).

[8] We only use the test set from the annotator A.

[9] Supertags that correspond to words – i.e. "htok" and "hhtok" – are considered as words and thus correspond to vectors of the same dimension as other words.

[10] We report results using the supertags where the input is the concatenation of several vectors with 32 dimensions representing each supertag.

It is clear that lexical information can help, for example to identify EDUs corresponding to complements of attribution verbs – the verb could be the word at the end of the previous EDU as in example (2a) or the word beggining the EDU as in example (2b) –, these verbs being part of a limited list (e.g. “declared”, “said”, “reported”).

(2) a. [Mercedes officials said] [they expect flat sales next year]

b. [Kodak understands] [HDTV is where everybody is going,”] [says RIT’s Mr. Spaull.]

More precisely, we found out that only $1,409$ tokens are an EDU boundary in the En-DT training set (over about $16,577$ tokens in the vocabulary). Among them, 909 only appear once as a boundary, and 104 are a boundary more than 10 times making for 79.7% of all the boundaries. Lexical information is thus not so sparse for this task.

Using POS tags alone allows to improve these results, but only when using PTB gold POS (+3.7%). Contrary to words, 99.7% of the POS tags from the PTB appear as an EDU boundary more than 10 times, but only a few are almost always indicating the beginning of an EDU (i.e. more than 70% of the occurrences), namely WDT, -LRB-, WP, WRB and WP$. Our results demonstrate that our model is able to take into account the context in terms of the surrounding POS tags to identify a boundary.

As expected, using predicted PTB POS tags leads to lower results than gold ones (-3.4%), reflecting the impact of the noise introduced. Moreover, using fine grained PTB POS tags, even predicted ones, is better than using coarse grained POS UD (-5.4%), indicating that the UD scheme lacks fine distinctions needed for the task. For example, WDT and WP$ are mapped to DET in the UD scheme, and WP to PRON, two categories that become very ambiguous between indicating an EDU boundary or not (respectively, 28% and 10%), thus inducing more errors. Note that using words only is better, or similar to using predicted or coarse-grained POS tags, demonstrating once again the usefulness of the lexical information.

Finally, using supertags (STAGS) leads to results similar to using words or predicted PTB POS tags, but higher than the ones obtained with the POS UD (+4.8%), reflecting that they include more information. Among the supertags, we found that using “sleft” and “sright” does not make real difference when the supertags are used alone (80.9% with them, and 81% without). This could come from the huge sparsity of this feature.[11] We decided to not include them in the rest of the experiments.

System	(Morpho-)syntax	F_1
	Gold tokenization	
(Subba and Di Eugenio, 2007)	Gold	86.1
(Subba and Di Eugenio, 2007)	Pred	84.4
(Xuan Bach et al., 2012)	Gold	92.5
(Xuan Bach et al., 2012) Rerank	Gold	93.7
(Xuan Bach et al., 2012)	Pred	90.7
(Xuan Bach et al., 2012) Rerank	Pred	91.0
(Fisher and Roark, 2007)	Pred	90.5
Words	-	81.3
POS PTB	Gold	85.0
POS PTB	Pred	81.6
POS UD	Pred	76.2
STAGS	Pred	81.0
Words+POS PTB	Gold	91.0
Words+POS PTB	Pred	87.6
Words+POS UD	Pred	87.4
POS UD+STAGS	Pred	79.6
Words+POS UD+STAGS	Pred	86.1
	Predicted tokenization	
Words	-	82.7
POS UD	Pred	74.0
Words + POS UD	Pred	86.3
Words + POS UD + STAGS	Pred	86.8

Table 2: Intra-sentential results on the En-DT. Xuan Bach et al. (2012) report the best results, Subba and Di Eugenio (2007) is a segmenter based on neural networks, Fisher and Roark (2007) proposed a study on syntactic information.

Combining features Combining words and gold PTB POS tags leads to our better results (91%), with a large increase over using only words (+9.7%) or PTB gold POS (+6%). Note that this score is as high as the one reported by (Xuan Bach et al., 2012; Fisher and Roark, 2007) when using predicted constituent trees: this indicates that a syntactic information that is noisy does not help that much, since perfect POS tags are enough to reach the same performance.

As previously, using predicted PTB POS tags or coarse-grained UD POS tags leads to a drop in performance compared to gold PTB POS tags,

[11] 94% of the tokens have no “sleft” tag and 90% no “sright” tag.

but the scores are still largely higher than when only one type of features is used, demonstrating that lexical and morpho-syntactic features bring complementary information. The gain in F_1 is even higher when using noisy/coarse grained POS tags than when using gold ones, showing that lexical information allows to replace part of the missing/incorrect information.

Finally, combining supertags leads to mixed results: they allow to improve over using only UD POS tags (+3.4%), showing that they convey new relevant information, but the scores are lowered compared to using only the supertags (-1.4%). Moreover, when combined also with words (Words+POS UD+STAGS), we observe a small drop in performance compared to only combining them with the UD POS tags (Words+POS UD, -1.3%). More importantly, using syntactic information does not lead to results as high as the ones obtained with gold PTB POS tags.

Predicted tokenization In general, relying on predicted tokens lowers the performance, probably because it leads to more errors for POS tagging (-2.2% when using only the UD POS tags compared to gold tokenization). However, it does not really affect performance with lexical information, and the other scores are similar to the ones obtained with gold tokens.

7.2 Document-level results

Multi-lingual and multi-domain results are presented in Table 3. Again, the use of syntactic information leads to mixed results: in general, results are similar with or without supertags, but it could also lead to a large drop in performance as it can be seen especially for the En-DT, the En-SFU-DT and the En-Instr-DT. It could come from more important differences in the annotation schemes for these very different domains.

Our results are in general better than the one reported in (Braud et al., 2017), which could come from the way features are incorporated (they encode each document as a sequence of words and POS tags, rather than directly combining the vectors). Our scores on the En-DT are a bit lower than those reported in (Braud et al., 2017), but note that these authors fine tuned their system at the document level, while we optimized it at the intra-sentential one.

	SOA	Words+UD	Words+UD+S-tags
En-DT (news)	**89.5**	89.0	87.0
En-SFU-DT	85.5	**87.6**	86.0
En-Instr-DT	87.1	**88.3**	86.4
Pt-DT	82.2	82.9	**83.0**
Es-DT	**79.3**	78.7	78.3
De-DT	85.1	85.8	**86.2**

Table 3: Multi-domain and multi-lingual document-level results. State-of-the-art (SOA) results reported in (Braud et al., 2017).

8 Discussion

In order to investigate the drop in F_1 between gold and predicted POS tags we looked at the distribution of the POS tags in the train set, and, for each POS, the percentage of instances being a discourse boundary and their accuracy when predicted.

Globally, the accuracy of POS-tagging on EDU boundaries is lower (95.6%) than on the non-EDU boundaries. However, the most frequent POS assigned to EDU boundaries (i.e. 'IN', 'CC', 'PRP', 'TO' and 'VBG') achieve accuracy between 97.4 and 100% and cover 50% of the EDU boundaries.

We also saw that some very frequent POS are rarely an EDU boundary, such as 'NN, 'JJ' or the comma.[12] But the low accuracy of some of these frequent POS tags (94.8 for 'NN' and 90.1 for 'JJ') can still hurt discourse segmentation as they often appear in the context of the EDU boundaries. On the contrary, some quite infrequent POS are really frequent EDU boundaries, such as 'WP' (Wh-pronoun), 'WDT' (Wh-determiner), 'WRB' (Wh-adverb), -LRB-, 'WP$' (Possessive wh-pronoun) and 'LS' (List item marker). Except for 'WDT' (90.8%) their POS-tagging scores are high (100% for 'WP', '-LRB-' and 'Wp$' and 98.3% for 'WRB'). But because they are infrequent, they could be hard to identify as boundaries. They could be even more difficult to identify using the UD scheme since these POS tags are mapped to frequent UD POS tags that cover very different tokens ('DET', 'PRON', 'ADV').

9 Conclusion

We proposed new discourse segmenters that make use of resources available for many languages and domains. We investigated the usefulness of syn-

[12] The only example with a comma corresponds probably to a segmentation error, the comma being preceded by a point corresponding to an acronym (Doc 1390).

tactic information when derived from dependency parse trees, and showed that this information is not as useful as expected, and that gold POS tags give as high results as using predicted constituent trees. We also showed that scores are lowered when considering a realistic setting, relying on predicted tokenization and not assuming gold sentences. We make our code available at https://bitbucket.org/chloebt/discourse/.

Acknowledgments

We would like to thank the anonymous reviewers for their comments. This research is funded by the ERC Starting Grant LOWLANDS No. 313695.

References

Stergos Afantenos, Pascal Denis, Philippe Muller, and Laurence Danlos. 2010. Learning recursive segments for discourse parsing. In *Proceedings of LREC*.

Nicholas Asher and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

Chloé Braud, Ophélie Lacroix, and Anders Søgaard. 2017. Cross-lingual and cross-domain discourse segmentation of entire documents. In *arXiv preprint arXiv::1704.04100, To appear in Proceedings of ACL 17*.

Paula C.F. Cardoso, Erick G. Maziero, Mara Luca Castro Jorge, Eloize R.M. Seno, Ariani Di Felippo, Lucia Helena Machado Rino, Maria das Gracas Volpe Nunes, and Thiago A. S. Pardo. 2011. CSTNews - a discourse-annotated corpus for single and multi-document summarization of news texts in Brazilian Portuguese. In *Proceedings of the 3rd RST Brazilian Meeting*, pages 88–105.

Lynn Carlson and Daniel Marcu. 2001. Discourse tagging reference manual. Technical report, University of Southern California Information Sciences Institute.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*.

Sandra Collovini, Thiago I Carbonel, Juliana Thiesen Fuchs, Jorge César Coelho, Lúcia Rino, and Renata Vieira. 2007. Summ-it: Um corpus anotado com informaçoes discursivas visandoa sumarizaçao automática. *Proceedings of TIL*.

Iria da Cunha, Eric SanJuan, Juan-Manuel Torres-Moreno, Marina Lloberas, and Irene Castellón. 2010. DiSeg: Un segmentador discursivo automático para el español. *Procesamiento del lenguaje natural*, 45:145–152.

Iria da Cunha, Eric SanJuan, Juan-Manuel Torres-Moreno, Marina Lloberes, and Irene Castellón. 2012. DiSeg 1.0: The first system for Spanish discourse segmentation. *Expert Syst. Appl.*, 39(2):1671–1678.

Iria da Cunha, Juan-Manuel Torres-Moreno, and Gerardo Sierra. 2011. On the development of the RST Spanish Treebank. In *Proceedings of the Fifth Linguistic Annotation Workshop, LAW*.

Vanessa Wei Feng and Graeme Hirst. 2014. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of ACL*.

Seeger Fisher and Brian Roark. 2007. The utility of parse-derived features for automatic discourse segmentation. In *Proceedings ACL*.

Alex Graves and Jrgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, pages 5–6.

Hugo Hernault, Danushka Bollegala, and Mitsuru Ishizuka. 2010. A sequential model for discourse segmentation. In *Proceedings of CICLing*.

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

Yangfeng Ji and Jacob Eisenstein. 2014. Representation learning for text-level discourse parsing. In *Proceedings of ACL*.

Shafiq Joty, Giuseppe Carenini, and Raymond T. Ng. 2015. Codra: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41:3.

Shafiq R. Joty, Giuseppe Carenini, Raymond T. Ng, and Yashar Mehdad. 2013. Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In *Proceedings of ACL*.

Huong Le Thanh, Geetha Abeysinghe, and Christian Huyck. 2004. Generating discourse structures for written text. In *Proceedings of COLING*.

Jiwei Li, Rumeng Li, and Eduard H. Hovy. 2014. Recursive deep models for discourse parsing. In *Proceedings of EMNLP*.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text*, 8:243–281.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Erick G. Maziero, Graeme Hirst, and Thiago A. S. Pardo. 2015. Adaptation of discourse parsing models for Portuguese language. In *Proceedings of the Brazilian Conference on Intelligent Systems (BRACIS)*.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Riyaz Ahmad Bhat, Cristina Bosco, Gosse Bouma, Sam Bowman, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Çar Çöltekin, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Sebastian Garza, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gokirmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Normunds Grūzītis, Bruno Guillaume, Jan Hajič, Dag Haug, Barbora Hladká, Radu Ion, Elena Irimia, Anders Johannsen, Hüner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Jessica Kenney, Simon Krek, Veronika Laippala, Lucia Lam, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Keiko Sophie Mori, Shunsuke Mori, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Vitaly Nikolaev, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalnia, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Loganathan Ramasamy, Laura Rituma, Rudolf Rosa, Shadi Saleh, Baiba Saulīte, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Carolyn Spadine, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jing Xian Wang, Jonathan North Washington, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2016. Universal dependencies 1.3. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Hiroki Ouchi, Kevin Duh, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Transition-based dependency parsing exploiting supertags. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):2059–2068.

Thiago A. S. Pardo and Maria das Graças Volpe Nunes. 2003. A construção de um corpus de textos científicos em Português do Brasil e sua marcação retórica. Technical report, Technical Report.

Thiago A. S. Pardo and Maria das Graças Volpe Nunes. 2004. Relações retóricas e seus marcadores superficiais: Análise de um corpus de textos científicos em Português do Brasil. *Relatório Técnico NILC*.

Thiago A. S. Pardo and Maria das Graças Volpe Nunes. 2008. On the development and evaluation of a Brazilian Portuguese discourse parser. *Revista de Informática Teórica e Aplicada*, 15(2):43–64.

Thiago A. S. Pardo and Eloize R. M. Seno. 2005. Rhetalho: Um corpus de referłncia anotado retoricamente. In *Proceedings of Encontro de Corpora*.

Uladzimir Sidarenka, Andreas Peldszus, and Manfred Stede. 2015. Discourse segmentation of german texts. *Journal of Language Technology and Computational Linguistics*, 30(1):71–98.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of NAACL*.

Caroline Sporleder and Mirella Lapata. 2005. Discourse chunking and its application to sentence compression. In *Proceedings of HLT/EMNLP*.

Manfred Stede. 2004. The potsdam commentary corpus. In *Proceedings of the ACL Workshop on Discourse Annotation*.

Manfred Stede and Arne Neumann. 2014. Potsdam commentary corpus 2.0: Annotation for discourse research. In *Proceedings of LREC*.

Milan Straka, Jan Hajič, and Straková. 2016. UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*.

Rajen Subba and Barbara Di Eugenio. 2007. Automatic discourse segmentation using neural networks. In *Workshop on the Semantics and Pragmatics of Dialogue*.

Rajen Subba and Barbara Di Eugenio. 2009. An effective discourse parser that uses rich linguistic information. In *Proceedings of ACL-HLT*.

Milan Tofiloski, Julian Brooke, and Maite Taboada. 2009. A syntactic and lexical-based discourse segmenter. In *Proceedings of ACL-IJCNLP*.

Nynke van der Vliet. 2010. Syntax-based discourse segmentation of Dutch text. In *15th Student Session, ESSLLI*.

Nynke van der Vliet and Gisela Redeker. 2011. Complex sentences as leaky units in discourse parsing. In *Proceedings of Constraints in Discourse*.

Ngo Xuan Bach, Nguyen Le Minh, and Akira Shimazu. 2012. A reranking model for discourse segmentation using subtree features. In *Proceedings of Sigdial*.

Deal or No Deal? End-to-End Learning for Negotiation Dialogues

Mike Lewis[1], **Denis Yarats**[1], **Yann N. Dauphin**[1], **Devi Parikh**[2,1] and **Dhruv Batra**[2,1]
[1]Facebook AI Research [2]Georgia Institute of Technology
{mikelewis,denisy,ynd}@fb.com {parikh,dbatra}@gatech.edu

Abstract

Much of human dialogue occurs in semi-cooperative settings, where agents with different goals attempt to agree on common decisions. Negotiations require complex communication and reasoning skills, but success is easy to measure, making this an interesting task for AI. We gather a large dataset of human-human negotiations on a multi-issue bargaining task, where agents who cannot observe each other's reward functions must reach an agreement (or a deal) via natural language dialogue. For the first time, we show it is possible to train end-to-end models for negotiation, which must learn both linguistic and reasoning skills with no annotated dialogue states. We also introduce *dialogue rollouts*, in which the model plans ahead by simulating possible complete continuations of the conversation, and find that this technique dramatically improves performance. Our code and dataset are publicly available.[1]

1 Introduction

Intelligent agents often need to cooperate with others who have different goals, and typically use natural language to agree on decisions. Negotiation is simultaneously a linguistic and a reasoning problem, in which an intent must be formulated and then verbally realised. Such dialogues contain both cooperative and adversarial elements, and require agents to understand, plan, and generate utterances to achieve their goals (Traum et al., 2008; Asher et al., 2012).

We collect the first large dataset of natural language negotiations between two people, and show that end-to-end neural models can be trained to negotiate by maximizing the likelihood of human actions. This approach is scalable and domain-independent, but does not model the strategic skills required for negotiating well. We further show that models can be improved by training and decoding to maximize reward instead of likelihood—by training with self-play reinforcement learning, and using rollouts to estimate the expected reward of utterances during decoding.

To study semi-cooperative dialogue, we gather a dataset of 5808 dialogues between humans on a negotiation task. Users were shown a set of items with a value for each, and asked to agree how to divide the items with another user who has a different, unseen, value function (Figure 1).

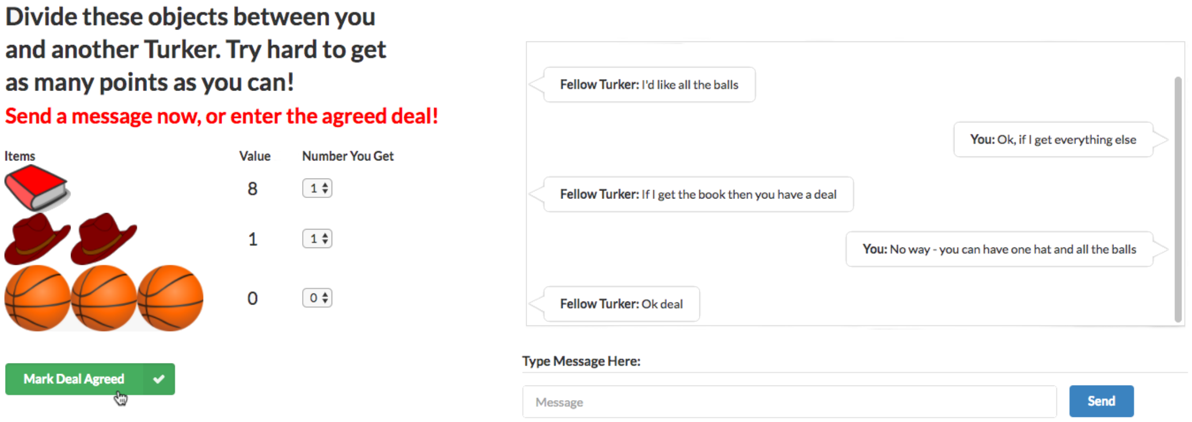
We first train recurrent neural networks to imitate human actions. We find that models trained to maximise the likelihood of human utterances can generate fluent language, but make comparatively poor negotiators, which are overly willing to compromise. We therefore explore two methods for improving the model's strategic reasoning skills—both of which attempt to optimise for the agent's goals, rather than simply imitating humans:

Firstly, instead of training to optimise likelihood, we show that our agents can be considerably improved using *self play*, in which pre-trained models practice negotiating with each other in order to optimise performance. To avoid the models diverging from human language, we interleave reinforcement learning updates with supervised updates. For the first time, we show that end-to-end dialogue agents trained using reinforcement learning outperform their supervised counterparts in negotiations with humans.

Secondly, we introduce a new form of planning for dialogue called *dialogue rollouts*, in which an

[1] https://github.com/facebookresearch/end-to-end-negotiator

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2443–2453
Copenhagen, Denmark, September 7–11, 2017.



Divide these objects between you and another Turker. Try hard to get as many points as you can!

Send a message now, or enter the agreed deal!

Items	Value	Number You Get
	8	1
	1	1
	0	0

Fellow Turker: I'd like all the balls

You: Ok, if I get everything else

Fellow Turker: If I get the book then you have a deal

You: No way - you can have one hat and all the balls

Fellow Turker: Ok deal

Type Message Here:

Figure 1: A dialogue in our Mechanical Turk interface, which we used to collect a negotiation dataset.

agent simulates complete dialogues during decoding to estimate the reward of utterances. We show that decoding to maximise the reward function (rather than likelihood) significantly improves performance against both humans and machines.

Analysing the performance of our agents, we find evidence of sophisticated negotiation strategies. For example, we find instances of the model feigning interest in a valueless issue, so that it can later 'compromise' by conceding it. Deceit is a complex skill that requires hypothesising the other agent's beliefs, and is learnt relatively late in child development (Talwar and Lee, 2002). Our agents have *learnt to deceive* without any explicit human design, simply by trying to achieve their goals.

The rest of the paper proceeds as follows: §2 describes the collection of a large dataset of human-human negotiation dialogues. §3 describes a baseline supervised model, which we then show can be improved by goal-based training (§4) and decoding (§5). §6 measures the performance of our models and humans on this task, and §7 gives a detailed analysis and suggests future directions.

2 Data Collection

2.1 Overview

To enable end-to-end training of negotiation agents, we first develop a novel negotiation task and curate a dataset of human-human dialogues for this task. This task and dataset follow our proposed general framework for studying semi-cooperative dialogue. Initially, each agent is shown an input specifying a space of possible actions and a reward function which will score the outcome of the negotiation. Agents then sequentially take turns of either sending natural language messages, or selecting that a final decision has been reached. When one agent selects that an agreement has been made, both agents independently output what they think the agreed decision was. If conflicting decisions are made, both agents are given zero reward.

2.2 Task

Our task is an instance of *multi issue bargaining* (Fershtman, 1990), and is based on DeVault et al. (2015). Two agents are both shown the same collection of items, and instructed to divide them so that each item assigned to one agent.

Each agent is given a different randomly generated value function, which gives a non-negative value for each item. The value functions are constrained so that: (1) the total value for a user of all items is 10; (2) each item has non-zero value to at least one user; and (3) some items have non-zero value to both users. These constraints enforce that it is not possible for both agents to receive a maximum score, and that no item is worthless to both agents, so the negotiation will be competitive. After 10 turns, we allow agents the option to complete the negotiation with no agreement, which is worth 0 points to both users. We use 3 item types (*books*, *hats*, *balls*), and between 5 and 7 total items in the pool. Figure 1 shows our interface.

2.3 Data Collection

We collected a set of human-human dialogues using Amazon Mechanical Turk. Workers were paid \$0.15 per dialogue, with a \$0.05 bonus for maximal scores. We only used workers based in the United States with a 95% approval rating and at least 5000 previous HITs. Our data collection interface was adapted from that of Das et al. (2016).

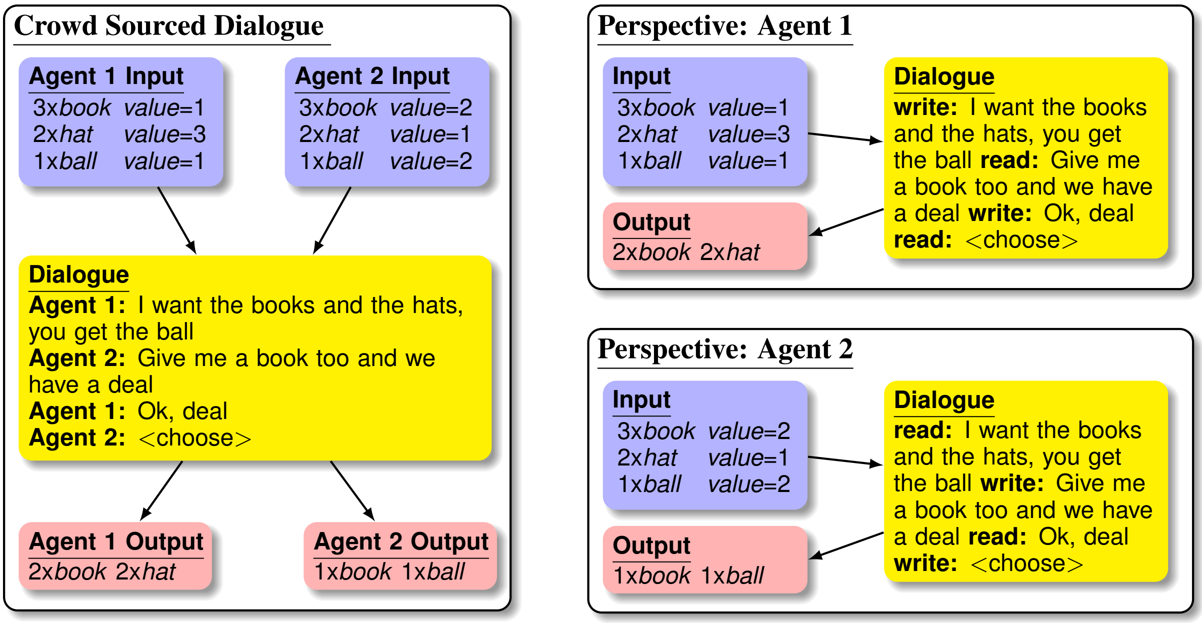


Figure 2: Converting a crowd-sourced dialogue (left) into two training examples (right), from the perspective of each user. The perspectives differ on their input goals, output choice, and in special tokens marking whether a statement was read or written. We train conditional language models to predict the dialogue given the input, and additional models to predict the output given the dialogue.

We collected a total of 5808 dialogues, based on 2236 unique scenarios (where a scenario is the available items and values for the two users). We held out a test set of 252 scenarios (526 dialogues). Holding out test scenarios means that models must generalise to new situations.

3 Likelihood Model

We propose a simple but effective baseline model for the conversational agent, in which a sequence-to-sequence model is trained to produce the complete dialogue, conditioned on an agent's input.

3.1 Data Representation

Each dialogue is converted into two training examples, showing the complete conversation from the perspective of each agent. The examples differ on their input goals, output choice, and whether utterances were read or written.

Training examples contain an input goal g, specifying the available items and their values, a dialogue x, and an output decision o specifying which items each agent will receive. Specifically, we represent g as a list of six integers corresponding to the count and value of each of the three item types. Dialogue x is a list of tokens $x_{0..T}$ containing the turns of each agent interleaved with symbols marking whether a turn was written by the agent or their partner, terminating in a special token indicating one agent has marked that an agreement has been made. Output o is six integers describing how many of each of the three item types are assigned to each agent. See Figure 2.

3.2 Supervised Learning

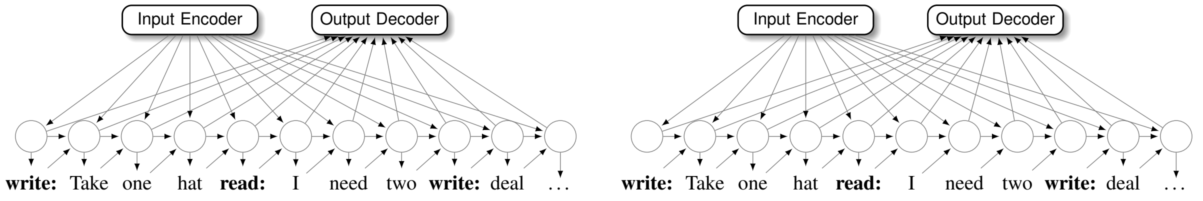
We train a sequence-to-sequence network to generate an agent's perspective of the dialogue conditioned on the agent's input goals (Figure 3a).

The model uses 4 recurrent neural networks, implemented as GRUs (Cho et al., 2014): GRU$_w$, GRU$_g$, GRU$_{\overrightarrow{o}}$, and GRU$_{\overleftarrow{o}}$.

The agent's input goals g are encoded using GRU$_g$. We refer to the final hidden state as h^g. The model then predicts each token x_t from left to right, conditioned on the previous tokens and h^g. At each time step t, GRU$_w$ takes as input the previous hidden state h_{t-1}, previous token x_{t-1} (embedded with a matrix E), and input encoding h^g. Conditioning on the input at each time step helps the model learn dependencies between language and goals.

$$h_t = \text{GRU}_w(h_{t-1}, [Ex_{t-1}, h^g]) \qquad (1)$$

The token at each time step is predicted with a softmax, which uses weight tying with the embed-


Input Encoder
Output Decoder
write: Take one hat read: I need two write: deal ...
Input Encoder
Output Decoder
write: Take one hat read: I need two write: deal ...

(a) Supervised Training

(b) Decoding, and Reinforcement Learning

Figure 3: Our model: tokens are predicted conditioned on previous words and the input, then the output is predicted using attention over the complete dialogue. In supervised training (3a), we train the model to predict the tokens of *both* agents. During decoding and reinforcement learning (3b) some tokens are sampled from the model, but some are generated by the other agent and are only encoded by the model.

ding matrix E (Mao et al., 2015):

$$p_\theta(x_t|x_{0..t-1}, g) \propto \exp(E^T h_t) \tag{2}$$

Note that the model predicts both agent's words, enabling its use as a forward model in Section 5.

At the end of the dialogue, the agent outputs a set of tokens o representing the decision. We generate each output conditionally independently, using a separate classifier for each. The classifiers share bidirectional GRUs and an attention mechanism (Bahdanau et al., 2014) over the dialogue, and additionally condition on the input goals.

$$h_t^{\overrightarrow{o}} = \text{GRU}_{\overrightarrow{o}}(h_{t-1}^{\overrightarrow{o}}, [Ex_t, h_t]) \tag{3}$$

$$h_t^{\overleftarrow{o}} = \text{GRU}_{\overleftarrow{o}}(h_{t+1}^{\overleftarrow{o}}, [Ex_t, h_t]) \tag{4}$$

$$h_t^o = [h_t^{\overleftarrow{o}}, h_t^{\overrightarrow{o}}] \tag{5}$$

$$h_t^a = W^a[\tanh(W^h h_t^o)] \tag{6}$$

$$\alpha_t = \frac{\exp(w \cdot h_t^a)}{\sum_{t'} \exp(w \cdot h_{t'}^a)} \tag{7}$$

$$h^s = \tanh(W^s[h^g, \sum_t \alpha_t h_t]) \tag{8}$$

The output tokens are predicted using softmax:

$$p_\theta(o_i|x_{0..t}, g) \propto \exp(W^{o_i} h^s) \tag{9}$$

The model is trained to minimize the negative log likelihood of the token sequence $x_{0..T}$ conditioned on the input goals g, and of the outputs o conditioned on x and g. The two terms are weighted with a hyperparameter α.

$$L(\theta) = -\underbrace{\sum_{x,g} \sum_t \log p_\theta(x_t|x_{0..t-1}, g)}_{\text{Token prediction loss}} - \alpha \underbrace{\sum_{x,g,o} \sum_j \log p_\theta(o_j|x_{0..T}, g)}_{\text{Output choice prediction loss}} \tag{10}$$

Unlike the Neural Conversational Model (Vinyals and Le, 2015), our approach shares all parameters for reading and generating tokens.

3.3 Decoding

During decoding, the model must generate an output token x_t conditioned on dialogue history $x_{0..t-1}$ and input goals g, by sampling from p_θ:

$$x_t \sim p_\theta(x_t|x_{0..t-1}, g) \tag{11}$$

If the model generates a special *end-of-turn* token, it then encodes a series of tokens output by the other agent, until its next turn (Figure 3b).

The dialogue ends when either agent outputs a special *end-of-dialogue* token. The model then outputs a set of choices o. We choose each item independently, but enforce consistency by checking the solution is in a feasible set O:

$$o^* = \operatorname*{argmax}_{o \in O} \prod_i p_\theta(o_i|x_{0..T}, g) \tag{12}$$

In our task, a solution is feasible if each item is assigned to exactly one agent. The space of solutions is small enough to be tractably enumerated.

4 Goal-based Training

Supervised learning aims to imitate the actions of human users, but does not explicitly attempt to maximise an agent's goals. Instead, we explore pre-training with supervised learning, and then fine-tuning against the evaluation metric using reinforcement learning. Similar two-stage learning strategies have been used previously (e.g. Li et al. (2016); Das et al. (2017)).

During reinforcement learning, an agent A attempts to improve its parameters from conversations with another agent B. While the other agent B could be a human, in our experiments we used

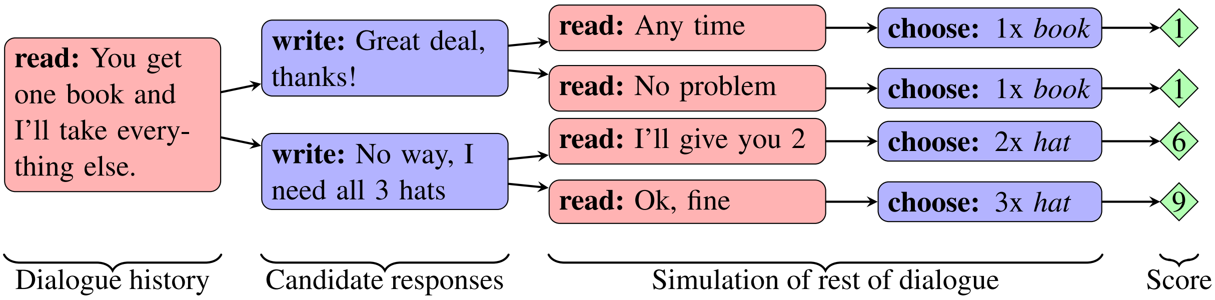


Figure 4: Decoding through rollouts: The model first generates a small set of candidate responses. For each candidate, it then simulates a number of possible complete future conversations by sampling, and estimates the expected future reward by averaging the scores. The system outputs the candidate with the highest expected reward.

our fixed supervised model that was trained to imitate humans. The second model is fixed as we found that updating the parameters of both agents led to divergence from human language. In effect, agent A learns to improve by simulating conversations with the help of a surrogate forward model.

Agent A reads its goals g and then generates tokens $x_{0..n}$ by sampling from p_θ. When x generates an end-of-turn marker, it then reads in tokens $x_{n+1..m}$ generated by agent B. These turns alternate until one agent emits a token ending the dialogue. Both agents then output a decision o and collect a reward from the environment (which will be 0 if they output different decisions). We denote the subset of tokens generated by A as X^A (e.g. tokens with incoming arrows in Figure 3b).

After a complete dialogue has been generated, we update agent A's parameters based on the outcome of the negotiation. Let r^A be the score agent A achieved in the completed dialogue, T be the length of the dialogue, γ be a discount factor that rewards actions at the end of the dialogue more strongly, and μ be a running average of completed dialogue rewards so far[2]. We define the future reward R for an action $x_t \in X^A$ as follows:

$$R(x_t) = \sum_{x_t \in X^A} \gamma^{T-t}(r^A(o) - \mu) \qquad (13)$$

We then optimise the expected reward of each action $x_t \in X^A$:

$$L_\theta^{RL} = \mathbb{E}_{x_t \sim p_\theta(x_t|x_{0..t-1},g)}[R(x_t)] \qquad (14)$$

The gradient of L_θ^{RL} is calculated as in REINFORCE (Williams, 1992):

$$\nabla_\theta L_\theta^{RL} = \sum_{x_t \in X^A} \mathbb{E}_{x_t}[R(x_t)\nabla_\theta \log(p_\theta(x_t|x_{0..t-1}, g))] \qquad (15)$$

5 Goal-based Decoding

Likelihood-based decoding (§3.3) may not be optimal. For instance, an agent may be choosing between accepting an offer, or making a counter offer. The former will often have a higher likelihood under our model, as there are fewer ways to agree than to make another offer, but the latter may lead to a better outcome. Goal-based decoding also allows more complex dialogue strategies. For example, a deceptive utterance is likely to have a low model score (as users were generally honest in the supervised data), but may achieve high reward.

We instead explore decoding by maximising expected reward. We achieve this by using p_θ as a forward model for the complete dialogue, and then deterministically computing the reward. Rewards for an utterance are averaged over samples to calculate expected future reward (Figure 4).

We use a two stage process: First, we generate c candidate utterances $U = u_{0..c}$, representing possible complete turns that the agent could make, which are generated by sampling from p_θ until the *end-of-turn* token is reached. Let $x_{0..n-1}$ be current dialogue history. We then calculate the expected reward $R(u)$ of candidate utterance $u = x_{n,n+k}$ by repeatedly sampling $x_{n+k+1,T}$ from p_θ, then choosing the best output o using Equation 12, and finally deterministically computing the reward $r(o)$. The reward is scaled by the probability of the output given the dialogue, be-

[2] As all rewards are non-negative, we instead re-scale them by subtracting the mean reward found during self play. Shifting in this way can reduce the variance of our estimator.

Algorithm 1 Dialogue Rollouts algorithm.

```
procedure ROLLOUT($x_{0..i}, g$)
  $u^* \leftarrow \varnothing$
  for $c \in \{1..C\}$ do                ▷ $C$ candidate moves
    $j \leftarrow i$
    do                                   ▷ Rollout to end of turn
      $j \leftarrow j + 1$
      $x_j \sim p_\theta(x_j | x_{0..j-1}, g)$
    while $x_k \notin \{read:, choose:\}$
    $u \leftarrow x_{i+1}..x_j$          ▷ $u$ is candidate move
    for $s \in \{1..S\}$ do              ▷ $S$ samples per move
      $k \leftarrow j$                   ▷ Start rollout from end of $u$
      while $x_k \neq choose:$ do
      ▷ Rollout to end of dialogue
        $k \leftarrow k + 1$
        $x_k \sim p_\theta(x_k | x_{0..k-1}, g)$
      ▷ Calculate rollout output and reward
      $o \leftarrow \text{argmax}_{o' \in O}\, p(o' | x_{0..k}, g)$
      $R(u) \leftarrow R(u) + r(o) p(o' | x_{0..k}, g)$
    if $R(u) > R(u^*)$ then
      $u^* \leftarrow u$
  return $u^*$                           ▷ Return best move
```

cause if the agents select different outputs then they both receive 0 reward.

$$R(x_{n..n+k}) = \mathbb{E}_{x_{(n+k+1..T;o)} \sim p_\theta}[r(o) p_\theta(o | x_{0..T})] \tag{16}$$

We then return the utterance maximizing R.

$$u^* = \underset{u \in U}{\text{argmax}}\, R(u) \tag{17}$$

We use 5 rollouts for each of 10 candidate turns.

6 Experiments

6.1 Training Details

We implement our models using PyTorch. All hyper-parameters were chosen on a development dataset. The input tokens are embedded into a 64-dimensional space, while the dialogue tokens are embedded with 256-dimensional embeddings (with no pre-training). The input GRU_g has a hidden layer of size 64 and the dialogue GRU_w is of size 128. The output GRU$_{\overrightarrow{o}}$ and GRU$_{\overleftarrow{o}}$ both have a hidden state of size 256, the size of h^s is 256 as well. During supervised training, we optimise using stochastic gradient descent with a minibatch size of 16, an initial learning rate of 1.0, Nesterov momentum with μ=0.1 (Nesterov, 1983), and clipping gradients whose L^2 norm exceeds 0.5. We train the model for 30 epochs and pick the snapshot of the model with the best validation perplexity. We then annealed the learning rate by a factor of 5 each epoch. We weight the terms in the loss function (Equation 10) using α=0.5. We do not train against output decisions where humans selected different agreements. Tokens occurring fewer than 20 times are replaced with an ‘unknown’ token.

During reinforcement learning, we use a learning rate of 0.1, clip gradients above 1.0, and use a discount factor of γ=0.95. After every 4 reinforcement learning updates, we make a supervised update with mini-batch size 16 and learning rate 0.5, and we clip gradients at 1.0. We used 4086 simulated conversations.

When sampling words from p_θ, we reduce the variance by doubling the values of logits (i.e. using temperature of 0.5).

6.2 Comparison Systems

We compare the performance of the following: LIKELIHOOD uses supervised training and decoding (§3), RL is fine-tuned with goal-based self-play (§4), ROLLOUTS uses supervised training combined with goal-based decoding using rollouts (§5), and RL+ROLLOUTS uses rollouts with a base model trained with reinforcement learning.

6.3 Intrinsic Evaluation

For development, we use measured the perplexity of user generated utterances, conditioned on the input and previous dialogue.

Results are shown in Table 3, and show that the simple LIKELIHOOD model produces the most human-like responses, and the alternative training and decoding strategies cause a divergence from human language. Note however, that this divergence may not necessarily correspond to lower quality language—it may also indicate different strategic decisions about what to say. Results in §6.4 show all models could converse with humans.

6.4 End-to-End Evaluation

We measure end-to-end performance in dialogues both with the likelihood-based agent and with humans on Mechanical Turk, on held out scenarios.

Humans were told that they were interacting with other humans, as they had been during the collection of our dataset (and few appeared to realize they were in conversation with machines).

	vs. LIKELIHOOD				vs. Human			
Model	Score (all)	Score (agreed)	% Agreed	% Pareto Optimal	Score (all)	Score (agreed)	% Agreed	% Pareto Optimal
LIKELIHOOD	5.4 vs. 5.5	6.2 vs. 6.2	87.9	49.6	4.7 vs. 5.8	6.2 vs. 7.6	**76.5**	66.2
RL	7.1 vs. 4.2	7.9 vs. 4.7	89.9	58.6	4.3 vs. 5.0	6.4 vs. 7.5	67.3	69.1
ROLLOUTS	7.3 vs. 5.1	7.9 vs. 5.5	92.9	63.7	**5.2 vs. 5.4**	7.1 vs. 7.4	72.1	78.3
RL+ROLLOUTS	**8.3 vs. 4.2**	**8.8 vs. 4.5**	**94.4**	**74.8**	4.6 vs. 4.2	**8.0 vs. 7.1**	57.2	**82.4**

Table 1: End task evaluation on heldout scenarios, against the LIKELIHOOD model and humans from Mechanical Turk. The maximum score is 10. *Score (all)* gives 0 points when agents failed to agree.

Metric	Dataset
Number of Dialogues	5808
Average Turns per Dialogue	6.6
Average Words per Turn	7.6
% Agreed	80.1
Average Score (/10)	6.0
% Pareto Optimal	76.9

Table 2: Statistics on our dataset of crowdsourced dialogues between humans.

Model	Valid PPL	Test PPL	Test Avg. Rank
LIKELIHOOD	5.62	5.47	521.8
RL	6.03	5.86	517.6
ROLLOUTS	-	-	844.1
RL+ROLLOUTS	-	-	859.8

Table 3: Intrinsic evaluation showing the average perplexity of tokens and rank of complete turns (out of 2083 unique human messages from the test set). Lower is more human-like for both.

We measure the following statistics:

Score: The average score for each agent (which could be a human or model), out of 10.

Agreement: The percentage of dialogues where both agents agreed on the same decision.

Pareto Optimality: The percentage of Pareto optimal solutions for agreed deals (a solution is Pareto optimal if neither agent's score can be improved without lowering the other's score). Lower scores indicate inefficient negotiations.

Results are shown in Table 1. Firstly, we see that the RL and ROLLOUTS models achieve significantly better results when negotiating with the LIKELIHOOD model, particularly the RL+ROLLOUTS model. The percentage of Pareto optimal solutions also increases, showing a better exploration of the solution space. Compared to human-human negotiations (Table 2), the best models achieve a higher agreement rate, better scores, and similar Pareto efficiency. This result confirms that attempting to maximise reward can outperform simply imitating humans.

Similar trends hold in dialogues with humans, with goal-based reasoning outperforming imitation learning. The ROLLOUTS model achieves comparable scores to its human partners, and the RL+ROLLOUTS model actually achieves higher scores. However, we also find significantly more cases of the goal-based models failing to agree a deal with humans—largely a consequence of their more aggressive negotiation tactics (see §7).

7 Analysis

Table 1 shows large gains from goal-based methods. In this section, we explore the strengths and weaknesses of our models.

Goal-based models negotiate harder. The RL+ROLLOUTS model has much longer dialogues with humans than LIKELIHOOD (7.2 turns vs. 5.3 on average), indicating that the model is accepting deals less quickly, and negotiating harder.

A negative consequence of more aggressive negotiation strategies is that humans were more likely to walk away with no deal, resulting in lower agreement rates. Even though failing to agree was worth 0 points, people often preferred this course over capitulating to an uncompromising opponent—a factor not well captured by the simulated partner in reinforcement learning training or rollouts (as reflected by the larger gains from goal-based models in dialogues with the LIKELIHOOD model). In particular, the goal-based models are prone to simply rephrasing the same demand each turn, which is a more effective strategy against the LIKELIHOOD model than humans.

One approach to mitigating this issue would be to learn the impact of uncooperative strategies across multiple rounds of negotiation (Lerer and Peysakhovich, 2017). In single dialogues, failing to agree is never better than reaching some agreement. However, across multiple rounds, learning to walk away without a deal may incentivize partners to cooperate more in future.

Figure 5 shows an example of our goal-based

Input		
Human	1x**book** *value*=3 1x**hat** *value*=1 3x**ball** *value*=2	
RL+ROLLOUTS	1x**book** *value*=6 1x**hat** *value*=4 3x**ball** *value*=0	
RL+ROLLOUTS	You can have the balls if i can have the hat and book	
Human	Nope, give me the books and you can have everything else	
RL+ROLLOUTS	Ok. I will take the book and one hat	
Human	I get all the balls?	
RL+ROLLOUTS	Yes. I can do that.	
Human	Ok, deal	
Output		**Reward**
RL+ROLLOUTS	1x**book** 1x**hat**	10/10
Human	3x**ball**	6/10

Figure 5: Example of a model stubbornly holding out for maximum points in a dialogue.

Input		
Human	3x**book** *value*=2 3x**hat** *value*=1 1x**ball** *value*=1	
RL+ROLLOUTS	3x**book** *value*=1 3x**hat** *value*=2 1x**ball** *value*=1	
Human	I'd like the books and the hats.	
RL+ROLLOUTS	I will take the books and hats	
Human	I really need the books and at least one other object.	
RL+ROLLOUTS	Ok, you can have one book and one ball	
Human	If I can have all the books, I can leave you the rest.	
RL+ROLLOUTS	Deal	
Output		**Reward**
RL+ROLLOUTS	3x**hat** 1x**ball**	7/10
Human	3x**book**	6/10

Figure 6: Example of model compromising.

Input		
RL+ROLLOUTS	4x**book** *value*=0 2x**hat** *value*=5 1x**ball** *value*=0	
Human	4x**book** *value*=1 2x**hat** *value*=2 1x**ball** *value*=2	
RL+ROLLOUTS	You can have the ball if I can have everything else.	
Human	No can't do that the ball is not going to give me anything	
RL+ROLLOUTS	I need the hats and you can have the rest	
Human	Deal	
RL+ROLLOUTS	Great! Thank you!	
Output		**Reward**
RL+ROLLOUTS	2x**hat**	10/10
Human	4x**book** 1x**ball**	6/10

Figure 7: Dialogue in which the model's initial interest in the valueless books allows it to compromise while achieving a maximum score.

model stubbornly negotiating until it achieves a good outcome.

Models learn to be deceptive. Deception can be an effective negotiation tactic. We found numerous cases of our models initially feigning interest in a valueless item, only to later 'compromise' by conceding it. Figure 7 shows an example. This strategy has previously been coded into negotiation agents by Gratch et al. (2016), whereas our agents have learned it from data.

Models produce meaningful novel sentences. One interesting question is whether our models are capable of generating novel sentences in the new circumstances they find themselves in, or if they simply repeat messages from the training data verbatim. We find that 76% of messages produced by the LIKELIHOOD model in self-play were found in the training data. We manually examined the novel utterances produced by our model, and found that the overwhelming majority were fluent English sentences in isolation—showing that the model has learnt a good language model for the domain (in addition to results that show it uses language effectively to achieve its goals). These results suggest that although neural models are prone to the safer option of repeating sentences from training data, they are capable of generalising when necessary. Future work should choose domains that force a higher degree of diversity in utterances.

Maintaining multi-sentence coherence is challenging. One common linguistic error we see RL+ROLLOUTS make is to start a message by indicating agreement (e.g. *I agree* or *Deal*), but then going on to propose a counter offer—a behaviour that human partners found frustrating. One explanation is that the model has learnt that in the supervised data, messages beginning with *I agree* are often at the end of the dialogue, and partners rarely reply with further negotiation—so the models using rollouts and reinforcement learning believe this tactic will help their offer to be accepted.

8 Related Work

Most work on goal orientated dialogue systems has assumed that state representations are annotated in the training data (Williams and Young, 2007; Henderson et al., 2014; Wen et al., 2016). The use of state annotations allows a cleaner separation of the reasoning and natural language aspects of dialogues, but our end-to-end approach makes data collection cheaper and allows tasks where it is unclear how to annotate state. Bordes and Weston (2016) explore end-to-end goal orientated dialogue with a supervised model—we show improvements over supervised learning with goal-based training and decoding. Recently, He et al. (2017) use task-specific rules to combine the task input and dialogue history into a more structured state representation than ours.

Reinforcement learning (RL) has been applied in many dialogue settings. RL has been widely used to improve dialogue managers, which manage transitions between dialogue states (Singh et al., 2002; Pietquin et al., 2011; Rieser and Lemon, 2011; Gašic et al., 2013; Fatemi et al., 2016). In contrast, our end-to-end approach has no explicit dialogue manager that can be updated in isolation, and we found it necessary to interleave RL and supervised learning to avoid RL reducing the quality of language generation. Li et al. (2016) improve metrics such as diversity for non-goal-orientated dialogue using RL, which would make an interesting extension to our work. Das et al. (2017) use reinforcement learning to improve cooperative bot-bot dialogues. RL has also been used to allow agents to invent new languages (Das et al., 2017; Mordatch and Abbeel, 2017). To our knowledge, our model is the first to use RL to improve the performance of an end-to-end goal orientated dialogue system in dialogues with humans.

Work on learning end-to-end dialogues has concentrated on 'chat' settings, without explicit goals (Ritter et al., 2011; Vinyals and Le, 2015; Li et al., 2015). These dialogues contain a much greater diversity of vocabulary than our domain, but do not have the challenging adversarial elements. Such models are notoriously hard to evaluate (Liu et al., 2016), because the huge diversity of reasonable responses, whereas our task has a clear objective. Our end-to-end approach would also be much more straightforward to integrate into a general-purpose dialogue agent than one that relied on annotated dialogue states (Dodge et al., 2016).

There is a substantial literature on multi-agent bargaining in game-theory, e.g. Nash Jr (1950). There has also been computational work on modelling negotiations (Baarslag et al., 2013)—our work differs in that agents communicate in unrestricted natural language, rather than pre-specified symbolic actions, and our focus on improving performance relative to humans rather than other automated systems. Our task is based on that of DeVault et al. (2015), who study natural language negotiations for pedagogical purposes—their version includes speech rather than textual dialogue, and embodied agents, which would make interesting extensions to our work. The only automated natural language negotiations systems we are aware of have first mapped language to domain-specific logical forms, and then focused on choosing the next dialogue act (Rosenfeld et al., 2014; Cuayáhuitl et al., 2015; Keizer et al., 2017). Our end-to-end approach is the first to learn comprehension, reasoning and generation skills in a domain-independent data driven way.

Our use of a combination of supervised and reinforcement learning for training, and stochastic rollouts for decoding, builds on strategies used in game playing agents such as AlphaGo (Silver et al., 2016). Our work is a step towards real-world applications for these techniques. Our use of rollouts could be extended by choosing the other agent's responses based on sampling, using Monte Carlo Tree Search (MCTS) (Kocsis and Szepesvári, 2006). However, our setting has a higher branching factor than in domains where MCTS has been successfully applied, such as Go (Silver et al., 2016)—future work should explore scaling tree search to dialogue modelling.

9 Conclusion

We have introduced end-to-end learning of natural language negotiations as a task for AI, arguing that it challenges both linguistic and reasoning skills while having robust evaluation metrics. We gathered a large dataset of human-human negotiations, which contain a variety of interesting tactics. We have shown that it is possible to train dialogue agents end-to-end, but that their ability can be much improved by training and decoding to maximise their goals, rather than likelihood. There remains much potential for future work, particularly in exploring other reasoning strategies, and in improving the diversity of utterances without diverging from human language. We will also explore other negotiation tasks, to investigate whether models can learn to share negotiation strategies across domains.

Acknowledgments

We would like to thank Luke Zettlemoyer and the anonymous EMNLP reviewers for their insightful comments, and the Mechanical Turk workers who helped us collect data.

References

Nicholas Asher, Alex Lascarides, Oliver Lemon, Markus Guhe, Verena Rieser, Philippe Muller, Stergos Afantenos, Farah Benamara, Laure Vieu, Pascal Denis, et al. 2012. Modelling Strategic Conversation: The STAC project. *Proceedings of SemDial*, page 27.

Tim Baarslag, Katsuhide Fujita, Enrico H Gerding, Koen Hindriks, Takayuki Ito, Nicholas R Jennings, Catholijn Jonker, Sarit Kraus, Raz Lin, Valentin Robu, et al. 2013. Evaluating Practical Negotiating

Agents: Results and Analysis of the 2011 International Competition. *Artificial Intelligence*, 198:73–103.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*.

Antoine Bordes and Jason Weston. 2016. Learning End-to-End Goal-oriented Dialog. *arXiv preprint arXiv:1605.07683*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-decoder Approaches. *arXiv preprint arXiv:1409.1259*.

Heriberto Cuayáhuitl, Simon Keizer, and Oliver Lemon. 2015. Strategic Dialogue Management via Deep Reinforcement Learning. *arXiv preprint arXiv:1511.08099*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2016. Visual Dialog. *arXiv preprint arXiv:1611.08669*.

Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. 2017. Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning. *arXiv preprint arXiv:1703.06585*.

David DeVault, Johnathan Mell, and Jonathan Gratch. 2015. Toward Natural Turn-taking in a Virtual Human Negotiation Agent. In *AAAI Spring Symposium on Turn-taking and Coordination in Human-Machine Interaction. AAAI Press, Stanford, CA*.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander H. Miller, Arthur Szlam, and Jason Weston. 2016. Evaluating Prerequisite Qualities for Learning End-to-End Dialog Systems. *ICLR*, abs/1511.06931.

Mehdi Fatemi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman. 2016. Policy Networks with Two-stage Training for Dialogue Systems. *arXiv preprint arXiv:1606.03152*.

Chaim Fershtman. 1990. The Importance of the Agenda in Bargaining. *Games and Economic Behavior*, 2(3):224–238.

Milica Gašic, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2013. POMDP-based Dialogue Manager Adaptation to Extended Domains. In *Proceedings of SIGDIAL*.

Jonathan Gratch, Zahra Nazari, and Emmanuel Johnson. 2016. The Misrepresentation Game: How to win at negotiation while seeming like a nice guy. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 728–737. International Foundation for Autonomous Agents and Multiagent Systems.

H. He, A. Balakrishnan, M. Eric, and P. Liang. 2017. Learning Symmetric Collaborative Dialogue Agents with Dynamic Knowledge Graph Embeddings. In *Association for Computational Linguistics (ACL)*.

Matthew Henderson, Blaise Thomson, and Jason Williams. 2014. The Second Dialog State Tracking Challenge. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, volume 263.

Simon Keizer, Markus Guhe, Heriberto Cuayáhuitl, Ioannis Efstathiou, Klaus-Peter Engelbrecht, Mihai Dobre, Alexandra Lascarides, and Oliver Lemon. 2017. Evaluating Persuasion Strategies and Deep Reinforcement Learning methods for Negotiation Dialogue agents. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL 2017)*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based Monte-Carlo Planning. In *European conference on machine learning*, pages 282–293. Springer.

Adam Lerer and Alexander Peysakhovich. 2017. Maintaining Cooperation in Complex Social Dilemmas using Deep Reinforcement Learning. *arXiv preprint arXiv:1707.01068*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A Diversity-promoting Objective Function for Neural Conversation Models. *arXiv preprint arXiv:1510.03055*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep Reinforcement Learning for Dialogue Generation. *arXiv preprint arXiv:1606.01541*.

Chia-Wei Liu, Ryan Lowe, Iulian V. Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Junhua Mao, Xu Wei, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan L. Yuille. 2015. Learning Like a Child: Fast Novel Visual Concept Learning From Sentence Descriptions of Images. In *The IEEE International Conference on Computer Vision (ICCV)*.

Igor Mordatch and Pieter Abbeel. 2017. Emergence of Grounded Compositional Language in Multi-Agent Populations. *arXiv preprint arXiv:1703.04908*.

John F Nash Jr. 1950. The Bargaining Problem. *Econometrica: Journal of the Econometric Society*, pages 155–162.

Yurii Nesterov. 1983. A Method of Solving a Convex Programming Problem with Convergence Rate O (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376.

Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. 2011. Sample-efficient Batch Reinforcement Learning for Dialogue Management Optimization. *ACM Trans. Speech Lang. Process.*, 7(3):7:1–7:21.

Verena Rieser and Oliver Lemon. 2011. *Reinforcement Learning for Adaptive Dialogue Systems: A Data-driven Methodology for Dialogue Management and Natural Language Generation*. Springer Science & Business Media.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven Response Generation in Social Media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 583–593. Association for Computational Linguistics.

Avi Rosenfeld, Inon Zuckerman, Erel Segal-Halevi, Osnat Drein, and Sarit Kraus. 2014. NegoChat: A Chat-based Negotiation Agent. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 525–532, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. *Journal of Artificial Intelligence Research*, 16:105–133.

Victoria Talwar and Kang Lee. 2002. Development of lying to conceal a transgression: Children's control of expressive behaviour during verbal deception. *International Journal of Behavioral Development*, 26(5):436–444.

David Traum, Stacy C. Marsella, Jonathan Gratch, Jina Lee, and Arno Hartholt. 2008. Multi-party, Multi-issue, Multi-strategy Negotiation for Multi-modal Virtual Agents. In *Proceedings of the 8th International Conference on Intelligent Virtual Agents*, IVA '08, pages 117–130, Berlin, Heidelberg. Springer-Verlag.

Oriol Vinyals and Quoc Le. 2015. A Neural Conversational Model. *arXiv preprint arXiv:1506.05869*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A Network-based End-to-End Trainable Task-oriented Dialogue System. *arXiv preprint arXiv:1604.04562*.

Jason D Williams and Steve Young. 2007. Partially Observable Markov Decision Processes for Spoken Dialog Systems. *Computer Speech & Language*, 21(2):393–422.

Ronald J Williams. 1992. Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning. *Machine learning*, 8(3-4):229–256.

Agent-Aware Dropout DQN for Safe and Efficient On-line Dialogue Policy Learning

Lu Chen and **Xiang Zhou** and **Cheng Chang** and **Runzhe Yang** and **Kai Yu**
Key Lab. of Shanghai Education Commission for Intelligent Interaction and Cognitive Eng.
SpeechLab, Department of Computer Science and Engineering
Brain Science and Technology Research Center
Shanghai Jiao Tong University, Shanghai, China
{chenlusz,owenzx,cheng.chang,yang_runzhe,kai.yu}@sjtu.edu.cn

Abstract

Hand-crafted rules and reinforcement learning (RL) are two popular choices to obtain dialogue policy. The rule-based policy is often reliable within predefined scope but not self-adaptable, whereas RL is evolvable with data but often suffers from a bad initial performance. We employ a *companion learning* framework to integrate the two approaches for *on-line* dialogue policy learning, in which a predefined rule-based policy acts as a teacher and guides a data-driven RL system by giving example actions as well as additional rewards. A novel *agent-aware dropout* Deep Q-Network (AAD-DQN) is proposed to address the problem of when to consult the teacher and how to learn from the teacher's experiences. AAD-DQN, as a data-driven student policy, provides (1) two separate experience memories for student and teacher, (2) an uncertainty estimated by dropout to control the timing of consultation and learning. Simulation experiments showed that the proposed approach can significantly improve both *safety* and *efficiency* of on-line policy optimization compared to other companion learning approaches as well as supervised pre-training using static dialogue corpus.

1 Introduction

A task-oriented spoken dialogue system (SDS) is a system that can continuously interact with a human to accomplish a predefined task through speech. Dialogue manager, which maintains the dialogue state and decides how to respond, is the core of an SDS. In this paper, we focus on the dialogue policy.

At the early research, the spoken dialogue systems assume observable dialogue states. Dialogue policy is simply a set of hand-crafted mapping rules from state to machine action. This is referred to as rule-based policy, which often has acceptable performance but has no ability of self-adaption. Nowadays rule-based policy is popular in commercial dialogue systems.

However, in real world scenarios, unpredictable user behavior, inevitable automatic speech recognition, and spoken language understanding errors make it difficult to maintain the true dialogue state and make the decision. Hence, in recent years, there is a research trend towards statistical dialogue management. A well-founded theory for this is the partially observable Markov decision process (POMDP) (Kaelbling et al., 1998), which can provide robustness to errors from the input module and automatic policy optimization by reinforcement learning. Most POMDP based policy learning research is usually carried out using either user simulator or employed users (Williams and Young, 2007; Young et al., 2010). The trained policy is not guaranteed to work well in real world scenarios. Therefore, on-line policy training has been of great interest (Gašić et al., 2011). Recently, Chen et al. (2017) proposed two qualitative metrics [1] to measure on-line policy learning: *safety* and *efficiency*. Safety reflects whether the initial policy can satisfy the quality-of-service requirement in real-world scenarios during the on-line policy learning period. Efficiency reflects how long it takes for the on-line policy training algorithm to reach a satisfactory performance level.

Most traditional RL-based policy training suf-

[1] The quantitative evaluation metrics of safety and efficiency are proposed in section 4.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2454–2464
Copenhagen, Denmark, September 7–11, 2017.

fers poor initial performance, i.e. causes the safety problem. In light of above, Chen et al. (2017) proposed a safe and efficient on-line policy optimization framework, i.e. *companion teaching* (CT), in which a human teacher is added in the classic POMDP. The teacher has two missions: one is to show example actions, another is to act as a critic to give the student extra reward which can make the learning of policy more efficient. The example actions not only make the learning safer but also can be directly used by the training of the student policy. However, there are costs to the teaching of a human teacher.

Based on CT, *companion learning* (CL) framework is proposed to integrate rule-based policy and RL-based policy, resulting in *safe* and *efficient* on-line policy learning. Here, the rule-based policy acts as a virtual teacher which replaces the human teacher in CT. There are a few differences between these two kinds of teachers. First, because it has no marginal cost when it's deployed, the rule teacher can be consulted at any time if needed. On the other hand, the rule policy is not as good as the human teacher, therefore it's important to determine when and how much the student policy depends on the rule teacher. Here, we propose an *agent-aware dropout* Deep Q-Network (AAD-DQN) as the student statistical policy, which provides (1) two separate experience replay pools for student and teacher, (2) an uncertainty estimated by dropout which can be used to control the timing of consultation and learning.

In summary, our main contributions are three-folds: (1) *Companion learning* (CL) framework was proposed to integrate rule-based policy and RL-based policy. (2) An *agent-aware dropout* Deep Q-Network (AAD-DQN) was proposed as the statistical student policy. (3) Compared with other companion teaching approaches (Chen et al., 2017) as well as supervised pre-training using static dialogue corpus (Fatemi et al., 2016), CL with AAD-DQN can achieve better performance.

2 Related Work

Most previous studies of on-line policy learning have been focused on the *efficiency* issue, such as Gaussian Process Reinforcement Learning (GPRL) (Gašić et al., 2010). In GPRL, the kernel function defines prior correlations of the objective function given different belief states, which can significantly speed up the policy learning (Gašić and Young, 2014). Alternative methods include Kalman temporal difference reinforcement learning (Pietquin et al., 2011).

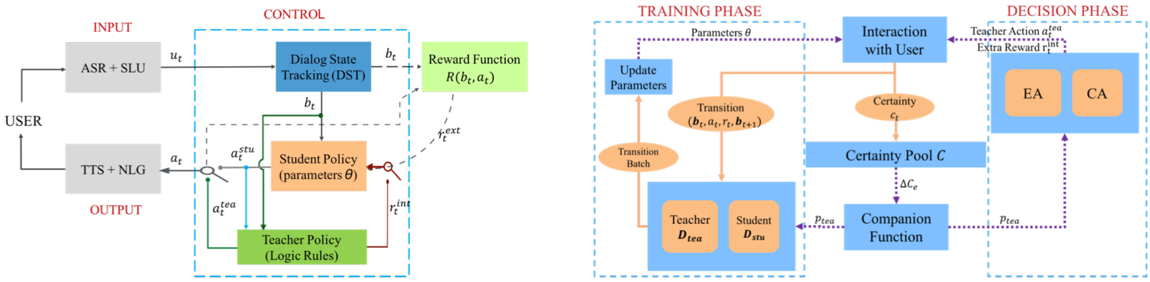
More recently, deep reinforcement learning (DRL) (Mnih et al., 2015) is applied in dialogue policy optimization, including deep Q-Network (DQN) (Cuayáhuitl et al., 2015; Fatemi et al., 2016; Zhao and Eskenazi, 2016; Lipton et al., 2016) and policy gradient (PG) methods, e.g. REINFORCE (Williams and Zweig, 2016; Su et al., 2016; Williams et al., 2017), Advantage Actor-Critic (A2C) (Fatemi et al., 2016). In order to speed up the learning of DQN, Lipton et al. (2016) proposed an efficient exploration technique based on Thompson sample from a Bayesian neural network. Furthermore, they showed that using a few successful dialogues generated by a rule-based policy to pre-fill the replay buffer can benefit the learning at the beginning. To improve the efficiency of PG methods, policy network is initialized with supervised learning (SL) before RL training (Williams and Zweig, 2016; Williams et al., 2017; Su et al., 2016, 2017; Fatemi et al., 2016), which is similar to the idea in (Silver et al., 2016). However, combining RL with SL for dialogue policy optimization is not new. Henderson et al. (2008) were among the first to prove the benefits of combining supervised and reinforcement learning. In the experiments, we will compare CL with these pre-training methods.

Although the improvement of efficiency can benefit the safety of learning process, no matter how efficient the algorithm is, an unsafe on-line learned policy can lead to bad user experience at the beginning of learning period and consequently fail to attract sufficient real users to continuously improve the policy. Therefore, it is important to address the *safety* issue. There are few works about the safety issue of on-line dialogue policy optimization. Williams (2008) proposed a method for integrating business rules and POMDPs. The rules act as the action mask, i.e. the rules nominate a set of one or more actions, and the POMDP chooses the optimal action.

3 Proposed Framework

3.1 Companion Learning for On-line Policy Optimization

In the CL framework, there are two agents: one is the student policy, another is the teacher policy. Here, *teacher policy* is the extra part com-



(a) **(b)**

Figure 1: (a) RL-based Companion Learning(CL) Framework with Logic Rules in an SDS. (b) Agent-Aware Dropout DQN (AAD-DQN) for CL.

pared with the classic statistical dialogue manager architecture (Young et al., 2013). The goal of online policy training is to optimize the student policy from data via interaction with users in real scenarios. The teacher guides the policy learning at each turn as a companion of the dialogue policy, hence, referred to as *companion learning* [2]. The CL framework is described in Figure 1(a).

At each turn, the input module (ASR and SLU) receives an acoustic input signal from the human user and the dialogue state tracker keeps the dialogue state up-to-date. The dialogue state is then transmitted to both the student policy and the teacher policy. The student policy first generates a candidate action a_t^{stu} and when it needs help from the teacher policy, it sends a_t^{stu} with some auxiliary information which will be transmitted to the teacher. The teacher policy can then help the student policy with one of the following ways or both:

- **Example Action (EA)**: The teacher generates an action a_t^{tea} instead of a_t^{stu} according to its policy. It corresponds to the left switch in Figure 1(a).
- **Critic Advice (CA)**: The teacher will not explicitly show an action. Instead, it gives an extra reward r_t^{int} to the student policy. It corresponds to the right switch in Figure 1(a).

The action from control module is then transmitted to the output module, which generates the nature text and audio. At each turn, an extrinsic reward signal r_t^{ext} will be given to the student policy by the environment, i.e. the user. The extrinsic reward r_t^{ext} with the extra intrinsic reward r_t^{int} will be used to update the policy parameters θ using reinforcement learning algorithms.

In the CL framework, there are two things that matter: one is when to consult the teacher, another is how to use the teacher's experiences. In this paper, an *agent-aware dropout* DQN (AAD-DQN) is proposed. As shown in Figure 1(b), the certainty information during the interaction is used to define a *companion function*, which controls how often to sample the teacher's experiences for updating parameters during the training phase (left), and when to use EA or CA teaching method during decision phase (right).

The rest of this section is organized as follows. The next subsection introduces the *agent-aware* experience replay in DQN. The definition of certainty in DQN and the companion function are presented in subsection 3.3. The rule-based teacher policy is described in subsection 3.4.

3.2 Agent-Aware Experience Replay in DQN

A Deep Q-Network (DQN) is a multi-layer neural network which maps a belief state $\mathbf{b}_t$ to the Q values of the possible actions a_t at that state, $Q(\mathbf{b}_t, a_t; \theta)$, where θ is the weight vector of the neural network. Neural networks for the approximation of value functions have long been investigated (Lin, 1993). However, these methods were previously quite unstable (Mnih et al., 2013). In DQN, Mnih et al. (2013, 2015) proposed two techniques to overcome this instability, namely experience replay and the use of a target network.

At every turn, the transition including the previous belief state $\mathbf{b}_t$, previous action a_t, corresponding reward r_t and current belief state $\mathbf{b}_{t+1}$ is put in a finite pool (Lin, 1993). In this pa-

[2] The name *companion learning* has another potential meaning that the agents can learn from each other, i.e. the rules guide the RL training, and the optimised RL policy can provide some intuition for the revision of rules. We will give some preliminary discussions about this point in section 5.3.

per, two pools $\mathcal{D}_{stu}$ and $\mathcal{D}_{tea}$ are used to store the student's experiences and the teacher's experiences respectively as shown in Figure 1(b). When the teaching method EA is used in the t-th turn, $a_t = a_t^{tea}$ and the transition is put in $\mathcal{D}_{tea}$, otherwise $a_t = a_t^{stu}$ and the transition is put in $\mathcal{D}_{stu}$. When CA is used, $r_t = r_t^{ext} + r_t^{int}$, otherwise $r_t = r_t^{ext}$. Once any of the pool has reached its predefined maximum size, adding a new transition results in deleting the oldest transition in the pool. During training, a pool is first selected from $\mathcal{D}_{tea}$ and $\mathcal{D}_{stu}$. The probability of selecting $\mathcal{D}_{tea}$ is p_{tea}, i.e. $\mathcal{D} \sim Ber(\mathcal{D}_{tea}, \mathcal{D}_{stu}; p_{tea})$ [3].Then a mini-batch of transitions is uniformly sampled from the selected pool, i.e. $(\mathbf{b}_t, a_t, r_t, \mathbf{b}_{t+1}) \sim U(\mathcal{D})$. We call this *agent-aware* experience replay.

Except for the experience replay, a target network with weight vector θ^- is used. This target network is similar to the Q-network except that its weights are only copied every K steps from the Q-network, and remain fixed during all the other steps. The loss function for the Q-network at each iteration takes the following form:

$$L(\theta) = \mathrm{E}_{\mathcal{D}\sim Ber(\mathcal{D}_{tea},\mathcal{D}_{stu};p_{tea}),\ (\mathbf{b}_t,a_t,r_t,\mathbf{b}_{t+1})\sim U(\mathcal{D})} \left[\left(r_t + \gamma \max_{a_{t+1}} Q(\mathbf{b}_{t+1}, a_{t+1}; \theta^-) - Q(\mathbf{b}_t, a_t; \theta) \right)^2 \right] \tag{1}$$

where $\gamma \in [0, 1]$ is the discount factor.

The probability p_{tea} controls how often the student learns from the teacher's experiences. As the learning goes on, the probability will decrease. More details will be described in the next section.

3.3 Companion Strategy

It's important for the student to estimate an appropriate point to end the reliance on the teacher. If the reliance is ended too early, the student itself may not reach an acceptable performance, resulting in the sharp drop of performance, which is the *safety* problem. However, if the student always relies on the teacher, it's hard to improve its performance to surpass the teacher's performance, which is the *efficiency* problem.

We get some inspirations from the studying process of a call center service agent. Consider how a new call center service agent gets started. At first, an experienced call center agent tells him some basic *rules* and the new agent works by often consulting these rules. His confidence about how to make decisions gradually increases during the continuous practice. Eventually, he is so confident about his own decisions that he no longer needs any consultation to these rules and even explores some better response ways through interaction with users which are not initially included in the rules. Similarly, we can use the uncertainty/certainty of the Q-network to determine the teaching time.

There are several methods to estimate the uncertainty/certainty in deep neural networks, e.g. Bayesian neural networks (Blundell et al., 2015), dropout (Gal and Ghahramani, 2016), bootstrap (Osband et al., 2016) . Here we use the dropout to estimate the certainty of Q-Network. We call this Q-network DropoutQNetwork. Dropout is a technique used to avoid over-fitting in neural networks. It was introduced several years ago by (Hinton et al., 2012) and studied more extensively in (Srivastava et al., 2014). When dropout is used in training, the elements of the output of each hidden layer $\mathbf{h}$ is randomly set to zero with probability p, i.e. $\mathbf{h}' = \mathbf{h} \odot \mathbf{z}$ [4] where $\mathbf{z}$ is binary vector and each element $z_i \sim Ber(1-p)$. $\mathbf{h}'$ is scaled by $\frac{1}{1-p}$ and then fed to the next layer. At test time the dropout is disabled, i.e. the output of each hidden layer $\mathbf{h}$ is directly fed to the next layer. Although dropout was suggested as an ad-hoc technique, recently it was theoretically proven that the dropout training in deep neural networks is an approximate Bayesian inference in deep Gaussian processes (Gal and Ghahramani, 2016). Therefore, a direct result of this theory gives us tools to model uncertainty with dropout neural networks. To obtain the uncertainty, similar with that at train phrase the dropout is enabled at test phrase. For each input instance (i.e. dialogue belief state) $\mathbf{b}_t$, performing N stochastic forward passes through the network and averaging the output $\mathbf{q}_i \triangleq [q_{i1}, \cdots, q_{iM}]$ to get the mean and the variance. Generally, the variance can be utilized to measure the uncertainty of output. However, it's not a normalized criteria, and it's hard to set a threshold below which we should be confident with the output.

Instead, we proposed a novel method to measure the certainty of the decision of student policy at t-th turn. For each stochastic forward passes, the action $a_{ti} = \arg\max_j q_{ij}$ is regarded as a vote. After N passes [5], there is a committee

[3] Ber is short for Bernoulli.

[4] Here $\odot$ is the element-wise product.

[5] The N forward passes can be done in parallel, e.g. the

$\{a_{t1}, \cdots, a_{tN}\}$ consisting of N votes. The action a_t^{stu} that should be taken in the belief state $\mathbf{b}_t$ is the one with the largest percentage of the votes, and the corresponding percentage is defined as certainty c_t. The process is described in Algorithm 1.

Algorithm 1 The Decision Procedure of Student Policy $\pi^{stu}(\mathbf{b}_t, N)$

```
Require:
    The repeat times $N$ and the belief state $\mathbf{b}_t$
Initial the probability vector $\mathbf{p} = [p_1, \cdots, p_M]$ with zero vector, where $M$ is the number of actions.
for $i = 1, N$ do
    $\mathbf{q}_i \leftarrow \mathsf{DropoutQNetwork}(\mathbf{b}_t)$
    $a_{ti} \leftarrow \arg\max_j q_{ij}$
    $\mathbf{p}[a_{ti}] \leftarrow \mathbf{p}[a_{ti}] + 1/N$
end for
$c_t \leftarrow \max_j p_j$
$a_t^{stu} \leftarrow \arg\max_j p_j$
return $a_t^{stu}$, $c_t$
```

At the end of e-th dialogue, the average certainty of all turns is computed, i.e. $C_e = \frac{1}{T_e}\sum_{t=0}^{T_e} c_t$, where T_e is the number of turns in e-th dialogue. Generally, the variance of C_e between successive dialogues is high. In order to the smooth the estimation, here we use the moving average of C_e in previous W dialogues to represent the certainty of student at current dialogue, i.e.

$$\overline{C}_e = \frac{1}{W}\sum_{i=e-W}^{e-1} C_i. \qquad (2)$$

As the training goes on, $\overline{C}_e$ grows until it converges. If $\overline{C}_e$ in all successive W dialogues are greater then a threshold C_{th} as shown in Figure 2, it's assumed that the student reaches a point where it is confident enough with its own decision steadily. Therefore, the teaching, both EA and CA, should be ended from now on.

Before the end of the teaching, CA is done in all turns. However, if EA is always done, the disappearance of the teacher may cause a dramatic change in the hybrid decision policy, which results in a sharp drop of performance. To deal with this issue, a monotonically increasing function of the relative certainty $P_{tea}(\Delta C_e)$ is proposed to control the frequency of EA teaching. ΔC_e represents the distance between $\overline{C}_e$ and C_{th}, i.e. $\Delta C_e = \max(0, C_{th} - \overline{C}_e)$. The effect of $P_{tea}(\Delta C_e)$ is that the closer $\overline{C}_e$ is to C_{th}, the more unlikely EA teaching is executed. Besides controlling how often the student directly consult the teacher, another mission of $P_{tea}(\Delta C_e)$ is to control how often the teacher's experiences are replayed, i.e. the probability p_{tea} described in section 3.2. Implementation details of $P_{tea}(\Delta C_e)$ are described in Appendix C.

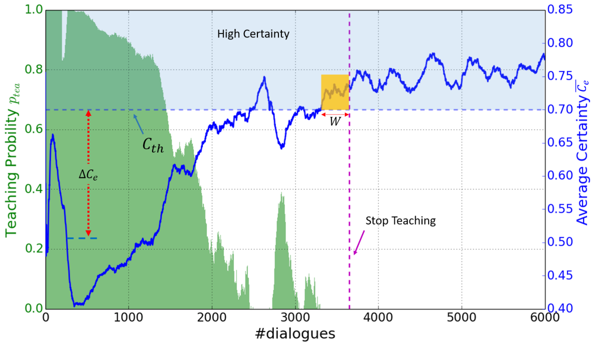


Figure 2: Illustration of average certainty $\overline{C}_e$ and the probability p_{tea}.

The full procedure of companion learning with logic rules is described in Algorithm 2.

3.4 Teacher Policy: Logic Rules

Rule-based policy is popular in commercial dialogue systems (Williams, 2008). The policy, i.e. the dialogue plan/flow, is designed by a domain expert. His knowledge of task domain and business rules is encoded in the rules. There are many methods to represent the decision rules, e.g. propositional logic, first-order logic, decision tree. Here, we use the ordered propositional logic rules, which can be easily translated into IF-THEN rules. When making the decision, these rules are executed in pre-defined order. If the conditions of any rule are satisfied, the decision process will be terminated and the output is the corresponding action. In this paper, three hand-crafted logic rules, R1, R2, and R3 , were used as the teacher:

- R1: confirm the most likely value in slots where the most likely value has probability between 0.1 and 0.6[6];
- R2: offer a restaurant if there is at least one slot in which the belief of most likely value is more than the belief of special value "*none*";

dialogue state can be repeated N times to form a mini-batch, then one forward is executed to get N outputs simultaneously.

[6] This threshold is the best one we have tried.

Algorithm 2 Companion Learning with Logic Rules

```
Require:
    The number of stochastic forward pass $N$, the maximal extra reward $\delta > 0$.
Initialize the parameters $\theta$ of student policy
Initialize replay pools $\mathcal{D}_{tea}$ and $\mathcal{D}_{stu}$ with $\{\}$, certainty memory $\mathcal{C}$ with $\{\}$, $teaching$ with $True$.
for $e = 1, E$ do
    Update the dialogue belief state $\mathbf{b}_0$
    Initialize the average certainty $C_e \leftarrow 0$
    if $teaching$ is $True$ then
        $teaching, p_{tea} \leftarrow \mathsf{Companion}(\mathcal{C})$
    end if
    for $t = 0, T_e$ do
        Set intrinsic reward $r_t^{int} \leftarrow 0$
        Get system action and the corresponding certainty, i.e. $a_t^{stu}, c_t \leftarrow \pi^{stu}(\mathbf{b}_t, N)$
        $C_e \leftarrow C_e + c_t$
        Get action from the rule-based policy, i.e. $a_t^{tea} \leftarrow \pi^{tea}(\mathbf{b}_t)$
        $EA \sim Ber(p_{tea})$
        if $teaching$ is $True$ and $EA$ is $True$ then
            $a_t \leftarrow a_t^{tea}$
        else
            $a_t \leftarrow a_t^{stu}$
        end if
        if $teaching$ is $True$ then
            $r_t^{int} \leftarrow (2 \times \mathbf{1}\{a_t = a_t^{tea}\} - 1)\delta$
        end if
        $C_e \leftarrow \frac{1}{T_e} C_e$, and store $C_e$ in $\mathcal{C}$
        Give the action $a_t$ to the environment, observe the extrinsic reward $r_t^{ext}$ and update the dialogue belief state $\mathbf{b}_{t+1}$
        $r_t \leftarrow r_t^{int} + r_t^{ext}$
        if $EA$ is $True$ then
            Store $\{\mathbf{b}_t, a_t, r_t, \mathbf{b}_{t+1}\}$ in $\mathcal{D}_{tea}$
        else
            Store $\{\mathbf{b}_t, a_t, r_t, \mathbf{b}_{t+1}\}$ in $\mathcal{D}_{stu}$
        end if
        Update the parameters $\theta$ of DropoutQNetwork according to the equation (1).
    end for
end for
return $\theta$
```

Algorithm 3 Companion Function $\mathsf{Companion}(\mathcal{C})$

```
Require:
    The average certainty memory $\mathcal{C}$ at $e$-th dialogue and the moving window size $W$.
Initialize $teaching$ with $False$, $p_{tea}$ with 0
for $i = 0, W$ do
    Compute the moving average certainty $\overline{C}_{e-i}$ in ($e$-$i$)-th dialogue with equation (2).
    if $\overline{C}_{e-i} < C_{th}$ then
        $teaching \leftarrow True$
        break
    end if
end for
if $teaching$ is $True$ then
    $\Delta C_e \leftarrow \max(0, C_{th} - \overline{C}_e)$
    $p_{tea} \leftarrow P_{tea}(\Delta C_e)$
end if
return $teaching, p_{tea}$
```

- R3: request values for a slot which is uniformly selected from a pre-defined slot list.

The corresponding pseudo-codes are presented in Appendix B.

4 Evaluation Metrics of On-line Policy Optimization

Most previous work on the evaluation of RL-based dialogue policy optimization focuses on the final performance (FP) when the system converges to a steady level. However, for on-line policy optimization, it's important to measure the learning process. Except for FP, we proposed two quantitative metrics: safety loss and efficiency loss.

4.1 Safety Loss

In the on-line training process, unless the performance of the system reaches the acceptable performance S_a, the interaction between users and the system will be unsafe and causes trouble to continuing training. So the safety of the system is defined to be the system's ability to maintain performance above the acceptable performance S_a.

We quantify the safety loss of the system by summing up the performance gap between the acceptable performance and the system performance S_e in every episode during the on-line learning. Suppose there are E dialogues, then $L_1 = \sum_{e=1}^{E} \max(0, S_a - S_e)$. The safety loss has an intuitive interpretation as the area of the region below the threshold and above training curve. This metric is similar to the integral of absolute error (IAE) (Shinners, 1998) metric commonly adopted in the evaluation of control systems (Gaing, 2004; Jesus and Tenreiro MacHado, 2008).

4.2 Efficiency Loss

Another important issue of on-line learning is efficiency. The efficiency indicates the speed at which the system reaches a specific performance level. In reality, we can tolerate a system to make mistakes at the beginning but it should improve at a significant speed until reaching the ideal performance S_i. Therefore, later failures should weight more than early failures to evaluate efficiency. Similar to the integral of time multiplied by absolute error (ITAE) (Shinners, 1998) metric, we propose a metric efficiency loss. We multiply the performance gap between ideal performance and current performance with the episode index, thus giving later failure greater penalty. Specifically, $L_2 = \sum_{e=1}^{E} \max(0, S_i - S_e)e$.

More illustrations about safety loss and efficiency loss are given in Appendix D.

5 Experiments

Our experiments have three objectives: (1) Comparing our proposed *dropout* DQN in Algorithm 1 with some baselines when there is no teacher. (2) Comparing CL with other two baselines when the teacher gets involved, and investigating the benefits of our proposed *agent-aware* experience replay. (3) Visually analyzing the differences in behaviors between the rule-based teacher policy and the optimized student policy.

An agenda-based user simulator (Schatzmann et al., 2007a) with error model (Schatzmann et al., 2007b) was implemented to emulate the behavior of the human user, and a rule-based policy with 0.695 success rate described in section 3.2 was used as the teacher in our experiments. The purpose of the user's interacting with SDS is to find restaurant information in the Cambridge (UK) area (Henderson and Thomson, 2014). This domain has 7 slots of which 4 can be used by the system to constrain the database search. The summary action space consists of 16 summary actions. More details are described in Appendix A.

For reward, at each turn, an extrinsic reward of -0.05 is given to the student policy. At the end of the dialogue, a reward of $+1$ is given for dialogue success. The maximal extra reward δ is 0.05.

For each set-up, 10000 dialogues are used for training, the moving dialogue success rate is recorded with a window size of 1000. The final results are the average of 40 runs.

5.1 Policy Learning without Teaching

In this section, four policies without teaching are compared:

- `DQN`: A vanilla deep Q-Network (Mnih et al., 2015) which has two hidden layers, each with 128 nodes.
- `A2C`: An advantage actor-critic policy which consists of an actor network and a critic network (Fatemi et al., 2016).
- `Dropout_DQN_1` and `Dropout_DQN_32`: They both have a dropout layer after each hidden layer. The dropout rate is 0.2. Their difference is that the number of stochastic forward pass N of `Dropout_DQN_32` in Algorithm 1 is 32, while that of `Dropout_DQN_1` is 1. `Dropout_DQN_1` makes decision according to one output of Q-network similar to that of vanilla `DQN`. `Dropout_DQN_1` was first proposed in (Gal and Ghahramani, 2016), and was confirmed that `Dropout_DQN_1` can obtain more efficient exploration.

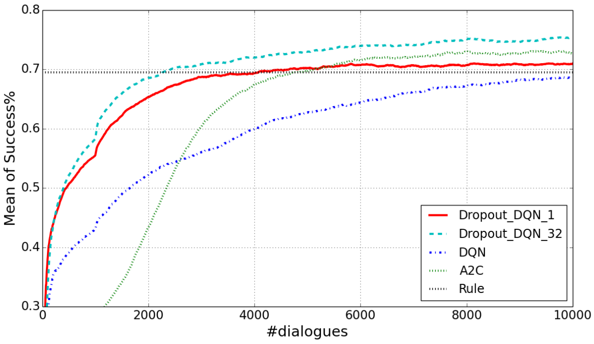


Figure 3: Comparison of four policies without teaching.

The learning curves are described in Figure 3 and the evaluation results are described in Table 1. Comparing `Dropout_DQN_1` with `DQN` in figure 3, the improvement of efficiency caused by

Metrics	**No Teaching**				**Teaching**			
	DQN	Dropout_DQN_1	Dropout_DQN_32	A2C	A2C_PreTrain	EA	CL_D	CL_AAD
Safety	1043.3	321.4	258.0	1020.8	176.9	48.8	30.6	**18.6**
Efficiency($\times 10^4$)	534.7	249.1	75.6	299.0	205.4	58.0	62.6	**53.5**
FP	0.684	0.709	0.749	0.727	0.726	**0.751**	0.749	**0.751**

Table 1: The quantitative evaluation results of different methods. Here final performance (FP) is the success rate of last 2000 dialogues. The FP of Dropout_DQN_32 0.749 is used as the ideal performance S_i for computing efficiency loss, and the performance of the rules 0.695 is used as the acceptable performance S_a for computing safety loss.

dropout can be observed as claimed in (Gal and Ghahramani, 2016). However, Dropout_DQN_1 seems to suffer premature and sub-optimal convergence, while our proposed Dropout_DQN_32, whose decision is based on multi votes (algorithm 1), can result in improvement of efficiency and better final performance. Moreover, Dropout_DQN_32 also performs much better than the policy gradient method A2C.

For the following experiments, the times of stochastic forward pass N in Algorithm 1 is 32.

5.2 Policy Learning with Teaching

In this section, four methods of teaching by the rule-based policy are compared:

- EA: 500 dialogues are taught with EA at the beginning (Chen et al., 2017).
- A2C_PreTrain: At the beginning, 500 dialogue are collected with rule-based policy. These examples are used to pre-train the actor network with supervised learning. After the pre-training, the policy is continuously optimized with the A2C algorithm (Fatemi et al., 2016).
- CL_AAD: Full CL with AAD-DQN described in section 3.
- CL_D: CL without agent-aware experience repay, i.e. the teacher's experiences and student's experiences are put in one pool and are uniformly sampled for the experience replay in equation (1).

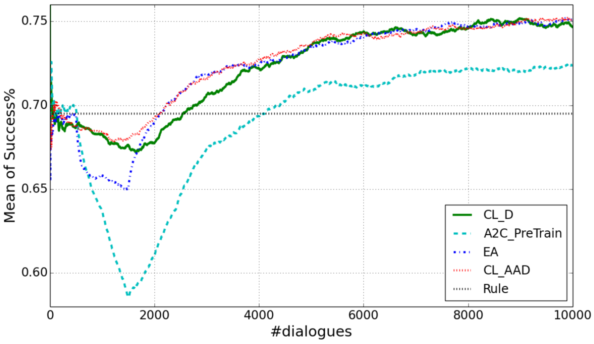


Figure 4: Comparison of four methods with teaching by rule-based teacher.

As can be seen in Figure 4, there is a big dip in the performance of A2C_PreTrain. One possible explanation is that because the rule-based policy is sub-optimal, the pre-training makes the student policy reach a local minimum point. The rl-training should first make it escape from the local minimum point, which results in a temporary loss in performance.

Comparing CL methods (CL_D and CL_AAD) with EA in Figure 4 and in Table 1, we can conclude that CL can significantly boost the safety of learning process. Moreover, except for safety, CL_AAD can boost the efficiency, which benefits from the *agent-aware* experience replay.

5.3 Comparison of Optimized Student Policy and Rule-based Teacher Policy

To interpret what the student has learnt, we further compare the rules and an optimized student policy with 76.7% success rate. The rule-based policy is used to collect 5000 dialogues, while in each turn the decision made by the student policy is also recorded. Figure 5 is a confusion matrix. The x-axis denotes the student's decision and the y-axis denotes the rules' decision. The numbers on the left are the statistics for each action in 5000 dialogues. Each element in the matrix denotes the normalized number of turns when the rule chooses the action in the corresponding line, the student chooses the action in the corresponding column.

As is shown in Figure 5, *offer* and *confirm* are two action types used most frequently. In more than half of turns when the rule-based pol-

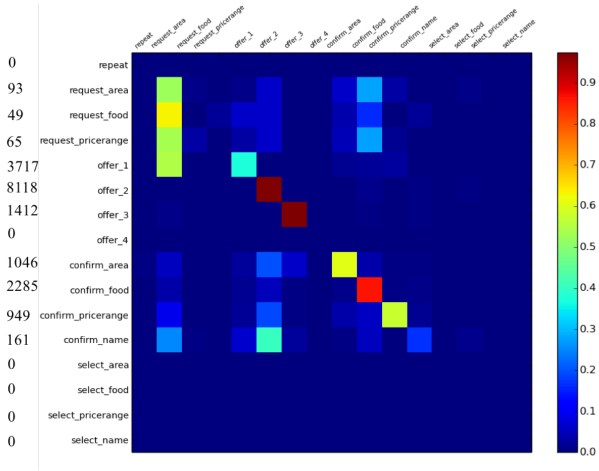


Figure 5: Confusion matrix between the decisions of rule-based policy and the decisions of the optimised student policy. The x-axis denotes the student's decision and the y-axis denotes the rules' decision.

icy chooses *offer_1*, the student policy will choose a different action. Furthermore, from the element in line *offer_1* and column *request_area*, we can find that in this situation the student policy prefers the action *request_area*. Inspired by this disagreement, we designed a new rule:

- R4: request values for slot area when there is only one other slot constraint for the database query.

Similarly, as can be seen in Figure 5, in a considerable proportion of turns when the rule-based policy chooses *confirm_area, confirm_pricerange*, or *confirm_name*, the student policy will choose the action *offer_2*, which may mean that for slots *area, pricerange*, or *name*, when there are values for database query, the system should offer a restaurant instead of confirming the slot-value constraints. Therefore, the rule R1 in section 3.2 was revised as follows:

- R1*: For slot food, confirm the most likely value has the probability between 0.1 and 0.6; For slot area, pricerange and name, confirm the most likely value, the belief of which is smaller than the belief of the special value "*none*" and is larger than 0.1.

Ordered Rules	**Success Rate**	**#Turn**	**Reward**
R1, R2, R3	0.695	4.58	0.4657
R1, R4, R2, R3	0.749	5.16	0.4910
R1*, R2, R3	0.705	4.44	0.4824
R1*, R4, R2, R3	**0.753**	4.98	**0.5042**

Table 2: Evaluation results of different ordered rules. As a reference, the performance of optimised student policy is success rate 0.767, #turn 5.10 and reward 0.5124.

Table 2 is the evaluation results of different ordered rules. The rule R4 can significantly boost the success rate (comparing line 2 with line 1), while the rule R1* can both boost the success rate and decrease the dialogue length (comparing line 3 with line 1). The combination of R4 and R1* takes respective advantages (comparing line 4 with line 1, line 2 and line 3). The performance of final order rules is comparable to the performance of optimized student policy.

It is worth noting that the primary rules R1, R2, and R3 in section 3.2 don't distinguish between different slots. However, the new rules R4 and R1* are all slot-specific, which it is difficult to design at the beginning.

6 Conclusion

This paper has proposed a *companion learning* framework to unify rule-based policy and RL-based policy. Here, the rule-based policy acts as a teacher, which either directly shows example action or gives an extra reward. Based on the uncertainty estimated using a dropout Q-Network, a companion strategy is proposed to control when the student policy directly consults rules and how often the student policy learns from the teacher's experiences. Simulation experiments showed that our proposed framework can significantly improve both *safety* and *efficiency* of on-line policy optimization. Additionally, we visually analyzed the differences in behaviors between the rule-based teacher policy and the optimized student policy, which gave us some inspirations to refine the rules.

Acknowledgments

This work was supported by the Shanghai Sailing Program No. 16YF1405300, the China NSFC projects (No. 61573241 and No. 61603252) and the Interdisciplinary Program (14JCZ03) of Shanghai Jiao Tong University in China. Experiments have been carried out on the PI supercomputer at Shanghai Jiao Tong University.

References

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622.

Lu Chen, Runzhe Yang, Cheng Chang, Zihao Ye, Xiang Zhou, and Kai Yu. 2017. On-line dialogue policy learning with companion teaching. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 198–204.

Heriberto Cuayáhuitl, Simon Keizer, and Oliver Lemon. 2015. Strategic dialogue management via deep reinforcement learning. *NIPS Deep Reinforcement Learning Workshop*.

Mehdi Fatemi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman. 2016. Policy networks with two-stage training for dialogue systems. In *17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 101–110.

Zwe-Lee L Gaing. 2004. A Particle Swarm Optimization Approach for Optimum Design of PID Controller in AVR System. *IEEE Transactions on Energy Conversion*, 19(2):384–391.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*.

Milica Gašić, Filip Jurčíček, Simon Keizer, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. *Gaussian processes for fast policy optimisation of POMDP-based dialogue managers*. Association for Computational Linguistics.

Milica Gašić, Filip Jurčíček, Blaise Thomson, Kai Yu, and Steve Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 312–317. IEEE.

Milica Gašić and Steve Young. 2014. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40.

James Henderson, Oliver Lemon, and Kallirroi Georgila. 2008. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics*, 34(4):487–511.

Matthew Henderson and Blaise Thomson. 2014. The second dialog state tracking challenge. In *SIGDIAL*, volume 263, Stroudsburg, PA, USA. Association for Computational Linguistics.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Isabel S. Jesus and J. A. Tenreiro MacHado. 2008. Fractional control of heat diffusion systems. *Nonlinear Dynamics*, 54(3):263–282.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134.

Long-Ji Lin. 1993. *Reinforcement learning for robots using neural networks*. Ph.D. thesis, Fujitsu Laboratories Ltd.

Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng. 2016. Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *arXiv preprint arXiv:1608.05081*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, pages 4026–4034.

Olivier Pietquin, Matthieu Geist, and Senthilkumar Chandramohan. 2011. Sample Efficient On-line Learning of Optimal Dialogue Policies with Kalman Temporal Differences. In *IJCAI*, pages 1878–1883.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007a. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *NAACL*, pages 149–152, Morristown, NJ, USA. Association for Computational Linguistics.

Jost Schatzmann, Blaise Thomson, and Steve Young. 2007b. Error simulation for training statistical dialogue systems. In *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, pages 526–531. IEEE.

Stanley M Shinners. 1998. *Modern control system theory and design*. John Wiley & Sons.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young. 2017. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689*.

Jason D Williams. 2008. The best of both worlds: unifying conventional dialog systems and pomdps. In *INTERSPEECH*, pages 1173–1176.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422.

Jason D Williams and Geoffrey Zweig. 2016. End-to-end LSTM-based dialog control optimized with supervised and reinforcement learning. *CoRR*.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech and Language*, 24(2):150–174.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 1–10.

Towards Debate Automation: a Recurrent Model for Predicting Debate Winners

Peter Potash and **Anna Rumshisky**
Department of Computer Science
University of Massachusetts Lowell
{ppotash,arum}@cs.uml.edu

Abstract

In this paper we introduce a practical first step towards the creation of an automated debate agent: a state-of-the-art recurrent predictive model for predicting debate winners. By having an accurate predictive model, we are able to objectively rate the quality of a statement made at a specific turn in a debate. The model is based on a recurrent neural network architecture with attention, which allows the model to effectively account for the entire debate when making its prediction. Our model achieves state-of-the-art accuracy on a dataset of debate transcripts annotated with audience favorability of the debate teams. Finally, we discuss how future work can leverage our proposed model for the creation of an automated debate agent. We accomplish this by determining the model input that will maximize audience favorability toward a given side of a debate at an arbitrary turn.

1 Introduction

Conversational agents are a well-researched area of natural language generation (Pilato et al., 2007; Bigham et al., 2008; Augello et al., 2008; Agostaro et al., 2005; Bessho et al., 2012). Elsewhere in the field of natural language generation, there is work that seeks to generate persuasive text (Carenini and Moore, 2006; Reiter et al., 2003; Rosenfeld and Kraus, 2016), which is a logical first step towards creating an automated debate agent. One major deficiency of existing work in this area is its assessment of how convincing (or compelling) a piece of text is; the approaches use theory-driven models of persuasion, rather than being empirically motivated. Furthermore, none of these works provide a model that can optimize persuasiveness at an arbitrary point in a conversation.

One of the main reasons for a lack of empirically-driven persuasive generation systems is the absence of labeled data. In order to alleviate this problem (though not directly for the sake of producing an automated debate agent), Zhang et al. (2016) have introduced a dataset of debate transcripts from the "Intelligence Squared" (IQ2)[1] debates. In these debates, two teams are present, arguing either for or against a given topic. For each debate, an audience poll is conducted both prior to and after the debate. Whichever team has the largest gain in audience support between the pre/post debate polls is the winner. This is a natural way to account for the fact that some sides of a debate may be harder to argue than others, and that audience members may be initially biased given a debate topic.

Because of the sequential nature of debating, a Recurrent Neural Network (RNN) is an attractive choice for modeling the problem. Rather than just using the final hidded state for prediction, which likely has lost information from early in the debate, we propose to use an attention mechanism (Bahdanau et al., 2014) that creates a weighted sum over all hidden states, and is subsequently used for the final prediction. We motivate the use of an RNN, as opposed to a temporally flat classifier, for several reasons. First, using an RNN allows us to naturally incorporate predicting audience favorability at each turn while explicitly modeling the turn sequence. Logistic regression, on the other hand, would not allow us to model the sequence explicitly. Secondly, our model allows us to take raw features as input, without having to compute summary statistics necessary for the fea-

[1] http://www.intelligencesquaredus.org/

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2465–2475
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

tures used in the model of Zhang et al. (2016). Finally, since our end goal is debate automation, an RNN is a natural choice for debate turn generation.

There are two major difficulties dealing with the IQ2 dataset: first, since the construction of the dataset is non-trivial, there are only 108 data points, resulting in Zhang et al.'s proposal for leave-one-out (LOO) evaluation. Second, considering the use of an RNN, the sequences are long, with an average length of 246 (and a standard deviation of 67). In order to overcome this, we incorporate signals based on implicit audience feedback during the debate into the model's loss function. Instead of just training the model based on error from the audience's final verdict, propogated through a substantial amount of timesteps, there are intermittent errors propogated backward through the network based on audience reactions, such as applauding or laughing. These internal signals also help regularize the network. In a way, they help generalize the hidden representation of the RNN, allowing it to better contain a distributed representation of the audience's favorability towards a given team.

In our proposed model, the audience's opinion is directly a function of the weighted hidden representations. Since the previous hidden representations are all fixed at a given timestep, and the current hidden representation is directly a function of these previous hidden representations as well as the current input, the audience's current poll depends directly on the timestep's input. Therefore, at a given timestep, our framework allows us to determine the input that would maximize the audience's favorability toward the orating team. This is due to the fact that the inputs are themselves representations of a given team's statement at a particular turn in the debate.

We evaluate our model on the dataset from Zhang et al., posting state-of-the-art accuracy. Our results show that our proposed regularization technique is imperative for the RNN-based model to perform competitively with the models previously proposed by Zhang et al.. The attention mechanism also contributes to the best performing system. Afterward, we show how our model can be used to track audience favorability throughout the debate, as well as the aforementioned input optimization, using it in a case study to instruct a debate team about optimal debate strategy at a given turn.

2 Related Work

Previous work that focuses on conversational language seeks to predict such qualities as disagreements (Allen et al., 2014; Wang and Cardie, 2016), divergence (Niculae and Danescu-Niculescu-Mizil, 2016), and participant stance (Sridhar et al., 2015; Somasundaran and Wiebe, 2010; Thomas et al., 2006; Rosenthal and McKeown, 2015). What is most relevant for our purposes are the methods these models use for dealing with conversational data. Allen et al. (2014) apply discourse parsing (Joty et al., 2013) and fragment quotation graph (Carenini et al., 2007) tools to detect disagreement in online discussion threads. Wang and Cardie (2016) believe that disagreement can be predicted by the presence of substantially long sequences of negative sentiment, motivating them to build a sequential sentiment prediction model using a particular kind of Conditional Random Field (Mao and Lebanon, 2007). Niculae and Danescu-Niculescu-Mizil (2016) use several novel features that capture the flow of ideas in the data, as well as team dynamics. Ultimately, however, all these models apply manually derived, preprocessed features and use a basic classifier, like Random Forest or Logistic Regression. In contrast, an RNN model is able to learn which interactions and overall sequences of rhetoric are important for predictive power.

There is much less work that approaches the problem of predicting persuasiveness of text. This is due primarily to the lack applicable datasets. However, Habernal and Gurevych (2016b) have recently presented a dataset where argument pairs are annotated for argument convincingness, as well as finer-grained annotations related to the effectiveness of arguments (Habernal and Gurevych, 2016a). The authors experimented with feature-based classifiers, as well as various RNN architectures to construct predictive models for the dataset.

The most relevant work for this paper is of course Zhang et al. (2016). The authors use a set of features derived from the notion of idea flow in the debate. More specifically, they follow the method of Monroe et al. (2008) to identify talking points used by the sides present in a debate. The authors then create features based on the coverage of talking points during the debate. Finally, a Logistic Regression model uses these features to predict which team wins the debate. We also note the work of Santos et al. (2016), which also makes predictions on a dataset derived from the IQ2 debates. In contrast, their work analyses speech signals, as opposed to textual data.

3 Predictive Model

In this section we explain how we apply an RNN to the task of predicting debate winners. We start by addressing the fact that for IQ2 dataset, each timestep involves a text span, as opposed to single tokens, and explaining how we convert this text span into a vector representation for RNN input. Secondly, we explain our RNN model architecture, including our use of an attention mechanism to create a weighted sum over all hidden states, as well as a regularization technique based on implicit audience reaction.

3.1 Representing Debate Turns

Our work follows that of Zhang et al. (2016), and uses talking point-based features, specifically a 'bag of talking points'. Talking points for each debate are identified using a term frequency inverse document frequency (tfidf) metric applied to text tokens. Token counts, whether at a document or corpus level, occur only for the introduction text, as done by Zhang et al. This is based on the belief that the introductory arguments best showcase potential talking points. We take the 10 tokens with highest tfidf scores for each debate, and, across all debates, each token ranking maps to a fixed index in the turn representation. This representation is binary.

Zhang et al.'s results suggest that the interaction of talking points between debate teams can possess strong predictive power. Therefore, we also calculate talking points at a team level within debates. We accomplish this by simply taking term frequency counts for tokens spoken by a given team. Like with the overall debate talking points, we chose the 10 highest ranked talking points from each side and include them in the input representation. Moreover, we believe we can use a simpler talking point metric than that proposed by Monroe et al. (2008) (and used by Zhang et al.) because the recurrent nature of the model will naturally capture the interaction, coverage, and ignorance of the two team's (and overall) talking points.

Aside from talking point-based features, we include the following linguistic features: 1) bag-of-words for tokens that have been used in at least 50 debates; 2) GloVe embeddings of tokens (Pennington et al., 2014). We use max pooling over all the tokens' embeddings to create the embedding features. We also use the following non-linguistic features: 1) whether the turn occurs during the opening, discussion, or conclusion phase of the debate; 2) whether the turn is from the 'for' or 'against' team, as well as moderator or other speakers, such as show host etc; 3) the initial audience poll is provided at each timestep. This is similar in spirit to Cho et al. (2014)'s decoder model that accesses the final encoder hidden state at each timestep.

We acknowledge that it would be possible to model individual turns (sequences of tokens) with a separate RNN. We choose to use hand-engineered features for two reasons: First, the current representation, mainly the talking points and BOW features, are easily interpretable given the goal of providing rhetorical strategy for debaters. Using an RNN for this purpose would require training a decoder in order to interpret the optimal rhetoric at a given turn (see Section 7). Secondly, it follows that having a trainable representation would introduce additional parameters into the model, which is a concern, given the limited amount of data.

3.2 Recurrent Architecture

Our RNN model uses a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) component. At each timestep, the model receives as input a turn representation defined in Section 3.1. After consuming all turn representations, a simple model without attention woud pass the final hidden state, h_f, through two fully-connected layers (with an intermediate representation h_a to which we apply sigmoid activation), whose weights have subscripts $post$ to identify that this transformation happens after the debate:

$$h_a = \sigma(W^1_{post} h_f + b^1_{post}) \tag{1}$$

$$a = W^2_{post} h_a + b^2_{post} \tag{2}$$

where σ is the sigmoid function. This transformation outputs a vector with three dimensions, which corresponds to the fact that the audience poll has three possibilities: for, against, and undecided.

Since the polling is given as a percentage breakdown, we apply $softmax$ to create a valid probability probability distribution for the audience,

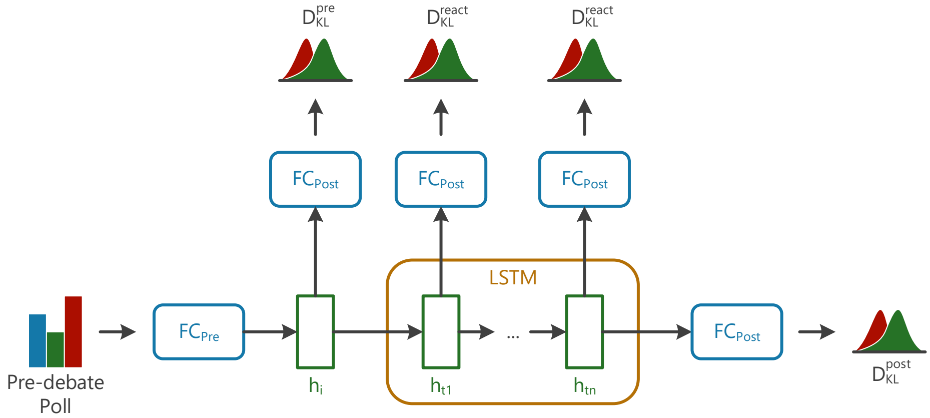


Figure 1: An illustration of our training objective from Equation 9 unrolled over time. FC_{pre} and FC_{post} refer to Equations 1/2 and 8, respectively.

$p(A)$:

$$p(A|\Theta) = \text{softmax}(a) \tag{3}$$

which is for a given set of model paramters, Θ.

We train the model to minimize the Kullback-Leibler (KL) divergence between the target and predicted audience poll percentages. Given a training corpus of debates D with target post-debate audience polls A_i^{target}, the optimization objective is:

$$\arg\min_{\Theta} \sum_{d \in D} D_{KL}(p(A_d^{target})||p(A_d|\Theta)) \tag{4}$$

which simply sums the KL-Divergence of the target and predicted audience poll percentages (probabilities) across all training examples. At test time, the model uses the percentages from $p(A|\Theta)$ to calculate which team increased their support from the audience the most, using the pre-debate audience poll, which is given. For notation purposes, we refer to this KL-divergence for post debate audience polling D_{KL}^{post}. The optimization objective from Equation 4 describes our base model. Shortly, we will describe how we regularize this base model using implicit audience feedback.

3.2.1 Attention Mechanism

The model we have described to this point uses the final hidden state to predict the final audience poll. A concern with this approach is that the final hidden state has a difficult time encoding the activity from the earlier parts of the debate. We propose to rectify this issue by creating a weighted sum over all hidden states, following the the attention mechanism from Bahdanau et al. (2014). Given hidden states from all RNN timesteps, $(h_1, ..., h_f)$, we determine the weight for h_i as follows. First, we compute a raw attention score:

$$r_i = v^T \tanh(W_a h_i + b_a) \tag{5}$$

where v, W_a, b_a are model parameters. h_i's weight is computed from applying softmax to r:

$$\alpha_i = \text{softmax}(r)_i \tag{6}$$

which we use to compute the weighted sum across all hidden states:

$$h_s = \sum_{i=1}^{f} \alpha_i h_i \tag{7}$$

Therefore, the attention version of our model uses h_s in Equation 1 to predict the final audience poll.

3.2.2 Initializing RNN Hidden State

As we have mentioned, audience polls occur both before and after the debate. Thus, we continue the theme of using the RNN hidden state to express audience polling by exploiting the initial audience poll to initialize the RNN hidden state, h_0. The model uses the initial audience poll, a_{pre}, and applies a fully-connected layer with parameters W_{pre} and b_{pre}:

$$h_0 = \tanh(W_{pre} a_{pre} + b_{pre}) \tag{8}$$

We choose $\tanh$ for the activation function because it is the same activation function used by the LSTM cell. The RNN now is initialized with a hidden state that reflects the audience's initial attitude towards a given debate topic.

3.2.3 Regularization via Implicit Audience Feedback

The IQ2 dataset offers two challenges for implementing an RNN-based approach. First, which is a difficulty for any type of supervised model, is the small dataset size. There are a total of 108 data points, which, even with LOO evaluation, leaves only 107 examples for training a model. For neural networks in particular, there is worry that overfitting easily occurs when the amount of model parameters is much greater than the dataset size (Lawrence et al., 1998; Ingrassia and Morlini, 2005). Aside from the dataset size, the sequences of debate turns are long, averaging 246. This means that, on average, our model will run for 246 timesteps, making it difficult to train the network (Bengio et al., 1994) (the structure of the LSTM memory cell was designed to solve this issue, which motivates our use of it in our model).

In order to overcome these difficulties, we propose to regularize our network based on implicit audience feedback that occurs during the debate, and is provided as metadata with the debate transcript. Specifically, provided along side each debate turn, there is a 'non-text' field that indicates if any sounds occurred during the turn such as applause or laughter from the audience. We view the presence of applause or laughter from the audience as a sign of endorsement during that particular turn. Therefore, at that particular timestep, the hidden state should be able to directly predict this occurrence. Considering applause as a sign of endorsement is not controversial, but laughter could be viewed as more ambiguous. However, consider the audience of the debates: the debates air on the Bloomberg network and National Public Radio, suggesting a higher level of maturity of the audience, which is less likely to laugh *at* the participants, rather than at their jokes. For example, here is a turn in the debate 'Men are Finished' wherein laughter occurs: "Wait. What was that phrase you used, surviving off the fumes of sexism? I think we are our finest example there." This is an intentional joke by the speaker, who is part of of the winning team in the debate.

This signal can be integrated in a supervised manner into the loss function by converting the audience reaction at a given timestep into a three-dimensional vector, representing the current, implied audience favorability. We create such a vector at a debate turn if either applause or laughter occurs at that timestep, and the speaker is one of the debate teams. On possibility is to create have a one-hot vector implying the audience favorability at the turn, with the mapping of side to index dictated by the target vector, A_i^{target}, and is set for the corpus. There is a major problem with using a one-hot vector: the probability distributions learned by the model will become too skewed, since the ultimate goal is to better generalize the prediction of debate polls, rarely are the polls so unbalanced toward one side. Moreover, the one-hot vector will only ever have mass in the indices for the 'for' and 'against' teams, and neglecting the 'undecided' index, which is an important sector in the polling. Therefore, we create a soft vector as follows: a random number, n, is chosen in the interval $(\frac{1}{3}, 1)$. The index corresponding to the speaking team at timestep i has value n. The remaining two indices have value $\frac{1-n}{2}$. This vector is notated A_{it}^{target}, specifying that the reaction occurred at timestep t for debate i. On average, such reactions occur 21 times during a debate, with a standard deviation of 10. Consequently, this approach adds 2,268 more supervised signals to the dataset.

As we did with the post-debate poll, we can compute a lost based on the kl-divergence between A_{it}^{target} and the prediction probability at timestep t, which is a function of h_t using the same transformations described in Equations 1, 2, and 3, but replacing h_f with h_t. The attention model can been used as well. In this case, we compute h_{s_t} by slicing r (from equation 5) to only include indices up to t. We denote the KL-divergence between target and prediction distributions across all timesteps of a training example is D_{KL}^{react}, since these signals are based on audience reaction.

The same strategy can be applied to h_i using the pre-debate polls. Although this signal does not propagate through the RNN, it can still train the weights of the fully-connected layers used in our model. We refer to this KL-divergence as D_{KL}^{pre}, since it uses the pre-debate poll. Bringing together these separate error signals, we arrive at the training objective of our full model:

$$\arg\min_{\Theta} \sum_{d \in D} D_{KL}^{pre} + D_{KL}^{react} + D_{KL}^{post} \qquad (9)$$

where Θ is the model parameters used to produce the prediction probabilities. Figure 1 provides an illustration of our training objective, unrolled over time.

With this new optimization objective, each example now trains our model based on (on average) 23 supervised signals. As a result, each training example allows the model to become more generalizable, particularly because the hidden states are now better-tuned to encode audience favorability. This methodology allows the model to better leverage the small dataset size. Moreover, the intermittent error signals from audience reaction, D_{KL}^{react}, combined with the pre-debate error signal, D_{KL}^{pre}, help assuage the difficulties of training our model based on a final error signal propagated for many timesteps. We would like to reiterate that this regularization technique is only used to *train* the model, and not used for prediction, and therefore will not be an issue when making predictions for new debates, nor will it create an unrealistic circumstance for using the model for creating a debate agent.

4 Experimental Design

Our experiments are conducted on the IQ2 dataset (Zhang et al., 2016). We use LOO evaluation, resulting in a training set of 107 examples. The evaluation metric is simply prediction accuracy for debate winners. The winning team is based on audience polling. Polls are conducted before and after the debate, and audience members can vote as being either for or against a given debate topic, as well as being undecided. The team that has the highest increase in audience support from the pre to post debate poll is the winning team. The model trains for 100 epochs. Once training is complete, we test on the held-out data point. As Zhang et al. note, there are three debates in the dataset that have a tie between the debate teams. Following their procedure, we do not test on these data points. However, we still include these examples in the training sets, because our training objective is to predict polls, not debate winners. The final test accuracy is averaged across the remaining 105 LOO runs. Furthermore, we note that the dataset is effectively balanced, as there are 53 and 52 examples with the two possible labels.

We implement all our models in TensorFlow (Abadi et al., 2016). We use the LSTM cell equipped with peephole connections (Gers et al., 2002). This architecture allows the gates to see the current cell state, along with the hidden state. We believe that because of the long sequences present in the dataset, it is important to have all the gates take into account the cell state when producing a hidden layer. This adds a stronger notion of memory to the model. While we expect the hidden state to represent audience favorability, we believe the cell state can capture the further latent notion of debate strategy, observable through the interaction of talking points between the debate teams. The models have cell and hidden size of 128, and the intermediate layer from Equation 2 has size 16. Lastly, we use a batch size of 8.

Model	Accuracy
LR BOW	0.50
LR React	0.60
LR Flow	0.63
LR Flow*	0.65
LSTM	0.55
LSTM + Att	0.57
LSTM + Reg	0.64
LSTM + Att, Reg	**0.71**
LSTM + Att, Drpt	0.60

Table 1: The results of LOO evaluation on the IQ2 dataset. See the beginning of Section 5 for an explanation of the models.

5 Results

The results of our experiment are presented in Table 1. Att means the model has the attention mechanism from Section 3.2.1; Reg means the model uses the optimization objective from Equation 9 (all other models use the optimization objective from Equation 4); Drpt means the model uses dropout (a popular regularization technique for neural networks (Srivastava et al., 2014)) of 0.5. We compare our results against the best models from Zhang et al. (2016). Each model uses a Logistic Regression (LR) classifier, and distinguishes itself by the features it uses. The main features developed by the authors relate to the interaction (flow) of talking points between the debate teams. There are two types of models that use the flow features: LR Flow and LR Flow*. Whereas the former uses all developed flow features, the latter uses feature selection to keep the most powerful flow features. LR React uses features based on audience reaction metadata, and LR BOW uses bag-of-words features.

The results show that the LSTM attention model regularized by audience reaction achieves the highest accuracy. Furthermore, the results highlight the importance of this regularization technique, since the simple LSTM model records the second lowest accuracy of any of the models presented. This leads us to believe that the regular LSTM model falls victim to the lack of training data, preventing the larger amount of model parameters (compared to a logistic regression model) from generalizing. The results also show that the attention model has higher performance than the regular LSTM model, and the difference in performance is heightened when the regularization technique is applied. We believe this is because the attention mechanism adds additional parameters to the model, so it seems reasonable that adding additional training signals helps the model to generalize better. Lastly, our proposed regularization technique is far superior for generalization than the popular dropout method. We believe the strong performance of the proposed regularization technique is because it causes the LSTM's hidden states to better generalize the notion of encoding audience favorability. Furthermore, our model's goal is to predict *distributions*, as opposed to labels. Whereas dropout can be effective at aiding in collapsing representations of the same class into neighboring points of a latent space, our model needs to be able to predict polls that it may have not encountered in training. Our regularization technique aids in this as well by providing more training data, more polls.

6 Tracking Audience Favorability

One of the advantages of mapping a recurrent model's hidden states to audience favorability is that we can produce a favorability poll at any turn (timestep) during the debate. In contrast, a temporally flat model, such as the logistic regression models from Zhang et al., produce a prediction of audience favorability based on features extracted from the entire debate. Using our mapping of hidden states to audience favorability, we can determine, at each turn, the current audience favorability, and track it throughout the entire debate. Figure 2 shows this applied to the "men are finished" debate, wherein the lines on the graph, cut vertically, represent predicted audience polls at a given debate turn. This debate saw the greatest increase in audience support from the pre to post debate poll: the 'for' increased their favorability by 46% (46 points). The three lines correspond to the three possible positions an audience member can take regarding the debate topic. This visualization can be particularly useful for rhetorical analysis of debate performance, because the resulting graph allows us to see inflection points in audience favorability. These inflection points suggest that a debate team used very effective (or ineffective) rhetoric at that particular turn.

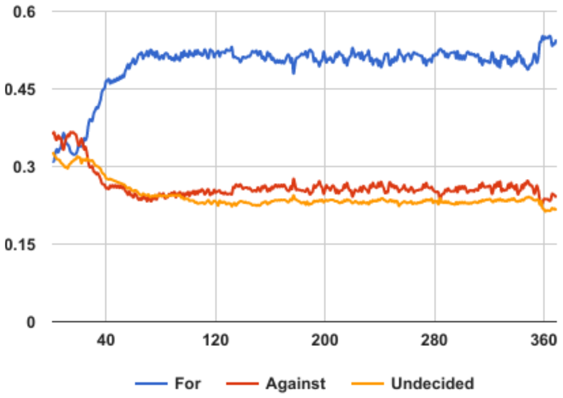


Figure 2: A visualization of audience favorability for the debate "men are finished". At each turn in the debate, our model predicts the audience favorability. The y-axis shows the percentage of the audience that supports a given side, and the x-axis show the turn number for a given poll. Even though these are purely predictions from the model, it is able to show the rise in audience favorability for the 'for' team, as well as the decline in favorability for the 'against' team. From the graph, we can see that the 'for' team had a large spike in audience support roughly between turns 20 and 40, which corresponds to the beginning of the debate's discussion section.

7 Optimizing Input for Audience Favorability

Aside from achieving a new state-of-the-art result on the IQ2 debate corpus, the main appeal of the model we have introduced is that it creates a mapping between the hidden states and audience favorability of the debate teams. This mapping is given in Equations 1 and 2, where a weighted sum over all over all hidden states (the actual notation in these equation apply the fully-connected transformation to a final hidden state, h_f, unlike the attention model which uses h_s from Equation 7) is transformed into a real-valued 3-dimensional vector a. The values of the vector indicate 'raw' favorability, which is realized as a probability distribution (or alternatively, a poll of the audience) after applying the softmax activation. Furthermore, given fixed model parameters Θ, the current hidden state is a function of the previous hidden states, previous cell state (if, like our model, an LSTM cell is used), as well as the current input. At a given timestep, the previous hidden and cell states are known. Therefore, a is directly a function of the current input x at a given timestep. This notion of optimizing input for a target 'class' is akin to the work of Simonyan et al. (2013), who use a trained convolutional neural network to find the optimal input image for a desired object class.

7.1 Input Optimization Objective

Similar to our approach in Section 3.2.3 to encode implicit audience feedback, we can construct a three-dimensional one-hot vector with the index switched on that corresponds to the debate team whose favorability we seek to optimize. We will call this vector A^{fav}. Given input x_i at timestep i, we seek to to optimize the probability of A^{fav} given x_i:

$$\arg\max_{x_i} p(A^{fav}|x_i, h_1, ..., h_{i-1}, c_{i-1}; \Theta) \quad (10)$$

Where $i \in (1, ..., T)$ and T is the maximum number of timesteps (turns) for a debate. In practice, we achieve this optimization by minimizing the cross-entropy between the the target one-hot vector and the output of applying the softmax function to a, as in Equation 3.

7.2 Applying Optimized Input for Persuasive Strategy

In the debate 'men are finished', the 'for' team won the debate, increasing their favorability by an astonishing 46% (conversely, the 'against' team saw a 25% *decrease* in favorability). According to our model's sequential predictions (and visible in Figure 2), a major turning point occurred at turn 27. Quantitatively, we can examine the turn-by-turn change in audience favorability: from this we see that one of the largest increases in audience favorability occurred at turn 27. It is not a surprise to find out that the team that spoke during turn 27 was the 'for' team. When asked by the moderator if there can be equality between the sexes without deeming men as being finished, the 'for' team said the following (the text is annotated for the presence of talking points, marked by a subscript that specifies whose talking point it is: A (against), F (for), or G, a general talking point based on overall token frequency (see Section 3.1)):

> It is possible, but it just doesn't work that way. I mean, if we can all agree that there was male dominance for a long time and that male dominance is over, then I think we agree that $\underline{\text{men}}_{G,A}$ are finished. So the resolution is about male dominance which we've taken for granted for so many tens of thousands of years. And so, even if you have parity, you have the end of male dominance. I mean, if you have $\underline{\text{women}}_{F}$ rising and catching up to $\underline{\text{men}}_{G,A}$, then you no longer have male dominance. And so that's what I meant when I, early on, tried to define the resolution as $\underline{\text{men}}_{G,A}$ are finished, the era of male dominance, it's finished, which we've taken for granted for all this time.

Note that the term 'women' is only a talking point for the 'for' team. In their response the 'against' team says:

> They are not finished. That's absurd. You agreed to it in your opening that you didn't want to say $\underline{\text{men}}_{G,A}$ are finished. You thought there might be inklings of a suggestion that it may be happening. But what you're defending now is that $\underline{\text{men}}_{G,A}$ are finished. I'm saying it's absurd. I'm saying that some $\underline{\text{men}}_{G,A}$ are in trouble. But rather than declare their extinction, we should be doing what we can to help them.

To determine our model's strategy immediately after the 27th turn, we apply the previous hidden and cell states to the optimization objective in Equation 10, taking the place of $h_1, ..., hi - 1$ and c_{i-1}, respectively. We fit the training objective to the current states, as well as the weights of the previously trained predictive model, and examine the resulting optimized input vector. We train the optimized input model for 15,000 epochs, which goes very fast because there is a 'single' training data point, and the model is not recurrent. As we can see in the actual 'against' team's response, the only talking point brought up is 'men', which can hardly be viewed as an enlightening notion in the context of the debate. Alternatively, the highest rated talking point from the optimized input is in fact the exact talking point brought up by the 'for' team: 'women'. This suggestion by our model is in line with the hypothesis of Zhang et al. (2016), that winning teams are effective in adopting their opponent's talking points. In terms of bag-of-word features: the optimized input ranks the following tokens as the ten highest (in descending order of score, and note the tokens have been stemmed): 'sound', 'present', 'recent', 'line', 'decid', 'veri', 'spent', 'save', 'moder', and 'found'. Most of these tokens remain somewhat vague with respect to their relevance to the debate. The token 'recent' seems relevant, given that the debate topic has an inherent temporal nature. 'Save' is relevant in that some of the debate discussion approaches the question of whether men need saving. In the top 20 tokens we also find 'done', 'compar', 'grow', and 'without', which are all relevant: 'done' is synonymous with 'finished', 'compar' given that the debate is often comparing men to women, 'grow' could refer to the growth of women in society, and 'without' is a token specifically in the question the moderator asked prior to turn 27 (equality between the sexes without deeming men as being finished).

8 Conclusion

We have presented an RNN model for predicting debate winners, with the specific goal of predicting the final (or intermediate) audience poll. The model takes at each timestep a representation of a given debate turn. The model uses an attention mechanism that creates a weighted sum over all hidden states. In order to achieve state-of-the-art results on a corpus of debate transcripts (Zhang et al., 2016), we regularize the RNN model by propagating errors based on implicit audience reaction. Our results show that this regularization technique is critical for obtaining a state-of-the-art result. We have also shown the practical application of our model in two scenarios. First, the model can be used to make a prediction of audience polling at every debate turn. This allows for an analysis of the key turning points during the debate, based on inflections in audience favorability. Second, the model can be used to determine the optimal input at a given debate turn. Knowledge of this input can inform debaters as to the best current persuasive strategy. Future work can leverage optimal inputs to create a language model that can become an automated debate agent. However, since the input is partially based on the knowledge of talking points, there is a potential for an information retrieval-based task to provide the talking points for the debate agent (if one desires a fully-automated system than can work without the presence of introductory remarks, from which talking points are currently extracted). Finally, future work can also examine the trained model itself in further detail, seeking to understand the debate strategy.

Acknowledgments

This work was supported in part by the U.S. Army Research Office under Grant No. W911NF-16-1-0174. We would like to thank Alexey Romanov for his help with the figures.

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Francesco Agostaro, Agnese Augello, Giovanni Pilato, Giorgio Vassallo, and Salvatore Gaglio. 2005. A conversational agent based on a conceptual interpretation of a data driven semantic space. In *Congress of the Italian Association for Artificial Intelligence*, pages 381–392. Springer.

Kelsey Allen, Giuseppe Carenini, and Raymond T Ng. 2014. Detecting disagreement in conversations using pseudo-monologic rhetorical structure. In *EMNLP*, pages 1169–1180.

Agnese Augello, Gaetano Saccone, Salvatore Gaglio, and Giovanni Pilato. 2008. Humorist bot: Bringing computational humour in a chat-bot system. In *Complex, Intelligent and Software Intensive Systems, 2008. CISIS 2008. International Conference on*, pages 703–708. IEEE.

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

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Fumihiro Bessho, Tatsuya Harada, and Yasuo Kuniyoshi. 2012. Dialog system using real-time crowdsourcing and twitter large-scale corpus. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 227–231. Association for Computational Linguistics.

Jeffrey P Bigham, Maxwell B Aller, Jeremy T Brudvik, Jessica O Leung, Lindsay A Yazzolino, and Richard E Ladner. 2008. Inspiring blind high school students to pursue computer science with instant messaging chatbots. In *ACM SIGCSE Bulletin*, volume 40, pages 449–453. ACM.

Giuseppe Carenini and Johanna D Moore. 2006. Generating and evaluating evaluative arguments. *Artificial Intelligence*, 170(11):925–952.

Giuseppe Carenini, Raymond T Ng, and Xiaodong Zhou. 2007. Summarizing email conversations with clue words. In *Proceedings of the 16th international conference on World Wide Web*, pages 91–100. ACM.

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

Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. 2002. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143.

Ivan Habernal and Iryna Gurevych. 2016a. What makes a convincing argument? empirical analysis and detecting attributes of convincingness in web argumentation. In *EMNLP*, pages 1214–1223.

Ivan Habernal and Iryna Gurevych. 2016b. Which argument is more convincing? analyzing and predicting convincingness of web arguments using bidirectional lstm. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

Salvatore Ingrassia and Isabella Morlini. 2005. Neural network modeling for small datasets. *Technometrics*, 47(3):297–311.

Shafiq R Joty, Giuseppe Carenini, Raymond T Ng, and Yashar Mehdad. 2013. Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. In *ACL (1)*, pages 486–496.

Steve Lawrence, C Lee Giles, and Ah Chung Tsoi. 1998. What size neural network gives optimal generalization? convergence properties of backpropagation. Technical report.

Yi Mao and Guy Lebanon. 2007. Isotonic conditional random fields and local sentiment flow. *Advances in neural information processing systems*, 19:961.

Burt L Monroe, Michael P Colaresi, and Kevin M Quinn. 2008. Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372–403.

Vlad Niculae and Cristian Danescu-Niculescu-Mizil. 2016. Conversational markers of constructive discussions. *arXiv preprint arXiv:1604.07407*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Giovanni Pilato, Agnese Augello, Giorgio Vassallo, and Salvatore Gaglio. 2007. Sub-symbolic semantic layer in cyc for intuitive chat-bots. In *Semantic Computing, 2007. ICSC 2007. International Conference on*, pages 121–128. IEEE.

Ehud Reiter, Roma Robertson, and Liesl M Osman. 2003. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*, 144(1-2):41–58.

Ariel Rosenfeld and Sarit Kraus. 2016. Strategical argumentative agent for human persuasion. In *ECAI 2016: 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands-Including Prestigious Applications of Artificial Intelligence (PAIS 2016)*, volume 285, page 320. IOS Press.

Sara Rosenthal and Kathleen McKeown. 2015. I couldnt agree more: The role of conversational structure in agreement and disagreement detection in online discussions. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 168.

Pedro Bispo Santos, Lisa Beinborn, and Iryna Gurevych. 2016. A domain-agnostic approach for opinion prediction on speech. *PEOPLES 2016*, page 163.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124. Association for Computational Linguistics.

Dhanya Sridhar, James R Foulds, Bert Huang, Lise Getoor, and Marilyn A Walker. 2015. Joint models of disagreement and stance in online debate. In *ACL (1)*, pages 116–125.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 327–335. Association for Computational Linguistics.

Lu Wang and Claire Cardie. 2016. A piece of my mind: A sentiment analysis approach for online dispute detection. *arXiv preprint arXiv:1606.05704*.

Justine Zhang, Ravi Kumar, Sujith Ravi, and Cristian Danescu-Niculescu-Mizil. 2016. Conversational flow in oxford-style debates. *arXiv preprint arXiv:1604.03114*.

Further Investigation into Reference Bias in Monolingual Evaluation of Machine Translation

Qingsong Ma[*†] **Yvette Graham**[†] **Timothy Baldwin**[‡] **Qun Liu**[†]

[*]Institute of Computing Technology
Chinese Academy of Sciences
maqingsong@ict.ac.cn

[†]ADAPT Centre
Dublin City University
firstname.surname@dcu.ie

[‡]Computing and Info Systems
University of Melbourne
tb@ldwin.net

Abstract

Monolingual evaluation of Machine Translation (MT) aims to simplify human assessment by requiring assessors to compare the meaning of the MT output with a reference translation, opening up the task to a much larger pool of genuinely qualified evaluators. Monolingual evaluation runs the risk, however, of bias in favour of MT systems that happen to produce translations superficially similar to the reference and, consistent with this intuition, previous investigations have concluded monolingual assessment to be strongly biased in this respect. On re-examination of past analyses, we identify a series of potential analytical errors that force some important questions to be raised about the reliability of past conclusions, however. We subsequently carry out further investigation into reference bias via direct human assessment of MT adequacy via quality controlled crowd-sourcing. Contrary to both intuition and past conclusions, results show no significant evidence of reference bias in monolingual evaluation of MT.

1 Introduction

Despite it being known for some time now that automatic metrics, such as BLEU (Papineni et al., 2002), provide a less than perfect substitute for human assessment (Callison-Burch et al., 2006), evaluation in MT more often than not still comprises BLEU scores. Besides increased time and resources required by the alternative, human evaluation of systems, human assessment of MT faces additional challenges, in particular the fact that human assessors of translation quality tend to be highly inconsistent. In recent Conference on Machine Translation (WMT) shared tasks, for example, manual evaluators complete a relative ranking (RR) of the output of five alternate MT systems, where they must rank the quality of competing translations from best to worst. Within this set-up, when presented with the same pair of MT output translations, human assessors often disagree with one another's preference, and even their own previous judgment about which translation is better (Callison-Burch et al., 2007; Bojar et al., 2016). Low levels of inter-annotator agreement in human evaluation of MT not only cause problems with respect to the reliability of MT system evaluations, but unfortunately have an additional knock-on effect with respect to the meta-evaluation of metrics, in providing an unstable gold standard. As such, provision of a fair and reliable human evaluation of MT remains a high priority for empirical evaluation.

Direct assessment (DA) (Graham et al., 2013, 2014, 2016) is a relatively new human evaluation approach that overcomes previous challenges with respect to lack of reliability of human judges. DA collects assessments of translations separately in the form of both fluency and adequacy on a 0–100 rating scale, and, by combination of repeat judgments for translations, produces scores that have been shown to be highly reliable in self-replication experiments (Graham et al., 2015). The main component of DA used to provide a primary ranking of systems is *adequacy*, where the MT output is assessed via a monolingual similarity of meaning assessment. A reference translation is displayed to the human assessor (rendered in gray) and below it the MT output (in black), with the human judge asked to state the degree to which they agree that *The black text adequately expresses the meaning of the gray text in English.*[1] The motivation behind

[1] Instructions are translated into a given target language.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2476–2485
Copenhagen, Denmark, September 7–11, 2017.

constructing DA as a monolingual MT evaluation are as follows:

- Monolingual assessment of MT opens up the annotation task to a larger pool of genuinely qualified human assessors;
- Crowd-sourced workers are unlikely to make use of information that is not entirely necessary for completing a given task; and are therefore unlikely to use the source language input if the reference is also displayed or to make use of the source input inconsistently;
- Displaying only the source without a reference greatly increases both the difficulty of the task and the time required to complete each annotation, which is too serious a trade-off when we wish to carry out human assessment on a very large scale;
- Varying levels of proficiency in the source language across different human assessors could contribute to inconsistency in bilingual MT evaluations.

Although DA has been shown to overcome the long-standing challenge of lack of reliability in human evaluation of MT, the possibility still exists that, although scores collected with DA have been shown to be almost perfectly reliable in self-replication experiments, both sets of scores, although consistent with each other, could in fact both be biased in the same way. Graham et al. (2013) include in the design of DA a number of criteria aimed at minimizing such bias: (i) assessment of individual translations in isolation from others to avoid a given system being scored unfairly low due to its translations being assessed more frequently alongside high quality translations (Bojar et al., 2011); (ii) elicit assessment scores via a Likert-style question without intermediate labeling, motivated by medical research showing patients' ratings of their own health to be highly dependent on the exact wording of descriptors (Seymour et al., 1985); (iii) accurate quality control by assessing the consistency of judges with reference only to their own rating distributions, to accurately remove inconsistent crowd-sourced data while avoiding removal of data that legitimately diverges from the scoring strategy of a given expert judge; and (iv) score standardization to avoid bias introduced by legitimate variations in scoring strategies.

Despite efforts to avoid bias in Graham et al. (2013), since DA is a monolingual evaluation of MT that operates via comparison of MT output with a reference translation, it is therefore still possible, while avoiding other sources of bias, that DA incurs reference bias where the level of superficial similarity of translations with reference translations results in an unfair gain, or indeed an unfair disadvantage for systems that yield translations that legitimately deviate from the surface form of reference translations. Following this intuition, Fomicheva and Specia (2016) carry out an investigation into bias in monolingual evaluation of MT and conclude that in a monolingual setting, human assessors of MT are strongly biased by the reference translation. In this paper, we provide further analysis of experiments originally provided in Fomicheva and Specia (2016), in addition to further investigation into the degree to which the intuition about reference bias can be supported.

2 Background

Fomicheva and Specia (2016) provide an investigation into reference bias in monolingual evaluation of MT. 100 Chinese to English MT output translations are assessed by 25 human judges on a five-point scale, in the form of their response (*None*, *Little*, *Much*, *Most*, or *All*) to the following question: *how much of the meaning of the human translation is also expressed in the machine translation?*. Precisely the same 100 translations were assessed by all 25 judges. Human judges were divided into five groups of five: Group 1 (G_1) was shown the source language input and the MT output only and carried out a bilingual assessment, while Groups 2–5 (G_2–G_5) were not shown the source input but instead compared the MT output to a human-generated reference translation. A distinct set of reference translations was assigned to each group G_2–G_5. Inter-annotator agreement (IAA) was measured for pairs of judges as follows (the total number of judge pairs resulting from each setting is provided in parentheses):

- SOURCE: a given pair of judges assessed translations in a bilingual setting (all possible pairs within $G_1 = \binom{5}{2} = 10$ pairs);
- SAME: a given pair of judges assessed translations in a monolingual setting by comparison with precisely the *same reference translation* (the sum of all possible pairs result-

DIFF	SAME	SOURCE
0.163 ± 0.01	0.197 ±0.01	0.190 ± 0.02

Table 1: Average Kappa coefficients and 99% confidence intervals reported in Fomicheva and Specia (2016)

ing from each individual group G$_2$–G$_5$ = $\binom{5}{2} + \binom{5}{2} + \binom{5}{2} + \binom{5}{2} = 40$ pairs);

- DIFF: a given pair of judges assessed translations in a monolingual setting by comparison with a *distinct reference translation* (cross product of judges belonging to the four groups G$_2$–G$_5$ = G$_2$×G$_3$×G$_4$×G$_5$ = 150 pairs).

Reference bias is investigated by comparison of levels of IAA, via Cohen's Kappa (κ) and weighted Kappa coefficients. The hypothesis, although not explicitly stated, is that if agreement of human assessors of MT in SAME is *higher* than that of assessors in DIFF, then the likely cause is reference bias in human assessment scores. Agreement in terms of Cohen's Kappa reported in Fomicheva and Specia (2016) are reproduced here in Table 1, where a small increase of 0.034 in average Kappa is shown for pairs of human assessors in SAME over that of DIFF. To avoid drawing conclusions from a difference that is likely to have occurred simply by chance, confidence intervals (CIs) are provided and the non-overlapping CIs for SAME and DIFF shown in Table 1 provide the basis for the conclusion that IAA is significantly higher for SAME compared to DIFF and subsequently that monolingual evaluation of MT is strongly biased by the reference translation. On examination of the analysis that led to the conclusion of strong reference bias, we unfortunately discover a series of methodological issues with respect to confidence interval estimation, however, that raise doubt about the reliability of this conclusion.[2]

A clear indication of the precise approach to CI estimation attempted in Fomicheva and Specia (2016) is unfortunately not explicitly stated but out of the range of methods that exist the approach that is applied most resembles bootstrap resampling. Conventionally speaking, bootstrap resampling can be applied to CI estimation of a point estimate for a *sample*, D, of size N, by simulating the variance in the *population* sampling distribution (Efron and Tibshirani, 1993). A standard method of estimating CIs via bootstrap resampling is to generate a bootstrap distribution for the statistic of interest made up of M repeat computations of it, each time drawing a random sample of size N from D with replacement. Although most similar to bootstrap resampling, the application in Fomicheva and Specia (2016) to CI estimation of Kappa coefficients diverges in some important ways from a standard application, however. We therefore provide a comparison of the analysis drawn in Fomicheva and Specia (2016) with a standard bootstrap implementation.

Figure 1(a) shows SAME and DIFF bootstrap distributions, reproduced from code released with Fomicheva and Specia (2016), originally yielding non-overlapping CIs that led to the conclusion of strong reference bias.[3] Although the level of statistical significance is reported to be 99%, CIs in Figure 1(a) show that the proportion of each bootstrap distribution was substantially underestimated leading to overly narrow CI limits for both SAME and DIFF. In contrast, Figure 1(b) shows CIs resulting from an accurately computed proportion of 95% of the same bootstrap distribution, where even at the lower level of 95% significance (as opposed to 99%) CIs for SAME and DIFF now overlap, reversing the conclusion of strong reference bias.

In addition, CI estimation diverges from bootstrap resampling with respect to the number of bootstrap samples employed. Since there are a total of N^N possible distinct bootstrap samples for a given sample D (taking order into account), in a conventional bootstrap implementation a Monte Carlo approximation of size M is employed, and the larger M is, the closer the distribution approaches the true bootstrap distribution (Chernick and LaBudde, 2014). In Fomicheva and Specia (2016), CIs are computed via only 50 bootstrap samples, however.[4] Figure 1(c) shows the change in location of CIs for a typical M =1,000, as opposed to M =50 (Figure 1(b)).

[2] We provide a re-analysis of experiment data specifically with respect to Cohen's Kappa. All errors outlined for Cohen's Kappa also lead to the same inaccuracies for weighted Kappa in Fomicheva and Specia (2016), however.

[3] CIs correspond closely to those of the original (Table 1) but differ by a tiny amount due to the randomness involved in regeneration from the code.

[4] We note that M is described as 100 in the publication, but 50 in the released code. Our question raised about the methodology also stands for M =100.

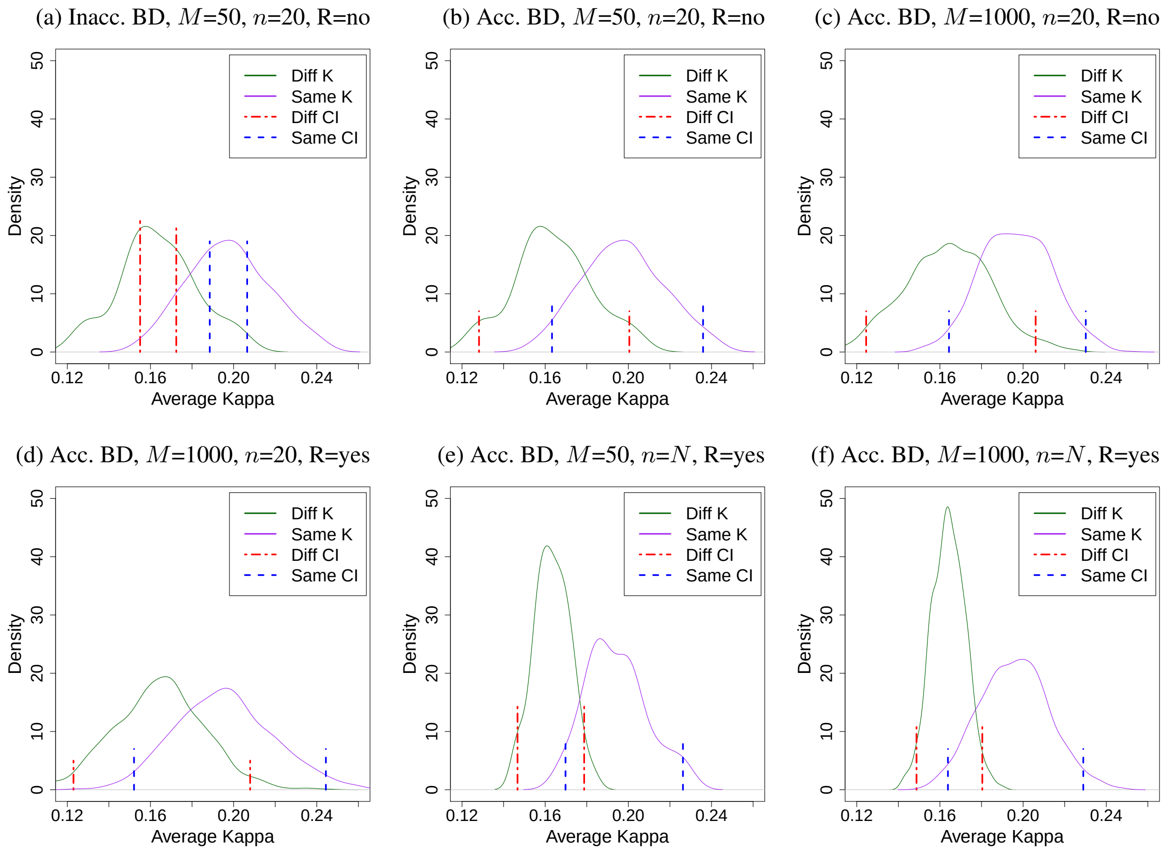


Figure 1: (a) Original bootstrap distribution (BD) and confidence intervals (CI) for average Kappa coefficients when human annotators employ the same reference translation (Same K) or a different reference translation (Diff K) in Fomicheva and Specia (2016) ("Inacc. BD"=inaccurate BD proportion; "Acc. BD"=accurate BD proportion; "M"=number of bootstrap samples; "n"=bootstrap sample size; "R=yes": sampled with replacement; "R=no": sampled without replacement); (b) is (a) with accurate BD proportion; (c) is (b) with conventional M; (d) is (c) with R=yes; (f) is (d) with N=n (N is the full sample size); (e) is (f) with M=50; (f) corresponds to correct BD with all CI errors corrected.

Furthermore, bootstrap distributions in Fomicheva and Specia (2016) are computed by random sampling *without* replacement, and the size of each bootstrap does not equal the original sample size N.[5] Figure 1(d) shows bootstrap distributions of Figure 1(c) when the sampling without replacement error is corrected, and Figure 1(f) shows bootstrap distributions of Figure 1(d) when the sample size error is corrected.

In summary, Figure 1(f) shows all errors with respect to CI estimate in Fomicheva and Specia (2016) corrected, and subsequently CIs for a standard implementation of bootstrap, which can be contrasted to those that led to the original conclusion of strong reference bias in Figure 1(a). CIs in Figure 1(f) for SAME and DIFF now overlap revealing that experiments in Fomicheva and Specia (2016), thus far do not show any evidence of reference bias.

2.1 Measures of Central Tendency

Even if the correct implementation of bootstrap resampling, shown in Figure 1(f), had shown non-overlapping confidence intervals, it would still unfortunately not have been appropriate to draw a conclusion from this of reference bias, however, due to the fact that significant differences are not investigated for the statistic of interest, the Kappa coefficient, but only for a measure of central tendency of two Kappa coefficient distributions, the *average* Kappa of each Kappa distribution. One reason for avoiding a comparison based on significant differences in average Kappa, as opposed to the Kappa point estimates themselves, is that it is

[5] A variant of bootstrap does exist where N is intentionally lowered to appropriately reduce the variance estimate but is only applicable when that of standard bootstrap is known to be over-estimated (Chernick and LaBudde, 2014).

possible for the average of two distributions to be equal, or indeed have a small but non-significant difference, while the underlying distributions differing considerably in several other respects.

Figure 2 shows Kappa coefficient distributions for all pairs of judges in SAME (40 pairs), DIFF (150 pairs) and SOURCE (10 pairs), revealing all distributions to have very similar Kappa coefficient distributions, with the one exception arising for SOURCE, where two of the human annotator pairs had an unusually high agreement level.[6]

A more informative comparison about levels of agreement in SAME and DIFF examines significant differences in Kappa point estimates, as opposed to comparison based on a measure of central tendency. For this reason, despite there being no significant difference in *average Kappa* for SAME and DIFF, we also examine the *proportion of Kappa point estimates of judge pairs in* SAME *that are significantly different from agreement levels of judge pairs in* DIFF, which will provide genuine insight into differences in levels of agreement between the two groups.

Table 2 shows proportions of all judge pairs with significant differences in Kappa point estimates (non-overlapping confidence intervals) for each combination of settings (Revelle, 2014).[7] The number of significant differences in Kappa point estimates for pairs of judges in SAME and DIFF is only 13%, or, in other words, 87% of judge pairs across SAME and DIFF have no significant difference in agreement levels. Table 2 also includes proportions of significant differences for Kappa point estimates resulting from judges belonging to a single setting (significance testing all Kappa of SAME with respect to all *other* Kappa belonging to SAME, for example), revealing that the proportion of significant differences within SAME (12%) to be very similar to that of SAME × DIFF (13%), and similarly for DIFF (12%), with only a single percentage point difference in both cases in proportions of significant differences. Subsequently, even after correcting the measure of central tendency error in Fomicheva and Specia (2016), evidence of reference bias can still not be concluded.

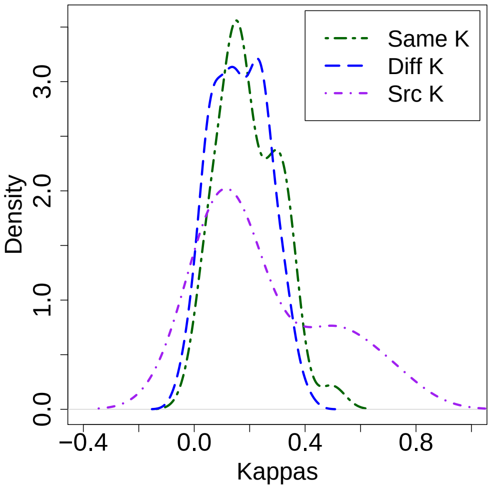


Figure 2: Distribution of Kappa coefficients for translations assessed with the same reference translation ("Same K"), different reference translations ("Diff K") and source sentences ("Src K") (Fomicheva and Specia (2016) data set).

	SOURCE	SAME	DIFF
SOURCE	47% (45)	29% (400)	27% (1,500)
SAME	–	12% (780)	13% (6,000)
DIFF	–	–	12% (11,175)

Table 2: Percentage of human annotator pairs in Fomicheva and Specia (2016) with significant differences in Kappa coefficients for pairs of annotators shown the same reference translation (SAME), different reference translations (DIFF) or the source language input only (SOURCE), total numbers of annotator comparisons in each case are provided within parentheses, numbers of annotator pairs was 10 for SOURCE, 40 for SAME and 150 for DIFF.

2.2 Differences in Ratings

The effect that reference bias may or may not have on actual 1–5 ratings attributed to translations, is again only reported in terms of a measure of central tendency, i.e. average ratings, in Fomicheva and Specia (2016). The average rating of each group shown a distinct reference translation is reported, showing distinct average scores for assessors employing a distinct set of reference translations. Due to the fact that each group had a distinct average rating, the conclusion is drawn that *MT quality is perceived differently depending on the human translation used as gold-standard.* It is however, entirely possible that, the difference in average ratings is in part or even fully caused by

[6] The difference in distributions for SOURCE is exaggerated to some degree due to the total number of annotator pairs in SOURCE being substantially lower than the other two settings (only 10 pairs).

[7] Our re-analysis code is available at https://github.com/qingsongma/percentage-refBias

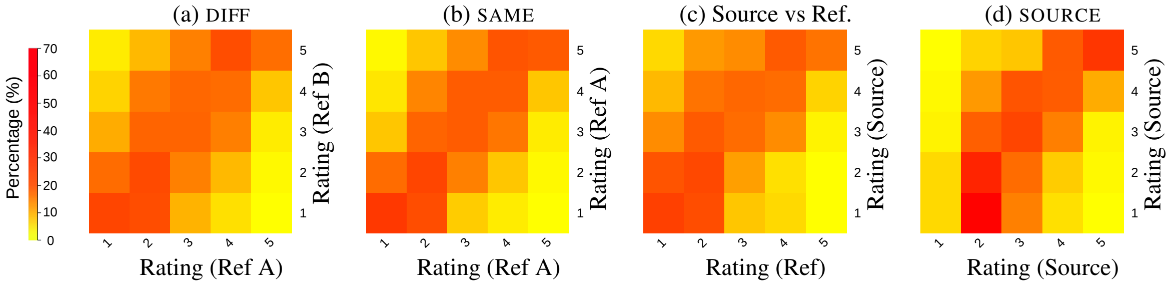


Figure 4: Proportions of 1–5 ratings (1=lowest; 5=highest) for translations when human assessors are shown different reference translations (DIFF), the same reference translation (SAME), the source input versus a reference translation (Source vs Ref.) or the source input (SOURCE) for data in Fomicheva and Specia (2016).

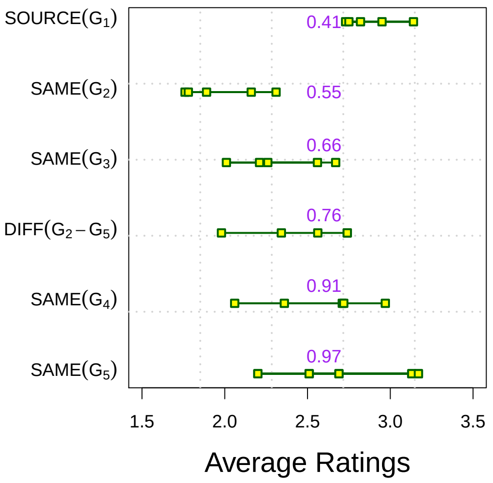


Figure 3: Average rating of human assessors shown the source input (SOURCE), the same reference translation (SAME), or a distinct reference translation (DIFF); range of average ratings provided adjacent to each setting.

the known lack of consistency across human annotators in general.

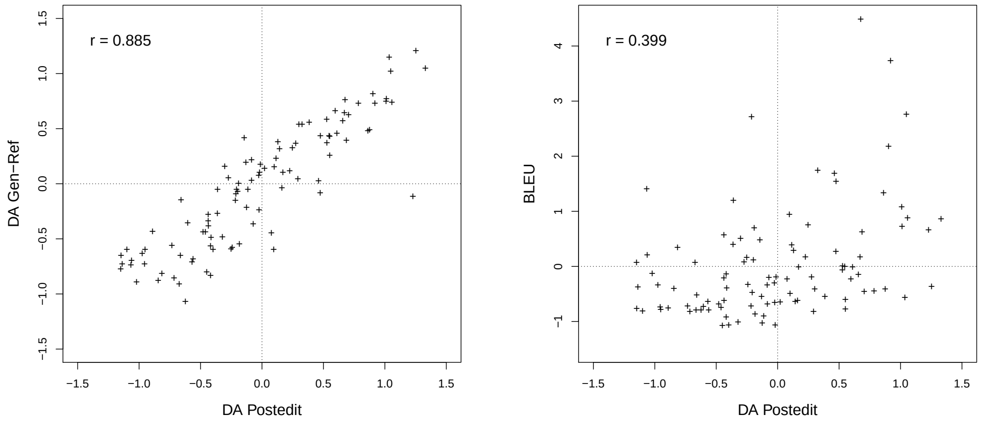
Quite a substantial leap is made therefore between the difference in average ratings and the cause of that difference. To investigate this further, we reproduce the average ratings for assessors shown a distinct reference translation, each represented by a green square along the line labeled “DIFF(G_2–G_5)” in Figure 3, where the overall range in average ratings is 0.76. The extremity of this range is better put into context by comparison with the average rating of human assessors shown the same reference translation, each labeled SAME in Figure 3, where the range of average ratings attributed to human assessors shown *the same reference* can be as large as 0.97 (G5). Thus, it *cannot* be concluded from a difference in average ratings for annotators shown distinct reference translations that the cause of this difference is the reference translation.

However, comparison of ratings based only on averages, again hides detail that an analysis could otherwise benefit from. We therefore examine the distribution of individual ratings attributed to translations, and how well ratings for the same translation correspond when pairs of annotators employ the same or distinct reference translation (or indeed the source input) in Figure 4.[8] The rating pattern in Figure 4 (a) of judge pairs employing a distinct reference translation compared to those in Figure 4 (b), where assessors employ the same reference translation, shows agreement at the level of individual ratings to be almost indistinguishable, showing no evidence of reference bias.

3 Alternate Reference Bias Investigation

Although we can now say that experiments in Fomicheva and Specia (2016) showed no evidence of reference bias, a further issue lies in the fact that low IAA was incurred throughout the study, and low IAA unfortunately provides no assurance with respect to the reliability of conclusions, even when corrected for analytical errors. In addition, the fact that IAA was itself the measure by which bias was investigated is also likely to exacerbate any problems with respect to reliability of conclusions. We therefore provide our own additional investigation into reference bias in monolingual evaluation of MT. Instead of investigating via IAA, we explore the degree to which unfairly high or low ratings might be assigned to translations with respect to

[8] The sum of percentages in a given row equals 100% in each heat map.



(a)

(b)

Figure 5: (a) Scatter plot of direct assessment (DA) scores for 100 Chinese to English translations carried out by comparison with a generic reference translation (DA Gen-Ref) or DA with the reference replaced by a human post-edit of the MT output (DA Postedit); (b) sentence-level (smoothed) BLEU scores for the same translations also plotted against DA POST-EDIT; translations and references of (a) and (b) data set of Fomicheva and Specia (2016); post-edits provided by professional translators with access to the source and MT output only. BLEU and DA scores are standardized for ease of comparison in all plots.

surface similarity or dissimilarity with the reference translation.

Reference-similarity bias is the attribution of unfairly high scores to translations due to high surface-similarity with the reference translation even though the translation is not high quality. A converse kind of reference bias can also occur, which we call *reference-dissimilarity bias*, where unfairly low scores are attributed to translations that are superficially dissimilar to the reference translation but are in fact high quality translations. The challenge in investigating reference bias lies in the ability to accurately distinguish between translations that receive *unfair* scores due to surface-similarity or dissimilarity from those that achieved a *fair* score due to the translation being genuinely high or low quality.

To separate genuine high quality translations from those that score *unfairly* high, we carry out two separate assessments of the same set of translations. Firstly, we carry out a standard monolingual MT evaluation that employs a generic reference translation (GEN-REF setting), the scores that potentially encounter reference bias. Secondly, we carry out an additional human evaluation of the same translations, where, instead of the generic reference, the human assessor compares the MT output with a human post-edit of it (POST-EDIT setting). The latter human assessment is highly unlikely to encounter any form of reference bias because the assessment employs a post-edit of the MT output, which itself will only differ the MT output with respect to the parts of it that are genuinely incorrect. Translations encountering reference-similarity bias can then be identified by a high GEN-REF score combined with a low POST-EDIT score, and vice-versa for reference-dissimilarity, a low GEN-REF score combined with a high POST-EDIT score.

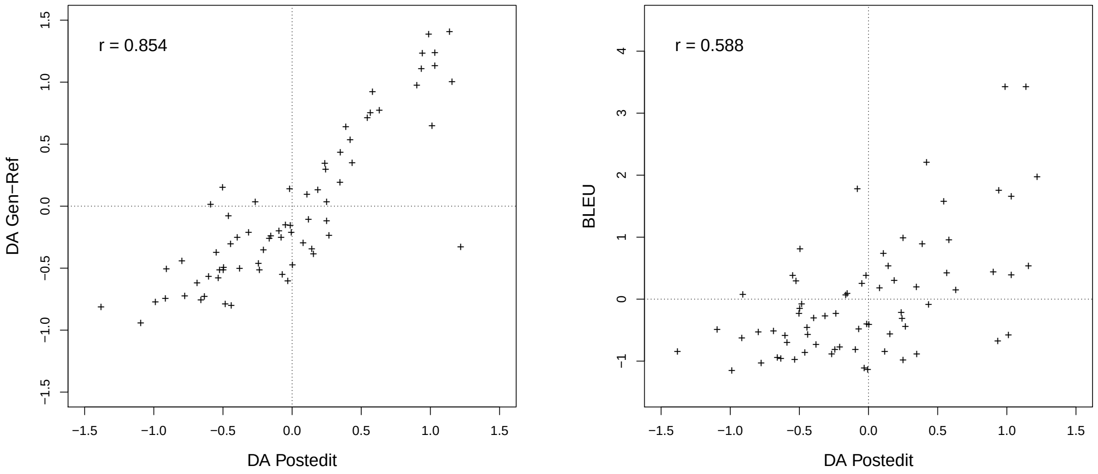
3.1 Reference Bias Experiments

Experiments were carried out using the original 100 Chinese to English translations released by Fomicheva and Specia (2016), in addition to 70 English to Spanish MT translations (WMT-13 Quality Estimation Task 1.1).[9] Professional translators, entirely blind to the purpose of the study, were employed to post-edit the MT outputs used in the POST-EDIT setting, and were shown the source input document and the MT output document only (no reference translations).[10]

Once post-edits had been created, DA was employed in two separate runs on Amazon Mechani-

[9] A single generic reference translation was chosen at random from the Chinese to English data set; only a single reference is available for each translation in the English to Spanish data set.

[10] Post-editors were paid at the standard rate.



(a)

(b)

Figure 6: (a) Scatter plot of direct assessment (DA) scores for 70 English to Spanish translations carried out by comparison with a generic reference translation (DA Gen-Ref) or DA with the reference replaced by a human post-edit of the MT output (DA Postedit); (b) sentence-level (smoothed) BLEU scores for the same translations also plotted against DA POST-EDIT; translations and generic references for (a) and (b) WMT-13 Quality Estimation Task 1.1 (Bojar et al., 2013) data set; post-edits provided by professional translators with access to the source and MT output only. DA and BLEU scores are standardized for ease of comparison in all plots.

cal Turk,[11] once for GEN-REF and once for POST-EDIT. Besides employing distinct reference translations in the assessment, all other set-up criteria were identical for both evaluation settings, including the conventional segment-level DA setting, where a minimum of 15 human assessments are combined into a mean DA score for a given translation, after strict quality control measures and score standardization have been applied.

3.2 Results and Discussion

Figure 5(a) shows a scatter-plot of DA scores attributed to translations for GEN-REF compared to POST-EDIT in the Chinese to English experiment. Translations that encounter reference-dissimilarity bias are expected to appear in the lower-right quadrant of Figure 5(a), receiving an unfairly low GEN-REF score combined with a high POST-EDIT score. As can be seen from Figure 5(a) only a very small number of translations fall into this quadrant, all of which are very closely located to adjacent upper-right and lower-left quadrants. A single translation in Figure 5(a) is an outlier in this respect, receiving a high POST-EDIT score in combination with a lower than average GEN-REF score, possibly indicating reference bias. On closer inspection, however, the score combination is in fact the result of a mistake in the reference translation. Although the low GEN-REF score was the result of an error in the reference translation, a single translation having this score combination is not sufficient evidence to conclude strong reference bias. In future work we would like to investigate the frequency of erroneous reference translations in existing MT test sets, although we expect them to be few, accurate statistics would provide a better indication of the degree to which they could negatively impact the accuracy of DA evaluations.

Figure 5(a) is also void of evidence of reference-similarity bias, as only a small number of translations lie in the upper-left quadrant and are all very close to the origin and/or adjacent quadrants.

Contrasting Figure 5(a), the correspondence of GEN-REF scores to POST-EDIT scores, with Figure 5(b), the correspondence of known reference-biased BLEU scores, in contrast a large number of BLEU scores for translations do encounter reference bias, as seen by the spread of translations appearing across both the bottom-right and upper-left quadrants.

[11] https://www.mturk.com

Similarly for English to Spanish, the correspondence between GEN-REF and POST-EDIT scores for translations are shown in Figure 6(a), where, again, only a small number of translations appear in the bottom-right and upper-left quadrants, all lying very close to adjacent quadrants, again, showing no significant indication of reference bias. A single translation appears to break the trend again, however, receiving a low GEN-REF score combined with a high POST-EDIT score, located in the lower-right quadrant of Figure 6(a). On closer inspection, the low GEN-REF score is the result of something unexpected, as the MT output is in fact an accurate translation while at the same time the generic reference is also correct, but unusually the meaning of the two diverge from each other.[12] Again, a single translation receiving this score combination is not sufficient evidence to conclude reference bias to be a significant problem for monolingual evaluation. The lack of reference bias in Figure 6(a) can again be contrasted to known reference-biased BLEU scores in Figure 6(b) for English to Spanish.

4 Conclusions

In this paper, we provided an investigation into reference bias in monolingual evaluation of MT. Our review of past investigations reveals potential analytical errors and raises questions about the reliability of past conclusions of strong reference bias. This motivates our further investigation for Chinese to English and English to Spanish MT employing direct human assessment in a monolingual MT evaluation setting. Results showed no significant evidence of reference bias, contrary to prior reports and intuition.

Acknowledgments

This project has received funding from NSFC Grant No. 61379086 and the European Union Horizon 2020 research and innovation programme under grant agreement 645452 (QT21) and Science Foundation Ireland in the ADAPT Centre for Digital Content Technology (www.adaptcentre.ie) at Dublin City University funded under the SFI Research Centres Programme (Grant 13/RC/2106) co-funded under the European Regional Development Fund.

[12] Source: *A straightforward man*; MT: *Un hombre sencillo*; Reference: *Un hombre sincero*

References

Ondrej Bojar, Miloš Ercegovčevic, Martin Popel, and Omar F. Zaidan. 2011. A grain of salt for the WMT manual evaluation. In *Proceedings of the 6th Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Edinburgh, Scotland, pages 1–11.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1–44.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, pages 131–198.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2007. (meta-) evaluation of machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Prague, Czech Republic, pages 136–158.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In *Proceedings of the 11th Conference European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Trento, Italy, pages 249–256.

Michael R Chernick and Robert A LaBudde. 2014. *An introduction to bootstrap methods with applications to R*. John Wiley & Sons.

Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman & Hall, New York City, NY.

Marina Fomicheva and Lucia Specia. 2016. Reference bias in monolingual machine translation evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, pages 77–82.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2013. Continuous measurement scales in human evaluation of machine translation. In *Proceedings of the 7th Linguistic Annotation Workshop* *and Interoperability with Discourse*. Association for Computational Linguistics, Sofia, Bulgaria, pages 33–41.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2014. Is machine translation getting better over time? In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Gothenburg, Sweden, pages 443–451.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2016. Can machine translation systems be evaluated by the crowd alone? *Natural Language Engineering* pages 1–28. https://doi.org/10.1017/S1351324915000339.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2015. Accurate evaluation of segment-level machine translation metrics. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 1183–1191.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, PA, pages 311–318.

William Revelle. 2014. psych: Procedures for personality and psychological research. *Northwestern University, Evanston. R package version* 1(1).

Robin A. Seymour, Judy. M. Simpson, J. Ed Charlton, and Michael E. Phillips. 1985. An evaluation of length and end-phrase of visual analogue scales in dental pain. *Pain* 21:177–185.

A Challenge Set Approach to Evaluating Machine Translation

Pierre Isabelle and Colin Cherry
National Research Council Canada
first.last@nrc-cnrc.gc.ca

George Foster
Google*
fosterg@google.com

Abstract

Neural machine translation represents an exciting leap forward in translation quality. But what longstanding weaknesses does it resolve, and which remain? We address these questions with a challenge set approach to translation evaluation and error analysis. A challenge set consists of a small set of sentences, each hand-designed to probe a system's capacity to bridge a particular structural divergence between languages. To exemplify this approach, we present an English–French challenge set, and use it to analyze phrase-based and neural systems. The resulting analysis provides not only a more fine-grained picture of the strengths of neural systems, but also insight into which linguistic phenomena remain out of reach.

1 Introduction

The advent of neural techniques in machine translation (MT) (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014) has led to profound improvements in MT quality. For "easy" language pairs such as English/French or English/Spanish in particular, neural (NMT) systems are much closer to human performance than previous statistical techniques (Wu et al., 2016). This puts pressure on automatic evaluation metrics such as BLEU (Papineni et al., 2002), which exploit surface-matching heuristics that are relatively insensitive to subtle differences. As NMT continues to improve, these metrics will inevitably lose their effectiveness. Another challenge posed by NMT systems is their opacity: while it was usually clear which phenomena were ill-handled by previous statistical systems—and why—these questions are more difficult to answer for NMT.

Src	The repeated calls from his mother **should** have alerted us.
Ref	Les appels répétés de sa mère **auraient** dû nous alerter.
Sys	Les appels répétés de sa mère devraient nous avoir alertés.
Is the subject-verb agreement correct (y/n)? **Yes**	

Figure 1: Example challenge set question.

We propose a new evaluation methodology centered around a *challenge set* of difficult examples that are designed using expert linguistic knowledge to probe an MT system's capabilities. This methodology is complementary to the standard practice of randomly selecting a test set from "real text," which remains necessary in order to predict performance on new text. By concentrating on difficult examples, a challenge set is intended to provide a stronger signal to developers. Although we believe that the general approach is compatible with automatic metrics, we used manual evaluation for the work presented here. Our challenge set consists of short sentences that each focus on one particular phenomenon, which makes it easy to collect reliable manual assessments of MT output by asking direct yes-no questions. An example is shown in Figure 1.

We generated a challenge set for English to French translation by canvassing areas of linguistic divergence between the two language pairs, especially those where errors would be made visible by French morphology. Example choice was also partly motivated by extensive knowledge of the weaknesses of phrase-based MT (PBMT). Neither of these characteristics is essential to our method, however, which we envisage evolving as NMT progresses. We used our challenge set to evalu-

[*] Work performed while at NRC.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2486–2496
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

ate in-house PBMT and NMT systems as well as Google's GNMT system.

In addition to proposing the novel idea of a challenge set evaluation, our contribution includes our annotated English–French challenge set, which we provide in both formatted text and machine-readable formats (see supplemental materials). We also supply further evidence that NMT is systematically better than PBMT, even when BLEU score differences are small. Finally, we give an analysis of the challenges that remain to be solved in NMT, an area that has received little attention thus far.

2 Related Work

A number of recent papers have evaluated NMT using broad performance metrics. The WMT 2016 News Translation Task (Bojar et al., 2016) evaluated submitted systems according to both BLEU and human judgments. NMT systems were submitted to 9 of the 12 translation directions, winning 4 of these and tying for first or second in the other 5, according to the official human ranking. Since then, controlled comparisons have used BLEU to show that NMT outperforms strong PBMT systems on 30 translation directions from the United Nations Parallel Corpus (Junczys-Dowmunt et al., 2016a), and on the IWSLT English-Arabic tasks (Durrani et al., 2016). These evaluations indicate that NMT performs better on average than previous technologies, but they do not help us understand what aspects of the translation have improved.

Some groups have conducted more detailed error analyses. Bentivogli et al. (2016) carried out a number of experiments on IWSLT 2015 English-German evaluation data, where they compare machine outputs to professional post-edits in order to automatically detect a number of error categories. Compared to PBMT, NMT required less post-editing effort overall, with substantial improvements in lexical, morphological and word order errors. NMT consistently out-performed PBMT, but its performance degraded faster as sentence length increased. Later, Toral and Sánchez-Cartagena (2017) conducted a similar study, examining the outputs of competition-grade systems for the 9 WMT 2016 directions that included NMT competitors. They reached similar conclusions regarding morphological inflection and word order, but found an even greater degradation in NMT performance as sentence length increased, perhaps due to these systems' use of subword units.

Most recently, Sennrich (2016) proposed an approach to perform targeted evaluations of NMT through the use of contrastive translation pairs. This method introduces a particular type of error automatically in reference sentences, and then checks whether the NMT system's conditional probability model prefers the original reference or the corrupted version. Using this technique, they are able to determine that a recently-proposed character-based model improves generalization on unseen words, but at the cost of introducing new grammatical errors.

Our approach differs from these studies in a number of ways. First, whereas others have analyzed sentences drawn from an existing bitext, we conduct our study on sentences that are manually constructed to exhibit canonical examples of specific linguistic phenomena. We focus on phenomena that we expect to be more difficult than average, resulting in a particularly challenging MT test suite (King and Falkedal, 1990). These sentences are designed to dive deep into linguistic phenomena of interest, and to provide a much finer-grained analysis of the strengths and weaknesses of existing technologies, including NMT systems.

However, this strategy also necessitates that we work on fewer sentences. We leverage the small size of our challenge set to manually evaluate whether the system's actual output correctly handles our phenomena of interest. Manual evaluation side-steps some of the pitfalls that can come with Sennrich (2016)'s contrastive pairs, as a ranking of two contrastive sentences may not necessarily reflect whether the error in question will occur in the system's actual output.

3 Challenge Set Evaluation

Our challenge set is meant to measure the ability of MT systems to deal with some of the more difficult problems that arise in translating English into French. This particular language pair happened to be most convenient for us, but similar sets can be built for any language pair.

One aspect of MT performance excluded from our evaluation is robustness to sparse data. To control for this, when crafting source and reference sentences, we chose words that occurred at least 100 times in our training corpus (section 4.1).[1]

[1] With two exceptions: *spilt* (58 occurrences), which is

The challenging aspect of the test set we are presenting stems from the fact that the source English sentences have been chosen so that their closest French equivalent will be *structurally divergent* from the source in some crucial way. Translational divergences have been extensively studied in the past—see for example (Vinay and Darbelnet, 1958; Dorr, 1994). We expect the level of difficulty of an MT test set to correlate well with its density in divergence phenomena, which we classify into three main types: morpho-syntactic, lexico-syntactic and purely syntactic divergences.

3.1 Morpho-syntactic divergences

In some languages, word morphology (e.g. inflections) carries more grammatical information than in others. When translating a word towards the richer language, there is a need to recover additional grammatically-relevant information from the context of the target language word. Note that we only include in our set cases where the relevant information is available in the *linguistic* context.[2]

One particularly important case of morpho-syntactic divergence is that of *subject–verb agreement*. French verbs typically have more than 30 different inflected forms, while English verbs typically have 4 or 5. As a result, English verb forms strongly underspecify their French counterparts. Much of the missing information must be filled in through forced agreement in person, number and gender with the grammatical subject of the verb. But extracting these parameters can prove difficult. For example, the agreement features of a coordinated noun phrase are a complex function of the coordinated elements: a) the gender is feminine if all conjuncts are feminine, otherwise masculine wins; b) the conjunct with the smallest person (p1<p2<p3) wins; and c) the number is always plural when the coordination is "et" ("and") but the case is more complex with "ou" ("or").

A second example of morpho-syntactic divergence between English and French is the more explicit marking of the *subjunctive mood* in French subordinate clauses. In the following example, the verb "partiez", unlike its English counterpart, is marked as subjunctive:

> He demanded that you leave immediately. → Il a exigé que vous *partiez* immédiatement.

When translating an English verb within a subordinate clause, the context must be examined for possible subjunctive triggers. Typically these are specific lexical items found in a governing position with respect to the subordinate clause: verbs such as "exiger que", adjectives such as "regrettable que" or subordinate conjunctions such as "à condition que".

3.2 Lexico-syntactic divergences

Syntactically governing words such as verbs tend to impose specific requirements on their complements: they *subcategorize* for complements of a certain syntactic type. But a source language governor and its target language counterpart can diverge on their respective requirements. The translation of such words must then trigger adjustments in the target language complement pattern. We can only examine here a few of the types instantiated in our challenge set.

A good example is *argument switching*. This refers to the situation where the translation of a source verb V_s as V_t is correct but only provided the arguments (usually the subject and the object) are flipped around. The translation of "to miss" as "manquer à" is such a case:

> John misses Mary → Mary *manque à* John.

Failing to perform the switch results in a severe case of mistranslation.

A second example of lexico-syntactic divergence is that of "crossing movement" verbs. Consider the following example:

> Terry swam across the river → Terry *a traversé* la rivière *à la nage*.

The French translation could be glossed as, "Terry crossed the river by swimming." A literal translation such as "Terry a nagé à travers la rivière," is ruled out.

part of an idiomatic phrase, and *guitared* (0 occurrences), which is meant to test the ability to deal with "nonce words" as discussed in section 5.

[2] The so-called Winograd Schema Challenges (en.wikipedia.org/wiki/Winograd_Schema_Challenge) often involve cases where common-sense reasoning is required to correctly choose between two potential antecedent phrases for a pronoun. Such cases become En → Fr translation challenges if the relevant English pronoun is *they* and its alternative antecedents happen to have different grammatical genders in French: *they* → *ils/elles*.

3.3 Syntactic divergences

Some syntactic divergences are not relative to the presence of a particular lexical item but rather stem from differences in the set of available syntactic patterns. Source-language instances of structures missing from the target language must be mapped onto equivalent structures. Here are some of the types appearing in our challenge set.

The position of French pronouns is a major case of divergence from English. French is basically an SVO language like English but it departs from that canonical order when post-verbal complements are pronominalized: the pronouns must then be rendered as *proclitics*, that is phonetically attached to the verb on its left side.

> He gave Mary a book. → Il a donné un livre à Marie.
>
> He gave$_i$ it$_j$ to her$_k$. → Il *le*$_j$ *lui*$_k$ a donné$_i$.

Another example of syntactic divergence between English and French is that of *stranded prepositions*. In both languages, an operation known as "WH-movement" will move a relativized or questioned element to the front of the clause containing it. When this element happens to be a prepositional phrase, English offers the option to leave the preposition in its normal place, fronting only its pronominalized object. In French, the preposition is always fronted alongside its object:

> The girl whom$_i$ he was dancing with$_j$ is rich. → La fille *avec*$_j$ *qui*$_i$ il dansait est riche.

A final example of syntactic divergence is the use of the so-called *middle voice*. While English uses the passive voice in agentless generic statements, French tends to prefer the use of a special pronominal construction where the pronoun "se" has no real referent:

> Caviar is eaten with bread. → Le caviar *se mange* avec du pain.

This completes our exemplification of morphosyntactic, lexico-syntactic and purely syntactic divergences. Our actual test set includes several more subcategories of each type. The ability of MT systems to deal with each such subcategory is then tested using at least three different test sentences. We use short test sentences so as to keep the targeted divergence in focus. The 108 sentences that constitute our current challenge set can be found in Appendix 7.

3.4 Evaluation Methodology

Given the very small size of our challenge set, it is easy to perform a human evaluation of the respective outputs of a handful of different systems. The obvious advantage is that the assessment is then absolute instead of relative to one or a few reference translations.

The intent of each challenge sentence is to test one and only one system capability, namely that of coping correctly with the particular associated divergence subtype. As illustrated in Figure 1, we provide annotators with a question that specifies the divergence phenomenon currently being tested, along with a reference translation with the areas of divergence highlighted. As a result, judgments become straightforward: was the targeted divergence correctly bridged, yes or no?[3] There is no need to mentally average over a number of different aspects of the test sentence as one does when rating the global translation quality of a sentence, e.g. on a 5-point scale. However, we acknowledge that measuring translation performance on complex sentences exhibiting many different phenomena remains crucial. We see our approach as being complementary to evaluations of overall translation quality.

One consequence of our divergence-focused approach is that faulty translations will be judged as successes when the faults lie outside of the targeted divergence zone. However, this problem is mitigated by our use of short test sentences.

4 Machine Translation Systems

We trained state-of-the-art neural and phrase-based systems for English-French translation on data from the WMT 2014 evaluation.

4.1 Data

We used the LIUM shared-task subset of the WMT 2014 corpora,[4] retaining the provided tokenization

[3] Sometimes the system produces a translation that circumvents the divergence issue. For example, it may dodge a divergence involving adverbs by reformulating the translation to use an adjective instead. In these rare cases, we instruct our annotators to abstain from making a judgment, regardless of whether the translation is correct or not.

[4] http://www.statmt.org/wmt14/translation-task.html http://www-lium.univ-lemans.fr/∼schwenk/nnmt-shared-task

corpus	lines	en words	fr words
train	12.1M	304M	348M
mono	15.9M	—-	406M
dev	6003	138k	155k
test	3003	71k	81k

Table 1: Corpus statistics. The WMT12/13 eval sets are used for dev, and the WMT14 eval set is used for test.

and corpus organization, but mapping characters to lowercase. Table 1 gives corpus statistics.

4.2 Phrase-based systems

To ensure a competitive PBMT baseline, we performed phrase extraction using both IBM4 and HMM alignments with a phrase-length limit of 7; after frequency pruning, the resulting phrase table contained 516M entries. For each extracted phrase pair, we collected statistics for the hierarchical reordering model of Galley and Manning (2008).

We trained an NNJM model (Devlin et al., 2014) on the HMM-aligned training corpus, with input and output vocabulary sizes of 64k and 32k. Words not in the vocabulary were mapped to one of 100 mkcls classes. We trained for 60 epochs of 20k × 128 minibatches, yielding a final dev-set perplexity of 6.88.

Our set of log-linear features consisted of forward and backward Kneser-Ney smoothed phrase probabilities and HMM lexical probabilities (4 features); hierarchical reordering probabilities (6); the NNJM probability (1); a set of sparse features as described by Cherry (2013) (10,386); word-count and distortion penalties (2); and 5-gram language models trained on the French half of the training corpus and the French monolingual corpus (2). Tuning was carried out using batch lattice MIRA (Cherry and Foster, 2012). Decoding used the cube-pruning algorithm of Huang and Chiang (2007), with a distortion limit of 7.

We include two phrase-based systems in our comparison: PBMT-1 has data conditions that exactly match those of the NMT system, in that it does not use the language model trained on the French monolingual corpus, while PBMT-2 uses both language models.

4.3 Neural systems

To build our NMT system, we used the Nematus toolkit,[5] which implements a single-layer neural sequence-to-sequence architecture with attention (Bahdanau et al., 2015) and gated recurrent units (Cho et al., 2014). We used 512-dimensional word embeddings with source and target vocabulary sizes of 90k, and 1024-dimensional state vectors. The model contains 172M parameters.

We preprocessed the data using a BPE model learned from source and target corpora (Sennrich et al., 2016). Sentences longer than 50 words were discarded. Training used the Adadelta algorithm (Zeiler, 2012), with a minibatch size of 100 and gradients clipped to 1.0. It ran for 5 epochs, writing a checkpoint model every 30k minibatches. Following Junczys-Dowmunt et al. (2016b), we averaged the parameters from the last 8 checkpoints. To decode, we used the AmuNMT decoder (Junczys-Dowmunt et al., 2016a) with a beam size of 4.

While our primary results will focus on the above PBMT and NMT systems, where we can describe replicable configurations, we have also evaluated Google's production system,[6] which has recently moved to NMT (Wu et al., 2016). Notably, the "GNMT" system uses (at least) 8 encoder and 8 decoder layers, compared to our 1 layer for each, and it is trained on corpora that are "two to three decimal orders of magnitudes bigger than the WMT." The evaluated outputs were downloaded in December 2016.

5 Experiments

The 108-sentence English–French challenge set presented in Appendix 7 was submitted to the four MT systems described in section 4: PBMT-1, PBMT-2, NMT, and GNMT. Three bilingual native speakers of French rated each translated sentence as either a success or a failure according to the protocol described in section 3.4. For example, the 26 sentences of the subcategories S1–S5 of Appendix 7 are all about different cases of subject-verb agreement. The corresponding translations were judged successful if and only if the translated verb correctly agrees with the translated subject.

The different system outputs for each source sentence were grouped together to reduce the burden on the annotators. That is, in figure 1, anno-

[5] https://github.com/rsennrich/nematus

[6] https://translate.google.com

tators were asked to answer the question for each of four outputs, rather than just one as shown. The outputs were listed in random order, without identification. Questions were also presented in random order to each annotator. Appendix A in the supplemental materials contains the instructions shown to the annotators.

5.1 Quantitative comparison

Table 2 summarizes our results in terms of percentage of successful translations, globally and over each main type of divergence. For comparison with traditional metrics, we also include BLEU scores measured on the WMT 2014 test set.

As we can see, the two PBMT systems fare very poorly on our challenge set, especially in the morpho-syntactic and purely syntactic types. Their somewhat better handling of lexico-syntactic issues probably reflects the fact that PBMT systems are naturally more attuned to lexical cues than to morphology or syntax. The two NMT systems are clear winners in all three categories. The GNMT system is best overall with a success rate of 68%, likely due to the data and architectural factors mentioned in section 4.3.[7]

WMT BLEU scores correlate poorly with challenge-set performance. The large gap of 2.3 BLEU points between PBMT-1 and PBMT-2 corresponds to only a 1% gain on the challenge set, while the small gap of 0.4 BLEU between PBMT-2 and NMT corresponds to a 21% gain.

Inter-annotator agreement (final column in table 2) is excellent overall, with all three annotators agreeing on almost 90% of system outputs. Syntactic divergences appear to be somewhat harder to judge than other categories.

5.2 Qualitative assessment of NMT

We now turn to an analysis of the strengths and weaknesses of neural MT through the microscope of our divergence categorization system, hoping that this may help focus future research on key issues. In this discussion we ignore the results obtained by PBMT-2 and compare: a) the results obtained by PBMT-1 to those of NMT, both systems having been trained on the same dataset; and b) the results of these two systems with those of Google NMT which was trained on a much larger dataset.

In the remainder of the present section we will refer to the sentences of our challenge set using the subcategory-based numbering scheme S1-S26 as assigned in Appendix 7. A summary of the category-wise performance of PBMT-1, NMT and Google NMT is provided in Table 3.

Strengths of neural MT

Overall, both neural MT systems do much better than PBMT-1 at bridging divergences. In the case of morpho-syntactic divergences, we observe a jump from 16% to 72% in the case of our two local systems. This is mostly due to the NMT system's ability to deal with many of the more complex cases of subject-verb agrement:

- *Distractors.* The subject's head noun agreement features get correctly passed to the verb phrase across intervening noun phrase complements (sentences S1a–c).
- *Coordinated verb phrases.* Subject agreement marks are correctly distributed across the elements of such verb phrases (S3a–c).
- *Coordinated subjects.* Much of the logic that is at stake in determining the agreement features of coordinated noun phrases (cf. our relevant description in section 3.1) appears to be correctly captured in the NMT translations of S4.
- *Past participles.* Even though the rules governing French past participle agreement are notoriously difficult (especially after the "avoir" auxiliary), they are fairly well captured in the NMT translations of (S5b–e).

The NMT systems are also better at handling lexico-syntactic divergences. For example:

- *Double-object verbs.* There are no such verbs in French and the NMT systems perform the required adjustments flawlessly (sentences S8a–S8c).
- *Overlapping subcat frames.* NMT systems manage to discriminate between an NP complement and a sentential complement starting with an NP: cf. *to know NP* versus *to know NP is VP* (S11b–e)
- *NP-to-VP complements.* These English infinitival complements often need to be rendered as finite clauses in French and the NMT systems are better at this task (S12a–c).

[7] We cannot offer a full comparison with the pre-NMT Google system. However, in October 2016 we ran a smaller 35-sentence version of our challenge set on both the Google system and our PBMT-1 system. The Google system only got 4 of those examples right (11.4%) while our PBMT-1 got 6 right (17.1%).

Divergence type	PBMT-1	PBMT-2	NMT	Google NMT	Agreement
Morpho-syntactic	16%	16%	72%	65%	94%
Lexico-syntactic	42%	46%	52%	62%	94%
Syntactic	33%	33%	40%	75%	81%
Overall	31%	32%	53%	68%	89%
WMT BLEU	34.2	36.5	36.9	—	—

Table 2: Summary performance statistics for each system under study, including challenge set success rate grouped by linguistic category (aggregating all positive judgments and dividing by total judgments), as well as BLEU scores on the WMT 2014 test set. The final column gives the proportion of system outputs on which all three annotators agreed.

Finally, NMT systems also turn out to better handle purely syntactic divergences. For example:

- *Yes-no question syntax.* The differences between English and French yes-no question syntax are correctly bridged by the two NMT systems (S17a–c).
- *French proclitics.* NMT systems are significantly better at transforming English pronouns into French proclitics, i.e. moving them before the main verb and case-inflecting them correctly (S23a–e).
- Finally, we note that the Google system manages to overcome several additional challenges. It correctly translates *tag questions* (S18a–c), constructions with *stranded prepositions* (S19a–f), most cases of the *inalienable possession* construction (S25a–e) as well as *zero relative pronouns* (S26a–c).

The large gap observed between the results of the in-house and Google NMT systems indicates that current neural MT systems are extremely data hungry. But given enough data, they can successfully tackle some challenges that are often thought of as extremely difficult. A case in point here is that of stranded prepositions (see discussion in section 3.3), in which we see the NMT model capture some instances of WH-movement, the textbook example of long-distance dependencies.

Weaknesses of neural MT

In spite of its clear edge over PBMT, NMT is not without some serious shortcomings. We already mentioned the degradation issue with long sentence which, by design, could not be observed with our challenge set. But an analysis of our results will reveal many other problems. Globally, we note that even using a staggering quantity of data and a highly sophisticated NMT model, the Google system fails to reach the 70% mark on our challenge set. The fine-grained error categorization associated with the challenge set will help us single out precise areas where more research is needed. Here are some relevant observations.

Incomplete generalizations. In several cases where partial results might suggest that NMT has correctly captured some basic generalization about linguistic data, further instances reveals that this is not fully the case.

- *Agreement logic.* The logic governing the agreement features of coordinated noun phrases (see section 3.1) has been mostly captured by the NMT systems (cf. the 12 sentences of S4), but there are some gaps. For example, the Google system runs into trouble with mixed-person subjects (sentences S4d1–3).
- *Subjunctive mood triggers.* While some subjunctive mood triggers are correctly registered (e.g. "demander que" and "malheureux que"), the case of such a highly frequent subordinate conjunction as *provided that* → *à condition que* is somehow being missed (sentence S6a–c).
- *Noun compounds.* The French translation of an English compound N_1 N_2 is usually of the form N_2 *Prep* N_1. For any given headnoun N_2 the correct preposition *Prep* depends on the semantic class of N_1. For example *steel/ceramic/plastic knife* → *couteau **en** acier/céramique/plastique* but *butter/meat/steak knife* → *couteau **à** beurre/viande/steak*. Given that neural models are known to perform some semantic generalizations, we find their performance disappointing on our compound noun examples (S14a–i).

Category	Subcategory	#	PBMT-1	NMT	Google NMT
Morpho-syntactic	Agreement across distractors	3	0%	100%	100%
	through control verbs	4	25%	25%	25%
	with coordinated target	3	0%	100%	100%
	with coordinated source	12	17%	92%	75%
	of past participles	4	25%	75%	75%
	Subjunctive mood	3	33%	33%	67%
Lexico-syntactic	Argument switch	3	0%	0%	0%
	Double-object verbs	3	33%	67%	100%
	Fail-to	3	67%	100%	67%
	Manner-of-movement verbs	4	0%	0%	0%
	Overlapping subcat frames	5	60%	100%	100%
	NP-to-VP	3	33%	67%	67%
	Factitives	3	0%	33%	67%
	Noun compounds	9	67%	67%	78%
	Common idioms	6	50%	0%	33%
	Syntactically flexible idioms	2	0%	0%	0%
Syntactic	Yes-no question syntax	3	33%	100%	100%
	Tag questions	3	0%	0%	100%
	Stranded preps	6	0%	0%	100%
	Adv-triggered inversion	3	0%	0%	33%
	Middle voice	3	0%	0%	0%
	Fronted should	3	67%	33%	33%
	Clitic pronouns	5	40%	80%	60%
	Ordinal placement	3	100%	100%	100%
	Inalienable possession	6	50%	17%	83%
	Zero REL PRO	3	0%	33%	100%

Table 3: Summary of scores by fine-grained categories. "#" reports number of questions in each category, while the reported score is the percentage of questions for which the divergence was correctly bridged. For each question, the three human judgments were transformed into a single judgment by taking system outputs with two positive judgments as positive, and all others as negative.

- The so-called French "inalienable possession" construction arises when an agent performs an action on one of her body parts, e.g. *I brushed my teeth*. The French translation will normally replace the possessive article with a definite one and introduce a reflexive pronoun, e.g. *Je me suis brossé les dents* ('I brushed myself the teeth'). In our dataset, the Google system gets this right for examples in the first and third persons (sentences S25a,b) but fails to do the same with the example in the second person (sentence S25c).

Then there are also phenomena that current NMT systems, even with massive amounts of data, appear to be completely missing:

- *Common and syntactically flexible idioms.* While PBMT-1 produces an acceptable translation for half of the idiomatic expressions of S15 and S16, the local NMT system misses them all and the Google system does barely better. NMT systems appear to be short on raw memorization capabilities.
- *Control verbs.* Two different classes of verbs can govern a subject NP, an object NP plus an infinitival complement. With verbs of the "object-control" class (e.g. "persuade"), the object of the verb is understood as the semantic subject of the infinitive. But with those of the "subject-control" class (e.g. "promise"), it is rather the subject of the verb which plays that semantic role. None of the systems tested here appear to get a grip on subject control cases, as evidenced by the lack of correct feminine agreement on the French adjectives in sentences S2b–d.
- *Argument switching verbs.* All systems tested

here mistranslate sentences S7a–c by failing to perform the required argument switch: *NP_1 misses NP_2 → NP_2 manque à NP_1.*

- *Crossing movement verbs.* None of the systems managed to correctly restructure the regular manner-of-movement verbs e.g. *swim across X → traverser X à la nage* in sentences S10a-c. Unsurprisingly, all systems also fail on the even harder example S10d, in which the "nonce verb" *guitared* is a spontaneous derivation from the noun *guitar* being cast as an ad hoc manner-of-movement verb. [8]
- *Middle voice.* None of the systems tested here were able to recast the English "generic passive" of S21a–c into the expected French "middle voice" pronominal construction.

6 Conclusions

We have presented a radically different kind of evaluation for MT systems: the use of challenge sets designed to stress-test MT systems on "hard" linguistic material, while providing a fine-grained linguistic classification of their successes and failures. This approach is not meant to replace our community's traditional evaluation tools but to supplement them.

Our proposed error categorization scheme makes it possible to bring to light different strengths and weaknesses of PBMT and neural MT. With the exception of idiom processing, in all cases where a clear difference was observed it turned out to be in favor of neural MT. A key factor in NMT's superiority appears to be its ability to overcome many limitations of n-gram language modeling. This is clearly at play in dealing with subject-verb agreement, double-object verbs, overlapping subcategorization frames and last but not least, the pinnacle of Chomskyan linguistics, WH-movement (in this case, stranded prepositions).

But our challenge set also brings to light some important shortcomings of current neural MT, regardless of the massive amounts of training data it may have been fed. As may have been already known or suspected, NMT systems struggle with the translation of idiomatic phrases. Perhaps more interestingly, we notice that neural MT's impressive generalizations still seem somewhat brittle. For example, the NMT system can appear to have mastered the rules governing subject-verb agreement or inalienable possession in French, only to trip over a rather obvious instantiation of those rules. Probing where these boundaries are, and how they relate to the neural system's training data and architecture is an obvious next step.

7 Future Work

It is our hope that the insights derived from our challenge set evaluation will help inspire future MT research, and call attention to the fact that even "easy" language pairs like English–French still have many linguistic issues left to be resolved. But there are also several ways to improve and expand upon our challenge set approach itself.

First, though our human judgments of output sentences allowed us to precisely assess the phenomena of interest, this approach is not scalable to large sets, and requires access to native speakers in order to replicate the evaluation. It would be interesting to see whether similar scores could be achieved through automatic means. The existence of human judgments for this set provides a gold-standard by which proposed automatic judgments may be meta-evaluated.

Second, the construction of such a challenge set requires in-depth knowledge of the structural divergences between the two languages of interest. A method to automatically create such a challenge set for a new language pair would be extremely useful. One could imagine approaches that search for divergences, indicated by atypical output configurations, or perhaps by a system's inability to reproduce a reference from its own training data. Localizing a divergence within a difficult sentence pair would be another useful subtask.

Finally, we would like to explore how to train an MT system to improve its performance on these divergence phenomena. This could take the form of designing a curriculum to demonstrate a particular divergence to the machine, or altering the network structure to capture such generalizations.

Acknowledgments

We would like to thank Cyril Goutte, Eric Joanis and Michel Simard, who graciously spent the time required to rate the output of four different MT systems on our challenge sentences. We also thank Roland Kuhn for valuable discussions, and comments on an earlier version of the paper.

[8] On the concept of nonce word, see https://en.wikipedia.org/wiki/Nonce_word.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations (ICLR)*. San Diego, USA. http://arxiv.org/abs/1409.0473.

Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. 2016. Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 257–267. https://aclweb.org/anthology/D16-1025.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, pages 131–198. http://www.aclweb.org/anthology/W16-2301.

Colin Cherry. 2013. Improved reordering for phrase-based translation using sparse features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 22–31. http://www.aclweb.org/anthology/N13-1003.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Montréal, Canada, pages 427–436. http://www.aclweb.org/anthology/N12-1047.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734. http://www.aclweb.org/anthology/D14-1179.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 1370–1380. http://www.aclweb.org/anthology/P14-1129.

Bonnie J. Dorr. 1994. Machine translation divergences: a formal description and proposed solution. *Computational Linguistics* 20:4. http://aclweb.org/anthology/J/J94/J94-4004.pdf.

Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and Stephan Vogel. 2016. QCRI machine translation systems for IWSLT 16. In *Proceedings of the 13th International Workshop on Spoken Language Translation (IWSLT)*. Seattle, Washington. https://workshop2016.iwslt.org/downloads/qcri-machine-translation.pdf.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Honolulu, Hawaii, pages 848–856. http://www.aclweb.org/anthology/D08-1089.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, Prague, Czech Republic, pages 144–151. http://www.aclweb.org/anthology/P07-1019.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016a. Is neural machine translation ready for deployment? a case study on 30 translation directions. In *Proceedings of the 13th International Workshop on Spoken Language Translation (IWSLT)*. Seattle, Washington.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Rico Sennrich. 2016b. The amu-uedin submission to the wmt16 news translation task: Attention-based nmt models as feature functions in phrase-based smt. In *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, Berlin, Germany, pages 319–325. http://www.aclweb.org/anthology/W16-2316.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1700–1709. http://www.aclweb.org/anthology/D13-1176.

Margaret King and Kirsten Falkedal. 1990. Using test suites in evaluation of machine translation systems. In *Proceedings of the 1990 Conference on Computational Linguistics*. Association for Computational Linguistics, Helsinki, Finland. http://aclweb.org/anthology/C/C90/C90-2037.pdf.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, pages 311–318. https://doi.org/10.3115/1073083.1073135.

Rico Sennrich. 2016. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. *CoRR* abs/1612.04629. http://arxiv.org/abs/1612.04629.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1715–1725. http://www.aclweb.org/anthology/P16-1162.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., pages 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Antonio Toral and Víctor M. Sánchez-Cartagena. 2017. A multifaceted evaluation of neural versus statistical machine translation for 9 language directions. In *Proceedings of the The 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Valencia, Spain, pages 1063–1073. http://aclweb.org/anthology/E/E17/E17-1100.pdf.

Jean-Paul Vinay and Jean Darbelnet. 1958. *Stylistique comparée du français et de l'anglais*, volume 1. Didier, Paris.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144. http://arxiv.org/abs/1609.08144.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR* abs/1212.5701. http://arxiv.org/abs/1212.5701.

Supplemental Materials

The supplemental materials comprise two separate files:

- `challenge-a.pdf`—instructions to authors, and rendered version of the challenge set (with annotator scores); and
- `Challenge_set-v2hA.json`—machine-readable version of the challenge set.

Knowledge Distillation for Bilingual Dictionary Induction

Ndapandula Nakashole and **Raphael Flauger**
Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093
nnakashole@eng.ucsd.edu

Abstract

Leveraging zero-shot learning to learn mapping functions between vector spaces of different languages is a promising approach to bilingual dictionary induction. However, methods using this approach have not yet achieved high accuracy on the task. In this paper, we propose a bridging approach, where our main contribution is a knowledge distillation training objective. As teachers, rich resource translation paths are exploited in this role. And as learners, translation paths involving low resource languages learn from the teachers. Our training objective allows seamless addition of teacher translation paths for any given low resource pair. Since our approach relies on the quality of monolingual word embeddings, we also propose to enhance vector representations of both the source and target language with linguistic information. Our experiments on various languages show large performance gains from our distillation training objective, obtaining as high as 17% accuracy improvements.

1 Introduction

In traditional supervised learning, a classifier is trained on a labeled dataset of the form $(\mathbf{X}, \mathbf{Y})$. Each $x_i \in X$ is a feature vector representing a single training instance and $y_i \in Y$ is the label associated with x_i. In zero-shot learning (Mitchell et al., 2008), at test time we can encounter a test instance x_j whose corresponding label was not seen at training time. This setting occurs in domains where $\mathbf{Y}$ can take on many values, and obtaining labeled examples for all possible Y values is expensive. Computer vision is one such domain, where there are thousands of objects a system needs to recognize yet at training time we may only see examples of some of the objects. In zero-shot learning, instead of learning parameters associated with each possible label in Y, the learning task is cast as a problem of learning a single mapping function from the vector space of input instances to the vector space of the output labels. The resulting induced function can then be applied to test instances x_j whose labels may not have been seen at training time, producing a projected vector, $\hat{y}_j$, in the label space. The nearest neighbor of the mapped vector in the label space is then considered to be the label of x_j.

In this paper, we study zero-shot learning in the context of bilingual dictionary induction, which is the problem of mapping words from a source language to equivalent words in a target language. The label space is the full vocabulary of the target language which can be on the order of millions of tokens. First, word embeddings are learned separately for each language, and second, using a given seed dictionary, we train a mapping function to connect the two monolingual vector spaces, thereby facilitating bilingual dictionary induction. The advantage of zero-shot learning is that it can help reduce the amount of labeled data for applications with many possible labels, such as the application we study in this paper, bilingual dictionary induction. However, the state-of-the-art accuracy on zero-shot bilingual dictionary induction is still low. On the task of English to Italian $(en \rightarrow it)$, top-1 and top-10 accuracies are around 40% and 60%, respectively (Lazaridou et al., 2015; Dinu et al., 2014).

An important aspect of zero-shot learning for bilingual dictionary induction is that, it relies on availability of a large seed dictionary[1]. Such large

[1] 5000 seed pairs for the $(en \rightarrow it)$ dataset.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2497–2506
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

$$\begin{tikzcd}
es \arrow[r, dashed] & en & nl \arrow[r, dashed] & en & sv \arrow[r, dashed] & en \\
pt \arrow[u, dashed] \arrow[ur] & & af \arrow[u, dashed] \arrow[ur] & & da \arrow[u, dashed] \arrow[ur] &
\end{tikzcd}$$

Figure 1: Trilingual paths for Portuguese(pt) to English(en) via Spanish (es), Afrikaans(af) to (en) via Dutch (nl), and Danish(da) to (en) via Swedish(sv).

training dictionaries might not be available for all languages. However, for a given language with only a small seed dictionary, there could be a highly related language with a much larger seed dictionary. For example, we might have a small seed dictionary for translating Portuguese to English ($pt \rightarrow en$), but a large seed dictionary for translating Spanish to English language ($es \rightarrow en$). At training time, we can train the ($pt \rightarrow en$) mapping function not only using the small seed dictionary, but also make use of the trilingual path going through Spanish, ($pt \rightarrow es \rightarrow en$). Since pt and es are highly related, a small amount of data may be sufficient to learn the projection ($pt \rightarrow es$). This is the idea of using a bridge or pivot language in machine translation (Utiyama and Isahara, 2007). Our contribution is a knowledge distillation training objective function that encourages the mapping function ($pt \rightarrow en$) to predict the true English target words as well as to match the predictions of the trilingual path ($pt \rightarrow es \rightarrow en$) within a margin. This is approach allows seamless Example trilingual paths are shown in Figure 1.

By setting up our objective function in this way, we are distilling knowledge (Bucilu et al., 2006; Hinton et al., 2015) from the trilingual paths to train a single mapping function for ($pt \rightarrow en$). In our experiments, we show performance gains for several language pairs, 17% for top-10 precision for ($pt \rightarrow en$). We also show that, for a given language pair, our objective seamlessly allows us to distill from several related languages. Moreover, we learn weights for each of the distillation paths, thereby automatically learning indicative weights of how useful each distillation path is. Finally, we show that even when we only use unlabeled data to distill knowledge from trilingual paths, we still obtain performance gains over a model trained on a small seed dictionary.

Since our approach relies on the quality of monolingual word embeddings, we also propose to enhance vector representations of both the source and target language with linguistic information. In particular, we augment word vectors with additional dimensions capturing corpus statistics of part-of-speech tags of words. Second, we model sub-word information in the vector representations of words.

2 Related Work

Cross Vector Space Mapping with Seed Dictionaries. Our work is most related to models that do zero-shot learning for bilingual dictionary induction, using maps between vector spaces with seed dictionaries as training data. Examples include the models of (Mikolov et al., 2013; Dinu et al., 2014; Lazaridou et al., 2015; Vulic and Korhonen, 2016). Like these approaches, we first learn word embeddings for each language, then use a seed dictionary to train a mapping function between the two vector spaces. In a departure from these prior methods, we propose to distill knowledge from trilingual paths of nearby languages for languages with small seed dictionaries using a distillation training objective. Additionally, we model linguistic information in the vector space of the source and target languages. Another line of research in this vein is the work of (Vulic and Korhonen, 2016), who analyze how properties of the seed dictionary affect bilingual dictionary induction across different dimensions (i.e., lexicon source, lexicon size, translation method, translation pair reliability). However, methodologically, their approach is based on prior work (Mikolov et al., 2013; Dinu et al., 2014).

Bilingual word embeddings. There is a rich body of work on bilingual embeddings. Bilingual word embedding learning methods produce a shared bilingual word embedding space where words from two languages are represented in the new space so that similar words, which may be in different languages, have similar representations. Such bilingual word embeddings have been used in a number of tasks including semantic word similarity (Faruqui and Dyer, 2014; Ammar et al., 2016) learning bilingual word lexicons (Mikolov et al., 2013; Gouws et al., 2015; Vulic and Korhonen, 2016), parsing (Guo et al., 2015; Täckström et al., 2012), information retrieval (Vulic and Moens, 2015), and cross-lingual document classification (Klementiev et al., 2012; Kočiskỳ et al., 2014).

Some bilingual word embedding methods such as (Blunsom and Hermann, 2014; Gouws et al., 2015) require sentence or word aligned data, which our approach does not require. We compare our approach to the bilingual embeddings produced by the recent method of (Ammar et al., 2016). Like our approach, this work does not require availability of parallel corpora but only a seed dictionary.

On the aspect of enriching word embeddings with linguistic knowledge for the purpose of machine translation, Sennrich and Barry (Sennrich and Haddow, 2016) introduce linguistic features in sequence to sequence neural machine translation. Like our work, they also represent such features in the embedding layer. In addition to part-of-speech tags and morphological features, they also use syntactic dependency labels which are not applicable to our model since we work at the word level while their model is at the sentence level.

Knowledge Distillation. Knowledge distillation was introduced for model compression to learn small models from larger models (Bucilu et al., 2006; Hinton et al., 2015). For example, from a large neural network model a smaller model can be distilled such that it generalizes in the same way as the large model (Romero et al., 2014). Knowledge distillation was also used by (Hu et al., 2016) to distill knowledge from logical rules in the tasks of named entity recognition and sentiment analysis, thereby enforcing constraints on the trained model. Our approach is different from this prior work on knowledge distillation in that we distill knowledge from mapping functions of related languages into mapping functions of languages with only small seed dictionaries.

Domain adaptation, for which there is a long history, is also related to our work (Ben-David et al., 2007; Daumé III, 2007; Pan et al., 2010; Long and Wang, 2015). (Daumé III, 2007) proposed feature augmentation, suggesting that a model should have features that are general across domains, as well as features that are domain-specific. Thus the model learns from all domains while preserving domain-specific information. These kinds of models have to be retrained when a new domain is added. Our work however only has to train mapping functions that involve a new language, all others can be distilled without retraining them.

3 Embedding Linguistic Information

Since our approach relies on the quality of monolingual word embeddings, we would like to work with high quality word embeddings. We therefore, first seek to enhance the vector representations of words in the source and target languages so that they can capture useful linguistic information. The intuition is that such information can help narrow down the words in the target language that are considered valid translations for a given source language word. To that end, we model both part of speech (POS) tag distributions of words and subword information in the vector representations.

3.1 Part of Speech Distributions

The idea behind modeling POS tags is that words should have the same part of speech tag in different languages. For example, if we are translating the noun *Katze* from German to English, in English we expect the singular noun *cat* and not the plural *cats*. While this information may be monolithically represented in word vectors generated by embedding methods such as Skip-gram and CBOW, here we seek to explicitly model POS tags. Since each word can have multiple POS tags, we model a word's part of speech information as a distribution over all the possible POS tags that it can take on. We learn POS tag statistics by first tagging a large corpus of each language, we then use tag counts to generate distributions. For example, if the English word, *bark* appears tagged as a verb 30 times in our corpus, and tagged as a noun 10 times, we generate a vector which puts 2/3 in the verb direction, and 1/3 in the noun direction, and 0 in the directions of all other POS tags. While these statistics can be noisy, we hope they can still provide useful signals. We use the universal POS tags, there are 12 tags in the universal POS tags (Petrov et al., 2011).

For a given word w, we compute a vector representation $w_i \in \mathbb{R}^d$ using a word embedding method. For now, let's assume we use the Skip-gram model. In the next section, we describe an enhanced word embedding method. We compute a POS corpus statistics vector $v_i \in \mathbb{R}^{12}$ for the word using the 12 universal POS tags. With this new information, the representation for word i is given by

$$x_i = (w_i, v_i) \in \mathbb{R}^{d+12}\,. \tag{1}$$

3.2 Word Internal Structure

Morphology carries information that is useful for capturing the identity of a word. It represents information such as tense. When doing cross-lingual zero-shot projection of a word in a source language, we wish to translate to words that have the same linguistic properties. For example, the German word *gewinnen* should be translated to the present tense win, not the past tense won which in German is *gewonnen*. We approximate morphology by incorporating sub-word information into the vector representations. There are several ways of doing this, one approach is to work on the level of characters. We go for the middle-ground, in which a word is represented as a combination of a vector for the word itself with vectors of sub-word units that comprise it. In particular, for a given a word we learn a vector representation for the word itself, and also for each n-gram of $>=$ 3 and $<$ 6 in the word (Bojanowski et al., 2017). Each word is thus represented by the sum of the vector representations of its n-grams, including the word itself. This representation is then used to replace w_i in equation 1.

4 Training Objective

A common objective function used in prior work (Mikolov et al., 2013; Dinu et al., 2014) for learning cross vector space mapping functions is the regularized least squares error:

$$\hat{\mathbf{W}} = \underset{\mathbf{W}\in\mathbb{R}^{s\times t}}{\arg\min} ||\mathbf{XW} - \mathbf{Y}||_F + \lambda||\mathbf{W}|| \qquad (2)$$

where matrix $\hat{\mathbf{W}}$ is the learned mapping function, $\mathbf{X}$ and $\mathbf{Y}$ represent the matrices containing the vectors for the source language words and vectors for the target language words, respectively. Instead of the least squares loss shown in equation 2, we use a ranking loss, as in (Lazaridou et al., 2015), which aims to rank correct training data pairs (x_i, y_i) higher than incorrect pairs (x_i, y_j) with a margin of at least γ. The margin γ is a hyper-parameter which is application specific, and the incorrect labels, y_j can be selected randomly such that $j \neq i$ or in a more application specific manner[2].

Given a seed dictionary training data of the form $D^{tr} = \{x_i, y_i\}_{i=1}^{m}$, the margin-based ranking loss is defined as:

$$J_{\text{single}} = \sum_{i=1}^{m}\sum_{j\neq i}^{k} \max\Big(0, \gamma + d(y_i, \hat{y}_i) - d(y_j, \hat{y}_i)\Big) \qquad (3)$$

where $\hat{y}_i = \mathbf{W}x_i$ is the prediction, k is the number of incorrect examples per training instance, and $d(x, y) = (x - y)^2$ is the distance measure.

For a given correct pair and incorrect pair, substituting $\hat{y}_i = \mathbf{W}x_i$. The loss is given by:

$$\max\Big(0, \gamma + (y_i - \hat{y}_i)^2 - (y_j - \hat{y}_i)^2\Big) : j \neq i. \qquad (4)$$

To evaluate the derivative analytically, we can write:

$$\begin{aligned} &\max\left(0, \gamma + (y_i - \hat{y}_i)^2 - (y_j - \hat{y}_i)^2\right) \\ &= \theta\left(\gamma + (y_i - \hat{y}_i)^2 - (y_j - \hat{y}_i)^2\right) \times \\ &\qquad \left[\gamma + (y_i - \hat{y}_i)^2 - (y_j - \hat{y}_i)^2\right] \end{aligned} \qquad (5)$$

where $\theta(x)$ denotes the Heaviside θ-function. The derivative with respect to the elements of the matrix $\mathbf{W}$ is then approximated by, after neglecting a term that would only contribute if the difference $(y_j - \hat{y}_i)^2 - (y_i - \hat{y}_i)^2$ were exactly γ

$$\begin{aligned} &\frac{\partial}{\partial W_{ab}}(\theta\left(\gamma + (y_i - \hat{y}_i)^2 - (y_j - \hat{y}_i)^2\right) \times \\ &\qquad \left[\gamma + (y_i - \hat{y}_i)^2 - (y_j - \hat{y}_i)^2\right] \\ &\simeq 2\theta\left(\gamma + (y_i - \hat{y}_i)^2 - (y_j - \hat{y}_i)^2\right) \times \\ &\qquad x_{i\,b}\,(y_{j\,a} - y_{i\,a}) \end{aligned} \qquad (6)$$

5 Model Distillation

In zero-shot learning for bilingual dictionary induction a large seed dictionary is used to train a mapping function. Such large training dictionaries might not be available for all languages. However, for a given language with only a small seed dictionary, there could be a highly related language with a much larger seed dictionary. We propose a method for leveraging mapping functions of nearby languages to train mapping functions for languages where large seed dictionaries may not be available. Our method is related to notion of having a bridge or pivot language as done in sentence level translation (Utiyama and Isahara, 2007). We develop a distillation training objective that allows us to seamlessly leverage several bridge languages for word level translation.

[2] In our experiments, we explored several application specific approaches for choosing negative examples, including one that picks negative examples among words whose part of speech class is different from the positive example. However, these approaches did produce significant improvement, and we resorted back to randomly selected negative examples.

5.1 Trilingual Paths for distillation

Let us consider the problem of translating from a given source language to English. As a running example, we use Portuguese(pt) as the source language. We wish to learn a mapping function from word vectors in Portuguese to word vectors in English. We can set up a learning task, using a training dataset $D = \{x_i, y_i\}_{i=1}^m$ and the loss defined in Equation 3. This gives us the projection function in the form of a matrix: $\mathbf{W}^{(pt\rightarrow en)}$. We can thus translate Portuguese words to English as follows:

$$\hat{y_i}^{(en)\hookleftarrow(pt)} = \mathbf{W}^{(pt\rightarrow en)} x_i^{(pt)} \tag{7}$$

If the seed dictionary for Portuguese to English is small, $\mathbf{W}^{(pt\rightarrow en)}$ might generalize poorly, producing many wrong translations when using Equation 7. Suppose, a related language, for example, Spanish has a lot of training data available, and we have independently trained its mapping function, which can make predictions from Spanish to English as follows:

$$\hat{y_i}^{(en)\hookleftarrow(es)} = \mathbf{W}^{(es\rightarrow en)} x_i^{(es)} \tag{8}$$

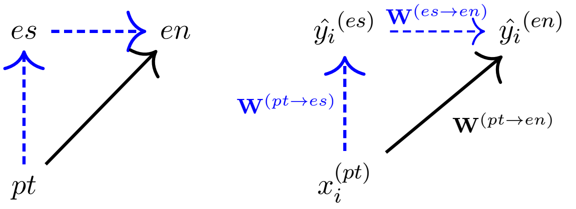
Since $\mathbf{W}^{(es\rightarrow en)}$ is trained with a lot of data, we expect it to generalize better and make more accurate predictions than $\mathbf{W}^{(pt\rightarrow en)}$. One insight here is that since the languages es and pt are highly related, we need much less data to train an accurate mapping matrix $\mathbf{W}^{(pt\rightarrow es)}$ than we to need to learn an accurate $\mathbf{W}^{(pt\rightarrow en)}$. Therefore we train a mapping function from Portuguese to Spanish, which makes predictions as follows.

$$\hat{y_i}^{(es)\hookleftarrow(pt)} = \mathbf{W}^{(pt\rightarrow es)} x_i^{(pt)} \tag{9}$$

We now have a second path that goes from Portuguese to English much like Equation 7 but this path goes via Spanish as follows:

$$\begin{aligned} \hat{y_i}^{(es)\hookleftarrow(pt)} &= \mathbf{W}^{(pt\rightarrow es)} x_i^{(pt)} \\ \hat{y_i}^{(en)\hookleftarrow(es)\hookleftarrow(pt)} &= \mathbf{W}^{(es\rightarrow en)} \hat{y_i}^{(es)\hookleftarrow(pt)} \end{aligned} \tag{10}$$

Figure 2 illustrates the two paths from Portuguese to English. Our main insight is to use knowledge distillation, to improve the accuracy of the mapping matrix $\mathbf{W}^{(pt\rightarrow en)}$ through $\hat{y_i}^{(en)\hookleftarrow(es)\hookleftarrow(pt)}$. This distillation is done by modifying our learning objective.


es
en
pt
$\hat{y_i}^{(es)}$
$\mathbf{W}^{(es\rightarrow en)}$
$\hat{y_i}^{(en)}$
$\mathbf{W}^{(pt\rightarrow es)}$
$\mathbf{W}^{(pt\rightarrow en)}$
$x_i^{(pt)}$

(a)

$$\hat{y_i}^{(es)\hookleftarrow(pt)} = \mathbf{W}^{(pt\rightarrow es)} x_i^{(pt)}$$
$$\hat{y_i}^{(en)\hookleftarrow(es)\hookleftarrow(pt)} = \mathbf{W}^{(es\rightarrow en)} \hat{y_i}^{(es)\hookleftarrow(pt)}$$

(b)

$$\hat{y_i}^{(en)\hookleftarrow(pt)} = \mathbf{W}^{(pt\rightarrow en)} x_i^{(pt)}$$

Figure 2: Translating with both a trilingual path (dotted lines, and equation (a)) , and a bilingual path (solid line, and equation (b))

5.2 Distillation Objective

For a given Portuguese word $x_i^{(pt)}$, Equation 7 makes the prediction $\hat{y_i}^{(en)\hookleftarrow(pt)}$ and Equation 10 makes the trilingual prediction $\hat{y_i}^{(en)\hookleftarrow(es)\hookleftarrow(pt)}$ which involves three languages. We would like to improve predictions made by Equation 7 by improving the mapping matrix $\mathbf{W}^{(en\rightarrow pt)}$. Therefore when training using the Portuguese to English training data, we want our objective to both minimize the loss defined in Equation 3 and simultaneously to let $\mathbf{W}^{(en\rightarrow pt)}$ mimic predictions made through the path $\hat{y_i}^{(en)\hookleftarrow(es)\hookleftarrow(pt)}$ as "soft targets" within a margin. The distillation objective is as follows:

$$\begin{aligned} J_d = \sum_{i=1}^{m} \max \Big(0, \\ \big(\hat{y_i}^{(en)\hookleftarrow(pt)} - \hat{y_i}^{(en)\hookleftarrow(es)\hookleftarrow(pt)} \big)^2 - \phi \Big), \end{aligned} \tag{11}$$

where ϕ is the margin. We combine J_{single} and J_d through a weighted average of the two different objective functions. Notice that J_d can be computed without having labeled training data. In our experiments, we show that even in this case of unlabeled data, which gets rid of J_{single} since it requires labeled data, J_d outperforms models trained using only J_{single} when the training data is small.

5.3 Multiple Trilingual Paths

We are not restricted to distilling Portuguese through Spanish only. Our model can, in addition, for example distill through German, French, and other languages. We can modify the distillation loss as follows:

$$J_{d-multi} = \sum_{i=1}^{m} \sum_{j=1}^{n} \psi_j \max \Big(0, \Big(\hat{y_i}^{(en) \hookleftarrow (pt)} - \hat{y_i}^{(en) \hookleftarrow (j) \hookleftarrow (pt)}\Big)^2 - \phi\Big), \quad (12)$$

where j labels the distillation language. With the objective $J_{d-multi}$ combined with J_{single}, we are training a mapping function which mimics the behavior of many trilingual paths, as “soft targets” within a margin ψ. We keep ϕ the same in our experiments across all trilingual paths. The ψ_i are weights that reflect how much we penalize our model if it diverges from the predictions of a particular trilingual path. Intuitively, if a language is similar to our source language, (pt) in this case, its corresponding ψ value should be high. For example, if Spanish is considered more related to Portuguese than any other language in the trilingual paths in Equation 12, than we expect $\forall i \neq 1, \psi_1 > \phi_i$. This is assuming that the second parts of the trilingual paths have similar accuracies, ie. $\mathbf{W}^{(es \rightarrow en)}$, $\mathbf{W}^{(fr \rightarrow en)}$, and $\mathbf{W}^{(\ldots \rightarrow en)}$ have similar projection accuracies. The most similar language is expected to be the easiest to project into from Portuguese. For example we might expect $\mathbf{W}^{(pt \rightarrow es)}$ to be more accurate than $\mathbf{W}^{(pt \rightarrow de)}$, if we have similar amounts of training data for learning both of these. We next present how we learn the ψ_i values for the multiple paths.

5.4 Weighted Trilingual Paths

Going back to the example, we first learn the weights using the Portuguese to English training set, $D = \{x_i^{pt}, y_i^{en}\}_{i=1}^{m}$, and then input the weights into the model before training with $J_{d-multi}$ and J_{single}. Suppose we want to compute ψ_1 which corresponds to Spanish in Equation 12. For a given Portuguese word $x_i^{pt} \in D$, whose English translation is y_i^{en}, we can compute:

$$\psi_{1i}^{dot} = (y_i^{en})^T \hat{y_i}^{(en) \hookleftarrow (es) \hookleftarrow (pt)} \quad (13)$$

We also experimented with a bilinear term:

$$\psi_{1i}^{bilinear} = (y_i^{en})^T H \hat{y_i}^{(en) \hookleftarrow (es) \hookleftarrow (pt)} \quad (14)$$

We found a better performing approach to be:

$$V = \Big(\hat{y_i}^{(en) \hookleftarrow (pt)} - \hat{y_i}^{(en) \hookleftarrow (es) \hookleftarrow (pt)}\Big)^2$$

$$\psi_{1i}^{euclid} = \exp(\frac{1}{V}). \quad (15)$$

	P@1	**P@5**	**P@10**
	Italian ($en \rightarrow it$)		
THIS	51.0	66.6	72.4
THIS w/pos	**51.6**	**68.5**	**73.4**
Ridge	29.7	44.2	49.1
Lazaridou et. al	40.2	54.2	60.4
MultiCluster	2.40	7.30	11.0
MultiCCA	0	0.1	0.3

Table 1: Translation accuracy on the English to Italian dataset of (Dinu et al., 2014).

6 Experimental Evaluation

In this section, we study the following questions:

- What is the effect of modeling linguistic information in the vector representations of the source and target languages on accuracy of bilingual dictionary induction?

- Can our knowledge distillation objective from trilingual paths involving related languages improve accuracy of mapping functions of languages with small seed dictionaries?

6.1 Data and Experimental Setup

In most of our experiments, we use the training data that was used to train the multi-lingual embeddings in (Ammar et al., 2016). We indicate when this is not the training data used. This data was obtained automatically by using Google Translate. For test data, we use manual translations either from prior work or from searching the Web, including genealogical word lists [3].

For word vector representations, we use Wikipedia to train 300 dimensional vectors for all languages we evaluate on. Based on a validation set, we set the margin γ in Equation 3 through Equation 6 to be $\gamma = 0.4$, ϕ in Equations 11, 12, and 15 to be $\phi = 0.01$. We estimate model parameters using stochastic gradient descent.

6.2 Methods Under Comparison

In our experiments, we compare performance of the following methods.

- The method **THIS** refers to our model which uses a max-margin loss function as defined

[3] For example: https://familysearch.org/wiki/en/Afrikaans_Word_List

in Equation 3. It uses sub-word information in the vector representations of words. One variation of our method is, **THIS w/pos**, which includes POS tag statistics as additional dimensions. The distilled variations of our method explicitly indicate the languages involved, for example $(pt \rightarrow es \rightarrow en)$.

- The method **Ridge** is used in a number of prior work (Mikolov et al., 2013; Dinu et al., 2014; Vulic and Korhonen, 2016). These approaches use an L2-regularized least-squares error objective as shown in Equation 2.
- The method **Lazaridou et al.** was proposed by (Lazaridou et al., 2015). It uses a max-margin ranking function and introduces a way of picking negative examples in computing the loss.
- The methods **MultiCluster** and **MultiCCA** refer to the multilingual word embeddings introduced by (Ammar et al., 2016). They extend canonical correlation analysis (CCA) based methods (Haghighi et al., 2008; Faruqui and Dyer, 2014) to a multi-lingual setting where they treat English as the common vector space. For these methods, we use their pre-trained word embeddings.

6.3 Linguistic Information Evaluation

To address the first of our evaluation questions, we performed experiments on the dataset introduced by (Dinu et al., 2014), where the state-of-the art is the work of (Lazaridou et al., 2015). This is an Italian to English dataset, which consists of $5K$ translation pairs as training data, and $1.5K$ pairs as test data. In both (Dinu et al., 2014) and (Lazaridou et al., 2015), the embeddings were trained on Wikipedia and additional corpora, we only train on Wikipedia.

The results for this experiment are shown in Table 1. Our method, THIS, performs well above the previous state of the art (Lazaridou et al., 2015). For top-1 precision, as can been seen in Table 1, we obtained an 11% gain. From Table 1, we can also see that the POS statistics are only marginally helpful. The word embeddings generated with MultiCluster and MultiCCA perform poorly, with MultiCluster doing better than MultiCCA.

	de ↓ en	es ↓ en	fr ↓ en	it ↓ en	nl ↓ en	sv ↓ en
Train	400k*	400k*	100k*	5k	1,392	110k*
Test	1,180	1,109	810	2,148	296	471

Table 2: Training and test sets for various language pairs. The training datasets marked with (*) are from (Ammar et al., 2016) obtained through Google Translate. Italian to English is from (Dinu et al., 2014). The Dutch to English training dataset is introduced in this paper. With the exception of Italian to English, all test datasets are introduced in this paper.

	de ↓ en	es ↓ en	fr ↓ en	it ↓ en	nl ↓ en	sv ↓ en
	P@10					
THIS	**57.8**	**59.5**	**67.4**	**70.0**	**60.8**	**54.6**
Ridge	32.8	54.2	59.9	66.4	58.8	44.6
MultiCluster	12.2	8.1	4.6	6.9	-	9.0
MultiCCA	6.7	4.3	2.9	5.6	-	10.1

Table 3: Top-10 precision for eight languages translated to English. The high accuracy on Italian can be explained by the fact that, unlike other language pairs, for Italian we do not use Google Translate training data, but the data of (Dinu et al., 2014), as shown in Table 2.

We additionally carried out experiments on 8 other language pairs, further showing our method outperforming prior work. The results are shown in Table 3, and the corresponding data is shown in Table 2. For these language pairs, we do not show results for our method, THIS w/pos, since POS taggers are not available for some of the languages. We also do not show (Lazaridou et al., 2015), as they did not do experiments on these data sets, and we did not have an implementation of their approach. Additionally, (Ammar et al., 2016) did not have trained embeddings for Dutch (nl).

6.4 Trilingual Paths for Distillation

To address our second evaluation question, we carried out experiments with languages for which we only had small seed dictionaries. The training and test datasets for this setup are shown in Table 4. We gathered these datasets by searching for manually created datasets. In the cases were we could not find any, we used Google Translate, which, however produces some noisy translations. This is partly due to the fact that the translations are done out of context.

We begin with thorough experiments on the

	pt ↓ en	pt ↓ es	pt ↓ fr	pt ↓ de	da ↓ en	da ↓ sv	af ↓ en	af ↓ nl
Train	573	701	1,808	465	3,000	1,980	3,744	2,000
Test	296	0	0	0	262	0	459	0

Table 4: Training and test datasets used in the trilingual path distillation experiments. We evaluated sub-parts of trilingual paths such as $pt \rightarrow es$, and $pt \rightarrow fr$ using cross validation hence the test sets for those languages are zero.

		P@10
	Portuguese ($pt \rightarrow en$)	
1	THIS ($pt \rightarrow en$)	65.2
2	($pt \rightarrow en$) +($pt \rightarrow es \rightarrow en$) [unlabeled data]	74.0
3	($pt \rightarrow en$) + ($pt \rightarrow es \rightarrow en$)	**82.1**
4	$(pt \rightarrow en) + (pt \rightarrow \begin{pmatrix} de \\ es \\ fr \end{pmatrix} \rightarrow en)$ [Weighted]	81.8
5	$(pt \rightarrow en) + (pt \rightarrow \begin{pmatrix} de \\ es \\ fr \end{pmatrix} \rightarrow en)$[Unweighted]	78.4
6	Ridge	60.8

Table 5: Trilingual path distillation results for Portuguese to English.

Portuguese-English language pair. The results are shown in Table 5. First, we see that if we distill through the Spanish trilingual path ($pt \rightarrow es \rightarrow en$), without using any labeled data from $pt \rightarrow en$, we already obtain a 9% gain in accuracy, line 2 in Table 5. If, in addition to distilling through Spanish, we use the available training data $pt \rightarrow en$, 573 translation pairs, line 3 in Table 5, we obtain a 17% gain in accuracy. We see however that adding the distillation paths via French, and German did not improve performance, line 4 in Table 5. This can be attributed to the fact that with multiple distillation paths, the model has to optimize a more difficult function. On the other hand, we see that our trilingual weighting mechanism is effective. Without path weights, top-10 accuracy is 78.4% vs 81.8% with weights, lines 4 and 5 in Table 5. The learned weights for the three languages involved in the trilingual paths for Portuguese are shown in Figure 3. Spanish is the highest weighted, followed by French, and German has the lowest weight. By definition, the learned weights add up to 1. In Figure 4, we show accuracy while varying the size of the seed dictionary. We can see that, given the small size of the training data, distillation provides a strong advantage.

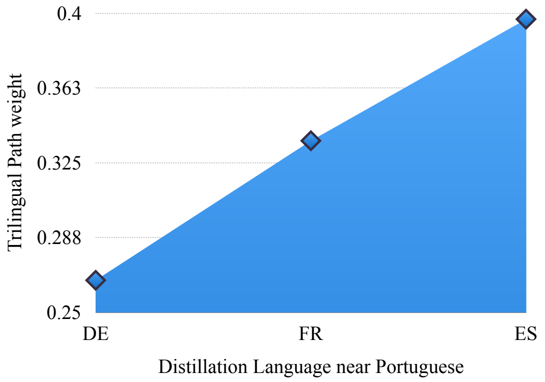


Figure 3: Learned weights for languages involved in trilingual paths for translating Portuguese to English. Spanish is the highest weighted and German is the lowest.

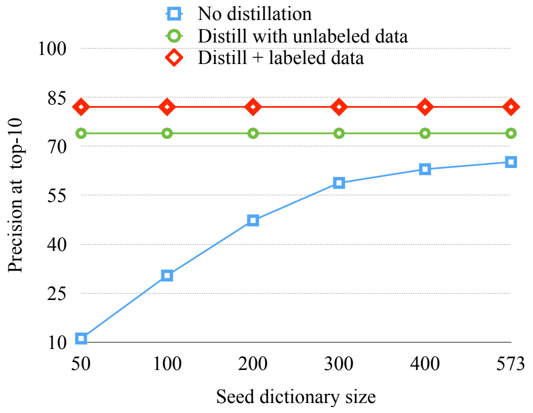


Figure 4: Varying the size of the seed dictionary for ($pt \rightarrow en$).

Finally, we applied our distillation method to Afrikaans and Danish. Afrikaans distills from Dutch, and Danish distills from Swedish. As shown in Table 6, in both cases, we obtained performance gains. However, in both of these cases, performance gains are modest. Unlike Portuguese to English, the seed dictionaries involved in training these language pairs were obtained automatically using Google Translate and contain noisy translations.

7 Conclusion

We have presented a knowledge distillation training objective that leverages trilingual paths of related languages to improve mapping functions of languages with small seed dictionaries. The model produces substantial gains in accuracy for several language pairs.

There are several future directions. First, due

	P@10
Afrikaans $(af \rightarrow en)$	
THIS $(af \rightarrow en)$	46.4
$(af \rightarrow en)$+ $(af \rightarrow nl \rightarrow en)$ [unlabeled data]	49.9
$(af \rightarrow en)$+$(af \rightarrow nl \rightarrow en)$	51.0
Ridge	38.6
Danish $(da \rightarrow en)$	
THIS $(da \rightarrow en)$	44.4
$(da \rightarrow en)$ + $(da \rightarrow sv \rightarrow en)$ [unlabeled data]	45.2
$(da \rightarrow en)$ + $(da \rightarrow sv \rightarrow en)$	47.2
Ridge	37.1

Table 6: Trilingual path distillation results for Afrikaans and Danish.

to advances in methods for extracting general purpose knowledge (Mitchell et al., 2015; Nakashole et al., 2013; Wijaya et al., 2014), the use of semantic knowledge has been explored for several natural language tasks (Nakashole and Mitchell, 2015; Yang and Mitchell, 2017). However, for bilingual dictionary induction, and more generally, machine translation, the role of semantic knowledge has not been fully explored. We consider this to be a promising line of future work. Second, although we focus on bilingual dictionary induction, our knowledge distillation training objective that enables seamless use of paths of rich resource languages as teachers of low resource languages is general and can be applied to problems such as multilingual tagging and parsing.

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. *CoRR*, abs/1602.01925.

Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. 2007. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137.

Phil Blunsom and Karl Moritz Hermann. 2014. Multilingual distributed representations without word alignment.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL.*

Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. *ACL 2007*, page 256.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2014. Improving zero-shot learning by mitigating the hubness problem. *arXiv preprint arXiv:1412.6568.*

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *EACL*, pages 462–471.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *IICML*, pages 748–756.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *ACL*, pages 1234–1244.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *ACL*, pages 771–779.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531.*

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *COLING*, pages 1459–1474.

Tomáš Kočiskỳ, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. *arXiv preprint arXiv:1405.0947.*

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *ACL*, pages 270–280.

Mingsheng Long and Jianmin Wang. 2015. Learning multiple tasks with deep relationship networks. *arXiv preprint arXiv:1506.02117.*

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

Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil Antonios Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Tanti Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. 2015. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, (AAAI)*, pages 2302–2310.

Tom M Mitchell, Svetlana V Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L Malave, Robert A Mason, and Marcel Adam Just. 2008. Predicting human brain activity associated with the meanings of nouns. *science*, 320(5880):1191–1195.

Ndapandula Nakashole and Tom M. Mitchell. 2015. A knowledge-intensive model for prepositional phrase attachment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, (ACL)*, pages 365–375.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2013. Discovering semantic relations from the web and organizing them with patty. *ACM SIGMOD Record*, 42(2):29–34.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760. ACM.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany. Association for Computational Linguistics.

Oscar Täckström, Ryan T. McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *NAACL*, pages 477–487.

Masao Utiyama and Hitoshi Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *HLT-NAACL*, pages 484–491.

Ivan Vulic and Anna Korhonen. 2016. On the role of seed lexicons in learning bilingual word embeddings. ACL.

Ivan Vulic and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *ACL*, pages 719–725.

Derry Tanti Wijaya, Ndapandula Nakashole, and Tom M Mitchell. 2014. Ctps: Contextual temporal profiles for time scoping facts using state change detection. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Bishan Yang and Tom M. Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, (ACL)*.

Machine Translation, it's a question of style, innit?
The case of English tag questions

Rachel Bawden
LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, F-91405 Orsay, France
rachel.bawden@limsi.fr

Abstract

In this paper, we address the problem of generating English tag questions (TQs) (e.g. *it is, **isn't it?***) in Machine Translation (MT). We propose a post-edition solution, formulating the problem as a multi-class classification task. We present (i) the automatic annotation of English TQs in a parallel corpus of subtitles and (ii) an approach using a series of classifiers to predict TQ forms, which we use to post-edit state-of-the-art MT outputs. Our method provides significant improvements in English TQ translation when translating from Czech, French and German, in turn improving the fluidity, naturalness, grammatical correctness and pragmatic coherence of MT output.

1 Introduction

When it comes to the machine translation (MT) of discourse, revisiting the question of what constitutes a high quality translation is essential; which aspects of language should be tackled and how to evaluate them. A first step is to identify the many stylistic aspects of speech that pose a problem for current MT techniques and to study how they could be taken into account and evaluated.

We take a step in this direction by addressing a new aspect of discourse in MT, related to speaker attitude and style: the English tag question (hereafter TQ), i.e. utterances such as *catchy, **ain't it?*** and *it wasn't him, **was it?***. When translating into English, TQs present two main challenges. The first is knowing when to generate one. Similar to the translation of discourse connectives, TQ use is a question of style and speaker attitude, and importantly, there is not often a direct, lexical correspondence across languages. TQs are common in English and far less so in other languages, which means that other contextual cues are necessary to determine whether a TQ should appear in the English translation. The second, in particular for canonical TQs (*e.g. was it?, isn't it?*), is that the overall grammaticality of the utterance is determined by the correct choice of tag, which, in a similar way to anaphor translation, is grammatically dependent on the rest of the MT output.

Our aim in this paper is to improve the generation of English TQs in an MT setting with English as the target language. We formulate this as a multi-class classification task, using features from both source sentences and machine translated outputs. The prediction of the appropriate question tag to use in the English translation (if any) is then used to post-edit MT outputs. Our results, when translating from Czech (CS), French (FR) and German (DE) into English (EN), display significant improvements, as shown by automatic and manual evaluations (Sec. 5).[1]

2 English tag questions (TQs)

TQs are interrogative constructions, common in spoken English, formed of a main clause (typically declarative), followed by a peripheral interrogative element, the *question tag*:

(1) *You do believe in happy endings*, **don't you?**

(2) *He can't do that*, **can he?**

In its canonical form, the English question tag (in bold) is formed of an auxiliary verb, which can be negated, followed by a pronoun. It parallels the verb and subject (underlined) of the preceding host clause (in italics). The grammatical structure of TQs and agreement between the host and the tag gives them the name of *grammatical TQs*.[2]

[1] All scripts and annotations are freely available at http://diamt.limsi.fr.

[2] Although in theory their form is relatively systematic, their attested usage is more complex.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2507–2512
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

There exists a second type of TQ, *lexical TQs*, formed of a word or phrase that is invariant to the subject and verb of the host clause. For example:

(7) *He's a proper bad man*, **innit?**

(8) *There's got to be a cure*, **right?**

TQs' functions are complex and communicate information about speaker attitude, tone, the relationship between dialogue participants, common ground and dialogue flow (McGregor, 1995). Yet few languages have such a systematic use of TQs, in particular of grammatical TQs, as English (Axelsson, 2011). When translating into English, the first difficulty is appropriately generating English TQs from a source sentence that does not have a TQ; the complex and often ambiguous functions of TQs (e.g. expressing doubt or surprise) can be expressed differently, and subtly, in the source language. The second difficulty is ensuring grammatical coherence of canonical TQs. Consider the German sentence *Sie lebt noch, nicht wahr?* and its English translation *She's alive, isn't she?*. The choice of the question tag *isn't she?* is dependent on the subject and verb of the anchor clause. Had the translation been *She still lives*, the correct question tag would have been *doesn't she?*.

3 Related work

Discourse is a growing field in MT (Le Nagard and Koehn, 2010; Hardmeier, 2012, 2014). To our knowledge, there has been no previous work on TQs in MT, but the two main challenges described above are similar to those associated with two previously studied discursive aspects: the translation of discourse connectives and of anaphoric pronouns. As with TQs, their frequency is relatively low, but their mistranslation has a high impact on coherence, naturalness and therefore human understanding of translations (Meyer and Popescu-Belis, 2012).

Discourse connectives (e.g. *since*, *because*) indicate the relation between discursive units and are linked to the overall coherence of a text in a similar way to TQs. They often have no direct mapping when translated (Meyer and Webber, 2013) and it is often necessary to generate a discourse connective or a TQ where one is not present in the source sentence. Previous work by Meyer and Webber (2013) consists in the disambiguation of discourse connectives in source sentences prior to translation, using automatic sense classification, which guides the MT system's choice of how a discourse connective should be translated (if at all). However they do not handle the case of generating discourse relations from a source sentences in which they do not appear lexically, as is our aim for TQs.

The difficulty of anaphoric pronoun translation is ensuring grammatical agreement between a pronoun and its coreferent, when the information relevant to grammatical agreement in the target language is not present in the source language. For example, the French translation of the pronoun *it* in *I hear an owl but I can't see it* is translated as *le* or *la*, depending on whether *owl* is translated as masc. *hibou* or fem. *chouette*. The position of the coreferent, as with the subject and verb for TQs, is not pre-determined, and identifying which words the translated pronoun or TQ must agree with is not always easy. The majority of works perform classification of pronominal forms in view to post-editing MT output (Guillou, 2016), encouraged by the shared task on cross-lingual pronoun prediction at DiscoMT15 (Hardmeier et al., 2015) and WMT16 (Guillou et al., 2016). We use the same strategy here, since the coherent use of TQs, like anaphors, is dependent on the MT translation.

4 Our post-edition approach

Our method to improve the generation of English TQs in MT is the automatic post-edition of state-of-the-art MT outputs. We formulate the problem as a supervised, multi-class classification task, exploiting lexical features from source sentences and their machine translations (Sec. 4.2). Possible labels are the different question tag forms (e.g. *isn't it*, *ok*). Predicted tags are either used to replace the question tag already present in the MT output or appended to it otherwise. We test our method for three source languages (CS, DE and FR) into EN.

4.1 Corpus annotation for English TQs

The first step is to produce annotated data. The corpora used cover three language pairs: CS-EN, DE-EN and FR-EN, and are large subsets of the most recent films of the OpenSubtitles[3] parallel corpus (Lison and Tiedemann, 2016). The subtitles were automatically cleaned using heuristics and processed with the MElt tokeniser (Denis and Sagot, 2012) and the Moses truecaser (Koehn et al., 2007). We then developed robust, manually defined lexical rules to identify English TQs. We

[3] www.opensubtitles.org

identify both the presence of TQs and the question tags themselves (e.g. *is it?*, *right?*).

A manual evaluation on a random subset of 500 grammatical TQs and 500 lexical TQs shows that annotations are near perfect (accuracy of ≈98% and recall of 100% on sentence-final grammatical TQs whose host clause is in the same subtitle).[4]

Each corpus was divided into three sets: TRAIN ($\frac{2}{3}$), DEV ($\frac{1}{6}$) and TEST ($\frac{1}{6}$). The distribution of TQs is shown in Tab. 1. TQs make up approximately 1% of subtitles, but are common among questions (≈20%). There are between 238 and 285 distinct English question tags, depending on the language pair (including a label *none* for non-TQs). The most frequent question tag is *right?* (≈20%), followed by *ok?* (≈16%). The majority of labels are grammatical question tags, but the most frequent (*isn't it?*) represents only ≈3% of all TQs, revealing a huge class imbalance.

		#English TQs			#labels	
	#sents	all	gram.	lex.	gram.	lex.
CS-EN	15.1M	146,782	44,572	102,210	269	15
DE-EN	6.2M	57,435	18,396	39,039	221	16
FR-EN	15.1M	149,847	44,651	105,196	254	16

Table 1: TQ distribution for each language pair.

4.2 Question tag classification

Given the class imbalance (Sec. 4.1) and the different nature of lexical and grammatical TQs, we hypothesise that first predicting the presence of an English TQ before selecting tag forms is preferable to directly predicting tags in a single, direct pass. We compare a single statistical classifier (hereafter CL-ONE), which directly predicts the question tag (including the label *none*), to a more complex system using a sequence of classifiers (hereafter CL-SEQ, see Fig. 1). In CL-SEQ, a first classifier (CL-SEQ$_{\text{coarse}}$) predicts a coarse-grained label *gram* (grammatical TQ), *lex* (lexical TQ) or *none* (non-TQ), which determines which classifier (CL-SEQ$_{\text{lex}}$ or CL-SEQ$_{\text{gram}}$) is used to predict the tag form. Both CL-SEQ$_{\text{coarse}}$ and CL-SEQ$_{\text{lex}}$ are statistical classifiers. CL-SEQ$_{\text{lex}}$ is trained on the TRAIN examples assigned the label *lex* by CL-SEQ$_{\text{coarse}}$, which explains why it provides a second chance to predict grammatical or non-TQs. CL-SEQ$_{\text{gram}}$ is a rule-based system, a choice that is better adapted to the sparse labels of grammatical TQs. Where there is no rule available, this system too can predict the label *none*.

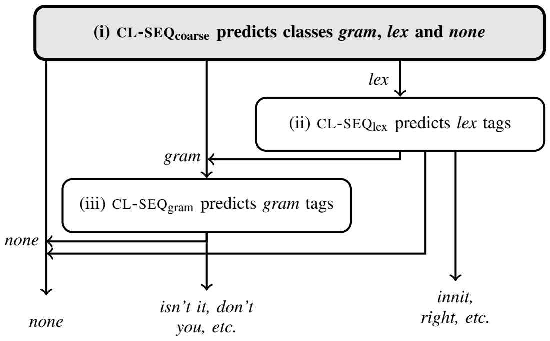

(i) CL-SEQ$_{\text{coarse}}$ predicts classes *gram*, *lex* and *none*
lex
(ii) CL-SEQ$_{\text{lex}}$ predicts *lex* tags
gram
(iii) CL-SEQ$_{\text{gram}}$ predicts *gram* tags
none
none
isn't it, don't you, etc.
innit, right, etc.

Figure 1: CL-SEQ classification: (i) into coarse-grained classes, (ii–iii) prediction of forms

Experimental setup All statistical classifiers are linear classifiers trained using Vowpal Wabbit (Langford et al., 2009).[5,6] To account for class imbalance, examples are weighted according to their relative frequency in the TRAIN set, and the degree of weighting is optimised on the DEV set.[7] Features used for all statistical classifiers are described just below. The rule-based CL-SEQ$_{\text{gram}}$ system relies on the MT output alone.

Features We use automatically and manually defined lexical feature templates. Unless indicated the features apply to both the source sentence and the MT output. The first set of features are automatically identified bag-of-word features, which represent the 500 uni-, bi- and tri-grams most associated with a TQ, as measured by a G^2 test. The second set of features are manually defined, based on language-specific question-response patterns and recognisable lexical clues. They include (i) the presence of a question tag (and its form), (ii) the presence of a final question mark, (iii) (CS and DE only) whether the following subtitle contains a verb that appears in the current subtitle (and if so, we include as a feature the verb type and the preceding word in both the current and following subtitles)[8], (iv) the following subtitle contains a specific response (from a predefined list of replies

[4] Recall for lexical TQs cannot be accurately measured, as identification relies on a closed list of forms found in the literature and observed in the data.

[5] http://hunch.net/~vw/

[6] We use "OAA", FTRL-proximal optimisation, L2 regularisation ($\lambda = 1e - 6$) and quadratic features.

[7] We vary weights from equal for all examples to weights that fully counterbalance the class distribution.

[8] In German and Czech, it is common for a reply to a yes/no question to repeat the verb of the question, e.g. *Poslala jsi mu to?* 'Did you send it to him?' — *Poslala jsem* 'Yes, I did' (lit. 'send (I)_did') (Gruet-Skrabalova, 2013).

		Gram TQs				Lex TQs				Non-TQs			Overall
Lang. pair		P	R	F	P*	P	R	F	P*	P	R	F	P*
CS→EN	baseline	52.22	43.83	47.66	35.75	49.76	57.51	53.35	45.35	99.69	99.63	99.66	99.11
	CL-ONE	66.15	15.57	25.20	50.09	55.19	40.33	46.60	51.20	99.43	99.84	99.63	99.16
	CL-SEQ	56.87	44.96	50.22	38.85	60.76	46.68	52.79	56.58	99.57	99.79	99.68	99.21
DE→EN	baseline	45.72	28.68	35.25	9.97	69.07	45.16	54.61	66.59	99.51	99.83	99.67	99.21
	CL-ONE	69.76	9.42	16.59	48.54	61.63	45.49	52.34	59.78	99.46	99.88	99.67	99.26
	CL-SEQ	59.27	42.74	49.67	35.48	68.70	53.21	59.97	66.69	99.62	99.84	99.73	99.32
FR→EN	baseline	41.15	47.18	43.96	12.95	57.63	38.25	46.18	52.55	99.53	99.72	99.62	99.03
	CL-ONE	66.30	9.36	16.41	44.87	55.05	28.80	37.81	51.73	99.32	99.89	99.60	99.12
	CL-SEQ	58.48	33.95	42.96	38.22	63.02	38.22	47.59	59.42	99.46	99.85	99.65	99.19

Table 2: Precision (P), Recall (R), F-score (F) and fine-grained labelling precision (P*) for the TEST set on each language pair. Results are given for each coarse-grained TQ class (*gram*, *lex* and *non-TQ*). Labelling precision is calculated on the subtitles with the corresponding predicted coarse-grained label. Marked in grey are the cells containing the best F-scores for coarse-grained label groupings and the overall labelling precision (for fine-grained classes).

such as *OK*, *yes*, *no*, etc.), and (v) the first words of the MT output (1-4 gram), the last auxiliary, the last pronoun and the last pronoun-auxiliary pair.

Rule-based grammatical TQ prediction Our rule-based approach is designed to predict which grammatical tag should be appended to a given translation. The rules consist in the identification of certain lexical cues from the translation. For instance, utterance-initial words can be a good indicator of the use of a particular question tag: imperatives such as *let's ...* (indicative of the tag *shall we*), and claims about the interlocutor's perception such as *you think...* or *you know...* (indicative of the tag *don't you*). When there is a single auxiliary and subject, these are directly used to construct a question tag, using, as a simplification, the opposite polarity to that of the anchor clause, which is the most common polarity pattern in TQs. We include several rules to account for complex clauses and perform grammatical checking between the subject and auxiliary of the question tag. The complete set of rules is available at the address cited in Footnote 1.

"Baseline" MT outputs For Czech and German, we use the top systems at WMT16, both attentional encoder-decoder NMT models (Sennrich et al., 2016). For French, we trained a phrase-based model with Moses (Koehn et al., 2007).[9] Baseline predictions are automatically extracted from the MT outputs using our English TQ identification rules (Sec. 4.1).

5 Results and analysis

As mentioned by Hardmeier (2012), evaluating coherence-related MT phenomena is problematic. A question tag can be the correct choice without matching the question tag form in the reference translation (Sec. 2), making traditional metrics involving lexical comparison (including all standard MT evaluation metrics) ill-adapted to the task.

Despite this, we provide in Tab. 2 results using traditional metrics. To get a better view of the scores, we group final predicted question tags into their coarse-grained classes (*gram*, *lex*, *none*) and calculate the precision (P), recall (R) and F-score (F) for these three classes. Within each coarse-grained class, we also provide labelling precision (P*), corresponding to the number of question tags within that coarse-grained class assigned the correct question tag form according to the gold label. Labelling precision is also given overall for all test sentences (in the final column).

Overall labelling precision is significantly improved for all language pairs with both classification systems, but in particular for CL-SEQ. This is partly due to a better prediction of non-TQs, represented by the high corresponding F-scores for CL-SEQ for all three language pairs. However it is also linked to a better labelling of grammatical and lexical TQs, which can be seen by the high labelling precision (P*) in the context of high recall (R). The higher scores of CL-SEQ over CL-ONE, particularly in terms of recall, which are most likely a result of question tag label sparsity, show that our two-tier strategy of predicting grammatical and lexical tags separately is better adapted than a single classifier.

There is a notable drop in recall between CL-

[9] We use a combination of three phrase tables and three 4-gram KenLM language models (Heafield et al., 2013), trained on Europarl, Ted Talks and 3M-sentence subtitles, tuned using `kbmira` on a disjoint 2.5K-sentence subset.

Source	Reference	Baseline	CL-SEQ	Judgement
Du hast das gemacht, nicht wahr?	You did this , **didn't you?**	You've done that, **don't you?**	haven't you?	Improved
Das Gehirn zerstören, wisst ihr?	Kill the brain, **you know?**	Destroying the brain, **you know?**	none	Degraded
Das stimmt, oder?	I know, **right?**	That's true, **right?**	isn't it	Equal (correct)
Das ist merkwürdig, oder?	It's weird, **isn't it?**	That 's odd, **does it?**	none	Equal (incorrect)

Table 3: Some examples from the manual comparison of the baseline translation and CL-SEQ predictions. An example is given for each of the possibilities: the prediction is better, worse or the translation and the prediction are equally good or poor with respect to tag question prediction.

SEQ and CL-SEQ when it comes to grammatical TQ prediction, which is not as marked for lexical TQ prediction. This is most likely due to the huge class imbalance in the different question tags (221 grammatical tags vs. 16 lexical tags), which causes the purely statistical one-pass system to favour the more frequent lexical tags and struggle to predict the wide range of much rarer grammatical tags.

		baseline			
		gram	lex	none	Total
gold	gram	871	180	1986	3037
	lex	429	2878	3066	6373
	none	605	1109	1019408	1021122

		predicted			
		gram	lex	none	Total
gold	gram	1298	367	1372	3037
	lex	418	3391	2564	6373
	none	474	1178	1019470	1021122

Table 4: Confusion matrix of for baseline and predicted versus gold tags (when question tags are grouped into their three coarse-grained classes) for DE→EN.

Tab. 4 shows a comparison of baseline and predicted tags for the DE→EN test set. For ease of illustration, the question tags are again regrouped into their three coarse-grained classes (*gram*, *lex* and *none*). The matrix reveals that for predicted lexical tags and non-TQs, the majority were correctly classed into these coarse-grained classes. However, grammatical tags proved more difficult to predict, the majority being classed as non-TQs, most likely a result of the fact that no such tag question was present on the German source side. The most common errors on this test set were predicting a non-TQ when a lexical tag was expected. For example, CL-SEQ predicted *none* 1112 times when the gold tag *right?* was expected. However the number of correct predictions of *right* exceeded this number (1371 cases) and compared to baseline predictions, all coarse-grained categories show an improvement in recall.

Manual analysis Given the drawbacks of automatic metrics, we manually evaluated a set of final translations post-edited with the predictions produced by the CL-SEQ system for DE→EN. We randomly selected 100 examples for which the baseline translation was modified and, comparing the baseline and CL-SEQ prediction, we labelled the example as *improved* (baseline incorrect and prediction correct) or *degraded* (baseline correct and prediction incorrect) or *equal* (both baseline and prediction (in)correct). Examples of these choices are provided in Tab. 3. We found that post-edition made an improved choice of tag in 59/100 examples (compared to 13 worse choices). Only 25 of the 59 improved examples exactly matched the reference tag, confirming the problem of relying solely on traditional metrics. We notice in particular an improvement in the choice of grammatical TQs (33 out of the 59 improvements).

6 Discussion and perspectives

Improvements in the generation of English TQs in MT outputs, as seen by our manual analysis, result in improved grammatical coherence, particularly for grammatical TQs. However, TQ translation is far from solved. As a stylistic aspect, its prediction and evaluation are complex and should be further explored. Possible improvements include improving the choice of linguistic information used and using this work to explore how TQs' functions are portrayed in languages other than English.

Interesting future work would be to compare the opposite approach to the task; augmenting source sentences with disambiguating information prior to translation, particularly within an NMT framework, which has good potential for handling non-local context and integrating extra features.

References

Karin Axelsson. 2011. A cross-linguistic study of grammatically-dependent question tags. *Studies in Language*, 35(4):793–851.

Pascal Denis and Benoît Sagot. 2012. Coupling an annotated corpus and a lexicon for state-of-the-art POS tagging. *Language Resources and Evaluation*, 46(4):721–736.

Hana Gruet-Skrabalova. 2013. Verbs and particles in minimal answers to yes-no questions in Czech. In *Formal Description of Slavic Languages 10*, Slavic Grammar from a Formal Perspective, the 10th anniversary FDSL conference, pages 197–215, Leipzig, Germany.

Liane Guillou. 2016. *Incorporating Pronoun Function into Statistical Machine Translation*. Ph.D. thesis, School of Informatics. University of Edinburgh.

Liane Guillou, Christian Hardmeier, Preslav Nakov, Sara Stymne, Jörg Tiedemann, Yannick Versley, Mauro Cettolo, Bonnie Webber, and Andrei Popescu-Belis. 2016. Findings of the 2016 WMT Shared Task on Cross-lingual Pronoun Prediction. In *Proceedings of the 1st Conference on Machine Translation*, WMT'16, pages 525–542, Berlin, Germany.

Christian Hardmeier. 2012. Discourse in statistical machine translation. a survey and a case study. *Discours [online]*, 11.

Christian Hardmeier. 2014. *Discourse in Statistical Machine Translation*. Ph.D. thesis, Uppsala University, Department of Linguistics and Philology, Uppsala, Sweden.

Christian Hardmeier, Preslav Nakov, Sara Stymne, Jörg Tiedemann, Yannick Versley, and Mauro Cettolo. 2015. Pronoun-focused MT and cross-lingual pronoun prediction: Findings of the 2015 DiscoMT shared task on pronoun translation. In *Proceedings of the 2nd Workshop on Discourse in Machine Translation*, DiscoMT'15, pages 1–16, Lisbon, Portugal.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable Modified Kneser-Ney Language Model Estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, pages 690–696, Sofia, Bulgaria.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, ACL'07, pages 177–180, Prague, Czech Republic.

John Langford, Lihong Li, and Tong Zhang. 2009. Sparse Online Learning via Truncated Gradient. *The Journal of Machine Learning Research*, pages 777–801.

Ronan Le Nagard and Philipp Koehn. 2010. Aiding pronoun translation with co-reference resolution. In *Proceedings of the 5th Workshop on Statistical Machine Translation*, WMT'10, pages 252–261, Uppsala, Sweden.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles. In *Proceedings of the 10th Language Resources and Evaluation Conference*, LREC'16, pages 923–929, Portorož, Slovenia.

William McGregor. 1995. The English 'tag question': A new analysis, is(n't) it? *On Subject and theme: A discourse functional perspective*, 118:91–121.

Thomas Meyer and Andrei Popescu-Belis. 2012. Machine Translation of Labeled Discourse Connectives. In *Proceedings of the 10th Biennial Conference of the Association for Machine Translation in the Americas*, pages 129–138, San Diego, California, USA.

Thomas Meyer and Bonnie Webber. 2013. Implicitation of Discourse Connectives in (Machine) Translation. In *Proceedings of the 1st Workshop on Discourse in Machine Translation*, DISCOMT '13, pages 19–26, Sofia, Bulgaria.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh Neural Machine Translation Systems for WMT 16. In *Proceedings of the 1st Conference on Machine Translation*, WMT '16, pages 368–373, Berlin, Germany.

Deciphering Related Languages

Nima Pourdamghani, Kevin Knight
Information Sciences Institute & Department of Computer Science
University of Southern California
{damghani,knight}@isi.edu

Abstract

We present a method for translating texts between close language pairs. The method does not require parallel data, and it does not require the languages to be written in the same script. We show results for six language pairs: Afrikaans/Dutch, Bosnian/Serbian, Danish/Swedish, Macedonian/Bulgarian, Malaysian/Indonesian, and Polish/Belorussian. We report BLEU scores showing our method to outperform others that do not use parallel data.

1 Introduction

Statistical Natural Language Processing (NLP) tools often need large amounts of training data in order to achieve good performance. This limits the use of current NLP tools to a few resource-rich languages. Assume an incident happens in an area with a low-resource language, known as the Incident Language (IL). For a quick response, we need to build NLP tools with available data, as finding or annotating new data is expensive and time consuming. For many languages this means that we only have a small amount of often out-of-domain parallel data (e.g. a Bible or Ubuntu manual), some monolingual data and almost no annotation such as part of speech tags.

Fortunately, many low-resource languages have one or more higher-resource, closely Related Languages (RL). Examples of such IL/RL pairs are Afrikaans/Dutch and Bosnian/Serbian. A natural idea is to use RL resources to improve the task for IL. But this requires some kind of conversion between RL and IL. Assume the required NLP capability is named entity tagging. If we can convert RL to IL, we can convert all RL training data along with annotations into IL and train the tagger for IL. Or, if we can convert IL to RL we can use the potentially existing RL named entity tagger on converted IL data and project back the tags.

Following this idea, Currey et al. (2016) use a rule-based translation system to convert Italian and Portuguese into Spanish, to improve Spanish (here, IL) language modeling, Nakov and Ng (2009) convert RL/English parallel data to IL/English where both RL and IL have Latin orthography to improve IL/English machine translation. Hana et al. (2006) use cognates to adapt Spanish resources to Brazilian Portuguese to train a part-of-speech tagger. Mann and Yarowsky (2001) use Spanish/Portuguese cognates to convert an English/Spanish lexicon to English/Portuguese. These works prove the usefulness of RL data to improve NLP for IL, but they are designed for specific tasks and IL/RL pairs.

In this paper we propose a universal method for translating texts between closely related languages. We assume that IL and RL are mostly cognates, having roughly the same word order. Our method is orthography-agnostic for alphabetic systems, and crucially, it does not need any parallel data. From now on, we talk about converting RL to IL, but the method does not distinguish between RL and IL; as mentioned above, each direction of translation can have its own potential uses.

To translate RL to IL, we train a character-based cipher model and connect it to a word-based language model. The cipher model is trained in a noisy channel model where a character language model produces IL characters and the model converts them to RL. Expectation Maximization is used to train the model parameters to maximize the likelihood of a set of RL monolingual data. At decoding time, the cipher model reads the RL text character by character in which words are separated by a special character, and produces a weighted lattice of characters representing all the possible translations for each of the input tokens.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2513–2518
Copenhagen, Denmark, September 7–11, 2017.

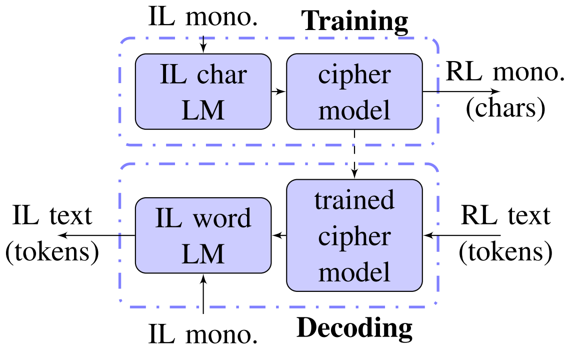


Figure 1: The process used for training the cipher model and decoding RL text to IL

The word-based language model takes this lattice and produces a sequence of output words that maximize the language model score times the cipher model score. Figure 1 depicts this process.

Our cipher models one-to-one, one-to-two and two-to-one character mappings. This allows us to handle cases like Cyrillic 'ч' and Latin 'ch', and also subtle differences in pronunciation between RL and IL like Portuguese 'justiça' and Spanish 'justicia'. Using a character-based cipher model provides the flexibility to generate unseen words. In other words, the vocabulary is limited by the decoding LM, not the cipher model. Separation of training and decoding language models enables us to train the decoding LM on as much data as is available without worrying about training speed or memory issues. We can also transliterate out of vocabulary words by spelling out the best path produced by cipher model in case no good match is found for a token in the decoding LM.

2 Related Work

Previous work on translation between related languages can be categorized into three groups:

Systems for specific language pairs such as Czech-Slovak (Haji et al., 2000), Turkish-Crimean Tatar (Cicekli, 2002), Irish-Scottish Gaelic (Scannell, 2006), and Indonesian-Malaysian (Larasati and Kubo, 2010). Another similar trend is translation between dialects of the same language like Arabic dialects to standard Arabic (Hitham et al., 2008; Sawaf, 2010; Salloum and Habash, 2010). Also, work has been done on translating back the Romanized version of languages like Greeklish to Greek (Chalamandaris et al., 2006) and Arabizi to Arabic (May et al., 2014). These methods cannot be applied to our problem because time and resources are limited to build a translation system for the specific language pair.

Machine learning systems that use parallel data: These methods cover a broader range of languages but require parallel text between related languages. They include character-level machine translation (Vilar et al., 2007; Tiedemann, 2009) or combination of word-level and character-level machine translation (Nakov and Tiedemann, 2012) between related languages.

Use of non-parallel data: Cognates can be extracted from monolingual data and used as a parallel lexicon (Hana et al., 2006; Mann and Yarowsky, 2001; Kondrak et al., 2003). However, our task is whole-text transformation, not just cognate extraction.

Unsupervised deciphering methods, which require no parallel data, have been used for bilingual lexicon extraction and machine translation. Word-based deciphering systems ignore sub-word similarities between related languages (Koehn and Knight, 2002; Ravi and Knight, 2011b; Nuhn et al., 2012; Dou and Knight, 2012; Ravi, 2013). Haghighi et al. (2008) and Naim and Gildea (2015) propose models that can use orthographic similarities. However, the model proposed by (Naim and Gildea, 2015) is only capable of producing a parallel lexicon and not translation. Furthermore, both systems require the languages to have the same orthography and their vocabulary is limited to what they see during training.

Character-based decipherment is the model we use for solving this problem. Character-based decipherment has been previously applied to problems like solving letter substitution ciphers (Knight et al., 2006; Ravi and Knight, 2011a) or transliterating Japanese katakana into English (Ravi and Knight, 2009), but not for translating full texts between related languages.

3 Translating RL to IL

We learn a character-based cipher model for translating RL to IL. At decoding, this model is combined with a word based IL language model to produce IL text from RL.

3.1 Cipher Model

Our noisy-channel cipher model converts a sequence of IL characters $s_1, ..., s_n$ to a sequence of RL characters $t_1, ..., t_m$. It is a WFST composed of three components (Figure 2):

WFST1 is a one-to-one letter substitution model. For each IL character s it writes one RL character t with probability $p_1(t|s)$.

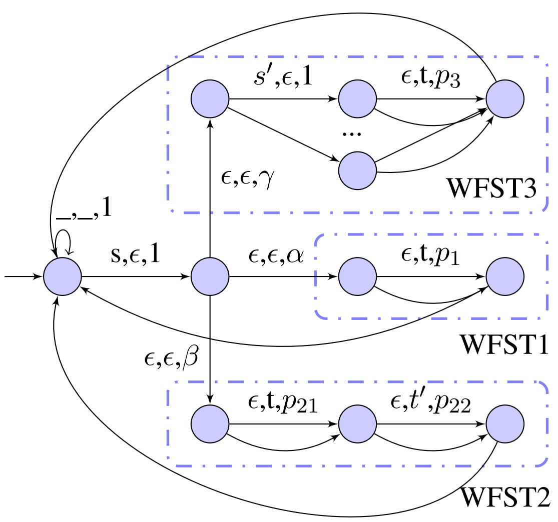


Figure 2: Part of the cipher model corresponding to reading IL character s from start state. The same pattern repeats for any IL character. After reading s, the model goes to WFST1, WFST2, or WFST3 with respective probability $\alpha(s)$, $\beta(s)$, or $\gamma(s)$. In WFST1, the model produces each RL character t with probability $p_1(t|s)$. In WFST2, the model produces each two RL characters t and t' with probability $p_{21}(t|s)$ and $p_{22}(t'|s)$. In WFST3, the model reads each IL character s' and produces each RL character t with probability $p_3(t|ss')$. From the last state of WFST1, WFST2, and WFST3, the model returns to the start state without reading or writing.The model has a loop on start state that reads and writes space.

WFST2 is a one-to-two letter substitution model. For each IL character s, it writes two RL characters t and t' with respective probabilities $p_{21}(t|s)$ and $p_{22}(t'|s)$.

We assume $p_{22}(t'|s)$ is independent of t. As a result we can estimate $p(tt'|s) = p(t|s)p(t'|t, s) \simeq p_{21}(t|s)p_{22}(t'|s)$ as modeled in WFST2. This simplification is required to make the model practicable. Otherwise, the size of the cipher model would become cubic in the number of RL and IL characters, and combining it with a language model would make the system unfeasibly large for training.

WFST3 is a two-to-one letter substitution cipher. For each IL character s, it reads another IL character s' with probability 1, and then writes one RL character t with probability $p_3(t|ss')$. As we will discuss in Section 3.2 we train p_3 directly from p_{21} and p_{22}, hence the cubic number of parameters does not cause a problem.

The start state reads each IL character s and goes to WFST1, WFST2, or WFST3 with respective probability $\alpha(s)$, $\beta(s)$, or $\gamma(s)$. The last state of each component returns to start without reading or writing anything. The start state also reads and writes space with probability one.

3.2 Training the Model

The cipher model described in Section 3.1 is much more flexible than a one-to-one letter substitution cipher. A few thousand sentences of RL monolingual data is not enough to train the model as a whole, and more training data makes the process too slow to be practical. Hence, we break the full model into WFST1, WFST2, and WFST3 and train the parameters of each component, i.e. p_1, p_{21} and p_{22}, and p_3 in separate steps. A final step trains the probability of moving into each of the components, i.e. α, β, and γ.

Each step of the training uses EM algorithm to maximize the likelihood of 500 sentences of RL text in a noisy channel model where a fixed 5-gram character based IL language model (trained on 5000 IL sentences) produces an IL text character by character and the cipher model converts RL characters into RL (top section of Figure 1).

Step one: We set $\alpha(s) = 1$ and $\beta(s) = \gamma(s) = 0$ and train $p_{1IL\to RL}(t|s)$ for each IL character s and each RL character t. In parallel we reverse RL and IL and train $p_{1RL\to IL}(s|t)$ for each RL character t and each IL character s. We use $p_1(t|s) = \frac{1}{2}(p_{1IL\to RL}(t|s) + p_{1RL\to IL}(s|t))$ to set WFST1 parameters in the next steps.

Step two: We set $\alpha(s) = \beta(s) = \frac{1}{2}$ and $\gamma(s) = 0$, fix p_1 and train $p_{21IL\to RL}(t|s)$ and $p_{22IL\to RL}(t'|s)$ for each IL character s and each pair of RL characters t and t'. In parallel we reverse RL and IL and train $p_{21RL\to IL}(s|t)$ and $p_{22RL\to IL}(s'|t)$ for each RL character t and each pair of IL characters s and s'.

Step three: Our cipher model has to decide after reading one IL character if it will perform a one-to-one, one-to-two or two-to-one mapping. In the first two scenarios the model has enough information to decide, but for the two-to-one mapping the model has to decide before reading the second IL character. For instance, consider converting Bosnian to Serbian. When the model reads the character “c” it has to decide between one-to-one, one-to-two and two-to-one mappings. A good decision will be two-to-one mapping because “ch” maps to ч, hence the system learns a large γ for character “c” but the same γ applies to any other

	afr	dut	bel	pol	bos	srb	dan	swe	mkd	bul	mal	ind
#sent.	0.9M	4M	1.9M	4M	0.8M	1.6M	4M	4M	0.9M	2.1M	0.4M	4M
#tok.	19M	74M	26M	58M	17M	29M	71M	70M	16M	33M	8M	75M

Table 1: Size of monolingual data available for each language. IL and RL are presented in pairs, IL first.

character that follows “c” which is not desirable.

One way to overcome this problem is to change the model to make the decision after reading two IL characters, but this will over-complicate the model. We use a simpler trick instead. We compute $p_{3IL \rightarrow RL}(t|ss')$ from $p_{21RL \rightarrow IL}(s|t)$ and $p_{22RL \rightarrow IL}(s'|t)$ using Bayes rule:

$$p_3(t|ss') = \frac{p(ss'|t)p(t)}{p(ss')} \simeq \frac{p_{21}(s|t)p_{22}(s'|t)p(t)}{p(ss')} \quad (1)$$

The estimate is based on our assumption from the previous step that $p_{22}(s'|t)$ is independent of s. For each RL character t we compute the empirical probability $p(t)$ from monolingual data and $p(ss')$ is the normalization factor.

We set p_3 parameters using equation (1), but before normalizing we manually prune the probabilities. If for IL characters s and s' there exists no RL character t such that $p_{21}(s|t)p_{22}(s'|t)p(t) > 0.01$ we assume that ss' does not map to any RL character. Otherwise, we only keep RL characters for which $p_{21}(s|t)p_{22}(s'|t)p(t) > 0.01$ and then apply the normalization.

Step four: In the final step we fix p_1, p_{21}, p_{22}, and p_3 to the trained values and train $\alpha(s)$, $\beta(s)$, and $\gamma(s)$ for each IL character s.

3.3 Decoding

In the decoding step we compose the cipher WFST with an IL word based language model WFST and find the best path for the input sentence in the resulting WFST (bottom section of Figure 1). If the best path has a high enough score the model outputs the corresponding IL token. Otherwise it outputs the highest scored character sequence produced by the cipher model as and OOV. In our experiments we use 1-gram and 2-gram language models trained on all the existing IL monolingual data (Table 1).

4 Data

We collect data for six pairs of related languages: Afrikaans(afr) / Dutch(dut), Bosnian(bos) / Serbian(srb), Danish(dan) / Swedish(swe), Macedonian(mkd) / Bulgarian(bul), Malaysian(mal) / Indonesian(ind), and Polish(pol) / Belorussian(bel). For each language, we download the monolingual data from Leipzig corpora (Goldhahn et al., 2012). The domain of the data is news, web, and Wikipedia. We consider the language with more data as RL and the one with less data as IL. Table 1 shows the size of available data for each language.

We also extract the list of alphabets for each language from Wikipedia, and collect the Universal Declaration of Human Rights (UDHR) for each IL and RL. We manually sentence align these documents and get 104 sentences and about 1.5K tokens per language. We use these documents for testing the conversion accuracy.

We tokenize and lowercase all the monolingual, parallel and UDHR data with Moses scripts. We remove all non-alphabetic characters from each text according to the alphabet extracted from Wikipedia. This includes numbers, punctuations, and rare/old characters that are not considered as official characters of the language. We keep all the accented variations of characters.

5 Experiments

We translate the UDHR between the related languages using the following methods:

Copy: Copying the text. This is not applicable for languages with different orthography.

LS: One-to-one Letter Substitution cipher. This is equivalent to using WFST1 without a decoding language model.

LS+1g LM: One-to-one letter substitution cipher with a 1-gram word language model at decoding.

PM+1g LM, **PM+2g LM:** The Proposed Method with respectively 1-gram and 2-gram word language model at decoding.

Results are reported for both directions of translation in Tables 2, and 3. For all the language pairs except Malaysian(mal) / Indonesian(ind), the proposed method is the best model with a large margin. Malaysian/Indonesian is a special case where, although the languages have a different vocabulary and a slightly different grammar, they have a common alphabet, and almost all of their cognates are exactly the same. See Figure 3 for an example. As a result the proposed method cannot learn much more than copying.

	afr→dut	bel→pol	bos→srb	dan→swe	mkd→bul	mal→ind
Copy	1.9 / 29.1	- / -	- / -	1.2 / 17.6	5.6 / 34.2	10.0 / 42.7
LS	1.9 / 29.1	0.0 / 7.9	33.2 / 59.4	1.3 / 20.7	9.1 / 39.1	10.0 / 42.7
LS+1g LM	2.9 / 33.3	0.7 / 15.1	32.8 / 59.2	4.5 / 31.3	9.1 / 39.1	10.3 / 43.1
PM+1g LM	4.1 / 36.3	0.0 / 21.8	39.9 / 64.6	6.4 / 36.5	11.8 / 43.3	10.4 / 42.8
PM+2g LM	4.3 / 36.2	1.9 / 24.2	39.2 / 64.4	6.9 / 38.8	11.9 / 43.0	10.4 / 42.8

Table 2: BLEU scores for IL-to-RL translation of UDHR text. Format is BLEU4/BLEU1. Polish / Belorussian and Serbian / Bosnian have different orthographies hence copying is not applicable.

	dut→afr	pol→bel	srb→bos	swe→dan	bul→mkd	ind→mal
Copy	1.9 / 25.0	- /-	- / -	1.2 / 18.7	5.6 / 33.5	10.0 / 41.6
LS	2.13 / 26.5	0.0 / 12.8	33.3 / 60.6	1.3 / 20.7	5.94 / 34.6	10.0 / 41.6
LS+1g LM	3.07 / 27.6	0.7 / 19.7	33.0 / 60.5	3.8 / 32.7	6.9 / 37.6	10.1 / 41.7
PM+1g LM	3.9 / 29.4	1.3 / 23.7	42.3 / 67.8	7.7 / 41.1	9.4 / 40.6	10.1 / 41.7
PM+2g LM	5.2 / 31.2	1.9 / 25.2	42.3 / 67.8	7.6 / 41.2	10.2 / 41.3	10.0 / 41.7

Table 3: BLEU scores for RL-to-IL translation of UDHR text. Format is BLEU4/BLEU1. Polish / Belorussian and Serbian / Bosnian have different orthographies hence copying is not applicable.

mal:	**semua**	manusia	**dilahirkan**	bebas	**dan**	samarata	dari	segi	kemuliaan	**dan**	**hakhak**
ind:	**semua**	orang	**dilahirkan**	merdeka	**dan**	mempunyai	martabat	**dan**	**hakhak**	yang	sama

Figure 3: First sentence of the first article of UDHR in Malaysian (mal) and Indonesian (ind). These languages have a different vocabulary, but their cognates (shown in bold) are exact matches.

afr:	alle	menslike	wesens	word	vry	met	gelyke	--	waardigheid	en	regte
a2d:	**alle**	menslike	wezens	werd	**vrij**	met	gelijke	--	**waardigheid**	**en**	rechte
dut:	**alle**	mensen	------	worden	**vrij**	en	gelijk	in	**waardigheid**	**en**	rechten
afr2en:	all	human	beings	are	free	with	equal	--	dignity	and	rights
a2d2en:	all	human	beings	were	free	with	equal	--	dignity	and	straight
dut2en:	all	people	------	are	free	and	equal	in	dignity	and	rights

Figure 4: First sentence of the first article of UDHR in Afrikaans (afr), Dutch (dut) and its conversion from Afrikaans to Dutch using PM+2-gram LM (a2d), along with their translations to English.

The proposed method translates between Serbian (srb) and Bosnian (bos) almost perfectly. For other pairs, we translate between a quarter and half of the words correctly, but we get few of the higher n-grams. Figure 4 visualizes the conversion of the first sentence of the first article of UDHR from Afrikaans (afr) to Dutch (dut) using PM+2g LM (4.3 BLEU4, 36.2 BLEU1). Observe that 4 out of 10 tokens are translated correctly, close to the 36.2 BLEU1 score, and there is no 3 or 4-gram match. For other tokens except "menslike" the translation is either correct but non-existent in the dutch sentence (wezens = beings, met = with) or has a meaning similar enough that can be useful in the downstream applications (werd = were v.s. worden = are, gelijke = equal(noun) v.s. gelijk = equal(adjective), rechte = straight/right v.s. rechten = rights). The token "menslike" in a2d is an OOV. The model is not able to convert "menslike" (afr) to "mensen" (dut). The language model does not accept other potential conversions and passes out "menslike" (a2d) as the best output of the cipher model.

6 Conclusion

In this paper we present a method for translating texts between closely related languages with potentially different orthography, without needing any parallel data. The only requirement is a few thousand lines of monolingual data for each language and a word language model for the target. Our experiments on six language pairs show the proposed method outperforms others that do not use parallel data.

Acknowledgments

This work was supported by DARPA contract HR0011-15-C-0115. The authors would like to thank Marjan Ghazvininejad, Ulf Hermjakob and Jonathan May for their comments and suggestions.

References

Aimilios Chalamandaris, Athanassios Protopapas, Pirros Tsiakoulis, and Spyros Raptis. 2006. All Greek to me! An automatic Greeklish to Greek transliteration system. In *Proc. LREC*.

Ilyas Cicekli. 2002. A machine translation system between a pair of closely related languages. In *Proc. ISCIS*.

Anna Currey, Alina Karakanta, and Jonathan Poitz. 2016. Using related languages to enhance statistical language models. In *Proc. NAACL*.

Qing Dou and Kevin Knight. 2012. Large scale decipherment for out-of-domain machine translation. In *Proc. EMNLP*.

Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. 2012. Building large monolingual dictionaries at the Leipzig corpora collection: From 100 to 200 Languages. In *Proc. LREC*.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Klein Dan. 2008. Learning bilingual lexicons from monolingual corpora. In *Proc. ACL*.

Jan Haji, Hric Jan, and Kubo Vladislav. 2000. Machine translation of very close languages. In *Proc. ANLP*.

Jirka Hana, Anna Feldman, Chris Brew, and Luiz Amaral. 2006. Tagging Portuguese with a Spanish tagger using cognates. In *Proc. ACL workshop on Cross-Language Knowledge Induction*.

Abo Bakr Hitham, Khaled Shaalan, and Ibrahim Ziedan. 2008. A hybrid approach for converting written Egyptian colloquial dialect into diacritized Arabic. In *Proc. INFOS*.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proc. COLING*.

Philipp Koehn and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proc. ACL workshop on Unsupervised lexical acquisition*.

Grzegorz Kondrak, Daniel Marcu, and Kevin Knight. 2003. Cognates can improve statistical translation models. In *Proc. NAACL*.

Septina Dian Larasati and Vladislav Kubo. 2010. A study of Indonesian-to-Malaysian MT system. In *Proc. MALINDO workshop*.

Gideon S Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *Proc. NAACL*.

Jonathan May, Yassine Benjira, and Abdessamad Echihabi. 2014. An Arabizi-English social media statistical machine translation system. In *Proc. AMTA*.

Iftekhar Naim and Daniel Gildea. 2015. Feature-based decipherment for large vocabulary machine translation. In *arXiv*.

Preslav Nakov and Hwee Tou Ng. 2009. Improved statistical machine translation for resource-poor languages using related resource-rich languages. In *Proc. EMNLP*.

Preslav Nakov and Jörg Tiedemann. 2012. Combining word-level and character-level models for machine translation between closely-related languages. In *Proc. ACL*.

Malte Nuhn, Arne Mauser, and Hermann Ney. 2012. Deciphering foreign language by combining language models and context vectors. In *Proc. ACL*.

Sujith Ravi. 2013. Scalable decipherment for machine translation via hash sampling. In *Proc. ACL*.

Sujith Ravi and Kevin Knight. 2009. Learning phoneme mappings for transliteration without parallel data. In *Proc. ACL*.

Sujith Ravi and Kevin Knight. 2011a. Bayesian inference for Zodiac and other homophonic ciphers. In *Proc. ACL*.

Sujith Ravi and Kevin Knight. 2011b. Deciphering foreign language. In *Proc. ACL*.

Wael Salloum and Nizar Habash. 2010. Dialectal to standard Arabic paraphrasing to improve Arabic-English statistical machine translation. In *Proc. ACL workshop on algorithms and resources for modeling of dialects and language varieties*.

Hassan Sawaf. 2010. Arabic dialect handling in hybrid machine translation. In *Proc. AMTA*.

Kevin P. Scannell. 2006. Machine translation for closely related language pairs. In *Proc. LREC Workshop on Strategies for developing machine translation for minority languages*.

Jörg Tiedemann. 2009. Character-based PSMT for closely related languages. In *Proc. EAMT*.

David Vilar, Jan-T. Peter, and Hermann Ney. 2007. Can we translate letters? In *Proc. ACL workshop on Statistical Machine Translation*.

Identifying Cognate Sets Across Dictionaries of Related Languages

Adam St Arnaud
Dept of Computing Science
University of Alberta
ajstarna@ualberta.ca

David Beck
Dept of Linguistics
University of Alberta
dbeck@ualberta.ca

Grzegorz Kondrak
Dept of Computing Science
University of Alberta
gkondrak@ualberta.ca

Abstract

We present a system for identifying cognate sets across dictionaries of related languages. The likelihood of a cognate relationship is calculated on the basis of a rich set of features that capture both phonetic and semantic similarity, as well as the presence of regular sound correspondences. The similarity scores are used to cluster words from different languages that may originate from a common protoword. When tested on the Algonquian language family, our system detects 63% of cognate sets while maintaining cluster purity of 70%.

1 Introduction

Cognates are words in related languages that have originated from the same word in an ancestor language; for example English *earth* and German *Erde*. On average, cognates display higher phonetic and semantic similarity than random word pairs between languages that are indisputably related (Kondrak, 2013). The term *cognate* is sometimes used within computational linguistics to denote orthographically similar words that have the same meaning (Nakov and Tiedemann, 2012). In this work, however, we adhere to the strict linguistic definition of cognates and aim to distinguish them from lexical borrowings by detecting regular sound correspondences.

Cognate information between languages is critical to the field of historical and comparative linguistics, where it plays a central role in determining the relations and structures of language families (Trask, 1996). Automated phylogenetic reconstructions often rely on cognate information as input (Bouchard-Côté et al., 2013). The percentage of shared cognates can also be used to estimate the time of pre-historic language splits (Dyen et al., 1992). While cognates are valuable to linguists, their identification is a time-consuming process, even for experts, who have to sift through hundreds or even thousands of words in related languages. The languages that are the least well studied, and therefore the ones in which historical linguists are most interested, often lack cognate information.

A number of computational methods have been proposed to automate the process of cognate identification. Many of the systems focus on identifying cognates within classes of semantically equivalent words, such as Swadesh lists of basic concepts. Those systems, which typically consider only the phonetic or orthographic forms of words, can be further divided into the ones that operate on language pairs (*pairwise*) vs. multilingual approaches. However, because of semantic drift, many cognates are no longer exact synonyms, which severely limits the effectiveness of such systems. For example, a cognate pair like English *bite* and French *fendre* "to split" cannot be detected because these words are listed under different basic meanings in the Comparative Indoeuropean Database (Dyen et al., 1992). In addition, the number of basic concepts is typically small.

In this paper, we address the challenging task of identifying cognate sets across multiple languages directly from dictionary lists representing related languages, by taking into account both the forms of words and their dictionary definitions (c.f. Figure 1). Our methods are designed for less-studied languages — we assume only the existence of basic dictionaries containing a substantial number of word forms in a semi-phonetic notation, with the meaning of words conveyed using one of the major languages. Such dictionaries are typically created before Bible translations, which have been accomplished for most of the world's languages.

While our approach is unsupervised, assuming no cognate sets from the analyzed language fam-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2519–2528
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

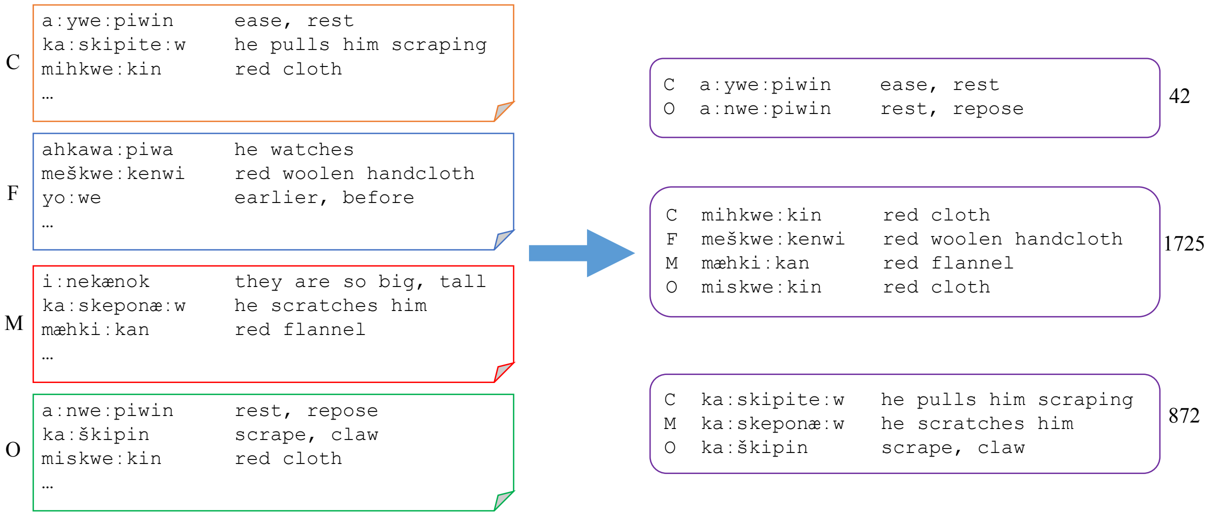


Figure 1: Example of multilingual cognate set identification across four Algonquian dictionaries: Cree (C), Fox (F), Menominee (M) and Ojibwa (O). Cognate set numbers are shown on the right.

ily to start with, it incorporates supervised machine learning models that either leverage cognate data from unrelated families, or use self-training on subsets of likely cognate pairs. We derive two types of models to classify pairs of words across languages as either cognate or not. The language-independent *general model* employs a number of features defined on both word forms and definitions, including word vector representations. The additional *specific models* exploit regular sound correspondences between specific pairs of languages. The scores from the general and specific models inform a clustering algorithm that constructs the proposed cognate sets.

We evaluate our system on dictionary lists that represent four indigenous North American languages from the Algonquian family. On the task of pairwise classification, we achieve a 42% error reduction with respect to the state of the art. On the task of multilingual clustering, our system detects 63% of gold sets, while maintaining a cluster purity score of 70%. The system code is publicly available.[1]

2 Related Work

Most previous work in automatic cognate identification only consider words as cognates if they have identical definitions. As such, they make limited or no use of semantic information. The simplest variant of this task is to make pairwise cognate classifications based on orthographic or phonetic forms. Turchin et al. (2010) apply a heuristic based on consonant classes to identify the ratio of cognate pairs to non-cognate pairs between languages in an effort to determine the likelihood that they are related. Ciobanu and Dinu (2013) find cognate pairs by referring to dictionaries containing etymological information. Rama (2015) experiments with features motivated by string kernels for pairwise cognate classification.

A more challenging version of the task is to cluster cognates within lists of words that have identical definitions. Hauer and Kondrak (2011) use confidence scores from a binary classifier that incorporates a variety of string similarity features to guide an average score clustering algorithm. Hall and Klein (2010, 2011) define generative models that model the evolution of words along a phylogeny according to automatically learned sound laws in the form of parametric edit distances. List and Moran (2013) propose an approach based on sound class alignments and an average score clustering algorithm. List et al. (2016) extend the approach to include partial cognates within word lists.

Cognate identification that considers semantic information is a less-studied problem. Again, the task can be framed as either a pairwise classification or multi-lingual clustering. In a pairwise context, Kondrak (2004) describes a system for identifying cognates between language dictionaries which is based on phonetic similarity, complex multi-phoneme correspondences, and seman-

[1] *https://github.com/ajstarna/SemaPhoR*

tic information. The method of Wang and Sitbon (2014) employs word sense disambiguation combined with classic string similarity measures for finding cognate pairs in parallel texts to aid language learners.

Finally, very little has been published on creating cognate sets based on both phonetic and semantic information, which is the task that we focus on in this paper. Kondrak et al. (2007) combine phonetic and sound correspondence scores with a simple semantic heuristic, and create cognate sets by using graph-based algorithms on connected components. Steiner et al. (2011) aim at a fully automated approach to the comparative method, including cognate set identification and language phylogeny construction. Neither of those systems and datasets are publicly available for the purpose of direct comparison to our method.

3 Methods

In this section, we describe the design of our language-independent general model, as well as the language-specific models. Given a pair of words from related languages, the models produce a score that reflects the likelihood of the words being cognate. The models are implemented as Support Vector Machine (SVM) classifiers via the software package SVM-Light (Joachims, 1999). The scores from both types of models are used to cluster words from different languages into cognate sets.

3.1 Features of the General Model

The general model is a supervised classifier that makes cognate judgments on pairs of words accompanied by their semantic definitions. The model is intended to be language-independent, so that it can be trained on cognate annotations from well-studied languages, and applied to completely unrelated families. The features of the general model are of two kinds: phonetic, which pertain to the analyzed word forms, and semantic, which refer to their definitions.

The **phonetic features** are defined on the word forms, represented in ASJP format (Brown et al., 2008), which is a simplified phonetic representation.

- *Normalized edit distance* is calculated at the character level, and normalized by the length of the longer word.
- *LCSR* is the longest common subsequence ratio of the words.
- *Alignment score* reflects an overall phonetic similarity, provided by the ALINE phonetic aligner (Kondrak, 2009).
- *Consonant match* returns the number of aligned consonants normalized by the number of consonants in the longer word.

For example, consider the words *meškweːkenwi* and *mæhkiːkan* (meSkwekenwi and mEhkikan in ASJP notation) from cognate set 1725 in Figure 1. The corresponding values for the above four features are 0.364, 0.364, 0.523, and 0.714, respectively.

The **semantic features** refer to the dictionary definitions of words. We assume that the definitions are provided in a single meta-language, such as English or Spanish. We consider not only a definition in its entirety, but also its *sub-definitions*, which are separated by commas and semicolons. We distinguish between a closed class of about 300 *stop words*, which express grammatical relationships, and an open class of *content words*, which carry a meaning. Filtering out stopwords reduces the likelihood of spurious matches between dictionary definitions.

Our semantic features can be divided into those that focus on surface definition resemblance, and those that attempt to detect the affinity of meaning. The features of the first type are the following:

- *Sub-definition match* denotes an exact match between any of the word sub-definitions (c.f. set 42 of Figure 1).
- *Sub-definition content match* is performed after removing stop words from definitions.
- *Normalized word-level edit distance* calculates the minimum distance between sub-definitions at the level of words, normalized by the length of the longer sub-definition.
- *Content overlap* fires if any sub-definitions have at least one content word in common.

The second type of semantic features are aimed at detecting deeper meaning connections between definitions. We use WordNet (Fellbaum, 1998) to identify the relations of synonymy and hypernymy, and to associate different inflectional forms of words. The WordNet-based features are as follows:

- *Synonym overlap* indicates a WordNet synonymy relation between content words across sub-definitions (e.g. "ease" and "repose").
- *Hypernym overlap* indicates a WordNet hypernymy relation between content words across sub-definitions (e.g. "flannel" and "cloth").
- *Inflection overlap* is a feature that associates inflectional variants of content words (e.g. "scrape" and "scraping").
- *Inflection synonym overlap* indicates a synonymy relation between lemmas of content words (e.g. "scratches" and "scraping").
- *Inflection hypernym overlap* is defined analogously to the inflection synonym overlap feature.

In order to detect subtle definition similarity that goes beyond inflectional variants and simple semantic relations, we add two features designed to take advantage of recent advances in word vector representations. The two vector-based features are:

- *Vector cosine similarity* is the cosine similarity between the two vectors that represent the average of each vector within a sub-definition.
- *Content vector cosine similarity* is analogous, but only includes content words.

As an example, consider the definitions "he is in mourning" and "she is widowed," from Table 5, which do not fire any of the WordNet-based features. Using the entire definitions yields a vector cosine similarity of 0.566, while considering only the content words "mourning" and "widowed" produces a feature value of 0.146.

3.2 Regular Correspondences

The features described in the previous section are language-independent, but we would also like to take into account cognate information that is specific to pairs of languages, namely regular sound correspondences. For example, *th/d* is a sound correspondence between English and German, occurring in words such as *think/denken* and *leather/Leder*. A model trained on another language family would not be able to learn that a corresponding *th* and *d* is an important indicator of cognation in English/German pairs.

For each language pair, we derive a specific model by implementing the approach of Bergsma and Kondrak (2007). As features, we extract pairs of substrings, up to length 3, that are consistent with the alignment induced by the minimum edit distance algorithm. The models are able to learn when a certain substring in one language corresponds to a certain substring in another language.

In order to train the specific models, we need a substantial number of cognate pairs, which are not initially available in our unsupervised setting. We use a heuristic method to overcome this limitation. We create sets of words that satisfy the following two constraints: (1) identical dictionary definition, and (2) identical first letter. For example, this heuristic will correctly cluster the two words defined as "red cloth" in Figure 1, but will miss the two other cognates from Set 1725. We ensure that every set contains words from at least two languages. The resulting word sets are mutually exclusive, and contain mostly cognates. (In fact, we use this method as our baseline in the Experiments section.) We extract positive training examples from these high-precision sets, and create negative examples by sampling random entries from the language dictionaries. A separate specific model is learned for each language pair in order to capture regular sound correspondences. Note that the specific models include no semantic features. We combine the specific models with the general model by simply averaging their respective scores.

3.3 Cognate Clustering

We apply our general and specific models to score pairs of words across languages. Featurizing all possible pairs of words from all languages is very time consuming, so we first filter out dissimilar word pairs that obtain a normalized score below 0.35 from ALINE. In development experiments, we observed that over 95% of cognate pairs exceed this threshold.

Once pairwise scores have been computed, we cluster words into putative cognate sets using a variant of the UPGMA clustering algorithm (Sokal and Michener, 1958), which has been used in previous work on cognate clustering (Hauer and Kondrak, 2011; List et al., 2016). Initially, all words are placed into their own cluster. The score between clusters is computed as the average of all pairwise scores between the words within those clusters. In each iteration, the two clusters with the highest average score are merged. For efficiency reasons, only positive scores are included in the pairwise similarity matrix, which implies that merges are only performed if all pairwise scores between two clusters are positive. The algorithm terminates when no pair of clusters have a positive average score.

4 Experiments

In this section, we discuss two evaluation experiments. After describing the datasets, we compare our cognate classifier to the current state of the art in pairwise classification. We then consider the evaluation metrics for our main task of cognate set recovery from raw language dictionaries, which is followed by the results on the Algonquian dataset. We refer to our system as SemaPhoR, to reflect the fact that it exploits three kinds of evidence: **Sema**ntic, **Pho**netic, and **R**egular Sound Correspondences.

4.1 Data Sets

Our experiments involve three different language families: Algonquian, Polynesian, and Totonacan.

The Algonquian dataset consists of four dictionary lists (c.f. Figure 1) compiled by Hewson (1993) and normalized by Kondrak (2004). We convert the phonetic forms into a Unicode encoding. The gold-standard annotation consists of 3661 cognate sets, which were established by Hewson on the basis of the regular correspondences identified by Bloomfield (1946). The dataset contains as many as 22,747 unique definitions, which highlights the difference between our task and previous work in cognate identification within word lists, where cognate relationships are restricted to a limited set of basic concepts.

The second dataset corresponds to a version of POLLEX, a large-scale comparative dictionary of over 60 Polynesian languages (Greenhill and Clark, 2011). Table 1 shows that nearly 99% of words in the POLLEX dataset belong to a cognate set, meaning that it is composed almost entirely of cognate sets rather than language dictionaries. This makes the POLLEX dataset unsuitable for system evaluation; however, we use it to train our general classifier, by randomly selecting 25,000 cognate and 250,000 non-cognate word pairs. For calculating our word vector based features, we use the Python package *gensim* (Řehůřek and Sojka, 2010) applied to word vectors pre-trained on approximately 100 billion English words using the approach of Mikolov et al. (2013).[2] The positive training instances are constrained to involve languages that belong to different Polynesian subfamilies.

Family	Lang.	Entries	Sets	Cognates
Algonquian	4	26,985	3,661	8,675
Polynesian	62	27,049	3,690	26,699
Totonacan	10	43,073	?	?

Table 1: The number of languages, total dictionary entries, cognate sets, and cognate words for each language family.

The final dataset consists of 10 dictionaries of the Totonacan language family spoken in Mexico. Since the definitions of the Totonacan dictionaries are in Spanish, we use the Spanish WordNet, a list of 200 stop words, and approximately 1 billion pre-trained Spanish word vectors (Cardellino, 2016) for this dataset.[3] The Totonacan data is yet to be fully analyzed by historical linguists, and as such provides an important motivation for developing our system.

Although the Totonacan dataset includes no cognate information, we manually evaluated a number of candidate cognate sets generated by our system in the development stage. From these annotations, we created a pairwise development set, including all possible 6755 cognate pairs and 67,550 randomly selected non-cognate pairs, and used it for testing our general model that was trained on the Polynesian dataset. The resulting pairwise F-Score of 88.0% shows that our cognate classification model need not be trained on the same language family that it is applied to. Moreover, it confirms that our system can function on datasets where definitions are written in a meta-language that is different from the one used in the training set.

4.2 Pairwise Classification Results

Although our main objective is multilingual clustering, the goal of the first experiment is to compare the effectiveness of our pairwise classifiers against the system of Kondrak (2004), which was designed to process one language pair at a time. As much as possible, we try to follow the original evaluation methodology, which reports 11-point interpolated precision (Manning et al., 2008, page

[2] *https://code.google.com/archive/p/word2vec*

[3] *http://crscardellino.me/SBWCE*

	K-2004	SemaPhoR	
Dev/Train:	CO	CO	POLLEX
CO	78.7	84.8	82.3
CF	69.8	77.8	76.6
CM	61.8	78.4	80.5
FM	65.2	81.7	81.8
FO	69.5	83.3	79.3
MO	64.1	80.3	81.7
Average	66.1	80.3	80.0

Table 2: 11-Point interpolated precision on the Algonquian noun dataset.

158) on lists of positively classified word pairs that have been sorted according to their confidence scores. We also use the same dataset, which is limited to the nouns in the Algonquian data. As the original system contained no machine-learning component, it required no training data, but the Cree-Ojibwa language pair served as the development and tuning set.

We evaluate two variants of our general model: one trained on the Cree-Ojibwa (CO) noun subset, and another on the POLLEX dataset. The language-specific models are trained on each respective language pair, using the unsupervised heuristic approach described in Section 3.2.

Table 2 shows the results on each language pair. K-2004 denotes the results reported in Kondrak (2004). The increase in the average 11-point precision on the five test sets (except Cree-Ojibwa) from 66.1% to 80.3% represents an error reduction of 42%. This improvement demonstrates the superiority of a machine learning approach with a rich feature set over a categorical approach with manually-tuned parameters. When our classifier is trained instead on cognate data from an unrelated Polynesian language family, the average 11-point precision on the test sets drops only slightly to 80.0%, which confirms its generality.

The correspondence-based specific models contribute towards the high accuracy of our system. Without them, the average results on the test sets decrease by 0.9% to 79.4% for the CO-trained model, and by 3.0% to 77.0% for the POLLEX-trained model. We conjecture that the language-specific models are less helpful in the former case because the general model already incorporates much of the information that is particular to the Algonquian family.

4.3 Evaluation Metrics for Clustering

The choice of evaluation metrics for multilingual cognate clustering, which is our main task, requires careful consideration. Pairwise F-score works well for pairwise cognate classification, but in the context of clustering, the number of word pairs grows quadratically with the size of a set, which creates a bias against smaller sets. For example, a set containing 10 words may contribute as much to the pairwise recall as 45 two-word sets.

For the task of clustering words with identical definitions, Hauer and Kondrak (2011) propose to use B-Cubed F-score (Bagga and Baldwin, 1998). However, we found that B-Cubed F-score assigns counter-intuitive scores to clusterings involving datasets of dictionary size, in which many words are outside of any cognate set in the reference annotation. For example, on the Algonquian dataset, a trivial strategy of placing each word into its own cluster (*MaxPrecision*) would achieve a B-Cubed F-Score of 89.6%.

In search for a better metric, we considered MUC (Vilain et al., 1995), which is designed to score co-reference algorithms. MUC assigns precision, recall and F-Score based on the number of *missing links* in the proposed clusters. However, as pointed out by Bagga and Baldwin (1998), when penalizing incorrectly placed elements, MUC is insensitive to the size of the cluster in question. For example, a completely useless clustering of all Algonquian words into one giant set (*MaxRecall*) yields a higher MUC F-Score than most of the reasonably effective approaches.

We believe that an appropriate measure of *recall* for a cognate clustering system is the total number of *found sets*. A set that exists in the gold annotation is considered found if any of the words that belong to the set are clustered together by the system. We report both partially and completely found sets. Arguably, the number of partially found sets may be more important, as it is easier for a linguist to extend a found set to other languages than to discover the set in the first place. In fact, a discovery of a single pair of cross-lingual cognates implies the existence of a corresponding proto-word in their ancestor language, which is likely to have reflexes in the other languages of the family.

As a corresponding measure of *precision*, we report *cluster purity*, which has previously been used to evaluate cognate clusterings by Hall and

Klein (2011) and Bouchard-Côté et al. (2013). In order to calculate purity, each output set is first matched to a gold set with which it has the most words in common. Then purity is calculated as the fraction of total words that are matched to the correct gold set. More formally, let $G = \{G_1, G_2, ..., G_n\}$ be a gold clustering and $C = \{C_1, C_2, ..., C_m\}$ be a proposed clustering. Then

$$purity(C, G) = \frac{1}{N} \sum_{i=1}^{m} max_j |G_j \cap C_i|$$

where N is the total number of words. The tradeoff between the number of found sets and cluster purity gives a good idea of the performance of a cognate clustering. For example, both of the *MaxRecall* and *MaxPrecision* strategies mentioned above would obtain 100% scores according to one of the measures, but close to 0% according to the other.

4.4 Cognate Clustering Results

In our main experiment, we apply our system to the task of creating cognate sets from the Algonquian dataset. The general classification model is trained on the POLLEX dataset, as described in Section 4.2, while the language-specific models are derived following the procedure described in Section 3.2. The scores from both models are then used to guide the clustering process. Only one word from each language is allowed per cluster.

Since most work done in the area of cognate clustering starts from semantically aligned word lists, it is difficult to make a direct comparison. We report the results obtained with LEXSTAT (List and Moran, 2013).[4] The system has no capability to consider the degree of semantic similarity between words, so we first group together the words that have identical definitions and provide these as its input. As a baseline, we adopt the heuristic described in Section 3.2, which creates sets from words that have identical definitions and start with the same letter.

System	Found Sets	Purity
Heuristic Baseline	18.9 (9.9)	96.4
LEXSTAT	19.6 (10.5)	97.1
SemaPhoR	**63.1 (48.2)**	**70.3**

Table 3: Cognate clustering results on the Algonquian dataset (in %). The absolute percentage of fully found sets is given in parentheses.

Table 3 shows the results. LEXSTAT performs slightly better than the heuristic baseline, but both are limited by their inability to relate words that have non-identical definitions. In fact, only 21.4% of all gold cognate sets in the Algonquian dataset contain at least two words with the same definition, which establishes an upper bound on the number of found sets for systems that are designed to operate on word lists, rather than dictionaries. For example, most of the cognates in Figure 1 cannot be captured by such systems.

Our system, SemaPhoR, finds approximately three times as many cognate sets as LEXSTAT, and over 75% of those sets are complete with respect to the gold annotation. In practical terms, our system is able to provide concrete evidence for the existence of most of the proto-words that have reflexes in the recorded languages, and identifies the majority of those reflexes in the process. The purity of the produced clusters indicates that there are many more hits than misses in the system output. In addition, the clusters can be sorted according to their confidence scores, in order to facilitate the analysis of the results by an expert linguist.

5 Discussion

In this section, we interpret the results of our feature ablation experiments, and analyze several types of errors made by our system.

5.1 Feature Ablation

In order to determine the relative effect of the features described in Section 3.1, we test four variants of the general model, which employ increasingly complex subsets of features. The simplest variant uses only the phonetic features that are defined on the word forms. The next variant adds the features that consider surface definition resemblance. The third variant also includes WordNet-based semantic features. The final variant is the full system configuration that incorporates the features defined on word vector representations, but without language-specific models.

Table 4 shows the results. The phonetic features alone are sufficient to detect just over half of the cognate sets. Each successive variant substantially improves the recall at a cost of slightly lower precision. The full feature set yields a 27% relative increase in the number of found sets over

[4] *http://lingpy.org*

Features	Found Sets	Purity
Phonetic only	52.0 (36.3)	70.2
+ Definitions	57.4 (41.7)	68.4
+ WordNet	61.9 (46.9)	68.1
+ Word Vectors	66.2 (51.3)	66.5

Table 4: Cognate clustering results on the Algonquian dataset (in %) with subsets of features.

the phonetic-only variant, with only a 5% drop in cluster purity.

In comparison with our full system, which incorporates the language-specific models, the final variant finds a greater number of the cognate sets, but with a trade-off in overall precision (c.f. *SemaPhoR* in Table 3). This shows that our system is able to exploit regular sound correspondences to filter out a substantial number of false cognates, such as lexical borrowings or chance resemblances. However, the overall contribution of the specific models is relatively small. One possible explanation is that the Algonquian languages are relatively closely related, which enables the general model to discern most of the cognate relationships on the basis of phonetic and semantic similarity. For example, many of the regular correspondences detected by the specific models, such as *s:s* and *hk:kk*, involve identical phonemes. The impact of the specific models could be greater for a more distantly-related language family.

5.2 Error Analysis

A number of omission errors can be traced to the imperfect heuristic that constrains the positive training instances for the language-specific models to begin with the same letter. Indeed, 88.1% of Algonquian cognate sets are composed of words that share the initial phoneme. While this constraint yields high-precision training sets that satisfy the transitivity condition, it also introduces a bias against cognates that differ in the first letter.

The second type of errors made by our system are caused by semantic drift that has altered the meaning of the original proto-word. For example, “sickness” is difficult for our general model to associate with “bitterness, pain.” On the other hand, there are many instances where our system is successful in identifying non-obvious semantic similarity, often thanks to the word vector features of our model. Table 5 provides examples of cognates found by our system that would have been very difficult to identify without word vector technology.

C	maːyaːčiteːheːw	he is angry
M	mianaːčetæhæːw	he is nauseated
C	piːsisiw	he is in bits
O	piːssisi	he is ground up
C	ayiwiskaweːw	he is taller than someone else
O	aniwiškaw	precede, surpass someone
C	siːkaːwiw	he is in mourning
M	seːkawew	she is widowed

Table 5: Examples of cognates found with the assistance of word vector features.

A substantial number of apparent errors made by our system are due to the complex polysynthetic morphology of Algonquian, in which a single Algonquian word can express a meaning of several English words. A number of distinct cognate sets are highly similar in their definitions and phonetic forms. For example, our system erroneously places the Menominee word *aːkuaqtæːhsen* into a cluster with two similar Cree and Ojibwa words, instead of associating it with the Ojibwa word *aːkawaːtteːššin* (Table 6). Although it could be argued that such closely-related forms are all cognate, we refrain from modifying any gold annotations, even if this negatively impacts the overall accuracy of our system.

C	aːkawaːsteːsimoːw	he lies down in the shade
O	aːkawaːtteːššimoːn	be in the shadow
M	**aːkuaqtæːhsen**	**he is in the shade**
O	aːkawaːtteːššin	make shadow

Table 6: A clustering error due to morphology.

Finally, some apparent errors made by our system may not be errors at all, but rather reflect the incompleteness of the gold annotation. For example, consider the two false positive pairs in Table 7. Even though they are not listed in Hewson’s (1993) etymological dictionary, the exact definition match, coupled with striking phonetic similarity and the presence of regular sound correspondences strongly suggest that they are actually cognates.

M	pekuač	growing wild
O	pekwači	growing wild
C	nisoːteːw	twin
O	niːšoːteːnq	twin

Table 7: Examples of proposed cognate sets that are not found in the gold data.

6 Conclusion

We have presented a comprehensive system for the novel task of identifying cognate sets directly from dictionaries of related languages by leveraging both word forms and word definitions. To the best of our knowledge, it is the first system to use word vector representations for cognate identification. The main insight from our work is that a cognate classification model can be trained on one language family, and achieve impressive results when classifying a completely unrelated language family. This allows cognate information from a high-resource language family to guide cognate identification between languages that little is known about.

There are aspects of cognate identification that can only be detected by human experts, such as cognates that have undergone extensive phonetic and semantic changes, or large-scale lexical borrowing between languages. However, we believe that our system represents a step towards automated cognate identification, and will prove a useful tool for historical linguists.

Acknowledgments

We thank the following students that contributed to our cognate-related projects over the last few years (in alphabetical order): Matthew Darling, Jacob Denson, Philip Dilts, Bradley Hauer, Mildred Lau, Tyler Lazar, Garrett Nicolai, Dylan Stankievech, Nicholas Tam, and Cindy Xiao. This research was partially funded by the Natural Sciences and Engineering Research Council of Canada.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*.

Shane Bergsma and Grzegorz Kondrak. 2007. Alignment-based discriminative string similarity. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.

Leonard Bloomfield. 1946. Algonquian. In Harry Hoijer et al., editor, *Linguistic Structures of Native America*, volume 6 of *Viking Fund Publications in Anthropology*.

Alexandre Bouchard-Côté, David Hall, Thomas L. Griffiths, and Dan Klein. 2013. Automated reconstruction of ancient languages using probabilistic models of sound change. *PNAS*.

Cecil H. Brown, Eric W. Holman, and Soren Wichmann. 2008. Automated classification of the world's languages: A description of the method and preliminary results. In *Language Typology and Universals*, volume 61.

Cristian Cardellino. 2016. Spanish billion words corpus and embeddings.

Aline Maria Ciobanu and Liviu P. Dinu. 2013. A dictionary-based approach for evaluating orthographic methods in cognates identification. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*.

Isidore Dyen, Joseph B Kruskal, and Paul Black. 1992. An Indoeuropean classification: A lexicostatistical experiment. *Transactions of the American Philosophical society*, 82(5).

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

Simon J. Greenhill and Ross Clark. 2011. Pollex-online: The Polynesian lexicon project online. In *Oceanic Linguistics*, volume 50.

David Hall and Dan Klein. 2010. Finding cognate groups using phylogenies. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*.

David Hall and Dan Klein. 2011. Large-scale cognate recovery. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Bradely Hauer and Grzegorz Kondrak. 2011. Clustering semantically equivalent words into cognate sets in multilinguial lists. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*.

John Hewson. 1993. *A computer-generated dictionary of proto-Algonquian*. Canadian Museum of Civilization, Hull, Quebec.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

Grzegorz Kondrak. 2004. Combining evidence in cognate identification. In *Proceedings of the Seventeenth Canadian Conference on Artificial Intelligence*.

Grzegorz Kondrak. 2009. Identification of cognates and recurrent sound correspondences in word lists. *Traitement automatique des langues et langues anciennes*, 50(2):201–235.

Grzegorz Kondrak. 2013. Word similarity, cognation, and translational equivalence. In *Approaches to Measuring Linguistic Differences*, pages 375–386. De Gruyter Mouton.

Grzegorz Kondrak, David Beck, and Philip Dilts. 2007. Creating a comparative dictionary of Totonac-Tepehua. In *Proceedings of the ACL Workshop on Computing and Historical Phonology (9th Meeting of SIGMORPHON)*, pages 134–141.

Johann-Mattis List, Philippe Lopez, and Eric Bapteste. 2016. Using sequence similarity networks to identify partial cognates in multilingual wordlists. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Johann-Mattis List and Steven Moran. 2013. An open source toolkit for quantitative historical linguistics. In *Proceedings of the ACL 2013 System Demonstrations*.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 1. Cambridge University Press.

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

Preslav Nakov and Jörg Tiedemann. 2012. Combining word-level and character-level models for machine translation between closely-related languages. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Taraka Rama. 2015. Automatic cognate identification with gap-weighted string subsequences. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC Workshop on New Challenges for NLP Frameworks*.

Robert R. Sokal and Charles D. Michener. 1958. A statistical method for evaluating systematic relationships. In *University of Kansas Science Bulletin*, volume 38.

Lydia Steiner, Peter F Stadler, and Michael Cysouw. 2011. A pipeline for computational historical linguistics. *Language Dynamics and Change*, 1(1).

R.L. Trask. 1996. *Historical Linguistics*. St Martin's Press, Inc., New York, NY.

Peter Turchin, Ilia Peiros, and Murray Gell-Mann. 2010. Analyzing genetic connections between languages by matching consonant classes. In *Journal of Language Relationship*.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Conference on Message Understanding*.

Haoxing Wang and Laurianne Sitbon. 2014. Multilingual lexical resources to detect cognates in non-aligned texts. In *The Twelfth Annual Workshop of the Australasia Language Technology Association*.

Learning Language Representations for Typology Prediction

Chaitanya Malaviya and **Graham Neubig** and **Patrick Littell**
Language Technologies Institute
Carnegie Mellon University
{cmalaviy,gneubig,pwl}@cs.cmu.edu

Abstract

One central mystery of neural NLP is what neural models "know" about their subject matter. When a neural machine translation system learns to translate from one language to another, does it learn the syntax or semantics of the languages? Can this knowledge be extracted from the system to fill holes in human scientific knowledge? Existing typological databases contain relatively full feature specifications for only a few hundred languages. Exploiting the existence of parallel texts in more than a thousand languages, we build a massive many-to-one neural machine translation (NMT) system from 1017 languages into English, and use this to predict information missing from typological databases. Experiments show that the proposed method is able to infer not only syntactic, but also phonological and phonetic inventory features, and improves over a baseline that has access to information about the languages' geographic and phylogenetic neighbors.[1]

1 Introduction

Linguistic typology is the classification of human languages according to syntactic, phonological, and other classes of features, and the investigation of the relationships and correlations between these classes/features. This study has been a scientific pursuit in its own right since the 19th century (Greenberg, 1963; Comrie, 1989; Nichols, 1992), but recently typology has borne practical fruit within various subfields of NLP, particularly on problems involving lower-resource languages.

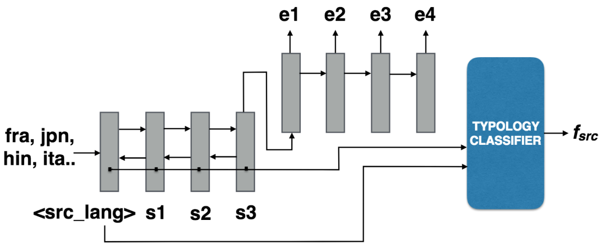


Figure 1: Learning representations from multilingual neural MT for typology classification. (Model MTBOTH)

Typological information from sources like the World Atlas of Language Structures (WALS) (Dryer and Haspelmath, 2013), has proven useful in many NLP tasks (O'Horan et al., 2016), such as multilingual dependency parsing (Ammar et al., 2016), generative parsing in low-resource settings (Naseem et al., 2012; Täckström et al., 2013), phonological language modeling and loanword prediction (Tsvetkov et al., 2016), POS-tagging (Zhang et al., 2012), and machine translation (Daiber et al., 2016).

However, the needs of NLP tasks differ in many ways from the needs of scientific typology, and typological databases are often only sparsely populated, by necessity or by design.[2] In NLP, on the other hand, what is important is having a relatively full set of features for the particular group of languages you are working on. This mismatch of needs has motivated various proposals to reconstruct missing entries, in WALS and other databases, from known entries (Daumé III and Campbell, 2007; Daumé III, 2009; Coke et al., 2016; Littell et al., 2017).

In this study, we examine whether we can

[1] Code and learned vectors are available at http://github.com/chaitanyamalaviya/lang-reps

[2] For example, each chapter of WALS aims to provide a statistically balanced set of languages over language families and geographical areas, and so many languages are left out in order to maintain balance.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2529–2535
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

tackle the problem of inferring linguistic typology from parallel corpora, specifically by training a massively multi-lingual neural machine translation (NMT) system and using the learned representations to infer typological features for each language. This is motivated both by prior work in linguistics (Bugarski, 1991; García, 2002) demonstrating strong links between translation studies and tools for contrastive linguistic analysis, work in inferring typology from bilingual data (Östling, 2015) and English as Second Language texts (Berzak et al., 2014), as well as work in NLP (Shi et al., 2016; Kuncoro et al., 2017; Belinkov et al., 2017) showing that syntactic knowledge can be extracted from neural nets on the word-by-word or sentence-by-sentence level. This work presents a more holistic analysis of whether we can discover what neural networks learn about the linguistic concepts of an entire language by aggregating their representations over a large number of the sentences in the language.

We examine several methods for discovering feature vectors for typology prediction, including those learning a language vector specifying the language while training multilingual neural language models (Östling and Tiedemann, 2017) or neural machine translation (Johnson et al., 2016) systems. We further propose a novel method for aggregating the values of the latent state of the encoder neural network to a single vector representing the entire language. We calculate these feature vectors using an NMT model trained on 1017 languages, and use them for typlogy prediction both on their own and in composite with feature vectors from previous work based on the genetic and geographic distance between languages (Littell et al., 2017). Results show that the extracted representations do in fact allow us to learn about the typology of languages, with particular gains for syntactic features like word order and the presence of case markers.

2 Dataset and Experimental Setup

Typology Database: To perform our analysis, we use the URIEL language typology database (Littell et al., 2017), which is a collection of binary features extracted from multiple typological, phylogenetic, and geographical databases such as WALS (World Atlas of Language Structures) (Collins and Kayne, 2011), PHOIBLE (Moran et al., 2014), Ethnologue (Lewis et al., 2015), and Glottolog (Hammarström et al., 2015). These features are divided into separate classes regarding syntax (e.g. whether a language has prepositions or postpositions), phonology (e.g. whether a language has complex syllabic onset clusters), and phonetic inventory (e.g. whether a language has interdental fricatives). There are 103 syntactical features, 28 phonology features and 158 phonetic inventory features in the database.

Baseline Feature Vectors: Several previous methods take advantage of typological implicature, the fact that some typological traits correlate strongly with others, to use known features of a language to help infer other unknown features of the language (Daumé III and Campbell, 2007; Takamura et al., 2016; Coke et al., 2016). As an alternative that does not necessarily require pre-existing knowledge of the typological features in the language at hand, Littell et al. (2017) have proposed a method for inferring typological features directly from the language's k nearest neighbors (k-NN) according to geodesic distance (distance on the Earth's surface) and genetic distance (distance according to a phylogenetic family tree). In our experiments, our baseline uses this method by taking the 3-NN for each language according to normalized geodesic+genetic distance, and calculating an average feature vector of these three neighbors.

Typology Prediction: To perform prediction, we trained a logistic regression classifier[3] with the baseline k-NN feature vectors described above and the proposed NMT feature vectors described in the next section. We train individual classifiers for predicting each typological feature in a class (syntax etc). We performed 10-fold cross-validation over the URIEL database, where we train on 9/10 of the languages to predict 1/10 of the languages for 10 folds over the data.

3 Learning Representations for Typology Prediction

In this section we describe three methods for learning representations for typology prediction with multilingual neural models.

LM Language Vector Several methods have been proposed to learn multilingual language

[3] We experimented with a non-linear classifier as well, but the logistic regression classifier performed better.

models (LMs) that utilize vector representations of languages (Tsvetkov et al., 2016; Östling and Tiedemann, 2017). Specifically, these models train a recurrent neural network LM (RNNLM; Mikolov et al. (2010)) using long short-term memory (LSTM; Hochreiter and Schmidhuber (1997)) with an additional vector representing the current language as an input. The expectation is that this vector will be able to capture the features of the language and improve LM accuracy. Östling and Tiedemann (2017) noted that, intriguingly, agglomerative clustering of these language vectors results in something that looks roughly like a phylogenetic tree, but stopped short of performing typological inference. We train this vector by appending a special token representing the source language (e.g. "⟨fra⟩" for French) to the beginning of the source sentence, as shown in Fig. 1, then using the word representation learned for this token as a representation of the language. We will call this first set of feature vectors LMVec, and examine their utility for typology prediction.

NMT Language Vector In our second set of feature vectors, MTVec, we similarly use a language embedding vector, but instead learn a multilingual neural MT model trained to translate from many languages to English, in a similar fashion to Johnson et al. (2016); Ha et al. (2016). In contrast to LMVec, we hypothesize that the alignments to an identical sentence in English, the model will have a stronger signal allowing it to more accurately learn vectors that reflect the syntactic, phonetic, or semantic consistencies of various languages. This has been demonstrated to some extent in previous work that has used specifically engineered alignment-based models (Lewis and Xia, 2008; Östling, 2015; Coke et al., 2016), and we examine whether these results apply to neural network feature extractors and expand beyond word order and syntax to other types of typology as well.

NMT Encoder Mean Cell States Finally, we propose a new vector representation of a language (MTCell) that has not been investigated in previous work: the average hidden cell state of the encoder LSTM for all sentences in the language. Inspired by previous work that has noted that the hidden cells of LSTMs can automatically capture salient and interpretable information such as syntax (Karpathy et al., 2015; Shi et al., 2016) or sentiment (Radford et al., 2017), we expect that the cell states will represent features that may be linked to the typology of the language. To create vectors for each language using LSTM hidden states, we obtain the mean of cell states ($\boldsymbol{c}$ in the standard LSTM equations) for all time steps of all sentences in each language.[4]

	Syntax		Phonology		Inventory	
	-Aux	+Aux	-Aux	+Aux	-Aux	+Aux
None	69.91	83.07	77.92	86.59	85.17	90.68
LMVec	71.32	82.94	80.80	86.74	87.51	89.94
MTVec	74.90	83.31	82.41	87.64	89.62	90.94
MTCell	75.91	85.14	84.33	88.80	90.01	90.85
MTBoth	**77.11**	**86.33**	**85.77**	**89.04**	**90.06**	**91.03**

Table 1: Accuracy of syntactic, phonological, and inventory features using LM language vectors (LMVec), MT language vectors (MTVec), MT encoder cell averages (MTCell) or both MT feature vectors (MTBoth). Aux indicates auxiliary information of geodesic/genetic nearest neighbors; "None -Aux" is the majority class chance rate, while "None +Aux" is a 3-NN classification.

4 Experiments

4.1 Multilingual Data and Training Regimen

To train a multilingual neural machine translation system, we used a corpus of Bible translations that was obtained by scraping a massive online Bible database at `bible.com`.[5] This corpus contains data for 1017 languages. After preprocessing the corpus, we obtained a training set of 20.6 million sentences over all languages.

The implementation of both the LM and NMT models described in §3 was done in the DyNet toolkit (Neubig et al., 2017). In order to obtain a manageable shared vocabulary for all languages, we divided the data into subwords using joint byte-pair encoding of all languages (Sennrich et al., 2016) with 32K merge operations. We

[4] We also tried using the mean of final hidden cell states of the encoder LSTM, but the mean cell state over all words in the sentence gave improved performance. Additionally, we tried using the hidden states $\boldsymbol{h}$, but we found that these had significantly less information and lesser variance, due to being modulated by the output gate at each time step.

[5] A possible concern is that Bible translations may use archaic language not representative of modern usage. However, an inspection of the data did not turn up such archaisms, likely because the bulk of world Bible translation was done in the late 19th and 20th centuries. In addition, languages that do have antique Bibles are also those with many other Bible translations, so the effect of the archaisms is likely limited.

used LSTM cells in a single recurrent layer with 512-dimensional hidden state and input embedding size. The Adam optimizer was used with a learning rate of 0.001 and a dropout of 0.5 was enforced during training.

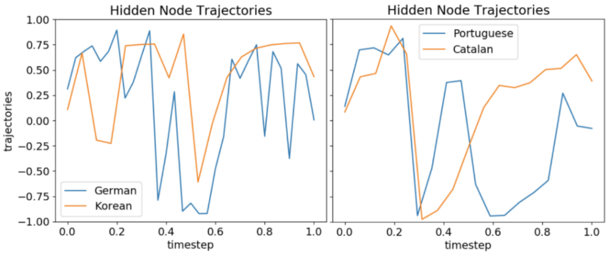
4.2 Results and Discussion

The results of the experiments can be found in Tab. 1. First, focusing on the “-Aux” results, we can see that all feature vectors obtained by the neural models improve over the chance rate, demonstrating that indeed it is possible to extract information about linguistic typology from unsupervised neural models. Comparing LMVEC to MTVEC, we can see a convincing improvement of 2-3% across the board, indicating that the use of bilingual information does indeed provide a stronger signal, allowing the network to extract more salient features. Next, we can see that MTCELL further outperforms MTVEC, indicating that the proposed method of investigating the hidden cell dynamics is more effective than using a statically learned language vector. Finally, combining both feature vectors as MTBOTH leads to further improvements. To measure statistical significance of the results, we performed a paired bootstrap test to measure the gain between NONE+AUX and MTBOTH+AUX and found that the gains for syntax and inventory were significant (p=0.05), but phonology was not, perhaps because the number of phonological features was fewer than the other classes (only 28).

When further using the geodesic/genetic distance neighbor feature vectors, we can see that these trends largely hold although gains are much smaller, indicating that the proposed method is still useful in the case where we have a-priori knowledge about the environment in which the language exists. It should be noted, however, that the gains of LMVEC evaporate, indicating that access to aligned data may be essential when inferring the typology of a new language. We also noted that the accuracies of certain features decreased from NONE-AUX to MTBOTH-AUX, particularly gender markers, case suffix and negative affix, but these decreases were to a lesser extent in magnitude than the improvements.

Feature	NONE	MT	Gain
S_NUMERAL_AFTER_NOUN	37.40	81.26	43.86
S_NUMERAL_BEFORE_NOUN	46.49	83.22	36.73
S_POSSESSOR_AFTER_NOUN	42.05	75.60	33.55
S_OBJECT_BEFORE_VERB	50.97	80.89	29.92
S_ADPOSITION_AFTER_NOUN	52.41	79.10	26.69
P_UVULAR_CONTINUANTS	77.57	97.37	19.80
P_LATERALS	67.30	86.48	19.18
P_LATERAL_L	64.05	78.16	14.10
P_LABIAL_VELARS	82.16	95.93	13.76
P_VELAR_NASAL_INITIAL	72.14	85.82	13.68
I_VELAR_NASAL	39.89	62.08	22.20
I_ALVEOLAR_LATERAL_APPROXIMANT	60.92	79.32	18.40
I_ALVEOLAR_NASAL	81.49	92.98	11.48
I_VOICED_LABIODENTAL_FRICATIVE	65.75	77.10	11.36
I_VOICELESS_PALATAL_FRICATIVE	82.41	93.66	11.25

Table 2: Top 5 improvements from “NONE -Aux” to “MTBOTH -Aux” in the syntax (“S_”), phonology (“P_”), and inventory (“I_”) classes.



GER: Ich bin das A und das O , der Anfang und das Ende , spricht Gott der HERR , der da ist und der da war und der da kommt , der Allmächtige .

KOR: 지금도 계시고 전에도 계셨고 앞으로 오실 전능하신 주 하나님께서 “ 나는 알파요 오메가다 ” 하고 말씀하십니다 .

POR: Paulo , chamado para ser apóstolo de Cristo Jesus pela vontade de Deus , e o irmão Sóstenes

CAT: Pau , cridat per voler de Déu a ser apòstol de Jesucrist , i el germà Sòstenes

Figure 2: Cell trajectories for sentences in languages where S_OBJ_BEFORE_VERB is either active or inactive.

Interestingly, and in contrast to previous methods for inferring typology from raw text, which have been specifically designed for inducing word order or other syntactic features (Lewis and Xia, 2008; Östling, 2015; Coke et al., 2016), our proposed method is also able to infer information about phonological or phonetic inventory features. This may seem surprising or even counterintuitive, but a look at the most-improved phonology/inventory features (Tab. 2) shows a number of features in which languages with the “non-default” option (e.g. having uvular consonants or initial velar nasals, *not* having lateral consonants, etc.) are concentrated in particular geographical regions. For example, uvular consonants are not common world-wide, but are common in particular geographic regions like the North American Pacific Northwest and the Caucasus (Maddieson, 2013b), while initial velar nasals are common in Southeast Asia (Anderson, 2013), and lateral consonants are *uncommon* in the Amazon Basin (Maddieson, 2013a). Since these are also regions with a particular and sometimes distinct syntactic character, we think the model may be finding regional clusters through syntax, and seeing an improvement in regionally-distinctive phonology/inventory features as a side effect.

Finally, given that MTCELL uses the feature vectors of the latent cell state to predict typology, it is of interest to observe how these latent cells behave for typologically different languages. In Fig. 2 we examine the node that contributed most to the prediction of "S_OBJ_BEFORE_VERB" (the node with maximum weight in the classifier) for German and Korean, where the feature is active, and Portuguese and Catalan, where the feature is inactive. We can see that the node trajectories closely track each other (particularly at the beginning of the sentence) for Portuguese and Catalan, and in general the languages where objects precede verbs have higher average values, which would be expressed by our mean cell state features. The similar trends for languages that share the value for a typological feature (S_OBJ_BEFORE_VERB) indicate that information stored in the selected hidden node is consistent across languages with similar structures.

5 Conclusion and Future Work

Through this study, we have shown that neural models can learn a range of linguistic concepts, and may be used to impute missing features in typological databases. In particular, we have demonstrated the utility of learning representations with parallel text, and results hinted at the importance of modeling the dynamics of the representation as models process sentences. We hope that this study will encourage additional use of typological features in downstream NLP tasks, and inspire further techniques for missing knowledge prediction in under-documented languages.

Acknowledgments

We thank Lori Levin and David Mortensen for their useful comments and also thank the reviewers for their feedback about this work.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Gregory D.S. Anderson. 2013. The velar nasal. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Yevgeni Berzak, Roi Reichart, and Boris Katz. 2014. Reconstructing native language typology from foreign language usage. In *Eighteenth Conference on Computational Natural Language Learning (CoNLL)*.

Ranko Bugarski. 1991. Contrastive analysis of terminology and the terminology of contrastive analysis. *Languages in Contact und Contrast. Essays in Contact Linguistics/Edited by Vladimir Ivir and Damir Kalogjera.–Berlin*, pages 73–82.

Reed Coke, Ben King, and Dragomir Radev. 2016. Classifying syntactic regularities for hundreds of languages. *arXiv preprint arXiv:1603.08016*.

Chris Collins and Richard Kayne. 2011. *Syntactic Structures of the World's Languages*. New York University, New York.

Bernard Comrie. 1989. *Language Universals and Linguistic Typology: Syntax and Morphology*. Blackwell, Oxford.

Joachim Daiber, Miloš Stanojević, and Khalil Sima'an. 2016. Universal reordering via linguistic typology. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3167–3176, Osaka, Japan. The COLING 2016 Organizing Committee.

Hal Daumé III. 2009. Non-parametric Bayesian areal linguistics. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 593–601, Boulder, Colorado. Association for Computational Linguistics.

Hal Daumé III and Lyle Campbell. 2007. A Bayesian model for discovering typological implications. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 65–72, Prague, Czech Republic. Association for Computational Linguistics.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Noelia Ramón García. 2002. Contrastive linguistics and translation studies interconnected: The corpus-based approach. *Linguistica Antverpiensia, New Series–Themes in Translation Studies*, (1).

Joseph Greenberg. 1963. Some universals of grammar with particular reference to the order of meaningful elements. In Joseph Greenberg, editor, *Universals of Language*, pages 110–113. MIT Press, London.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798*.

Harald Hammarström, Robert Forkel, Martin Haspelmath, and Sebastian Bank. 2015. *Glottolog 2.6*. Max Planck Institute for the Science of Human History, Jena.

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

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

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258, Valencia, Spain. Association for Computational Linguistics.

M. Paul Lewis, Gary F. Simons, and Charles D. Fennig. 2015. *Ethnologue: Languages of the World, Eighteenth edition*. SIL International, Dallas, Texas.

William D Lewis and Fei Xia. 2008. Automatically identifying computationally relevant typological features. In *Proceedings of the Third International Joint Conference on Natural Language Processing, Volume II*, pages 685–690.

Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. Uriel and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14, Valencia, Spain. Association for Computational Linguistics.

Ian Maddieson. 2013a. Lateral consonants. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Ian Maddieson. 2013b. Uvular consonants. In Matthew S. Dryer and Martin Haspelmath, editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3.

Steven Moran, Daniel McCloy, and Richard Wright. 2014. *PHOIBLE Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 629–637. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Joanna Nichols. 1992. *Linguistic Diversity in Space and Time*. University of Chicago Press, Chicago.

Helen O'Horan, Yevgeni Berzak, Ivan Vulic, Roi Reichart, and Anna Korhonen. 2016. Survey on the use of typological information in natural language processing. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1297–1308, Osaka, Japan. The COLING 2016 Organizing Committee.

Robert Östling. 2015. Word order typology through multilingual word alignment. In *The 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 205–211.

Robert Östling and Jörg Tiedemann. 2017. Continuous multilinguality with language vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 644–649, Valencia, Spain. Association for Computational Linguistics.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071, Atlanta, Georgia. Association for Computational Linguistics.

Hiroya Takamura, Ryo Nagata, and Yoshifumi Kawasaki. 2016. Discriminative analysis of linguistic features for typological study. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1357–1366, San Diego, California. Association for Computational Linguistics.

Yuan Zhang, Roi Reichart, Regina Barzilay, and Amir Globerson. 2012. Learning to map into a universal POS tagset. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1368–1378. Association for Computational Linguistics.

Cheap Translation for Cross-Lingual Named Entity Recognition

Stephen Mayhew Chen-Tse Tsai Dan Roth
University of Illinois, Urbana-Champaign
201 N. Goodwin
Urbana, Illinois, 61801
{mayhew2, ctsai12, danr}@illinois.edu

Abstract

Recent work in NLP has attempted to deal with low-resource languages but still assumed a resource level that is not present for most languages, e.g., the availability of Wikipedia in the target language. We propose a simple method for cross-lingual named entity recognition (NER) that works well in settings with *very* minimal resources. Our approach makes use of a lexicon to "translate" annotated data available in one or several high resource language(s) into the target language, and learns a standard monolingual NER model there. Further, when Wikipedia is available in the target language, our method can enhance Wikipedia based methods to yield state-of-the-art NER results; we evaluate on 7 diverse languages, improving the state-of-the-art by an average of 5.5% F1 points. With the minimal resources required, this is an extremely portable cross-lingual NER approach, as illustrated using a truly low-resource language, Uyghur.

1 Introduction

In recent years, interest in the natural language processing (NLP) community has expanded to include multilingual applications. Although this uptick of interest has produced diverse annotated corpora, most languages are still classified as low-resource. In order to build NLP tools for low-resource languages, we either need to annotate data (a costly exercise, especially for languages with few native speakers), or find a way to use annotated data in other languages in service to the cause. We refer to the latter techniques as cross-lingual techniques.

In this paper, we address cross-lingual named entity recognition (NER). Prior methods (described in detail in Section 2) depend heavily on limited and expensive resources such as Wikipedia or large parallel text. Concretely, there are about 3800 written languages in the world.[1] Wikipedia exists in about 280 languages, but most versions are too sparse to be useful. Parallel text may be found on an ad-hoc basis for some languages, but it is hardly a general solution. Religious texts, such as the Bible and the Koran, exist in many languages, but the unique domain makes them hard to use. This leaves the vast majority of the world's languages with no general method for NER.

	German	Spanish	Dutch	Avg
Baseline	22.61	45.77	43.10	37.27
Previous SOA	48.12	60.55	61.56	56.74
Cheap Translation	57.53	65.18	66.50	62.65

Table 1: We show dramatic improvement on 3 European languages in a low-resource setting. More detailed results in Table 2 show that this improvement continues to a wide variety of languages. The baseline is a simple direct transfer model. The previous state-of-the-art (SOA) is Tsai et al. (2016)

We propose a simple solution that requires only minimal resources. We translate annotated data in a high-resource language into a low-resource language, using just a lexicon.[2] We refer to this as *cheap translation*, because in general, lexicons are much cheaper and easier to find than parallel text (Mausam et al., 2010).

One of the biggest efforts at gathering lexicons is Panlex (Kamholz et al., 2014), which has lexicons for 10,000 language varieties available to download today. The quality and size of these dic-

[1] https://www.ethnologue.com/enterprise-faq/how-many-languages-world-are-unwritten-0

[2] We use the terms 'lexicon' and 'dictionary' interchangeably.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2536–2545
Copenhagen, Denmark, September 7–11, 2017.

tionaries may vary, but in Section 5.3 we showed that even small dictionaries can give improvements. If there is no dictionary, or if the quality is poor, then the Uyghur case study outlined in Section 6 suggests that effort is best spent in developing a high-quality dictionary, rather than gathering questionable-quality parallel text.

We show that our approach gives non-trivial scores across several languages, and when combined with orthogonal features from Wikipedia, improves on state-of-the-art scores. Table 1 compares a simple direct transfer baseline, the previous state-of-the-art in cross-lingual NER, and our proposed algorithm. For these languages, we beat the baseline by 25.4 points, and the state-of-the-art by 5.9 points. In addition, we found that translating from a language related to the target language gives a further boost. We conclude with a case study of a truly low-resource language, Uyghur, and show a good score, despite having almost no target language resources.

2 Related Work

There are two main branches of work in cross-lingual NLP: projection across parallel data, and language independent methods.

2.1 Projection

Projection methods take a parallel corpus between source and target languages, annotate the source side, and push annotations across learned alignment edges. Assuming that source side annotations are of high quality, success depends largely on the quality of the alignments, which depends, in turn, on the size of the parallel data.

There is work on projection for POS tagging (Yarowsky et al., 2001; Das and Petrov, 2011; Duong et al., 2014), NER (Wang and Manning, 2014; Kim et al., 2012; Ehrmann et al., 2011; Ni and Florian, 2016), mention detection (Zitouni and Florian, 2008), and parsing (Hwa et al., 2005; McDonald et al., 2011).

For NER, the received wisdom is that parallel projection methods work very well, although there is no consensus on the necessary size of the parallel corpus. Most approaches require millions of sentences, with a few exceptions which require thousands. Accordingly, the drawback to this approach is the difficulty of finding any parallel data, let alone millions of sentences. Religious texts (such as the Bible and the Koran) exist in a large number of languages, but the domain is too far removed from typical target domains (such as newswire) to be useful. As a simple example, the Bible contains almost no entities tagged as 'organization'. We approach the problem with the assumption that little to no parallel data is available.

2.2 Language Independent

The second common tool for cross-lingual NLP is to use language independent features. This is often called direct transfer, in the sense that a model is trained on one language and then applied without modification on a dataset in a different language. Lexical or lexical-derived features are typically not used unless there is significant vocabulary overlap between languages.

Täckström et al. (2012) experiments with direct transfer of dependency parsing and NER, and showed that using word cluster features can help, especially if the clusters are forced to conform across languages. The cross-lingual word clusters were induced using large parallel corpora.

Building on this work, Täckström (2012) focuses solely on NER, and includes experiments on self-training and multi-source transfer for NER.

Tsai and Roth (2016) link words and phrases to entries in Wikipedia and use page categories as features. They showed that these wikifier features are strong language independent features. We build on this work, and use these features in our experiments.

Bharadwaj et al. (2016) build a transfer model using phonetic features instead of lexical features. These features are not strictly language-independent, but can work well when languages share vocabulary but with spelling variations, as in the case of Turkish, Uzbek, and Uyghur.

2.3 Others

In a technique similar to ours, Carreras et al. (2003) use Spanish resources for Catalan NER. They translate the features in the weight vector, which has the flavor of a language independent model with the lexical features of a projection model. Our work is a natural extension of this paper, but explores these techniques on many more languages, showing that with some modifications, it has a broad applicability. Further, we experiment with orthogonal features, and with combining multiple source languages to get state of the art results on standard datasets.

Irvine and Callison-Burch (2016) build a machine translation system for low-resource languages by inducing bilingual dictionaries from monolingual texts.

Koehn and Knight (2001) experiment with varying knowledge levels on the task of translating German nouns in a small parallel German-English corpus. A lexicon along with monolingual text can correctly translate 79% of the nouns in the evaluation set. They reach a score of 89% when a parallel corpus is available along with a lexicon, but also comment on the scarcity of parallel corpora.

The main takeaways from the viewpoint of our work are a) word level translation can be effective, at least for nouns, and b) obtaining the correct word pair is more difficult than choosing between a set of options.

3 Our method: Cheap Translation

We create target language training data by translating source data into the target language. It is effectively the same as standard phrase-based statistical machine translation systems (such as MOSES (Koehn et al., 2007)), except that the translation table is not induced from expensive parallel text, but is built from a lexicon, hence the name *cheap translation*.

The entries in our lexicon contain word-to-word translations, as well as word-to-phrase, phrase-to-word, and phrase-to-phrase translations. Entries typically do not have any further information, such as part of speech or sense disambiguation. The standard problems related to ambiguity in language apply: a source language word may have several translations, and several source language words may have the same translation.

We are mostly concerned with the problem of multiple translations of a source language word. For example, in the English-Spanish lexicon, the English word *woman* translates into about 50 different words, with meanings ranging from *woman*, to *female golfer*, to *youth*. Although all candidates might be technically correct, we are interested in the most prominent translation. To estimate this, we gathered cooccurrence counts of each source-target word pair in the lexicon. For Spanish, in the case of *woman*, the most probable translation is *mujer*, because it shows up in other contexts in the dictionary, such as *farm woman* or *young woman*, whereas translations such as *joven* cooccur infrequently with *woman*. We normalize these cooccurrence counts in each candidate set, and call this the *prominence score*.

Algorithm 1 Our translation algorithm

```
Input
    $D_E$: Annotated data in $E$
    $L$: Lexicon between $E$–$F$
Returns
    $D_F$: Annotated data in $F$
for $\forall w_i \in D_E$ do
    $p = w_i w_{i+1} ... w_{i+j}$        ▷ Window of size $j$
    while $p$ not in $L$ and $j \geq 0$ do
        Decrement $j$
        $p = w_i w_{i+1} ... w_{i+j}$
    end while
    if $p$ in $L$ then
        if $|L[p]| > 1$ then
            resolve with LM and prominence
        end if
        $D_F$ add ($L[p]$, labels of $p$)
    else
        $D_F$ add ($w_i$, label of $w_i$)
    end if
    Increment $i$ by length of $p$
end for
```

With these probabilities in hand, we have effectively constructed a phrase translation table. We use a simple greedy decoding method (as shown in Algorithm 1) where options from the lexicon are resolved by a language model multiplied by the prominence score of each option. We use SRILM (Stolcke et al., 2002) trained on Wikipedia (although any large monolingual corpus will do).

During decoding, once we have chosen a candidate, we copy all labels from the source phrase to the target phrase. Since the translation is phrase-to-phrase, we can copy gold labels directly,[3] without worrying about getting good alignments. The result is annotated data in the target language.

Notice that the algorithm allows for no reordering beyond what exists in the phrase-to-phrase entries of the lexicon. Compared to phrase-tables learned from massive parallel corpora, our lexicon-based phrase tables are not large enough or expressive enough for robust reordering. We leave explorations of reordering to future work.

See Figure 1 for a representative example of translation from English to Turkish, with a human translation as reference. There are sin-

[3] We use a standard BIO labeling scheme.

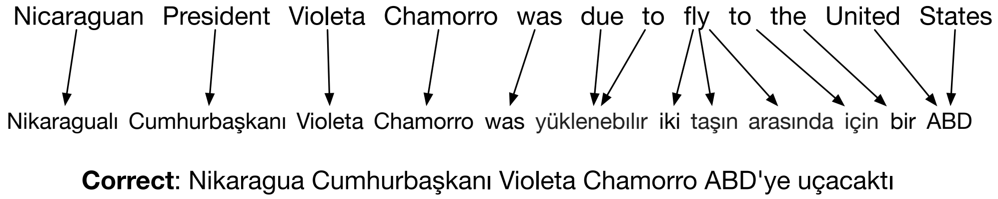

Nicaraguan President Violeta Chamorro was due to fly to the United States
Nikaraguali Cumhurbaşkanı Violeta Chamorro was yüklenebılır iki taşın arasında için bir ABD
Correct: Nikaragua Cumhurbaşkanı Violeta Chamorro ABD'ye uçacaktı

Figure 1: Demonstration of word translation. The top is English, the bottom is Turkish. Lines represent dictionary translations (e.g. *the* translates to *bir*). **Correct** is the correct translation. This illustrates congruence in named entity patterns between languages, as well as some errors we are prone to make.

gle words translated into phrases, named entities copied over verbatim, and phrases translated into single words. Some words are translated correctly (*President* into *Cumhurbaşkanı*) and some incorrectly (*fly* into *iki taşın arasında*, which loosely translates to 'between two stones'). We see ignorance of morphology (seen in translation of United States), and confused word order. But in spite of all these mistakes, the context around the entities, which is what matters for NER, is reasonably well-preserved. Notably, the word *President/Cumhurbaşkanı* is a strong context feature for both LOC (Nicaragua) and PER (Violeta Chamorro) in both languages.

4 Experimental Setup

Before we describe our experiments, we describe some of the tools we used.

4.1 Lexicons

We use lexicons provided by (Rolston and Kirchhoff, 2016), which are harvested from PanLex, Wiktionary, and various other sources. There are 103 lexicons, each mapping between English and a target language. These vary in size from 56K entries to 1.36M entries, as shown in the second row of Table 2. There are also noisy translations. Some entries consist of a single English letter, some are morphological endings, others are misspellings, others are obscure translations of metaphors, and still others are just wrong.

4.2 Datasets

We use data from CoNLL2002/2003 shared tasks (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003). The 4 languages represented are English, German, Spanish, and Dutch. All training is on the train set, and testing is on the test set (TestB). The evaluation metric for all experiments is phrase level F1, as explained in Tjong Kim Sang (2002).

In order to experiment on a broader range of languages, we also use data from the REFLEX (Simpson et al., 2008), and LORELEI projects. From LORELEI, we use Turkish and Hausa [4] From REFLEX, we use Bengali, Tamil, and Yoruba.[5] We use the same set of test documents as used in Tsai et al. (2016).

We also use Hindi and Malayalam data from FIRE 2013,[6] pre-processed to contain only PER, ORG, and LOC tags.

While several of these languages are decidedly high-resource, we limit the resources used in order to show that our techniques will work in truly low-resource settings. In practice, this means generating training data where high-quality manually annotated data is already available, and using dictionaries where translation is available.

4.3 NER Model

In all of our work we use the Illinois NER system (Ratinov and Roth, 2009) with standard features (forms, capitalization, affixes, word prior, word after, etc.) as our base model. We train Brown clusters on the entire Wikipedia dump for any given language (again, any monolingual corpus will do), and include the multilingual gazetteers and wikifier features proposed in Tsai et al. (2016).

5 Experiments

We performed two different sets of experiments: first translating only from English, then translating from additional languages selected to be similar to the target language.

5.1 Translation from English

We start by translating from the highest resourced language in the world, English. We first show that

[4] LDC2014E115,LDC2015E70

[5] LDC2015E13,LDC2015E90,LDC2015E83, LDC2015E91

[6] http://au-kbc.org/nlp/NER-FIRE2013/

our technique gives large improvement over a simple baseline, then combine with orthogonal features, then compare against a ceiling obtained with Google Translate.

Baseline Improvement

To get the baseline, we trained a model on English CoNLL data (train set), and applied the model directly to the target language, mismatching lexical features notwithstanding. We did not use gazetteers in this approach. For the non-Latin script languages, Tamil and Bengali, we transliterated the entire English corpus into the target script. These results are in Table 2, row "Baseline".

In our approach ("Cheap Translation"), for each test language, we translated the English CoNLL data (train set) into that language. The first row of Table 2 shows the coverage of each dictionary. For example, in the case of Spanish, 90.94% of the words were translated into Spanish. This gives an average of 14.6 points F1 improvement over the baseline. This shows that simple translation is surprisingly effective across the board. The improvement is most noticeable for Bengali and Tamil, which are languages with non-Latin script. This mostly shows that the trivial baseline doesn't work across scripts, even with transliteration. Spanish shows the least improvement over the baseline, which may be because English and Spanish are so similar that the baseline is already high.

We found that we needed to normalize the Yoruba text (that is, remove all pronunciation symbols on vowels) in order to make the data less sparse. Since the training data for Bengali and Tamil never shares a script with the test data, we omit using the word surface form as a feature. This is indicated by the † in Table 2. Brown clusters, which implicitly use the word form, are still used.

Wikifier Features

Now we show that our approach is also orthogonal to other approaches, and can be combined with great effect. Wikifier features (Tsai et al., 2016) are obtained by grounding words and phrases to English Wikipedia pages, and using the categories of the linked page as NER features for the surface text. Our approach can be naturally combined with wikifier features. We show results in Table 2, in the row marked 'Cheap Translation+Wiki'.

Using wikifier features improves scores for all 7 languages. Further, for all languages we beat Tsai et al. (2016), with an average of 3.92 points F1 improvement. For the three European languages (Dutch, German, and Spanish), we have an average improvement of 4.8 points F1 over Tsai et al. (2016). This may reflect the fact that English is more closely related to European languages than Indian or African languages, in terms of lexical similarities, word order, and spellings and distribution of named entities. This suggests that it is advantageous to select a source language similar to the target language (by some definition of similar). We explore this hypothesis in Section 5.2.

Google Translate

Since we are performing translation, we compared against a high-quality translation system to get a ceiling. We used Google Translate to translate the English CoNLL training data into the target language, sentence by sentence. We aligned the source-target data using fast align (Dyer et al., 2013), and projected labels across alignments.[7] Since this is high-quality translation, we treat it as an upper bound on our technique, but with the caveat that the alignments can be noisy given the relatively small amount of text. This introduces a source of noise that is not present in our technique, but the loss from this noise is small compared to the gain from the high-quality translation. As with the other approaches, we found that Brown cluster features were an important signal.

Surprisingly, Google Translate beats our basic approach with a margin of only 4.3 points. Despite the naïvete of our approach, we are relatively close to the ceiling. Further, Google Translate is limited to 103 languages, whereas our approach is limited only by available dictionaries. In low-resource settings, such as the one presented in Section 6, Google Translate is not available, but dictionaries are available, although perhaps only by pivoting through a high-resource language.

5.2 Translation from Similar Languages

Observing that English as a source works well for European languages, but not as well for non-European languages, we form a key hypothesis: *cheap translation between similar languages should be better than between different languages.* There are several reasons for this. First, similar languages should have similar word orderings. Since we do no reordering in translation, this means the target text has a better chance of a

[7] Google Translate does not output alignments. If we had an in-house translation system, we could avoid this step.

Method	Dutch	German	Spanish	Turkish	Bengali	Tamil	Yoruba	Avg
Lexicon Coverage	88.01	89.97	90.94	83.80	83.34	73.84	74.60	–
E-L Dict size	961K	1.36M	1.25M	578K	217K	182K	334K	–
Baseline	43.10	22.61	45.77	34.63	6.40	4.60	37.70	27.83
Google Translate ceiling	65.71	56.65	53.65	45.63	37.84	29.11	39.18	46.82
Wiki (Tsai et al., 2016)	61.56	48.12	60.55	47.12	43.27	29.64	36.65	46.70
Cheap Translation	53.94	50.96	51.82	46.37	30.47†	25.91†	37.58	42.43
Cheap Translation+Wiki	63.37	57.23	64.10	51.79	46.28†	33.10†	38.52	50.62
Best Combination	64.48	57.53	65.95	48.50	31.70†	27.63†	39.12	47.84
Best Combination+Wiki	66.50	59.11	65.43	53.44	45.70†	34.90†	40.88	52.28

Table 2: Baseline is naive direct transfer, with no gazetteers. 'Cheap Translation' translates from English into the target. Google Translate translates whole sentences, and does not use gazetteers. 'Cheap Translation+Wiki' incorporates wikifier features. 'Best Combination' uses language combinations from Table 3 for source training data. † denotes that this run does not use word features.

Target	Train lang
Dutch	English, German
German	English, Dutch
Spanish	English, Dutch
Turkish	English, Uzbek
Bengali	English, Hindi
Tamil	English, Malayalam
Yoruba	English, Hausa

Table 3: This shows the language selection results. In each row, we see the target language, and the languages used for training. For example, when testing on Dutch, we train on German and English. These scores came from WALS.

coherent ordering. Second, in case of dictionary misses, vocabulary common between languages will be correct in the target language.

This requires two new resources: annotated data in a similar language S, and a lexicon that maps from S to T, the target language.

Data in other languages

For most target languages, English is not the closest language, and it is likely that there exists an annotated dataset in a closer language. There are annotated datasets available in many languages with a diversity of script and family. We have datasets annotated in about 10 different languages, although more exist.

One caveat is that the source dataset must have a matching tagset with the target dataset. At present, we accept this as a limitation, with the understanding that there is a common set of coarse-grained tags that is widely used (PER, ORG, LOC). We leave further exploration to future work.

Pivoting Lexicons

Although we cannot expect to find lexicons between all pairs of languages, we can usually expect that a language will have at least one lexicon with a high-resource language. Often that language is English. We can use this high-resource language as a pivot to transitively create an S–T dictionary, although perhaps with some loss of precision.

Assume we want a Turkish-Bengali lexicon and we have only English-Bengali and English-Turkish lexicons. We collect all English words that appear in both dictionaries. Each such English word has two sets of candidate translations, one set in Turkish, the other in Bengali. To create transitive pairs, we take the Cartesian product of these two sets of candidate translations. This will create too many entries, some of which will be incorrect, but usually the correct entry is there.

Notice also that the resulting dictionary contains only those English words that appear in both original dictionaries. If either of the original dictionaries is small, the result will be smaller still.

Source Selection and Combination

To choose a related source language, we used syntactic features of languages retrieved from the World Atlas of Language Structures (WALS) (Dryer and Haspelmath, 2013). Each language is represented as a binary vector with each index indicating presence or absence of a syntactic feature in that language. We used the feature set called *syntax_knn*, which includes 103 syntactic features, such as *subject before verb*, and *possessive suffix*, and uses k-Nearest Neighbors to predict missing values. We measure similarity as cosine distance between language vectors.

In the absence of criteria for a similarity cutoff, we chose to include only the top most similar language as source for that target language. The results of this similarity calculation are shown in Table 3. For example, when the target language is Dutch, German is the closest. We also included English in the training, as the highest resource language, and with the highest quality dictionaries.

Results

Our results are in Table 2, in the row named 'Best Combination'. The average over all languages surpasses the English-source average by 5.4 points, and also beats (Tsai et al., 2016). We also add wikifier features, and report results in row 'Best Combination+Wiki.' This shows improvement on all but Spanish, with an average improvement of 5.58 points F1 over Tsai et al. (2016). To the best of our knowledge, these are the best cross-language settings scores for all these datasets.

While these scores are lower than those seen on typical NER tasks (70-90% F1), we emphasize first that cross-lingual scores will necessarily be much lower than monolingual scores, and second that these are the best available given the setup.

5.3 Dictionary Ablation

The most expensive resource we require is a lexicon. In this section, we briefly explore what effect the size of the lexicon has on the end result. Using Turkish, we vary the size of the dictionary by randomly removing entries. The sizes vary from no entries to full dictionary (rows 'Baseline' and 'Cheap Translation' in Table 2, respectively), with several gradations in the middle. With each reduced dictionary, we translate from English to generate Turkish training data as in Section 5.1. As before, we train an NER model on the generated data, and test on the Turkish test data. Results are shown in Figure 2.

Interestingly, we see improvement over the baseline even with only 500 entries. This improvement continues until 125K entries. It is important to note that only a small number of dictionary entries – words that typically show up in the contexts of named entities, such as *president*, *university* or *town* – are likely to be useful. The larger the dictionary, the more likely these valuable entries are present. Further, our random removal process may unfairly prioritize less common words, compared to a manually compiled dictionary which would prioritize common words. It is likely that a small but carefully constructed manual dictionary could have a large impact.

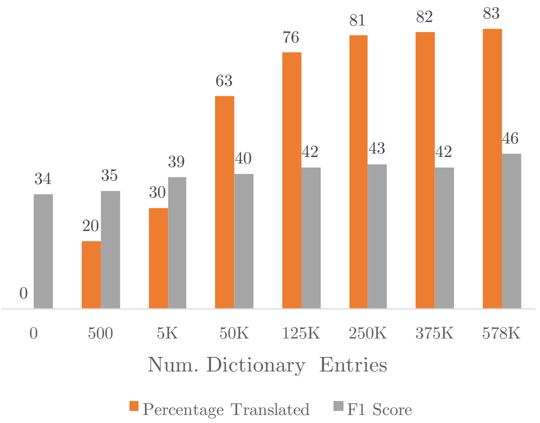


Figure 2: Effect of dictionary size on F1 score for Turkish. Each column is an experiment with a randomly reduced dictionary. The orange bars represent how much of the corpus is translated.

6 Case study: Uyghur

We have shown in the previous sections that our method is effective across a variety of languages. However, all of the tested languages have some resources, most notably, Google Translate and reasonably sized Wikipedias. In this section, we show that our methods hold up on a truly low-resource language, Uyhgur.

Uyghur is a language native to northwest China, with about 25 million speakers.[8] It is a Turkic language, and is related most closely to Uzbek, although it uses an Arabic writing system. Uyghur is not supported by Google Translate, and the Uyghur Wikipedia has less than 3,000 articles. In contrast, the smallest Wikipedia size language in our test set is Yoruba, with 30K articles. Because of the small Wikipedia size, we do not use any wikifier features.

We did this work as part of the NIST LoReHLT evaluation in the summer of 2016. The official evaluation scores were calculated over a set of 4500 Uyghur documents. Each team was given the unannotated version of those documents, with the task being to submit annotations on that set. Our official scores are reported in Table 4, and compared with Bharadwaj et al. (2016).

After the evaluation, NIST released 199 of the annotated evaluation documents, called the *unse-*

[8] https://en.wikipedia.org/wiki/Uyghur_language, accessed July 21, 2017

Method	All	Unseq.
Bharadwaj et al. (2016)	51.2	–
Ours	55.6	51.32

Table 4: F1 scores of official submissions in LoReHLT16. The numbers in the "All" column are the scores on the entire evaluation data reported from NIST. We evaluate our submissions on the unsequestered data in order to compare with the results in Table 5.

Method	F1
Monolingual	69.92
Standard Translation	
Train: English	27.20
Train: Turkish	33.02
Train: Uzbek	27.88
Language Specific (stemmed)	
Train: English	30.84
Train: Turkish	40.04
Train: Uzbek	40.15
Induced dictionaries	43.46
Manual annotations	42.51
All Lang. Spec.	51.32

Table 5: F1 scores for Uyghur. Monolingual scores are on the 41 document test set. All other scores are on the full unsequestered data. We omit forms or gazetteers but use Brown clusters. 'Standard Translation' uses the same resources as the scores in Table 2 (e.g. without stemming)

questered set. In this section, we will drill into the various methods we used to build the transfer model, and report finer-grained results using the unsequestered set.

The following are some of the language-specific techniques we employed.

- **Dictionary** The dictionary provided for Uyghur from Rolston and Kirchhoff (2016) had only 5K entries, so we augmented this with the dictionary provided in the LORELEI evaluation, which resulted in 116K entries.

- **Name Substitution** As with Bengali and Tamil, very few names were translated. We found transliteration models were too noisy, so instead, we gathered a list of gazetteers from Uyghur Wikipedia, categorized by tag type (PER, LOC, GPE, ORG). Upon encountering an untranslatable NE, we replaced it with a randomly selected NE from the gazetteer list corresponding to the tag. This led to improbable sentences like *John Kerry has joined the Baskin Robbins*, but it meant that NEs were fluent in the target text.

- **Stemming** We created a very simple stemmer for Uyghur. This consists of 45 common suffixes sorted longest first. For each Uyghur word in a corpus, we removed all possible suffixes (Uyghur words can take multiple suffixes). We stemmed all train and test data.

We report results in Table 5. The first row is from a monolingual model trained on 158 documents in the unsequestered set, and tested on the remaining 41. All other rows test on the complete unsequestered set. The next section, 'Standard Translation', refers to the method described above. Notably, we do not use stemming for train or test data here. As with Bengali and Tamil, we omit form features.

We translate from English, Turkish, and Uzbek, which are the closest languages predicted by WALS. Next, we incorporated language specific methods. The scores we get from training on English, Turkish and Uzbek all go up because the stemming makes the features more dense. Next we generated dictionaries using observations over Uyghur and Uzbek, and we used non-native speakers to annotate Uyghur data.

6.1 Language Specific Dictionary Induction

We began by romanizing Uyghur text into the Uyghur Latin alphabet (ULY) so we could read it. We noticed that Uzbek and Uyghur are very similar, sharing a sizable amount of vocabulary, and several morphological rules. However, while there is a shared vocabulary, the words are usually spelled slightly differently. For example, the word for "southern" is "janubiy" in Uzbek and "jenubiy" in Uyghur.

We tried several ideas for gathering a mapping for this shared vocabulary: manual mapping, edit-distance mapping, and cross-lingual CCA with word vectors.

Manual mapping: We manually translated about 100 words often found around entities, such as *president*, and *university*

Edit-distance mapping: We gathered (Uyghur, Uzbek) word pairs with low-edit distance, using a modified edit-distance algorithm that allowed certain substitutions at zero cost. For example, this discovered such pairs as *pokistan-pakistan* and *telegraph-télégraf*.

Cross-lingual CCA with word vectors: We projected Uyghur and Uzbek monolingual vectors into a shared semantic space, using CCA (Faruqui and Dyer, 2014). We used the list of low edit-distance word pairs as the dictionary for the projection. Once all the vectors were in the same space, we found the closest Uyghur word to each Uzbek word.

6.2 Results

Scores are in Table 5. Interestingly, the language specific methods evaluated individually did not improve much over the generic word translation methods. But with all language specific methods combined, 'All Lang. Spec.', the score increased by nearly 10 points, suggesting that the different training data covers many angles.

To the best of our knowledge, there are no published scores on the unsequestered data set. Our best score is comparable to the score of our evaluation submission on the unsequestered dataset.

7 Conclusion

We have shown a novel cross-lingual method for generating NER data that gives significant improvement over state-of-the-art on standard datasets. The method benefits from annotated data in many languages, combines well with orthogonal features, and works even when resources are virtually nil. The simplicity and minimal use of resources makes this approach more portable than all previous approaches.

Acknowledgments

This material is based on research sponsored by the US Defense Advanced Research Projects Agency (DARPA) under agreement numbers FA8750-13-2-0008 and HR0011-15-2-0025. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

Akash Bharadwaj, David R. Mortensen, Chris Dyer, and Jaime G. Carbonell. 2016. Phonologically aware neural model for named entity recognition in low resource transfer settings. In *EMNLP*.

Xavier Carreras, Lluís Màrquez, and Lluís Padró. 2003. Named entity recognition for catalan using spanish resources. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 43–50. Association for Computational Linguistics.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Portland, OR. Association for Computational Linguistics.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Long Duong, Trevor Cohn, Karin Verspoor, Steven Bird, and Paul Cook. 2014. What can we get from 1000 tokens? A case study of multilingual POS tagging for resource-poor languages. In *EMNLP*, pages 886–897. Citeseer.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. Association for Computational Linguistics.

Maud Ehrmann, Marco Turchi, and Ralf Steinberger. 2011. Building a multilingual named entity-annotated corpus using annotation projection. In *RANLP*.

M. Faruqui and C. Dyer. 2014. Improving vector space word representations using multilingual correlation. Association for Computational Linguistics.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara I. Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11:311–325.

Ann Irvine and Chris Callison-Burch. 2016. End-to-end statistical machine translation with zero or small parallel texts. *Natural Language Engineering*, 22:517–548.

David Kamholz, Jonathan Pool, and Susan M Colowick. 2014. Panlex: Building a resource for panlingual lexical translation. In *LREC*, pages 3145–3150.

Sungchul Kim, Kristina Toutanova, and Hwanjo Yu. 2012. Multilingual named entity recognition using parallel data and metadata from Wikipedia. In *ACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Philipp Koehn and Kevin Knight. 2001. Knowledge sources for word-level translation models.

Stephen Mausam, Soderland, Oren Etzioni, Daniel S Weld, Kobi Reiter, Michael Skinner, Marcus Sammer, Jeff Bilmes, et al. 2010. Panlingual lexical translation via probabilistic inference. *Artificial Intelligence*, 174(9-10):619–637.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 62–72, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Jian Ni and Radu Florian. 2016. Improving multilingual named entity recognition with wikipedia entity type mapping.

L. Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*.

Leanne Rolston and Katrin Kirchhoff. 2016. Collection of bilingual data for lexicon transfer learning. Technical report, University of Washington.

Heather Simpson, Christopher Cieri, Kazuaki Maeda, Kathryn Baker, and Boyan Onyshkevych. 2008. Human language technology resources for less commonly taught languages: Lessons learned toward creation of basic language resources. *Collaboration: interoperability between people in the creation of language resources for less-resourced languages*, page 7.

Andreas Stolcke et al. 2002. Srilm-an extensible language modeling toolkit. In *Interspeech*, volume 2002, page 2002.

Oscar Täckström. 2012. Nudging the envelope of direct transfer methods for multilingual named entity recognition. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 55–63. Association for Computational Linguistics.

Oscar Täckström, Ryan T. McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *NAACL*.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158. Taipei, Taiwan.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. *CoNLL 2016*, page 219.

Chen-Tse Tsai and Dan Roth. 2016. Cross-lingual wikification using multilingual embeddings. In *NAACL*.

Mengqiu Wang and Christopher D Manning. 2014. Cross-lingual projected expectation regularization for weakly supervised learning. In *TACL*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, pages 1–8. Association for Computational Linguistics.

Imed Zitouni and Radu Florian. 2008. Mention detection crossing the language barrier. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 600–609. Association for Computational Linguistics.

Cross-Lingual Induction and Transfer of Verb Classes Based on Word Vector Space Specialisation

Ivan Vulić[1] , **Nikola Mrkšić**[2] and **Anna Korhonen**[1]
[1] Language Technology Lab, University of Cambridge, UK
[2] Dialogue Systems Group, University of Cambridge, UK
{iv250,nm480,alk23}@cam.ac.uk

Abstract

Existing approaches to automatic VerbNet-style verb classification are heavily dependent on feature engineering and therefore limited to languages with mature NLP pipelines. In this work, we propose a novel cross-lingual transfer method for inducing VerbNets for multiple languages. To the best of our knowledge, this is the first study which demonstrates how the architectures for learning word embeddings can be applied to this challenging syntactic-semantic task. Our method uses cross-lingual translation pairs to tie each of the six target languages into a bilingual vector space with English, jointly specialising the representations to encode the relational information from English VerbNet. A standard clustering algorithm is then run on top of the VerbNet-specialised representations, using vector dimensions as features for learning verb classes. Our results show that the proposed cross-lingual transfer approach sets new state-of-the-art verb classification performance across all six target languages explored in this work.

1 Introduction

Playing a key role in conveying the meaning of a sentence, verbs are famously complex. They display a wide range of syntactic-semantic behaviour, expressing the semantics of an event as well as relational information among its participants (Jackendoff, 1972; Gruber, 1976; Levin, 1993, inter alia).

Lexical resources which capture the variability of verbs are instrumental for many Natural Language Processing (NLP) applications. One of the richest verb resources currently available for English is VerbNet (Kipper et al., 2000; Kipper, 2005).[1] Based on the work of Levin (1993), this largely hand-crafted taxonomy organises verbs into classes on the basis of their shared syntactic-semantic behaviour. Providing a useful level of generalisation for many NLP tasks, VerbNet has been used to support semantic role labelling (Swier and Stevenson, 2004; Giuglea and Moschitti, 2006), semantic parsing (Shi and Mihalcea, 2005), word sense disambiguation (Brown et al., 2011), discourse parsing (Subba and Di Eugenio, 2009), information extraction (Mausam et al., 2012), text mining applications (Lippincott et al., 2013; Rimell et al., 2013), research into human language acquisition (Korhonen, 2010), and other tasks.

This benefit for English NLP has motivated the development of VerbNets for languages such as Spanish and Catalan (Aparicio et al., 2008), Czech (Pala and Horák, 2008), and Mandarin (Liu and Chiang, 2008). However, end-to-end manual resource development using Levin's methodology is extremely time consuming, even when supported by translations of English VerbNet classes to other languages (Sun et al., 2010; Scarton et al., 2014). Approaches which aim to learn verb classes automatically offer an attractive alternative. However, existing methods rely on carefully engineered features that are extracted using sophisticated language-specific resources (Joanis et al., 2008; Sun et al., 2010; Falk et al., 2012, i.a.), ranging from accurate parsers to pre-compiled subcategorisation frames (Schulte im Walde, 2006; Li and Brew, 2008; Messiant, 2008). Such methods are limited to a small set of resource-rich languages.

It has been argued that VerbNet-style classification has a strong cross-lingual element (Jackendoff, 1992; Levin, 1993). In support of this argument, Majewska et al. (2017) have shown that English VerbNet has high translatability across different,

[1] http://verbs.colorado.edu/∼mpalmer/projects/verbnet.html

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2546–2558
Copenhagen, Denmark, September 7–11, 2017.

even typologically diverse languages. Based on this finding, we propose an automatic approach which exploits readily available annotations for English to facilitate efficient, large-scale development of VerbNets for a wide set of target languages.

Recently, unsupervised methods for inducing distributed word vector space representations or *word embeddings* (Mikolov et al., 2013a) have been successfully applied to a plethora of NLP tasks (Turian et al., 2010; Collobert et al., 2011; Baroni et al., 2014, i.a.). These methods offer an elegant way to learn directly from large corpora, bypassing the feature engineering step and the dependence on mature NLP pipelines (e.g., POS taggers, parsers, extraction of subcategorisation frames). In this work, we demonstrate how these models can be used to support automatic verb class induction. Moreover, we show that these models offer the means to exploit inherent cross-lingual links in VerbNet-style classification in order to guide the development of new classifications for resource-lean languages. To the best of our knowledge, this proposition has not been investigated in previous work.

There has been little work on assessing the suitability of embeddings for capturing rich syntactic-semantic phenomena. One challenge is their reliance on the distributional hypothesis (Harris, 1954), which coalesces fine-grained syntactic-semantic relations between words into a broad relation of semantic relatedness (e.g., *coffee:cup*) (Hill et al., 2015; Kiela et al., 2015). This property has an adverse effect when word embeddings are used in downstream tasks such as spoken language understanding (Kim et al., 2016a,b) or dialogue state tracking (Mrkšić et al., 2016, 2017a). It could have a similar effect on verb classification, which relies on the similarity in syntactic-semantic properties of verbs within a class. In summary, we explore three important questions in this paper:

(Q1) Given their fundamental dependence on the distributional hypothesis, to what extent can unsupervised methods for inducing vector spaces facilitate the automatic induction of VerbNet-style verb classes across different languages?

(Q2) Can one boost verb classification for lower-resource languages by exploiting general-purpose cross-lingual resources such as BabelNet (Navigli and Ponzetto, 2012; Ehrmann et al., 2014) or bilingual dictionaries such as PanLex (Kamholz et al., 2014) to construct better word vector spaces for these languages?

(Q3) Based on the stipulated cross-linguistic validity of VerbNet-style classification, can one exploit rich sets of readily available annotations in one language (e.g., the full English VerbNet) to automatically bootstrap the creation of VerbNets for other languages? In other words, is it possible to exploit a cross-lingual vector space to transfer VerbNet knowledge from a resource-rich to a resource-lean language?

To investigate Q1, we induce standard distributional vector spaces (Mikolov et al., 2013b; Levy and Goldberg, 2014) from large monolingual corpora in English and six target languages. As expected, the results obtained with this straightforward approach show positive trends, but at the same time reveal its limitations for all the languages involved. Therefore, the focus of our work shifts to Q2 and Q3. The problem of inducing VerbNet-oriented embeddings is framed as vector space specialisation using the available external resources: BabelNet or PanLex, and (English) VerbNet. Formalised as an instance of post-processing *semantic specialisation* approaches (Faruqui et al., 2015; Mrkšić et al., 2016), our procedure is steered by two sets of linguistic constraints: **1)** cross-lingual (translation) links between languages extracted from BabelNet (targeting Q2); and **2)** the available VerbNet annotations for a resource-rich language. The two sets of constraints jointly target Q3.

The main goal of vector space specialisation is to pull examples standing in desirable relations, as described by the constraints, closer together in the transformed vector space. The specialisation process can capitalise on the knowledge of VerbNet relations in the *source* language (English) by using translation pairs to transfer that knowledge to each of the *target* languages. By constructing shared bilingual vector spaces, our method facilitates the transfer of semantic relations derived from VerbNet to the vector spaces of resource-lean target languages. This idea is illustrated by Fig. 1.

Our results indicate that cross-lingual connections yield improved verb classes across all six target languages (thus answering Q2). Moreover, a consistent and significant boost in verb classification performance is achieved by propagating the VerbNet-style information from the source language (English) to any other target language (e.g., Italian, Croatian, Polish, Finnish) for which no VerbNet-style information is available during the

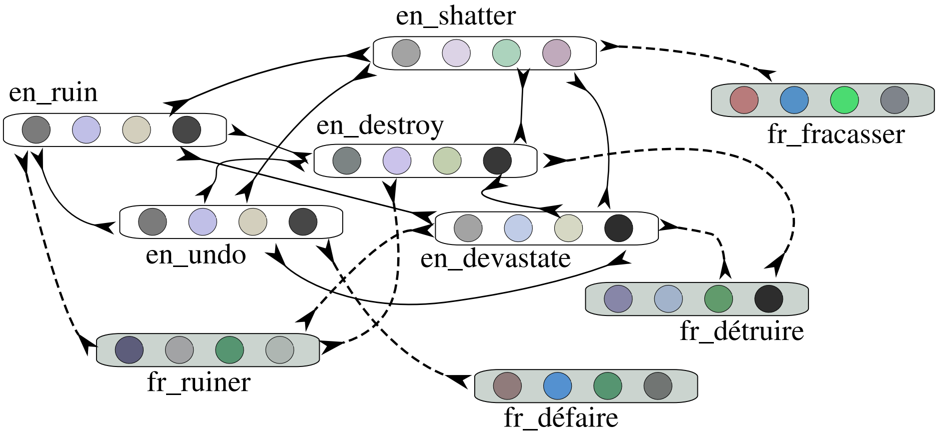


Figure 1: Transferring VerbNet information from a resource-rich to a resource-lean language through a word vector space: an *English*→ *French* toy example. Representations of words described by two types of ATTRACT constraints are pulled closer together in the joint vector space. (1) Monolingual pairwise constraints in English (e.g., *(en_ruin, en_shatter), (en_destroy, en_undo)*) reflect the EN VerbNet structure and are generated from the readily available verb classification in English (solid lines). They are used to specialise the distributional EN vector subspace for the VerbNet relation. (2) Cross-lingual English-French pairwise constraints (extracted from BabelNet) describe cross-lingual synonyms (i.e., translation links) such as *(en_ruin, fr_ruiner) or (en_shatter, fr_fracasser)* (dashed lines). The post-processing fine-tuning specialisation procedure based on (1) and (2) effectively transforms the initial distributional French vector subspace to also emphasise the VerbNet-style structure, facilitating the induction of verb classes in French.

fine-tuning process (thus answering Q3).[2] We report state-of-the-art verb classification performance for all six languages in our experiments. For instance, we improve the state-of-the-art F-1 score from prior work from 0.55 to 0.79 for French, and from 0.43 to 0.74 for Brazilian Portuguese.

2 Methodology: Specialising for VerbNet

Motivation: Verb Classes and VerbNet VerbNet is a hierarchical, domain-independent, broad-coverage verb lexicon based on Levin's classification and taxonomy of English (EN) verbs (Levin, 1993; Kipper, 2005). Verbs are grouped into classes (e.g. the class PUT-9.1 for verbs such as *place*, *position*, *insert*, and *arrange*) based on their shared meaning components and syntactic behaviour, defined in terms of their participation in *diathesis alternations*, i.e., alternating verb frames that are related with the same or similar meaning. VerbNet extends and refines Levin's classification, providing more fine-grained syntactic and semantic information for individual classes. Each VerbNet class is characterised by its member verbs, syntactic frames, semantic predicates and typical verb arguments.[3] The current version of VerbNet (v3.2) contains 8,537 distinct English verbs grouped into 273 VerbNet main classes.

The inter-relatedness of syntactic behaviour and meaning of verbs is not limited to English (Levin, 1993). The basic meaning components underlying verb classes are said to be *cross-linguistically valid* (Jackendoff, 1992; Merlo et al., 2002)[4] and therefore the classification has a strong cross-lingual dimension. A recent investigation of Majewska et al. (2017) show that it is possible to manually translate VerbNet classes and class members to different, typologically diverse languages with high accuracy.

The practical usefulness of VerbNet style classification both within and across languages has been limited by the fact that few languages boast resources similar to the English VerbNet. Some VerbNets have been developed completely manually from scratch, aiming to capture properties specific to the language in question, e.g., the resources for Spanish and Catalan (Aparicio et al., 2008),

[2] On a high level, we demonstrate that a constraints-driven fine-tuning framework can specialise word embeddings to reflect VerbNet-style relations which rely not only on verb sense similarity, but also on similarity in syntax, selectional preferences, and diathesis alternations.

[3] The usefulness of VerbNet is further accentuated by available mappings (Loper et al., 2007) to a number of other verb resources such as WordNet (Fellbaum, 1998), FrameNet (Baker et al., 1998), and PropBank (Palmer et al., 2005).

[4] For example, Levin (1993) notes that verbs in Warlpiri manifest analogous behavior to English with respect to the conative alternation. In another example, Polish verbs have the same patterns as EN verbs in terms of the middle construction.

Czech (Pala and Horák, 2008), and Mandarin (Liu and Chiang, 2008). Other VerbNets were created semi-automatically, with the help of other lexical resources, e.g., for French (Pradet et al., 2014) and Brazilian Portuguese (Scarton and Aluısio, 2012). These approaches involved substantial amounts of specialised linguistic and translation work. Finally, automatic methods have been developed, e.g., for French (Sun et al., 2010; Falk et al., 2012) and Brazilian Portuguese (Scarton et al., 2014), with insufficient accuracy (as emphasised in Sect. 4). Until now, work in this area has been limited to a small number of languages, due to the large requirements in terms of human input and/or the availability of mature NLP pipelines which exist only for a few resource-rich languages (e.g., English, German).

In this work, we propose a novel, fully automated approach for inducing VerbNets for multiple languages - one based on cross-lingual transfer. Unlike earlier approaches, our method does not require any parsed data or manual annotations for the target language. It encodes the cross-linguistic validity of Levin-style verb classifications into the vector-space specialisation framework (Sect. 2.1) driven by linguistic constraints. A standard clustering algorithm is then run on top of the VerbNet-specialised representations using vector dimensions as features to learn verb clusters (Sect. 2.2). Our approach attains state-of-the-art verb classification performance across all six target languages.

2.1 Vector Space Specialisation

Specialisation Model Our departure point is a state-of-the-art specialisation model for fine-tuning vector spaces termed PARAGRAM (Wieting et al., 2015).[5] The PARAGRAM procedure injects similarity constraints between word pairs in order to make their vector space representations more similar; we term these the ATTRACT constraints. Let $V = V_s \sqcup V_t$ be the vocabulary consisting of the source language and target language vocabularies V_s and V_t, respectively. Let C be the set of word pairs standing in desirable lexical relations; these include: **1)** verb pairs from the same VerbNet class (e.g. (*en_transport, en_transfer*) from verb class SEND-11.1); and **2)** the cross-lingual synonymy pairs (e.g. (*en_peace, fi_rauha*)). Given the initial distributional space and collections of such ATTRACT pairs C, the model gradually modifies the space to bring the designated word vectors closer together, working in mini-batches of size k. The method's cost function can be expressed as:

$$O(\mathcal{B}_C) = O_C(\mathcal{B}_C) + R(\mathcal{B}_C) \qquad (1)$$

The first term of the method's cost function (i.e., $\mathcal{O}_C$) pulls the ATTRACT examples $(x_l, x_r) \in C$ closer together (see Fig. 1 for an illustration). B_C refers to the current mini-batch of ATTRACT constraints. This term is expressed as follows:

$$\begin{aligned} O_C(\mathcal{B}_C) \;=\; \sum_{(x_l,x_r)\in\mathcal{B}_C} & \big[\, \tau\left(\delta_{att} + \mathbf{x}_l\mathbf{t}_l - \mathbf{x}_l\mathbf{x}_r\right) \\ & + \tau\left(\delta_{att} + \mathbf{x}_r\mathbf{t}_r - \mathbf{x}_l\mathbf{x}_r\right) \big] \qquad (2) \end{aligned}$$

$\tau(x) = \max(0, x)$ is the standard rectified linear unit or the hinge loss function (Tsochantaridis et al., 2004; Nair and Hinton, 2010). δ_{att} is the "attract" margin: it determines how much vectors of words from ATTRACT constraints should be closer to each other than to their negative examples. The negative example $\mathbf{t}_i$ for each word x_i in any ATTRACT pair is always the vector closest to $\mathbf{x}_i$ taken from the pairs in the current mini-batch, distinct from the other word paired with x_i, and x_i itself.[6]

The second $R(\mathcal{B}_C)$ term is the regularisation which aims to retain the semantic information encoded in the initial distributional space as long as this information does not contradict the used ATTRACT constraints. Let $\mathbf{x}_i^{init}$ refer to the initial distributional vector of the word x_i and let $\mathcal{V}(\mathcal{B}_C)$ be the set of all word vectors present in the given mini-batch. If λ_{reg} denotes the L2 regularisation constant, this term can be expressed as:

$$R(\mathcal{B}_C) = \sum_{\mathbf{x}_i \in \mathcal{V}(\mathcal{B}_C)} \lambda_{reg} \left\| \mathbf{x}_i^{init} - \mathbf{x}_i \right\|_2 \qquad (3)$$

Linguistic Constraints: Transferring VerbNet-Style Knowledge The fine-tuning procedure effectively blends the knowledge from external resources (i.e., the input ATTRACT set of constraints) with distributional information extracted directly from large corpora. We show how to propagate annotations from a knowledge source such as VerbNet from source to target by combining two types of constraints within the specialisation framework:

[5] The original PARAGRAM model as well as other fine-tuning models from prior work inject pairwise linguistic constraints into existing vector spaces in order to improve their ability to capture semantic similarity/paraphrasability. In this work, we demonstrate that the same generic specialisation framework can be used to transform vector spaces to capture other types of lexical relations such as VerbNet relations.

[6] Effectively, this term forces word pairs from the in-batch ATTRACT constraints to be closer to one another than to any other word in the current mini-batch.

LEARN-14	BREAK-45.1	ACCEPT-77
(learn, study)	(break, dissolve)	(accept, understand)
(learn, relearn)	(crash, crush)	(accept, reject)
(read, study)	(shatter, split)	(repent, rue)
(cram, relearn)	(break, rip)	(reject, discourage)
(read, assimilate)	(crack, smash)	(encourage, discourage)
(learn, assimilate)	(shred, splinter)	(reject, discourage)
(read, relearn)	(snap, tear)	(disprefer, understand)

Table 1: Example pairwise ATTRACT constraints extracted from three VerbNet classes in English.

a) cross-lingual (translation) links between languages, and **b)** available VerbNet annotations in a resource-rich language transformed into pairwise constraints. Cross-lingual constraints such as *(pl_wojna, it_guerra)* are extracted from BabelNet (Navigli and Ponzetto, 2012), a large-scale resource which groups words into cross-lingual BABEL synsets (and is currently available for 271 languages). The wide and steadily growing coverage of languages in BabelNet means that our proposed framework promises to support the transfer of VerbNet-style information to numerous target languages (with increasingly high accuracy).

To establish that the proposed transfer approach is in fact independent of the chosen cross-lingual information source, we also experiment with another cross-lingual dictionary: PanLex (Kamholz et al., 2014), which was used in prior work on cross-lingual word vector spaces (Duong et al., 2016; Adams et al., 2017). This dictionary currently covers around 1,300 language varieties with over 12 million expressions, thus offering support also for low-resource transfer settings.[7]

VerbNet constraints are extracted from the English VerbNet class structure in a straightforward manner. For each class VN_i from the 273 VerbNet classes, we simply take the set of all n_i verbs $CL_i = \{v_{1,i}, v_{2,i}, \ldots, v_{n_i,i}\}$ associated with that class, including its subclasses, and generate all unique pairs (v_k, v_l) so that $v_k, v_l \in CL_i$ and $v_k \neq v_l$. Example VerbNet pairwise constraints are shown in Tab. 1. Note that VerbNet classes in practice contain verb instances standing in a variety of lexical relations, including synonyms, antonyms, troponyms, hypernyms, and the class membership is determined on the basis of connections between the syntactic patterns and the underlying semantic relations (Kipper et al., 2006, 2008).

2.2 Clustering Algorithm

Given the initial distributional or specialised collection of target language vectors $\mathbf{V_t}$, we apply an off-the-shelf clustering algorithm on top of these vectors in order to group verbs into classes. Following prior work (Brew and Schulte im Walde, 2002; Sun and Korhonen, 2009; Sun et al., 2010), we employ the MNCut spectral clustering algorithm (Meila and Shi, 2001), which has wide applicability in similar NLP tasks which involve high-dimensional feature spaces (Chen et al., 2006; von Luxburg, 2007; Scarton et al., 2014, i.a.). Again, following prior work (Sun et al., 2010, 2013), we estimate the number of clusters K_{Clust} using the self-tuning method of Zelnik-Manor and Perona (2004). This algorithm finds the optimal number by minimising a cost function based on the eigenvector structure of the word similarity matrix. We refer the reader to the relevant literature for further details.

3 Experimental Setup

Languages We experiment with six target languages: French (FR), Brazilian Portuguese (PT), Italian (IT), Polish (PL), Croatian (HR), and Finnish (FI). All statistics regarding the source and size of training and test data, and linguistic constraints for each target language are summarised in Tab. 2.

Automatic approaches to verb class induction have been tried out in prior work for FR and PT. To the best of our knowledge, our cross-lingual study is the first aiming to generalise an automatic induction method to more languages using an underlying methodology which is language-pair independent.

Initial Vector Space: Training Data and Setup All target language vectors were trained on large monolingual running text using the same setup: 300-dimensional word vectors, the frequency cut-off set to 100, bag-of-words (BOW) contexts, and the window size of 2 (Levy and Goldberg, 2014; Schwartz et al., 2016). All tokens were lowercased, and all numbers were converted to a placeholder symbol <NUM>.[8] FR and IT word vectors were trained on the standard frWaC and itWaC corpora (Baroni et al., 2009), and vectors for other target languages were trained on the corpora of similar style and size: HR vectors were trained on the hrWaC corpus (Ljubešić and Klubička, 2014), PT

[7] Similar to BabelNet, the translations in PanLex were derived from various sources such as glossaries, dictionaries, and automatic inference from other languages. This results in a high-coverage lexicon containing a certain amount of noise.

[8] Other SGNS parameters were also set to standard values (Baroni et al., 2014; Vulić and Korhonen, 2016): 15 epochs, 15 negative samples, global learning rate: .025, subsampling rate: $1e-4$. Similar trends in results persist with $d = 100, 500$.

	French:FR	Portuguese:PT	Italian:IT	Polish:PL	Croatian:HR	Finnish:FI
Training						
Corpus	frWaC	brWaC	itWaC	Araneum	hrWaC	fiWaC
Corpus size (in tokens)	1.6B	2.7B	2B	1.2B	1.4B	1.7B
Vocabulary size	242,611	257,310	178,965	373,882	396,103	448,863
Constraints						
# monolingual VerbNet-EN	220,052	220,052	220,052	220,052	220,052	220,052
# mono TARGET (MONO-SYN)	428,329	292,937	362,452	423,711	209,626	377,548
# cross-ling EN-TARGET (BNet)	310,410	245,354	258,102	219,614	160,963	284,167
# cross-ling EN-TARGET (PLex)	225,819	187,386	216,574	154,159	201,329	257,106
Test						
# Verbs (# Classes)	169 (16)	660 (17)	177 (17)	258 (17)	277 (17)	201 (17)
Coverage of test instances	94.1%	95.5%	96.6%	93.4%	98.2%	84.6%

Table 2: Statistics of the experimental setup for each target language: training/test data and constraints. *Coverage* refers to the percentage of test verbs represented in the target language vocabularies.

vectors on ptWaC (Wagner Filho et al., 2016), FI vectors on fiWaC (Ljubešić et al., 2016), and PL vectors on the Araneum Polonicum Maius Web corpus (Benko, 2014). Note that we do not utilise any VerbNet-specific knowledge in the target language to induce and further specialise these word vectors.

Source EN vectors were taken directly from the work of Levy and Goldberg (2014): they are trained with SGNS on the cleaned and tokenised Polyglot Wikipedia (Al-Rfou et al., 2013) containing $\sim$75M sentences, $\sim$1.7B word tokens and a vocabulary of $\sim$180k words after lowercasing and frequency cut-off. To measure the importance of the starting source language space as well as to test if syntactic knowledge on the source side may be propagated to the target space, we test two variant EN vector spaces: SGNS with (a) BOW contexts and the window size 2 (SGNS-BOW2); and (b) dependency-based contexts (SGNS-DEPS) (Padó and Lapata, 2007; Levy and Goldberg, 2014).

Linguistic Constraints We experiment with the following constraint types: (a) monolingual synonymy constraints in each target language extracted from BabelNet (*Mono-Syn*); (b) cross-lingual EN-TARGET constraints from BabelNet; (c) cross-lingual EN-TARGET constraints plus EN VerbNet constraints (see Sect. 2.1 and Fig. 1). Unless stated otherwise, we use BabelNet as the default source of cross-lingual constraints for (b) and (c).

Vector Space Specialisation The PARAGRAM model's parameters are adopted directly from prior work (Wieting et al., 2015) without any additional fine-tuning: $\delta_{att} = 0.6$, $\lambda_{reg} = 10^{-9}$, $k = 50$. We train for 5 epochs without early stopping using AdaGrad (Duchi et al., 2011). PARAGRAM is in fact a special case of the more general ATTRACT-REPEL specialisation framework (Mrkšić et al., 2017b): we use this more recent and more efficient TensorFlow implementation of the model in all experiments.[9]

Test Data The development of an automatic verb classification approach requires an initial gold standard (Sun et al., 2010): these have been developed for FR (Sun et al., 2010), PT (Scarton et al., 2014), IT, PL, HR, and FI (Majewska et al., 2017). They were created using the methodology of Sun et al. (2010), based on the EN gold standard of Sun et al. (2008) which contains 17 fine-grained Levin classes with 12 member verbs each. For instance, the class PUT-9.1 in French contains verbs such as *accrocher, déposer, mettre, répartir, réintégrer*, etc.

Evaluation Measures We use standard evaluation measures from prior work on verb clustering (Ó Séaghdha and Copestake, 2008; Sun and Korhonen, 2009; Sun et al., 2010; Falk et al., 2012, i.a.). The mean *precision* of induced verb clusters labelled *modified purity* (MPUR) is computed as:

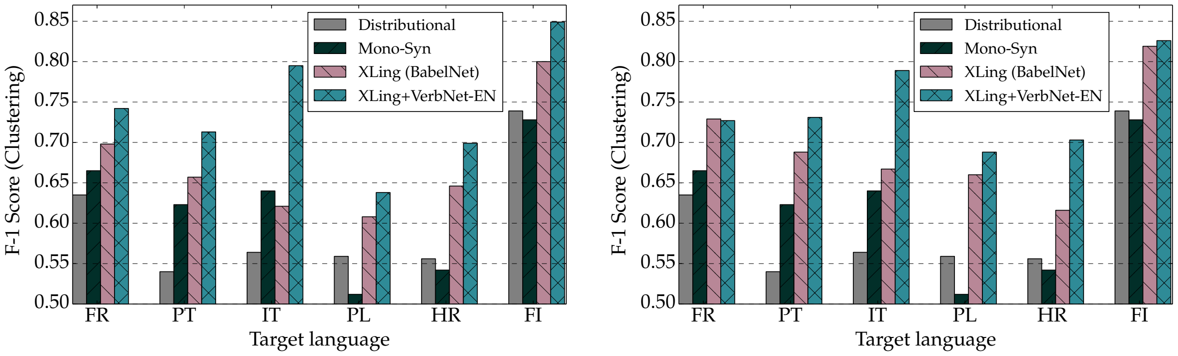
$$\text{MPUR} = \frac{\sum_{C \in \mathbf{Clust}, n_{prev(C)} > 1} n_{prev(C)}}{\#\text{test\_verbs}} \quad (4)$$

Here, each cluster C from the set of all K_{Clust} induced clusters **Clust** is associated with its prevalent class/cluster from the gold standard, and the number of verbs in an induced cluster C taking this prevalent class is labelled $n_{prev(C)}$. All other verbs not taking the prevalent class are considered errors.[10] *#test_verbs* denotes the total number of test verb instances. The second measure targeting *recall* is *weighted class accuracy* (WACC), computed as:

$$\text{WACC} = \frac{\sum_{C \in \mathbf{Gold}} n_{dom(C)}}{\#\text{test\_verbs}} \quad (5)$$

[9] https://github.com/nmrksic/attract-repel

[10] Clusters with $n_{prev(C)} = 1$ are discarded from the count to avoid an undesired bias towards singleton clusters (Sun and Korhonen, 2009; Sun et al., 2010).



(a) EN Vectors: SGNS-BOW2

(b) EN Vectors: SGNS-DEPS

Figure 2: *F-1* scores in six target languages using the post-processing specialisation procedure from Sect. 2.1 and different sets of constraints: *Distributional* refers to the initial vector space in each target language; *Mono-Syn* is the vector space tuned using monolingual synonymy constraints from BabelNet; *XLing* uses cross-lingual EN-TARGET constraints from BabelNet (TARGET refers to any of the six target languages); *XLing+VerbNet-EN* is a fine-tuned vector space which uses both cross-lingual EN-TARGET constraints plus EN VerbNet constraints. Results are provided with (a) SGNS-BOW2 and (b) SGNS-DEPS source vector space in English for the *XLing* and *XLing+VerbNet* variants, see Sect. 3.

For each cluster C from the set of gold standard clusters **Gold**, we have to find the dominant cluster from the set of induced clusters: this cluster has the most verbs in common with the gold cluster C, and that number is $n_{dom(C)}$. As measures of precision and recall, MPUR and WACC may be combined into an F-1 score, computed as the balanced harmonic mean, which we report in this work.[11]

4 Results and Discussion

Cross-Lingual Transfer Model F-1 verb classification scores for the six target languages with different sets of constraints are summarised in Fig. 2. We can draw several interesting conclusions. First, the strongest results on average are obtained with the model which transfers the VerbNet knowledge from English (as a resource-rich language) to the resource-lean target language (providing an answer to question Q3, Sect. 1). These improvements are visible across all target languages, empirically demonstrating the cross-lingual nature of VerbNet-style classifications. Second, using cross-lingual constraints alone (*XLing*) yields strong gains over initial distributional spaces (answering Q1 and Q2). Fig. 2 also shows that cross-lingual similarity constraints are more beneficial than the monolingual ones, despite a larger total number of the monolingual constraints in each language (see Tab. 2). This suggests that such cross-lingual similarity links are strong implicit indicators of class membership. Namely, target language words which map to the same source language word are likely to be synonyms and consequently end up in the same verb class in the target language. However, the cross-lingual links are even more useful as means for transferring the VerbNet knowledge, as evidenced by additional gains with *XLing+VerbNet-EN*.

The absolute classification scores are the lowest for the two Slavic languages: PL and HR. This may be partially explained by the lowest number of cross-lingual constraints for the two languages covering only a subset of their entire vocabularies (see Tab. 2 and compare the total number of constraints for HR and PL to the numbers for e.g. FI or FR). Another reason for weaker performance of these two languages could be their rich morphology, which induces data sparsity both in the initial vector space estimation and in the coverage of constraints.

State-of-the-Art A direct comparison of previous state-of-the-art classification scores available for FR (Sun et al., 2010) and PT (Scarton et al., 2014) on the same test data exemplifies the extent of improvement achieved by our transfer model. F-1 scores improve from 0.55 to 0.75 for FR and from 0.43 to 0.73 for PT. Scarton et al. (2014) explain the low performance by "the lower quality NLP tools". This issue is largely mitigated by our VerbNet transfer model, which exploits the assump-

[11] We have also experimented with V-measure (Rosenberg and Hirschberg, 2007), another standard evaluation measure which balances between homogeneity (precision) and completeness (recall); we do not report these scores for brevity as similar trends in results are observed.

	French		Italian	
Vector space	*XLing*	*XLing+VN*	*XLing*	*XLing+VN*
EN+FR	0.698	0.742	–	–
EN+IT	–	–	0.621	**0.795**
EN+FR+IT	**0.728**	0.745	0.650	0.768
EN+FR+PL	0.697	0.717	–	–
EN+IT+PL	–	–	0.658	0.765
EN+FI+FR+IT	0.719	**0.751**	0.662	0.777
EN+FR+IT+PT	0.710	0.718	**0.688**	0.760

Table 3: The effect of multilingual vector space specialisation. Results are reported for FR and IT using: a) cross-lingual constraints only (*XLing*); and b) the VerbNet transfer model (*XLing+VN*).

tion of cross-linguistic class consistency directly through a specialised vector space, and also avoids any reliance on target-language-specific NLP tools.

Starting Source Vector Space Fig. 2a and Fig. 2b enable a brief analysis of the influence of the starting EN vector space on the results for each target language. We observe small but consistent gains with SGNS-DEPS, which utilises syntactic information stemming from a dependency parser on the source side, over SGNS-BOW for the *XLing* variant. The improvements are +2.1 points on average, visible for 5 out of 6 target languages. We again see an increase in performance with the *XLing+VerbNet* model, but we do not observe any major difference between the two starting source spaces now: average slight score difference of 0.3 is in favour of SGNS-BOW2, which outperforms SGNS-DEPS for 3 out of 6 target languages. This finding indicates that VerbNet-based linguistic constraints are more important for the final classification performance, and mitigate the artefacts of the starting distributional source space.

Bilingual vs. Multilingual The transfer model can operate with more than two languages, effectively inducing a multilingual vector space. We analyse such multilingual training based on the results on FR and IT (Tab. 3). On average, the results with *XLing* improve with more languages (see also the results for EN in Tab. 4), as the model relies on more constraints for the vector space specialisation. Yet additional languages do not lead to clear improvements with *XLing+VerbNet*: we hypothesise that the specialisation procedure becomes dominated by cross-lingual constraints which may diminish the importance of VerbNet-based EN constraints. The language configuration in the multilingual vector space also makes a difference:

	VC:XLing	*Sim*:XLing	*Sim*:XLing+VN
EN-Dist	0.484	0.275	–
+FR	0.608	0.556	0.481
+PT	0.633	0.537	0.466
+IT	0.602	0.524	0.476
+PL	0.597	0.469	0.431
+HR	0.582	0.497	0.446
+FI	0.662	0.598	0.491
+FR+IT	0.633	0.571	0.526
+FI+FR+IT	0.641	0.635	0.558
+FR+IT+PT	0.674	0.596	0.515
EN-VN	**0.956**	–	0.358

Table 4: Comparison of verb classification (*VC*) and verb semantic similarity (*Sim*) for English. *VC* is measured on the EN test set of Sun et al. (2008). *Sim* is measured on SimVerb-3500 (Gerz et al., 2016). The scores are Spearman's ρ correlation scores. EN-Dist is the initial distributional English vector space: SGNS-BOW2; EN-VN is the same space transformed using monolingual EN VerbNet constraints only, an upper bound for the specialisation-based approach in EN.

e.g., combining PL with the Romance languages degrades the performance, while FI surprisingly boosts it slightly. For brevity, we only report the results for FR and IT. Similar trends are observed when making the transition from bilingual to multilingual vector spaces for other target languages.

Clustering Algorithm Since vector space specialisation is detached from the application of the clustering algorithm, our framework allows straightforward experimentation with other algorithms. Following prior work (Brew and Schulte im Walde, 2002; Sun et al., 2010), we also test K-means clustering. Results for the six languages using the EN SGNS-BOW2 source space and *Xling+VerbNet-EN* are on average 3.8 points lower than the ones reported in Fig. 2a. K-Means is outperformed for each target language, confirming the superiority of spectral clustering established in prior work, e.g., (Scarton et al., 2014). On the other hand, we find results with another clustering algorithm, hierarchical agglomerative clustering with Ward's linkage (Ward, 1963), on par with spectral clustering (1.4 points on average in favour of spectral, which is better on 4 out of 6 languages). We believe that further gains in verb class induction could be achieved by additional fine-tuning of the clustering algorithm.

Other Cross-Lingual Sources Replacing BabelNet with PanLex as the alternative source of cross-

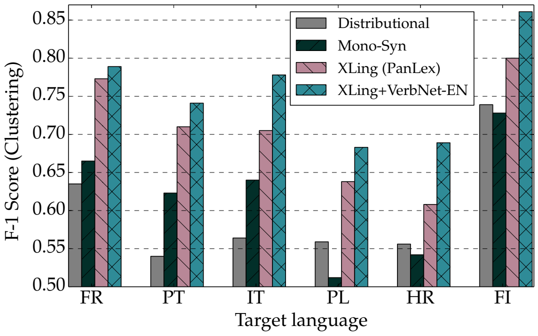


Figure 3: *F-1* scores when PanLex is used as the source of cross-lingual ATTRACT constraints (instead of BabelNet). EN Vectors: SGNS-BOW2.

lingual information again leads to large gains with the cross-lingual transfer model, as is evident from Fig. 3. This suggests that the proposed approach does not depend on a particular source of information - it can be used with any general-purpose bilingual dictionary. We mark slight improvements for 3/6 target languages when comparing the results with the ones from Fig. 2a. The new state-of-the-art F-1 scores are 0.79 for FR and 0.74 for PT.

Verb Classification vs. Semantic Similarity An interesting question originating from prior work on verb representation learning, e.g., (Baker et al., 2014) touches upon the correlation between verb classification and semantic similarity. Due to the availability of VerbNet constraints and a recent similarity evaluation set (SimVerb-3500; it contains human similarity ratings for 3,500 verb pairs) (Gerz et al., 2016), we perform the analysis on English: the results are summarised in Tab. 4. They clearly indicate that cross-lingual synonymy constraints are useful for both relationship types (compare the scores with *XLing*), with strong gains over the non-specialised distributional space. However, the inclusion of VerbNet information, while boosting classification scores for target languages and (trivially) for EN, deteriorates EN similarity scores across the board (compare *XLing+VN* against *XLing* in Tab. 4). This suggests that the VerbNet-style class membership is definitely not equivalent to pure semantic similarity captured by SimVerb.

4.1 Further Discussion and Future Work

This work has proven the potential of transferring lexical resources from resource-rich to resource-poor languages using general-purpose cross-lingual dictionaries and bilingual vector spaces as means of transfer within a semantic specialisation framework. However, we believe that the proposed basic framework may be upgraded and extended across several research paths in future work.

First, in the current work we have operated with standard single-sense/single-prototype representations, thus effectively disregarding the problem of verb polysemy. While several polysemy-aware verb classification models for English were developed recently (Kawahara et al., 2014; Peterson et al., 2016), the current lack of polysemy-aware evaluation sets in other languages impedes this line of research. Evaluation issues aside, one idea for future work is to use the ATTRACT-REPEL specialisation framework for sense-aware cross-lingual transfer relying on recently developed multi-sense/prototype word representations (Neelakantan et al., 2014; Pilehvar and Collier, 2016, inter alia).

Another challenge is to apply the idea from this work to enable cross-lingual transfer of other structured lexical resources available in English such as FrameNet (Baker et al., 1998), PropBank (Palmer et al., 2005), and VerbKB (Wijaya and Mitchell, 2016). Other potential research avenues include porting the approach to other typologically diverse languages and truly low-resource settings (e.g., with only limited amounts of parallel data), as well as experiments with other distributional spaces, e.g. (Melamud et al., 2016). Further refinements of the specialisation and clustering algorithms may also result in improved verb class induction.

5 Conclusion

We have presented a novel cross-lingual transfer model which enables the automatic induction of VerbNet-style verb classifications across multiple languages. The transfer is based on a word vector space specialisation framework, utilised to directly model the assumption of cross-linguistic validity of VerbNet-style classifications. Our results indicate strong improvements in verb classification accuracy across all six target languages explored. All automatically induced VerbNets are available at: `github.com/cambridgeltl/verbnets`.

Acknowledgments

This work is supported by the ERC Consolidator Grant LEXICAL: Lexical Acquisition Across Languages (no 648909). The authors are grateful to the entire LEXICAL team, especially to Roi Reichart, and also to the three anonymous reviewers for their helpful and constructive suggestions.

References

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of EACL*, pages 937–947.

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

Juan Aparicio, Mariona Taulé, and Maria Antònia Martí. 2008. AnCora-Verb: A lexical resource for the semantic annotation of corpora. In *Proceedings of LREC*, pages 797–802.

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

Simon Baker, Roi Reichart, and Anna Korhonen. 2014. An unsupervised model for instance level subcategorization acquisition. In *Proceedings of EMNLP*, pages 278–289.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky wide web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of ACL*, pages 238–247.

Vladimír Benko. 2014. Aranea: Yet another family of (comparable) Web corpora. In *Proceedings of TSD*, pages 247–254.

Chris Brew and Sabine Schulte im Walde. 2002. Spectral clustering for German verbs. In *Proceedings of EMNLP*, pages 117–124.

Susan Windisch Brown, Dmitriy Dligach, and Martha Palmer. 2011. VerbNet class assignment as a WSD task. In *Proceedings of IWCS*, pages 85–94.

Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. 2006. Relation extraction using label propagation based semi-supervised learning. In *Proceedings of ACL*, pages 129–136.

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

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2016. Learning crosslingual word embeddings without bilingual corpora. In *Proceedings of EMNLP*, pages 1285–1295.

Maud Ehrmann, Francesco Cecconi, Daniele Vannella, John Philip Mccrae, Philipp Cimiano, and Roberto Navigli. 2014. Representing multilingual data as linked data: The case of BabelNet 2.0. In *Proceedings of LREC*, pages 401–408.

Ingrid Falk, Claire Gardent, and Jean-Charles Lamirel. 2012. Classifying French verbs using French and English lexical resources. In *Proceedings of ACL*, pages 854–863.

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

Christiane Fellbaum. 1998. *WordNet*.

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

Ana-Maria Giuglea and Alessandro Moschitti. 2006. Semantic role labeling via FrameNet, VerbNet and PropBank. In *Proceedings of ACL*, pages 929–936.

Jeffrey Gruber. 1976. *Lexical structure in syntax and semantics*.

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

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

Ray S. Jackendoff. 1972. *Semantic interpretation in generative grammar*.

Ray S. Jackendoff. 1992. *Semantic structures*.

Eric Joanis, Suzanne Stevenson, and David James. 2008. A general feature space for automatic verb classification. *Natural Language Engineering*, 14(3):337–367.

David Kamholz, Jonathan Pool, and Susan M. Colowick. 2014. PanLex: Building a resource for panlingual lexical translation. In *Proceedings of LREC*, pages 3145–3150.

Daisuke Kawahara, Daniel W. Peterson, and Martha Palmer. 2014. A step-wise usage-based method for inducing polysemy-aware verb classes. In *Proceedings of ACL*, pages 1030–1040.

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

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

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

Karin Kipper. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis.

Karin Kipper, Hoa Trang Dang, and Martha Stone Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of AAAI*, pages 691–696.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2006. Extending VerbNet with novel verb classes. In *Proceedings of LREC*.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation*, 42(1):21–40.

Anna Korhonen. 2010. Automatic lexical classification: Bridging research and practice. *Philoshophical Transactions A of the Royal Society*, 368:3621–3632.

Beth Levin. 1993. *English verb classes and alternation, A preliminary investigation*.

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

Jianguo Li and Chris Brew. 2008. Which are the best features for automatic verb classification. In *Proceedings of ACL*, pages 434–442.

Thomas Lippincott, Laura Rimell, Karin Verspoor, and Anna Korhonen. 2013. Approaches to verb subcategorization for biomedicine. *Journal of Biomedical Informatics*, 46(2):212–227.

Mei-Chun Liu and Ting-Yi Chiang. 2008. The construction of Mandarin VerbNet: A frame-based study of statement verbs. *Language and Linguistics*, 9(2):239–270.

Nikola Ljubešić and Filip Klubička. 2014. {bs,hr,sr}WaC – Web corpora of Bosnian, Croatian and Serbian. In *Proceedings of the 9th Web as Corpus Workshop*, pages 29–35, Gothenburg, Sweden. Association for Computational Linguistics.

Nikola Ljubešić, Tommi Pirinen, and Antonio Toral. 2016. Finnish Web corpus fiWaC 1.0. Slovenian language resource repository CLARIN.SI.

Edward Loper, Szu-Ting Yi, and Martha Palmer. 2007. Combining lexical resources: Mapping between PropBank and VerbNet. In *Proceedings of IWCS*.

Ulrike von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416.

Olga Majewska, Ivan Vulić, Diana McCarthy, Yan Huang, Akira Murakami, Veronika Laippala, and Anna Korhonen. 2017. Investigating the cross-lingual translatability of verbnet-style classification. *Language Resources and Evaluation*.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of EMNLP*, pages 523–534.

Marina Meila and Jianbo Shi. 2001. A random walks view of spectral segmentation. In *Proceedings of AISTATS*.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. Context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of CoNLL*, pages 51–61.

Paola Merlo, Suzanne Stevenson, Vivian Tsang, and Gianluca Allaria. 2002. A multilingual paradigm for automatic verb classification. In *Proceedings of ACL*, pages 207–214.

Cédric Messiant. 2008. A subcategorization acquisition system for French verbs. In *Proceedings of ACL Student Research Workshop*, pages 55–60.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of ICLR Workshop Papers*.

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

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL*, pages 142–148.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Tsung-Hsien Wen, and Steve Young. 2017a. Neural Belief Tracker: Data-driven dialogue state tracking. In *Proceedings of ACL*.

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

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of ICML*, pages 807–814.

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

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of EMNLP*, pages 1059–1069.

Diarmuid Ó Séaghdha and Ann Copestake. 2008. Semantic classification with distributional kernels. In *Proceedings of COLING*, pages 649–656.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Karel Pala and Aleš Horák. 2008. Can complex valency frames be universal? In *Proceedings of RASLAN*, pages 41–49.

Martha Palmer, Paul Kingsbury, and Daniel Gildea. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Daniel W. Peterson, Jordan Boyd-Graber, Martha Palmer, and Daisuke Kawahara. 2016. Leveraging VerbNet to build corpus-specific verb clusters. In *Proceedings of *SEM*, pages 102–107.

Mohammad Taher Pilehvar and Nigel Collier. 2016. De-conflated semantic representations. In *Proceedings of EMNLP*, pages 1680–1690.

Quentin Pradet, Laurence Danlos, and Gaël de Chalendar. 2014. Adapting VerbNet to French using existing resources. In *Proceedings of LREC*, pages 1122–1126.

Laura Rimell, Thomas Lippincott, Karin Verspoor, Helen L. Johnson, and Anna Korhonen. 2013. Acquisition and evaluation of verb subcategorization resources for biomedicine. *Journal of Biomedical Informatics*, 46(2):228–237.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of EMNLP*, pages 410–420.

Carolina Scarton and Sandra Aluısio. 2012. Towards a cross-linguistic VerbNet-style lexicon for Brazilian Portuguese. In *Proceedings of the LREC 2012 Workshop on Creating Cross-language Resources for Disconnected Languages and Styles*.

Carolina Scarton, Lin Sun, Karin Kipper Schuler, Magali Sanches Duran, Martha Palmer, and Anna Korhonen. 2014. Verb clustering for Brazilian Portuguese. In *Proceedings of CICLing*, pages 25–39.

Sabine Schulte im Walde. 2006. Experiments on the automatic induction of German semantic verb classes. *Computational Linguistics*, 32(2):159–194.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2016. Symmetric patterns and coordinations: Fast and enhanced representations of verbs and adjectives. In *Proceedings of NAACL-HLT*, pages 499–505.

Lei Shi and Rada Mihalcea. 2005. Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing. In *Proceedings of CICLing*, pages 100–111.

Rajen Subba and Barbara Di Eugenio. 2009. An effective discourse parser that uses rich linguistic information. In *Proceedings of NAACL-HLT*, pages 566–574.

Lin Sun and Anna Korhonen. 2009. Improving verb clustering with automatically acquired selectional preferences. In *Proceedings of EMNLP*, pages 638–647.

Lin Sun, Anna Korhonen, and Yuval Krymolowski. 2008. Verb class discovery from rich syntactic data. In *Proceedings of CICLing*, pages 16–27.

Lin Sun, Diana McCarthy, and Anna Korhonen. 2013. Diathesis alternation approximation for verb clustering. In *Proceedings of ACL*, pages 736–741.

Lin Sun, Thierry Poibeau, Anna Korhonen, and Cédric Messiant. 2010. Investigating the cross-linguistic potential of VerbNet-style classification. In *Proceedings of COLING*, pages 1056–1064.

Robert S. Swier and Suzanne Stevenson. 2004. Unsupervised semantic role labelling. In *Proceedings of EMNLP*, pages 95–102.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of ICML*.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

Ivan Vulić and Anna Korhonen. 2016. Is "universal syntax" universally useful for learning distributed word representations? In *Proceedings of ACL*, pages 518–524.

Jorge Alberto Wagner Filho, Rodrigo Wilkens, Leonardo Zilio, Marco Idiart, and Aline Villavicencio. 2016. Crawling by readability level. In *Proceedings of PROPOR*, pages 306–318.

Joe H. Ward. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244.

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

Derry Tanti Wijaya and Tom M. Mitchell. 2016. Mapping verbs in different languages to knowledge base relations using web text as interlingua. In *Proceedings of NAACL-HLT*, pages 818–827.

Lihi Zelnik-Manor and Pietro Perona. 2004. Self-tuning spectral clustering. In *Proceedings of NIPS*, pages 1601–1608.

Classification of telicity using cross-linguistic annotation projection

Annemarie Friedrich[1] **Damyana Gateva**[2]
[1]Center for Information and Language Processing, LMU Munich
[2]Department of Computational Linguistics, Saarland University
anne@cis.uni-muenchen.de dgateva@coli.uni-saarland.de

Abstract

This paper addresses the automatic recognition of telicity, an aspectual notion. A *telic* event includes a natural endpoint (*she walked home*), while an *atelic* event does not (*she walked around*). Recognizing this difference is a prerequisite for temporal natural language understanding. In English, this classification task is difficult, as telicity is a covert linguistic category. In contrast, in Slavic languages, aspect is part of a verb's meaning and even available in machine-readable dictionaries. Our contributions are as follows. We successfully leverage additional silver standard training data in the form of projected annotations from parallel English-Czech data as well as context information, improving automatic telicity classification for English significantly compared to previous work. We also create a new data set of English texts manually annotated with telicity.

1 Introduction

This paper addresses the computational modeling of *telicity*, a linguistic feature which represents whether the event type evoked by a sentence's verb constellation (i.e., the verb and its arguments and modifiers) has a natural endpoint or not (Comrie, 1976; Smith, 1997), see (1a) and (1b).

(1) (a) Mary ate an apple. (*telic*)
(b) I gazed at the sunset. (*atelic*)

Automatic recognition of telicity is a necessary step for natural language understanding tasks that require reasoning about time, e.g., natural language generation, summarization, question answering, information extraction or machine translation (Moens and Steedman, 1988; Siegel and McKeown, 2000). For example, there is an entailment relation between English Progressive and Perfect constructions (as shown in (2)), but only for atelic verb constellations.

(2) (a) He was swimming in the lake. (*atelic*)
$\models$ He has swum in the lake.
(b) He was swimming across the lake. (*telic*)
$\not\models$ He has swum across the lake.

We model telicity at the word-sense level, corresponding to the *fundamental aspectual class* of Siegel and McKeown (2000), i.e., we take into account the verb and its arguments and modifiers, but no additional aspectual markers (such as the Progressive). In (2) we classify whether the event types "swim in the lake" and "swim across the lake" have natural endpoints. This is defined on a linguistic level rather than by world knowledge requiring inference. "Swimming in the lake" has no natural endpoint, as also shown by the linguistic test presented in (2). In contrast, "swimming across the lake" will necessarily be finished once the other side is reached.

In English, the aspectual notion of telicity is a covert category, i.e., a semantic distinction that is not expressed overtly by lexical or morphological means. As illustrated by (2) and (3), the same *verb type* (lemma) can introduce telic and atelic events to the discourse depending on the context in which it occurs.

(3) (a) John drank coffee. (*atelic*)
(b) John drank a cup of coffee. (*telic*)

In Slavic languages, aspect is a component of verb meaning. Most verb types are either *perfective* or *imperfective* (and are marked as such in dictionaries). For example, the two occurrences of "drink" in (3) are translated into Czech using the imperfective verb "pil" and the perfective verb

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2559–2565
Copenhagen, Denmark, September 7–11, 2017.

"vypil," respectively (Filip, 1994):[1]

(4) (a) Pil kávu. (*imperfective*)
He was drinking (some) coffee.
(b) Vypil kávu. (*perfective*)
He drank up (all) the coffee.

Our contributions are as follows: (1) using the English-Czech part of InterCorp (Čermák and Rosen, 2012) and a valency lexicon for Czech verbs (Žabokrtský and Lopatková, 2007), we create a large silver standard with automatically derived annotations and validate our approach by comparing the labels given by humans versus the projected labels; (2) we provide a freely available data set of English texts taken from MASC (Ide et al., 2010) manually annotated for telicity; (3) we show that using contextual features and the silver standard as additional training data improves computational modeling of telicity for English in terms of F_1 compared to previous work.

2 Related work

Siegel and McKeown (2000, henceforth SMK2000) present the first machine-learning based approach to identifying *completedness*, i.e., telicity, determining whether an event reaches a culmination or completion point at which a new state is introduced. Their approach describes each verb occurrence exclusively using features reflecting corpus-based statistics of the corresponding verb type. For each verb type, they collect the co-occurrence frequencies with 14 linguistic markers (e.g., present tense, perfect, combination with temporal adverbs) in an automatically parsed background corpus. They call these features *linguistic indicators* and train a variety of machine learning models based on 300 clauses, of which roughly 2/3 are *culminated*, i.e., telic. Their test set also contains about 300 clauses, corresponding to 204 distinct non-stative verbs. Their data sets are not available, but as this work is the most closely related to ours, we reimplement their approach and compare to it in Section 5.

Samardžić and Merlo (2016) create a model for real-world duration of events (as *short* or *long*) of English verbs as annotated in TimeBank (Pustejovsky et al., 2003). The model is informed by temporal boundedness information collected from parallel English-Serbian data. Their only features are how often the respective verb type was aligned to Serbian verbs carrying certain affixes that indicate perfectiveness or imperfectiveness. Their usage of "verb type" differs from ours as they do not lemmatize, i.e., they always predict that "falling" is a long event, while "fall" is short. Our approach shares the idea of projecting aspectual information from Slavic languages to English, but in contrast to classifying verb types, we classify whether an event type introduced by the verb constellation of a clause is telic or atelic, making use of a machine-readable dictionary for Czech instead of relying on affix information.

Loáiciga and Grisot (2016) create an automatic classifier for boundedness, defined as whether the endpoint of an event has occurred or not, and show that this is useful for picking the correct tense in French translations of the English Simple Past. Their classifier employs a similar but smaller feature set compared to ours. Other related work on predicting aspect include systems aiming at identifying lexical aspect (Siegel and McKeown, 2000; Friedrich and Palmer, 2014) or habituals (Mathew and Katz, 2009; Friedrich and Pinkal, 2015).

Cross-linguistic annotation projection approaches mostly make use of existing manually created annotations in the source language; similar to our approach, Diab and Resnik (2002) and Marasović et al. (2016) leverage properties of the source language to automatically induce annotations on the target side.

3 Data sets and annotation projection

We conduct our experiments based on two data sets: (a) English texts from MASC manually annotated for telicity, on which we train and test our computational models, and (b) a silver standard automatically extracted via annotation projection from the Czech-English part of the parallel corpus InterCorp, which we use as additional training data in order to improve our models.[2]

3.1 Gold standard: MASC (EN)

We create a new data set consisting of 10 English texts taken from MASC (Ide et al., 2010), annotated for telicity. Texts include two essays, a journal article, two blog texts, two history texts from travel guides, and three texts from the fic-

[1] In Czech, aspectual verb pairs may be related by affixes as in this example, but this is not always the case. They may even use different lexemes (Vintr, 2001).

[2] Annotations, guidelines and code available from `https://github.com/annefried/telicity`

	MASC (gold standard)	InterCorp (silver standard)	intersection
clauses (instances)	1863	457,000	-
% telic	82	55	-
% atelic	18	45	-
distinct verb types (lemmas)	567	2262	510
ambiguous verb types	70	1130	69

Table 1: Corpus statistics.

tion genre. Annotation was performed using the web-based SWAN system (Gühring et al., 2016). Annotators were given a short written manual with instructions. We model telicity for *dynamic* (*eventive*) verb occurrences because *stative* verbs (e.g., "like") do not have built-in endpoints by definition. Annotators choose one of the labels *telic* and *atelic* or they skip clauses that they consider to be stative. In a first round, each verb occurrence was labeled by three annotators (the second author of this paper plus two paid student assistants). They unanimously agreed on telicity labels for 1166 verb occurrences; these are directly used for the gold standard. Cases in which only two annotators agreed on a telicity label (the third annotator may have either disagreed or skipped the clause) are labeled by a fourth independent annotator (the first author), who did not have access to the labels of the first rounds. This second annotation round resulted in 697 further cases in which three annotators gave the same telicity label. Statistics for our final gold standard, which consists of all instances for which at least three out of the four annotators agreed, are shown in Table 1; "ambiguous" verb types are those for which the gold standard contains both telic and atelic instances. 510 of the 567 verb types also occur in the InterCorp silver standard, which provides training instances for 69 out of the 70 ambiguous verb types.

Finally, there are 446 cases for which no three annotators supplied the same label. Disagreement and skipping was mainly observed for verbs indicating attributions ("critics claim" or "the film uses"), which can be perceived either as statives or as instances of historic present. Other difficult cases include degree verbs ("increase"), aspectual verbs ("begin"), perception verbs ("hear"), iteratives ("flash") and the verb "do." For these cases, decisions how to treat them may have to be made depending on the concrete application; for now, they are excluded from our gold standard. Another source of error is that despite the training, annotators sometimes conflate their world knowledge (i.e., that some events necessarily come to an end eventually, such as the "swimming in the lake" in (2)) with the annotation task of determining telicity at a linguistic level.

3.2 Silver standard: InterCorp (EN-CZ)

We create a silver standard of approximately 457,000 labeled English verb occurrences (i.e., clauses) extracted from the InterCorp parallel corpus project (Čermák and Rosen, 2012). We leverage the sentence alignments, as well as part-of-speech and lemma information provided by InterCorp. We use the data from 151 sentence-aligned books (novels) of the Czech-English part of the corpus and further align the verbs of all 1:1-aligned sentence pairs to each other using the verbs' lemmas, achieving high precision by making sure that the translation of the verbs is licensed by the free online dictionary Glosbe.[3] We then look up the aspect of the Czech verb in Vallex 2.8.3 (Žabokrtský and Lopatková, 2007), a valency lexicon for Czech verbs, and project the label *telic* to English verb occurrences corresponding to a *perfective* Czech verb and the label *atelic* to instances translated using *imperfective* verbs.

Our annotation projection approach leverages the fact that most perfective Czech verbs will be translated into English using verb constellations that induce a telic event structure, as they describe one-time finished actions. Imperfective verbs, in contrast, are used for actions that are presented as unfinished, repeated or extending in time (Vintr, 2001). They are often, but not always, translated using atelic verb constellations. A notable exception is the English Progressive: "John was reading a book" signals an ongoing event in the past, which is telic at the word-sense level but would require translation using the imperfective Czech verb "četl." The initial corpus contained 4% sentences in the Progressive, out of which 89% were translated using imperfectives.[4] Due to the above

[3] https://glosbe.com

[4] For comparison, in the manually annotated validation

described mismatch, we remove all English Progressive sentences from our silver standard. Statistics for the final automatically created silver standard are shown in Table 1.

For validation, we sample 2402 instances from the above created silver standard and have our three annotators from the first annotation round mark them in the same way as the MASC data. Sampling picked one instance per verb type but was otherwise random. A majority agreement among the three annotators can be reached in 2126 cases (due to allowing skipping).[5] In this sample, 77.8% of the instances received the label telic from the human annotators, 61.5% received the label telic from the projection method. The accuracy of our projection method can be estimated as about 78%; F_1 for the telic class is 0.84, F_1 for atelic is 0.65. Errors made by the projection include for instance habituals, which use the imperfective in Czech but are not necessarily atelic at the event type level as in "John cycles to work every day."

4 Computational modeling

In this section, we describe the computational models for telicity classification, which we test on the MASC data and which we improve by adding the InterCorp silver standard data.

Features. We model each instance by means of a variety of syntactic-semantic features, using the toolkit provided by Friedrich et al. (2016).[6] Preprocessing is done using Stanford CoreNLP (Chen and Manning, 2014) based on dkpro (Eckart de Castilho and Gurevych, 2014). For the verb's lemma, the features include the WordNet (Fellbaum, 1998) sense and supersense and linguistic indicators (Siegel and McKeown, 2000) extracted from GigaWord (Graff et al., 2003). Using *only* the latter as features corresponds to the system by SMK2000 as described in Section 2. The feature set also describes the verb's subject and objects; among others their number, person, countability[7], their most frequent WordNet sense and the respective supersenses, and dependency relations between the argument and its governor(s). In addition, tense, voice and whether the clause is in the Perfect or Progressive aspect is reflected, as well as the presence of clausal (e.g., temporal) modifiers. For replicability we make the configuration files for the feature set available.

Classifier. We train L1-regularized multi-class logistic regression models using LIBLINEAR (Fan et al., 2008) with parameter settings ε=0.01 and bias=1. For each instance described by feature vector $\vec{x}$, the probability of each possible label y (here telic or atelic) is computed according to

$$P(y|\vec{x}) = \frac{1}{Z(\vec{x})} exp\left(\sum_{i=1}^{m} \lambda_i f_i(y, \vec{x})\right),$$

where f_i are the feature functions, λ_i are the weights learned for each feature function, and $Z(\vec{x})$ is a normalization constant (Klinger and Tomanek, 2007). The feature functions f_i indicate whether a particular feature is present, e.g., whether the tense of the verb is "past."

5 Experiments

Experimental settings. We evaluate our models via 10-fold cross validation (CV) on the MASC data set. We split the data into folds by documents in order to make sure that no training data from the same document is available for each instance in order to avoid an unfair bias. We report results in terms of accuracy, F_1 per class and macro-average F_1 (the harmonic mean of macro-average precision and recall). We test significance between differences in F_1 (for each class) using approximate randomization (Yeh, 2000; Padó, 2006) with $p < 0.1$ and significance between differences in accuracy using McNemar's test (McNemar, 1947) with $p < 0.01$. Table 2 shows our results: significantly different scores are marked with the same symbol where relevant (per column).

Results. A simple baseline of labeling each instance with the overall majority class (telic) has a very high accuracy, but the output of this baseline is uninformative and results in a low F_1. Rows titled "verb type" use the verb's lemma as their single feature and thus correspond to the informed baseline of using the training set majority class for each verb type. Rows labeled "+IC" indicate that the full set of instances with projected labels extracted from InterCorp has been added as additional training data in each fold; in rows titled "+ICs," the telic instances in InterCorp have been upsampled to match the 80:20 distribution in MASC. Our model using the full set of features significantly outperforms the verb type baseline

sample only 66% of Progressives received the label atelic.

[5] Of the 2402 cases, annotators completely agreed on 1577 cases (1114 telic, 203 atelic, 260 skipped). 85 cases were 2x atelic + 1x skipped, 219 cases were 2x telic + 1x skipped.

[6] https://github.com/annefried/sitent

[7] http://celex.mpi.nl

as well as SMK2000 (see † ‡ ∗). Using the additional training data from InterCorp results in a large improvement in the case of the difficult (because infrequent) atelic class (see ⋆), leading to the best overall results in terms of F_1. The best results regarding accuracy *and* F_1 are reached using the sampled version of the silver standard; the differences compared to the respective best scores in each column (in bold) are not significant.

Ablation experiments on the MASC data show that features describing the clause's main verb are most important: when ablating part-of-speech tag and tense and aspect (Progressive or Perfect), performance deteriorates by 1.8% in accuracy and 5% F_1, hinting at a correlation between telicity and choice of tense-aspect form. Whether this is due to an actual correlation of how telic and atelic verbs are used in context or merely due to annotation errors remains to be investigated in future work.

In sum, our experiments show that using annotations projected onto English text from parallel Czech text as cheap additional training data is a step forward to creating better models for the task of classifying telicity of verb occurrences.

6 Conclusion

Our model using a diverse set of features representing both verb-type relevant information and the context in which a verb occurs strongly outperformed previous work on predicting telicity (Siegel and McKeown, 2000). We have shown that silver standard data induced from parallel Czech-English data is useful for creating computational models for recognizing telicity in English. Our new manually annotated MASC data set is freely available; the projected annotations for InterCorp are published in a stand-off format due to license restrictions.

7 Future work

Aspectual distinctions made by one language rarely *completely* correspond to a linguistic phenomenon observed in another language. As we have discussed in Section 3.2, telicity in English and perfectiveness in Czech are closely related. As shown by our experiments, the projected labels cover useful information for the telicity classification task. One idea for future work is thus to leverage additional projected annotations from similar phenomena in additional languages, possibly improving overall performance by combining complementary information. Clustering more than two languages may also enable us to induce clusters corresponding to the different usages of imperfective verbs in Czech.

	Acc.	**F_1**	**F_1(telic)**	**F_1(atelic)**
maj. class	82.0	45.0	90.1	0.0
SMK2000	†83.0	63.9	†90.4	†26.8
SMK2000+IC	78.6	65.6	86.8	44.2
SMK2000+ICs	∗81.8	58.2	89.9	∗12.4
verb type	‡83.8	66.7	‡91.0	‡24.9
verb type+IC	82.4	73.5	89.0	57.1
verb type+ICs	85.1	72.2	91.2	∗51.9
our model	†‡**86.7**	74.5	†‡**92.2**	† ‡ ⋆53.7
our model+IC	82.3	**76.4**	88.6	**61.4**
our model+ICs	∗**86.2**	**76.2**	**91.6**	⋆∗**60.6**

Table 2: Results for telicity classification on MASC data (1863 instances), 10-fold CV.

The presence of endpoints has consequences for the temporal interpretation of a discourse (Smith, 1997; Smith and Erbaugh, 2005), as endpoints introduce new states and therefore signal an advancement of time. In English, boundedness, i.e., whether an endpoint of an event has actually occurred, is primarily signaled by the choice of tense and Progressive or Perfect aspect. In tense-less languages such as Mandarin Chinese, boundedness is a covert category and closely related to telicity. We plan to leverage similar ideas as presented in this paper to create temporal discourse parsing models for such languages.

When translating, telic and atelic constructions also require different lexical choices and appropriate selection of aspectual markers. Hence, telicity recognition is also relevant for machine translation research and could be a useful component in computer aided language learning systems, helping learners to select appropriate aspectual forms.

Acknowledgments

We thank Klára Jágrová, Irina Stenger and Andrea Fischer for their help with Czech-specific questions and with finding appropriate corpora, and Lucie Poláková for pointing us to Vallex. Melissa Peate Sørensen and Christine Bocionek helped with the annotation. Finally, thanks to the anonymous reviewers, Alexis Palmer, Manfred Pinkal, Andrea Horbach, Benjamin Roth, Katharina Kann and Heike Adel for their helpful comments. This research was supported in part by the MMCI Cluster of Excellence of the DFG.

References

František Čermák and Alexandr Rosen. 2012. The case of InterCorp, a multilingual parallel corpus. *International Journal of Corpus Linguistics* 17(3):411–427.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 740–750.

Bernard Comrie. 1976. *Aspect: An introduction to the study of verbal aspect and related problems*, volume 2 of *Cambridge Textbooks in Linguistics*. Cambridge University Press.

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 255–262.

Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*. Dublin, Ireland, pages 1–11.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research* 9:1871–1874.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Massachussetts.

Hana Filip. 1994. Aspect and the semantics of noun phrases. *Tense and aspect in discourse* 75:227.

Annemarie Friedrich and Alexis Palmer. 2014. Automatic prediction of aspectual class of verbs in context. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*. Baltimore, USA.

Annemarie Friedrich, Alexis Palmer, and Manfred Pinkal. 2016. Situation entity types: automatic classification of clause-level aspect. In *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*. Berlin, Germany.

Annemarie Friedrich and Manfred Pinkal. 2015. Automatic recognition of habituals: a three-way classification of clausal aspect. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Lisbon, Portugal.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English Gigaword. *Linguistic Data Consortium, Philadelphia* .

Timo Gühring, Nicklas Linz, Rafael Theis, and Annemarie Friedrich. 2016. SWAN: an easy-to-use web-based annotation system. In *Proceedings of Konferenz zur Verarbeitung natürlicher Sprache (KONVENS)*. Bochum, Germany.

Nancy Ide, Christiane Fellbaum, Collin Baker, and Rebecca Passonneau. 2010. The manually annotated sub-corpus: A community resource for and by the people. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*. Uppsala, Sweden, pages 68–73.

Roman Klinger and Katrin Tomanek. 2007. Classical probabilistic models and conditional random fields. *TU Dortmund Algorithm Engineering Report* .

Sharid Loáiciga and Cristina Grisot. 2016. Predicting and using a pragmatic component of lexical aspect. *Linguistic Issues in Language Technology, Special issue on Modality in Natural Language Understanding* 13.

Ana Marasović, Mengfei Zhou, Alexis Palmer, and Anette Frank. 2016. Modal Sense Classification At Large: Paraphrase-Driven Sense Projection, Semantically Enriched Classification Models and Cross-Genre Evaluations. In *Linguistic Issues in Language Technology, Special issue on Modality in Natural Language Understanding*. CSLI Publications, Stanford, CA., volume 14 (2).

Thomas A. Mathew and E. Graham Katz. 2009. Supervised Categorization of Habitual and Episodic Sentences. In *Sixth Midwest Computational Linguistics Colloquium*. Bloomington, Indiana: Indiana University.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* 12(2):153–157.

Marc Moens and Mark Steedman. 1988. Temporal ontology and temporal reference. *Computational linguistics* 14(2):15–28.

Sebastian Padó. 2006. *User's guide to `sigf`: Significance testing by approximate randomisation.*

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, David Day, Lisa Ferro, Robert Gaizauskas, Marcia Lazo, Andrea Setzer, and Beth Sundheim. 2003. The TimeBank corpus. In *Corpus linguistics*. page 40.

Tanja Samardžić and Paola Merlo. 2016. Aspect-based learning of event duration using parallel corpora. In *Essays in Lexical Semantics and Computational Lexicography – In Honor of Adam Kilgarriff*, Springer Series Text, Speech, and Language Technology.

Eric V Siegel and Kathleen R McKeown. 2000. Learning methods to combine linguistic indicators: Improving aspectual classification and revealing linguistic insights. *Computational Linguistics* 26(4):595–628.

Carlota S Smith. 1997. *The parameter of aspect*, volume 43. Springer Science & Business Media.

Carlota S Smith and Mary S Erbaugh. 2005. Temporal interpretation in Mandarin Chinese. *Linguistics* 43(4):713–756.

Josef Vintr. 2001. *Das Tschechische: Hauptzüge seiner Sprachstruktur in Gegenwart und Geschichte*. Sagner.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pages 947–953.

Zdeněk Žabokrtský and Markéta Lopatková. 2007. Valency information in VALLEX 2.0: Logical structure of the lexicon. *The Prague Bulletin of Mathematical Linguistics* (87):41–60.

Counterfactual Learning from Bandit Feedback under Deterministic Logging: A Case Study in Statistical Machine Translation

Carolin Lawrence
Heidelberg University,
Germany

Artem Sokolov
Amazon Development Center
& Heidelberg University, Germany

Stefan Riezler
Heidelberg University,
Germany

{lawrence,sokolov,riezler}@cl.uni-heidelberg.de

Abstract

The goal of counterfactual learning for statistical machine translation (SMT) is to optimize a target SMT system from logged data that consist of user feedback to translations that were predicted by another, historic SMT system. A challenge arises by the fact that risk-averse commercial SMT systems deterministically log the most probable translation. The lack of sufficient exploration of the SMT output space seemingly contradicts the theoretical requirements for counterfactual learning. We show that counterfactual learning from deterministic bandit logs is possible nevertheless by smoothing out deterministic components in learning. This can be achieved by additive and multiplicative control variates that avoid degenerate behavior in empirical risk minimization. Our simulation experiments show improvements of up to 2 BLEU points by counterfactual learning from deterministic bandit feedback.

1 Introduction

Commercial SMT systems allow to record large amounts of interaction log data at no cost. Such logs typically contain a record of the source, the translation predicted by the system, and the user feedback. The latter can be gathered directly if explicit user quality ratings of translations are supported, or inferred indirectly from the interaction of the user with the translated content. Indirect feedback in form user clicks on displayed ads has been shown to be a valuable feedback signal in response prediction for display advertising (Bottou et al., 2013). Similar to the computational advertising scenario, one could imagine a scenario where SMT systems are optimized from partial information in form of user feedback to predicted translations, instead of from manually created reference translations. This learning scenario has been investigated in the areas of *bandit learning* (Bubeck and Cesa-Bianchi, 2012) or *reinforcement learning* (RL) (Sutton and Barto, 1998). Figure 1 illustrates the learning protocol using the terminology of *bandit structured prediction* (Sokolov et al., 2016; Kreutzer et al., 2017), where at each round, a *system* (corresponding to a *policy* in RL terms) makes a prediction (also called *action* in RL, or pulling an *arm* of a bandit), and receives a *reward*, which is used to update the system.

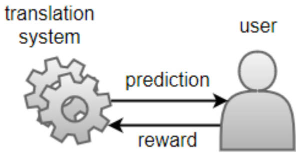


Figure 1: Online learning from partial feedback.

Counterfactual learning attempts to reuse existing interaction data where the predictions have been made by a historic system different from the target system. This enables *offline* or *batch* learning from logged data, and is important if

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2566–2576
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

online experiments that deploy the target system are risky and/or expensive. Counterfactual learning tasks include *policy evaluation*, i.e. estimating how a target policy would have performed if it had been in control of choosing the predictions for which the rewards were logged, and *policy optimization* (also called *policy learning*), i.e. optimizing parameters of a target policy given the logged data from the historic system. Both tasks are called *counterfactual*, or *off-policy* in RL terms, since the target policy was actually not in control during logging. Figure 2 shows the learning protocol for off-policy learning from partial feedback.

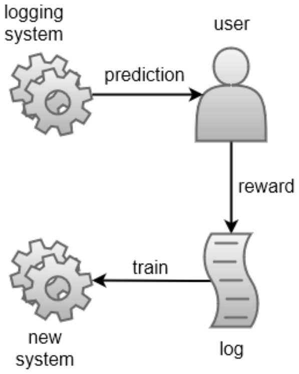


Figure 2: Offline learning from partial feedback.

The crucial trick to obtain unbiased estimators to evaluate and to optimize the off-policy system is to correct the sampling bias of the logging policy. This can be done by importance sampling where the estimate is corrected by the inverse propensity score (Rosenbaum and Rubin, 1983) of the historical algorithm, mitigating the problem that predictions there were favored by the historical system are over-represented in the logs. As shown by Langford et al. (2008) or Strehl et al. (2010), a sufficient exploration of the output space by the logging system is a prerequisite for counterfactual learning. If the logging policy acts stochastically in predicting outputs, this condition is satisfied, and inverse propensity scoring can be applied to correct the sampling bias. However, commercial SMT systems usually try to avoid any risk and only log the most probable translation. This effectively results in deterministic logging policies, making theory and practice of off-policy methods inapplicable to counterfactual learning in SMT.

This paper presents a case study in counterfactual learning for SMT that shows that policy optimization from deterministic bandit logs is possible despite these seemingly contradictory theoretical requirements. We formalize our learning problem as an empirical risk minimization over logged data. While a simple empirical risk minimizer can show degenerate behavior where the objective is minimized by avoiding or over-representing training samples, thus suffering from decreased generalization ability, we show that the use of control variates can remedy this problem. Techniques such as doubly-robust policy evaluation and learning (Dudik et al., 2011) or weighted importance sampling (Jiang and Li, 2016; Thomas and Brunskill, 2016) can be interpreted as additive (Ross, 2013) or multiplicative control variates (Kong, 1992) that serve for variance reduction in estimation. We observe that a further effect of these techniques is that of smoothing out deterministic components by taking the whole output space into account. Furthermore, we conjecture that while outputs are logged deterministically, the stochastic selection of inputs serves as sufficient exploration in parameter optimization over a joint feature representation over inputs and outputs. We present experiments using simulated bandit feedback for two different SMT tasks, showing improvements of up to 2 BLEU in SMT domain adaptation from deterministically logged bandit feedback. This result, together with a comparison to the standard case of policy learning from stochastically logged simulated bandit feedback, confirms the effectiveness our proposed techniques.

2 Related Work

Counterfactual learning has been known under the name of off-policy learning in various fields that deal with partial feedback, namely contextual bandits (Langford et al. (2008); Strehl et al.

(2010); Dudik et al. (2011); Li et al. (2015), *inter alia*), reinforcement learning (Sutton and Barto (1998); Precup et al. (2000); Jiang and Li (2016); Thomas and Brunskill (2016), *inter alia*), and structured prediction (Swaminathan and Joachims (2015a,b), *inter alia*). The idea behind these approaches is to first perform policy evaluation and then policy optimization, under the assumption that better evaluation leads to better optimization. Our work puts a focus on policy optimization in an empirical risk minimization framework for deterministically logged data. Since our experiment is a simulation study, we can compare the deterministic case to the standard scenario of policy optimization and evaluation under stochastic logging.

Variance reduction by additive control variates has implicitly been used in doubly robust techniques (Dudik et al., 2011; Jiang and Li, 2016). However, the connection to Monte Carlo techniques has not been made explicit until Thomas and Brunskill (2016), nor has the control variate technique of optimizing the variance reduction by adjusting a linear interpolation scalar (Ross, 2013) been applied in off-policy learning. Similarly, the technique of weighted importance sampling has been used as variance reduction technique in off-policy learning (Precup et al., 2000; Jiang and Li, 2016; Thomas and Brunskill, 2016). The connection to multiplicative control variates (Kong, 1992) has been made explicit in Swaminathan and Joachims (2015b). To our knowledge, our analysis of both control variate techniques from the perspective of avoiding degenerate behavior in learning from deterministically logged data is novel.

3 Counterfactual Learning from Deterministic Bandit Logs

Problem Definition. The problem of counterfactual learning (in the following used in the sense of counterfactual optimization) for bandit structured prediction can be described as follows: Let $\mathcal{X}$ be a structured input space, let $\mathcal{Y}(x)$ be the set of possible output structures for input x, and let $\Delta : \mathcal{Y} \rightarrow [0,1]$ be a reward function (and $\delta = -\Delta$ be the corresponding task loss function)[1] quantifying the quality of structured outputs. We are given a data log of triples $\mathcal{D} = \{(x_t, y_t, \delta_t)\}_{t=1}^{n}$ where outputs y_t for inputs x_t were generated by a logging system, and loss values δ_t were observed only at the generated data points. In case of stochastic logging with probability π_0, the inverse propensity scoring approach (Rosenbaum and Rubin, 1983) uses importance sampling to achieve an unbiased estimate of the expected loss under the parametric target policy π_w:

$$\begin{aligned}\hat{R}_{\text{IPS}}(\pi_w) &= \frac{1}{n}\sum_{t=1}^{n}\delta_t \frac{\pi_w(y_t|x_t)}{\pi_0(y_t|x_t)} \qquad (1)\\ &\approx \mathbb{E}_{p(x)}\mathbb{E}_{\pi_0(y|x)}[\delta(y)\frac{\pi_w(y|x)}{\pi_0(y|x)}]\\ &= \mathbb{E}_{p(x)}\mathbb{E}_{\pi_w(y|x)}[\delta(y)].\end{aligned}$$

In case of deterministic logging, we are confined to empirical risk minimization:

$$\hat{R}_{\text{DPM}}(\pi_w) = \frac{1}{n}\sum_{t=1}^{n}\delta_t \pi_w(y_t|x_t). \qquad (2)$$

Equation (2) assumes deterministically logged outputs with propensity $\pi_0 = 1, t = 1, \ldots, n$ of the historical system. We call this objective the *deterministic propensity matching (DPM)* objective since it matches deterministic outputs of the logging system to outputs in the n-best list of the target system. For optimization under deterministic logging, a sampling bias is unavoidable since objective (2) does not correct it by importance sampling. Furthermore, the DPM estimator may show a degenerate behavior in learning. This problem can be remedied by the use of control variates, as we will discuss in Section 5.

Learning Principle: Doubly Controlled Empirical Risk Minimization. Our first modification of Equation (2) has been originally motivated by the use of weighted importance sampling in inverse propensity scoring because of

[1] We will use both terms, reward and loss, in order to be consistent with the respective literature.

$\nabla \hat{R}_{\text{DPM}} = \frac{1}{n}\sum_{t=1}^{n} \delta_t \pi_w(y_t \mid x_t)\nabla \log \pi_w(y_t \mid x_t).$
$\nabla \hat{R}_{\text{DPM+R}} = \frac{1}{n}\sum_{t=1}^{n} [\delta_t \bar{\pi}_w(y_t \mid x_t)(\nabla \log \pi_w(y_t \mid x_t) - \sum_{u=1}^{n} \bar{\pi}_w(y_u \mid x_u)\nabla \log \pi_w(y_u \mid x_u))].$
$\nabla \hat{R}_{\hat{c}\text{DC}} = \frac{1}{n}\sum_{t=1}^{n} [(\delta_t - \hat{c}\hat{\delta}_t)\bar{\pi}_w(y_t \mid x_t)(\nabla \log \pi_w(y_t \mid x_t) - \sum_{u=1}^{n} \bar{\pi}_w(y_u \mid x_u)\nabla \log \pi_w(y_u \mid x_u))$ $+\hat{c}\sum_{y\in\mathcal{Y}(x_t)} \hat{\delta}(x_t, y)\pi_w(y \mid x_t)\nabla \log \pi_w(y \mid x_t)].$

Table 1: Gradients of counterfactual objectives.

its observed stability and variance reduction effects (Precup et al., 2000; Jiang and Li, 2016; Thomas and Brunskill, 2016). We call this objective the *reweighted deterministic propensity matching (DPM+R)* objective:

$$\begin{aligned}\hat{R}_{\text{DPM+R}}(\pi_w) &= \frac{1}{n}\sum_{t=1}^{n} \delta_t \bar{\pi}_w(y_t|x_t) \qquad (3)\\ &= \frac{1}{n}\sum_{t=1}^{n} \delta_t \frac{\pi_w(y_t|x_t)}{\sum_{t=1}^{n} \pi_w(y_t|x_t)}.\end{aligned}$$

From the perspective of Monte Carlo simulation, the advantage of this modification can be explained by viewing reweighting as a multiplicative control variate (Swaminathan and Joachims, 2015b). Let $Z = \delta_t \pi_w(y_t|x_t)$ and $W = \pi_w(y_t|x_t)$ be two random variables, then the variance of $r = \frac{\frac{1}{n}\sum_{t=1}^{n} Z}{\frac{1}{n}\sum_{t=1}^{n} W}$ can be approximately written as follows (Kong, 1992): $\text{Var}(r) \approx \frac{1}{n}(r^2\text{Var}(W) + \text{Var}(Z) - 2r\,\text{Cov}(W, Z))$. This shows that a positive correlation between the variable W, representing the target model probability, and the variable Z, representing the target model scaled by the task loss function, will reduce the variance of the estimator. Since there are exponentially many outputs to choose from for each input during logging, variance reduction is useful in counterfactual learning even in the deterministic case. Under a stochastic logging policy, a similar modification can be done to objective (1) by reweighting the ratio $\rho_t = \frac{\pi_w(y_t|x_t)}{\pi_0(y_t|x_t)}$ as $\bar{\rho}_t = \frac{\rho_t}{\sum_t \rho_t}$. We will use this reweighted IPS objective, called IPS+R, in our comparison experiments that use stochastically logged data.

A further modification of Equation (3) is motivated by the incorporation of a direct reward estimation method in the inverse propensity scorer as proposed in the doubly-robust estimator (Dudik et al., 2011; Jiang and Li, 2016; Thomas and Brunskill, 2016). Let $\hat{\delta}(x_t, y_t)$ be a regression-based reward model trained on the logged data, and let $\hat{c}$ be a scalar that allows to optimize the estimator for minimal variance (Ross, 2013). We define a *doubly controlled* empirical risk minimization objective $\hat{R}_{\hat{c}\text{DC}}$ as follows (for $\hat{c} = 1$ we arrive at a similar objective called $\hat{R}_{\text{DC}}$):

$$\begin{aligned}\hat{R}_{\hat{c}\text{DC}}(\pi_w) =& \frac{1}{n}\sum_{t=1}^{n} \Big[(\delta_t - \hat{c}\hat{\delta}_t)\ \bar{\pi}_w(y_t|x_t) \qquad (4)\\ &+ \hat{c} \sum_{y\in\mathcal{Y}(x_t)} \hat{\delta}(x_t, y)\ \pi_w(y|x_t)\Big].\end{aligned}$$

From the perspective of Monte Carlo simulation, the doubly robust estimator can be seen as variance reduction via additive control variates (Ross, 2013). Let $X = \delta_t$ and $Y = \hat{\delta}_t$ be two random variables. Then $\bar{Y} = \sum_{y\in\mathcal{Y}(x_t)} \hat{\delta}(x_t, y)\ \pi_w(y|x_t)$ is the expectation[2] of Y, and Equation (4) can be rewritten as $\mathbb{E}_{\bar{\pi}_w(x)}(X - \hat{c}\,Y) + \hat{c}\,\bar{Y}$. The variance of the term $X - \hat{c}\,Y$ is $\text{Var}(X - \hat{c}\,Y) = \text{Var}(X) + \hat{c}^2\text{Var}(Y) - 2\hat{c}\,\text{Cov}(X, Y)$. (Ross (2013), Chap. 9.2). Again this shows that variance of the estimator can be reduced if the variable X, representing the reward function, and the variable Y, representing the regression-based reward model, are positively correlated. The optimal scalar parameter $\hat{c}$

[2] Note that we introduce a slight bias by using π_w versus $\bar{\pi}_w$ in sampling probability and control variate.

can be derived easily by taking the derivative of variance term, leading to

$$\hat{c} = \frac{\mathrm{Cov}(X,Y)}{\mathrm{Var}(Y)}. \quad (5)$$

In case of stochastic logging the reweighted target probability $\bar{\pi}_w(y_t|x_t)$ is replaced by a reweighted ratio $\bar{\rho}_t$. We will use such reweighted models of the original doubly robust model, with and without optimal $\hat{c}$, called DR and $\hat{c}$DR, in our experiments that use stochastic logging.

Learning Algorithms. Applying a stochastic gradient descent update rule $w_{t+1} = w_t - \eta \nabla \hat{R}(\pi_w)_t$ to the objective functions defined above leads to a variety of algorithms. The gradients of the objectives can be derived by using the score function gradient estimator (Fu, 2006) and are shown in Table 1. Stochastic gradient descent algorithms apply to any differentiable policy π_w, thus our methods can be applied to a variety of systems, including linear and non-linear models. Since previous work on off-policy methods in RL and contextual bandits has been done in the area of linear classification, we start with an adaptation of off-policy methods to linear SMT models in our work. We assume a Gibbs model

$$\pi_w(y_t|x_t) = \frac{e^{\alpha(w^\top \phi(x_t,y_t))}}{\sum_{y \in \mathcal{Y}(x_t)} e^{\alpha(w^\top \phi(x_t,y))}}, \quad (6)$$

based on a feature representation $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$, a weight vector $w \in \mathbb{R}^d$, and a smoothing parameter $\alpha \in \mathbb{R}^+$, yielding the following simple derivative $\nabla \log \pi_w(y_t|x_t) = \alpha\big(\phi(x_t, y_t) - \sum_{y \in \mathcal{Y}(x_t)} \phi(x_t, y)\pi_w(y_t|x_t)\big)$.

4 Experiments

Setup. In our experiments, we aim to simulate the following scenario: We assume that it is possible to divert a small fraction of the user interaction traffic for the purpose of policy evaluation and to perform stochastic logging on this small data set. The main traffic is assumed to be logged deterministically, following a conservative regime where one-best translations are used for an SMT system that does not change frequently over time. Since our experiments are simulation studies, we will additionally perform stochastic logging, and compare policy learning for the (realistic) case of deterministic logging with the (theoretically motivated) case of stochastic logging.

	TED DE-EN	News FR-EN
train	122k	30k
validation	3k	1k
test	3k	2k

Table 2: Number of sentences for in-domain data splits of SMT train, validation, and test data.

In our deterministic-based policy learning experiments, we evaluate the empirical risk minimization algorithms derived from objectives (3) (DPM+R) and (4). For the doubly controlled objective we employ two variants: First, $\hat{c}$ is set to 1 as in (Dudik et al., 2011) (DC). Second, we calculate $\hat{c}$ as described in Equation (5) ($\hat{c}$DC). The algorithms used in policy evaluation and for stochastic-based policy learning are variants of these objectives that replace $\bar{\pi}$ by $\bar{\rho}$ to yield estimators IPS+R, DR, and $\hat{c}$DR of the expected loss.

All objectives will be employed in a domain adaptation scenario for machine translation. A system trained on out-of-domain data will be used to collect feedback on in-domain data. This data will serve as the logged data $\mathcal{D}$ in the learning experiments. We conduct two SMT tasks with hypergraph re-decoding: The first is German-to-English and is trained using a concatenation of the Europarl corpus (Koehn, 2005), the Common Crawl corpus[3] and the News Commentary corpus (Koehn and Schroeder, 2007). The goal is to adapt the trained system to the domain of transcribed TED talks using the TED parallel corpus (Tiedemann, 2012). A second task uses the French-to-English Europarl data

[3] http://www.statmt.org/wmt13/training-parallel-commoncrawl.tgz

with the goal of domain adaptation to news articles with the News Commentary corpus (Koehn and Schroeder, 2007). We split off two parts from the TED corpus to be used as validation and test data for the learning experiments. As validation data for the News Commentary corpus we use the splits provided at the WMT shared task, namely `nc-devtest2007` as validation data and `nc-test2007` as test data. An overview of the data statistics can be seen in Table 2.

As baseline, an out-of-domain system is built using the SCFG framework CDEC (Dyer et al., 2010) with dense features (10 standard features and 2 for the language model). After tokenizing and lowercasing the training data, the data were word aligned using CDEC's `fast_align`. A 4-gram language model is build on the target languages for the out-of-domain data using KENLM (Heafield et al., 2013). For News, we additionally assume access to in-domain target language text and train another in-domain language model on that data, increasing the number of features to 14 for News.

The framework uses a standard linear Gibbs model whose distribution can be peaked using a parameter α (see Equation (6)): Higher value of α will shift the probability of the one-best translation closer to 1 and all others closer to 0. Using $\alpha > 1$ during training will promote to learn models that are optimal when outputting the one-best translation. In our experiments, we found $\alpha = 5$ to work well on validation data.

Additionally, we tune a system using CDEC's MERT implementation (Och, 2003) on the in-domain data with their references. This full-information in-domain system conveys the best possible improvement using the given training data. It can thus be seen as the oracle system for the systems which are learnt using the same input-side training data, but have only bandit feedback available to them as a learning signal. All systems are evaluated using the corpus-level BLEU metric (Papineni et al., 2002).

The logged data $\mathcal{D}$ is created by translating the in-domain training data of the corpora using the original out-of-domain systems, and logging the one-best translation. For the stochastic experiments, the translations are sampled from the model distribution. The feedback to the logged translation is simulated using the reference and sentence-level BLEU (Nakov et al., 2012).

	TED	News
macro avg.	0.67	0.23
micro avg.	15.03	10.87

Table 3: Evaluation of regression-based reward estimation by average BLEU differences between estimated and true rewards.

		IPS+R	DR	$\hat{c}$DR
TED	avg. estimate	+4.00	+7.98	+6.07
	std. dev.	0.64	3.83	2.06
News	avg. estimate	-7.78	+6.63	+0.95
	std. dev.	0.97	4.13	2.33

Table 4: Policy evaluation by macro averaged difference between estimated and ground truth BLEU on 10k stochastically logged data, averaged over 5 runs.

Direct Reward Estimation. When creating the logged data $\mathcal{D}$, we also record the feature vectors of the translations to train the direct reward estimate that is needed for ($\hat{c}$)DC. Using the feature vector as input and the per-sentence BLEU as the output value, we train a regression-based random forest with 10 trees using scikit-learn (Pedregosa et al., 2011). To measure performance, we perform 5-fold cross-validation and measure the macro average between estimated rewards and the true rewards from the log: $|\frac{1}{n}\sum\delta(x_t, y_t) - \frac{1}{n}\sum\hat{\delta}(x_t, y_t)|$. We also report the micro average which quantifies how far off one can expect the model to be for a random sample: $\frac{1}{n}\sum|\delta(x_t, y_t) - \hat{\delta}(x_t, y_t)|$. The final model used in the experiments is trained on the full training data. Cross-validation results for the regression-based direct reward model can be found in Table 3.

Policy Evaluation. Policy evaluation aims to use the logged data $\mathcal{D}$ to estimate the performance of the target system π_w. The small logged data $\mathcal{D}_{eval}$ that is diverted for policy evaluation is created by translating only 10k sentences of the in-domain training data with the out-of-domain system and sample translations according to the model probability. Again we record the sentence-level BLEU as the feedback. The reference translations that also exist for those 10k sentences are used to measure the ground truth BLEU value for translations using the full-information in-domain system. The goal of evaluation is to achieve a value of IPS+R, DR, and $\hat{c}$ DR on $\mathcal{D}_{eval}$ that are as close as possible to the ground truth BLEU value.

To be able to measure variance, we create five folds of $\mathcal{D}_{eval}$, differing in random seeds. We report the average difference between the ground truth BLEU score and the value of the log-based policy evaluation, as well as the standard deviation in Table 4. We see that IPS+R underestimates the BLEU value by 7.78 on News. DR overestimates instead. $\hat{c}$ DR achieves the closest estimate, overestimating the true value by less than 1 BLEU. On TED, all policy evaluation results are overestimates. For the DR variants the overestimation result can be explained by the random forests' tendency to overestimate. Optimal $\hat{c}$ DR can correct for this, but not always in a sufficient way.

Policy Learning. In our learning experiments, learning starts with the weights w_0 from the out-of-domain model. As this was the system that produced the logged data $\mathcal{D}$, the first iteration will have the same translations in the one-best position. After some iterations, however, the translation that was logged may not be in the first position any more. In this case, the n-best list is searched for the correct translation. Due to speed reasons, the scores of the translation system are normalized to probabilities using the first 1,000 unique entries in the n-best list, rather than using the full hypergraph. Our experiments showed that this did not impact the quality of learning.

In order for the multiplicative control variate to be effective, the learning procedure has to utilize mini-batches. If the mini-batch size is chosen too small, the estimates of the control variates may not be reliable. We test mini-batch sizes of 30k and 10k examples, whereas 30k on News means that we perform batch training since the mini-batch spans the entire training set. Mini-batch size β and early stopping point where selected by choosing the setup and iteration that achieved the highest BLEU score on the one-best translations for the validation data. The learning rate η was selected in the same way, whereas the possible values were $1\mathrm{e}{-4}, 1\mathrm{e}{-5}, 1\mathrm{e}{-6}$ or, alternatively, Adadelta (Zeiler, 2012), which sets the learning rate on a per-feature basis. The results on both validation and test set are reported in Table 5. Statistical significance of the out-of-domain system compared to all other systems is measured using Approximate Randomization testing (Noreen, 1989).

For the deterministic case, we see that in general DPM+R shows the lowest increase but can still significantly outperform the baseline. An explanation of why DPM+R cannot improve any further, will be addressed separately below. DC yields improvements of up to 1.5 BLEU points, while $\hat{c}$ DC obtains improvements of up to 2 BLEU points over the out-of-domain baseline. In more detail on the TED data, DC can close the gap of nearly 3 BLEU by half between the out-of-domain and the full-information in-domain system. $\hat{c}$ DC can improve by further 0.6 BLEU which is a significant improvement at $p = 0.0017$. Also note that, while $\hat{c}$ DC takes more iterations to reach its best result on the validation data, $\hat{c}$ DC already outperforms DC at the stopping iteration of DC. At this point $\hat{c}$ DC is better by 0.18 BLEU on the validation set and continues to increase until its own stopping iteration. The final results of $\hat{c}$ DC falls only 0.8 BLEU behind the oracle system that had references available during its learning process. Considering the substantial difference in information that both systems had available, this is remark-

			BLEU	BLEU difference			BLEU
			out-of-domain	DPM+R	DC	$\hat{c}$DC	in-domain
deterministic	TED	validation	22.39	+0.59	+1.50	+1.89	25.43
		test	22.76	+0.67	+1.41	+2.02	25.58
	News	validation	24.64	+0.62	+0.99	+1.02	27.62
		test	25.27	+0.94	+1.05	+1.13	28.08
			out-of-domain	IPS+R	DR	$\hat{c}$DR	in-domain
stochastic	TED	validation	22.39	+0.57	+1.92	+1.95	25.43
		test	22.76	+0.58	+2.04	+2.09	25.58
	News	validation	24.64	+0.71	+1.00	+0.71	27.62
		test	25.27	+0.81	+1.18	+0.95	28.08

Table 5: BLEU increases for learning, over the out-of-domain baseline on validation and test set. Out-of-domain is the baseline and starting system and in-domain is the oracle system tuned on in-domain data with references. For the deterministic case, all results are statistically significant at $p \leq 0.001$ with regards to the baseline. For the stochastic case, all results are statistically significant at $p \leq 0.002$ with regards to the baseline, except for IPS+R on the News corpus.

able. The improvements on the News corpus show similar tendencies. Again there is a gap of nearly 3 BLEU to close and with an improvement of 1.05 BLEU points, DC can achieve a notable result. $\hat{c}$DC was able to further improve on this but not as successfully as was the case for the TED corpus. Analyzing the actual $\hat{c}$ values that were calculated in both experiments allows us to gain an insight as to why this was the case: For TED, $\hat{c}$ is on average 1.35. In the case of News, however, $\hat{c}$ has a maximum value of 1.14 and thus stays quite close to 1, which would equate to using DC. It is thus not surprising that there is no significant difference between DC and $\hat{c}$DC.

Comparison to the Stochastic Case. Even if not realistic for commercial applications of SMT, our simulation study allows us to stochastically log large amounts of data in order to compare learning from deterministic logs to the standard case. As shown in Table 5, the relations between algorithms and even the absolute improvements are similar for stochastic and deterministic logging. Significance tests between each deterministic/stochastic experiment pair show a significant difference only in case of DC/DR on TED data. However, the DR result still does not significantly outperform the best deterministic objective on TED ($\hat{c}$DC). The p values for all other experiment pairs lie above 0.1. From this we can conclude that it is indeed an acceptable practice to log deterministically.

5 Analysis

Langford et al. (2008) show that counterfactual learning is impossible unless the logging system sufficiently explores the output space. This condition is seemingly not satisfied if the logging systems acts according to a deterministic policy. Furthermore, since techniques such as “exploration over time” (Strehl et al., 2010) are not applicable to commercial SMT systems that are not frequently changed over time, the case of counterfactual learning for SMT seems hopeless. However, our experiments present evidence to the contrary. In the following, we present an analysis that aims to explain this apparent contradiction.

Implicit Exploration. In an experimental comparison between stochastic and deterministic logging for bandit learning in computational advertising, Chapelle and Li (2011) observed that varying contexts (representing user and page visited) induces enough exploration into ad selection such that learning becomes possible. A similar implicit exploration can also be attributed to the case of SMT: An identical input word or phrase can lead, depending on the other words and phrases in the input sentence, to different output words and phrases. Moreover, an identical output word or phrase can appear in different output sentences. Across the entire log, this implicitly performs the exploration on phrase translations that seems to be missing at first glance.

Smoothing by Multiplicative Control Variates. The DPM estimator can show a degenerate behavior in that the objective can be minimized simply by setting the probability of every logged data point to 1.0. This over-represents logged data that received low rewards, which is undesired. Furthermore, systems optimized with this objective cannot properly discriminate between the translations in the output space. This can be seen as a case of translation invariance of the objective, as has been previously noted by Swaminathan and Joachims (2015b): Adding a small constant c to the probability of every data point in the log increases the overall value of the objective without improving the discriminative power between high-reward and low-reward translations.

DPM+R solves the degeneracy of DPM by defining a probability distribution over the logged data by reweighting via the multiplicative control variate. After reweighting, the objective value will decrease if the probability of a low-reward translation increased, as it takes away probability mass from other, higher reward samples. Because of this trade-off, balancing the probabilities over low-reward and high-reward samples becomes important, as desired.

Smoothing by Additive Control Variates. Despite reweighting, DPM+R can still show a degenerate behavior by setting the probabilities of only the highest-reward samples to 1.0, while avoiding all other logged data points. This clearly hampers the generalization ability of the model since inputs that have been avoided in training will not receive a proper ranking of their translations.

The use of an additive control variate can solve this problem by using a reward estimate that takes the full output space into account. The objective will now be increased if the probability of translations with high estimated reward is increased, even if they were not seen in training. This will shift probability mass to unseen data with high estimated-reward, and thus improve the generalization ability of the model.

6 Conclusion

In this paper, we showed that off-policy learning from deterministic bandit logs for SMT is possible if smoothing techniques based on control variates are used. These techniques will avoid degenerate behavior in learning and improve generalization of empirical risk minimization over logged data. Furthermore, we showed that standard off-policy evaluation is applicable to SMT under stochastic logging policies.

To our knowledge, this is the first application of counterfactual learning to a complex structured prediction problem like SMT. Since our objectives are agnostic of the choice of the underlying model π_w, it is also possible to transfer our techniques to non-linear models such as neural machine translation. This will be a desideratum for future work.

Acknowledgments

The research reported in this paper was supported in part by the German research foundation (DFG), and in part by a research cooperation grant with the Amazon Development Center Germany.

References

Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipanakar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14:3207–3260.

Sébastian Bubeck and Nicolò Cesa-Bianchi. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122.

Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of Thompson sampling. In *Advances in Neural Information Processing Systems (NIPS)*, Granada, Spain.

Miroslav Dudik, John Langford, and Lihong Li. 2011. Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, Bellevue, WA.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden.

Michael C. Fu. 2006. Gradient estimation. In S.G. Henderson and B.L. Nelson, editors, *Handbook in Operations Research and Management Science*, volume 13, pages 575–616.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sofia, Bulgaria.

Nan Jiang and Lihong Li. 2016. Doubly robust off-policy value evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, New York, NY.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the Machine Translation Summit*, Phuket, Thailand.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Workshop on Machine Translation (WMT)*, Prague, Czech Republic.

Augustine Kong. 1992. A note on importance sampling using standardized weights. Technical Report 348, Department of Statistics, University of Chicago, Illinois.

Julia Kreutzer, Artem Sokolov, and Stefan Riezler. 2017. Bandit structured prediction for neural sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver, Canada.

John Langford, Alexander Strehl, and Jennifer Wortman. 2008. Exploration scavenging. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, Helsinki, Finland.

Lihong Li, Shunbao Chen, Jim Kleban, and Ankur Gupta. 2015. Counterfactual estimation and optimization of click metrics in search engines: A case study. In *Proceedings of the International World Wide Web Conference (WWW)*, Florence, Italy.

Preslav Nakov, Francisco Guzmán, and Stephan Vogel. 2012. Optimizing for sentence-level bleu+1 yields short translations. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, Bombay, India.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses: An Introduction*. Wiley, New York.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the Human Language Technology Conference and the 3rd Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, Edmonton, Cananda.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL)*, Stroudsburg, PA.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Doina Precup, Richard S. Sutton, and Satinder P. Singh. 2000. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth*

International Conference on Machine Learning (ICML), San Francisco, CA.

Paul R. Rosenbaum and Donald B. Rubin. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55.

Sheldon M. Ross. 2013. *Simulation*, fifth edition. Elsevier.

Artem Sokolov, Julia Kreutzer, Christopher Lo, and Stefan Riezler. 2016. Stochastic structured prediction under bandit feedback. In *Advances in Neural Information Processing Systems (NIPS)*, Barcelona, Spain.

Alexander L. Strehl, John Langford, Lihong Li, and Sham M. Kakade. 2010. Learning from logged implicit exploration data. In *Advances in Neural Information Processing Sytems (NIPS)*, Vancouver, Canada.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning. An Introduction*. The MIT Press.

Adith Swaminathan and Thorsten Joachims. 2015a. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research*, 16:1731–1755.

Adith Swaminathan and Thorsten Joachims. 2015b. The self-normalized estimator for counterfactual learning. In *Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada.

Philip S. Thomas and Emma Brunskill. 2016. Data-efficient off-policy policy evaluation for reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, New York, NY.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC)*, Istanbul, Turkey.

Matthew D. Zeiler. 2012. ADADELTA: An adaptive learning rate method. ArXiv:1212.5701 [cs.LG].

Learning Fine-grained Relations from Chinese User Generated Categories

Chengyu Wang, Yan Fan, Xiaofeng He[*], Aoying Zhou
Shanghai Key Laboratory of Trustworthy Computing,
School of Computer Science and Software Engineering, East China Normal University
{chywang2013,eileen940531}@gmail.com
{xfhe,ayzhou}@sei.ecnu.edu.cn

Abstract

User generated categories (UGCs) are short texts that reflect how people describe and organize entities, expressing rich semantic relations implicitly. While most methods on UGC relation extraction are based on pattern matching in English circumstances, learning relations from Chinese UGCs poses different challenges due to the flexibility of expressions. In this paper, we present a weakly supervised learning framework to harvest relations from Chinese UGCs. We identify is-a relations via word embedding based projection and inference, extract non-taxonomic relations and their category patterns by graph mining. We conduct experiments on Chinese Wikipedia and achieve high accuracy, outperforming state-of-the-art methods.

1 Introduction

UGCs are descriptive phrases related to entities, frequently appearing in online encyclopedias and vertical websites. These texts are concise and informative, reflecting the way people organize and characterize entities (Xu et al., 2016a).

UGCs (especially Wikipedia categories) are important sources for knowledge harvesting. Previous approaches (Flati et al., 2014; Ponzetto and Strube, 2007; Ponzetto and Navigli, 2009) focus on inferring is-a relations between entities and UGCs for taxonomy construction. A few others extract multiple types of relations from Wikipedia categories (Nastase and Strube, 2008; Suchanek et al., 2007). These methods are mostly designed for English language by employing language-specific patterns or linguistic rules.

For Chinese, harvesting semantic relations from texts poses different challenges. There is no distinction between singular and plural forms and no word spaces in Chinese. Word orders can be arranged in multiple ways with very flexible expressions. As illustrated in Qiu and Zhang (2014); Chen et al. (2014), the research of relation extraction from Chinese texts makes less significant process than the research for English. Although several approaches are proposed to construct Chinese taxonomies from Wikipedia categories (Li et al., 2015; Wang et al., 2014), extracting fine-grained and multi-typed relations from UGCs still needs further study. This is because there exist very few high-quality lexical patterns for relation identification in Chinese UGCs (in contrast to Nastase and Strube (2008); Suchanek et al. (2007)). Hence this problem is similar to "open relation extraction" (Etzioni et al., 2011) from Chinese short texts, without pre-defined relation types.

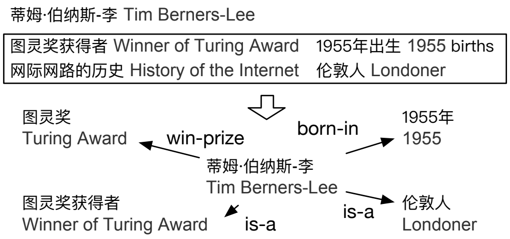


Figure 1: An illustrative example with respect to "Tim Berners-Lee" in Chinese Wikipedia.

In this paper, we propose a weakly supervised learning framework to mine fine-grained and multiple-typed relations from Chinese UGCs. A simple example is illustrated in Figure 1[1]. Inspired by Fu et al. (2014); Wang et al. (2017), is-a relations are extracted based on word embedding

[*] Corresponding author.

[1] The category "Winner of Turing Award" can serve as a class of "Tim Berners-Lee" (similar to Wu et al. (2012)) and be treated as a relational category (similar to Suchanek et al. (2007)). We regard both are valid and extract two relations.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2577–2587
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

based projection models. We further refine prediction results by collective inference and hypernym expansion. For non-taxonomic relations, relation types and corresponding category patterns are identified jointly based on graph clique mining. Finally, these mined "raw" relations are mapped to canonicalized relation triples. In our work, except for a set of heuristic rules, the proposed approach is weakly supervised without manual labeling.

In the experiments, given only 0.6M entities and their respective 2.4M categories in Chinese Wikipedia, our method extracts 1.52M relations with an overall accuracy of 93.6%. The experiments also show that our approach outperforms previous methods for both is-a and non-taxonomic relation extraction from Chinese UGCs. The extracted relations and the labeled test set are publicly available[2].

The rest of this paper is as follows. Section 2 summarizes related work. Details of our approach are described in Section 3 to Section 5, with experiments in Section 6. Finally, we conclude our paper and discuss the future work in Section 7.

2 Related Work

In this section, we overview the related work on relation extraction from UGCs.

2.1 Is-a Relation Extraction

Is-a relations are backbones in taxonomies. In YAGO (Suchanek et al., 2007), a Wikipedia category is regarded as conceptual if it matches the pattern "pre-modifier + head word + post-modifier". WikiTaxonomy (Ponzetto and Strube, 2007) constructs a taxonomy from Wikipedia categories using multiple types of features. The taxonomy is reconstructed and improved in Ponzetto and Navigli (2009). Other similar projects use classifiers and rule based inference to predict is-a relations for taxonomy learning (Flati et al., 2014; Mahdisoltani et al., 2015; Nastase et al., 2010; Alfarone and Davis, 2015; Shwartz et al., 2016; Gupta et al., 2016). Since harvesting English is-a relations is not our focus, we do not elaborate here.

For Chinese, this task is more challenging because there are few category patterns that can be used to extract is-a relations from UGCs. Based on the word formation of Wikipedia categories, Li et al. (2015) propose a classification method to build a large Chinese taxonomy from Wikipedia. A similar approach is presented in Lu et al. (2015). Besides encyclopedias, Fu et al. (2013) generate candidate hypernyms and employ an SVM-based ranking model to detect the most likely hypernym of an entity. These methods have relatively high precision but require careful feature engineering and a large amount of human work.

Another thread of related work is cross-lingual approaches, which use larger English knowledge sources to supervise Chinese is-a relations extraction. For example, Wang et al. (2014) propose a dynamic adaptive boosting model to learn taxonomic prediction functions for English and Chinese. Xu et al. (2016b) link Chinese entities with DBpedia types based on cross-lingual links between Chinese and English entities. Other approaches can be found in Wu et al. (2016); Mahdisoltani et al. (2015). These methods take advantages of languages with richer resources but are constrained by cross-lingual links.

To capture linguistic regularities of is-a relations, deep learning approaches map the vectors of entities to the vectors of their hypernyms. Fu et al. (2014) design piecewise linear projection models to learn Chinese semantic hierarchies based on word embeddings (Mikolov et al., 2013). Wang and He (2016) improve this approach by adding an iterative update strategy and a pattern-based validation mechanism. Wang et al. (2017) design a transductive learning approach by considering the semantics of both is-a and not-is-a relations, linguistic rules and the unlabeled data jointly. In this work, we further propose a word embedding based model that consider the word formation of UGCs to improve the prediction results.

2.2 Non-taxonomic Relation Extraction

Unlike the case of is-a relations, the task of extracting non-taxonomic relations from UGCs has rarely been addressed. A possible cause is that harvesting relations from short texts is more challenging. The pioneer work Nastase and Strube (2008) extracts relations by lexical pattern matching and inference. Pasca (2017) studies how to decompose Wikipedia categories into attribute-value pairs. YAGO (Suchanek et al., 2007) uses regular expression based matching to harvest relations. While patterns in English are more regular, enumerating patterns for Chinese requires a large amount of human labor. In our work, we solve this problem by graph mining, which has high pre-

[2] https://chywang.github.io/data/emnlp17.zip

cision and requires minimal human intervention. Note that our work is also similar to open relation extraction (Etzioni et al., 2011) due to the unknown number of relation types. The difference is that our work focuses on UGCs which are very short phrases rather than sentences.

3 General Framework

In Wikipedia, each entity e is associated with a collection of UGCs $Cat(e)$. We first learn a prediction model $f(e, c)$ to distinguish is-a relations from not-is-a relations where $c \in Cat(e)$, and extract all is-a relations (Section 4). For example, we can obtain is-a relations "(Tim Berners-Lee, is-a, Londoner)" and "(Tim Berners-Lee, is-a, Winner of Turing Award)", as shown in Figure 1.

After that, we mine non-taxonomic relations from Wikipedia UGCs (Section 5). Our algorithm first makes a single pass over all categories to mine significant category patterns (Section 5.1). For example, the pattern "[E]获得者(Winner of [E])" is extracted, which frequently appears in UGCs and may refers to a type of relation where "[E]" is a placeholder for entities. Candidate relation instances for such patterns are obtained by a graph clique mining algorithm (Section 5.2). The instances extracted based on the previous pattern are "(Tim Berners-Lee, Turing Award)", "(Albert Einstein, Nobel Prize for Physics)", etc. Finally, the extracted "raw" instances are mapped to canonicalized triples (Section 5.3). In this step, a relation predicate "win-prize" is defined for the pattern and these pairs are mapped to "win-prize" relations.

4 Mining Is-a Relations

In this section, we introduce how to learn $f(e, c)$ and extract is-a relations from UGCs.

4.1 Training Data Generation

The training of $f(e, c)$ requires positive and negative entity-category pairs. To avoid the time-consuming labeling process, we generate the training set automatically. The first part is borrowed from Fu et al. (2014), containing 1,391 positive pairs and 4,294 negative pairs. However, the number of positive pairs is not sufficient for our propose. We design a heuristic rule to generate more positive pairs from Wikipedia categories. We treat a pair (e, c) as positive if the following two conditions hold:

- The category c matches the pattern "premodifier + 的+ head word" or the head words of e and c are the same[3].
- The head word of a category name is a noun and is *not* in a Chinese thematic lexicon extended from the dictionary used in Li et al. (2015), containing 184 thematic words (e.g., "军事(Military)", "娱乐(Entertainment)".

In total, we sample 5,000 pairs to add to our training set. The TP rate is 98.7%, estimated over 300 pairs, indicating the effectiveness of rules.

4.2 Projection-based Model Prediction

Except for the previous pattern, other Chinese is-a relations can not be directly extracted by lexical matching. Inspired by Wang et al. (2017), we employ projection models to learn the semantics of is-a and not-is-a relations.

A projection model is a linear model that maps the embedding vector of a word to the vector of another where the two words satisfy a particular relation (Fu et al., 2014). In Wikipedia, most category names are relatively long and fine-grained, making it difficult to learn the embeddings precisely. We find that given a pair (e, c), if the head word of category c is a valid hypernym of e, so it is for c itself, e.g., "英格兰计算机科学家(CS scientist in England)" for "Tim Berners-Lee". Denote $\vec{v}(e)$ as the embedding vector of entity e, with the dimensionality as n. Let c_h be the head word of c. For each pair in the positive training set $(e, c) \in D^+$, assume there is a positive projection model such that $\mathbf{M}^+\vec{v}(e) + \mathbf{B}^+ \approx \vec{v}(c_h)$ where $\mathbf{M}^+$ is an $n \times n$ projection matrix and $\mathbf{B}^+$ is an $n \times 1$ bias vector. Similarly, for pairs in negative training set $(e', c') \in D^-$, we learn a negative model $\mathbf{M}^-\vec{v}(e') + \mathbf{B}^- \approx \vec{v}(c'_h)$. Note that we do not impose explicit connections between two models because the semantics of Chinese is-a and not-is-a relations are very complicated and difficult to model (Fu et al., 2014; Wang and He, 2016). In our work, we let the algorithms to learn representations of is-a/not-is-a relations.

This approach learns is-a and not-is-a relation representations implicitly and does not require deep NLP analysis on UGCs, which is suitable to deal with the flexible expressions in Chinese. In

[3] The head word of a category name is the root word in the dependency parsing tree. "的" is an auxiliary word in Chinese, usually appearing between adjectives and nouns.

the training phase, we aim to minimize the objective function for positive projection learning:

$$J(\mathbf{M}^+, \mathbf{B}^+) = \sum_{(e,c)\in D^+} \|\mathbf{M}^+\vec{v}(e) + \mathbf{B}^+ - \vec{v}(c_h)\|_F^2 + \frac{\lambda}{2}\|\mathbf{M}^+\|_F^2 + \frac{\lambda}{2}\|\mathbf{B}^+\|_F^2$$

where $\lambda > 0$ gives an additional Tikhonov smoothness effect on the projection matrices (Golub et al., 1999). For negative model, we have

$$J(\mathbf{M}^-, \mathbf{B}^-) = \sum_{(e,c)\in D^-} \|\mathbf{M}^-\vec{v}(e) + \mathbf{B}^- - \vec{v}(c_h)\|_F^2 + \frac{\lambda}{2}\|\mathbf{M}^-\|_F^2 + \frac{\lambda}{2}\|\mathbf{B}^-\|_F^2$$

After model training, for an unlabeled pair (e, c), if the category c is the correct hypernym of the entity e, the vector $\vec{v}(c_h)$ will be close to $\mathbf{M}^+\vec{v}(e) + \mathbf{B}^+$ and far away from $\mathbf{M}^-\vec{v}(e) + \mathbf{B}^-$. Denote $d^+(e, c)$ and $d^-(e, c)$ as:

$$d^+(e,c) = \|\mathbf{M}^+\vec{v}(e) + \mathbf{B}^+ - \vec{v}(c_h)\|_2$$

$$d^-(e,c) = \|\mathbf{M}^-\vec{v}(e) + \mathbf{B}^- - \vec{v}(c_h)\|_2$$

The prediction score is calculated as follows:

$$s(e,c) = \tanh(d^-(e,c) - d^+(e,c))$$

where $s(e, c) \in (-1, 1)$. High prediction score means a large probability of the existence of an is-a relation between e and c.

4.3 Collective Prediction Refinement

As indicated in Fu et al. (2013); Levy et al. (2015), some categories naturally serve as “prototypical hypernyms”, regardless of the entities. To encode this assumption into our method, we refine the previous prediction results by collective inference.

Consider the category “伦敦人(Londoner)” in Figure 1, which can be literally translated as “伦敦(London)人(person)”. “人(person)” is the “prototypical hypernym” here. Other categories whose head words are “人(person)” such as “哥本哈根人(Copenhagen person[4], people from Copenhagen)” are likely to be conceptual categories, too.

Denote H as the head word set of all Wikipedia UGCs. For each $h \in H$, let $D_h = \{(e, c)\}$ be the collection of unlabeled pairs (i.e., pairs not in the training set) where the head word of category c is h. D_h^+ is the collection of positive pairs with h as the head word of c in the training set (generated based on Section 4.1). We define the unnormalized global prediction score $\tilde{g}(h)$ for each $h \in H$:

$$\tilde{g}(h) = \ln(1 + |D_h| + |D_h^+|)\frac{|D_h^+| + \sum_{(e,c)\in D_h} s(e,c)}{|D_h| + |D_h^+|}$$

In this formula, each unlabeled data instance $(e, c) \in D_h$ has the weight of $s(e, c)$ and each training data instance $(e, c) \in D_h^+$ has the weight of 1. $\frac{|D_h^+|+\sum_{(e,c)\in D_h} s(e,c)}{|D_h|+|D_h^+|}$ is the average prediction score for categories with the head word h. $\ln(1 + |D_h| + |D_h^*|)$ gives a larger impact to $\tilde{g}(h)$ when the head word h appears more frequently in Wikipedia categories. This heuristic setting is inspired by transductive learning which takes both training and unlabeled data into consideration (Chapelle et al., 2006). It is also similar to the prior probability feature (Fu et al., 2013).

We normalize the global prediction score $g(h)$ as follows:

$$g(h) = \frac{\tilde{g}(h)}{\max_{h' \in H} |\tilde{g}(h')|}$$

The prediction function $f(e, c)$ for the entity e and the category c with the head word h is defined in a combination of $s(e, c)$ and $g(h)$:

$$f(e,c) = \beta s(e,c) + (1 - \beta) g(h)$$

where $\beta \in (0, 1)$ is a tuning parameter that controls the relative importance of the two scores.

We predict there is an is-a relation between entity e and category $c \in Cat(e)$ if at least one of the two conditions holds:

- (e, c) meets the two conditions in Section 4.1.
- $f(e, c) > \theta$ where θ is a threshold.

Finally, we regard c_h as a valid hypernym of e if c is predicted as a hypernym of e and c_h is also a Wikipedia concept. This step (called hypernym expansion) increases the number of hypernyms and hence the number of is-a relations.[5]

5 Mining Non-taxonomic Relations

In this section, we present our approach to extract non-taxonomic relations from Wikipedia UGCs.

[4] Literal translation.

[5] We do not extract all the entity-head word pairs (e, c_h) as is-a relations because word segmentation, tagging and parsing errors may occur when we extract head words by NLP tools. We observe that if c_h is also a Wikipedia concept, the head word extraction process is most probably correct.

5.1 Single-pass Category Pattern Miner

This module automatically learns important category patterns that appear frequently in Wikipedia and have a probability to represent certain semantic relations. Formally, a category pattern p is an ordered sequence of common words and entity tags. For example, the pattern of the category "图灵奖获得者(Winner of Turing Award)" is "[E]获得者(Winner of [E])". Define $R_p = \{(e_p, c_p)\}$ as the collection of entity pairs such that in Wikipedia page e_p, a category containing c_p matches the pattern p[6]. c_p is in the place of "[E]". Consider the previous example. In Wikipedia page "Tim Berners-Lee", there is a category "Winner of Turing Award" that matches the pattern "Winner of [E]". "Turing Award" is the "[E]" here. Thus we have e_p ="Tim Berners-Lee" and c_p ="Turing Award" as an entity pair in R_p. We can see that R_p is the collection of all candidate relation instances that may have the relation that p represents.

Let L_p be the number of common words in pattern p. We define the support of the pattern $supp(p)$ as follows:

$$supp(p) = |R_p| \cdot \ln(1 + L_p)$$

where $\ln(1 + L_p)$ gives larger support values to longer patterns because longer patterns tend to be more specific and may contain richer semantics.

In the implementation, we employ a CRF-based Chinese NER tagger (Qiu et al., 2013) and a dictionary consisting of all Wikipedia entities to recognize the entities and obtain these patterns. This step processes all the categories within a single pass and calculates their support values. It keeps top-k highest support patterns as the input of the next step, together with the matched entity pairs.

5.2 Graph-based Raw Relation Extractor

In this part, for each top-k highest support pattern p, we select a subset of pairs R_p^* from R_p as seed relation instances for an underlying relation that the pattern p may represent. After that, we filter out low quality patterns and extract relation instances $R_p^{'}$ from R_p as the final result.

5.2.1 Seed Relation Instance Extraction

To select seed relation instances R_p^*, we propose an unsupervised graph mining approach. Let $G_p = (C_p, L_p, W_p)$ be a weighted, undirected graph where C_p, L_p and W_p denote vertices, edges and edge weights, respectively. The vertices correspond to the matched entities in categories for pattern p, i.e., $C_p = \{c_p | (e_p, c_p) \in R_p\}$. The edge weights reflect the semantic similarities among entities in C_p. Because the link structure in Chinese Wikipedia is relatively sparse (Wang et al., 2016), we estimate the similarity between entities c_p and $c_p^{'}$ semantically as follows:

$$sim(c_p, c_p^{'}) = \frac{\sum_{c \in Cat(c_p)} \sum_{c^{'} \in Cat(c_p^{'})} \cos(\vec{v}(c_h), \vec{v}(c_h^{'}))}{|Cat(c_p)| \cdot |Cat(c_p^{'})|}$$

where $\cos(\cdot)$ is a cosine function to compute the similarity of two words in the embedding space.

Given a similarity threshold τ, iff $sim(c_p, c_p^{'}) > \tau$, we have $(c_p, c_p^{'}) \in L_p$ and $w(c_p, c_p^{'}) = sim(c_p, c_p^{'})$. In this way, entities in C_p are interconnected if they are similar in semantics.

In this paper, we model the problem of mining R_p^* from R_p as a Maximum Edge Weight Clique Problem (MEWCP) (Alidaee et al., 2007), which detects a maximum edge weight clique C_p^* from C_p in R_p to form R_p^*. Recall that in an undirected graph with edge weights, a maximum edge weight clique is a clique in which the sum of edge weights in the clique is the largest among all the cliques.

To produce a solution for MEWCP, several algorithms have been proposed in the optimization research community, e.g., unconstrained quadratic programming (Alidaee et al., 2007) and the branch-and-cut algorithm (Sørensen, 2004). However, they suffer from high computational complexity due to the NP-Hardness of the problem (Alidaee et al., 2007). In this paper, we introduce an approximate algorithm based on Monte Carlo methods. The general procedure is shown in Algorithm 1. It starts with an empty graph G_p^* to store the clique. In each iteration, it selects an edge $(c_p, c_p^{'})$ from G_p with the probability proportional to its weight $w(c_p, c_p^{'})$. After a particular edge $(c_p, c_p^{'})$ is chosen, the algorithm adds the edge to G_p^*, and removes the edge and other edges that do not connect with any nodes in C_p^* from G_p. This process iterates until no more edges in G_p can be added to G_p^*. Thus, the vertices in G_p^* form the desired clique C_p^*.

Because it is a random, approximate algorithm, the average runtime complexity depends on the input graph structure. We can see that the worst-case runtime complexity is $O(|L_p|^2)$. We run it k

[6] Without ambiguity, we use e_p to represent both the Wikipedia page with the title as e_p and the entity e_p itself.

Algorithm 1 Algorithm for MEWCP

```
Input: Graph $G_p = (C_p, L_p, W_p)$.
Output: Maximum edge weight clique $C_p^*$.
  Initialize temp graph $G_p^* = (C_p^*, L_p^*)$ with $C_p^* = \emptyset$ and $L_p^* = \emptyset$;
  while $L_p \neq \emptyset$ do
    Sample $(c_p, c_p^{'})$ from $L_p$ with prob $\propto w(c_p, c_p^{'})$;
    $C_p = C_p \setminus \{c_p, c_p^{'}\}$, $C_p^* = C_p^* \cup \{c_p, c_p^{'}\}$;
    $L_p = L_p \setminus \{(c_p, c_p^{'})\}$, $L_p^* = L_p^* \cup \{(c_p, c_p^{'})\}$;
    for each $(\tilde{c}_p, \tilde{c}_p^{'}) \in L_p$ do
      if $\tilde{c}_p \notin C_p^*$ and $\tilde{c}_p^{'} \notin C_p^*$ then
        $C_p = C_p \setminus \{\tilde{c}_p, \tilde{c}_p^{'}\}$;
        $L_p = L_p \setminus \{(\tilde{c}_p, \tilde{c}_p^{'})\}$;
      end if
    end for
  end while
  return Maximum edge weight clique $C_p^*$;
```

times and produce multiple results. We select the clique with largest edge weights as the maximum edge weight clique for G_p. The seed relation instance collection is defined as $R_p^* = \{(e_p, c_p) | c_p \in C_p^*, (e_p, c_p) \in R_p\}$. Thus the total runtime complexity is $O(k|L_p|^2)$. In this way, the NP-hard problem is effectively solved in quadratic time.

5.2.2 Relation Extraction and Filtering

After the seed relation instances R_p^* are detected, we employ a confidence score to quantify the quality of pattern p. Intuitively, if pattern p represents entity pairs with the same clear semantic relation, the size of R_p^* and the sum of edge weights in C_p^* will be sufficiently large. Here, we define the confidence score of pattern p as follows:

$$conf(p) = \frac{\ln(1 + |R_p^*|)}{|R_p^*| \cdot (|R_p^*| - 1)} \sum_{c_p, c_p^{'} \in C_p^*, c_p \neq c_p^{'}} sim(c_p, c_p^{'})$$

Based on the formula, patterns with low confidence scores can be filtered. For the remaining patterns, given each $(e_p, c_p) \in R_p$, we add it to the final extracted relation instance collection $R_p^{'}$ if $(e_p, c_p) \in R_p^*$ or it is similar enough to entity pairs in R_p^*. Denote γ as a parameter that controls the precision-recall trade-off. The criteria is:

$$\frac{\sum_{c_p^{'} \in C_p^*} sim(c_p, c_p^{'})}{|C_p^*|} > \frac{\gamma \sum_{c_p^{'}, c_p^{''} \in C_p^*, c_p^{'} \neq c_p^{''}} sim(c_p^{'}, c_p^{''})}{|R_p^*| \cdot (|R_p^*| - 1)}$$

In general, our method detects most probably correct pairs as "seeds" and extract other pairs that are similar enough to seeds. Because it is difficult to ensure high precision for short text relation extraction, we do not use iterative extraction method to avoid "semantic drift" (Carlson et al., 2010).

5.3 Relation Mapping

The final step is to map $R_p^{'}$ to relation triples with a proper relation predicate. Based on category patterns, we have three types of mappings:

Direct Verbal Mapping If the head word of the pattern is a verb, we can use it as the relation predicate. For example, in "[E]出生([E] births)", "出生(born in)" is expressed as a verb in Chinese and is taken as a predicate.

Direct Non-verbal Mapping If the category pattern does not contain a verb but expresses a semantic relation by one/many non-verbs, we define the relation predicate and map the entity pairs to relation triples by logical rules. For example, in the pattern "[E]获得者(Winner of [E])", "获得者(winner)" is a noun that indicates the "得奖(win-prize)" relation.

Indirect Mapping Similar to Suchanek et al. (2007), a few patterns do not describe relations between entity pairs, but should be mapped to other relations indirectly[7]. In "[E]军事([E] military)", it indicates that the entity is related to the topic "军事(military)". Thus, we define a new relation predicate "话题(topic-of)" and establish the relations between entities and "军事(military)".

As seen, the only manual work in our approach is to define relation predicates for direct non-verbal mappings and indirect mappings. In our work, such logical mapping rules are required for only a couple of relation types. Therefore, the proposed approach needs very minimal human work.

6 Experiments

In this section, we conduct experiments to evaluate our method and compare it with state-of-the-art approaches. We also present the overall extraction performance to make the convincing conclusion.

[7] There are also a few is-a relations that are generated by indirect mapping. For example, the pattern "[E]单曲([E] digital single)" infers that the entity that is associated with the category is a song. However, most of the cases are related to non-taxonomic relations.

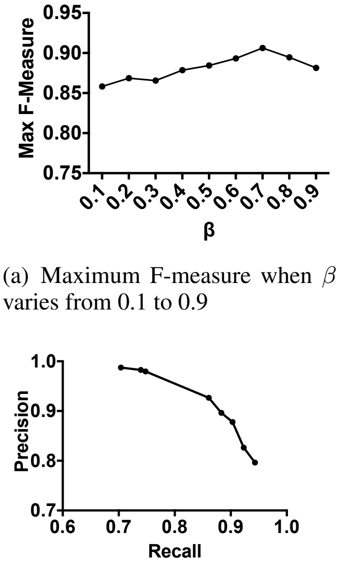
6.1 Data Source and Experimental Settings

The data source is downloaded from the Chinese Wikipedia dump of the version January 20th, 2017[8]. Because some Wikipedia pages are not related to entities, we use heuristic rules to filter out disambiguation, redirect, template and list pages. Finally, we obtain 0.6M entities and 2.4M entity-category pairs. The open-source toolkit FudanNLP (Qiu et al., 2013) is employed for Chinese NLP analysis. The word embeddings are trained via a Skip-gram model using a large corpus from Wang and He (2016) and set to 100 dimensions.

6.2 Is-a Relation Extraction

Test Set Generation We randomly select 2,000 entity-category pairs and ask multiple human annotators to label the relations (i.e., is-a and not-is-a). We discard all the pairs that have inconsistent labels across different annotators and obtain a dataset of 1,788 pairs. 30% of the data are used for parameter tuning and the rest for testing. The dataset is publicly available for research.[9]

Parameter Analysis Two parameters are required to be tuned in our method, i.e., β and θ. We vary the value of β from 0.1 to 0.9. With a fixed value of β, we change the value of θ to achieve the best performance over the development set. Figure 2(a) illustrates the maximum F-measure. Experimental results show our method is generally not very sensitive to the selection of β. When $\beta = 0.7$, it has the highest performance, indicating a good balance between the local and global prediction scores. Additionally, Figure 2(b) illustrates the precision-recall curve with respect to the change of θ when $\beta = 0.7$. The highest F-measure is achieved when we set $\theta = 0.05$.



(a) Maximum F-measure when β varies from 0.1 to 0.9

(b) Precision-recall curve when θ varies from -1 to 1

Figure 2: Parameter analysis.

Method	Precision	Recall	F1
Concat Model	79.5%	64.2%	67.2%
Sum Model	80.9%	70.1%	72.6%
Diff Model	78.3%	69.0%	71.5%
Piecewise Projection	78.9%	72.3%	75.5%
Our Method (w/o Exp)	89.2%	88.1%	88.7%
Our Method	**89.8%**	**88.3%**	**89.0%**

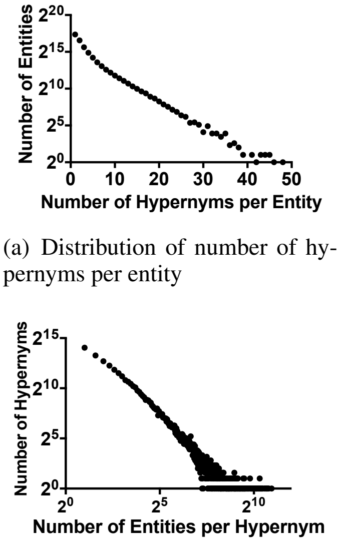
Table 1: Performance comparison over test set.

Comparative Study We set up the following strong baselines to compare our method with state-of-the-art approaches. The experimental results are shown in Table 1. To represent entity-category pairs with word embedding based features, we implement several state-of-the-art methods: the *concat* model $\vec{v}(e) \oplus \vec{v}(c_h)$, the *sum* model $\vec{v}(e) + \vec{v}(c_h)$ and the *diff* model $\vec{v}(e) - \vec{v}(c_h)$ (Baroni et al., 2012; Roller et al., 2014; Mirza and Tonelli, 2016). l_2-regularized logistic regression is trained to make the prediction due to the high performance in previous research. This approach achieves the highest F-measure of 72.6%. We also test the *piecewise projection* model proposed in Wang and He (2016) over Chinese Wikipedia, which is state-of-the-art for predicting is-a relations between Chinese words. It has a slight improvement in performance. As seen, our method without the hypernym expansion step (i.e., "Our Method (w/o Exp)" in Table 1) increases the F-measure by 13.2% (with $p < 0.01$) compared to Wang and He (2016). The full implementation of our method has the F-measure of 89.0%, which shows the effectiveness of our approach.

Overall Results In total, we extract 1.17M is-a relations from Chinese Wikipedia categories, consisting 412K entities and 113K distinct categories. In Figure 3(a), we present how many entities have a particular number of hypernyms. In average, each entity has 2.84 hypernyms. We can see that this distribution fits in a semi-log line, defined by

[8] http://download.wikipedia.com/zhwiki/20170120/

[9] There exist a few public datasets for Chinese is-a relations (Fu et al., 2013, 2014). But they aim to learn is-a relations between short concepts/terms and are not suitable for evaluating our work. We focus on understanding (relatively long) categories for entities.



(a) Distribution of number of hypernyms per entity

(b) Distribution of number of entities per hypernym

Figure 3: Distributional analysis on is-a relations.

Category Pattern	Relation Predicate
[E]校友(Alumni)	毕业(graduated-from)
[E]队教练(Coach)	执教(coach-team)
[E]省市镇 (City/Town in Province)	位于(located-in)
[E]获得者(Winner)	获奖(win-prize)

Table 2: Manually defined relation mappings.

a log scale on the y-axis and a linear scale on the x-axis. Similarly, each hypernym has 10.35 entities in average, with the distribution illustrated in Figure 3(b). The number of entities per hypernym follows the power-law distribution with a long tail.

6.3 Non-taxonomic Relation Extraction

Detailed Steps We first run the single-pass pattern miner and extract the category patterns with top-500 highest support values. This is because only fewer than 20 entities are matched for the rest of the patterns. For each of these patterns, we fix $\tau = 0.7$ and run the MEWCP algorithm three times to ensure the high reliability of the seed relation instances, and select top-250 most confident category patterns. To determine the value of γ, we carry out a preliminary experiment, which samples 200 entity pairs to estimate the accuracy. It shows that even we set γ to a relatively low value (i.e., 0.2), the accuracy is over 90%. Finally, 26 relation types are created automatically based on direct verb mapping. We design the mapping rules and relation predicates for the remaining 16 relation types manually, with examples in Table 2.

Evaluation To evaluate the correctness of extracted relations, we carry out two experimental tests: accuracy test and coverage test. Following Suchanek et al. (2007), in the accuracy test, we randomly sample 200 relation instances for each relation type and ask human annotators to label. We discard the results if human annotators disagree. The coverage test is to determine whether the extracted relations already exist in Chinese knowledge bases. Low coverage score means these relations are not present in existing Chinese knowledge bases. In the experiments, we take CN-DBpedia V2.0 (Xu et al., 2017) as the ground truth knowledge base. Up till February 2017, it contains 41M explicit semantic relations of 9M entities, excluding entity summaries, synonyms, etc. We use the CN-DBpedia API[10] to obtain relations for each entity and report the coverage of relation r as:

$$cov(r) = \frac{\#\text{Matched extractions in CN-DBpedia}}{\#\text{Correct extractions generated by our approach}}$$

For fair comparison, because relations in different knowledge base systems may express differently, we ask human annotators to determine whether the relations extracted by our approach and CN-DBpedia match or not. In Table 3, we present the size, accuracy and coverage values of eight non-taxonomic relations, each with over three thousand relation instances.

From the experimental results, we can see that the accuracy is over 90% for all the eight relations. Especially the accuracy values of some relations are over 98% or even equal to 100%. This means it is reliable to extract relations from Chinese UGCs based category pattern mining. The results of the coverage tests present a large variance among different relations. While some relations such as “born-in” have a relatively high coverage in CN-DBpedia, other relation instances that we extract are rarely present in the knowledge base. Overall, the average coverage is approximately 21.1%. This means although the Chinese knowledge base is relatively large in size, it is far from complete. Furthermore, most relations in Chinese knowledge bases are extracted from infoboxes, in the form of attribute-value pairs (Fang et al., 2016; Niu et al., 2011; Wang et al., 2013). Thus, the knowledge harvested from UGCs can be an important supplementary for these systems.

[10] http://knowledgeworks.cn:20313/cndbpedia/api/entityAVP

Relation	Size	Accu.	Cov.	Relation	Size	Accu.	Cov.
毕业(graduated-from)	44,118	98.0%	22.9%	位于(located-in)	29,460	97.2%	8.5%
建立(established-in)	20,154	95.0%	31.5%	出生(born-in)	11,671	98.3%	41.4%
成员(member-of)	8,445	96.0%	4.2%	启用(open-in)	8,956	98.2%	21.6%
逝世(died-in)	5,597	100.0%	18.4%	得奖(win-prize)	3,262	90.0%	27.3%

Table 3: Size, accuracy and coverage values of eight extracted relation types.

Type	Category Pattern	Example
Member pattern	[E]成员/总统 Member/President of [E]	中国科学院成员 Member of Chinese Academy of Sciences
Verb-NP pattern	[E]+Verb+(的)+Noun Phrase	1990年建立的组织 Organization founded in 1990
Verb pattern	[E]+Verb	1980年出生1980 births

Table 4: Category patterns that we design for CN-WikiRe.

Currently, we only focus on Chinese Wikipedia categories. We will study how to extend our approach to UGCs for other knowledge sources, especially domain-specific sources in the future.

Overall Results In summary, our approach extracts 1.52M relations, including 1.17M is-a relations and 0.36M others. The estimated accuracy values of is-a, other and all relations are 92.2%, 97.4% and 93.6% respectively. The accuracy values are estimated over random samples of 500 relations.

Comparison Harvesting non-taxonomic relations from UGCs is non-trivial with no standard evaluation frameworks available. Furthermore, the significant difference between English and Chinese makes it difficult to compare our method with similar research. Pasca (2017) focuses on modifier in categories and is not directly comparable to our work. In YAGO (Suchanek et al., 2007), relations in categories are extracted by handcrafting regular expressions. They extract nine non-taxonomic relations, with accuracy values of around 90%-98%. Our approach avoids the manual work to a large extent and harvests more types of relations with a comparable accuracy.

Next we compare our work with Nastase and Strube (2008), which heavily relies on prepositions in patterns such as “Verb in/of” and “Member/CEO/President of” to discover relations. In Chinese, prepositions are usually expressed implicitly and hence these patterns are not directly applicable. We implement a variant for Chinese (denoted as CN-WikiRe). The patterns that we used in CN-WikiRe are shown in Table 4. In the experiments, we extract 165,048 non-taxonomic relation instances using CN-WikiRe, containing 631 relation types. Although the number of relation types may seem large at the first glance, only 14% of them are actual relation predicates, with the rest being either incorrect or uninformative. The reasons are twofold: i) word segmentation and POS tagging for Chinese short texts still suffer from low accuracy and ii) not all verbs extracted by CN-WikiRe can serve as relation predicates (e.g., “传导(transmit)”, “缩小(shrink)”). We sample 500 relations from the collection where the extracted verbs are labeled as real relation predicates. The accuracy is 58.6%, much lower than our method. Furthermore, the partially explicit and implicit patterns (see (Nastase and Strube, 2008)) do not have their counterparts in Chinese. Therefore, our method is superior to existing systems.

7 Conclusion and Future Work

We propose a weakly supervised framework to extract relations from Chinese UGCs. For is-a relations, we introduce a word embedding based method and refine prediction results using collective inference. To extract non-taxonomic relations, we design a graph mining technique to harvest relation types and category patterns with minimal human supervision. Future work includes: i) improving our work for short text knowledge extraction and ii) designing a general framework for cross-lingual UGC relation extraction.

Acknowledgements

This work is supported by the National Key Research and Development Program of China under Grant No. 2016YFB1000904. Chengyu Wang would also like to thank the ECNU Outstanding Doctoral Dissertation Cultivation Plan of Action (No. YB2016040) for the support of his research.

References

Daniele Alfarone and Jesse Davis. 2015. Unsupervised learning of an IS-A taxonomy from a limited domain-specific corpus. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*. pages 1434–1441.

Bahram Alidaee, Fred Glover, Gary A. Kochenberger, and Haibo Wang. 2007. Solving the maximum edge weight clique problem via unconstrained quadratic programming. *European Journal of Operational Research* 181(2):592–597.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of 13th Conference of the European Chapter of the Association for Computational Linguistics*. pages 23–32.

Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third International Conference on Web Search and Web Data Mining*. pages 101–110.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. 2006. *Transductive Inference and Semi-Supervised Learning*. MIT Press.

Yanping Chen, Qinghua Zheng, and Wei Zhang. 2014. Omni-word feature and soft constraint for chinese relation extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. pages 572–581.

Oren Etzioni, Anthony Fader, Janara Christensen, Stephen Soderland, and Mausam. 2011. Open information extraction: The second generation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. pages 3–10.

Zhijia Fang, Haofen Wang, Jorge Gracia, Julia Bosque-Gil, and Tong Ruan. 2016. Zhishi.lemon: On publishing zhishi.me as linguistic linked open data. In *Proceedings of the 15th International Semantic Web Conference*. pages 47–55.

Tiziano Flati, Daniele Vannella, Tommaso Pasini, and Roberto Navigli. 2014. Two is bigger (and better) than one: the wikipedia bitaxonomy project. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. pages 945–955.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. pages 1199–1209.

Ruiji Fu, Bing Qin, and Ting Liu. 2013. Exploiting multiple sources for open-domain hypernym discovery. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. pages 1224–1234.

Gene H. Golub, Per Christian Hansen, and Dianne P. O'Leary. 1999. Tikhonov regularization and total least squares. *SIAM J. Matrix Analysis Applications* 21(1):185–194.

Amit Gupta, Francesco Piccinno, Mikhail Kozhevnikov, Marius Pasca, and Daniele Pighin. 2016. Revisiting taxonomy induction over wikipedia. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 2300–2309.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 970–976.

Jinyang Li, Chengyu Wang, Xiaofeng He, Rong Zhang, and Ming Gao. 2015. User generated content oriented chinese taxonomy construction. In *Proceedings of the 17th Asia-Pacific Web Conference*. pages 623–634.

Weiming Lu, Renjie Lou, Hao Dai, Zhenyu Zhang, Shansong Yang, and Baogang Wei. 2015. Taxonomy induction from chinese encyclopedias by combinatorial optimization. In *Proceedings of the 4th CCF Conference on Natural Language Processing and Chinese Computing*. pages 299–312.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. 2015. YAGO3: A knowledge base from multilingual wikipedias. In *Proceedings of the Seventh Biennial Conference on Innovative Data Systems Research*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*. pages 3111–3119.

Paramita Mirza and Sara Tonelli. 2016. On the contribution of word embeddings to temporal relation classification. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 2818–2828.

Vivi Nastase and Michael Strube. 2008. Decoding wikipedia categories for knowledge acquisition. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*. pages 1219–1224.

Vivi Nastase, Michael Strube, Benjamin Boerschinger, Cäcilia Zirn, and Anas Elghafari. 2010. Wikinet: A very large scale multi-lingual concept network. In *Proceedings of the International Conference on Language Resources and Evaluation*.

Xing Niu, Xinruo Sun, Haofen Wang, Shu Rong, Guilin Qi, and Yong Yu. 2011. Zhishi.me - weaving chinese linking open data. In *Proceedings of the*

10th International Semantic Web Conference. pages 205–220.

Marius Pasca. 2017. German typographers vs. german grammar: Decomposition of wikipedia category labels into attribute-value pairs. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. pages 315–324.

Simone Paolo Ponzetto and Roberto Navigli. 2009. Large-scale taxonomy mapping for restructuring and integrating wikipedia. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*. pages 2083–2088.

Simone Paolo Ponzetto and Michael Strube. 2007. Deriving a large-scale taxonomy from wikipedia. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*. pages 1440–1445.

Likun Qiu and Yue Zhang. 2014. ZORE: A syntax-based system for chinese open relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. pages 1870–1880.

Xipeng Qiu, Qi Zhang, and Xuanjing Huang. 2013. Fudannlp: A toolkit for chinese natural language processing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. pages 49–54.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the 25th International Conference on Computational Linguistics*. pages 1025–1036.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. page 2389–2398.

Michael M. Sørensen. 2004. New facets and a branch-and-cut algorithm for the weighted clique problem. *European Journal of Operational Research* 154(1):57–70.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*. pages 697–706.

Chengyu Wang and Xiaofeng He. 2016. Chinese hypernym-hyponym extraction from user generated categories. In *Proceedings of the 26th International Conference on Computational Linguistics*. pages 1350–1361.

Chengyu Wang, Junchi Yan, Aoying Zhou, and Xiaofeng He. 2017. Transductive non-linear learning for chinese hypernym prediction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Chengyu Wang, Rong Zhang, Xiaofeng He, and Aoying Zhou. 2016. Error link detection and correction in wikipedia. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. pages 307–316.

Zhigang Wang, Juanzi Li, Shuangjie Li, Mingyang Li, Jie Tang, Kuo Zhang, and Kun Zhang. 2014. Cross-lingual knowledge validation based taxonomy derivation from heterogeneous online wikis. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. pages 180–186.

Zhigang Wang, Juanzi Li, Zhichun Wang, Shuangjie Li, Mingyang Li, Dongsheng Zhang, Yao Shi, Yongbin Liu, Peng Zhang, and Jie Tang. 2013. Xlore: A large-scale english-chinese bilingual knowledge graph. In *Proceedings of the ISWC 2013 Posters & Demonstrations Track*. pages 121–124.

Tianxing Wu, Guilin Qi, Haofen Wang, Kang Xu, and Xuan Cui. 2016. Cross-lingual taxonomy alignment with bilingual biterm topic model. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. pages 287–293.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Qili Zhu. 2012. Probase: a probabilistic taxonomy for text understanding. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. pages 481–492.

Bo Xu, Chenhao Xie, Yi Zhang, Yanghua Xiao, Haixun Wang, and Wei Wang. 2016a. Learning defining features for categories. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. pages 3924–3930.

Bo Xu, Yong Xu, Jiaqing Liang, Chenhao Xie, Bin Liang, Wanyun Cui, and Yanghua Xiao. 2017. Cn-dbpedia: A never-ending chinese knowledge extraction system. In *Advances in Artificial Intelligence: From Theory to Practice - 30th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems*. pages 428–438.

Bo Xu, Yi Zhang, Jiaqing Liang, Yanghua Xiao, Seung-won Hwang, and Wei Wang. 2016b. Cross-lingual type inference. In *Proceedings of 21st International Conference on Database Systems for Advanced Applications*. pages 447–462.

Improving Slot Filling Performance with Attentive Neural Networks on Dependency Structures

Lifu Huang[1*], **Avirup Sil**[2], **Heng Ji**[1], **Radu Florian**[2]
[1] Rensselaer Polytechnic Institute, [2] IBM T.J. Watson Research Center
[1]{huangl7,jih}@rpi.edu, [2]{avi,raduf}@us.ibm.com

Abstract

Slot Filling (SF) aims to extract the values of certain types of attributes (or slots, such as person:cities_of_residence) for a given entity from a large collection of source documents. In this paper we propose an effective DNN architecture for SF with the following new strategies: (1). Take a regularized dependency graph instead of a raw sentence as input to DNN, to compress the wide contexts between query and candidate filler; (2). Incorporate two attention mechanisms: local attention learned from query and candidate filler, and global attention learned from external knowledge bases, to guide the model to better select indicative contexts to determine slot type. Experiments show that this framework outperforms state-of-the-art on both relation extraction (16% absolute F-score gain) and slot filling validation for each individual system (up to 8.5% absolute F-score gain).

1 Introduction

The goal of Slot Filling (SF) is to extract pre-defined types of attributes or slots (e.g., *per:cities_of_residence*) for a given query entity from a large collection of documents. The slot filler (attribute value) can be an entity, time expression or value (e.g., *per:charges*). The TAC-KBP slot filling task (Ji et al., 2011a; Surdeanu and Ji, 2014) defined 41 slot types, including 25 types for person and 16 types for organization.

One critical component of slot filling is relation extraction, namely to classify the relation between a pair of query entity and candidate slot filler into one of the 41 types or none. Most previous studies have treated SF in the same way as within-sentence relation extraction tasks in ACE [1] or SemEval (Hendrickx et al., 2009). They created training data based on crowd-sourcing or distant supervision, and then trained a multi-class classifier or multiple binary classifiers for each slot type based on a set of hand-crafted features.

Although Deep Neural Networks (DNN) such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have achieved state-of-the-art results on within-sentence relation extraction (Zeng et al., 2014; Liu et al., 2015; Santos et al., 2015; Nguyen and Grishman, 2015; Yang et al., 2016; Wang et al., 2016), there are limited studies on SF using DNN. Adel and Schütze (2015) and Adel et al. (2016) exploited DNN for SF but did not achieve comparable results as traditional methods. In this paper we aim to answer the following questions: What is the difference between SF and ACE/SemEval relation extraction task? How can we make DNN work for SF?

We argue that SF is different and more challenging than traditional relation extraction. First, a query and its candidate filler are usually separated by much wider contexts than the entity pairs in traditional relation extraction. As Figure 1 shows, in ACE data, for 70% of relations, two mentions are embedded in each other or separated by at most one word. In contrast, in SF, more than 46% of ⟨*query, filler*⟩ entity pairs are separated by at least 7 words. For example, in the following sentence:

E1. "**Arcandor**$_{query}$ owns a 52-percent stake in Europe's second biggest tourism group Thomas Cook, the Karstadt chain of department stores and iconic shops such as the **KaDeWe**$_{filler}$ in what used to be the commercial heart of West Berlin.",

[*] This work was carried out during an internship at IBM Research.

[1] http://www.itl.nist.gov/iad/mig/tests/ace/

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2588–2597
Copenhagen, Denmark, September 7–11, 2017.

Here, **Arcandor** and **KaDeWe** are far separated and it's difficult to determine the slot type as *org:subsidiaries* based on the raw wide contexts.

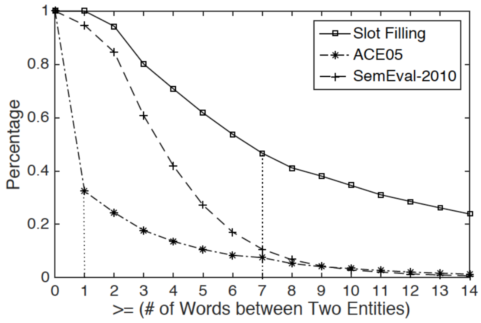


Figure 1: Comparison of the Percentage by the # of Words between two entity mentions in ACE05 and SemEval-2010 Task 8 relations, and between query and slot filler in KBP2013 Slot Filling.

In addition, compared with relations defined in ACE (18 types) and SemEval (9 types), slot types are more fine-grained and heavily rely on indicative contextual words for disambiguation. Yu et al. (2015) and Yu and Ji (2016) demonstrate that many slot types can be specified by contextual *trigger* words. Here, a *trigger* is defined as the word which is related to both the query and candidate filler, and can indicate the type of the target slot. Considering E1 again, *owns* is a trigger word between **Arcandor** and **KaDeWe**, which can indicate the slot type as *org:subsidiaries*. Most previous work manually constructed trigger lists for each slot type. However, for some slot types, the triggers can be implicit and ambiguous.

To address the above challenges, we propose the following new solutions:

- To compress wide contexts, we model the connection of query and candidate filler using dependency structures, and feed dependency graph to DNN. To our knowledge, we are the first to directly take dependency graphs as input to CNN.

- Motivated by the definition of *trigger*, we design two attention mechanisms: a local attention and a global attention using large external knowledge bases (KBs), to better capture implicit clues that indicate slot types.

2 Architecture Overview

Figure 2 illustrates the pipeline of a SF system. Given a query and a source corpus, the system retrieves related documents, identifies candidate fillers (including entities, time, values, and titles), extracts the relation between query and each candidate filler occurring in the same sentence, and finally determines the filler for each slot. Relation extraction plays a vital role in such a SF pipeline. In this work, we focus on relation extraction component and design a neural architecture.

Given a query, a candidate filler, and a sentence, we first construct a regularized dependency graph and take all $\langle$*governor, dependent*$\rangle$ word pairs as input to Convolutional Neural Networks (CNN).

Moreover, We design two attention mechanisms: **(1) Local Attention**, which utilizes the concatenation of *Query* and *Candidate Filler* vectors to measure the relatedness of each input bigram (we set filter width as 2) to the specific query and filler. **(2) Global attention**: We use prelearned slot type representations to measure the relatedness of each input bigram with each slot type via a transformation matrix. These two attention mechanisms will guide the pooling step to select the information which is related to query and filler and can indicate slot type.

3 Regularized Dependency Graph based CNN

3.1 Regularized Dependency Graph

Dependency parsing based features, especially the shortest dependency path between two entities, have been proved to be effective to extract the most important information for identifying the relation between two entities (Bunescu and Mooney, 2005; Zhao and Grishman, 2005; GuoDong et al., 2005; Jiang and Zhai, 2007). Several recent studies also explored transforming a dependency path into a sequence and applied Neural Networks to the sequence for relation classification (Liu et al., 2015; Cai et al., 2016; Xu et al., 2015).

However, for SF, the shortest dependency path between query and candidate filler is not always sufficient to infer the slot type due to two reasons. First, the most indicative words may not be included in the path. For example, in the following sentence:

E2. Survivors include two *sons* and *daughters*-in-law, **Troy**$_{filler}$ and Phyllis Perry, **Kenny**$_{query}$ and Donna Perry, all of Bluff City.

the shortest dependency path between **Kenny** and **Troy** is: "Troy $\xleftarrow{conj}$ Perry $\xleftarrow{conj}$ Kenny", which

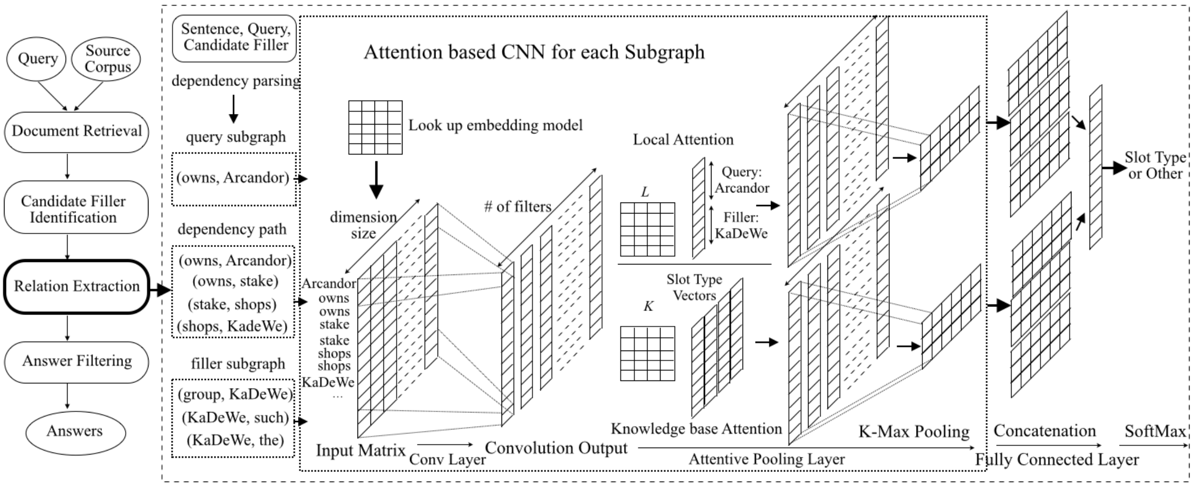


Figure 2: Overview of the Architecture.

does not include the most indicative words: *sons* and *daughters* for their *per:siblings* relation. In addition, the relation between query and candidate filler is also highly related to their entity types, especially for disambiguating slot types such as *per:country_of_birth* *per:state_of_birth* and *per:city_of_birth*. Entity types can be inferred by enriching query and filler related contexts. For example, in the following sentence:

E3. **Merkel**$_{query}$ died in the southern German city of **Passau**$_{filler}$ in 1967.

we can determine the slot type as *city* related by incorporating rich contexts (e.g., "*city*").

To tackle these problems, we propose to regularize the dependency graph, incorporating the shortest dependency path between query and candidate filler, as well as their rich contextual words.

Given a sentence s including a query q and candidate filler f, we first apply the Stanford Dependency Parser (Manning et al., 2014) to generate all dependent word pairs: ⟨*governor, dependent*⟩, then discover the shortest dependency path between query and candidate filler based on Breadth-First-Search (BFS) algorithm. The regularized dependency graph includes words on the shortest dependency path, as well as words which can be connected to query and filler within n hops. In our experiments, we set $n = 1$. Figure 3 shows the dependency parsing output for E1 mentioned in Section 1, and the regularized dependency graph with the bold circled nodes. We can see that, the most indicative trigger *owns* can be found in both the shortest dependency path of **Arcandor** and **KaDeWe**, and the context words of **Arcandor**. In addition, the context words, such as *shops*, can also infer the type of candidate filler **KaDeWe** as an Organization.

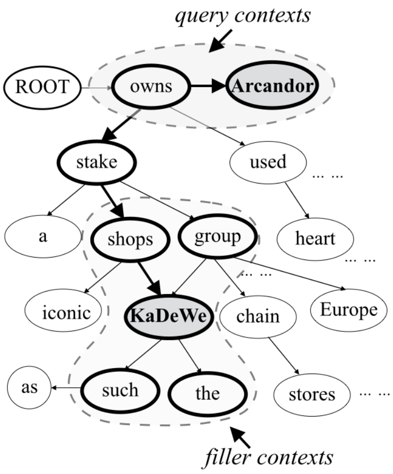


Figure 3: Regularized Dependency Graph for Query *Arcandor* and Filler *KaDeWe* in E1.

3.2 Graph based CNN

Previous work (Adel et al., 2016) split an input sentence into three parts based on the positions of the query and candidate filler and generate a feature vector for each part using a shared CNN. To compress the wide contexts, instead of taking the raw sentence directly as input, we split the regularized dependency graph into three parts: query related subgraph, candidate filler related subgraph, and the dependency path between query and filler. Each subgraph will be taken as input to a CNN, as illustrated in Figure 2. We now describe the details of each part as follows.

Input layer: Each subgraph or path G in the regularized dependency graph is represented as a set of dependent word pairs $G = \{\langle g_1, d_1\rangle, \langle g_2, d_2\rangle, ...\langle g_n, d_n\rangle\}$. Here, g_i, d_i denote the *governor* and *dependent* respectively. Each word is represented as a d-dimensional pre-trained vector. For the word which does not exist in the pre-trained embedding model, we assign a random vector for it. Each word pair $\langle g_i, d_i\rangle$ is converted to a $\mathbb{R}^{2\times d}$ matrix. We concatenate the matrices of all word pairs and get the input matrix $M \in \mathbb{R}^{2n\times d}$.

Convolution layer: For each subgraph, $M \in \mathbb{R}^{2n\times d}$ is the input of the convolution layer, which is a list of linear layers with parameters shared by filtering windows with various size. We set the stride as 2 to obtain all word pairs from the input matrix M. For each word pair $p_{in} = \langle v_{g_i}, v_{d_i}\rangle$, we compute the output vector p_{out} of a convolution layer as:

$$p_{out} = tanh(W \cdot p_{in} + b)$$

where p_{in} is the concatenation of vectors for the words v_{g_i} and v_{d_i}, W denotes the convolution weights, and b is the bias. In our work all three convolution layers share the same W and b.

K-Max Pooling Layer: we follow Adel et al. (2016) and use K-max pooling to select K values for each convolution layer. Later we will incorporate attention mechanisms into K-max pooling.

Fully Connected Layer: After getting the high-level features based on the (attentive) pooling layer for each input subgraph, we flatten and concatenate these three outputs as input to a fully connected layer. This layer connects each input to every single neuron it contains, and learns non-linear combinations based on the whole input.

Output Layer: It takes the output of the fully connected layer as input to a softmax regression function to predict the type. We use negative log-likelihood as loss function to train the parameters.

4 Attention Strategies for SF

4.1 Local Attention

The basic idea of attention mechanism is to assign a weight to each position of a lower layer when computing the representations for an upper layer, so that the model can be attentive to specific regions (Bahdanau et al., 2014). In SF, the indicative words are the most meaningful information that the model should pay attention to. Wang et al. (2016) applied attention from the entities directly to determine the most influential parts in the input sentence. Following the same intuition, we apply the attention from the query and candidate filler to the convolution output instead of the input, to avoid information vanishing during convolution process (Yin et al., 2016).

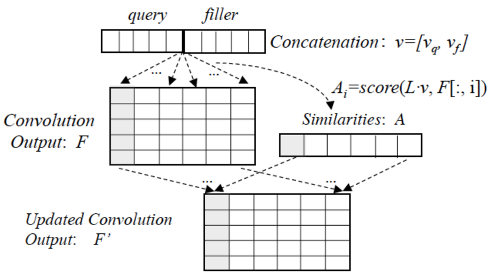

query
filler
Concatenation: $v=[v_q, v_f]$
A_i=score($L\cdot v$, F[:, i])
Convolution Output: F
Similarities: A
Updated Convolution Output: F'

Figure 4: Local Attention.

Figure 4 illustrates our approach to incorporate local attention. We first concatenate the vector of query q and candidate filler f using pre-trained embeddings $v = [v_q, v_f], \in \mathbb{R}^{2d}$. For q or f that includes multiple words, we average the vectors of all individual words. For each convolution output F, which is a feature map $\in \mathbb{R}^{\widehat{K}\times N}$, where N is the number of word pairs from the input, and $\widehat{K}$ is the number of filters, we define the attention similarity matrix $A \in R^{N\times 1}$ as:

$$A_i = cosine(L \cdot v, F[:, i])$$

where $L \in \mathbb{R}^{\widehat{K}\times 2d}$ is the transformation matrix between the concatenated vector v and convolution output. $F[:, i]$ denotes the vector of column i in F. Then we use the attention matrix A to update each column of the feature map F, and generate an updated attention feature map $F^{'}$ as follows:

$$F^{'}[:, i] = F[:, i] \cdot A[i]$$

4.2 Global Attention

Considering E1 in Section 1 again, the most discriminating word *owns* is not only related to the query and filler, but more specific to the type *org:subsidiaries*. Local attention aims to identify the query and filler related contexts. In order to detect type-indicative parts, we design global attention, using pre-learned slot type representations.

Wang et al. (2016) explored relation type attention with automatically learned type vectors from training data. However, in most cases, the training data is not balanced and some relation types cannot be assigned high-quality vectors with limited data. Thus, we designed two methods to generate pre-learned slot type representations.

First, we compose pre-trained lexical word embeddings of each slot type name to directly generate type representations. For example, for the type *per:date_of_birth*, we average the vectors of all single tokens (*person*, *birth*, *date*) within the type name as its representation.

Another new method is to take advantage of the large size of facts from external knowledge base (KB) to represent slot types. We use DBPedia as the target KB and manually map KB relations to slot types. For example, *per:alternate_names* can be mapped to *alternativeNames*, *birthName* and *nickName* in DBPedia. Thus for each slot type, we collect many triples: ⟨*query, slot, filler*⟩ and use TransE (Bordes et al., 2013), which models slot types as translations operating on the embeddings of query and filler, to derive a representation for each slot type. Compared with the first lexical based slot type representation induction approach, TransE jointly learns entity and relation representations and can better capture the correlation and differentiation among various slot types. Later, we will show the impact of these two types of slot type representations in Section 5.2.

Next we use the pre-learned slot type representations to guide the pooling process. Formally, let $R \in \mathbb{R}^{d \times r}$ be the matrix of all slot type vectors, where d is the vector dimension size and r is the number of slot types. Let $F \in \mathbb{R}^{\widehat{K} \times N}$ be a convolution output, which is the same as Section 4.1. We define the attention weight matrix S as:

$$S_{i,j} = cosine(F[:,i], W \cdot R[:,j])$$

where $W \in \mathbb{R}^{\widehat{K} \times d}$ is the transformation matrix for pre-learned slot type representations and convolution output. Given the weight matrix S, we generate the attention feature map $F^{''}$ as follows:

$$F^{''}[:,i] = F[:,i] \cdot \max_{j}\{S[i,j]\}$$

where $S[i,:]$ denotes the similarity scores between column i in F with all slot type vectors, and $\max\{S[i,:]\}$ is the max value among all similarity scores for column i in F.

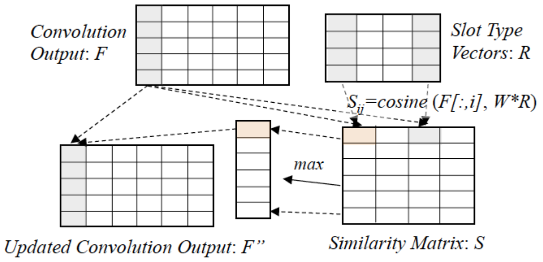


Figure 5: Global Attention.

We apply local attention to each convolution output of each subgraph, then take the concatenation of three flattened attentive pooling outputs to a fully connected layer and generate a robust feature representation. Similarly, another feature representation is generated based on global attention. We concatenate these two features to the softmax layer to get the predicted types.

5 Experiments

5.1 Data

For model training, Angeli et al. (2014) created some high-quality clean annotations for SF based on crowd-sourcing[2]. In addition, Adel et al. (2016) automatically created a larger size of noisy training data based on distant supervision, including about 1,725,891 positive training instances for 41 slot types. We manually assessed the correctness of candidate filler identification and their slot type annotation, and extracted a subset of their noisy annotations and combined it with the clean annotations. Ultimately, we obtain 23,993 positive and 3,000 negative training instances for all slot types.

We evaluate our approach in two settings: (1) relation extraction for all slot types, given the boundaries of query and candidate fillers. We use a script[3] to generate a test set (4892 instances) from KBP 2012/2013 slot filling evaluation data sets with manual assessment. (2) apply our approach to re-classify and validate the results of slot filling systems. We use the data from the KBP 2013 Slot Filling Validation (SFV) shared task, which consists of merged responses returned by 52 runs from 18 teams submitted to the Slot Filling task.

We used the May-2014 English Wikipedia dump to learn word embeddings based on the Continuous Skip-gram model (Mikolov et al., 2013).

[2] http://nlp.stanford.edu/software/mimlre-2014-07-17-data.tar.gz

[3] http://cistern.cis.lmu.de.

Table 1 shows the hyper-parameters that we use to train embeddings and our model.

Parameter	Parameter Name	Value
d	Word Embedding Size	50
λ	Initial Learning Rate	0.1
$\widehat{K}$	# of Filters in Convolution Layer	500
h	Hidden Unit Size in Fully Connected Layer	1000
k^p	Max Pooling Size	3

Table 1: Hyper-parameters.

5.2 Relation Extraction

We compare with several existing state-of-the-art slot filling and relation extraction methods on slot filling data sets. Besides, we also design several variants to demonstrate the effectiveness of each component in our approach. Table 2 presents the detailed approaches and the features used by these methods.

We report scores with Macro F_1 and Micro F_1. Macro F_1 is computed from the average precision and recall of all types while Micro F_1 is computed from the overall precision and recall, which is more useful when the size of each category varies. Table 3 shows the comparison results on relation extraction.

We can see that by incorporating the shortest dependency path or regularized dependency graph into neural networks, the model can achieve more than 13% micro F-score gain over the previously widely adopted methods by state-of-the-art systems for SemEval relation classification. It confirms our claim that SF is a different and more challenging task than traditional relation classification and also demonstrates the effectiveness of dependency knowledge for SF.

In addition, by incorporating local or global attention mechanism into the GraphCNN, the performance can be further improved, which proves the effectiveness of these two attention mechanisms. Our method finally achieves absolute 16% F-score gain by incorporating the regularized dependency graph and two attention mechanisms.

To better quantify the contribution of different attention mechanisms on each slot type, we further compared the performances on each single slot type. Table 4 shows the gain/loss percentage of the Micro F1 by adding local attention or global attention into GraphCNN for each slot type. We can see that both attentions yield improvement for most slot types.

5.3 Slot Filling Validation

In TAC-KBP 2013 Slot Filling Validation (SFV) (Ji et al., 2011b) task, there are 100 queries. We first retrieve the sentences from the source corpus (about 2,099,319 documents) and identify the query and candidate filler using the offsets generated by each response, then apply our approach to re-predict the slot type. Figure 6 shows the F-scores based on our approach and the original system. For a system which has multiple runs, we select one for comparison. We can see that our approach consistently improves the performance of almost all SF systems in an absolute gain range of [-0.18%, 8.48%]. With analysis of each system run, we find that our approach can provide more gains to the SF systems which have lower precision.

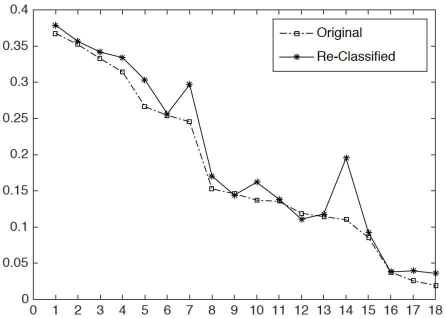


Figure 6: Comparison on Individual System

Previous studies (Tamang and Ji, 2011; Rodriguez et al., 2015; Zhi et al., 2015; Viswanathan et al., 2015; Rajani and Mooney, 2016a; Yu et al., 2014a; Rajani and Mooney, 2016b) for SFV trained supervised classifiers based on features such as confidence score of each response and system credibility. For comparison, we developed a new SFV approach: a new SVM classifier based on a set of features (docId, filler string, original predicted slot type and confidence score, new predicted slot type confidence score based on our neural architecture) for each response to take advantage of the redundant information from various system runs. Table 5 compares our SFV performance against previous reported scores on judging each response as true or false. We can see that our approach advances state-of-the-art methods.

5.4 Detailed Analysis

Significance Test: Table 3 shows the results of multiple variants of our approach. To demonstrate the difference between the results of these

	Method	Description	Features
Previous Methods	FCM (Yu et al., 2014b)	A factor-based compositional embedding model by deriving sentence-level and substructure representations	word embedding, dependency parse, WordNet, name tagging
	CR-CNN (Santos et al., 2015)	Applying a pairwise ranking loss function over CNNs	word embedding, word position embedding
	Context-CNN (Adel et al., 2016)	Splitting each sentence into three parts based on query and filler positions, and apply a CNNs to each part	word embedding
Our Methods	DepCNN	Applying CNNs to the shortest dependency path between query and filler	word embedding, dependency parse
	GraphCNN	DepCNN + applying CNNs to both query and filler related contextual graphs	word embedding, dependency parse
	GraphCNN+L	incorporating query and filler information as local attention into the GraphCNN	word embedding, dependency parse
	GraphCNN+G^1	incorporating slot type representations learned from type names as global attention into the GraphCNN	word embedding, dependency parse
	GraphCNN+G^2	incorporating slot type representations learned from external KB as global attention into the GraphCNN	word embedding, dependency parse, knowledge base
	GraphCNN+L+G^2	incorporating both local and KB based global attentions into the GraphCNN	word embedding, dependency parse, knowledge base

Table 2: Approach Descriptions for Multi-Class Relation Classification

	Method	Micro F1	Macro F1
Previous Methods	FCM	41.13	12.68
	CR-CNN	41.61	-
	ContextCNN	41.31	29.01
Variants of Our Methods	DepCNN	54.91	36.63
	GraphCNN	55.63	36.74
	GraphCNN+L	56.29	37.12
	GraphCNN+G^1	56.18	36.87
	GraphCNN+G^2	56.81	38.15
Our Method	GraphCNN+L+G^2	**57.39**	**38.26**

Table 3: Relation Extraction Component Performance on **Slot Filling Data Set** (%).

approaches are not random, we randomly sample 10 subsets (each contains 500 instances) from the testing dataset, and conduct paired t-test between each of these two approaches over these 10 data sets to check whether the average difference in their performances is significantly different or not. Table 6 shows the two-tailed P values. The differences are all considered to be statistically significant while all p-values are less than 0.05.

Impact of Training Data Size: We examine the impact of the size of training data on the performance for each slot type. Table 4 shows the distribution of training data and the F-score of each single type. We can see that, for some slot types, such as *per:date_of_birth* and *per:age*, the entity types of their candidate fillers are easy to learn and differentiate from other slot types, and their indicative words are usually explicit, thus our approach can get high f-score with limited training data (less than 507 instances). In contrast, for some slots, such as *org:location_of_headquarters*, their clues are implicit and the entity types of candidate fillers are difficult to be inferred. Although the size of training data is larger (more than 1,433 instances), the f-score remains quite low. One possible solution is to incorporate fine-grained entity types from existing tools into the neural architecture.

Impact of Wide Context Distribution: We further compared the performance and distribution of instances with wide contexts across all slot types. A context is considered as wide if the query and candidate filler are separated with more than 7 words. The last column of Table 4 shows the performance by incorporating regularized dependency graph (ContextCNN *v.s.* GraphCNN). We can see that, for most slot types with wide contexts, such as *per:states_of_residence* and *per:employee_of*, the f-scores are improved significantly while for some slots such as *per:date_of_birth*, the f-scores decrease because most date phrases do not exist in our pre-trained embedding model.

Error Analysis: Both of the relation extraction and SFV results showed that, more than 58% classification errors are spurious. Besides, we also observed many misclassifications that are caused by conflicting clues. There may be several indicative words within the contexts, but only one slot type is labeled, especially between *per:location_of_death* and *per:location_of_residence*. For example, in the following sentence:

E4. ***Billy Mays***$_{query}$*, a beloved and parodied pitchman who became a pop-culture figure through his commercials for cleaning prod-*

Slot Type	Impact of Attention (%)		Training Data Distribution (%)	F1 (%)	Wide Context Distribution (%)	Impact of Dependency Graph (%)
	Local	Global-KB				
state_of_death	9.8	-0.4	0.9	41.8	66.7	44.2
date_of_birth	7.3	121.3	**1.3**	**84.1**	**20.0**	**-81.9**
age	4.1	-5.3	**1.3**	**98.5**	15.9	28.5
per:alternate_names	-2.0	21.2	1.5	36.6	41.5	62.0
origin	-0.9	7.8	1.7	61.5	29.3	137.3
country_of_birth	16.7	12.0	1.9	61.5	55.6	162.5
city_of_death	1.1	3.3	1.9	61.3	70.3	24.4
state_of_headq.	9.7	-5.1	**3.1**	**51.7**	54.8	95.7
cities_of_residence	4.5	5.7	3.5	57.3	77.0	40.5
states_of_residence	-4.3	2.3	3.8	50.5	**45.9**	**175.8**
country_of_headq.	5.6	-0.8	**5.3**	**41.5**	54.4	146.3
city_of_headq.	1.6	-6.9	**6.7**	**30.3**	54.9	39.3
employee__of	14.9	4.9	7.3	65.9	**54.9**	**132.5**
countries_of_residence	37.7	8.6	7.4	47.4	47.2	134.9

Table 4: Comparison Analysis for Each Slot Type.

Methods	Precision (%)	Recall (%)	F-score (%)
Random	28.64	50.48	36.54
Voting	42.16	70.18	52.68
Linguistic Indicators	50.24	70.69	58.73
SVM	56.59	48.72	52.36
MTM (Yu et al., 2014a)	53.94	72.11	61.72
Our Approach	**70.46**	64.07	**67.11**

Table 5: Overall Performance for SFV: all the Baseline Systems are from Yu et al. (2014a).

Method 1	Method 2	P Value
DepCNN	GraphCNN	0.0165
GraphCNN	GraphCNN+L	0.0007
GraphCNN	GraphCNN+G^1	0.0160
GraphCNN	GraphCNN+G^2	<0.0001
GraphCNN+L	GraphCNN+L+G^2	0.0009
GraphCNN+G^2	GraphCNN+L+G^2	0.0010

Table 6: Statistical Significance Test.

> *ucts like Orange Glo, OxiClean and Kaboom, died Sunday at his home in* ***Tampa***$_{filler}$*, Fla.,*

the correct slot type is *per:city_of_death* while our approach mistakenly labeled it as *per:city_of_residence* with clue words like *home*. In addition, as we mentioned before, slot typing heavily relies on the fine-grained entity type of candidate filler, especially for the *location* (including *city, state, country*) related slot types. When the context is not specified enough, we can only rely on the pre-trained embeddings of candidate fillers, which may not be as informative as we hope. Such cases will benefit from introducing additional gazetteers such as *Geonames* [4].

6 Related Work

One major challenge of SF is the lack of labeled data to generalize a wide range of features and patterns, especially for slot types that are in the long-tail of the quite skewed distribution of slot fills (Ji et al., 2011a). Previous work has mostly focused on compensating the data needs by constructing patterns (Sun et al., 2011; Roth et al., 2014b), automatic annotation by distant supervision (Surdeanu et al., 2011; Roth et al., 2014a; Adel et al., 2016), and constructing trigger lists for unsupervised dependency graph mining (Yu and Ji, 2016; Yu et al., 2016). Some work (Rodriguez et al., 2015; Zhi et al., 2015; Viswanathan et al., 2015; Hong et al., 2015; Rajani and Mooney, 2016a; Yu et al., 2014a; Rajani and Mooney, 2016b; Ma et al., 2015) also attempted to validate slot types by combining results from multiple systems.

Our work is also related to dependency path based relation extraction. The effectiveness of dependency features for relation classification has been reported in some previous work (Bunescu and Mooney, 2005; Zhao and Grishman, 2005; GuoDong et al., 2005; Jiang and Zhai, 2007; Neville and Jensen, 2003; Ebrahimi and Dou, 2015; Xu et al., 2015; Ji et al., 2014). Liu et al. (2015), Cai et al. (2016) and Xu et al. (2015) applied CNN, bidirectional recurrent CNN and LSTM to CONLL relation extraction and demonstrated that the most important information has been included within the shortest paths between entities. Considering that the indicative words may not be included by the shortest dependency path between query and candidate filler, we enrich it to a regularized dependency graph by adding more contexts.

[4] http://www.geonames.org/

7 Conclusions and Future Work

In this work, we discussed the unique challenges of slot filling compared with tradition relation extraction tasks. We designed a regularized dependency graph based neural architecture for slot filling. By incorporating local and global attention mechanisms, this approach can better capture indicative contexts. Experiments on relation extraction and Slot Filling Validation data sets demonstrate the effectiveness of our neural architecture. In the future, we will combine additional rules, patterns, and constraints with DNN techniques to further improve slot filling.

Acknowledgments

This work was supported by the DARPA DEFT No. FA8750-13-2-0041, U.S. ARL NS-CTA No. W911NF-09-2-0053, and NSF IIS 1523198. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Heike Adel, Benjamin Roth, and Hinrich Schütze. 2016. Comparing convolutional neural networks to traditional models for slot filling. In *Proc. NAACL2016*.

Heike Adel and Hinrich Schütze. 2015. Cis at tac cold start 2015: Neural networks and coreference resolution for slot filling. In *Proc. TAC2015*.

Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D Manning. 2014. Combining distant and partial supervision for relation extraction. In *Proc. EMNLP2014*.

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

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proc. NIPS2013*.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proc. EMNLP2005*.

Rui Cai, Xiaodong Zhang, and Houfeng Wang. 2016. Bidirectional recurrent convolutional neural network for relation classification. In *Proc. ACL2016*.

Javid Ebrahimi and Dejing Dou. 2015. Chain based rnn for relation classification. In *Proc. NAACL2015*.

Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proc. ACL2005*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*.

Yu Hong, Di Lu, Dian Yu, Xiaoman Pan, Xiaobin Wang, Yadong Chen, Lifu Huang, and Heng Ji. 2015. Rpi blender tac-kbp2015 system description. In *Proc. TAC2015*.

Heng Ji, Taylor Cassidy, Qi Li, and Suzanne Tamang. 2014. Tackling representation, annotation and classification challenges for temporal knowledge base population. *Knowledge and Information Systems*.

Heng Ji, Ralph Grishman, and Hoa Trang Dang. 2011a. An overview of the tac2011 knowledge base population track. In *Proc. Text Analysis Conference (TAC2011)*.

Heng Ji, Ralph Grishman, and Hoa Trang Dang. 2011b. Overview of the tac2011 knowledge base population track.

Jing Jiang and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *HLT-NAACL*, pages 113–120.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, and Ming Zhou. 2015. A dependency-based neural network for relation classification. In *Proc. ACL2015*.

Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei Han. 2015. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *Proc. SIGKDD*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proc. ACL2014*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. NIPS2013*.

Jennifer Neville and David Jensen. 2003. Collective classification with relational dependency networks. In *Proceedings of the Second International Workshop on Multi-Relational Data Mining*.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of NAACL Workshop on Vector Space Modeling for NLP*.

Nazneen Fatema Rajani and Raymond J Mooney. 2016a. Combining supervised and unsupervised ensembles for knowledge base population. In *Proc. EMNLP2016*.

Nazneen Fatema Rajani and Raymond J Mooney. 2016b. Supervised and unsupervised ensembling for knowledge base population. *arXiv preprint arXiv:1604.04802*.

Miguel Rodriguez, Sean Goldberg, and Daisy Zhe Wang. 2015. University of florida dsr lab system for kbp slot filler validation 2015. In *Proc. TAC2015*.

Benjamin Roth, Tassilo Barth, Michael Wiegand, Mittul Singh, and Dietrich Klakow. 2014a. Effective slot filling based on shallow distant supervision methods. In *Proc. TAC KBP 2014*.

Benjamin Roth, Emma Strubell, John Sullivan, Lakshmi Vikraman, Kate Silverstein, and Andrew McCallum. 2014b. Universal schema for slot-filling, cold-start kbp and event argument extraction: Umassiesl at tac kbp 2014. In *TAC KBP*.

Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proc. ACL2015*.

Ang Sun, Ralph Grishman, Wei Xu, and Bonan Min. 2011. Nyu 2011 system for kbp slot filling. In *Proc. TAC2011*.

Mihai Surdeanu, Sonal Gupta, John Bauer, David McClosky, Angel X Chang, Valentin I Spitkovsky, and Christopher D Manning. 2011. Stanfords distantly supervised slot-filling system. In *Proc. TAC2011*.

Mihai Surdeanu and Heng Ji. 2014. Overview of the english slot filling track at the tac2014 knowledge base population evaluation. In *Proc. TAC2014*.

Suzanne Tamang and Heng Ji. 2011. Adding smarter systems instead of human annotators: re-ranking for system combination. In *Proceedings of the 1st international workshop on Search and mining entity-relationship data*, pages 3–8. ACM.

Vidhoon Viswanathan, Nazneen Fatema Rajani, Yinon Bentor, and Raymond Mooney. 2015. Stacked ensembles of information extractors for knowledge-base population. In *Proc. ACL2015*.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proc. ACL2016*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proc. EMNLP2015*.

Yunlun Yang, Yunhai Tong, Shulei Ma, and Zhi-Hong Deng. 2016. A position encoding convolutional neural network based on dependency tree for relation classification. In *Proc. EMNLP2016*.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. In *Proc. TACL2016*.

Dian Yu, Hongzhao Huang, Taylor Cassidy, Heng Ji, Chi Wang, Shi Zhi, Jiawei Han, and Malik Voss, Clare R anfd Magdon-Ismail. 2014a. The wisdom of minority: Unsupervised slot filling validation based on multi-dimensional truth-finding. In *Proc. COLING2014*.

Dian Yu and Heng Ji. 2016. Unsupervised person slot filling based on graph mining. In *Proc. ACL2016*.

Dian Yu, Heng Ji, Sujian Li, and Chin-Yew Lin. 2015. Why read if you can scan? trigger scoping strategy for biographical fact extraction. In *Proc. NAACL-HLT 2015*.

Dian Yu, Xiaoman Pan, Boliang Zhang, Lifu Huang, Di Lu, Spencer Whitehead, and Heng Ji. 2016. Rpi blender tac-kbp2016 system description.

Mo Yu, Matthew Gormley, and Mark Dredze. 2014b. Factor-based compositional embedding models. In *NIPS Workshop on Learning Semantics*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proc. COLING2014*.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proc. ACL2005*.

Shi Zhi, Bo Zhao, Wenzhu Tong, Jing Gao, Dian Yu, Heng Ji, and Jiawei Han. 2015. Modeling truth existence in truth discovery. In *Proc. SIGKDD2015*.

Identifying Products in Online Cybercrime Marketplaces: A Dataset for Fine-grained Domain Adaptation

Greg Durrett
UT Austin
gdurrett@cs.utexas.edu

Jonathan K. Kummerfeld
University of Michigan
jkummerf@umich.edu

Taylor Berg-Kirkpatrick
Carnegie Mellon University
tberg@cs.cmu.edu

Rebecca S. Portnoff
UC Berkeley
rsportnoff@cs.berkeley.edu

Sadia Afroz
ICSI, UC Berkeley
sadia@icsi.berkeley.edu

Damon McCoy
NYU
mccoy@nyu.edu

Kirill Levchenko
UC San Diego
klevchen@cs.ucsd.edu

Vern Paxson
ICSI, UC Berkeley
vern@berkeley.edu

Abstract

One weakness of machine-learned NLP models is that they typically perform poorly on out-of-domain data. In this work, we study the task of identifying products being bought and sold in online cybercrime forums, which exhibits particularly challenging cross-domain effects. We formulate a task that represents a hybrid of slot-filling information extraction and named entity recognition and annotate data from four different forums. Each of these forums constitutes its own "fine-grained domain" in that the forums cover different market sectors with different properties, even though all forums are in the broad domain of cybercrime. We characterize these domain differences in the context of a learning-based system: supervised models see decreased accuracy when applied to new forums, and standard techniques for semi-supervised learning and domain adaptation have limited effectiveness on this data, which suggests the need to improve these techniques. We release a dataset of 1,938 annotated posts from across the four forums.[1]

1 Introduction

NLP can be extremely useful for enabling scientific inquiry, helping us to quickly and efficiently understand large corpora, gather evidence, and test hypotheses (Bamman et al., 2013; O'Connor et al., 2013). One domain for which automated analysis is particularly useful is Internet security: researchers obtain large amounts of text data pertinent to active threats or ongoing cybercriminal activity, for which the ability to rapidly characterize that text and draw conclusions can reap major benefits (Krebs, 2013a,b). However, conducting automatic analysis is difficult because this data is out-of-domain for conventional NLP models, which harms the performance of both discrete models (McClosky et al., 2010) and deep models (Zhang et al., 2017). Not only that, we show that data from one cybercrime forum is even out of domain with respect to *another* cybercrime forum, making this data especially challenging.

TITLE: [buy] Backconnect bot
BODY: Looking for a solid backconnect bot .
If you know of anyone who codes them please let me know

(a) File 0-initiator4856

TITLE: Exploit cleaning ?
BODY: Have some Exploits i need fud .

(b) File 0-initiator10815

Figure 1: Example posts and annotations from Darkode, with annotated product tokens underlined. The second example exhibits jargon (*fud* means "fully undetectable"), nouns that could be a product in other contexts (*Exploit*), and multiple lexically-distinct descriptions of a single service. Note that these posts are much shorter than the average Darkode post (61.5 words).

In this work, we present the task of identifying products being bought and sold in the marketplace sections of these online cybercrime forums.

[1] Dataset and code to train models available at https://evidencebasedsecurity.org/forums/

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2598–2607
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Forum	Posts	Words per post	Products per post	Annotated posts	Annotators per post	Inter-annotator agreement 3-annotated	 all-annotated
Darkode	3,368	61.5	3.2	660/100/100	3/8/8	0.62	0.66
Hack Forums	51,271	58.9	2.2	758/140	3/4	0.58	0.65
Blackhat	167	174	3.2	80	3	0.66	0.67
Nulled	39,118	157	2.3	100	3	0.77	-

Table 1: Forum statistics. The left columns (posts and words per post) are calculated over all data, while the right columns are based on annotated data only. Note that products per post indicate product mentions per post, not product types. Slashes indicate the train/development/test split for Darkode and train/test split for Hack Forums. Agreement is measured using Fleiss' Kappa; the two columns cover data where three annotators labeled each post and a subset labeled by all annotators.

We define a token-level annotation task where, for each post, we annotate references to the product or products being bought or sold in that post. Having the ability to automatically tag posts in this way lets us characterize the composition of a forum in terms of what products it deals with, identify trends over time, associate users with particular activity profiles, and connect to price information to better understand the marketplace. Some of these analyses only require post-level information (what is the product being bought or sold in this post?) whereas other analyses might require token-level references; we annotate at the token level to make our annotation as general as possible. Our dataset has already proven enabling for case studies on these particular forums (Portnoff et al., 2017), including a study of marketplace activity on bulk hacked accounts versus users selling their own accounts.

Our task has similarities to both slot-filling information extraction (with provenance information) as well as standard named-entity recognition (NER). Compared to NER, our task features a higher dependence on context: we only care about the specific product being bought or sold in a post, not other products that might be mentioned. Moreover, because we are operating over forums, the data is substantially messier than classical NER corpora like CoNLL (Tjong Kim Sang and De Meulder, 2003). While prior work has dealt with these messy characteristics for syntax (Kaljahi et al., 2015) and for discourse (Lui and Baldwin, 2010; Kim et al., 2010; Wang et al., 2011), our work is the first to tackle forum data (and marketplace forums specifically) from an information extraction perspective.

Having annotated a dataset, we examine supervised and semi-supervised learning approaches to the product extraction problem. Binary or CRF classification of tokens as products is effective, but performance drops off precipitously when a system trained on one forum is applied to a different forum: in this sense, even two different cybercrime forums seem to represent different "fine-grained domains." Since we want to avoid having to annotate data for every new forum that might need to be analyzed, we explore several methods for adaptation, mixing type-level annotation (Garrette and Baldridge, 2013; Garrette et al., 2013), token-level annotation (Daume III, 2007), and semi-supervised approaches (Turian et al., 2010; Kshirsagar et al., 2015). We find little improvement from these methods and discuss why they fail to have a larger impact.

Overall, our results characterize the challenges of our fine-grained domain adaptation problem in online marketplace data. We believe that this new dataset provides a useful testbed for additional inquiry and investigation into modeling of fine-grained domain differences.

2 Dataset and Annotation

We consider several forums that vary in the nature of products being traded:

- Darkode: Cybercriminal wares, including exploit kits, spam services, ransomware programs, and stealthy botnets.
- Hack Forums: A mixture of cyber-security and computer gaming blackhat and non-cybercrime products.
- Blackhat: Blackhat Search Engine Optimization techniques.
- Nulled: Data stealing tools and services.

Table 1 gives some statistics of these forums. These are the same forums used to study product activity in Portnoff et al. (2017). We collected all available posts and annotated a subset of them. In total, we annotated 130,336 tokens; accounting for multiple annotators, our annotators considered 478,176 tokens in the process of labeling the data.

Figure 1 shows two examples of posts from Darkode. In addition to aspects of the annotation, which we describe below, we see that the text exhibits common features of web text: abbreviations, ungrammaticality, spelling errors, and visual formatting, particularly in thread titles. Also, note how some words that are not products here might be in other contexts (e.g., *Exploits*).

2.1 Annotation Process

We developed our annotation guidelines through six preliminary rounds of annotation, covering 560 posts. Each round was followed by discussion and resolution of every post with disagreements. We benefited from members of our team who brought extensive domain expertise to the task. As well as refining the annotation guidelines, the development process trained annotators who were not security experts. The data annotated during this process is not included in Table 1.

Once we had defined the annotation standard, we annotated datasets from Darkode, Hack Forums, Blackhat, and Nulled as described in Table 1.[2] Three people annotated every post in the Darkode training, Hack Forums training, Blackhat test, and Nulled test sets; these annotations were then merged into a final annotation by majority vote. The development and test sets for Darkode and Hack Forums were annotated by additional team members (five for Darkode, one for Hack Forums), and then every disagreement was discussed and resolved to produce a final annotation. The authors, who are researchers in either NLP or computer security, did all of the annotation.

We preprocessed the data using the tokenizer and sentence-splitter from the Stanford CoreNLP toolkit (Manning et al., 2014). Note that many sentences in the data are already delimited by line breaks, making the sentence-splitting task much easier. We performed annotation on the tokenized data so that annotations would be consistent with surrounding punctuation and hyphenated words.

Our full annotation guide is available with our data release.[3] Our basic annotation principle is to annotate tokens when they are either the product that will be delivered or are an integral part of the method leading to the delivery of that product. Figure 1 shows examples of this for a deliverable product (*bot*) as well as a service (*cleaning*). Both a product and service may be annotated in a single example: for a post asking to *hack an account*, *hack* is the method and the deliverable is the *account*, so both are annotated. In general, methods expressed as verbs may be annotated in addition to nominal references.

When the product is a multiword expression (e.g., *Backconnect bot*), it is almost exclusively a noun phrase, in which case we annotate the head word of the noun phrase (*bot*). Annotating single tokens instead of spans meant that we avoided having to agree on an exact parse of each post, since even the boundaries of base noun phrases can be quite difficult to agree on in ungrammatical text.

If multiple different products are being bought or sold, we annotate them all. We do not annotate:

- Features of products
- Generic product references, e.g., *this*, *them*
- Product mentions inside "vouches" (reviews from other users)
- Product mentions outside of the first and last 10 lines of each post[4]

Table 1 shows inter-annotator agreement according to our annotation scheme. We use the Fleiss' Kappa measurement (Fleiss, 1971), treating our task as a token-level annotation where every token is annotated as either a product or not. We chose this measure as we are interested in agreement between more than two annotators (ruling out Cohen's kappa), have a binary assignment (ruling out correlation coefficients) and have datasets large enough that the biases Krippendorff's Alpha addresses are not a concern. The values indicate reasonable agreement.

2.2 Discussion

Because we annotate entities in a context-sensitive way (i.e., only annotating those in product context), our task resembles a post-level information

[2] The table does not include additional posts that were labeled by all annotators in order to check agreement.

[3] `https://evidencebasedsecurity.org/forums/annotation-guide.pdf`

[4] In preliminary annotation we found that content in the middle of the post typically described features or gave instructions without explicitly mentioning the product. Most posts are unaffected by this rule: 96% of Darkode, 77% of Hack Forums, 84% of Blackhat, and 93% of Nulled posts are less than 20 lines. However, the cutoff still substantially reduced annotator effort on the tail of very long posts.

extraction task. The product information in a post can be thought of as a list-valued slot to be filled in the style of TAC KBP (Surdeanu, 2013; Surdeanu and Ji, 2014), with the token-level annotations constituting provenance information. However, we chose to anchor the task fully at the token level to simplify the annotation task: at the post level, we would have to decide whether two distinct product mentions were actually distinct products or not, which requires heavier domain knowledge. Our approach also resembles the fully token-level annotations of entity and event information in the ACE dataset (NIST, 2005).

3 Evaluation Metrics

In light of the various views on this task and its different requirements for different potential applications, we describe and motivate a few distinct evaluation metrics below. The choice of metric will impact system design, as we discuss in the following sections.

Token-level accuracy We can follow the approach used in token-level tasks like NER and compute precision, recall, and F_1 over the set of tokens labeled as products. This most closely mimics our annotation process.

Type-level product extraction (per post) For many applications, the primary goal of the extraction task is more in line with KBP-style slot filling, where we care about the set of products extracted from a particular post. Without a domain-specific lexicon containing full synsets of products (e.g., something that could recognize that *hack* and *access* are synonymous), it is difficult to evaluate this in a fully satisfying way. However, we approximate this evaluation by comparing the set of product *types*[5] in a post with the set of product types predicted by the system. Again, we consider precision, recall, and F_1 over these two sets. This metric favors systems that consistently make correct post-level predictions even if they do not retrieve every token-level occurrence of the product.

Post-level accuracy Most posts contain only one product, but our type-level extraction will naturally be a conservative estimate of performance simply because there may seem to be multiple "products" that are actually just different ways of referring to one core product. Roughly 60% of posts in the two forums contain multiple annotated tokens that are distinct beyond stemming and lowercasing. However, we analyzed 100 of these multiple product posts across Darkode and Hack Forums, and found that only 6 of them were actually selling multiple products, indicating that posts selling multiple types of products are actually quite rare (roughly 3% of cases overall). In the rest of the cases, the variations were due to slightly different ways of describing the same product.

In light of this, we also might consider asking the system to extract *some* product reference from the post, rather than all of them. Specifically, we compute accuracy on a post-level by checking whether the first product type extracted by the system is contained in the annotated set of product types.[6] Because most posts feature one product, this metric is sufficient to evaluate whether we understood what the core product of the post was.

3.1 Phrase-level Evaluation

Another axis of variation in metrics comes from whether we consider token-level or phrase-level outputs. As noted in the previous section, we did not annotate noun phrases, but we may actually be interested in identifying them. In Figure 1, for example, extracting *Backconnect bot* is more useful than extracting *bot* in isolation, since *bot* is a less specific characterization of the product.

We can convert our token-level annotations to phrase-level annotations by projecting our annotations to the noun phrase level based on the output of an automatic parser. We used the parser of Chen and Manning (2014) to parse all sentences of each post. For each annotated token that was given a nominal tag (N*), we projected that token to the largest NP containing it of length less than or equal to 7; most product NPs are shorter than this, and when the parser predicts a longer NP, our analysis found that it typically reflects a mistake. In Figure 1, the entire noun phrase *Backconnect bot* would be labeled as a product. For products realized as verbs (e.g., *hack*), we leave the annotation as the single token.

Throughout the rest of this work, we will evaluate sometimes at the token-level and sometimes at

[5] Two product tokens are considered the same type if after lowercasing and stemming they have a sufficiently small edit distance: 0 if the tokens are length 4 or less, 1 if the lengths are between 5 and 7, and 2 for lengths of 8 or more

[6] For this metric we exclude posts containing no products. These are usually posts that have had their content deleted or are about forum administration.

the NP-level[7] (including for the product type evaluation and post-level accuracy); we will specify which evaluation is used where.

4 Models

We consider several baselines for product extraction, two supervised learning-based methods (here), and semi-supervised methods (Section 5).

Baselines One approach takes the most **frequent** noun or verb in a post and classifies all occurrences of that word type as products. A more sophisticated lexical baseline is based on a product **dictionary** extracted from our training data: we tag the most frequent noun or verb in a post that also appears in this dictionary. This method fails primarily in that it prefers to extract common words like *account* and *website* even when they do not occur as products. The most relevant off-the-shelf system is an **NER** tagging model; we retrain the Stanford NER system on our data (Finkel et al., 2005). Finally, we can tag the **first** noun phrase of the post as a product, which will often capture the product if it is mentioned in the title of the post.[8]

We also include human performance results. We averaged the results for annotators compared with the consensus annotations. For the phrase level evaluation, we apply the projection method described in Section 3.1.

Binary classifier/CRF One learning-based approach to this task is to employ a binary SVM classifier for each token in isolation. We also experimented with a token-level CRF with a binary tagset, and found identical performance, so we describe the binary classifier version.[9] Our features look at both the token under consideration as well as neighboring tokens, as described in the next paragraph. A vector of "base features" is extracted for each of these target tokens: these include 1) sentence position in the document and word position in the current sentence as bucketed indices; 2) word identity (for common words), POS tag, and dependency relation to parent for each word in a window of size 3 surrounding the current word; 3) character 3-grams of the current word. The same base feature set is used for every token.

Our token-classifying SVM extracts base features on the token under consideration as well as its syntactic parent. Before inclusion in the final classifier, these features are conjoined with an indicator of their source (i.e., the current token or the parent token). Our NP-classifying SVM extracts base features on first, last, head, and syntactic parent tokens of the noun phrase, again with each feature conjoined with its token source.

We weight false positives and false negatives differently to adjust the precision/recall curve (tuned on development data for each forum), and we also empirically found better performance by upweighting the contribution to the objective of singleton products (product types that occur only once in the training set).

Post-level classifier As discussed in Section 3, one metric we are interested in is whether we can find *any* occurrence of a product in a post. This task is easier than the general tagging problem: if we can effectively identify the product in, e.g., the title of a post, then we do not need to identify additional references to that product in the body of the post. Therefore, we also consider a post-level model, which directly tries to select one token (or NP) out of a post as the most likely product. Structuring the prediction problem in this way naturally lets the model be more conservative in its extractions, since highly ambiguous product mentions can be ignored if a clear product mention is present. Put another way, it supplies a useful form of prior knowledge, namely that each post has exactly one product in almost all cases.

Our post-level system is formulated as an instance of a latent SVM (Yu and Joachims, 2009). The output space is the set of all tokens (or noun phrases, in the NP case) in the post. The latent variable is the choice of token/NP to select, since there may be multiple correct choices of product tokens. The features used on each token/NP are the same as in the token classifier.

We trained all of the learned models by subgradient descent on the primal form of the objective (Ratliff et al., 2007; Kummerfeld et al., 2015). We use AdaGrad (Duchi et al., 2011) to speed convergence in the presence of a large weight vector with heterogeneous feature types. All product extractors in this section are trained for 5 iterations with ℓ_1-regularization tuned on the development set.

[7] Where NP-level means "noun phrases and verbs" as described in Section 3.1.

[8] Since this baseline fundamentally relies on noun phrases, we only evaluate it in the noun phrase setting.

[9] We further experimented with a bidirectional LSTM tagger and found similar performance as well.

Token Prediction							
	Tokens			Products			Posts
	P	R	F$_1$	P	R	F$_1$	Acc.
Freq	41.9	42.5	42.2	48.4	33.5	39.6	45.3
Dict	57.9	51.1	54.3	65.6	44.0	52.7	60.8
NER	59.7	62.2	60.9	60.8	62.6	61.7	72.2
Binary	62.4	76.0	**68.5**	58.1	77.6	66.4	75.2
Post	82.4	36.1	50.3	83.5	56.6	67.5	**82.4**
Human*	86.9	80.4	83.5	87.7	77.6	82.2	89.2

NP Prediction							
	NPs			Products			Posts
	P	R	F$_1$	P	R	F$_1$	Acc.
Freq	61.8	28.9	39.4	61.8	50.0	55.2	61.8
Dict	57.9	61.8	59.8	71.8	57.5	63.8	68.0
First	73.1	34.2	46.7	73.1	59.1	65.4	73.1
NER	63.6	63.3	63.4	69.7	70.3	70.0	76.3
Binary	67.0	74.8	**70.7**	65.5	82.5	73.0	82.4
Post	87.6	41.0	55.9	87.6	70.8	**78.3**	**87.6**
Human*	87.6	83.2	85.3	91.6	84.9	88.1	93.0

Table 2: Development set results on Darkode. Bolded F$_1$ values represent statistically-significant improvements over all other system values in the column with $p < 0.05$ according to a bootstrap resampling test. Our post-level system outperforms our binary classifier at whole-post accuracy and on type-level product extraction, even though it is less good on the token-level metric. All systems consistently identify product NPs better than they identify product tokens. However, there is a substantial gap between our systems and human performance.

4.1 Basic Results

Table 2 shows development set results on Darkode for each of the four systems for each metric described in Section 3. Our learning-based systems substantially outperform the baselines on the metrics they are optimized for. The post-level system underperforms the binary classifier on the token evaluation, but is superior at not only post-level accuracy but also product type F$_1$. This lends credence to our hypothesis that picking one product suffices to characterize a large fraction of posts. Comparing the automatic systems with human annotator performance we see a substantial gap. Note that our best annotator's token F$_1$ was 89.8, and NP post accuracy was 100%; a careful, well-trained annotator can achieve very high performance, indicating a high skyline.

The noun phrase metric appears to be generally more forgiving, since token distinctions within noun phrases are erased. The post-level NP system achieves an F-score of 78 on product type identification, and post-level accuracy is around 88%. While there is room for improvement, this system is accurate enough to enable analysis of Darkode with automatic annotation.

Throughout the rest of this work, we focus on NP-level evaluation and post-level NP accuracy.

5 Domain Adaptation

Table 2 only showed results for training and evaluating within the same forum (Darkode). However, we wish to apply our system to extract product occurrences from a wide variety of forums, so we are interested in how well the system will generalize to a new forum. Tables 3 and 4 show full results of several systems in within-forum and cross-forum evaluation settings. Performance is severely degraded in the cross-forum setting compared to the within-forum setting, e.g., on NP-level F$_1$, a Hack Forums-trained model is 14.6 F$_1$ worse at the Darkode task than a Darkode-trained model (61.2 vs. 75.8). Differences in how the systems adapt between different forums will be explored more thoroughly in Section 5.4.

In the next few sections, we explore several possible methods for improving results in the cross-forum settings and attempting to build a more domain-general system. These techniques generally reflect two possible hypotheses about the source of the cross-domain challenges:

Hypothesis 1: Product inventories are the primary difference across domains; context-based features will transfer, but the main challenge is not being able to recognize unknown products.

Hypothesis 2: Product inventories **and** stylistic conventions both differ across domains; we need to capture both to adapt models successfully.

5.1 Brown Clusters

To test Hypothesis 1, we investigate whether additional lexical information helps identify product-like words in new domains. A classic semi-supervised technique for exploiting unlabeled target data is to fire features over word clusters or word vectors (Turian et al., 2010). These features should generalize well across domains that the clusters are formed on: if product nouns occur in similar contexts across domains and therefore wind up in the same cluster, then a model trained on domain-limited data should be able to learn that that cluster identity is indicative of products.

We form Brown clusters on our unlabeled data from both Darkode and Hack Forums (see Table 1

System \ Eval data	Darkode			Hack Forums			Blackhat			Nulled			Avg
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	F_1
Trained on Darkode													
Dict	55.9	54.2	55.0	42.1	39.8	40.9	37.1	36.6	36.8	52.6	43.2	47.4	45.0
Binary	73.3	78.6	75.8	51.1	50.2	50.6	55.2	58.3	56.7	55.2	64.0	59.3	60.6
Binary + Brown Clusters	75.5	76.4	**76.0**	52.1	55.9	48.1	59.7	57.1	**58.4**	60.0	61.1	**60.5**	60.8
Binary + Gazetteers	73.1	75.6	74.3	52.6	51.1	**51.8**	–	–	–	–	–	–	–
Trained on Hack Forums													
Dict	57.3	44.8	50.3	50.0	52.7	51.3	45.0	44.7	44.8	51.1	43.6	47.1	48.4
Binary	67.0	56.4	61.2	58.0	64.2	61.0	62.4	60.8	**61.6**	71.0	68.9	69.9	63.4
Binary + Brown Clusters	67.2	52.5	58.9	59.3	64.7	**61.9**	61.9	59.6	60.7	73.1	67.4	**70.2**	62.9
Binary + Gazetteers	67.8	64.1	†**65.9**	59.9	61.3	60.6	–	–	–	–	–	–	–

Table 3: Test set results at the NP level in within-forum and cross-forum settings for a variety of different systems. Using either Brown clusters or gazetteers gives mixed results on cross-forum performance: only one of the improvements (†) is statistically significant with $p < 0.05$ according to a bootstrap resampling test. Gazetteers are unavailable for Blackhat and Nulled since we have no training data for those forums.

	Darkode	Hack Forums	Blackhat	Nulled
Trained on Darkode				
Dict	59.3	39.7	43.5	54.6
Post	**89.5**	66.9	**75.8**	79.0
+Brown	**89.5**	66.9	69.3	**84.8**
+Gaz	87.5	**72.1**	–	–
Trained on Hack Forums				
Dict	48.9	53.6	50.0	53.4
Post	78.1	78.6	74.1	81.3
+Brown	**82.2**	81.6	**77.4**	**82.5**
+Gaz	79.1	†**83.8**	–	–

Table 4: Test set results at the whole-post level in within-forum and cross-forum settings for a variety of different systems. Brown clusters and gazetteers give similarly mixed results as in the token-level evaluation; † indicates statistically significant gains over the post-level system with $p < 0.05$ according to a bootstrap resampling test.

for sizes). We use Liang (2005)'s implementation to learn 50 clusters.[10] Upon inspection, these clusters do indeed capture some of the semantics relevant to the problem: for example, the cluster 110 has as its most frequent members *service*, *account*, *price*, *time*, *crypter*, and *server*, many of which are product-associated nouns. We incorporate these as features into our model by characterizing each token with prefixes of the Brown cluster ID; we used prefixes of length 2, 4, and 6.

Tables 3 and 4 show the results of incorporating Brown cluster features into our trained models. These features do not lead to statistically-significant gains in either NP-level F_1 or post-level accuracy, despite small improvements in some cases. This indicates that Brown clusters might be a useful feature sometimes, but do not solve the domain adaptation problem in this context.[11]

5.2 Type-level Annotation

Another approach following Hypothesis 1 is to use small amounts of supervised data, One cheap approach for annotating data in a new domain is to exploit type-level annotation (Garrette and Baldridge, 2013; Garrette et al., 2013). Our token-level annotation standard is relatively complex to learn, but a researcher could quite easily provide a few exemplar products for a new forum based on just a few minutes of reading posts and analyzing the forum.

Given the data that we've already annotated, we can simulate this process by iterating through our labeled data and collecting annotated product names that are sufficiently common. Specifically, we take all (lowercased, stemmed) product tokens and keep those occurring at least 4 times in the training dataset (recall that these datasets are ≈ 700 posts). This gives us a list of 121 products in Darkode and 105 products in Hack Forums.

To incorporate this information into our system, we add a new feature on each token indicating whether or not it occurs in the gazetteer. At training time, we use the gazetteer scraped from the training set. At test time, we use the gazetteer from the target domain as a form of partial type-level supervision. Tables 3 and 4 shows the results of incorporating the gazetteer into the system. Gazetteers seem to provide somewhat consistent gains in cross-domain settings, though many of these individual improvements are not statistically significant, and the gazetteers can sometimes hurt performance when testing on the same domain the system was trained on.

[10] This value was chosen based on dev set experiments.

[11] We could also use vector representations of words here, but in initial experiments, these did not outperform Brown clusters. That is consistent with the results of Turian et al. (2010) who showed similar performance between Brown clusters and word vectors for chunking and NER.

Test	Darkode			Hack Forums			Blackhat			Nulled		
System	% OOV	R_{seen}	R_{OOV}	% OOV	R_{seen}	R_{OOV}	% OOV	R_{seen}	R_{OOV}	% OOV	R_{seen}	R_{OOV}
Binary (Darkode)	20	78	62	41	64	47	42	69	46	30	72	45
Binary (HF)	50	76	40	35	75	42	51	70	38	33	83	32

Table 5: Product token out-of-vocabulary rates on development sets (test set for Blackhat and Nulled) of various forums with respect to training on Darkode and Hack Forums. We also show the recall of an NP-level system on seen (R_{seen}) and OOV (R_{OOV}) tokens. Darkode seems to be more "general" than Hack Forums: the Darkode system generally has lower OOV rates and provides more consistent performance on OOV tokens than the Hack Forums system.

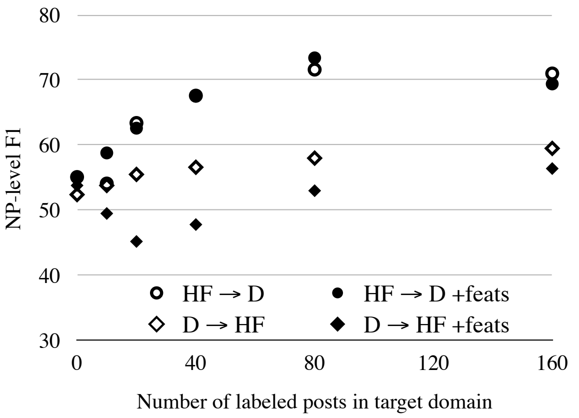


Figure 2: Token-supervised domain adaptation results for two settings. As our system is trained on an increasing amount of target-domain data (x-axis), its performance generally improves. However, adaptation from Hack Forums to Darkode is much more effective than the other way around, and using domain features as in Daume III (2007) gives little benefit over naïve use of the new data.

5.3 Token-level Annotation

We now turn our attention to methods that might address Hypothesis 2. If we assume the domain transfer problem is more complex, we really want to leverage labeled data in the target domain rather than attempting to transfer features based only on type-level information. Specifically, we are interested in cases where a relatively small number of labeled posts (less than 100) might provide substantial benefit to the adaptation; a researcher could plausibly do this annotation in a few hours.

We consider two ways of exploiting labeled target-domain data. The first is to simply take these posts as additional training data. The second is to also employ the "frustratingly easy" domain adaptation method of Daume III (2007). In this framework, each feature fired in our model is actually fired twice: one copy is domain-general and one is conjoined with the domain label (here, the name of the forum).[12] In doing so, the model should gain some ability to separate domain-general from domain-specific feature values, with regularization encouraging the domain-general feature to explain as much of the phenomenon as possible. For both training methods, we upweight the contribution of the target-domain posts in the objective by a factor of 5.

Figure 2 shows learning curves for both of these methods in two adaptation settings as we vary the amount of labeled target-domain data. The system trained on Hack Forums is able to make good use of labeled data from Darkode: having access to 20 labeled posts leads to gains of roughly 7 F_1. Interestingly, the system trained on Darkode is not able to make good use of labeled data from Hack Forums, and the domain-specific features actually cause a drop in performance until we include a substantial amount of data from Hack Forums (at least 80 posts). We are likely overfitting the small Hack Forums training set with the domain-specific features.

5.4 Analysis

In order to understand the variable performance and shortcomings of the domain adaptation approaches we explored, it is useful to examine our two initial hypotheses and characterize the datasets a bit further. To do so, we break down system performance on products seen in the training set versus novel products. Because our systems depend on lexical and character n-gram features, we expect that they will do better at predicting products we have seen before.

Table 5 confirms this intuition: it shows product out-of-vocabulary rates in each of the four forums relative to training on both Darkode and Hack Forums, along with recall of an NP-level system on both previously seen and OOV products. As expected, performance is substantially higher on in-

[12] If we are training on data from k domains, this gives rise to up to $k + 1$ total versions of each feature.

vocabulary products. OOV rates of a Darkode-trained system are generally lower on new forums, indicating that that forum has better all-around product coverage. A system trained on Darkode is therefore in some sense more domain-general than one trained on Hack Forums.

This would seem to support Hypothesis 1. Moreover, Table 3 shows that the Hack Forums-trained system achieves a 21% error reduction on Hack Forums compared to a Darkode-trained system, while a Darkode-trained system obtains a 38% error reduction on Darkode relative to a Hack Forums-trained system; this greater error reduction means that Darkode has better coverage of Hack Forums than vice versa. Darkode's better product coverage also helps explain why Section 5.3 showed better performance of adapting Hack Forums to Darkode than the other way around: augmenting Hack Forums data with a few posts from Darkode can give critical knowledge about new products, but this is less true if the forums are reversed. Duplicating features and adding parameters to the learner also has less of a clear benefit when adapting from Darkode, when the types of knowledge that need to be added are less concrete.

Note, however, that these results do not tell the full story. Table 5 reports recall values, but not all systems have the same precision/recall tradeoff: although they were tuned to balance precision and recall on their respective development sets, the Hack Forums-trained system is slightly more precision-oriented on Nulled than the Darkode-trained system.[13] In fact, Table 3 shows that the Hack Forums-trained system actually performs better on Nulled, largely due to better performance on previously-seen products. This indicates that there is some truth to Hypothesis 2: product coverage is not the only important factor determining performance.

6 Conclusion

We present a new dataset of posts from cybercrime marketplaces annotated with product references, a task which blends IE and NER. Learning-based methods degrade in performance when applied to new forums, and while we explore methods for fine-grained domain adaption in this data, effective methods for this task are still an open question.

Our datasets used in this work are available at https://evidencebasedsecurity.org/forums/ Code for the product extractor can be found at https://github.com/ccied/ugforum-analysis/tree/master/extract-product

Acknowledgments

This work was supported in part by the National Science Foundation under grants CNS-1237265 and CNS-1619620, by the Office of Naval Research under MURI grant N000140911081, by the Center for Long-Term Cybersecurity and by gifts from Google. We thank all the people that provided us with forum data for our analysis; in particular Scraping Hub and SRI for their assistance in collecting data for this study. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013. Learning Latent Personas of Film Characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Danqi Chen and Christopher D Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hal Daume III. 2007. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

J. L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

[13] While a hyperparameter controlling the precision/recall tradeoff could theoretically be tuned on the target domain, it is hard to do this in a robust, principled way without having access to a sizable annotated dataset from that domain. This limitation further complicates the evaluation and makes it difficult to set up apples-to-apples comparisons across domains.

Dan Garrette and Jason Baldridge. 2013. Learning a Part-of-Speech Tagger from Two Hours of Annotation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-World Semi-Supervised Learning of POS-Taggers for Low-Resource Languages. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Rasoul Kaljahi, Jennifer Foster, Johann Roturier, Corentin Ribeyre, Teresa Lynn, and Joseph Le Roux. 2015. Foreebank: Syntactic Analysis of Customer Support Forums. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Su Nam Kim, Li Wang, and Timothy Baldwin. 2010. Tagging and Linking Web Forum Posts. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL)*.

Brian Krebs. 2013a. Cards Stolen in Target Breach Flood Underground Markets.

Brian Krebs. 2013b. Who's Selling Credit Cards from Target?

Meghana Kshirsagar, Sam Thomson, Nathan Schneider, Jaime Carbonell, Noah A. Smith, and Chris Dyer. 2015. Frame-Semantic Role Labeling with Heterogeneous Annotations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick, and Dan Klein. 2015. An Empirical Analysis of Optimization for Max-Margin NLP. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Percy Liang. 2005. Semi-Supervised Learning for Natural Language Processing. In *Master's Thesis, Massachusetts Institute of Technology*.

Marco Lui and Timothy Baldwin. 2010. Classifying User Forum Participants: Separating the Gurus from the Hacks, and Other Tales of the Internet. In *Proceedings of the Australasian Language Technology Association Workshop (ALTA)*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations (ACL)*.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

NIST. 2005. The ACE 2005 Evaluation Plan. In *NIST*.

Brendan O'Connor, Brandon M. Stewart, and Noah A. Smith. 2013. Learning to Extract International Relations from Political Context. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Rebecca S. Portnoff, Sadia Afroz, Greg Durrett, Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick, Damon McCoy, Kirill Levchenko, and Vern Paxson. 2017. Tools for Automated Analysis of Cybercriminal Markets. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*.

Nathan J. Ratliff, Andrew Bagnell, and Martin Zinkevich. 2007. (Online) Subgradient Methods for Structured Prediction. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Mihai Surdeanu. 2013. Overview of the TAC2013 Knowledge Base Population Evaluation: English Slot Filling and Temporal Slot Filling,. In *Proceedings of the TAC-KBP 2013 Workshop*.

Mihai Surdeanu and Heng Ji. 2014. Overview of the English Slot Filling Track at the TAC2014 Knowledge Base Population Evaluation. In *Proceedings of the TAC-KBP 2014 Workshop*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word Representations: A Simple and General Method for Semi-Supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Li Wang, Marco Lui, Su Nam Kim, Joakim Nivre, and Timothy Baldwin. 2011. Predicting Thread Discourse Structure over Technical Web Forums. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning Structural SVMs with Latent Variables. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented Adversarial Networks for Domain Adaptation. In *Transactions of the Association for Computational Linguistics (TACL)*.

Labeling Gaps Between Words: Recognizing Overlapping Mentions with Mention Separators

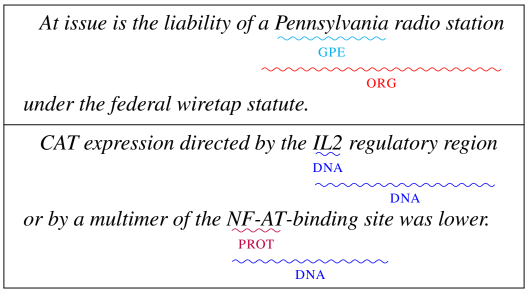
Aldrian Obaja Muis and **Wei Lu**
Singapore University of Technology and Design
{aldrian_muis,luwei}@sutd.edu.sg

Abstract

In this paper, we propose a new model that is capable of recognizing overlapping mentions. We introduce a novel notion of *mention separators* that can be effectively used to capture how mentions overlap with one another. On top of a novel multigraph representation that we introduce, we show that efficient and exact inference can still be performed. We present some theoretical analysis on the differences between our model and a recently proposed model for recognizing overlapping mentions, and discuss the possible implications of the differences. Through extensive empirical analysis on standard datasets, we demonstrate the effectiveness of our approach.

1 Introduction

Named entity recognition (NER), or in general the task of recognizing entity mentions[1] in a text, has been a research topic for many years (McCallum and Li, 2003; Nadeau and Sekine, 2007; Ratinov and Roth, 2009; Ling and Weld, 2012). However, as noted by Finkel and Manning (2009), many previous works ignored overlapping mentions, although they are quite common. Figure 1 illustrates some examples of overlapping mentions adapted from existing datasets. For example, the location mention *Pennsylvania* appears within the mention of type organization *a Pennsylvania radio station*. In practice, overlapping mentions have been found in many existing datasets across different domains (Doddington et al., 2004; Kim et al., 2003; Suominen et al., 2013). Developing algorithms that can effectively and efficiently extract overlapping mentions can be crucial for the performance of many downstream tasks such as relation extraction (Mintz et al., 2009; Gupta and Andrassy, 2016), event extraction (Lu and Roth, 2012; Li et al., 2013; Nguyen et al., 2016), coreference resolution (Chang et al., 2013; Lu et al., 2016), question answering (Mollá et al., 2007), and equation parsing (Roy et al., 2016).


At issue is the liability of a Pennsylvania radio station under the federal wiretap statute.

CAT expression directed by the IL2 regulatory region or by a multimer of the NF-AT-binding site was lower.

Figure 1: Examples of overlapping mentions.

Overlapping mention recognition is non-trivial, as existing methods that model mention recognition as a sequence prediction problem – e.g., using linear-chain conditional random fields (CRF) (Lafferty et al., 2001) – have difficulties in handling overlapping mentions (Alex et al., 2007). Finkel and Manning (2009) proposed to use a tree-based constituency parsing model to handle nested entities.[2] Due to the tree structured representation used, the resulting algorithm has a time complexity that is cubic in n for its inference procedure with n being the number of words in the sentence. This effectively makes the algorithm less scalable compared to models such as linear-chain CRF where the complexity is linear in n. Lu and Roth (2015) proposed an alternative approach which shows a time complexity that is linear in n. Their method differs from the conven-

[1] As noted in (Florian et al., 2004), mention recognition is more general than NER, where a mention can be either named, nominal, or pronominal.

[2] We note that nested entities are only one of the two kinds of overlapping entities, the other kind being crossing entities, where two entities overlap but neither is contained in another. However, it is extremely rare, and there is only one occurrence of crossing entity in our datasets.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2608–2618
Copenhagen, Denmark, September 7–11, 2017.

tional sequence labeling approach, in that a hypergraph representation was used in their model.

In this work, we make an observation that there exists an efficient model for recognizing overlapping mentions while still regarding the problem as a sequence labeling problem. As opposed to the conventional approach where we assign labels to natural language words, in our new approach we assign labels to the *gaps* between words, modeling the mention boundaries instead of modeling the role of words in forming mentions. Furthermore, while these gap-based labels can be modeled using conventional graphical models like linear-chain CRFs, we also propose a novel multigraph representation to utilize such gap-based labels efficiently. To the best of our knowledge, this is the first structured prediction model utilizing a gap-based annotation scheme to predict overlapping structures.

In this paper we make the following major contributions:

- We propose a set of *mention separators* which can be collectively used to define all possible mention combinations together with a novel multigraph representation, on top of which efficient and exact inference can be performed.
- Theoretically, we show that unlike a recently proposed state-of-the-art model that we compare against, our model does not exhibit the *spurious structures* issue in its learning procedure. On the other hand, it still maintains the same inference time complexity as the previous model.
- Empirically, we show that our model is able to achieve higher F_1-scores compared to previous models in multiple datasets.

We believe our proposed approach and the novel representations can be applied in other research problems involving predicting overlapping structures, and we hope this work can inspire further research along such a direction.

2 Related Work

NER or mention detection is normally regarded as a chunking task similar to base noun phrase chunking (Kudo and Matsumoto, 2001; Shen and Sarkar, 2005), and hence the entities or mentions are usually represented in a similar way, using BILOU (Beginning, Inside, Last, Outside, Unit-length mention) or the simpler BIO annotation scheme (Ratinov and Roth, 2009). As a chunking task, it is commonly modeled using sequence labeling models, such as the linear-chain CRF (Lafferty et al., 2001), which has time complexity $O(nT^2)$ with n being the number of words in the sentence and T the number of mention types.

On the task of recognizing mentions that may overlap with one another, one of the earliest works that attempted to regard this task as a structured prediction task was by McDonald et al. (2005). They represented entity mentions as top-k predictions with positive score from a structured multilabel classification model. Their model has a time complexity of $O(n^3T)$.

Alex et al. (2007) proposed a cascading approach using multiple linear-chain CRF models, each handling a subset of all the possible mention types, where the models which come later in the pipeline have access to the predictions of the models earlier in the pipeline. This results in the time complexity of roughly $O(nT)$ depending on how the pipeline was designed.

Finkel and Manning (2009) later proposed a constituency parser to handle nested entities by converting each sentence into a tree, and each mention is represented as one of the subtrees. Their model has the standard time complexity for a constituency parser with binary grammar: $O(n^3\,|G|)$, where $|G|$ is the size of the grammar, which in this case is proportional to T in the best case, and T^3 in the worst case. They showed that their model outperforms a semi-CRF baseline (Sarawagi and Cohen, 2004) in terms of F_1-score.

Recently, Lu and Roth (2015) proposed a hypergraph-based model called *mention hypergraph* that is able to handle overlapping mentions with a linear time complexity $O(nT)$. The model was shown to achieve competitive results compared to previous models on standard datasets. As we will be making extensive comparisons against this previous state-of-the-art model, we will describe this approach in the next section.

3 Mention Hypergraph

In the mention hypergraph model of Lu and Roth (2015), nodes and directed hyperedges[3] are used together to encode mentions and their combinations. The following five types of nodes are used at the position k of a sentence:

- $\mathbf{A}^k$ denotes all mentions starting at k or later,

[3] For brevity, in this paper we may also use *edge* to refer to *hyperedge* in some discussions.

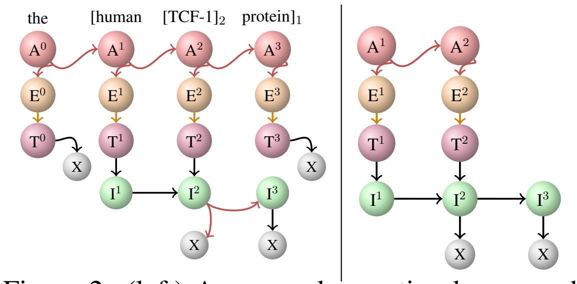


Figure 2: (left) An example mention hypergraph encoding two overlapping mentions. (right) An example of spurious structure.

- $\mathbf{E}^k$ denotes all mentions starting at k,
- $\mathbf{T}_t^k$ denotes all mentions (type t) starting at k,
- $\mathbf{I}_t^k$ denotes all mentions (type t) covering k,
- $\mathbf{X}$ denotes the end of a mention (leaf node).

Different hyperedges connecting these nodes are used to represent how the semantics of a node is composed from those of its child nodes.

Specifically, each $\mathbf{A}^k$ is connected to $\mathbf{A}^{k+1}$ and $\mathbf{E}^k$ through the hyperedge $\mathbf{A}^k \rightarrow (\mathbf{A}^{k+1}, \mathbf{E}^k)$, denoting the fact that the set of mentions that start at k or later is the union of the set of mentions that start at $k+1$ or later and the set of mentions that start at k. Each $\mathbf{E}^k$ is connected to $\mathbf{T}_1^k, \mathbf{T}_2^k, \ldots, \mathbf{T}_T^k$ through a hyperedge, denoting the fact that the mentions that start at k must be one of the T types. Each $\mathbf{T}_t^k$ can be connected to $\mathbf{I}_t^k$ through an edge (denoting there is a mention of type t that starts at the k-th token) or to $\mathbf{X}$ through another edge (denoting there are no mentions of type t that start at the k-th token). Each $\mathbf{I}_t^k$ can be connected to $\mathbf{I}_t^{k+1}$ (denoting there is a mention continuing to the next token), to $\mathbf{X}$ (denoting there is a mention ending here), or to both (with a single hyperedge, denoting the two cases above occur at the same time, a case of overlapping mentions).

In this mention hypergraph, each possible mention is represented as a path from a $\mathbf{T}$-node to the $\mathbf{X}$-node through a sequence of $\mathbf{I}$-nodes (each denoting the words which are part of the mention), and the set of all mentions present in a given sentence forms a hyperpath from the root node $\mathbf{A}_0$ to the leaf node $\mathbf{X}$. Figure 2 shows how the mention hypergraph represents the two mentions in the phrase "*the human TCF-1 protein*", which are "*TCF-1*" and "*human TCF-1 protein*". The edges $\mathbf{T}^1 - \mathbf{I}^1$ and $\mathbf{T}^2 - \mathbf{I}^2$ respectively denote that the words "*human*" and "*TCF-1*" are the beginning of a mention, and the edges from the $\mathbf{I}$-nodes to the $\mathbf{X}$-node define the end of the mentions. We remark that any mention hypergraph which encodes the mentions in a sentence, like this example, forms a hyperpath from the root node $\mathbf{A}^0$ to the leaf node $\mathbf{X}$, where a hyperpath is defined as a subgraph of a hypergraph with the property that each node has exactly one outgoing (hyper)edge except the last node, and the root node is connected to all nodes.

We refer the readers to Lu and Roth (2015) for more details on the model.

3.1 Spurious Structures

Mention hypergraph is trained by maximizing the likelihood of the training data, similar to training a linear-chain CRF. Recall that the likelihood of the training data can be calculated by taking the score of the correct structures and divide it by the normalization term, which is the total score of all possible structures. Lu and Roth (2015) used a dynamic programming algorithm to calculate the normalization term. However, the normalization term calculated this way contains additional terms, which we call the *spurious structures*. This leads to the following:

Theorem 3.1. *Let Z' be the normalization term as calculated using forward-backward algorithm on mention hypergraph, and let Z be the true normalization term. Then we have $Z' > Z$.*

Due to space limitation, we provide a proof sketch here. We refer the reader to the supplemental material for the details on spurious structures.

Proof sketch. First note that Z' includes all possible hyperpaths, so $Z' \geq Z$. Next, due to the presence of a node with multiple parents (e.g., node $\mathbf{I}^2$ in Figure 2 (left)), Z' includes the score of that node multiple times with different children, which results in a subgraph which is not a hyperpath. For example, Z' includes the score[4] of the structure shown in Figure 2 (right), where node $\mathbf{I}^2$ has two children, and so it is not a hyperpath. Since Z is the sum of all hyperpaths, this structure is not part of Z, but it is included in Z', so $Z' > Z$. □

Later we will see how this issue may affect the model's performance in predicting mentions.

4 Mention Separators

We now describe the mention separators which can be used to encode overlapping mentions in a sentence. Traditional encoding schemes that associate labels to words, such as BIO scheme, attach the semantics of the labels to the role of the words

[4] Note that structure scores $\exp(\mathbf{w} \cdot \mathbf{f})$ are always positive.

$w\ \ w$ X	$w\ [w$ S	$w]\ w$ E	$w]\ [w$ ES
w - w C	w -$[w$ CS	$w]$- w EC	$w]$-$[w$ ECS

Figure 3: An illustration of the 8 mention separators. The opening bracket ([), closing bracket (]), and dash (-) respectively refer to S, E, and C.

in forming mentions. For example, the label **B** in BIO scheme denotes the role of the word it is attached to, which is the first word of a mention.

This BIO scheme cannot be used directly to encode overlapping mentions, since they only encode whether a word is part of a mention and possibly their position in the mention. We notice that by encoding the mention boundaries instead, we can represent overlapping mentions. This can be accomplished by assigning what we call *mention separators* to the gaps between two words.

At each gap, we consider eight possible types of mention separators based on the combination of the following three cases:

1. A mention is starting at the next word (S)
2. A mention is ending at the previous word (E)
3. A mention is continuing to the next word (C)

Therefore, for each token, the possible combinations of cases are as follows: ECS, EC, CS, C, ES, E, S, and X, where X means none of the three cases applies. For example, the separator EC means there is a mention ending at the current token and another mention (overlapping) continuing to the next token. Note that there might be more than just two mentions involved here. Figure 3 shows an illustration of these separators, and Figure 4a shows how they can be used to encode the example in Figure 2.

Now we prove that the following theorem holds:

Theorem 4.1. *For any combination of mentions in a sentence, there is exactly one sequence of mention separators that encodes it.*

Proof. Consider the gap between any two adjacent words in the sentence. The combination of mentions present in the sentence uniquely defines what mention separator is associated with this gap. If there is a mention starting at the next word, then case S applies. Similarly, if there is a mention ending at the previous word, case E applies. And finally, if there is a mention covering both words, case C applies. By combining the cases, we get the corresponding mention separator for this gap. In this way, each gap in the sentence has a unique mention separator, which in turn defines the unique sequence of mention separators. □

Note that the converse of Theorem 4.1 is not true, as multiple mention combinations might encode to the same sequence of mention separators.

Now we describe two ways the mention separators can be used to encode overlapping mentions.

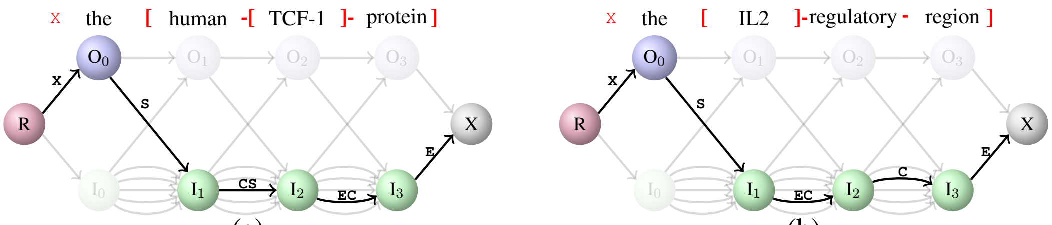
STATE-based The first is by directly using these mention separators to replace the standard mention encoding scheme (e.g., BIO encoding) in standard linear-chain CRF. So we assign each mention separator to a state in a linear-chain CRF model. Since this model encodes the gap between words and also the gap before the first word and after the last word, a sentence with n words is modeled by a sequence of $n + 1$ mention separators. Since each sequence of mention separators can only encode mentions of the same type, we support multiple types by using multiple sequences, one for each mention type.

EDGE-based Now, we propose a novel way of utilizing these mention separators. Since the mention separators encode the gaps between words, it is more intuitive to assign the mention separators to the edges of a graphical model, as opposed to the states, as described in the previous paragraph. To do this, we need to define the states of the models in such a way that all possible sequences of mention separators are accounted for. For this purpose we assign two states to each word at position k:

- $\mathbf{I}_k$: word at k is part of a mention,
- $\mathbf{O}_k$: word at k is not part of any mentions.

Next we define the edges between the states according to the eight possible mention separators between adjacent words. More specifically, each mention separator is mapped to an edge connecting one state in the current position to another state in the next position depending on whether the separator defines current and next word as part of an mention, so in total we have eight edges between two positions in the model. Some mention separators may connect the same two states, for example, the ES and C separator both connect $\mathbf{I}_k$ to $\mathbf{I}_{k+1}$ since in both cases the current word and the next word are part of a mention. In those cases, we simply define multiple edges between the pair of states. The resulting graph, where there can be multiple edges between two states, is known in graph theory literature as a **multigraph**[5].

[5] In this work, the multigraph representation can also be


x the [human -[TCF-1]- protein]
R
O0 O1 O2 O3
I0 I1 I2 I3
X
x
s
CS
EC
E
x the [IL2]-regulatory - region]
R
O0 O1 O2 O3
I0 I1 I2 I3
X
x
s
EC
C
E

(a) (b)

Figure 4: Our mention separator model with the EDGE representation encoding two phrases.

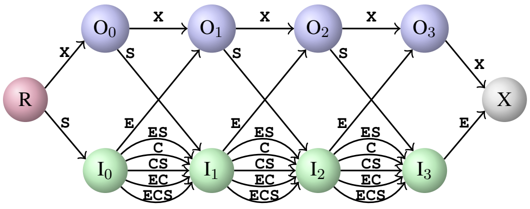


Figure 5: The full graph in EDGE-based model.

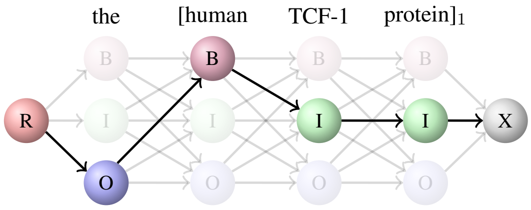


Figure 6: A linear-chain CRF model encoding a mention in BIO scheme.

The first **I**- and **O**-nodes in the sentence are connected to the root node, and the last **I**- and **O**-nodes are connected to the unique leaf node **X**.

Figure 4a shows how the EDGE-based model encodes the two mentions "*human TCF-1 protein*" and "*TCF-1*" in the phrase "*the human TCF-1 protein*", and Figure 4b shows the encoding of the phrase found in the second example in Figure 1. Note how each edge maps to a distinct mention separator visualized in the text in red.

Figure 5 shows the full graph of our EDGE-based model, in a format similar to the trellis graph for linear-chain CRFs in Figure 6. We remark that the EDGE-based model can be seen as an extension of linear-chain CRFs, with additional semantics attached to the edges. Also note that this graph encodes only one mention type. To support multiple types, similar to the STATE-based approach we can use multiple chains, one for each type.

Note that the edges in our EDGE-based representations are directed, with nodes on the left serving as parents to the nodes on the right. Such directed edges will be helpful when performing inference, to be discussed in the next section.

We remark that the way we utilize multigraph in the EDGE-based model can also be applied to the discontiguous mention model (DMM) by Muis and Lu (2016). In fact, it can be shown that the number of canonical structures as calculated in the supplementary material of DMM paper matches the number of possible paths in our multigraph-based model, as the transition matrix in DMM corresponds to the number of possible transitions from one position to the next position, which is encoded in our multigraph-based model as edges between adjacent positions. See the supplemental material for more discussion on this.

4.1 Training, Inference and Decoding

We follow the log-linear approach to define our model, using regularized log-likelihood in training data $\mathcal{D}$ as our objective function, as follows:

$$\mathcal{L}_{\mathcal{D}}(\mathbf{w}) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \left[\sum_{\mathbf{e}\in\mathbf{y}} \mathbf{w}\cdot\mathbf{f}(\mathbf{e}) - \log Z_{\mathbf{w}}(\mathbf{x}) \right] - \lambda||\mathbf{w}||^2 \tag{1}$$

Here, $(\mathbf{x}, \mathbf{y})$ is a training instance consisting of the sentence $\mathbf{x}$ and the correct output $\mathbf{y}$, $\mathbf{w}$ is the weight vector, $\mathbf{f}(\mathbf{e})$ is the feature vector defined over the edge $\mathbf{e}$, $Z_{\mathbf{w}}(\mathbf{x})$ is the normalization term, and λ is the l_2-regularization parameter. The objective function is then optimized until convergence using L-BFGS (Liu and Nocedal, 1989).

We note the mention hypergraph model also defines the objective in a similar manner. For both of our models, the inference is done based on a generalized inside-outside algorithm. Both models involve directed structures, on top of which the inference algorithm first calculates the inside score for each node from the leaf node to root, and then the outside score from the root to the leaf node, in very much the same way as how inference is done in a classic graphical model. Specifically, for our EDGE-based model, the inside scores are calculated using a bottom-up (right-to-left) dynamic programming procedure, where we calculate the inside score at each node by summing up the scores associated with each path connecting the current node to one of its child nodes. Each

regarded as a *lattice* where edges are associated with labels.

	ACE-2004			ACE-2005			GENIA		
	Train (%)	Dev (%)	Test (%)	Train (%)	Dev (%)	Test (%)	Train (%)	Dev (%)	Test (%)
# sentence	6,799	829	879	7,336	958	1,047	14,836	1,855	1,855
w/ o.l.	2,685 (39)	293 (35)	373 (42)	2,686 (37)	341 (36)	330 (32)	3,199 (22)	366 (20)	448 (24)
# mentions	22,207	2,511	3,031	24,687	3,217	3,027	46,473	5,014	5,600
o.l.	10,170 (46)	1,091 (43)	1,418 (47)	9,937 (40)	1,192 (37)	1,184 (39)	8,337 (18)	915 (18)	1,217 (22)
o.l. (s)	5,431 (24)	624 (25)	780 (26)	5,044 (20)	600 (19)	638 (21)	4,613 (10)	479 (10)	634 (11)

Table 1: Statistics of the datasets used in the experiments. *w/ o.l.*: sentences containing overlapping mentions; *o.l.*: overlapping mentions; *o.l. (s)*: overlapping mentions with the same type.

such path score is defined as the product of the inside score stored in that child node and the score defined over the edge connecting them. The computation of the outside scores can be done in an analogous manner from left to right. It can be verified that the time complexity of this inference procedure for our model is $O(nT)$, which is the same as the mention hypergraph model. Note that, however, both of our models do not have the spurious structures issue, as for any *path* in these models there are no nodes with multiple incoming edges.

During decoding, we perform MAP inference using a max-product procedure that is analogous to how the Viterbi decoding algorithm is used in conventional tree-structured graphical models to find out the highest-scoring subgraph, from which we extract mentions through the process that we call the *interpretation process*. As noted in previous section, there could be multiple mention combinations that correspond to the same sequence of mention separators, which presents an ambiguity during the interpretation process. For these ambiguous cases, we implemented the same interpretation process as that was done in the mention hypergraph model, which is by resolving ambiguous structures as nested mentions. For other cases, there is exactly one way to interpret the structure. For example, in Figure 4b, although there is only one gap marked as starting position (`S`) and two gaps marked as ending position (`EC` and `E`), the interpretation is clear that the two mentions here are "*IL2*" and "*IL2 regulatory region*".

5 Experiments

5.1 Datasets

To assess our model's capability in recognizing overlapping mentions and make comparisons with previous models, we looked at datasets where overlapping mentions are explicitly annotated. Following the previous work (Lu and Roth, 2015), our main results are based on the standard ACE-2004 and ACE-2005 datasets (Doddington et al., 2004). We also additionally looked at the GENIA dataset (Kim et al., 2003), which was used in the previous works (Finkel and Manning, 2009; Lu and Roth, 2015).

For ACE datasets, we used the same splits as used in our previous work (Lu and Roth, 2015), published on our website[6]. For GENIA, we used GENIAcorpus3.02p[7] that comes with POS tags for each word (Tateisi and Tsujii, 2004). Following previous works (Finkel and Manning, 2009; Lu and Roth, 2015), we first split the last 10% of the data as the test set. Next we used the first 80% and the subsequent 10% for training and development, respectively. We made the same modifications as described by Finkel and Manning (2009) by collapsing all *DNA*, *RNA*, and *protein* subtypes into *DNA*, *RNA*, and *protein*, keeping *cell line* and *cell type*, and removing other mention types, resulting in 5 mention types. The statistics of each dataset are shown in Table 1. We can see overlapping mentions are common in such datasets.

For more details on the dataset preprocessing, please refer to the supplemental material.

5.2 Features

For models that fall under the edge-based paradigm (mention hypergraph and our model), we define features over the edges in the models. Features are defined as string concatenations of input features – information extracted over the inputs (such as current word and POS tags of surrounding words) and output features – structured information extracted over the output structure. We carefully defined the input and output features in a way that allows us to make use of the identical set of features for both our mention separator model and the baseline mention hypergraph model, in order to make a proper comparison. We also followed Lu and Roth (2015) to add the additional *mention penalty* feature for our model and all baseline approaches so that we are able to tune F_1-scores on the development set. Roughly speak-

[6] http://statnlp.org/research/ie#mention-hypergraph

[7] http://geniaproject.org/genia-corpus/pos-annotation

	ACE-2004				ACE-2005				ACE-2004 (F_1 optimized)			ACE-2005 (F_1 optimized)		
	P	R	F_1	w/s	P	R	F_1	w/s	P	R	F_1	P	R	F_1
LCRF (single)	70.6	41.7	52.5	40.2	66.0	45.0	53.5	41.2	66.2	47.7	55.4	62.1	48.9	54.7
LCRF (multiple)	78.6	44.5	56.9	119.4	76.2	46.8	58.0	118.7	69.9	55.1	61.6	66.5	55.3	60.4
Lu and Roth (2015)	81.2	45.9	58.6	472.5	78.6	46.9	58.7	516.6	72.5	55.7	63.0	66.3	57.3	61.5
This work (STATE)	78.0	51.2	**61.8**	50.5	75.3	51.7	**61.3**	52.1	71.2	58.0	**64.0**	67.6	58.4	**62.7**
This work (EDGE)	79.5	51.1	**62.2**	251.5	75.5	51.7	**61.3**	253.3	72.7	58.0	**64.5**	69.1	58.1	**63.1**

Table 2: Main results (on ACE).

ing, the weight of this feature controls how confident the model should be in predicting more mentions. In other words, this is a way to balance the precision and recall of the model.

When defining the input features for both our model and the mention hypergraph model, we implemented the features used by previous works in each dataset based on the descriptions in their papers: we followed Lu and Roth (2015) for the features used in ACE datasets, and Finkel and Manning (2009) for features used in GENIA dataset. In general, they include surrounding words, surrounding POS tags, bag-of-words, Brown clusters (for GENIA only), and orthographic features. See the supplemental material for more details.

5.3 Experimental Setup

We trained each model in the training set, then tuned the l_2-regularization parameter based on the development set. For GENIA experiments, we also tuned the number of Brown clusters. Following Lu and Roth (2015), we also used each development set to tune the mention penalty to optimize the F_1-score and report the scores on the corresponding test sets separately. Similar to Finkel and Manning (2009), as another baseline model we also trained a standard linear-chain CRF using the BILOU scheme. Although this model does not support overlapping mentions, it gives us a baseline to see the extent to which our model's ability to recognize overlapping mentions can help the overall performance. There is also a simple extension[8] of this linear-chain CRF model that can support overlapping mentions of *different types* by considering each type separately using multiple chains, one for each type. We call this multiple-chain variant *LCRF (multiple)* and the earlier standard approach *LCRF (single)*. In all models, we also implement the mention penalty feature, adapted accordingly so that increasing the feature weight will increase the number of mentions predicted by the model. See supplemental material for more details.

We implemented all models using Java, and also made additional comparisons on running time by running them under the same machine. In addition, we also analyzed the convergence rate for different models.

6 Results and Discussion

6.1 Results on ACE

Table 2 shows the results on the ACE datasets, and these are our main results. Following previous works (Finkel and Manning, 2009; Lu and Roth, 2015), we report standard precision (P), recall (R) and F_1-score percentage scores. The highest results (F_1-score) and those results that are not significantly different from the highest results are highlighted in bold (based on bootstrap resampling test (Koehn, 2004), where $p > 0.01$). For ACE datasets, we make comparisons with the two versions of the linear-chain CRF baseline: LCRF (single) which does not support overlapping mentions at all and LCRF (multiple) which does not support overlapping mentions of the same type, as well as our implementation of the mention hypergraph baseline (Lu and Roth, 2015).

From such empirical results we can see that our proposed model using mention separators consistently yields significantly better results ($p < 0.01$) than the mention hypergraph model across these two datasets, under two setups (whether to optimize F_1-score or not). Specifically, when the state-based approach is used (STATE), our approach is able to obtain a much higher recall, resulting in improved F_1-score. Empirically, we found this approach was also faster than the LCRF baseline approach in terms of the number of words processed each second (w/s) during decoding, which is expected, since STATE uses fewer num-

[8] We also tried a more elaborate encoding scheme based on BIO scheme Tang et al. (2013), originally designed for discontiguous mentions, but is supposed to be able to also recognize overlapping mentions of the same type. However, the result is very similar to LCRF (multiple), perhaps due to the invalid structures issue noted by Muis and Lu (2016).

	P	R	F_1	w/s
LCRF (single)	77.1	63.3	69.5	81.6
LCRF (multiple)	75.9	66.1	**70.6**	175.8
Finkel and Manning (2009)	75.4	65.9	70.3	-
Lu and Roth (2015)	74.2	66.7	70.3	931.9
This work (STATE)	74.0	67.7	**70.7**	110.8
This work (EDGE)	75.4	66.8	**70.8**	389.2

Table 3: Results on GENIA.

ber of tags.[9] The edge-based approach (EDGE) using our proposed multigraph representation is able to achieve a significant speedup in comparison with the state-based approach. Although this model is still about 50% slower than the mention hypergraph model[10], but it yielded a significantly higher F_1-score (up to 3.6 points higher on ACE-2004 before optimizing F_1-score). These results largely confirm the effectiveness of our proposed mention separator model and the usefulness of the multigraph representation for learning the model.

And as expected, the LCRF baselines yields relatively lower results compared to the other models, since it cannot predict overlapping mentions.[11] However, such results give us some idea on how much performance increase we can gain by properly recognizing overlapping mentions by looking at the results of LCRF (single), which in this case can be up to 9.7 points in F_1-score in ACE-2004. We can also see the gain from recognizing overlapping mentions of the same type by looking at the results of LCRF (multiple), which can be up to 5.3 points in F_1-score in ACE-2004.

6.2 Results on GENIA

Table 3 shows the results of running the models with F_1-score tuning on GENIA dataset. All models include Brown clustering features learned from PubMed abstracts. Besides the mention hypergraph baseline, we also make comparisons with the system of Finkel and Manning (2009) that can also support overlapping mentions.

We see that the mention hypergraph model matches the performance of the constituency parser-based model of Finkel and Manning (2009), while our models based on mention separators yield significantly higher scores ($p < 0.05$) than all other baselines (except LCRF (multiple), which we will discuss shortly). There are two observations worth mentioning: (1) the absolute difference of F_1-scores of our models and the baseline models in GENIA is much smaller compared to that in ACE datasets, and (2) the LCRF (multiple) model in GENIA dataset can achieve higher scores compared to other more complex baseline models, although LCRF (multiple) does not support overlapping mentions of the same type. We suspect that these two observations are due to the small proportion of overlapping mentions in GENIA (18%, as compared to >40% in ACE datasets, see Table 1). To investigate this, we conduct a few more sets of experiments.

		%	Lu and Roth (2015)			This work (EDGE)		
			P	R	F_1	P	R	F_1
ACE-2004	O	42	72.5	52.4	60.8	72.1	55.3	62.6
	Ø	58	72.5	65.0	68.6	74.1	65.5	69.5
ACE-2005	O	32	68.1	52.6	59.4	70.4	55.0	61.8
	Ø	68	64.1	65.1	64.6	67.2	63.4	65.2
GENIA	O	24	76.3	60.8	67.7	76.5	60.3	67.4
	Ø	76	73.1	70.7	71.9	74.8	71.3	73.0

Table 4: Results on different types of sentences.

6.3 Further Experiments

On different types of sentences: As these datasets consist of both overlapping and non-overlapping mentions, to further understand the model's effectiveness in recognizing overlapping mentions (and non-overlapping mentions), we performed some additional experiments on the mention hypergraph model and our model.[12] Specifically, we split the test data into two portions, one that consists of only sentences that contain overlapping mentions (O) and those which do not (Ø). The results are shown in Table 4.

We can see that in ACE datasets, our model achieves higher F_1-scores compared to the mention hypergraph for both portions, but it achieves slightly lower results in GENIA dataset for the portion that contains overlapping mentions. We believe that our models learn parameters so as to obtain an optimal overall performance, and since the proportion of the overlapping mentions in GENIA is much smaller compared to that in ACE datasets, it learns to focus more on the non-overlapping mentions. This is supported by the fact that the difference of F_1-score between the mention hypergraph model and our model in GENIA is larger compared to the difference in ACE

[9] There are eight tags in STATE and nine in LCRF.

[10] Though both models have the same time complexity, they differ by a constant factor.

[11] LCRF (single) cannot predict any overlapping mentions, while LCRF (multiple) cannot predict overlapping mentions of the same type.

[12] We also performed this on other models. Due to space constraint, we do not include the results here. See the supplemental material for more details.

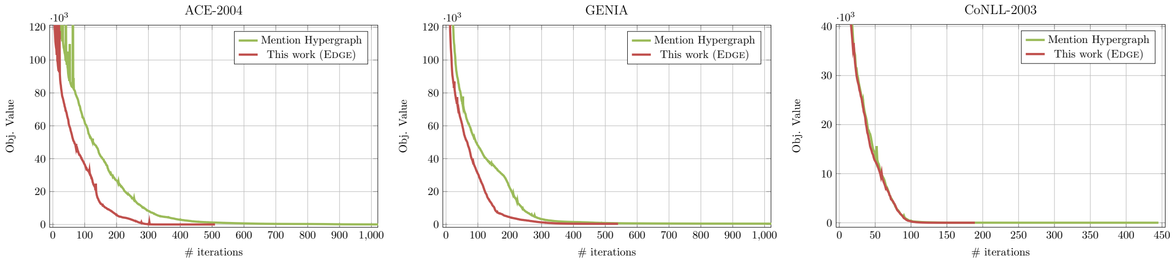


Figure 7: Objective vs. training iterations.

	P	R	F_1	w/s
LCRF (single)	84.2	83.5	**83.8**	148.6
LCRF (multiple)	91.5	78.2	**84.3**	283.4
Ratinov and Roth (2009)	-	-	83.7	-
Lu and Roth (2015)	91.1	77.0	83.5	1169.7
This work (STATE)	91.1	78.2	**84.2**	116.3
This work (EDGE)	91.3	78.2	**84.3**	554.0

Table 5: Results on CoNLL-2003 (without optimizing F_1-score).

datasets (1.1 points in GENIA, compared to 0.9 and 0.6 points in ACE).

These results also lead to the interesting empirical finding that our model appears to be able to do well also on recognizing non-overlapping mentions. This motivates us to conduct the next set of experiments.

On data without overlapping mentions: We also performed one additional set of experiments, on the standard CoNLL-2003 dataset (Tjong Kim Sang and De Meulder, 2003), which has no overlapping mentions.

The results (without optimizing F_1-score) are shown in Table 5. We see that our models based on mention separators outperform baseline models such as the Illinois NER system where external resources are not used (Ratinov and Roth, 2009), and a linear-chain CRF model, although the linear-chain CRF baseline models some interactions between distinct mention types and our models do not. Such results also suggest that modeling the interactions between distinct mention types may not be crucial to get a good performance in mention recognition. This is further corroborated by the result of LCRF (multiple), which is higher than the result of LCRF (single) by about 0.5 points.

When comparing our model against the mention hypergraph model, we note that our model consistently yields a higher recall. We speculate this is due to the fact that as our model does not exhibit the issue of spurious structures we discussed in Section 3.1, it is more confident in making its predictions.

On convergence: We also empirically analyzed the convergence properties of the two models. Empirically, as illustrated in Figure 7 which shows how the objective improves when the training progresses on ACE-2004, GENIA, and CoNLL-2003, we found that our EDGE-based model requires significantly less iterations to converge than the mention hypergraph on the former two datasets which contain overlapping mentions. We believe it is possible that this slower convergence is due to the spurious structures issue in mention hypergraphs, which causes the objective function to be more complex to optimize. However, some further analyses on the convergence issue and the impact of different ways of exploiting features (over different hyperedges) for the hypergraph-based models are needed.

7 Conclusion and Future Work

We proposed the novel *mention separators* for mention recognition where mentions may overlap with one another. We also proposed two ways these mention separators can be utilized to encode overlapping mentions, where one of them utilizes a novel multigraph-based representation. We showed that by utilizing mention separators, we can get better recognition results compared to previous models, and by utilizing the multigraph representation, we can maintain a good inference speed, albeit still slower than the mention hypergraph model. We also performed theoretical analysis on the model and showed that our model does not present the *spurious structures* issue associated with a previous state-of-the-art model, while still keeping the same inference time complexity.

Future work includes further investigations on how to apply the multigraph approach to other structured prediction tasks, as well as applications of the proposed model in other related NLP tasks that involve the prediction of overlapping structures, such as equation parsing (Roy et al., 2016).

The code used in this paper is available at http://statnlp.org/research/ie/.

Acknowledgments

We thank all the reviewers for their useful feedback to the earlier draft of this paper. This work is supported by MOE Tier 1 grant SUTDT12015008.

References

Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising Nested Named Entities in Biomedical Text. In *Proc. of the Workshop on BioNLP 2007*, June, pages 65–72.

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A Constrained Latent Variable Model for Coreference Resolution. In *Proc. of EMNLP*.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation. *LREC*, 2(1):837–840.

Jenny Rose Finkel and Christopher D. Manning. 2009. Nested Named Entity Recognition. In *Proc. of EMNLP*, page 141.

Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, H Nicolov, and Salim Roukos. 2004. A Statistical Model for Multilingual Entity Detection and Tracking. In *Proc. of HLT-NAACL*.

Pankaj Gupta and Bernt Andrassy. 2016. Table Filling Multi-Task Recurrent Neural Network for Joint Entity and Relation Extraction. In *Proc. of COLING*.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA Corpus–A Semantically Annotated Corpus for Bio-textmining. *Bioinformatics*, 19(Suppl 1):i180–i182.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP*, pages 388–395.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with Support Vector Machines. In *Proc. of NAACL*, volume 816, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. of ICML*, pages 282–289.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint Event Extraction via Structured Prediction with Global Features. In *Proc. of ACL*, pages 73–82.

Xiao Ling and Daniel S Weld. 2012. Fine-Grained Entity Recognition. In *Proc. of AAAI*.

Dong C. Liu and Jorge Nocedal. 1989. On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical Programming*, 45(1-3):503–528.

Jing Lu, Deepak Venugopal, Vibhav Gogate, and Vincent Ng. 2016. Joint Inference for Event Coreference Resolution. In *Proc. of COLING*, pages 3264–3275.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 835–844, Jeju Island, Korea. Association for Computational Linguistics.

Wei Lu and Dan Roth. 2015. Joint Mention Extraction and Classification with Mention Hypergraphs. In *Proc. of EMNLP*, pages 857–867. Association for Computational Linguistics.

Andrew McCallum and Wei Li. 2003. Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-enhanced Lexicons. In *Proc. of HLT-NAACL*, volume 4, pages 188–191, Morristown, NJ, USA. Association for Computational Linguistics.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Flexible Text Segmentation with Structured Multilabel Classification. In *Proc. of HLT-EMNLP*, October, pages 987–994, Morristown, NJ, USA. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant Supervision for Relation Extraction without Labeled Data. In *Proc. of ACL-IJCNLP*, pages 1003–1011.

Diego Mollá, Menno Van Zaanen, and Steve Cassidy. 2007. Named Entity Recognition in Question Answering of Speech Data. In *Proc. of the Australasian Language Technology Workshop*, pages 57–65.

Aldrian Obaja Muis and Wei Lu. 2016. Learning to Recognize Discontiguous Entities. In *Proc. of EMNLP*, pages 75–84, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Nadeau and Satoshi Sekine. 2007. A Survey of Named Entity Recognition and Classification. *Lingvisticae Investigationes*, 30(1):3–26.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint Event Extraction via Recurrent Neural Networks. In *Proc. of NAACL-HLT*, pages 300–309.

Lev Ratinov and Dan Roth. 2009. Design Challenges and Misconceptions in Named Entity Recognition. In *Proc. of CoNLL*, page 147. Association for Computational Linguistics.

Subhro Roy, Shyam Upadhyay, and Dan Roth. 2016. Equation Parsing : Mapping Sentences to Grounded Equations. In *Proc. of EMNLP*, pages 1088–1097, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sunita Sarawagi and William W. Cohen. 2004. Semi-Markov Conditional Random Fields for Information Extraction. In *Proc. of NIPS*, pages 1185–1192.

Hong Shen and Anoop Sarkar. 2005. Voting Between Multiple Data Representations for Text Chunking. In *Advances in Artificial Intelligence*, volume 3501, chapter 40, pages 389–400. Springer-Verlag, Berlin, Heidelberg.

Hanna Suominen, Sanna Salanterä, Sumithra Velupillai, Wendy W. Chapman, Guergana Savova, Noemie Elhadad, Sameer Pradhan, Brett R. South, Danielle L. Mowery, Gareth J. F. Jones, Johannes Leveling, Liadh Kelly, Lorraine Goeuriot, David Martinez, and Guido Zuccon. 2013. Overview of the ShARe/CLEF eHealth Evaluation Lab 2013. In P. Forner, editor, *Information Access Evaluation: Multilinguality, Multimodality, and Visualization*, volume 8138, chapter 24, pages 212–231. Springer-Verlag, Berlin, Heidelberg.

Buzhou Tang, Yonghui Wu, Min Jiang, Joshua C. Denny, and Hua Xu. 2013. Recognizing and Encoding Disorder Concepts in Clinical Text using Machine Learning and Vector Space. In *Proc. of the ShARe/CLEF Evaluation Lab*.

Yuka Tateisi and Jun'ichi Tsujii. 2004. Part-of-Speech Annotation of Biology Research Abstracts. In *Proc. of LREC*, pages 1267–1270.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition. In *Proc. of HLT-NAACL*, pages 142–147.

Deep Joint Entity Disambiguation with Local Neural Attention

Octavian-Eugen Ganea and **Thomas Hofmann**
Department of Computer Science
ETH Zurich
{octavian.ganea,thomas.hofmann}@inf.ethz.ch

Abstract

We propose a novel deep learning model for joint document-level entity disambiguation, which leverages learned neural representations. Key components are entity embeddings, a neural attention mechanism over local context windows, and a differentiable joint inference stage for disambiguation. Our approach thereby combines benefits of deep learning with more traditional approaches such as graphical models and probabilistic mention-entity maps. Extensive experiments show that we are able to obtain competitive or state-of-the-art accuracy at moderate computational costs.

1 Introduction

Entity disambiguation (ED) is an important stage in text understanding which automatically resolves references to entities in a given knowledge base (KB). This task is challenging due to the inherent ambiguity between surface form mentions such as names and the entities they refer to. This many-to-many ambiguity can often be captured partially by name-entity co-occurrence counts extracted from entity-linked corpora.

ED research has largely focused on two types of contextual information for disambiguation: *local* information based on words that occur in a context window around an entity mention, and, *global* information, exploiting document-level coherence of the referenced entities. Many state-of-the-art methods aim to combine the benefits of both, which is also the philosophy we follow in this paper. What is specific to our approach is that we use embeddings of entities as a common representation to assess local as well as global evidence.

In recent years, many text and language understanding tasks have been advanced by neural network architectures. However, despite recent work, competitive ED systems still largely employ manually designed features. Such features often rely on domain knowledge and may fail to capture all relevant statistical dependencies and interactions. The explicit goal of our work is to use deep learning in order to learn basic features and their combinations from scratch. To the best of our knowledge, our approach is the first to carry out this program with full rigor.

2 Contributions and Related Work

There is a vast prior research on entity disambiguation, highlighted by (Ji, 2016). We will focus here on a discussion of our main contributions in relation to prior work.

Entity Embeddings. We have developed a simple, yet effective method to embed entities and words in a common vector space. This follows the popular line of work on word embeddings, e.g. (Mikolov et al., 2013; Pennington et al., 2014), which was recently extended to entities and ED by (Yamada et al., 2016; Fang et al., 2016; Zwicklbauer et al., 2016; Huang et al., 2015). In contrast to the above methods that require data about entity-entity co-occurrences which often suffers from sparsity, we rather bootstrap entity embeddings from their canonical entity pages and local context of their hyperlink annotations. This allows for more efficient training and alleviates the need to compile co-linking statistics. These vector representations are a key component to avoid hand-engineered features, multiple disambiguation steps, or the need for additional *ad hoc* heuristics when solving the ED task.

Context Attention. We present a novel attention mechanism for local ED. Inspired by mem-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2619–2629
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

ory networks of (Sukhbaatar et al., 2015) and insights of (Lazic et al., 2015), our model deploys attention to select words that are informative for the disambiguation decision. A learned combination of the resulting context-based entity scores and a mention–entity prior yields the final local scores. Our local model achieves better accuracy than the local probabilistic model of (Ganea et al., 2016), as well as the feature-engineered local model of (Globerson et al., 2016). As an added benefit, our model has a smaller memory footprint and it's very fast for both training and testing.

There have been other deep learning approaches to define local context models for ED. For instance (Francis-Landau et al., 2016; He et al., 2013) use convolutional neural networks (CNNs) and stacked denoising auto-encoders, respectively, to learn representations of textual documents and canonical entity pages. Entities for each mention are locally scored based on cosine similarity with the respective document embedding. In a similar local setting, (Sun et al., 2015) embed mentions, their immediate contexts and their candidate entities using word embeddings and CNNs. However, their entity representations are restrictively built from entity titles and entity categories only. Unfortunately, the above models are rather 'black-box' (as opposed to ours which reveals the attention focus) and were never extended to perform joint document disambiguation.

Collective Disambiguation. Last, a novel deep learning architecture for global ED is proposed. Mentions in a document are resolved jointly, using a conditional random field (Lafferty et al., 2001) with parametrized potentials. We suggest to learn the latter by casting loopy belief propagation (LBP) (Murphy et al., 1999) as a rolled-out deep network. This is inspired by similar approaches in computer vision, e.g. (Domke, 2013), and allows us to backpropagate through the (truncated) message passing, thereby optimizing the CRF potentials to work well in conjunction with the inference scheme. Our model is thus trained end-to-end with the exception of the pre-trained word and entity embeddings. Previous work has investigated different approximation techniques, including: random graph walks (Guo and Barbosa, 2016), personalized PageRank (Pershina et al., 2015), intermention voting (Ferragina and Scaiella, 2010), graph pruning (Hoffart et al., 2011), integer linear programming (Cheng and Roth, 2013), or ranking SVMs (Ratinov et al., 2011). Mostly connected to our approach is (Ganea et al., 2016) where LBP is used for inference (but not learning) in a probabilistic graphical model and (Globerson et al., 2016) where a single round of message passing with attention is performed. To our knowledge, we are one of the first to investigate differentiable message passing for NLP problems.

3 Learning Entity Embeddings

In a first step, we propose to train entity vectors that can be used for the ED task (and potentially for other tasks). These embeddings compress the semantic meaning of entities and drastically reduce the need for manually designed features or co-occurrence statistics.

Entity embeddings are bootstrapped from word embeddings and are trained independently for each entity. A few arguments motivate this decision: (i) there is no need for entity co-occurrence statistics that suffer from sparsity issues and/or large memory footprints; (ii) vectors of entities in a subset domain of interest can be trained separately, obtaining potentially significant speed-ups and memory savings that would otherwise be prohibitive for large entity KBs;[1] (iii) entities can be easily added in an incremental manner, which is important in practice; (iv) the approach extends well into the tail of rare entities with few linked occurrences; (v) empirically, we achieve better quality compared to methods that use entity co-occurrence statistics.

Our model embeds words and entities in the same low-dimensional vector space in order to exploit geometric similarity between them. We start with a pre-trained word embedding map $\mathbf{x} : \mathcal{W} \to \mathbb{R}^d$ that is known to encode semantic meaning of words $w \in \mathcal{W}$; specifically we use word2vec pretrained vectors (Mikolov et al., 2013). We extend this map to entities $\mathcal{E}$, i.e. $\mathbf{x} : \mathcal{E} \to \mathbb{R}^d$, as described below.

We assume a generative model in which words that co-occur with an entity e are sampled from a conditional distribution $p(w|e)$ when they are generated. Empirically, we collect word-entity co-occurrence counts $\#(w, e)$ from two sources: (i) the canonical KB description page of the entity (e.g. entity's Wikipedia page in our case), and (ii) the windows of fixed size surrounding mentions of the entity in an annotated corpus (e.g. Wikipedia

[1] Notably useful with (limited memory) GPU hardware.

hyperlinks in our case). These counts define a practical approximation of the above word-entity conditional distribution, i.e. $\hat{p}(w|e) \propto \#(w,e)$. We call this the ”positive” distribution of words related to the entity. Next, let $q(w)$ be a generic word probability distribution which we use for sampling ”negative” words unrelated to a specific entity. As in (Mikolov et al., 2013), we choose a smoothed unigram distribution $q(w) = \hat{p}(w)^\alpha$ for some $\alpha \in (0,1)$. The desired outcome is that vectors of positive words are closer (in terms of dot product) to the embedding of entity e compared to vectors of random words. Let $w^+ \sim \hat{p}(w|e)$ and $w^- \sim q(w)$. Then, we use a max-margin objective to infer the optimal embedding for entity e:

$$\begin{aligned} J(\mathbf{z}; e) &:= \mathbb{E}_{w^+|e}\, \mathbb{E}_{w^-} \left[h\left(\mathbf{z}; w^+, w^-\right)\right] \\ h(\mathbf{z}; w, v) &:= [\gamma - \langle \mathbf{z}, \mathbf{x}_w - \mathbf{x}_v \rangle]_+ \\ \mathbf{x}_e &:= \underset{\mathbf{z}:\|\mathbf{z}\|=1}{\arg\min}\, J(\mathbf{z}; e) \end{aligned} \quad (1)$$

where $\gamma > 0$ is a margin parameter and $[\cdot]_+$ is the ReLU function. The above loss is optimized using stochastic gradient descent with projection over sampled pairs (w^+, w^-). Note that the entity vector is directly optimized on the unit sphere which is important in order to obtain qualitative embeddings.

We empirically assess the quality of our entity embeddings on entity similarity and ED tasks as detailed in Section 7 and Appendix A. The technique described in this section can also be applied, in principle, for computing embeddings of general text documents, but a comparison with such methods is left as future work.

4 Local Model with Neural Attention

We now explain our local ED approach that uses word and entity embeddings to steer a neural attention mechanism. We build on the insight that only a few context words are informative for resolving an ambiguous mention, something that has been exploited before in (Lazic et al., 2015). Focusing only on those words helps reducing noise and improves disambiguation. (Yamada et al., 2016) observe the same problem and adopt the restrictive strategy of removing all non-nouns. Here, we assume that a context word may be relevant, if it is strongly related to at least one of the entity candidates of a given mention.

Context Scores.

Let us assume that we have computed a mention–entity prior $\hat{p}(e|m)$ (procedure detailed in Section 6). In addition, for each mention m, a pruned candidate set $\Gamma(m)$ of at most S entities has been identified. Our model, depicted in Figure 1, computes a score for each $e \in \Gamma(m)$ based on the K-word local context $c = \{w_1, \ldots, w_K\}$ surrounding m, as well as on the prior. It is a composition of differentiable functions, thus it is smooth from input to output, allowing us to easily compute gradients and backpropagate through it.

Each word $w \in c$ and entity $e \in \Gamma(m)$ is mapped to its embedding via the pre-trained map $\mathbf{x}$ (cf. Section 3). We then compute an unnormalized support score for each word in the context as follows:

$$u(w) = \max_{e \in \Gamma(m)} \mathbf{x}_e^\top \mathbf{A} \mathbf{x}_w \quad (2)$$

where $\mathbf{A}$ is a parameterized diagonal matrix. The weight is high if the word is strongly related to at least one candidate entity. We often observe that uninformative words (e.g. similar to stop words) receive non-negligible scores which add undesired noise to our local context model. As a consequence, we (hard) prune to the top $R \leq K$ words with the highest scores[2] and apply a softmax function on these weights. Define the reduced context:

$$\bar{c} = \{w \in c | u(w) \in \text{topR}(\mathbf{u})\} \quad (3)$$

Then, the final attention weights are explicitly

$$\beta(w) = \begin{cases} \frac{\exp[u(w)]}{\sum_{v \in \bar{c}} \exp[u(v)]}. & \text{if } w \in \bar{c} \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

Finally, we define a β-weighted context-based entity-mention score via

$$\Psi(e, c) = \sum_{w \in \bar{c}} \beta(w)\, \mathbf{x}_e^\top \mathbf{B}\, \mathbf{x}_w \quad (5)$$

where $\mathbf{B}$ is another trainable diagonal matrix. We will later use the same architecture for the *unary* scores of our global ED model.

Local Score Combination.

We integrate these context scores with the context-independent scores encoded in $\hat{p}(e|m)$.

[2] We implement this in a differentiable way by setting the lowest K-R attention weights in $\mathbf{u}$ to $-\infty$ and applying a vanila softmax on top of them. We used the layers Threshold and TemporalDynamicKMaxPooling from Torch nn package, which allow subgradient computation.

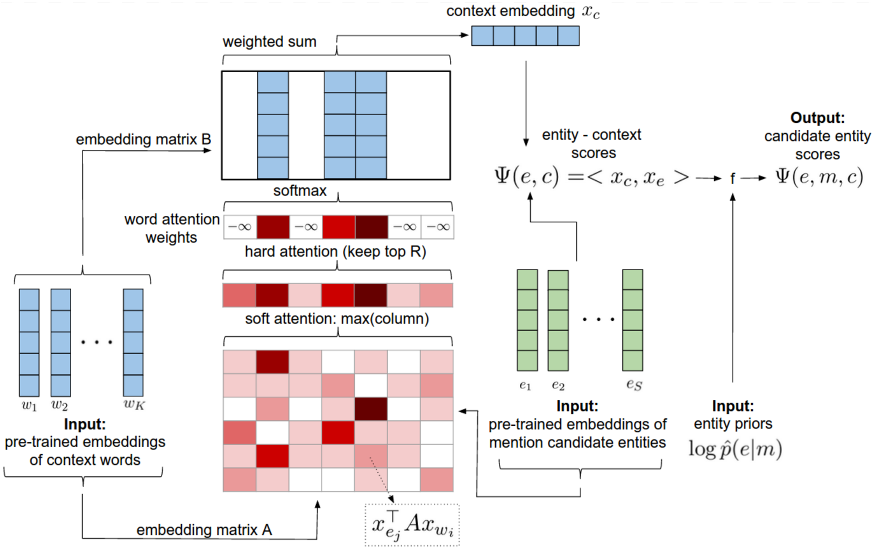


Figure 1: Local model with neural attention. Inputs: context word vectors, candidate entity priors and embeddings. Outputs: entity scores. All parts are differentiable and trainable with backpropagation.

Our final (unnormalized) local model is a combination of both $\Psi(e,c)$ and $\log \hat{p}(e|m)$:

$$\Psi(e,m,c) = f(\Psi(e,c), \log \hat{p}(e|m)) \qquad (6)$$

We find a flexible choice for f to be important and superior to a naïve weighted average combination model. We therefore use a neural network with two fully connected layers of 100 hidden units and ReLU non-linearities, which we regularize as suggested in (Denton et al., 2015) by constraining the sum of squares of all weights in the linear layer. We use standard projected SGD for training. The same network is also used in Section 5.

Prediction is done independently for each mention m_i and context c_i by maximizing the $\Psi(e, m_i, c_i)$ score.

Learning the Local Model.

Entity and word embeddings are pre-trained as discussed in Section 3. Thus, the only learnable parameters are the diagonal matrices $\mathbf{A}$ and $\mathbf{B}$, plus the parameters of f. Having few parameters helps to avoid overfitting and to be able to train with little annotated data. We assume that a set of known mention-entity pairs $\{(m, e^*)\}$ with their respective context windows have been extracted from a corpus. For model fitting, we then utilize a max-margin loss that ranks ground truth entities higher than other candidate entities. This leads us to the objective:

$$\theta^* = \arg\min_{\theta} \sum_{D\in\mathcal{D}} \sum_{m\in D} \sum_{e\in\Gamma(m)} g(e,m), \qquad (7)$$

$$g(e,m) := [\gamma - \Psi(e^*, m, c) + \Psi(e,m,c)]_+$$

where $\gamma > 0$ is a margin parameter and $\mathcal{D}$ is a training set of entity annotated documents. We aim to find a Ψ (i.e. parameterized by θ) such that the score of the correct entity e^* referenced by m is at least a margin γ higher than that of any other candidate entity e. Whenever this is not the case, the margin violation becomes the experienced loss.

5 Document-Level Deep Model

Next, we address global ED assuming document coherence among entities. We therefore introduce the notion of a document as consisting of a set of mentions $\mathbf{m} = m_1, \ldots, m_n$, along with their context windows $\mathbf{c} = c_1, \ldots c_n$. Our goal is to define a joint probability distribution over $\Gamma(m_1) \times \ldots \times \Gamma(m_n) \ni \mathbf{e}$. Each such $\mathbf{e}$ selects one candidate entity for each mention in the document. Obviously, the state space of $\mathbf{e}$ grows exponentially in the number of mentions n.

CRF Model.

Our model is a fully-connected pairwise conditional random field, defined on the log scale as

$$g(\mathbf{e},\mathbf{m},\mathbf{c}) = \sum_{i=1}^{n} \Psi_i(e_i) + \sum_{i<j} \Phi(e_i, e_j) \qquad (8)$$

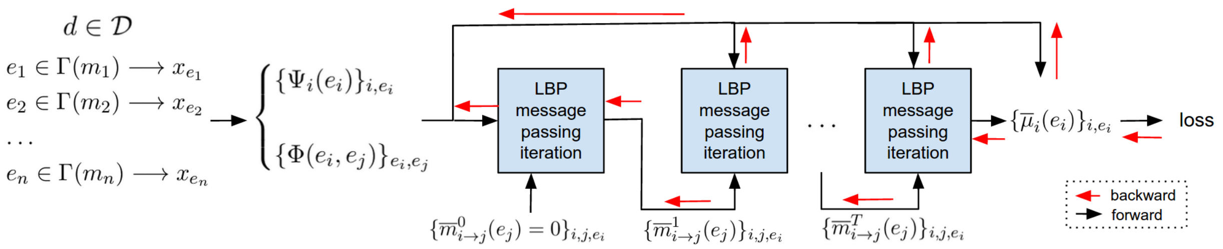

$d \in \mathcal{D}$
$e_1 \in \Gamma(m_1) \longrightarrow x_{e_1}$
$e_2 \in \Gamma(m_2) \longrightarrow x_{e_2}$
$\dots$
$e_n \in \Gamma(m_n) \longrightarrow x_{e_n}$
$\{\Psi_i(e_i)\}_{i,e_i}$
$\{\Phi(e_i,e_j)\}_{e_i,e_j}$
LBP message passing iteration
LBP message passing iteration
$\dots$
LBP message passing iteration
$\{\overline{\mu}_i(e_i)\}_{i,e_i}$
loss
$\{\overline{m}^0_{i\to j}(e_j)=0\}_{i,j,e_i}$
$\{\overline{m}^1_{i\to j}(e_j)\}_{i,j,e_i}$
$\{\overline{m}^T_{i\to j}(e_j)\}_{i,j,e_i}$
backward
forward

Figure 2: Global model: unrolled LBP deep network that is end-to-end differentiable and trainable.

The unary factors are the local scores $\Psi_i(e_i) = \Psi(e_i, c_i)$ described in Eq. (5). The pairwise factors are bilinear forms of the entity embeddings

$$\Phi(e, e') = \frac{2}{n-1}\, \mathbf{x}_e^\top \mathbf{C}\, \mathbf{x}_{e'}\,, \tag{9}$$

where $\mathbf{C}$ is a diagonal matrix. Similar to (Ganea et al., 2016), the above normalization helps balancing the unary and pairwise terms across documents with different numbers of mentions.

The function value $g(\mathbf{e}, \mathbf{m}, \mathbf{c})$ is supposedly high for semantically related sets of entities that also have local support. The goal of a global ED prediction method is to perform maximum-a-posteriori on this CRF to find the set of entities $\mathbf{e}$ that maximize $g(\mathbf{e}, \mathbf{m}, \mathbf{c})$.

Differentiable Inference.

Training and prediction in binary CRF models as the one above is NP-hard. Therefore, in learning one usually maximizes a likelihood approximation and during operations (i.e. in prediction) one may use an approximate inference procedure, often based on message-passing. Among many challenges of these approaches, it is worth pointing out that weaknesses of the approximate inference procedure are generally not captured during learning. Inspired by (Domke, 2011, 2013), we use *truncated fitting* of loopy belief propagation (LBP) to a fixed number of message passing iterations. Our model directly optimizes the marginal likelihoods, using the same networks for learning and prediction. As noted by (Domke, 2013), this method is robust to model mis-specification, avoids inherent difficulties of partition functions and is faster compared to double-loop likelihood training (where, for each stochastic update, inference is run until convergence is achieved).

Our architecture is shown in Figure 2. A neural network with T layers encodes T message passing iterations of synchronous max-product LBP[3] which is designed to find the most likely (MAP) entity assignments that maximize $g(\mathbf{e}, \mathbf{m}, \mathbf{c})$. We also use message damping, which is known to speed-up and stabilize convergence of message passing. Formally, in iteration t, mention m_i votes for entity candidate $e \in \Gamma(m_j)$ of mention m_j using the normalized log-message $\overline{m}^t_{i\to j}(e)$ computed as:

$$m^{t+1}_{i\to j}(e) = \max_{e' \in \Gamma(m_i)} \Big\{\Psi_i(e') + \Phi(e, e') + \sum_{k \neq j} \overline{m}^t_{k\to i}(e')\Big\}\,. \tag{10}$$

Herein the first part just reflects the CRF potentials, whereas the second part is defined as

$$\overline{m}^t_{i\to j}(e) = \log[\delta \cdot \mathrm{softmax}(m^t_{i\to j}(e)) + (1-\delta)\cdot \exp(\overline{m}^{t-1}_{i\to j}(e))] \tag{11}$$

where $\delta \in (0, 1]$ is a damping factor. Note that, without loss of generality, we simplify the LBP procedure by dropping the factor nodes. The messages at first iteration (layer) are set to zero.

After T iterations (network layers), the beliefs (marginals) are computed as:

$$\mu_i(e) = \Psi_i(e) + \sum_{k \neq i} \overline{m}^T_{k\to i}(e) \tag{12}$$

$$\overline{\mu}_i(e) = \frac{\exp[\mu_i(e)]}{\sum_{e' \in \Gamma(m_i)} \exp[\mu_i(e')]} \tag{13}$$

Similar to the local case, we obtain accuracy improvement when combining the mention-entity prior $\hat{p}(e|m)$ with marginal $\mu_i(e)$ using the same non-linear combination function f from Equation 6 as follows:

$$\rho_i(e) := f(\overline{\mu}_i(e), \log \hat{p}(e|m_i)) \tag{14}$$

The learned function f for global ED is non-trivial (see Figure 3), showing that the influence of the prior tends to weaken for larger $\mu(e)$,

[3] Sum-product and mean-field performed worse in our experiments.

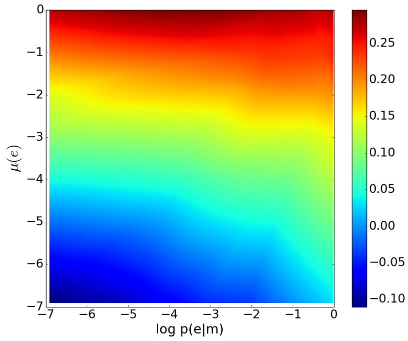


Figure 3: Non-linear scoring function of the belief and mention prior learned with a neural network. Achieves a 1.7% improvement on AIDA-B dataset compared to a weighted average scheme.

whereas it has a dominating influence whenever the document-level evidence is weak. We also experimented with the prior integrated directly inside the unary factors $\Psi_i(e_i)$, but results were worse because, in some cases, the global entity interaction is not able to recover from strong incorrect priors (e.g. country names have a strong prior towards the respective countries as opposed to national sports teams).

Parameters of our global model are the diagonal matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and the weights of the f network. As before, we find a margin based objective to be the most effective and we suggest to fit parameters by minimizing a ranking loss[4] defined as:

$$L(\theta) = \sum_{D \in \mathcal{D}} \sum_{m_i \in D} \sum_{e \in \Gamma(m_i)} h(m_i, e) \qquad (15)$$

$$h(m_i, e) = [\gamma - \rho_i(e_i^*) + \rho_i(e)]_+ \qquad (16)$$

Computing this objective is trivial by running T times the steps described by Eqs. (10), (11), followed in the end by the step in Eq. (13). Each step is differentiable and the gradient of the model parameters can be computed on the resulting marginals and back-propagated over messages using chain rule.

At test time, marginals $\rho_i(e)$ are computed jointly per document using this network, but prediction is done independently for each mention m_i by maximizing its respective marginal score.

6 Candidate Selection

We use a mention-entity prior $\hat{p}(e|m)$ both as a feature and for entity candidate selection. It is computed by averaging probabilities from two indexes build from mention entity hyperlink count statistics from Wikipedia and a large Web corpus (Spitkovsky and Chang, 2012). Moreover, we add the YAGO dictionary of (Hoffart et al., 2011), where each candidate receives a uniform prior.

Metric Method	NDCG@1	NDCG@5	NDCG@10	MAP
WikiLinkMeasure (WLM)	0.54	0.52	0.55	0.48
(Yamada et al., 2016) d = 500	0.59	0.56	0.59	0.52
our (canonical pages) d = 300	0.624	0.589	0.615	0.549
our (canonical&hyperlinks) d = 300	**0.632**	**0.609**	**0.641**	**0.578**

Table 1: Entity relatedness results on the test set of (Ceccarelli et al., 2013). WLM is a well-known similarity method of (Milne and Witten, 2008).

Dataset	**Number mentions**	**Number docs**	**Mentions per doc**	**Gold recall**
AIDA-train	18448	946	19.5	-
AIDA-A (valid)	4791	216	22.1	96.9%
AIDA-B (test)	4485	231	19.4	98.2%
MSNBC	656	20	32.8	98.5%
AQUAINT	727	50	14.5	94.2%
ACE2004	257	36	7.1	90.6%
WNED-CWEB	11154	320	34.8	91.1%
WNED-WIKI	6821	320	21.3	92%

Table 2: Statistics of ED datasets. *Gold recall* is the percentage of mentions for which the entity candidate set contains the ground truth entity. We only train on mentions with at least one candidate.

Candidate selection, i.e. construction of $\Gamma(e)$, is done for each input mention as follows: first, the top 30 candidates are selected based on the prior $\hat{p}(e|m)$. Then, in order to optimize for memory and run time (LBP has complexity quadratic in S), we keep only 7 of these entities based on the following heuristic: (i) the top 4 entities based on $\hat{p}(e|m)$ are selected, (ii) the top 3 entities based on the local context-entity similarity measured using the function from Eq. 5 are selected.[5]. We refrain from annotating mentions without any candidate entity, implying that precision and recall can be different in our case.

In a few cases, generic mentions of persons (e.g. ”Peter”) are coreferences of more specific mentions (e.g. ”Peter Such”) from the same document. We employ a simple heuristic to address this issue: for each mention m, if there exist mentions of persons that contain m as a continuous subse-

[4] Optimizing a marginal log-likelihood loss function performed worse.

[5] We have used a simpler context vector here computed by simply averaging all its constituent word vectors.

Methods	AIDA-B
Local models	
prior $\hat{p}(e\|m)$	71.9
(Lazic et al., 2015)	86.4
(Globerson et al., 2016)	87.9
(Yamada et al., 2016)	87.2
our (local, K=100, R=50)	**88.8**
Global models	
(Huang et al., 2015)	86.6
(Ganea et al., 2016)	87.6
(Chisholm and Hachey, 2015)	88.7
(Guo and Barbosa, 2016)	89.0
(Globerson et al., 2016)	91.0
(Yamada et al., 2016)	91.5
our (global)	**92.22 ± 0.14**

Table 3: In-KB accuracy for AIDA-B test set. All baselines use KB+YAGO mention-entity index. For our method we show 95% confidence intervals obtained over 5 runs.

quence of words, then we consider the merged set of the candidate sets of these specific mentions as the candidate set for the mention m. We decide that a mention refers to a person if its most probable candidate by $\hat{p}(e|m)$ is a person.

7 Experiments

7.1 ED Datasets

We validate our ED models on some of the most popular available datasets used by our predecessors[6]. We provide statistics in Table 2.

- AIDA-CoNLL dataset (Hoffart et al., 2011) is one of the biggest manually annotated ED datasets. It contains training (AIDA-train), validation (AIDA-A) and test (AIDA-B) sets.
- MSNBC (MSB), AQUAINT (AQ) and ACE2004 (ACE) datasets cleaned and updated by (Guo and Barbosa, 2016)[7]
- WNED-WIKI (WW) and WNED-CWEB (CWEB): are larger, but automatically extracted, thus less reliable. Are built from the ClueWeb and Wikipedia corpora by (Guo and Barbosa, 2016; Gabrilovich et al., 2013).

7.2 Training Details and (Hyper)Parameters

We explain training details of our approach. All models are implemented in the Torch framework.

Entity Vectors Training & Relatedness Evaluation. For entity embeddings only, we use Wikipedia (Feb 2014) corpus for training. Entity vectors are initialized randomly from a 0-mean normal distribution with standard deviation 1. We first train each entity vector on the entity's Wikipedia canonical description page (title words included) for 400 iterations. Subsequently, Wikipedia hyperlinks of the respective entities are used for learning until validation score (described below) stops improving. In each iteration, 20 positive words, each with 5 negative words, are sampled and used for optimization as explained in Section 3. We use Adagrad (Duchi et al., 2011) with a learning rate of 0.3. We choose embedding size $d = 300$, pre-trained (fixed) Word2Vec word vectors[8], $\alpha = 0.6$, $\gamma = 0.1$ and window size of 20 for the hyperlinks. We remove stop words before training. Since our method allows to train the embedding of each entity independently of other entities, we decide for efficiency reasons (and without loss of generality) to learn only the vectors of all entities appearing as mention candidates in all the test datasets described in Sec. 7.1, a total of 270000 entities. Training of those takes 20 hours on a single TitanX GPU with 12GB of memory.

Global methods	MSB	AQ	ACE	CWEB	WW
prior $\hat{p}(e\|m)$	89.3	83.2	84.4	69.8	64.2
(Fang et al., 2016)	81.2	88.8	85.3	-	-
(Ganea et al., 2016)	91	89.2	**88.7**	-	-
(Milne and Witten, 2008)	78	85	81	64.1	81.7
(Hoffart et al., 2011)	79	56	80	58.6	63
(Ratinov et al., 2011)	75	83	82	56.2	67.2
(Cheng and Roth, 2013)	90	**90**	86	67.5	73.4
(Guo and Barbosa, 2016)	92	87	88	77	**84.5**
our (global)	**93.7** ± **0.1**	88.5 ± 0.4	**88.5** ± **0.3**	**77.9** ± **0.1**	77.5 ± 0.1

Table 4: Micro F1 results for other datasets.

We test and validate our entity embeddings on the entity relatedness dataset of (Ceccarelli et al., 2013). It contains 3319 and 3673 queries for the test and validation sets. Each query consist of one target entity and up to 100 candidate entities with gold standard binary labels indicating if the two entities are related. The associated task requires ranking of related candidate entities higher than the others. Following previous work, we use different evaluation metrics: normalized discounted cumulative gain (NDCG) and mean average precision (MAP). The validation score used during learning is then the sum of the four metrics showed in Table 1. We perform candidate ranking based on cosine similarity of entity pairs.

[6] TAC-KBP datasets used by (Yamada et al., 2016; Globerson et al., 2016; Sun et al., 2015) are no longer available.

[7] Available at: `bit.ly/2gnSBLg`

[8] By Word2Vec authors: `http://bit.ly/1R9Wsqr`

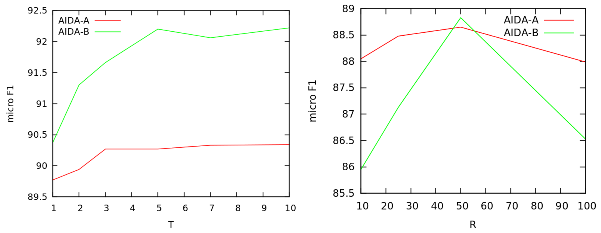


Table 5: Effects of two of the hyper-parameters. Left: A low T (e.g.5) is already sufficient for accurate approximate marginals. Right: Hard attention improves accuracy of a local model with K=100.

Local and Global Model Training. Our local and global ED models are trained on AIDA-train (multiple epochs), validated on AIDA-A and tested on AIDA-B and other datasets mentioned in Section 7.1. We use Adam (Kingma and Ba, 2014) with learning rate of 1e-4 until validation accuracy exceeds 90%, afterwards setting it to 1e-5. Variable size mini-batches consisting of all mentions in a document are used during training. We remove stop words. Hyper-parameters of the best validated global model are: $\gamma = 0.01, K = 100, R = 25, S = 7, \delta = 0.5, T = 10$. For the local model, $R = 50$ was best. Validation accuracy is computed after each 5 epochs. To regularize, we use early stopping, i.e. we stop learning if the validation accuracy does not increase after 500 epochs. Training on a single GPU takes, on average, 2ms per mention, or 16 hours for 1250 epochs over AIDA-train.

By using diagonal matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$, we keep the number of parameters very low (approx. 1.2K parameters). This is necessary to avoid overfitting when learning from a very small training set. We also experimented with diagonal plus low-rank matrices, but encountered quality degradation.

7.3 Entity Similarity Results

Results for the entity similarity task are shown in Table 1. Our method outperforms the well established Wikipedia link measure and the method of (Yamada et al., 2016) using less information (only word - entity statistics). We note that the best result on this dataset was reported in the unpublished work of (Huang et al., 2015). Their entity embeddings are trained on many more sources of information (e.g. KG links, relations, entity types). However, our focus was to prove that lightweight trained embeddings useful for the ED task can also perform decently for the entity similarity task. We emphasize that our global ED model outperforms Huang's ED model (Table 3), likely due to the power of our local and joint neural network architectures. For example, our attention mechanism clearly benefits from explicitly embedding words and entities in the same space.

Freq gold entity	Number mentions	Solved correctly	$\hat{p}(e\|m)$ **gold entity**	Number mentions	Solved correctly
0	5	80.0 %	≤ 0.01	36	89.19%
1-10	0	-	0.01 - 0.03	249	88.76%
11-20	4	100.0%	0.03 - 0.1	306	82.03%
21-50	50	90.0%	0.1 - 0.3	381	86.61%
> 50	4345	94.2%	> 0.3	3431	96.53%

Table 6: ED accuracy on AIDA-B for our best system splitted by Wikipedia hyperlink frequency and mention prior of the ground truth entity, in cases where the gold entity appears in the candidate set.

7.4 ED Baselines & Results

We compare with systems that report state-of-the-art results on the datasets. Some baseline scores from Table 4 are taken from (Guo and Barbosa, 2016). The best results for the AIDA datasets are reported by (Yamada et al., 2016) and (Globerson et al., 2016). We do not compare against (Pershina et al., 2015) since, as noted also by (Globerson et al., 2016), their mention index artificially includes the gold entity (guaranteed gold recall), which is not a realistic setting.

For a fair comparison with prior work, we use in-KB accuracy and micro F1 (averaged per mention) metrics to evaluate our approach. Results are shown in Tables 3 and 4. We run our system 5 times, each time we pick the best model on the validation set, and report results on the test set for these models. We obtain state of the art accuracy on AIDA which is the largest and hardest (by the accuracy of the $\hat{p}(e|m)$ baseline) manually created ED dataset . We are also competitive on the other datasets. It should be noted that all the other methods use, at least partially, engineered features. The merit of our proposed method is to show that, with the exception of the $\hat{p}(e|m)$ feature, a neural network is able to learn the best features for ED without requiring expert input.

To gain further insight, we analyzed the accuracy on the AIDA-B dataset for situations where gold entities have low frequency or mention prior. Table 6 shows that our method performs well in these harder cases.

Mention	Gold entity	$\hat{p}(e\|m)$ **of gold entity**	**Attended contextual words**
Scotland	Scotland national rugby union team	0.034	England Rugby team squad Murrayfield Twickenham national play Cup Saturday World game George following Italy week Friday selection dropped row month
Wolverhampton	Wolverhampton Wanderers F.C.	0.103	matches League Oxford Hull league Charlton Oldham Cambridge Sunderland Blackburn Sheffield Southampton Huddersfield Leeds Middlesbrough Reading Coventry Darlington Bradford Birmingham Enfield Barnsley
Montreal	Montreal Canadiens	0.021	League team Hockey Toronto Ottawa games Anaheim Edmonton Rangers Philadelphia Caps Buffalo Pittsburgh Chicago Louis National home Friday York Dallas Washington Ice
Santander	Santander Group	0.192	Carlos Telmex Mexico Mexican group firm market week Ponce debt shares buying Televisa earlier pesos share stepped Friday analysts ended
World Cup	FIS Alpine Ski World Cup	0.063	Alpine ski national slalom World Skiing Whistler downhill Cup events race consecutive weekend Mountain Canadian racing

Table 7: Examples of context words selected by our local attention mechanism. Distinct words are sorted decreasingly by attention weights and only words with non-zero weights are shown.

7.5 Hyperparameter Studies

In Table 5, we analyze the effect of two hyperparameters. First, we see that hard attention (i.e. $R < K$) helps reducing the noise from uninformative context words (as opposed to keeping all words when $R = K$).

Second, we see that a small number of LBP iterations (hard-coded in our network) is enough to obtain good accuracy. This speeds up training and testing compared to traditional methods that run LBP until convergence. An explanation is that a truncated version of LBP can perform well enough if used at both training and test time.

7.6 Qualitative Analysis of Local Model

In Table 7 we show some examples of context words attended by our local model for correctly solved hard cases (where the mention prior of the correct entity is low). One can notice that words relevant for at least one entity candidate are chosen by our model in most of the cases.

7.7 Error Analysis

We analyse some of the errors made by our model on the AIDA-B dataset. We mostly observe three situations: i) annotation errors, ii) gold entities that do not appear in mentions' candidate sets, or iii) gold entities with very low $p(e|m)$ prior whose mentions have an incorrect entity candidate with high prior. For example, the mention ”Italians” refers in some specific context to the entity ”Italy national football team” rather than the entity representing the country. The contextual information is not strong enough in this case to avoid an incorrect prediction. On the other hand, there are situations where the context can be misleading, e.g. a document heavily discussing about cricket will favor resolving the mention ”Australia” to the entity ”Australia national cricket team” instead of the gold entity ”Australia” (naming a location of cricket games in the given context).

8 Conclusion

We have proposed a novel deep learning architecture for entity disambiguation that combines entity embeddings, a contextual attention mechanism, an adaptive local score combination, as well as unrolled differentiable message passing for global inference. Compared to many other methods, we do not rely on hand-engineered features, nor on an extensive corpus for entity co-occurrences or relatedness. Our system is fully differentiable, although we chose to pre-train word and entity embeddings. Extensive experiments show the competitiveness of our approach across a wide range of corpora. In the future, we would like to extend this system to perform nil detection, coreference resolution and mention detection.

Our code and data are publicly available: `http://github.com/dalab/deep-ed`

Acknowledgments

We thank Aurelien Lucchi, Marina Ganea, Jason Lee, Florian Schmidt and Hadi Daneshmand for their comments and suggestions.

This research was supported by the Swiss National Science Foundation (SNSF) grant number 407540_167176 under the project ”Conversational Agent for Interactive Access to Information”.

References

Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. 2013. Learning relatedness measures for entity linking. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 139–148. ACM.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. *Urbana*, 51(61801):16–58.

Andrew Chisholm and Ben Hachey. 2015. Entity disambiguation with web links. *Transactions of the Association for Computational Linguistics*, 3:145–156.

Emily Denton, Jason Weston, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. 2015. User conditional hashtag prediction for images. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1731–1740. ACM.

Justin Domke. 2011. Parameter learning with truncated message-passing. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2937–2943. IEEE.

Justin Domke. 2013. Learning graphical model parameters with approximate marginal inference. *IEEE transactions on pattern analysis and machine intelligence*, 35(10):2454–2467.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. 2016. Entity disambiguation by knowledge and text jointly embedding. *CoNLL 2016*, page 260.

Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. *arXiv preprint arXiv:1604.00734*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0). *Note: http://lemurproject.org/clueweb09/FACC1/Cited by*, 5.

Octavian-Eugen Ganea, Marina Ganea, Aurelien Lucchi, Carsten Eickhoff, and Thomas Hofmann. 2016. Probabilistic bag-of-hyperlinks model for entity linking. In *Proceedings of the 25th International Conference on World Wide Web*, pages 927–938. International World Wide Web Conferences Steering Committee.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *ACL (1)*.

Zhaochen Guo and Denilson Barbosa. 2016. Robust named entity disambiguation with random walks.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. In *ACL (2)*, pages 30–34.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics.

Hongzhao Huang, Larry Heck, and Heng Ji. 2015. Leveraging deep neural networks and knowledge graphs for entity disambiguation. *arXiv preprint arXiv:1504.07678*.

Heng Ji. 2016. Entity discovery and linking reading list.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

John Lafferty, Andrew McCallum, Fernando Pereira, et al. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

Nevena Lazic, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2015. Plato: A selective context model for entity resolution. *Transactions of the Association for Computational Linguistics*, 3:503–515.

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

David Milne and Ian H Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM.

Kevin P Murphy, Yair Weiss, and Michael I Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the*

Fifteenth conference on Uncertainty in artificial intelligence, pages 467–475. Morgan Kaufmann Publishers Inc.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics.

Valentin I Spitkovsky and Angel X Chang. 2012. A cross-lingual dictionary for english wikipedia concepts.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *IJCAI*, pages 1333–1339.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. *CoNLL 2016*, page 250.

Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. 2016. Robust and collective entity disambiguation through semantic embeddings. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 425–434. ACM.

MinIE: Minimizing Facts in Open Information Extraction

Kiril Gashteovski[1] and **Rainer Gemulla**[1] and **Luciano del Corro**[2]
[1]Universität Mannheim, Mannheim, Germany
[1]{k.gashteovski,rgemulla}@uni-mannheim.de
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[2]corrogg@mpi-inf.mpg.de

Abstract

The goal of Open Information Extraction (OIE) is to extract surface relations and their arguments from natural-language text in an unsupervised, domain-independent manner. In this paper, we propose MinIE, an OIE system that aims to provide useful, compact extractions with high precision and recall. MinIE approaches these goals by (1) representing information about polarity, modality, attribution, and quantities with semantic annotations instead of in the actual extraction, and (2) identifying and removing parts that are considered overly specific. We conducted an experimental study with several real-world datasets and found that MinIE achieves competitive or higher precision and recall than most prior systems, while at the same time producing shorter, semantically enriched extractions.

1 Introduction

Open Information Extraction (OIE) (Banko et al., 2007) is the task of generating a structured, machine-readable representation of information expressed in natural language text in an unsupervised, domain-independent manner. In contrast to traditional IE systems, OIE systems do not require an upfront specification of the target schema (e.g., target relations) or access to background knowledge (e.g., a knowledge base). Instead, extractions are (usually) represented in the form of surface subject-relation-object triples. OIE serves as input for deeper understanding tasks such as relation extraction (Riedel et al., 2013; Petroni et al., 2015), knowledge base construction (Dong et al., 2014), question answering (Fader et al., 2014), word analogy (Stanovsky et al., 2015), or information retrieval (Löser et al., 2012).

Consider, for example, the sentence *"Superman was born on Krypton."* An OIE system aims to extract the triple *(Superman, was born on, Krypton)*, which most of the available systems will correctly produce. As another example, consider the more involved sentence *"Pinocchio believes that the hero Superman was not actually born on beautiful Krypton"*, and the corresponding extractions of various systems in Table 1, extractions 1–6. Although most of the extractions are correct, they are often overly specific in that their constituents contain specific modifiers or even complete clauses. Such extractions severely limit the usefulness of OIE results (e.g., they are often pruned in relation extraction tasks). The main goals of OIE should be (i) to provide useful, compact extractions and (ii) to produce extractions with high precision and recall. The key challenge in OIE is how to achieve both goals simultaneously. In fact, most of the available systems (often implicitly) focus on either compactness (e.g., ReVerb (Fader et al., 2011)) or precision/recall (e.g., ClausIE (Del Corro and Gemulla, 2013)).

We propose MinIE, an OIE system that aims to address and trade-off both goals. MinIE is built on top of ClausIE, a state-of-the-art OIE system that achieves high precision and recall, but often produces overly-specific extractions. To generate more useful and semantically richer extractions, MinIE (i) provides semantic annotations for each extraction, (ii) minimizes overly-specific constituents, and (iii) produces additional extractions that capture implicit relations. Table 1 shows the output of (variants of) MinIE for the example sentence. Note that MinIE's extractions are significantly more compact but retain correctness.

MinIE's semantic annotations represent information about polarity, modality, attribution, and quantities. The idea of using annotations has al-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2630–2640
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Pinocchio believes that the hero Superman was not actually born on beautiful Krypton.				
OLLIE	*1*	(Pinocchio,	believes that,	the hero [...] beautiful Krypton)
	2	(Superman,	was not actually born on,	beautiful Krypton)
	3	(Superman,	was not actually born on beau. Krypton in,	the hero)
ClausIE	*4*	(Pinocchio,	believes,	that the hero [...] beautiful K.)
	5	(the hero Superman,	was not born,	on beautiful Krypton)
	6	(the hero Superman,	was not born,	on beautiful Krypton actually)
Stanford OIE		*No extractions*		
MinIE-C(om-	*7*	(Superman,	was born actually on,	beautiful Krypton)
plete)		*A.: fact. (– [not], CT), attrib. (Pinocchio, +, PS [believes])*		
	8	(Superman,	was born on,	beautiful Krypton)
		A.: fact. (– [not], CT), attrib. (Pinocchio, +, PS [believes])		
	9	(Superman,	”is”,	hero)
		A.: fact. (+, CT)		
MinIE-S(afe)	*10*	(Superman,	was born on,	beautiful Krypton)
		A.: fact. (– [not], CT), attrib. (Pinocchio, +, PS [believes]), relation (was actually born on)		
	11	(Superman,	”is”,	hero)
		A.: fact. (+, CT)		
MinIE-D(ic-	*12*	(Superman,	was born on,	Krypton)
tionary)		*A.: fact. (– [not], CT), attrib. (Pinocchio, +, PS [bel.]), rel. (was act. born on), argument (beau. K.)*		
MinIE-A(gg-	*13*	(Superman,	”is”,	hero)
ressive)		*A.: fact. (+, CT)*		

A annotation; + positive polarity, – negative polarity; *PS* possibility, *CT* certainty; *fact.* factuality; *attrib.* attribution;

Table 1: Example extractions and annotations from various OIE systems

ready been explored by OLLIE (Mausam et al., 2012) for capturing the context of an extraction. MinIE follows OLLIE, but adds semantic annotations that make the extraction *itself* more compact and useful (as opposed to capturing context). For example, MinIE detects negations in the relation, removes them from the extraction, and adds a "negative polarity" (-) annotation. In fact, MinIE treats surface relations such as *was born on* and *was not born on* as equivalent up to polarity. The absence of negative evidence is a major concern for relation extraction and knowledge base construction tasks—e.g., addressed by using a local closed world assumption (Dong et al., 2014) or negative sampling (Riedel et al., 2013; Petroni et al., 2015)—and MinIE's annotations can help to alleviate this problem.

In addition to the semantic annotations, MinIE minimizes its extractions by identifying and removing parts that are considered overly specific. In general, such minimization is inherently limited in scope due to the absence of domain knowledge. Thus MinIE does not and cannot correctly minimize all its extractions in all cases. Instead, MinIE supports multiple minimization modes, which differ in their aggressiveness and effectively control the usefulness-precision trade-off. In particular, MinIE's complete mode (C) does not perform any minimizations. MinIE's safe mode (S) only performs minimizations that are considered universally safe. MinIE's dictionary mode (D) makes use of corpus-level statistics to inform the minimization process. Finally, MinIE's aggressive mode (A) only keeps parts that are considered universally necessary. The use of corpus-level statistics by MinIE-D is inspired by the pruning techniques of ReVerb, although we use these statistics for minimization instead of pruning (see Sec. 2). Tab. 1 shows the output of MinIE's various modes.

We conducted an experimental study with several real-world datasets and found that the various modes of MinIE produced much shorter extractions than most prior systems, while simultaneously achieving competitive or higher precision (depending on the mode being used). MinIE sometimes fell behind prior systems in terms of the total number of extractions. We found that in almost all of these cases, MinIE became competitive once redundant extractions were removed.

2 Related work

OIE was introduced by Banko et al. (2007). Since then, many different OIE systems have been proposed. Earlier systems—e.g., Fader et al.

(2011)—relied mostly on shallower NLP techniques such as POS tagging and chunking, while later systems often use dependency parsing in addition (Gamallo et al., 2012; Wu and Weld, 2010). Most OIE systems represent extractions in the form of triples, although some also produce n-ary extractions (Akbik and Löser, 2012; Del Corro and Gemulla, 2013) or nested representations (Bast and Haussmann, 2013; Bhutani et al., 2016). Some systems focus on non-verb-mediated relations (Yahya et al., 2014). MinIE is based on the state-of-the-art OIE system ClausIE (Del Corro and Gemulla, 2013).

A general challenge in OIE is to avoid both uninformative and overly-specific extractions. ReVerb (Fader et al., 2011) proposed to avoid overly-specific relations by making use of *lexical constraints*: relations that occur infrequently in a large corpus were considered overly-specific and pruned. MinIE's dictionary mode also makes use of the corpus frequency of constituents. In contrast to ReVerb, MinIE uses frequency to inform minimization (instead of to prune) and applies it to subjects and arguments as well. Perhaps the closest system in spirit to MinIE is Stanford OIE (Angeli et al., 2015), which uses aggressive minimization. Stanford OIE deletes all subconstituents connected by certain typed dependencies (e.g., *amod*). For some dependencies (e.g., *prep* or *dobj*), it uses a frequency constraint along the lines of ReVerb. MinIE differs from Stanford OIE in that it (i) separates out polarity, modality, attribution, and quantities; (ii) uses a different, more principled (and more precise) approach to minimization.

Annotated OIE extractions were introduced by OLLIE (Mausam et al., 2012), which uses two types of annotations: *attribution* (the supplier of information) and *clause modifier* (a clause modifying the triple). MinIE extends OLLIE's attribution by additional semantic annotations for polarity, modality, and quantities. Such annotations are not provided by prior OIE systems. CSD-IE (Bast and Haussmann, 2013) introduced the notion of nested facts (termed "minimal" in their paper) and produce extractions with "pointers" to other extractions. NestIE (Bhutani et al., 2016) takes up this idea. OLLIE's clause modifier has a similar purpose. MinIE currently does not handle nested extractions.

Another line of research explores the integration of background knowledge into OIE (Nakashole et al., 2012; Moro and Navigli, 2012, 2013). In general, OIE systems should use background knowledge when available, but remain open when not. MinIE currently does not use background knowledge, although it allows providing domain-dependent dictionaries.

3 Overview

The goal of MinIE is to provide minimized, semantically annotated OIE extractions. While the techniques employed here can potentially be integrated into any OIE system, we built MinIE on top of ClausIE. We chose ClausIE because (i) it separates the identification of the extractions from the generation of propositions, (ii) it detects clause types, which are also useful for MinIE, and (iii) it is a state-of-the-art OIE system with high precision and recall.

As ClausIE, MinIE focuses on extractions obtained from individual clauses (with the exception of attribution; Sec. 5.3). Each clause consists of one subject (S), one verb (V) and alternatively an indirect object (O_i), a direct object (O), a complement (C) and one or more adverbials (A). ClausIE identifies the clause type, which indicates which constituents are obligatory or optional from a syntactic point of view. Quirk et al. (1985) identified seven clause types for English: SV, SVA, SVC, SVO, SVOO, SVOA, and SVOC, where letters refer to obligatory constituents and each clause can be accompanied by additional optional adverbials.

MinIE consists of three phases. (1) Each input sentence is run through ClausIE and a separate extractor for implicit facts (Sec. 4.2). We rewrite ClausIE's extractions to make relations more informative (Sec. 4.1). We refer to the resulting extractions as *input extractions*. (2) MinIE then detects information about polarity (Sec. 5.1), modality (Sec. 5.2), attribution (Sec. 5.3), and quantities (Sec. 5.4) and represents it with semantic annotations. (3) To further minimize the resulting *annotated extractions*, MinIE provides various minimization modes (Sec. 6) with increasing levels of aggressiveness: MinIE-C(omplete), MinIE-S(afe), MinIE-D(ictionary), and MinIE-A(ggressive). The modes differ in the amount of minimizations being applied. The result of this phase is a *minimized extraction*.

Finally, MinIE outputs each minimized extraction along with its annotations. Semantic annotations (such as polarity) are crucial to correctly represent the extraction, whereas other annotations (such as original relation) provide additional information about the minimization process.

4 Input Extractions

We first describe how MinIE obtains meaningful input extractions.

4.1 Enriching Relations

As mentioned before, MinIE uses ClausIE as its underlying OIE system. The relations extracted by ClausIE consist of only verbs and negation particles (cf. Tab. 1). Fader et al. (2011) argue that such an approach can lead to uninformative relations. For example, from the sentence *"Faust made a deal with the Devil"*, ClausIE extracts triple *(Faust, made, a deal with the Devil)*, whereas the extraction *(Faust, made a deal with, the Devil)* has a more informative relation and a shorter argument. Indeed, the relation *made* is highly polysemous (49 synsets in WordNet), whereas *made a deal with* is not. MinIE aims to produce informative relations by deciding which constituents of the input sentence should be pushed into the relation. Our goal is to retain only one of the constituents of the input clause in the argument of the extraction whenever possible, while simultaneously retaining coherence. In particular, our approach uses the clause types detected by ClausIE to ensure that MinIE never removes obligatory constituents from a clause (which would lead to incoherent extractions); it instead may opt to move such constituents to the relation. Our approach is inspired by the syntactic patterns of ReVerb—which is similar to our handling of the SVA and SVO clause types—but, in contrast, applies to all clause types. Note that the relations produced in this step may sometimes be considered overly specific; they will be minimized further in subsequent steps.

SVA. If the adverbial is a prepositional complement, we push the preposition into the relation. For example, we rewrite *(Superman, lives, in Metropolis)* to *(Superman, lives in, Metropolis)*. This allows us to distinguish *live in* from relations such as *live during*, *live until*, *live through*, and so on.

SVO$_i$O, SVOC. We generally push the indirect object (SVO$_i$O) or direct object (SVOC) into the relation. In both cases, the verb requires two additional constituents: we use the first one to enrich the relation and the second one as an argument. For example, we rewrite *(Superman, declared, the city safe)* to *(Superman, declared the city, safe)*. As this example indicates, this rewrite is somewhat unsatisfying; further exploration is an interesting direction for future work.

SVOA. If the adverbial consists of a single adverb, we push it to the relation and use the object as an argument. This approach retains coherence because such adverbials are "fluent", i.e., they do not have a fixed position. Otherwise, we proceed as in SVOC, but additionally push the starting preposition (if present) of the adverbial to the relation. For example, *(Ana, turned, the light off)* becomes *(Ana, turned off, the light)*, and *(The doorman, leads, visitors to their destination)* becomes *(The doorman, leads visitors to, their destination)*.

Optional adverbials. If the clause contains optional adverbials, ClausIE creates one extraction without any optional adverbial and one additional extraction per optional adverbial. The former extractions are processed as above. The latter extractions are treated as if the adverbial were obligatory. For example, the extraction *(Faust, made, a deal with the Devil)* becomes *(Faust, made a deal with, the Devil)*. Here the actual clause type is SVO, but we process it as if it were SVOA.

Infinitive forms. If the argument starts with a to-infinitive verb, we move it to the relation. For example, *(Superman, needs, to defeat Lex)* becomes *(Superman, needs to defeat, Lex)*.

4.2 Implicit Extractions

ClausIE produces non-verb-mediated extractions from appositions and possessives. We refer to these extractions as *implicit extractions*. MinIE makes use of additional implicit extractors. In particular, we use the patterns of FINET (Del Corro et al., 2015) to detect explicit type mentions. For example, if the sentence contains "*president Barack Obama*", we obtain *(Barack Obama, is, president)*. We also include certain patterns involving named entities: pattern *ORG IN LOC* for extraction *(ORG, is IN, LOC)*; pattern *"Mr." PER* for *(PER, is, male)* (similarly, Ms. or Mrs.); and pattern *ORG POS? NP PER* for *(PER, is NP of, ORG)* from RelNoun (Pal and Mausam, 2016). Apart from providing additional high-quality extractions, we use implicit extractions as a signal for minimization (Sec. 6.2). The extractors above have thus been included both to increase recall and to be able to provide more effective minimizations.

Sentence	Factuality
S. does live in Metropolis.	(+, CT)
S. does not live in M.	(– [not], CT)
S. does probably live in M.	(+, PS [probably])
S. probably does not live in M.	(– [not], PS [probably])

Table 2: Factuality examples. MinIE extracts triple *(Superman; does live in; Metropolis)* from each sentence but the factuality annotations differ.

5 Semantic Annotations

Once input extractions have been created, MinIE detects information about polarity (Sec. 5.1), modality (Sec. 5.2), attribution (Sec. 5.3), and quantities (Sec. 5.4) and represents it using semantic annotations. Our focus is on simple, rule-based methods that are both domain-independent and (considered) safe to use in that they do not harm the accuracy of the extraction.

5.1 Polarity

MinIE annotates each extraction with information about its *factuality*. Following Saurí and Pustejovsky (2012), we represent the factuality of an extraction with two pieces of information: polarity (+ or -) and modality (CT or PS; for certainty or possibility, resp.). Tab. 2 lists some examples.

The *polarity* indicates whether or not a triple occurred in negated form. In order to assign a polarity value to a triple, we aim to detect whether the relation indicates a negative polarity. If so, we assign negative polarity to the whole triple. We detect negations using a small lexicon of negation words (e.g., *no, not, never, none*). If a word from the lexicon is detected, it is dropped from the relation and the triple is annotated with negative polarity (-) and the negation word. In Tab. 2, the extractions from sentences 2 and 4 are annotated as negative.

We found that this simple approach successfully spots many negations present in the input relations. Note that whenever a negation is present but not detected, MinIE still produces correct results because such negations are retained in the triple. For example, if a negations occurs in the subject or argument of the extraction, MinIE does not detect it. E.g., from sentence *"No people were hurt in the fire"*, MinIE extracts *(Q_1 people, were hurt in; fire)* with quantity Q_1=*no* (see Sec. 5.4). This extraction is correct, but can be further minimized to *(people; were hurt in; fire)* with a negative polarity annotation. We consider such advanced minimizations too dangerous to use.

Generally, negation detection is a hard problem and involves questions such as negation scope resolution, focus detection, and double negation (Blanco and Moldovan, 2011). MinIE does not address these problems, but restricts attention to the simple, safe cases.

5.2 Modality

The *modality* indicates whether the triple is a *certainty* (CT) or a *possibility* (PS) according to the clause in which it occurs. We proceed similarly as for the detection of negations and consider a triple certain unless we find evidence of possibility.

To find such evidence, MinIE searches the relation for (1) modal verbs such as *may* or *can*, (2) possibility-indicating words, and (3) certain infinitive verb phrases. For (2) and (3), we make use of a small domain-independent lexicon. Our lexicon is based on the lexicon of Saurí and Pustejovsky (2012) and the words in the corresponding WordNet synsets. It mainly contains adverbs such as *probably, possibly, maybe, likely* and infinitive verb phrases such as *is going to*, *is planning to*, or *intends to*. Whenever words indicating possibility are detected, we remove these words from the triple and annotate the triple as possible (PS) along with the words just removed. For example, sentences 3 and 4 in Tab. 2 are annotated PS with the possibility-indicating word *probably*.

5.3 Attribution

The *attribution* of a triple is the supplier of information given in the input sentence, if any. We adapt our attribution annotation from the notion of *source* of Saurí and Pustejovsky (2012), i.e., the attribution consists of a supplier of information (as in OLLIE) and an additional factuality (polarity and modality). The factuality is independent from the factuality of the extracted triple; it indicates whether the supplier expresses a negation or a possibility. Tab. 1 shows some examples.

We extract attributions from subordinate clauses and from "according to" patterns.

Subordinate clauses. MinIE searches for extractions that contain entire clauses as arguments. We then compare the relation against a domain-independent dictionary of relations indicating attributions (e.g., *say* or *believe*).[1] If we find a

[1] As with modality, the dictionary is based on Saurí and

match, we create an attribution annotation and use the subject of the extraction as the supplier of information. Each entry in the attribution dictionary is annotated with a modality. For example, relations such as *know*, *say*, or *write* express certainty, whereas relations such as *believe* or *guess* express possibility. If the relation is modified by a negation word, we mark the attribution with negative polarity (e.g., *never said that*). After the attribution has been established, we run ClausIE on the main clause and add the attribution to each extracted triple.

***"according to"* adverbial patterns.** We search for adverbials that start with *according to* and take whatever follows as the supplier with factuality (+, CT). The remaining part of the clause is processed as before.

5.4 Quantities

A *quantity* is a phrase that expresses an amount (or the absence) of something. It either modifies a noun phrase (e.g., *9 cats*) or is an independent complement (e.g., *I have 3*). Quantities include cardinals (*9*), determiners (*all*) or phrases (*almost 10*). If we detect a quantity, we replace it by a placeholder *Q* and add an annotation with the original quantity. The goal of this step is to unify extractions that only differ in quantities. For example, the phrases *9 cats*, *all cats* and *almost about 100 cats* are all rewritten to *Q cats*, only the quantity annotation differs.

We detect quantities by looking for numbers (NER types such as NUMBER or PERCENT) or words expressing quantities (such as *all, some, many*). We then extend these words via relevant typed dependencies, such as quantity modifiers (*quantmod*) and adverbial modifiers (*advmod*).

6 Minimization

After adding semantic annotations, MinIE minimizes extractions by dropping additional words. Since such minimization is risky, MinIE employs various minimization modes with different levels of aggressiveness, which effectively control the minimality-precision trade-off.

MinIE represents each constituent of an annotated extraction by its words, its dependency structure, its POS tags, and its named entities (detected by a named-entity recognizer). In general, each mode defines a set of *stable subconstituents*, which will always be fully retained, and subsequently searches for candidate words to drop outside of the stable subconstituents. Whenever a word is dropped from a constituent, we add an annotation with the original, unmodified constituent.

In all of MinIE's modes, noun sequences (which include the head) and named entities (from NER) are considered stable subconstituents. MinIE's minimization can be augmented with domain knowledge by providing information about additional stable subconstituents (e.g., collocations).

6.1 Complete Mode (MinIE-C)

MinIE's *complete mode* (MinIE-C) prunes all the extractions that contain subordinate clauses but does not otherwise modify the annotated extractions. The rationale is that extractions containing subordinate clauses are almost always overly specific. MinIE-C serves as a baseline.

6.2 Safe Mode (MinIE-S)

MinIE's *safe mode* only drops words which we consider universally safe to drop. We first drop all constituents that are covered by the implicit extractions discussed in Sec. 4.2 (e.g., *"Mr."* before persons). We then drop all determiners, possessive pronouns, adverbs modifying the verb in the relation, as well as adjectives and adverbs modifying words tagged as PERSON by the NER. An exception to these rules is given by named entities, which we consider as stable subconstituents (e.g., we do not drop *"Mr."* in (*Joe*, *cleans with*, *Mr. Muscle*)).

Note that this procedure cannot be considered safe when used on input extractions. We consider it safe, however, when applied to annotated extractions. In particular, all determiners, pronouns, and adverbs indicating negation, modality, or quantities are already processed and captured in annotations. The safe mode thus only performs simple rewrites such as *the great city* to *great city*, *his car* to *car*, *had also* to *had*, and *the eloquent president Mr. Barack Obama* to *Barack Obama*.

6.3 Dictionary Mode (MinIE-D)

Our dictionary mode uses a *dictionary $\mathcal{D}$ of stable constituents*. We first discuss how the dictionary is being used and subsequently how we construct it. An example is given in Fig. 1.

MinIE-D first performs all the minimizations of the safe mode and then searches for maximal noun phrases of the form $P \equiv$ *[adverbial|adjective]*$^+$

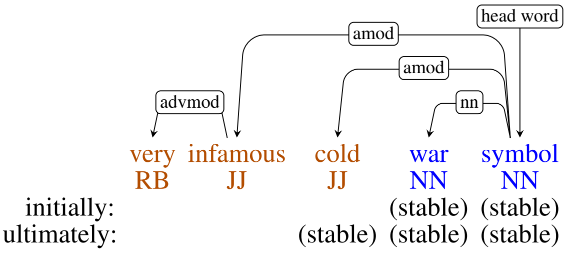
Pustejovsky (2012) plus WordNet synonyms.

[noun$^+$*|ner]*. For each instance of P, we drop a certain subset of its words. For example, a suitable minimization for *very infamous cold war symbol* (i.e., the Berlin wall) is *cold war symbol*, i.e., we consider *cold* as essential to the meaning of the constituent and *very infamous* as overly specific. The decision of what is considered essential and what overly specific is informed by dictionary $\mathcal{D}$. Note that in order to minimize mistakes, we consider for dropping only words in instances of pattern P. In particular, we do not touch subconstituents that contain prepositions because these are notoriously difficult to handle (e.g., we do not want to minimize *Bill of Rights* to *Bill*).

Our goal is to retain phrases occurring in $\mathcal{D}$, even if they occur in different order or with additional modifiers. We proceed as follows for each instance I of P. We first mark all nouns modifying the root (or the named entity) as *stable*. Afterwards, we create a set of *potentially stable subconstituents* (PSS). Each PSS is queried against dictionary $\mathcal{D}$. If it occurs in $\mathcal{D}$, all of its words are marked as stable. Once all PSS have been processed, we drop all words from I that are not marked stable. In our example, if $\{$*cold war*$\} \in \mathcal{D}$, we obtain *cold war symbol*.

To generate the set of PSS, we enumerate all syntactically valid subconstituents of I. For example, *infamous symbol* or *cold infamous war* are syntactically valid, whereas *very symbol* or *very cold war* are not. Conceptually,[2] we enumerate all subsequences of I and check whether (1) at least one noun (or named entity) is retained, and (2) whenever an adverb or adjective is not retained, neither are its modifiers. For each such subsequence, we generate all permutations of adverbial and adjective modifiers originating from the same dependency node, and each result as a PSS. This step ensures that the order of modifiers in I does influence whether or not a word is marked stable. The set of PSS for *very infamous cold war symbol* contains 22 entries.

The construction of dictionary $\mathcal{D}$ is inspired by the lexical constraint of Fader et al. (2011): Our assumption is that everything sufficiently frequent in a large corpus is not overly specific. To obtain $\mathcal{D}$, we process the entire corpus using the safe mode and include all frequent (e.g., frequency $\geq$ 10) subjects, relations, and arguments into $\mathcal{D}$. Applications can extend the dictionary using suitable collocations, either from domain-dependent dictionaries or by using methods to automatically extract collocations from a corpus (Gries, 2013).



PSS include: cold war symbol, cold symbol, **cold war**, infamous war symbol, infamous symbol, . . .

Figure 1: Illustration of PSS generation in MinIE-D. Initially stable words are marked blue. Entries in dictionary $\mathcal{D}$ are printed in bold face.

6.4 Aggressive Mode (MinIE-A)

All previous modes aimed to be conservative. MinIE-A proceeds the other way around: all words for which we are not sure if they need to be retained are dropped. For every word in a constituent of an annotated extraction, we drop all adverbial, adjective, possessive, and temporal modifiers (along with their modifiers). We also drop prepositional attachments (e.g., *man with apples* becomes *man*), quantities modifying nouns, auxiliary modifiers to the main verb (e.g., *have escalated* becomes *escalated*), and all compound nouns that have a different named-entity type than their head word (e.g., *European Union official* becomes *official*). In most cases, after applying these steps, only a single word, named entity, or a sequence of nouns remains for subject and argument constituents.

7 Experimental Study

The goal of our experimental study was to investigate the differences in the various modes of MinIE w.r.t. precision, recall, and extraction length as well as to compare it with popular prior methods.

7.1 Experimental Setup

Source code, dictionaries, datasets, extractions, labels, and labeling guidelines are made available.[3]

Datasets. We used (1) 10,000 random sentences from the New York Times Corpus (NYT-

[2] We generate both instances of P as well as the set of PSS directly from the dependency structure of the constituent.

[3] http://dws.informatik.uni-mannheim.de/en/resources/software/minie/

10k) (Sandhaus, 2008), (2) a random sample of 200 sentences from the same corpus (NYT), and (3) a random sample of 200 sentences from Wikipedia (Wiki). NYT and Wiki were used in the evaluation of ClausIE and NestIE.[4]

Methods. We used ClausIE, OLLIE, and Stanford OIE as baseline systems. We adapted the publicly available version of ClausIE to Stanford CoreNLP 3.8.0 and implemented MinIE on top. For MinIE-D, we built dictionary $\mathcal{D}$ from the entire NYT and Wikipedia corpus, respectively.

Labeling. Labelers provided two labels per extraction of NYT and Wiki: one for the triple (without attribution) and one for the attribution. A triple is labeled as correct if it is entailed by its corresponding clause; here factuality annotations are taken into account but attribution errors are ignored. For example, all triples except #3 of Tab. 1 are considered correct. An attribution is incorrect if there is an attribution in the sentence which is neither present in the triple nor in the attribution annotation. In Tab. 1, the attribution is incorrect for extractions #2, #3, #5, and #6. Attribution is labeled only when the fact triple is labeled correct. See the labeling guidelines for further details.

Overall, there were more than 9,400 distinct extractions on NYT and Wiki. Each extraction was labeled by two independent labelers. We treat an extraction as *correct* if both labelers labeled it as *correct*. The inter-annotator agreement was moderate (NYT: Cohen's $\kappa = 0.53$, 78% of labels agree; Wiki: $\kappa = 0.5$, 79% of labels agree).

Measures. For each system, we measured the total number of extractions, the total number of correct triples (*recall*), the fraction of correct triples out of all extractions (*factual precision*), and the fraction of correct triples that have correct attributions (*attribution precision*). We also determined the mean word count per triple (μ) and its standard deviation (σ) as a proxy for minimality. Finally, as some systems produced a large number of redundant extractions, we also report the number of non-redundant extractions. For simplicity, we consider a triple t_1 redundant if it appears as subsequence in some other triple t_2 produced by the same extractor from the same sentence (e.g., extraction #5 in Tab. 1 is redundant given extraction #6).

7.2 Extraction Statistics

In our first experiment, we used the larger but unlabeled NYT-10k dataset. The goal of this experiment was to investigate the total number of redundant and non-redundant extractions produced by each system and how frequently semantic annotations were produced (Tab. 3). For MinIE, we show the fraction of negative polarity and possibility annotations for triples only (i.e., we exclude the attribution polarity annotations).

In terms of number of extractions, MinIE (all modes) and Stanford OIE were roughly on par; OLLIE fell behind and ClausIE went ahead. The reason why ClausIE has more extractions than MinIE is that different (partly redundant) extractions from ClausIE may lead to the same minimized extraction. This is also also the reason why extraction numbers drop in the more aggressive modes of MinIE. We also determined the number of non-redundant extractions produced by each system and found that most systems produced only a moderate number of redundant extractions. A notable exception is Stanford OIE, which produced many extraction variants by dropping different subsets of words.

We observed that all modes of MinIE achieved significantly smaller extractions than ClausIE (its underlying OIE system), and that the average extraction length indeed dropped as we used more aggressive modes. Only MinIE-A produced shorter extractions than Stanford OIE. The main reason for the short extraction length of Stanford OIE is its aggressive creation of short redundant extractions (at the cost of precision; see below). We also found that to further minimize the extractions of MinIE-D, it is often necessary to minimize subjects and objects with prepositional modifiers (which MinIE currently avoids).

Only OLLIE and MinIE make use of annotations. The fraction of extracted attribution annotations was significantly smaller for OLLIE than for MinIE, mainly because OLLIE's attribution detection is limited to the *ccomp* dependency relation. Our results also indicate that MinIE frequently provides semantic annotations (with the notable exception of negative polarity).

7.3 Precision

In our second experiment, we compared the precision and recall of the various systems on the

[4] We did not use the OIE benchmark of Stanovsky and Dagan (2016) because it treats an extraction as correct if the heads of each constituent match the ones of a gold extraction. This is not suitable for us because it does not account for minimization (which does not change grammatical heads).

	OLLIE	ClausIE	Stanford	MinIE-C	MinIE-S	MinIE-D	MinIE-A
# non-redundant extr.	20,557	36,173	16,350	**37,465**	37,093	36,921	36,474
# with redundant extr.	24,316	58,420	43,360	47,637	45,492	45,318	42,842
$\mu \pm \sigma$	9.9 ± 5.8	10.9 ± 7.0	6.6 ± 3.0	8.3 ± 4.9	7.2 ± 4.2	7.0 ± 4.1	$\mathbf{4.7 \pm 1.9}$
with attributions	6.8%	-	-	10.8%	10.8%	10.7%	10.8%
with negative polarity	-	-	-	3.8%	3.7%	3.7%	3.8%
with possibility	-	-	-	10.1%	9.9%	10.0%	9.7%
with quantity	-	-	-	17.6%	17.8%	17.8%	1.9%

Table 3: Results on the unlabeled NYT-10k dataset (μ=avg. extraction length, σ=standard deviation)

	OLLIE	ClausIE	Stanford	MinIE-C	MinIE-S	MinIE-D	MinIE-A
				NYT			
# non-redundant (correct/total)	246/414	505/821	178/342	**581/785**	574/781	569/777	439/753
# w/ redundant (correct/total)	302/497	**792/1300**	530/1052	727/970	690/924	681/916	505/860
factual prec.	(0.61)	(0.61)	(0.5)	**(0.75)**	**(0.75)**	(0.74)	(0.59)
attr. prec.	(0.9)	-	-	**(0.94)**	(0.93)	(0.93)	(0.93)
				Wiki			
# non-redundant (correct/total)	229/479	424/704	217/398	**500/666**	489/661	486/669	401/658
# w/ redundant (correct/total)	284/565	628/1002	651/1519	**635/851**	602/816	593/816	474/783
factual prec.	(0.50)	(0.63)	(0.43)	**(0.75)**	(0.74)	(0.73)	(0.61)
attr. prec.	**(0.97)**	-	-	**(0.97)**	(0.96)	(0.96)	**(0.97)**

Table 4: Results on the labeled NYT and Wiki datasets

smaller NYT and Wiki datasets. Our results are summarized in Tab. 4.

We found that Stanford OIE had the lowest factual precision and recall for non-redundant extractions throughout; it produced many incorrect and many redundant extractions (e.g., Stanford OIE produced 400 extractions from five sentences on NYT). For MinIE, the factual precision dropped as expected when we use more aggressive modes. Interestingly, the drop in precision between MinIE-C and MinIE-D was quite low, even though extractions get shorter. The aggressive minimization of MinIE-A led to a more severe drop in precision. Surprisingly to us, even MinIE's aggressive mode achieved precision comparable to ClausIE and higher than Stanford OIE. Note that MinIE-C, MinIE-S, and MinIE-D had higher precision than ClausIE. Reasons include that MinIE produces additional high-precision implicit extractions and breaks up very long and thus error-prone extractions.We also tried enriching the dictionary of MinIE-D with WordNet and Wiktionary collocations; the precision was almost the same.

As for attribution precision, most of the sentences in our samples did not contain attributions; these numbers thus have low accuracy. OLLIE and MinIE achieved similar results, even though MinIE additionally annotated attributions with factuality information.

Errors. For all modes, errors in dependency parsing transfer over to errors in MinIE, which we believe was the main source of error in MinIE-C and MinIE-S. For MinIE-D, we sometimes drop adjectives which in fact form collocations (e.g., "*assistant* director") with the noun they are modifying. This happens when the collocation is not present in the dictionary; better collocation dictionaries may address this problem. Another source of error stems from the NER (e.g., the first word of the entity *Personal Ensign* was not recognized).

8 Conclusions

We believe that the use of minimized extractions with semantic annotations are a promising direction for OIE. The techniques presented here can be seen as a step towards this goal, but there are still many open questions. Important directions include additional annotation types (e.g., temporal/spatial), use of background knowledge, better handling of collocations, the use of nested representations, and multilingual OIE.

Acknowledgments

We would like to thank Simone Paolo Ponzetto, Goran Glavaš, Stefano Faralli, Daniel Ruffinelli, and the anonymous reviewers for their invaluable feedback and support.

References

Alan Akbik and Alexander Löser. 2012. Kraken: N-ary facts in open information extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 52–56. Association for Computational Linguistics.

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *International Joint Conference on Artificial Intelligence*, volume 7, pages 2670–2676.

Hannah Bast and Elmar Haussmann. 2013. Open information extraction via contextual sentence decomposition. In *International Conference on Semantic Computing*, pages 154–159. IEEE.

Nikita Bhutani, H V Jagadish, and Dragomir Radev. 2016. Nested propositions in open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 55–64.

Eduardo Blanco and Dan I Moldovan. 2011. Some issues on detecting negation from text. In *Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference*, pages 228–233.

Luciano Del Corro, Abdalghani Abujabal, Rainer Gemulla, and Gerhard Weikum. 2015. Finet: Context-aware fine-grained named entity typing. In *Conference on Empirical Methods in Natural Language Processing*, pages 868–878. Association for Computational Linguistics.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web*, pages 355–366. Association for Computing Machinery.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. Association for Computing Machinery.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1156–1165. Association for Computing Machinery.

Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. 2012. Dependency-based open information extraction. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP*, pages 10–18. Association for Computational Linguistics.

Stefan Th Gries. 2013. 50-something years of work on collocations: what is or should be next. *International Journal of Corpus Linguistics*, 18(1):137–166.

Alexander Löser, Sebastian Arnold, and Tillmann Fiehn. 2012. The goolap fact retrieval framework. In *Business Intelligence*, pages 84–97. Springer.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics.

Andrea Moro and Roberto Navigli. 2012. Wisenet: Building a wikipedia-based semantic network with ontologized relations. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1672–1676. ACM.

Andrea Moro and Roberto Navigli. 2013. Integrating syntactic and semantic analysis into the open information extraction paradigm. In *IJCAI*.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: a taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1135–1145. Association for Computational Linguistics.

Harinder Pal and Mausam. 2016. Demonyms and compound relational nouns in nominal open ie. In *Proceedings of Workshop on Automated Knowledge Base Construction*, pages 35–39.

Fabio Petroni, Luciano del Corro, and Rainer Gemulla. 2015. Core: Context-aware open relation extraction with factorization machines. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1763–1773. Association for Computational Linguistics.

Randolph Quirk, Sidney Greenbaum, Geoffrey Leech, Jan Svartvik, and David Crystal. 1985. *A comprehensive grammar of the English language*, volume 397. Cambridge Univ Press.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics*.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Roser Saurí and James Pustejovsky. 2012. Are you sure that this happened? assessing the factuality degree of events in text. *Computational Linguistics*, 38(2):261–299.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a large benchmark for open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Austin, Texas. Association for Computational Linguistics.

Gabriel Stanovsky, Mausam, and Ido Dagan. 2015. Open ie as an intermediate structure for semantic tasks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 303–308. Association for Computational Linguistics.

Fei Wu and Daniel S Weld. 2010. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127. Association for Computational Linguistics.

Mohamed Yahya, Steven Whang, Rahul Gupta, and Alon Y Halevy. 2014. Renoun: Fact extraction for nominal attributes. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 325–335. Association for Computational Linguistics.

Scientific Information Extraction with Semi-supervised Neural Tagging

Yi Luan   Mari Ostendorf   Hannaneh Hajishirzi
Department of Electrical Engineering, University of Washington
{luanyi, ostendor, hannaneh}@uw.edu

Abstract

This paper addresses the problem of extracting keyphrases from scientific articles and categorizing them as corresponding to a task, process, or material. We cast the problem as sequence tagging and introduce semi-supervised methods to a neural tagging model, which builds on recent advances in named entity recognition. Since annotated training data is scarce in this domain, we introduce a graph-based semi-supervised algorithm together with a data selection scheme to leverage unannotated articles. Both inductive and transductive semi-supervised learning strategies outperform state-of-the-art information extraction performance on the 2017 SemEval Task 10 ScienceIE task.

1 Introduction

As a research community grows, more and more papers are published each year. As a result there is increasing demand for improved methods for finding relevant papers and automatically understanding the key ideas in those papers. However, due to the large variety of domains and extremely limited annotated resources, there has been relatively little work on scientific information extraction. Previous research has focused on unsupervised approaches such as bootstrapping (Gupta and Manning, 2011; Tsai et al., 2013), where hand-designed templates are used to extract scientific keyphrases, and more templates are added through bootstrapping.

Very recently a new challenge on Scientific Information Extraction (ScienceIE) (Augenstein et al., 2017)[1] provides a dataset consisting of 500 scientific paragraphs with keyphrase annotations for three categories: TASK, PROCESS, MATERIAL across three scientific domains, Computer Science (CS), Material Science (MS), and Physics (Phy), as in Figure 1. This dataset enables the use of more advanced approaches such as neural network (NN) models. To that end, we cast the keyphrase extraction task as a sequence tagging problem, and build on recent progress in another information extraction task: Named Entity Recognition (NER) (Lample et al., 2016; Peng and Dredze, 2015). Like named entities, keyphrases can be identified by their linguistic context, e.g. researchers ”use” methods. In addition, keyphrases can be associated with different categories in different contexts. For example, ‘semantic parsing’ can be labeled as a TASK in one article and as a PROCESS in another. Scientific keyphrases differ in that they can include both noun phrases and verb phrases and in that non-standard “words” (equations, chemical compounds, references) can provide important cues.

Since the scale of the data is still small for supervised training of neural systems, we introduce semi-supervised methods to the neural tagging

Computer Science:
This paper addresses the task of **[named entity recognition]**$_{\text{Task}}$, using **[conditional random fields]**$_{\text{Process}}$. Our method is evlauated on the **[ConLL NER Corpus]**$_{\text{Material}}$.

Physics:
[Local field effects] $_{\text{Process}}$ on spontaneous emission rates within **[nanostructure photonics material]**$_{\text{Material}}$ for example are familiar, and have been well used.

Material Science:
The **[Kelvin probe force microscopy technique]** $_{\text{Process}}$ allows **[detection of local EWF]**$_{\text{Task}}$ between an **[atomic force micorscopy]**$_{\text{Material}}$ and **[metal surface]**$_{\text{Material}}$.

Figure 1: Annotated ScienceIE examples.

[1] SemEval (Task 10)https://scienceie.github.io/index.html

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2641–2651
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

model in order to take advantage of the large quantity of unlabeled scientific articles. This is particularly important because of the differences in keyphrases across domains. Our semi-supervised learning algorithm uses a graph-based label propagation scheme to estimate the posterior probabilities of unlabeled data. It additionally extends the training objective to leverage the confidence of the estimated posteriors. The new training treats low confidence tokens as missing labels and computes the sentence-level score by marginalizing over them.

Our experiments show that our neural tagging model achieves state-of-the-art results in the SemEval Science IE task. We further show that both inductive and transductive semi-supervised strategies significantly improve the performance. Finally, we provide in-depth analysis of domain differences as well as analysis of failure cases.

The key contributions of our work include: i) achieving state of the art in scientific information extraction SEMEVAL Task 10 by extending recent advances in neural tagging models; ii) introducing a semi-supervised learning algorithm that uses graph-based label propagation and confidence-aware data selection, iii) exploring different alternatives for taking advantage of large, multi-domain unannotated data including both unsupervised embedding initialization and semi-supervised model training.

2 Related Work

There has been growing interest in research on automatic methods to help researchers search and extract useful information from scientific literature. Past research has addressed citation sentiment (Athar and Teufel, 2012b,a), citation networks (Kas, 2011; Gabor et al., 2016; Sim et al., 2012; Do et al., 2013; Jaidka et al., 2014), summarization (Abu-Jbara and Radev, 2011) and some analysis of research community (Vogel and Jurafsky, 2012; Anderson et al., 2012; Luan et al., 2012, 2014b; Levow et al., 2014). However, due to scarce hand-annotated data resources, previous work on information extraction (IE) for scientific literature is very limited. Gupta and Manning (2011) first proposed a task that defines scientific terms for 474 abstracts from the ACL anthologhy (Bird et al., 2008) into three aspects: *domain*, *technique* and *focus* and apply template-based bootstrapping to tackle the problem. Based on this study, Tsai et al. (2013) improve the performance by introducing hand-designed features from NER (Collins and Singer, 1999) to the bootstrapping framework. QasemiZadeh and Schumann (2012) compile a dataset of scientific terms into 7 fine-grained categories for 171 abstracts of ACL anothology. Similar to our work, very recently Augenstein and Søgaard (2017) also evaluated on ScienceIE dataset, but use multi-task learning to improve the performance of a supervised neural approach. Instead, we introduce a semi-supervised neural tagging approach that leverages unlabeled data.

Neural tagging models have been recently introduced to tagging problems such as NER. For example, Collobert et al. (2011) use a CNN over a sequence of word embeddings and apply a CRF layer on top. Huang et al. (2015) use hand-crafted features with LSTMs to improve performance. There is currently great interest in using character-based embeddings in neural models. (Chiu and Nichols, 2016; Lample et al., 2016; Ballesteros et al., 2015; Ma and Hovy, 2016). Our approach also takes advantage of neural tagging models and character-based embeddings for IE in scientific articles.

Previous work on semi-supervised learning for neural models has mainly focused on transfer learning (Dai and Le, 2015; Luan et al., 2014a; Harsham et al., 2015) or initializing the model with pre-trained word embeddings (Mikolov et al., 2013; Pennington et al., 2014; Levy and Goldberg, 2014; Luan et al., 2016b, 2015, 2016a). In our work, we use pre-training but also use more powerful methods including graph-based semi-supervision (Subramanya and Bilmes, 2011; Liu and Kirchhoff, 2013, 2015, 2016a,b) and a method for leveraging partially labeled data (Kim et al., 2015). We show that the combination of these techniques gives better results than any one alone.

3 Problem Definition and Data

The purpose of this work is to extract phrases that can answer questions that researchers usually face when reading a paper: What TASK has the paper addressed? What PROCESS or method has the paper used or compared to? What MATERIALS has the paper utilized in experiments? While these fundamental concepts are important in a wide variety of scientific disciplines, the terms that are used in specific disciplines can be substantially different. For example, MATERIALS in computer science might be a text corpus, while they would be physical materials in physics or materials science.

Data We use the SemEval 2017 Task 10 ScienceIE dataset. Fig. 1 provides examples that illustrate the variation in domains, but also show that there are common cues such as "the task of", "using", "technique," etc. A challenge with this dataset is that the size of the training data is very small. It is built from ScienceDirect open access publications and consists of 500 journal articles, but only one paragraph of each article is manually labeled. Therefore, we use a large amount of external data to leverage the continuous-space representation of language in neural network model. We explore the effect of pre-training word embedding with two different external resources: i) a data set of Wikipedia articles as a general English resource, and ii) a data set of 50k Computer Science papers from ACM.[2]

Tagging Problem Formulation The task requires detecting the exact span of a keyphrase. In order to be able to distinguish spans of two consecutive keyphrases of the same type, we assign labels to every word in a sentence, indicating position in the phrase and the type of phrase. We formulate the problem as an IOBES (Inside, Outside, Beginning, End and Singleton) tagging problem where every token is labeled either as: B, if it is at the beginning of a keyphrase; E, if it ends the phrase; I, if it is inside a keyphrase but not the first or last token; S, if it is a single-word keyphrase; or O, otherwise. For example, "named entity recognition" in first sentence of Fig. 1 is labeled as "*B-Task I-task E-task*".

4 Neural Architecture Model

We introduce an end-to-end model to categorize scientific keyphrases, building on a neural named entity recognition model (Lample et al., 2016) and adding a feature-based embedding.

4.1 Model

We develop a 3-layer hierarchical neural model to tag tokens of the documents (details of the tokenization is in Sec. 6). (1) The token representation layer concatenates three components for each token: a bi-directional character-based embedding, a word embedding, and an embedding associated with orthographic and part-of-speech features. (2) The token LSTM layer uses a bidirectional LSTM to incorporate contextual cues from surrounding tokens to derive intermediate token embeddings. (3) The CRF tagging layer models token-level tagging decisions jointly using a CRF objective function to incorporate dependencies between tags.

Character-Based Embedding. The embedding for a token is derived from its characters as the concatenation of forward and backward representations from a bidirectional LSTM. The character lookup table is initialized at random. The advantage of building a character-based embedding layer is that it can handle out-of-vocabulary words and equations, which are frequent in this data, all of which are mapped to "UNK" tokens in the Word Embedding Layer.

Word Embedding. Words from a fixed vocabulary (plus the unknown word token) are mapped to a vector space, initialized using Word2vec pretraining with different combinations of corpora.

Feature Embedding. We map features to a vector space: capitalization (all capital, first capital, all lower, any capital but first letter) and Part-of-Speech tags.[3] We randomly initialize feature vectors and train them together as other parameters.

Token LSTM Layer We apply a bidirectional LSTM at the token level taking the concatenated character-word-feature embedding as input. The token representation obtained by stacking the forward and backward LSTM hidden states is passed as input to a linear layer that project the dimension to the size of label type space and is used as input to CRF layer.

CRF Layer Keyphrase categorization is a task where there is strong dependencies across output labels (e.g., I-TASK cannot follow B-Process). Therefore, instead of making independent tagging decisions for each output, we model them jointly using conditional random field (Lafferty et al., 2001). For an input sentence $\boldsymbol{x} = (x_1, x_2, x_3, \ldots, x_n)$, we consider P to be the matrix of scores output by the bidirectional LSTM network. P is of size $n \times m$, where n is the number of tokens in a sentence, and m is the number of distinct tags. $P_{t,i}$ corresponds to the score of

[2] Due to the difficulty of data collection, experiments with external data from the other two domains is left to future work.

[3] Dependency features were investigated but did not lead to performance gains.

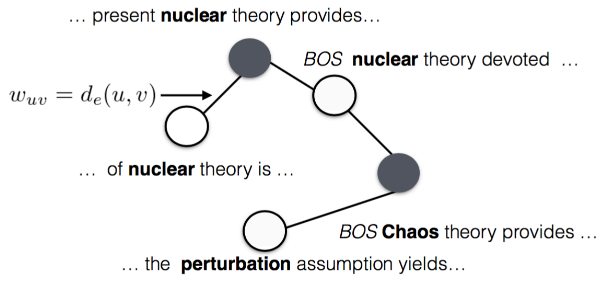

… present nuclear theory provides…
BOS nuclear theory devoted …
$w_{uv} = d_e(u,v)$
… of nuclear theory is …
BOS Chaos theory provides …
… the perturbation assumption yields…

Figure 2: Label propagation. Gray nodes indicates labeled data while white nodes are unlabeled. Bold font word indicates the current token. The assumption is if two instances are similar according to the graph, the output labels should be similar.

the i-th tag of the t-th word in a sentence. We use a first-order Markov Model and define a transition matrix $\boldsymbol{T}$ where $\boldsymbol{T}_{i,j}$ represents the score from tag i to tag j. We also add y_0 and y_n as the *start* and *end* tags of a sentence. Therefore $\boldsymbol{T}$ becomes a square matrix of dimension $m + 2$.

Given one possible output $\boldsymbol{y}$, and neural network parameters θ we define the score as

$$\phi(\boldsymbol{y}; \boldsymbol{x}, \theta) = \sum_{t=0}^{n} T_{y_t, y_{t+1}} + \sum_{t=1}^{n} P_{t, y_t} \tag{1}$$

The probability of sequence $\boldsymbol{y}$ is obtained by applying a softmax over all possible tag sequences

$$p_\theta(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp(\phi(\boldsymbol{y}; \boldsymbol{x}, \theta))}{\sum_{\boldsymbol{y}' \in Y} \exp(\phi(\boldsymbol{y}'; \boldsymbol{x}, \theta))} \tag{2}$$

where Y denotes all possible tag sequences. The normalization term is efficiently computed using the forward algorithm.

Supervised Training During training, we maximize the log-probability $\mathcal{L}(\boldsymbol{Y}; \boldsymbol{X}, \theta)$ of the correct tag sequence given the corpus $\{\boldsymbol{X}, \boldsymbol{Y}\}$. Backpropagation is done based on a gradient computed using sentence-level scores.

5 Semi-supervised Learning

We develop a semi-supervised algorithm that extends self-training by estimating the labels of unlabeled data and then using those labels for retraining. Specifically, we use a graph-based algorithm to estimate the posterior probabilities of unlabeled data and develop a new CRF training to take the uncertainty of the estimated labels into account while optimizing the objective function.

5.1 Graph-based Posterior Estimates

Our semi-supervised algorithm uses the following steps to estimate the posterior. It first constructs a graph of tokens based on their semantic similarity, then uses the CRF marginal as a regularization term to do label propagation on the graph. The smoothed posterior is then used to either interpolate with the CRF marginal or as an additional feature to the neural network.

Graph Construction Vertices in the graph correspond to tokens, and edges are distance between token features which capture semantic similarity. The total size of the graph is equal to the number of tokens in both labeled data V_l and unlabeled data V_u. The tokens are modelled with a concatenation of pre-trained word embeddings (with dimension d) of 5-gram centered by the current token, the word embedding of the closest verb, and a set of discrete features including part-of-speech tags and capitalization (43 and 4 dimension one-hot features). The resulting feature vector with dimension of $5d + d + 43 + 4$ is then projected down to 100 dimensions using PCA. We define the weight w_{uv} of the edge between nodes u and v as follows: $w_{uv} = d_e(u, v)$ if $v \in \mathcal{K}(u)$ or $u \in \mathcal{K}(v)$, where $\mathcal{K}(u)$ is the set of k-nearest neighbors of u and $d_e(u, v)$ is the Euclidean distance between any two nodes u and v in the graph. An example of our graph is in Fig. 2.

For every node i in the graph, we compute the marginal probabilities $\{\boldsymbol{q}_i\}$ using the forward-backward algorithm. Let θ^i represent the estimate of the CRF parameters after the n-th iteration, we compute the marginal probabilities $\tilde{\boldsymbol{p}}_{(j,t)} = p(y_t^j | \boldsymbol{x}; \theta^i)$ over IOBES tags for every token position t in sentence j in labeled and unlabeled data.

Label Propagation We use prior-regularized measure propagation (Liu and Kirchhoff, 2014; Subramanya and Bilmes, 2011) to propagate labels from the annotated data to their neighbors in the graph. The algorithm aims for the label distribution between neighboring nodes to be as similar to each other as possible by optimizing an objective function that minimizes the Kullback-Leibler distances between: i) the empirical distribution $\boldsymbol{r}_u$ of labeled data and the predicted label distribution $\boldsymbol{q}_u$ for all labeled nodes in the graph; ii) the distributions $\boldsymbol{q}_u$ and $\boldsymbol{q}_v$ for all nodes u in the graph and their neighbors v; iii) the distributions $\boldsymbol{q}_u$ and the CRF marginals $\tilde{\boldsymbol{p}}_u$ for all nodes. The third term regularizes the predicted distribution toward the

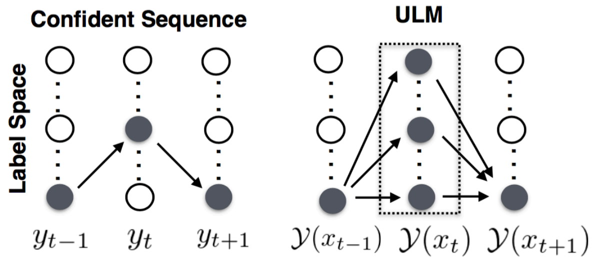


Figure 3: Lattice representation of ULM. Dashed box is the uncertain token which is going to be marginalized over. Arrows and grey nodes are paths to be summed over during training. When all tokens are confident, the score of only one path is calculated.

CRF prediction if the node is not connected to a labeled vertex, ensuring the algorithm performs at least as well as standard self-training.

Posterior Estimates We develop two strategies to estimate the new posteriors $\hat{p}(y_t|\boldsymbol{x};\theta)$, which can then be used in the CRF training.

The first strategy (called GRAPHINTERP) is the commonly used approach (Subramanya et al., 2010; Aliannejadi et al., 2014) that interpolates the smoothed posterior $\{\boldsymbol{q}\}$ with CRF marginals p:

$$\hat{p}(y_t|\boldsymbol{x};\theta) = \alpha p(y_t|\boldsymbol{x};\theta) + (1-\alpha)q(y) \qquad (3)$$

where α is a mixing coefficient.

A second strategy introduced here (called GRAPHFEAT) uses the smoothed posterior $\{\boldsymbol{q}\}$ as features and learns it with other parameters in the neural network. Given a sentence $\{x_1, \ldots, x_n\}$, let $Q = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n\}$ be the predicted label distribution from the graph. We then use Q as a feature input to neural network as $\tilde{P} = P + MQ$ where P is the $n \times m$ matrix output by the bidirectional LSTM network as in Eq. 1, and M is $m \times m$ matrix and is learned together with other parameters of neural network. We modify Eq. 1 by replacing P_{t,y_t} with $\tilde{P}_{t,y_t}$. Note that GRAPHFEAT can only be done in a transductive way since it requires output Q from the graph at test time.

5.2 CRF training with Uncertain Labels

A standard approach to self-training is to make hard decisions for labeling tokens based on the estimated posteriors and retrain the model. However, the estimated posteriors in our task are noisy due to the difficulty and variety of the ScienceIE task. Instead, we extend the CRF training to leverage the confidence of the estimated posteriors. The new CRF training (called Uncertain Label Marginalizing (ULM)) treats low confidence tokens as missing labels and computes the sentence-level score by marginalizing over them. A similar idea has been previously used in treating partially labeled data (Kim et al., 2015).

Specifically, given a sentence $\boldsymbol{x}$ we define a constrained *lattice* $\mathcal{Y}(\boldsymbol{x})$, where at each position t the allowed label types $\mathcal{Y}(x_t)$ are:

$$\mathcal{Y}(x_t) = \begin{cases} \{y_t\}, & \text{if} \quad p(y_t|\boldsymbol{x};\theta) > \eta \\ \text{All label types}, & \text{otherwise} \end{cases} \qquad (4)$$

where η is the confidence threshold, y_t is the prediction of posterior decoding and $p(y_t|\boldsymbol{x};\theta)$ is its CRF token marginal. The new neural network parameters θ are estimated by maximizing the log-likelihood of $p_\theta(\mathcal{Y}(\boldsymbol{x}^k)|\boldsymbol{x}^k)$ for every input sentence $\boldsymbol{x}^k$, where

$$p_\theta(\mathcal{Y}(\boldsymbol{x}^k)|\boldsymbol{x}^k) = \frac{\sum_{\boldsymbol{y}^k \in \mathcal{Y}(\boldsymbol{x}^k)} \exp(\phi(\boldsymbol{y}^k; \boldsymbol{x}^k, \theta))}{\sum_{\boldsymbol{y}' \in Y} \exp(\phi(\boldsymbol{y}'; \boldsymbol{x}, \theta))}$$

where $\boldsymbol{y}^k$ is an instance sequence of lattice $\mathcal{Y}(\boldsymbol{x})$, and k is the sentence index in the training set. Extreme cases are when all tokens are uncertain then the likelihood would be equal to 1, when all tokens of a sequence are confident, it would be equal to Eq. 2 where only one possible sequence, as in Fig. 3.

Inductive and Transductive Learning The semi-supervised training process is summarized as follow: It first computes marginals over the unlabeled data given a set of CRF parameters. It then uses the marginals as a regularization term for label propagation. The smoothed posteriors from the graph are then interpolated with the CRF marginal in GRAPHINTERP or used as an additional feature in GRAPHFEAT. It then uses the estimated labels for the unlabeled data combined with the labeled data to retrain the CRF using either the hard decision CRF training objective as Eq. 2 or the ULM data selection objective.

In the inductive setting, we only use the unlabeled data from the development set for the semi-supervision. In the transductive setting we also use the unlabeled data of the test set to construct the graph. In both cases, the parameters are tuned only on the dev set.

6 Experimental Setup

Data The SemEval ScienceIE (SE) corpus consists of 500 journal articles; one paragraph of each

Span Level	Classification (dev)	Classification (test)	Identification
Gupta et.al.(unsupervised)	-	9.8	6.4
Tsai et.al. (unsupervised)	-	11.9	8.0
MultiTask	45.5	-	-
Best Non-Neural SemEval$^+$	-	38	51
Best Neural SemEval$^+$	-	44	56
NN-Crf(supervised)	48.1	40.2	52.1
NN-Crf(semi)	51.9	45.3	56.9
NN-Crf(semi)*	**52.1**	**46.6**	**57.6**

Table 1: Overall span-level F1 results for keyphrase identification (SemEval Subtask A) and classification (SemEval Subtask B). * indicates tranductive setting. $^+$ indicates not documented as either transductive or inductive. - indicates score not reported or not applied.

Model	P	R	F1
NN-Crf(supervised)	46.2	48.2	47.2
No features	44.2	46.1	45.1
No bi-LSTM	45.2	44.7	44.9
No CRF	36.7	38.2	37.4
No char	45.7	46.2	45.9

Table 2: Ablation study showing impact of neural network configurations of our NN-Crf(supervised) model on the dev set.

article is randomly selected and annotated. The complete unlabeled articles and their metadata are provided together with the labeled data. The training data consists of 350 documents; 50 are kept for development and 100 for testing. The 500 articles come from 82 different journals evenly distributed in three domains. We manually labeled 82 journal names in the dataset into the three domains and do analysis based on the domain partitions. The 500 full articles contains 2M words and is 30 times the size of the annotated data.

Additionally, we use two external resources for pretraining word embeddings: i) WIKI, as for Wikipedia articles, specifically a full Wikipedia dump from 2012 containing 46M words, and ii) ACM, a collection of CS papers, containing 108M words.

Comparisons We compare our system with two template matching baselines and the state-of-the-art on the SemEval Science IE task. The first baseline (Gupta and Manning, 2011) is an unsupervised method to extract keyphrases by initially using seed patterns in a dependency tree, and then adding to seed patterns through bootstrapping. The second baseline (Tsai et al., 2013) improves the work of Gupta and Manning (2011) by adding Named Entity Features and use different set of seed patterns.

Implementation details All parameters are tuned on the dev set performance, the best parameters are selected and fixed for model switching and semi-supervised systems. The word embedding dimension is 250; the token-level hidden dimension is 100; the character-level hidden dimension is 25; and the optimization algorithm is SGD with a learning rate of 0.05. For building the graph, the best pre-trained embeddings for the supervised system (Sec. 7.2) are used in each domain. Two special tokens *BOS* and *EOS* are added when pre-training, indicating the begin and end of a sentence. The number of the graph vertices is 2M in tranductive setting and 1.4M in inductive setting. The ULM parameter η in Eq. 4 is tuned from 0.1 to 0.9, the best η is 0.4. The best parameters of label propagation are $\mu = 10^{-6}$ and $\nu = 10^{-5}$. The interpolation parameter α in Eq. 3 is tuned from 0.1 to 0.9, the best α is 0.3. We do iteration of semi-supervised learning until we obtain the best result on the dev set, which is mostly achieved in the second round.

We use Stanford CoreNLP (Manning et al., 2014) tokenizer to tokenize words. The tokenizer is augmented with a few hand-designed rules to handle equations (e.g. “fs(B,t)=Spel(t)S” is a single token) and other non-standard word phenomena (Cu40Zn, 20MW/m2) in scientific literature. We use Approximate Nearest Neighbor Searching (ANN)[4] to calculate the k-nearest neighbors. For all experiments in this paper, $k = 10$.

Setup We evaluate our system in both inductive and transductive settings. The systems with a * superscript in the table are transductive. The inductive setting uses 400 full articles in ScienceIE training and dev sets, while the transductive setting uses 500 full articles including the test set. In both settings parameters are tuned over the dev set.

[4] https://www.cs.umd.edu/~mount/ANN/

7 Experimental Results

We evaluate our NN-CRF model in both supervised and semi-supervised settings. We also perform ablations and try different variants to best understand our model.

7.1 Best Case System Performance

Table 1 reports the results of our neural sequence tagging model NN-CRF in both supervised and semi-supervised learning (ULM and graph-based), and compares them with the baselines and the state-of-the-art (best SemEval System (Augenstein et al., 2017)).

Augenstein and Søgaard (2017) use a multi-task learning strategy to improve the performance of supervised keyphrase classification, but they only report dev set performance on SemEval Task 10, we also include their result here and refer it as MULTITASK. We report results for both span identification (SemEval SubTask A) and span classification into TASK, PROCESS and MATERIAL (SemEval Subtask B).[5]

The results show that our neural sequence tagging models significantly outperforms the state of the art and both baselines. It confirms that our neural tagging model outperforms other non-neural and neural models for the SemEval ScienceIE challenge[6]. It further shows that our system achieves significant boost from semi-supervised learning using unlabeled data. Table 5 shows the detailed analysis of the system across different categories.

7.2 Supervised Learning

Impact of Neural Model Components Table 2 provides the results of an ablation study on the dev set showing the impact of different components of our NN-CRF on the Scientific IE task. For the basic model, the word embeddings are initialized by word2vec trained on the 350 full journal articles in the SE training set together with Wikipedia and ScienceIE data. The feature layer, character layer, and bi-LSTM word layers all improves the performance. Moreover, we observe a large improvement (20.6% relative) in the scientific IE task by adding the CRF layer.

Initialization Table 3 reports our NN-CRF performance when pretrained on different domains. We explore different word embedding pretraining with ScienceIE training set alone (SE), and adding other external resources including Wikipedia (wiki) and Computer Science articles (ACM). All alternatives use word2vec. Compared with using SE alone, introduction of all external data sources improve performance. Moreover, we observe that with the introduction of the ACM dataset, the performance on the CS domain is increased significantly in both the dev and test sets. Adding Wikipedia data benefits all three domains, with more significant improvement on the MS and Physics domains.

	Dev			Test		
Initialization	MS	Phy	CS	MS	Phy	CS
SE	49.4	39.4	45.0	42.9	33.0	30.5
+wiki	**52.9**	**40.5**	47.9	**46.1**	**39.2**	31.0
+ACM	50.3	39.8	**49.5**	42.2	37.8	34.2
+wiki+ACM	50.5	40.3	48.9	43.1	37.9	**34.4**

Table 3: F1 score on the dev and test sets for using different sources of data for pretraining.

Based on these observations, we select the best model on each domain according to the dev set and use the combined result as our best supervised system (called NN-CRF(supervised)). The F1 score improves from 39.4 to 40.2 when applying model switching strategy. The best model on the dev set is used for each domain: for MS and physics domain, we pretrain word embeddings with the SE and Wiki, and for the CS domain, we pretrain with the SE and ACM.

7.3 Semi-Supervision Learning

Table 4 reports the results of the semi-supervised learning algorithms in different settings. In particular we ablate incorporating the graph-based methods of computing the posterior and CRF training (ULM vs. hard decision). The table shows incorporating graph-based methods for computing posterior and ULM for CRF training outperforms their counterparts.

For computing the posterior, we explore two different strategies of the graph-based methods: i) GRAPHINTERP that interpolates the smoothed posterior from label propagation with CRF marginals; For inductive setting, GRAPHINTERP only uses un-annotated data from the dev set and uses the best model for decoding at test time. For transductive setting, GRAPHINTERP* uses unannoated data from test set to build the graph as

[5] The evaluation script is provided by the challenge, with a modification to report 3 decimal precision results.

[6] Best SemEval Numbers from https://scienceie.github.io/

Posterior	Training	Dev	Test
-	-	50.2	42.9
-	ULM	51.3	44.4
GraphInterp	-	50.9	43.3
GraphInterp	ULM	**51.9**	**45.3**
GraphInterp*	-	50.7	44.0
GraphInterp*	ULM	51.8	45.7
GraphFeat*	-	51.4	44.9
GraphFeat*	ULM	**52.1**	**46.6**

Table 4: F1 scores of semi-supervised Learning approaches; * shows transductive models.

Span Level	T	P	M	K
Best SemEval	19	44	48	55
supervised	13.3	40.5	43.7	52.1
ULM+GraphInterp	17.0	45.4	49.4	56.9
ULM+GraphFeat*	17.2	46.5	50.7	57.6
Token Level	T	P	M	K
supervised	29.6	56.0	59.3	70.8
ULM+GraphInterp	40.0	60.7	61.2	77.0
ULM+GraphFeat*	40.1	62.8	63.4	78.1

Table 5: F1 score results on the test set for different categories: T indicates Task, P indicates Process, M is Material and K is Keyword identification (SubTask A). * is transductive model.

well, and tune the parameters on the dev set. ii) GraphFeat uses the smoothed posterior from label propagation as additional feature to neural network and only has transductive setting.

As expected, the transductive approaches consistently outperform inductive approaches on the test set. With around the same performance on dev set, GraphInterp* seems to generalize better on test set with 1.6% relative improvement over GraphInterp. We observe higher improvement with GraphFeat* compared to GraphInterp. This is mainly because automatically learning the weight matrix M between neural network scores and graph outputs adds more flexibility compared to tuning an interpolation weight α. The performance is further improved by applying data selection through modifying the objective to ULM. The best inductive system is ULM+GraphInterp with 5.6% relative improvement over pure Self-Training that makes hard decisions, and the best transductive system is ULM+GraphFeat* with 8.6% relative improvement.

7.4 Category and Span Analysis

Table 5 details the performance of our method on the three categories at the span and token level. We observe significant improvement by using ULM+GraphInterp and ULM+GraphFeat over best SemEval and our best supervised system on all three categories at both token and span levels. We further observe that systems' performance on Task classification is much lower than Process and Material. This is in part because Task is much less frequent than the other types. In addition, Task keyphrases often include verb phrases while the other two domains mainly consists of noun phrases. An analysis of confusion patterns show that the most frequent type confusions are between Process and Material. However, we observe that ULM+GraphFeat* can greatly reduce the confusion, with 3.5% relative improvement of Process and 3.6% relative improvement of Process over ULM+GraphInterp on token level.

7.5 Error Analysis

We provide examples of typical errors that our system makes in Table 6. As described in the previous subsection, Task is the hardest type to identify with our system. Row 1 shows a failure to detect the verb phrase following 'to' as part of the Task, but detect 'enantiopure products' as Material. The system prefers to predict Process or Material since those classes have more samples than Task. Row 2 illustrates the problem of identifying general terms as keyphrases due to similar context, such as 'receptors' and 'drug action'. A third common error involves incorrectly labeling adjectives, such as 'neighbouring' in Row 3, which leads to span errors. Another common cause of error is insufficient context: in the last example, a larger context is needed to determine whether 'SWE' is a Process or Material.

8 Conclusion

This paper casts the scientific information extraction task as a sequence tagging problem and introduces a hierarchical LSTM-CRF neural tagging model for this task, building on recent results in NER. We introduced a semi-supervised learning algorithm that incorporates graph-based label propagation and confidence-aware data selection. We show the introduction of semi-supervision significantly outperforms the performance of the supervised LSTM-CRF tagging model. We additionally show that external resources are useful for initializing word embeddings. Both inductive and transductive semi-supervised strategies

Error types	Annotation and System Output
Verb phrases	A key requirement in aiming to [achieve [enantiopure products]$_{\text{Material}}$]$_{\text{Task}}$ is therefore a means to [quantitate [the enantiometric excess]$_{\text{Process}}$]$_{\text{Task}}$.
General terms	Since the [receptors]$_{\text{Material}}$ in human biology mostly consist of [chiral molecules]$_{\text{Material}}$, [drug action]$_{\text{Process}}$ mostly involves a specified enantiometric form.
Falsely predicted adjectives	It has been shown that the most efficient forms of energy transfer between the two occurs when there is a [neighbouring carotenoid species]$_{\text{Material}}$.
Lack of context	Other models use [SWEs]$^{\text{Material}}_{\text{Process}}$ but focus on the use of multi resolution grids or irregular mesh.

Table 6: Common errors, where blue means golden label our system misses, red means falsely predicted results, and green means correctly predicted spans.

achieve state-of-the-art performance in SemEval 2017 ScienceIE task. We also conducted a detailed analysis of the system and point out common error cases.

In our experiments, we observe that including in-domain data only for semi-supervised learning has slightly better performance than using cross-domain data. Reducing the amount of in-domain data hurts performance. Therefore, adding more in-domain unlabeled data may help when combined with selection schemes such as the ULM algorithms proposed here. It would be useful to assess the impact of matched unlabeled data for the physics and material science domain. Other future work includes leveraging global context, information of citation network.

9 Acknowledgments

This research was supported by the NSF (IIS 1616112), Allen Institute for AI (66-9175), Allen Distinguished Investigator Award, and gifts from Google, Samsung, and Bloomberg. We thank the anonymous reviewers for their helpful comments.

References

Amjad Abu-Jbara and Dragomir Radev. 2011. Coherent citation-based summarization of scientific papers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 500–509.

Mohammad Aliannejadi, Masoud Kiaeeha, Shahram Khadivi, and Saeed Shiry Ghidary. 2014. Graph-based semi-supervised conditional random fields for spoken language understanding using unaligned data. In *Australasian Language Technology Association Workshop*. page 98.

Ashton Anderson, Dan McFarland, and Dan Jurafsky. 2012. Towards a computational history of the ACL: 1980-2008. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*. Association for Computational Linguistics, pages 13–21.

Awais Athar and Simone Teufel. 2012a. Context-enhanced citation sentiment detection. In *Proceedings of the 2012 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies*. Association for Computational Linguistics, pages 597–601.

Awais Athar and Simone Teufel. 2012b. Detection of implicit citations for sentiment detection. In *Proceedings of the Workshop on Detecting Structure in Scholarly Discourse*. Association for Computational Linguistics, pages 18–26.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. *Proceedings of SemEval* .

Isabelle Augenstein and Anders Søgaard. 2017. Multi-task learning of keyphrase boundary classification. In *arXiv preprint arXiv:1704.00514*.

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *EMNLP*.

Steven Bird, Robert Dale, Bonnie J Dorr, Bryan R Gibson, Mark Thomas Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir R Radev, Yee Fan Tan, et al. 2008. The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *LREC*.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. In *TACL*.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the joint SIGDAT conference on empirical methods in natural language processing and very large corpora*. Citeseer, pages 100–110.

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

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*. pages 3079–3087.

Huy Hoang Nhat Do, Muthu Kumar Chandrasekaran, Philip S Cho, and Min Yen Kan. 2013. Extracting and matching authors and affiliations in scholarly documents. In *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*. ACM, pages 219–228.

Kata Gabor, Haifa Zargayouna, Davide Buscaldi, Isabelle Tellier, and Thierry Charnois. 2016. Semantic annotation of the ACL anthology corpus for the automatic analysis of scientific literature. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Paris, France.

Sonal Gupta and Christopher D Manning. 2011. Analyzing the dynamics of research by extracting key aspects of scientific papers. In *IJCNLP*. pages 1–9.

Bret Harsham, Shinji Watanabe, Alan Esenther, John Hershey, Jonathan Le Roux, Yi Luan, Daniel Nikovski, and Vamsi Potluru. 2015. Driver prediction to improve interaction with in-vehicle hmi. In *Proc. Workshop on Digital Signal Processing for In-Vehicle Systems (DSP)*.

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

Kokil Jaidka, Muthu Kumar Chandrasekaran, Beatriz Fisas Elizalde, Rahul Jha, Christopher Jones, Min-Yen Kan, Ankur Khanna, Diego Molla-Aliod, Dragomir R Radev, Francesco Ronzano, et al. 2014. The computational linguistics summarization pilot task. *Proceedings of TAC* .

Miray Kas. 2011. Structures and statistics of citation networks. Technical report, DTIC Document.

Young-Bum Kim, Minwoo Jeong, Karl Stratos, and Ruhi Sarikaya. 2015. Weakly supervised slot tagging with partially labeled sequences from web search click logs. In *HLT-NAACL*. pages 84–92.

John Lafferty, Andrew McCallum, Fernando Pereira, et al. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*. volume 1, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*.

Gina-Anne Levow, Valerie Freeman, Alena Hrynkevich, Mari Ostendorf, Richard Wright, Julian Chan, Yi Luan, and Trang Tran. 2014. Recognition of stance strength and polarity in spontaneous speech. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, pages 236–241.

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

Yuzong Liu and Katrin Kirchhoff. 2013. Graph-based semi-supervised learning for phone and segment classification. In *Proceedings of Annual Conference of the International Speech Communication Association (Interspeech)*.

Yuzong Liu and Katrin Kirchhoff. 2014. Graph-based semi-supervised acoustic modeling in DNN-based speech recognition. In *IEEE SLT*.

Yuzong Liu and Katrin Kirchhoff. 2015. Acoustic modeling with neural graph embeddings. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Yuzong Liu and Katrin Kirchhoff. 2016a. Graph-based semisupervised learning for acoustic modeling in automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24(11):1946–1956.

Yuzong Liu and Katrin Kirchhoff. 2016b. Novel front-end features based on neural graph embeddings for DNN-HMM and LSTM-CTC acoustic modeling. In *Proceedings of Annual Conference of the International Speech Communication Association (Interspeech)*. ISCA.

Yi Luan, Yangfeng Ji, Hannaneh Hajishirzi, and Boyang Li. 2016a. Multiplicative representations for unsupervised semantic role induction. In *The 54th Annual Meeting of the Association for Computational Linguistics*. page 118.

Yi Luan, Yangfeng Ji, and Mari Ostendorf. 2016b. Lstm based conversation models. In *arXiv preprint arXiv:1603.09457*.

Yi Luan, Daisuke Saito, Yosuke Kashiwagi, Nobuaki Minematsu, and Keikichi Hirose. 2014a. Semi-supervised noise dictionary adaptation for exemplar-based noise robust speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, pages 1745–1748.

Yi Luan, Masayuki Suzuki, Yutaka Yamauchi, Nobuaki Minematsu, Shuhei Kato, and Keikichi Hirose. 2012. Performance improvement of automatic pronunciation assessment in a noisy classroom. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*. IEEE, pages 428–431.

Yi Luan, Shinji Watanabe, and Bret Harsham. 2015. Efficient learning for spoken language understanding tasks with word embedding based pre-training. In *INTERSPEECH*. Citeseer, pages 1398–1402.

Yi Luan, Richard Wright, Mari Ostendorf, and Gina-Anne Levow. 2014b. Relating automatic vowel space estimates to talker intelligibility. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *ACL*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford CoreNLP natural language processing toolkit. In *ACL (System Demonstrations)*. pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *arXiv preprint arXiv:1301.3781*.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *EMNLP*. pages 548–554.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Behrang QasemiZadeh and Anne-Kathrin Schumann. 2012. The acl rd-tec 2.0: A language resource for evaluating term extraction and entity recognition methods. In *LREC*.

Yanchuan Sim, Noah A Smith, and David A Smith. 2012. Discovering factions in the computational linguistics community. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*. Association for Computational Linguistics, pages 22–32.

Amarnag Subramanya and Jeff Bilmes. 2011. Semi-supervised learning with measure propagation. *Journal of Machine Learning Research* 12(Nov):3311–3370.

Amarnag Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 167–176.

Chen-Tse Tsai, Gourab Kundu, and Dan Roth. 2013. Concept-based analysis of scientific literature. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, pages 1733–1738.

Adam Vogel and Dan Jurafsky. 2012. He said, she said: Gender in the ACL anthology. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*. Association for Computational Linguistics, pages 33–41.

NITE: A Neural Inductive Teaching Framework for Domain-Specific NER

Siliang Tang, Ning Zhang, Jinjian Zhang, Fei Wu and **Yueting Zhuang**
College of Computer Science, Zhejiang University, China
{siliang, aning, jibjianzhang, wufei, yzhuang}@zju.edu.cn

Abstract

In domain-specific NER, due to insufficient labeled training data, deep models usually fail to behave normally. In this paper, we proposed a novel Neural Inductive TEaching framework (NITE) to transfer knowledge from existing domain-specific NER models into an arbitrary deep neural network in a teacher-student training manner. NITE is a general framework that builds upon transfer learning and multiple instance learning, which collaboratively not only transfers knowledge to a deep student network but also reduces the noise from teachers. NITE can help deep learning methods to effectively utilize existing resources (i.e., models, labeled and unlabeled data) in a small domain. The experiment resulted on Disease NER proved that without using any labeled data, NITE can significantly boost the performance of a CNN-bidirectional LSTM-CRF NER neural network nearly over 30% in terms of F1-score.

1 Introduction

Domain-specific Named Entity Recognition (DNER), which aims to identify domain specific entity mentions and their categories, plays an important role in domain document classification, retrieval and content analysis. It is also a foundation for further level of complex information extraction tasks, serves as cornerstone in the knowledge computing process of transforming data into machine readable knowledge (Zhuang et al., 2017). Domain-specific NER is a challenging problem. For example, in biomedical domain, the number of unseen biomedical entity mentions (such as disease names, chemical names), their abbreviations or acronyms, as well as multiple names of the same entity is growing fast with the rapid increase of biomedical literatures and clinical records. However, the performance of a learning based NER system relies heavily on data annotation, which is quite expensive. The situation is even worse in domain-specific NER systems, since their data annotation requires the engage of domain experts. Therefore, in many special domains, only trained models or APIs are available, while their training data are private and inaccessible. On the other hand, due to insufficient labeled training data, deep models usually fail to behave normally in such domain, and state-of-the-art methods in these domains are usually dominated by rule based deductive methods or shallow model with hand-crafted features. However, the way of pre-defining useful domain specific hand-crafted features or rules are usually unavailable to the public.

In this paper, we proposed a novel Neural Inductive TEaching framework (NITE) to transfer knowledge from existing models into an arbitrary deep neural network. The idea of NITE is mainly borrowed from Transfer learning (Pan and Yang, 2010) where previously learned knowledge can aid current situation and solve problems with better solutions. In NITE, existing NER models behave like inefficient teachers to teach a deep neural network (we called student network) to identify named entities by giving it concrete examples. The knowledge transferred from these models is their posterior distributions on unlabeled data. These teachers are inefficient because they transfer not only useful information, but also errors to the student. The inputs of student network can be twofold, one is a small proportion from human labeled ground truth data (optional, like text book), and another is a large proportion from teachers, which is always noisy and less trustable.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2652–2657
Copenhagen, Denmark, September 7–11, 2017.

In such case, a student is overwhelmed and often inferior to the teachers, therefore in NITE, we introduced Multiple Instance Learning (MIL) trick (Dietterich et al., 1997; Babenko, 2008) to reduce the input noise during the model training.

In summary, NITE is a general framework that can help deep learning methods to make the best use of existing resources (i.e., models, labeled and unlabeled data). The experiment results on Disease NER (DNER) proved that without using any labeled data, NITE can significantly boost the performance of a CNN-bidirectional LSTM-CRF NER neural network (Ma and Hovy, 2016), which trained on NCBI training dataset nearly over 30% in terms of F1-score. It also outperformed the teacher model, which proved the correctness of our hypothesis.

2 Neural Inductive Teaching Framework

In this section we will define our NITE framework step by step, and apply it to Disease NER.

2.1 Inductive Teaching

Inductive teaching means teaching student by examples, our inductive teaching method builds upon teacher-student models (Ba and Caruana, 2014) and knowledge distillation (Hinton et al., 2015). The main idea of our method is to transfer discriminative knowledge from well-trained existing models (teachers) to a new and more capable model (student). The student learns by imitating the teachers' behaviors, and the teaching process can be defined as follows:

Let $x = \{w_1, w_2, \ldots, w_{|x|}\}$ be an input sentence of $|x|$ words, where w_k is the kth word in x. If l_k is the corresponding 3-dimensional one hot IOB (In-Out-Begin) vector for w_k, then the NER labeling sequence of x can be defined as $y = \{l_1, l_2, \ldots, l_{|x|}\}$.

For a given sentence x_i, we further define the posterior distribution of a teacher as $y_i^{f_t} = f_t(y_i|x_i)$, while the posterior distribution of a student network can be defined as $y_i^{f_s} = f_s(y_i|x_i;\theta)$, where θ is the parameters of the student network. During training, we measure the similarity between y^{f_t} and y^{f_s} with KL-divergence, and minimize their difference. Therefore, for a given x_i, we optimize:

$$\min_{\theta} \sum_{i=1}^{n} D_{KL}(f_t(y_i|x_i)||f_s(y_i|x_i;\theta)) \quad (1)$$

, where $D_{KL}(P||Q) = \sum_j P_j \log \frac{P_j}{Q_j}$ is the KL-divergence. This equation can be optimized through stochastic gradient descent over shuffled mini-batches with the Adadelta (Zeiler, 2012) update rule.

2.2 Multiple Instance Learning

Multiple Instance Learning is an effective training method that can help to train a supervised model to alleviate the wrong label problem (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012). Instead of predicting labels for each individual training sample, the objective of MIL is to predict the labels (positive or negative) of the unseen bags, where each bag contains a fixed number of instances (samples). The standard MIL assumption assumes that a bag is positively labeled if at least one instance in a bag is positive, and is negatively labeled if all instances in a bag are negative. MIL is generally used in training a binary classifier, to apply MIL in NITE, we redefine the label of a bag as the quality (correctness) of its containing samples. Thus, in NITE, a bag is positively labeled if at least one instance in it is labeled correctly. Furthermore, it is inappropriate to evaluate the correctness of IOB label (i.e., l_k) of each word (i.e., w_k), since the IOB sequence y_i of a sentence x_i is generated dependently. Therefore, we choose sentence x_i as our MIL instance, and the correctness of x_i is evaluated by the likelihood probability of all words with correct BIO tags. In general, our MIL can be formally defined as follows:

Randomly allocate training samples in a mini-batch $\mathcal{B}$ into M bags, i.e., $\mathcal{B} = \{B_1, B_2, \ldots, B_M\}$ with their corresponding labels $\{z_1, z_2, \ldots, z_M\}$, where $z_m \in \{-1, 1\}$. For bag B_m, it contains K instances, i.e., $B_m = \{x_1, x_2, \ldots, x_K\}$, where x_i is a sentence with its posterior evaluation $y_i^{f_s}$.

During the training, given a bag B_m, if $z_m = 1$, which means B_m is a positive bag. In order to reduce the noise, our MIL learner will select the most correct instance $y_{i*}^{f_s}$, which has the maximum likelihood among all other instances (i.e., sentence) in the bag B_m. That is $P(z_m = 1|B_m) = P(y_{i*}^{f_s}) = \arg\max_i\{P(y_i^{f_s}|x_i)\}$, where $1 \leq i \leq K, x_i \in B_m$. If $z_m = -1$, which means B_m is a negative bag, in order to better detect such negative bags, our MIL learner should select the most violated instance for learning, which is also the instance with maximum likelihood. Thus, the bag label z (which indicates the sentence is labeled correctly or incorrectly) is actually integrated out, since no matter what the value z is, MIL in NITE will always select the instance with the highest likelihood probability. Finally the MIL in NITE can be summarized as:

$$P(z_m|B_m) = P(y_{i*}^{f_s}) = \arg\max_{i=1:K}\{P(y_i^{f_s}|x_i)\} \tag{2}$$

In summary, MIL in NITE can be regarded as a mechanism for posterior selection, or regularization on posterior distribution of a student network. Therefore, MIL only affects the model training, and it will not affect the testing process.

2.3 Teacher Model & Student Network

Theoretically, the teacher model of NITE can be any existing well-trained model, while the student network can be an arbitrary deep neural network. In this paper, we focus on domain-specific NER, and more specifically on Disease NER, which is a small but typical domain that is suffering from insufficient labeled training data.

There are many existing DNER systems, and the most well-known systems are BANNER (Leaman et al., 2008), and DNorm (Leaman et al., 2013). BANNER is an open-source biomedical NER system implemented using conditional random fields (CRFs) (Lafferty et al., 2001). While, DNorm uses supervised semantic indexing, is trained with pairwise learning to rank, to score the mentions returned by BANNER. Therefore, DNorm can be regarded as an extension of BANNER, and the whole system depends on handcrafted features such as word spelling features and orthographic features. DNorm is the state-of-the-art DNER system, and therefore we adopt DNorm as our teacher model.

For the student network, we are looking for state-of-the-art solutions in general NER. There are many studies on applying complex deep learning models on general NER or other sequence labeling tasks. Without any feature engineering trick, deep models have achieved comparable or better performances than many other traditional methods. More recently, Ma and Hovy (2016) proposed a method that concatenated CNN, bidirectional LSTM, and CRF successively to form an end to end deep NER model (CLC for short). CLC achieved state-of-the-art performance in general NER, and therefore we take the CLC as our student network, Fig. 1 shows the overall architecture of our student network.

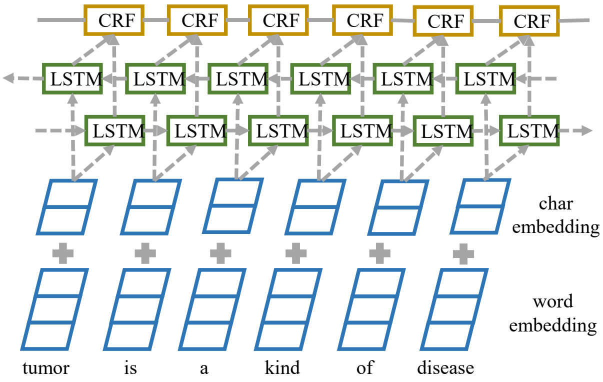


Figure 1: the Flowchart of the Student Network

As shown in Fig. 1, the character-level embeddings are generated by CNN layers, then are concatenated with pre-trained word embeddings, and finally fed into the bidirectional LSTM layer. The bidirectional LSTM is efficient to capture syntactic and semantic information both preceding and following simultaneously. Its output vectors are fed into the CRFs layer for IOB sequence labeling. It uses maximum conditional likelihood estimation to choose parameters during the finally CRFs training process, and its likelihood can be given as follows:

$$P(y_i^{f_s}|x_i) = \arg\max_{y\in\mathcal{Y}(x_i)} P(y|x_i) \tag{3}$$

, where $\mathcal{Y}(x_i)$ denotes the set of possible label sequences for x_i. Eq. 3 can be solved efficiently by adopting the Viterbi algorithm.

Fig. 2 shows the whole NITE-NER training process. For each training iteration, training samples in a mini-batch are randomly allocated into M bags, and then fed into the student network f_s. For bag B_m, the student network will generate posterior evaluation $y_i^{f_s}$ for each input instance $x_i \in B_m$ respectively. Then the MIL module will select the best sample $y_{i*}^{f_t}$ from all K instances according to Eq. 3 and 2. Finally, NITE will retrieve posterior evaluation $y_{i*}^{f_t}$ from the teacher, and update θ based on Eq. 1.

3 Experiments

In this section we designed several experiments to testify our hypothesis of inductive teaching as well as evaluate our NITE framework.

3.1 Training Corpus

Although NITE is a supervised learning framework, the discriminative knowledge of student net-

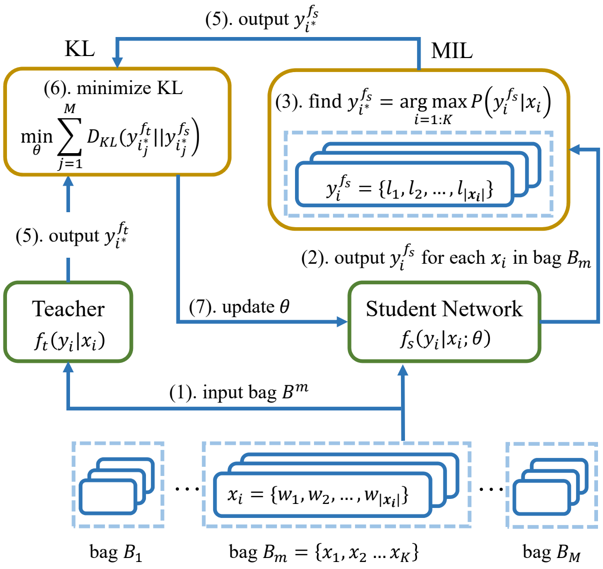

(5). output $y_{i^*}^{f_s}$
KL
MIL
(6). minimize KL
$\min_{\theta} \sum_{j=1}^{M} D_{KL}(y_{i_j^*}^{f_t} || y_{i_j^*}^{f_s})$
(3). find $y_{i^*}^{f_s} = \arg\max_{i=1:K} P\left(y_i^{f_s} | x_i\right)$
$y_i^{f_s} = \{l_1, l_2, \ldots, l_{|x_i|}\}$
(5). output $y_{i^*}^{f_t}$
(2). output $y_i^{f_s}$ for each x_i in bag B_m
Teacher
$f_t(y_i|x_i)$
(7). update θ
Student Network
$f_s(y_i|x_i;\theta)$
(1). input bag B^m
$x_i = \{w_1, w_2, \ldots, w_{|x_i|}\}$
bag B_1
bag $B_m = \{x_1, x_2 \ldots x_K\}$
bag B_M

Figure 2: The training process of NITE-NER.

work is learned indirectly from the teacher models, therefore NITE can be trained without any labeled data.

To evaluate the efficiency of the NITE framework, we trained two DNER models on NCBI disease corpus (Doğan et al., 2014; Islamaj Dogan and Lu, 2012). One is the well-known DNorm model, which is the state-of-the-art method in disease NER. Another one is the bi-directional LSTM-CNN-CRF NER neural network i.e., CLC (Ma and Hovy, 2016), which has the state-of-the-art performance in general NER task. The CLC architecture also serves as our student network.

The NCBI disease corpus is a widely used data corpus with disease name and related concept annotations in biomedical research field. The corpus is an extension of the AZDC corpus (Leaman et al., 2009) which was annotated only with disease mentions. The detailed characteristics of the NCBI disease corpus as well as how we partition the data are shown in Table 1.

NCBI	Train	Validate	Test
# of documents	593	100	100
# of sentences	5661	791	961
# of disease	5148	791	961
Specific Disease	2959	409	556
Disease Class	781	127	121
Modifier	1292	218	264
Composite Mention	116	37	20

Table 1: The description of the NCBI corpus as training, validating and testing sets for the recognition of disease named entity

3.2 Experiment Setup

The experiment's setup is as follows:

Our NITE-DNER is trained without any labeled data, we randomly sampled 2,000 unlabeled abstracts of biomedical literature from PubMed as our training data. The DNorm model is served as the teacher model in the NITE framework.

In student network, we initialized character embeddings with uniform samples from $[-\sqrt{\frac{3}{d}}, +\sqrt{\frac{3}{d}}]$, where we set the dimension $d = 30$. We use 30 filters with window length 3 in CNN and 200 hidden states in bi-directional LSTM. In training procedure we set initial learning rate $\eta_0 = 0.015$ with decay rate $\rho = 0.05$, the learning rate is updated as $\eta_t = \eta_0/(1.0 + \rho n)$, where n is the number of epochs. We use a fixed dropout rate 0.5 at CNN and both input and output vectors of bi-directional LSTM to mitigate overfitting. For MIL we set the bag size $K = 5$ with mini-batch size 30. We implemented neural networks on a GeForce GTX 1080 using Theano.

3.3 Results and Discussion

We evaluated all three DNER methods on the NCBI test set in terms of precision, recall and F1-score. All the measurements are based on exact location of extracted disease mentions in the given test sentences.

Method	**CLC-DNER**	**DNorm**	**NITE**
Labels	NCBI	NCBI	-
Remark	Student only	Teacher only	S+T+MIL
Precision	79.20	80.50	85.40
Recall	51.73	75.70	75.07
F1-score	62.58	78.06	**79.91**

Table 2: Performance comparisons.

The experiment results are presented in Table 2. As shown in Table 2, although the complex CLC network is the state-of-the-art method in general NER, it behaves poorly in domain-specific NER task due to insufficient labeled training data. However, with the help of our NITE framework, its performance is significantly boosted, and reached the comparable level of DNorm. This proved that knowledge transfer in NITE is efficient and important in training a deep model of domain-specific NER.

3.4 Conclusion

In this paper, we proposed a general framework, NITE, and demonstrated its efficiency in transferring DNER knowledge into an end to end deep NER model. Although we only proposed a solution for DNER, it could be easily applied to other domain-specific NER problems (e.g., chemical, gene, and protein) or even applications other than NER. The experiment results suggested that NITE can be very helpful on training a deep model when other resources are available. For future work, a NITE architecture with more than one teacher could be considered. Moreover, as mentioned in (Zhou et al., 2017), crowd knowledge can be used to reshape deep learning features. Our framework can also incorporate crowd knowledge easily, in which the teachers can be human crowds, and then the NITE can employs active learning (Olsson, 2009) or lifelong machine learning (Chen and Liu, 2016) to progressively polishing the student model.

Acknowledgments

This work was supported in part by the 973 program (No. 2015CB352302), NSFC (No. 61625107, U1611461, 61402401), Chinese Knowledge Center of Engineering Science and Technology (CKCEST), Qianjiang Talents Program of Zhejiang Province 2015 and Key program of Zhejiang Province (2015C01027).

References

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662.

Boris Babenko. 2008. Multiple instance learning: algorithms and applications. *View Article PubMed/NCBI Google Scholar*.

Zhiyuan Chen and Bing Liu. 2016. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(3):1–145.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1):31–71.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Rezarta Islamaj Dogan and Zhiyong Lu. 2012. *BioNLP: Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*, chapter An improved corpus of disease mentions in PubMed citations. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, volume 1, pages 282–289.

Robert Leaman, Rezarta Islamaj Doğan, and Zhiyong Lu. 2013. Dnorm: disease name normalization with pairwise learning to rank. *Bioinformatics*, page btt474.

Robert Leaman, Graciela Gonzalez, et al. 2008. Banner: an executable survey of advances in biomedical named entity recognition. In *Pacific symposium on biocomputing*, volume 13, pages 652–663.

Robert Leaman, Christopher Miller, and G Gonzalez. 2009. Enabling recognition of diseases in biomedical text with machine learning: corpus and benchmark. In *Proceedings of the 2009 Symposium on Languages in Biology and Medicine*, volume 82.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Fredrik Olsson. 2009. A literature survey of active machine learning in the context of natural language processing.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Le-kui Zhou, Si-liang Tang, Jun Xiao, Fei Wu, and Yue-ting Zhuang. 2017. Disambiguating named entitieswith deep supervised learning via crowd labels. *Frontiers of Information Technology & Electronic Engineering*, 18(1):97–106.

Yue-ting Zhuang, Fei Wu, Chun Chen, and Yun-he Pan. 2017. Challenges and opportunities: from big data to knowledge inai 2.0. *Frontiers of Information Technology & Electronic Engineering*, 18(1):3–14.

Speeding up Reinforcement Learning-based Information Extraction Training using Asynchronous Methods

Aditya Sharma
Indian Institute of Science
Bangalore, India
adisharma075@gmail.com

Zarana Parekh[*]
DA-IICT
Gandhinagar, India
zaranaparekh17@gmail.com

Partha Talukdar
Indian Institute of Science
Bangalore, India
ppt@iisc.ac.in

Abstract

RLIE-DQN is a recently proposed Reinforcement Learning-based Information Extraction (IE) technique which is able to incorporate external evidence during the extraction process. RLIE-DQN trains a single agent *sequentially*, training on one instance at a time. This results in significant training slowdown which is undesirable. We leverage recent advances in parallel RL training using asynchronous methods and propose RLIE-A3C. RLIE-A3C trains multiple agents in *parallel* and is able to achieve upto 6x training speedup over RLIE-DQN, while suffering no loss in average accuracy.

1 Introduction

Extracting information about an event (or entity) involves multiple decisions, as one first needs to identify documents relevant to the event, extract relevant information from those documents, and finally reconcile various values obtained for the same relation of the event from different sources (Ahn, 2006). Search based methods for Information Extraction have been increasingly investigated (West et al., 2014); (Hegde and Talukdar, 2015); (Zhang et al., 2016); (Bing et al., 2017).

(Kanani and McCallum, 2012) combine search and Information Extraction (IE) using Reinforcement Learning (RL), with the goal of selecting good actions while staying within resource constraints, but don't optimze for extraction accuracy. More recently, (Narasimhan et al., 2016) proposed a RL-based approach to model the IE process outlined above. We shall refer to this approach as RLIE-DQN in this paper. RLIE-DQN trains an RL agent using Deep Q-Network (DQN) (Mnih et al., 2015) to select optimal actions to query for documents and also reconcile extracted values.

DQN trains a single agent *sequentially*, updating parameters based on one instance at a time. Each such instance is sampled from the entire training data, also called the *experience replay*. Such sequential experience replay-based training results in slow learning, while requiring high memory and computation resources. In order to overcome this challenge, A3C (Asynchronous Advantage Actor-Critic), an asynchronous deep RL training algorithm, has been proposed recently (Mnih et al., 2016). A3C trains multiple RL agents in *parallel*, each of which estimates gradients locally, and asynchronously updates globally shared parameters. Recent work has explored applications for A3C in varied domains (Fernando et al., 2017), (Mirowski et al., 2016).

In this paper, we propose RLIE-A3C which uses A3C-based parallel asynchronous agents for training. This is in contrast to the sequential DQN training in RLIE-DQN. Differences between the training regimes of the two methods are shown in Figure 1. Through experiments on multiple real-world datasets, we find that RLIE-A3C achieves upto 6x training speedup compared to RLIE-DQN, while suffering no loss in accuracy. To the best of our knowledge, this is the first application of asynchronous deep RL methods in IE (and also in NLP), and we hope this paper will foster further adoption and research into such methods in the NLP community. RLIE-A3C code is available at https://github.com/adi-sharma/RLIE_A3C

[*] Research carried out during an internship at the Indian Institute of Science, Bangalore.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2658–2663
Copenhagen, Denmark, September 7–11, 2017.

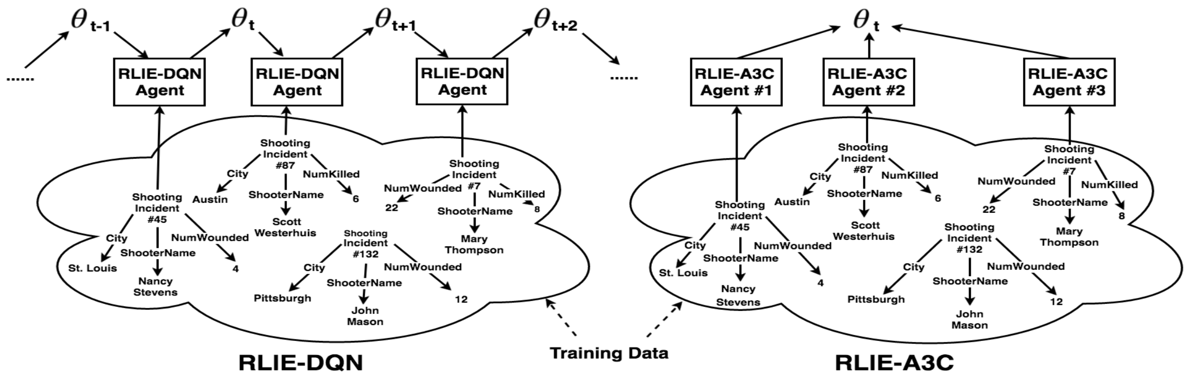


Figure 1: **Left:** DQN-based *sequential* learning framework used in RLIE-DQN (Narasimhan et al., 2016), as discussed in Section 2. At each time step, the agent looks at a specific instance from the training data. **Right:** A3C-based *parallel* learning framework in RLIE-A3C (proposed approach). The parallel agents look at different parts of the training data, estimate parameter update statistics locally, and then perform *asynchronous* updates on the globally shared parameters (θ_t) at time step t. See Section 3 for details. Due to the asynchronous parallel updates, RLIE-A3C achieves significant training speedup without loss in accuracy, as we shall see in Section 4.

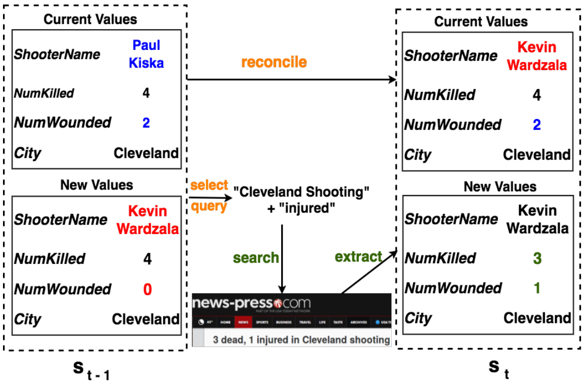


Figure 2: Sample state transition in the MDP of RLIE-DQN (and also RLIE-A3C). In each transition, two actions are carried out: (1) reconcile new values with current values,; and (2) issue query to retrieve other relevant documents and extract values from those documents. Please see Section 2 for details.

2 RLIE-DQN: Information Extraction using Reinforcement Learning

We first present a brief overview of RLIE-DQN (Narasimhan et al., 2016). Given a document to extract information about an event, RLIE-DQN issues a search query to retrieve other documents related to the event, extracts event information from those documents, and finally reconciles values extracted from the documents. If confidence in the extracted values are low, then RLIE-DQN repeats this process with additional queries. This way, RLIE-DQN incorporates evidences from external sources to improve information extraction (IE) from a given source document.

RLIE-DQN models the task as a Markov Decision Process (MDP) in order to reconcile newly extracted information selectively and dynamically. The MDP describes the environment in which the RL agent learns to make decisions. The MDP is represented using a tuple $\langle S, A, R, T\rangle$, where S is the set of states, $A = \{a_d, a_q\}$ is the set of actions, $R(s, a)$ is the reward for taking action $a \in A$ from state $s \in S$, and $T(s^{'}|s, (a_d, a_q))$ is the state transition function. Here, a_d is the reconciliation action, while a_q is the query action. Based on a_d, the agent may accept extracted values for one or all relations, reject all newly extracted values or stop (episode ends).

A sample state transition in RLIE-DQN's MDP is shown in Figure 2. State representation consists of many details such as confidence scores of current and newly extracted relation values, context statistics from which the extractions are performed, etc. But for better readability, only the set of current and new values are shown for the states in this figure. Each transition consists of two actions: reconcile decision and query. The RL agent uses the reconcile action (a_d) to update value of the *ShooterName* relation from *Paul Kiska* to *Kevin Wardzala*. The agent uses the query action (a_q) to issue a new query (*"Cleveland shooting"* + *"injured"*) to retrieve other relevant documents and extract new values 3 and 1 for relations *NumKilled* and *NumWounded*, respectively. The transitions stop whenever a_d is a stop decision.

The reward function at each state, is the cumulative difference of current and previous extracted accuracies, summed over all relations. Also, a negative reward per step is added to the reward in order to penalize the agent for longer episodes, since issuing queries to a search engine is expensive.

Deep Q-Learning (DQN): Let $Q(s, a)$ be the measure of long-term cumulative reward obtained by taking action a from state s. The RL agent makes use of $Q(s, a)$ to select the next action from a from state s. Q-learning (Watkins and Dayan, 1992) is a popular technique to estimate this function, in which the function for optimal Q-value is estimated using the Bellman equation (Sutton and Barto, 1998) $Q_{i+1}(s, a) = \mathbb{E}_{(s,a)}[R(s, a) + \gamma \max_{a'} Q_i(s', a')|s, a]$. Here, $R(s, a)$ is the immediate reward and γ is the discount factor for the value of future rewards.

For high dimensional state spaces, Deep Q-Network (DQN) (Mnih et al., 2013) approximates $Q(s, a)$ as $Q(s, a; \theta)$ using parameters θ of a deep network. RLIE-DQN used such a DQN-based agent to learn optimal policy, as shown in Figure 1.

3 Proposed Approach: RLIE-A3C

The DQN-based agent used in RLIE-DQN is sequential, as it moves from one instance to another to update parameters. This can result in significant training time slowdown, especially in large data settings. Instead of using experience replay of the DQN algorithm for stabilizing updates, we consider a framework with multiple asynchronous agents, each of which explore different areas of the environment in parallel.

Asynchronous Advantage Actor-critic (A3C) (Mnih et al., 2016) is a recently proposed deep RL algorithm which makes use of parallel agents for parameter estimation. We replace DQN with A3C in RLIE-DQN and call the resulting method RLIE-A3C – our proposed approach (See Figure 1).

At time instant t, RLIE-A3C poses decision policy $\pi_d(a_d|s_t)$, and query policy $\pi_q(a_q|s_t)$ as probability distributions over candidate actions a_d and a_q, respectively. RLIE-A3C also calculates the state value function $V(s_t)$, as an estimate of the cumulative long-term reward obtained starting from state s_t. Owing to the large continuous state space of the problem, RLIE-A3C approximates each of these three functions using three separate deep neural networks, which are parametrized by θ_d, θ_q, and θ_v, as $\pi_d(a_d|s_t) \approx \pi_d(a_d|s_t; \theta_d)$, $\pi_q(a_q|s_t) \approx \pi_q(a_q|s_t; \theta_d)$ and $V(s_t) \approx V(s_t; \theta_v)$. These parameters are updated by agents working in parallel. Based on the policies, each agent selects the query and decision actions to be performed and updates it's state accordingly. This is repeated up to t_{max} steps or until a terminal state is reached.

Local Gradient Calculation: The agents estimate parameter gradients using a local copy of the network parameters, and then perform *asynchronous* updates on the globally shared parameters θ_d, θ_q, and θ_v. Hence, the policy and value functions are jointly estimated. The policy gradient update equations calculated over the local copy of network parameters of each parallel RLIE-A3C agent p are as follows:

$$\begin{aligned} d\theta_x^p \leftarrow d\theta_x^p + \nabla_{\theta_x^p} \log \pi_d(a_{x_i}, s_;\theta_x^p) A(s_i, a_{x_t}; \theta_v^p) \\ + \beta_x \nabla_{\theta_x^p} H(\pi_x(s_t; \theta_x^p)) \end{aligned} \tag{1}$$

where, x $\in$ {d, q} for decision and query. The advantage function $A(s_t, a_t; \theta_v) = \sum_{i=0}^{k-1} \gamma^i R_{t+i} + \gamma^k V(s_{t+k}; \theta_v) - V(s_t; \theta_v)$ above significantly reduces the variance of the policy gradient, where γ is the discount factor and k can vary from state to state and is upper-bounded by t_{max}. Since the gradient updates are accumulated, training stability increases. Further, exploration is encouraged by introducing the entropy regularization term $\beta_x \nabla_{\theta_x^p} H(\pi_x(s_t; \theta_x^p))$ in the equation above. Here, H is the entropy function and β_x controls dominance of the entropy term.

The gradient update for the parameters of value function V is calculated locally by every parallel agent p as follows:

$$d\theta_v^p \leftarrow d\theta_v^p + \partial(G_t - V(s_t; \theta_v^p))^2 / \partial\theta_v^p$$

where $G_t = \mathbb{E}_{(s)}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}]$ is the return.

Global Parameter Update: The parameters θ_d, θ_q and θ_v are learnt using stochastic gradient descent with RMSprop (Tieleman and Hinton, 2012). The standard non-centered RMSProp update is used by the parallel agents p to update the shared parameters asynchronously using the gradients obtained from Equation (1), as follows:

$$g = \alpha g + (1 - \alpha)(d\theta_x^p)^2 \;\; ; \;\; \theta_x \leftarrow \theta_x - \eta \frac{d\theta_x^p}{\sqrt{g + \epsilon}}$$

where x $\in$ {d, q, v}, α is the decay factor, η is the learning rate, ϵ is the smoothing constant and

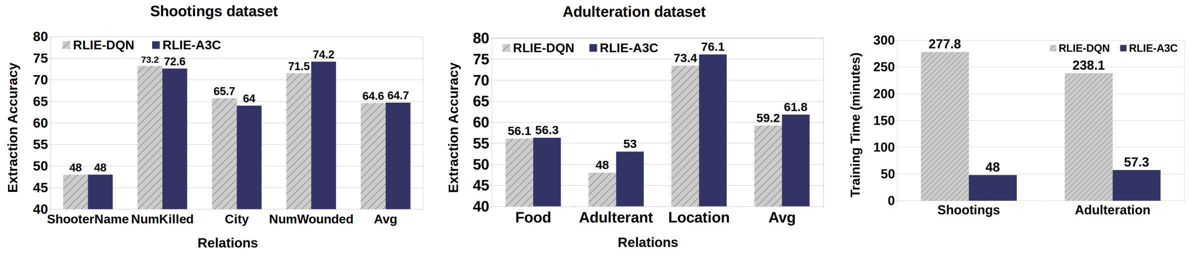


Figure 3: **Left, Middle Panels**: Extraction accuracy of the baseline (RLIE-DQN) and our system (RLIE-A3C) on the Shooting Incidents and Food Adulteration datasets. **Right Panel**: Training time comparison between RLIE-DQN and RLIE-A3C. Overall, we observe that RLIE-A3C results in upto 6x speedup over RLIE-DQN, without any loss in average extraction accuracy. This is our main result. Please see Section 4.1 for details.

g is the moving average of element-wise squared gradients. The pseudo-code for RLIE-A3C can be found in the Appendix.

4 Experiments and Results

Setup: We compare RLIE-A3C against RLIE-DQN using the same protocol and hyperparameters as reported in (Narasimhan et al., 2016). Also, we experiment with the same two datasets used in that paper: the Gun Violence Archive[1] and the Foodshield EMA database[2]. The train, dev and test datasets contain 372, 146 and 146 source articles respectively for the Shooting incident cases and 292, 42 and 148 source articles respectively for the food adulteration cases. We used the implementation of RLIE-DQN provided by the authors of that system. For more details on the dataset and other parameters, we refer the reader to (Narasimhan et al., 2016).

RLIE-A3C: This is our proposed method which is described in Section 3. For the sake of fair comparison, the network, base classifier, and evaluation metrics are same as that of RLIE-DQN. Each of the three deep networks in RLIE-A3C, one each for $\pi(a_d|s)$, $\pi(a_q|s)$ and $V(s)$, is built using two linear layers with 20 hidden units, followed by ReLUs. MaxEnt classifier is used as the base extractor, and the model is evaluated on the entire *test* set for 1.6 million steps. The *dev* set is used to tune all hyperparameters, which can be found in the Appendix. For RLIE-A3C, the evaluation is carried out 50 times after training and the average accuracy values are taken over the top 5 evaluations, as done in (Mnih et al., 2016).

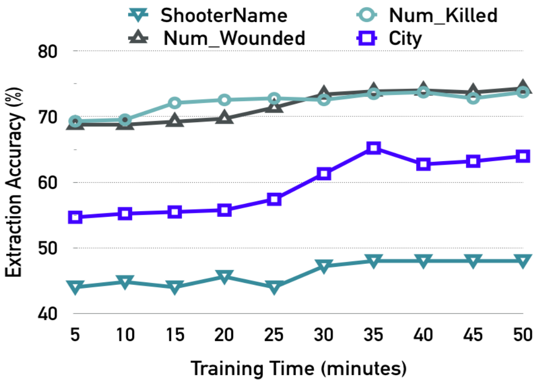


Figure 4: Evolution of accuracy of RLIE-A3C for the four relations on *test* set of the *Shooting Incidents* dataset. Please see Section 4.1 for details.

4.1 Results and Discussion

Extraction Accuracy: Experimental results comparing extraction accuracies of RLIE-DQN and RLIE-A3C are presented in Figure 3[3] (left and middle panels). From this figure, we observe that there is no loss in average accuracy in transitioning from RLIE-DQN to RLIE-A3C (in fact there is a slight gain in case of the Adulteration dataset). Please note that, improvement in accuracy is not our primary goal in the paper – it is the training time speedup, as discussed next.

Speedup: Training times of RLIE-DQN and RLIE-A3C over both datasets are compared in the right panel of Figure 3[4]. From Figure 3, we find that RLIE-A3C is able to achieve upto 6x training

[1] http://www.shootingtracker.com/Main_Page

[2] http://www.foodshield.org/member/login/

[3] The tolerance for all the relation accuracy values for RLIE-A3C is within ± 1%

[4] During test, policies learned by RLIE-DQN and RLIE-A3C are executed. Since this is an identical process for both, we don't compare test runtimes in this paper.

speedup over RLIE-DQN. In other words, RLIE-A3C is able to achieve significant speedup in training time over RLIE-DQN, without any loss in average accuracy. This is our main result.

While RLIE-DQN was implemented in Torch, RLIE-A3C was implemented in Python using TensorFlow framework. We note that Torch is known to be faster than TensorFlow (Bahrampour et al., 2015). This makes the speedup gains above even more impressive, and outlines the possibility that further gains may be possible with a Torch-based implementation of RLIE-A3C.

Figure 4 shows evolution of test accuracy of RLIE-A3C for the four relations of the *Shooting Incidents* dataset. For this dataset, state value function of RLIE-A3C converged at 48 minutes. However, from Figure 4, we observe that accuracies converge to the final values much before that.

Why do Asynchronous Methods work? An asynchronous approach fits more naturally with the training data, since different parallel agents look at different events at the same time (see Figure 1), and the model is able to exploit the regularities between events in the dataset. The gradient updates to the global network are less biased and the model is not easily distracted by noise in the data. The asynchronous parallel model is able to converge much faster as compared to replay memory based methods like DQN, as also seen in (Mnih et al., 2016).

5 Conclusion

In this paper, we proposed RLIE-A3C, an asynchronous deep Reinforcement Learning (RL) algorithm for Information Extraction (IE). In contrast to *sequential* training in previously proposed RLIE-DQN (Narasimhan et al., 2016), RLIE-A3C employs *asynchronous parallel* training. This results in upto 6x training speedup, without suffering any loss in average accuracy. We hope that this first application of asynchronous deep RL algorithms will open up more adoption of such techniques in the NLP community.

6 Acknowledgements

We thank members of the MALL Lab, IISc who read drafts of the paper and gave valuable feedback. We also thank the anonymous reviewers for their insightful comments. Special thanks to Karthik Narasimhan for all the helpful discussions. We also gratefully acknowledge support from a gift from Microsoft Research India.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8. Association for Computational Linguistics.

Soheil Bahrampour, Naveen Ramakrishnan, Lukas Schott, and Mohak Shah. 2015. Comparative study of deep learning software frameworks. *arXiv preprint arXiv:1511.06435*.

Lidong Bing, Zhiming Zhang, Wai Lam, and William W Cohen. 2017. Towards a language-independent solution: Knowledge base completion by searching the web and deriving language pattern. *Knowledge-Based Systems*, 115:80–86.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*.

Manjunath Hegde and Partha P Talukdar. 2015. An entity-centric approach for overcoming knowledge graph sparsity. In *EMNLP*, pages 530–535.

Pallika H Kanani and Andrew K McCallum. 2012. Selecting actions for resource-bounded information extraction using reinforcement learning. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 253–262. ACM.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. 2016. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. *arXiv preprint arXiv:1603.07954*.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2).

Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8(3-4):279–292.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*, pages 515–526. ACM.

Zhenzhong Zhang, Le Sun, and Xianpei Han. 2016. A joint model for entity set expansion and attribute extraction from web search queries. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3101–3107. AAAI Press.

Leveraging Linguistic Structures for Named Entity Recognition with Bidirectional Recursive Neural Networks

Peng-Hsuan Li
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd.
Taipei 10617, Taiwan
jacobvsdanniel@gmail.com

Ruo-Ping Dong
National Tsing Hua University
No. 101, Sec. 2, Kuang-Fu Rd.
Hsinchu 30013, Taiwan
dongruoping@gmail.com

Yu-Siang Wang
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd.
Taipei 10617, Taiwan
b03202047@ntu.edu.tw

Ju-Chieh Chou
National Taiwan University
No. 1, Sec. 4, Roosevelt Rd.
Taipei 10617, Taiwan
jjery2243542@gmail.com

Wei-Yun Ma
Academia Sinica
No. 128, Sec. 2, Academia Rd.
Taipei 11529, Taiwan
ma@iis.sinica.edu.tw

Abstract

In this paper, we utilize the linguistic structures of texts to improve named entity recognition by BRNN-CNN, a special bidirectional recursive network attached with a convolutional network. Motivated by the observation that named entities are highly related to linguistic constituents, we propose a constituent-based BRNN-CNN for named entity recognition. In contrast to classical sequential labeling methods, the system first identifies which text chunks are possible named entities by whether they are linguistic constituents. Then it classifies these chunks with a constituency tree structure by recursively propagating syntactic and semantic information to each constituent node. This method surpasses current state-of-the-art on OntoNotes 5.0 with automatically generated parses.

1 Introduction

Named Entity Recognition (NER) can be seen as a combined task of locating named entity chunks of texts and classifying which named entity category a chunk falls into. Traditional approaches label each token in texts as a part of a named entity chunk, e.g. "person_begin", and achieve high performances in several benchmark datasets (Ratinov and Roth, 2009; Passos et al., 2014; Chiu and Nichols, 2016).

Being formulated as a sequential labeling problem, NER systems could be naturally implemented by recurrent neural networks. These networks process a token at a time, taking, for each token, the hidden features of its previous token as well as its raw features to compute its own hidden features. Then they classify each token by these hidden features. With both forward and backward directions, networks learn how to propagate the information of a token sequence to each token. Chiu and Nichols (2016) utilize a variation of recurrent networks, bidirectional LSTM, attached with a CNN, which learns character-level features instead of handcrafting. They accomplish state-of-the-art results on both CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003) and OntoNotes 5.0 (Hovy et al., 2006; Pradhan et al., 2013) datasets.

Classical sequential labeling approaches take little information about phrase structures of sentences. However, according to our analysis, most named entity chunks are actually linguistic constituents, e.g. noun phrases. This motivates us to focus on a constituent-based approach for NER where the NER problem is transformed into a named entity classification task on every node of a

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2664–2669
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Algorithm 1 Binarization

```
function BINARIZE($node$)
    $n \leftarrow node.children.length$
    if $n > 2$ then
        if HEAD-FINDER($node$) $\neq node.children[n]$ then
            $newChild \leftarrow$ GROUP($node.children[1..n\text{-}1]$)
            $node.children \leftarrow [newChild, node.children[n]]$
        else
            $newChild \leftarrow$ GROUP($node.children[2..n]$)
            $node.children \leftarrow [node.children[1], newChild]$
        $newChild.pos \leftarrow node.pos$
    for $child$ in $node.children$ do
        BINARIZE($child$)
```

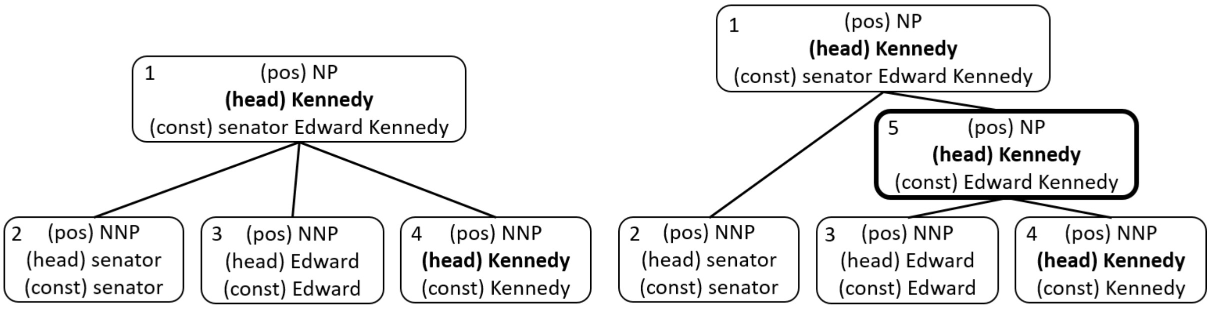


Figure 1: Applying Algorithm 1 to the parse of *senator Edward Kennedy*.

constituency structure.

To classify constituents and take into account their structures, we propose BRNN-CNN, a special bidirectional recursive neural network attached with a convolutional network. For each sentence, a constituency parse where every node represents a meaningful chunk of the sentence, i.e. a constituent, is first generated. Then BRNN-CNN recursively computes hidden state features of every node and classifies each node by these hidden features. To capture structural linguistic information, bidirectional passes are applied so that each constituent sees what it is composed of as well as what is containing it, both in a near-to-far fashion.

Our main contribution is the introduction of a novel constituent-based BRNN-CNN for named entity recognition, which successfully utilizes the linguistic structures of texts by recursive neural networks. We show that it achieves better scores than current state-of-the-art on OntoNotes 5.0, where good parses can be automatically generated. Additionally, we analyze the effects of only considering constituents and the effects of constituency parses.

2 Related Work

Collobert et al. (2011) achieved near state-of-the-art performance on CoNLL-2003 NER with an end-to-end neural network which had minimal feature engineering and external data. Chiu and Nichols (2016) achieved the current state-of-the-art on both CoNLL-2003 and OntoNotes 5.0 NER with a sequential bidirectional LSTM-CNN. They also did extensive studies of additional features such as character type, capitalization, and Senna and DBpedia lexicons.

Finkel and Manning (2009) explored training a parser for an NER-suffixed grammar, jointly tackling parsing and NER. They achieved competitive results on OntoNotes with a CRF-CFG parser.

Recursive neural networks have been successfully applied for parsing and sentiment analysis on Stanford sentiment treebank (Socher et al., 2010, 2013a,b; Tai et al., 2015). Their recursive networks, such as RNTN and Tree-LSTM, do sentiment combinations on phrase structures in a bottom-up fashion, showing the potential of such models in computing semantic compositions.

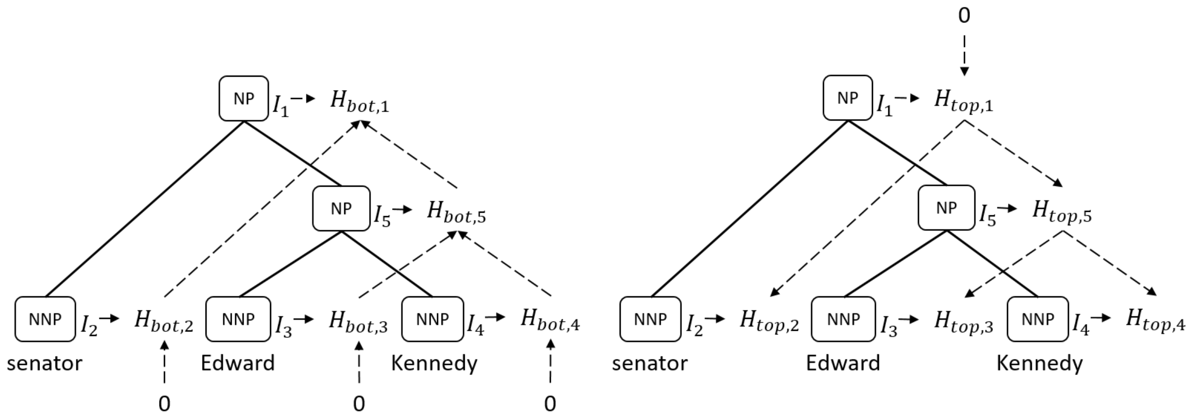


Figure 2: The bottom-up and top-down hidden layers applied to the binarized tree in Figure 1.

3 Method

For each input sentence, our constituent-based BRNN-CNN first extracts features from its constituency parse, then recursively classifies each constituent, and finally resolves conflicting predictions.

3.1 Preparing Constituency Structures

For a sentence and its associated constituency parse, our system first sets three features for each node: a POS, a word, and a head. While constituency tags and words should come readily, semantic head words are determined by a rule-based head finder (Collins, 1999). Additionally, a fourth feature vector is added to each node to utilize lexicon knowledge. The 3-bit vector records if the constituent of a node matches some phrases in each of the three SENNA (Collobert et al., 2011) lexicons of persons, organizations, and locations.

The system then tries to generate more plausible constituents while preserving linguistic structures by applying a binarization process which groups excessive child nodes around the head children. The heuristic is that a head constituent is usually modified by its siblings in a near to far fashion. Algorithm 1 shows the recursive procedure called for the root node of a parse. Figure 1 shows the application of the algorithm to the parse of *senator Edward Kennedy*. With the heuristic that *Edward* modifies the head node *Kennedy* before *senator*. The binarization process successfully adds a new node *Edward Kennedy* that corresponds to a person name.

3.2 Computing Word Embeddings

For each word, our network retrieves one embedding from a trainable lookup table initialized by GloVe (Pennington et al., 2014). However, to capture the morphology information of a word and help dealing with unseen words, the network computes another character-level embedding. Inspired by Kim et al. (2016), the network passes one-hot character vectors through a series of convolutional and highway layers to generate the embedding. These two embeddings are concatenated as the final embedding of a word.

3.3 Computing Hidden Features

Given a constituency parse tree, where every node represents a constituent, our network recursively computes two hidden state features for every node.

First, for each node i with left sibling l and right sibling r, the raw feature vector I_i is formed by concatenating the one-hot POS vectors of i, l, r, the head embeddings of i, l, r, the word embedding of i, the lexicon vector of i, and the mean of word embeddings in the sentence. Then, with the the set of child nodes C and the parent node p, the hidden feature vectors $H_{bot,i}$ and $H_{top,i}$ are computed by 2 hidden layers:

$$H_{bot,i} = ReLU((I_i \| \sum_{c \in C} H_{bot,c}) W_{bot} + b_{bot}) \quad (1)$$

$$H_{top,i} = ReLU((I_i \| H_{top,p}) W_{top} + b_{top}) \quad (2)$$

where Ws are weight matrices, bs are bias vectors, and $ReLU(x) = max(0, x)$. In cases when some needed neighboring nodes do not exist, or when i is a nonterminal and does not have a word, zero

		Validation			**Test**		
Model	Parser	Precision	Recall	F1	Precision	Recall	F1
BRNN-CNN	gold	86.6	87.0	86.77	88.9	88.9	88.92
BRNN	gold	87.5	86.7	87.11	89.5	88.3	88.91
BRNN-CNN	auto	85.5	84.7	85.08	88.0	86.5	**87.21**
BRNN	auto	86.0	84.7	85.34	88.0	86.2	87.10
Bidirectional Tree-LSTM	auto	85.2	84.5	84.84	87.3	86.2	86.74
Sequential Recurrent NN	-	83.1	83.7	83.38	84.5	84.4	84.40
Finkel and Manning (2009)	gold	-	-	-	84.04	80.86	82.42
Durrett and Klein (2014)	-	-	-	-	85.22	82.89	84.04
Chiu and Nichols (2016)	-	-	-	-	-	-	86.41

Table 1: Experiment results on whole dataset. BRNN is BRNN-CNN deprived of character-level embeddings. Human-labeled parses and automatically generated parses are indicated by gold and auto respectively. Finkel and Manning used gold parses in training a joint model for parsing and NER.

Model	**BC**	**BN**	**MZ**	**NW**	**TC**	**WB**
Test set size (# tokens)	32488	23209	17875	49235	10976	18945
Test set size (# entities)	1697	2184	1163	4696	380	1137
Finkel and Manning (2009)	78.66	87.29	82.45	85.50	67.27	72.56
Durrett and Klein (2014)	78.88	87.39	82.46	87.60	72.68	76.17
Chiu and Nichols (2016)	85.23	89.93	84.45	88.39	72.39	78.38
BRNN-CNN-auto	**85.98**	**90.96**	**84.93**	**89.18**	**73.18**	**80.39**

Table 2: F1 scores on different data sources. From left to right: broadcast conversation, broadcast news, magazine, newswire, telephone conversation, and blogs & newsgroups.

vectors are used as the missing parts of raw or hidden features.

Figure 2 shows the applications of the equations to the binarized tree in Figure 1. The computations are done recursively in two directions. The bottom-up direction computes the semantic composition of the subtree of each node, and the top-down counterpart propagates to that node the linguistic structures which contain the subtree. Together, hidden features of a constituent capture its structural linguistic information.

In addition, each hidden layer can be extended to a deep hidden network. For example, a 2-layer top-down hidden network is given by

$$H_{t\alpha,i} = ReLU((I_i \| H_{t\alpha,p})W_{t\alpha} + b_{t\alpha})$$

$$H_{t\beta,i} = ReLU((H_{t\alpha,i} \| H_{t\beta,p})W_{t\beta} + b_{t\beta})$$

where $t\alpha$ represents the first top-down hidden layer and $t\beta$ represents the second. Our best model is tuned to have 3 layers for both directions.

3.4 Forming Consistent Predictions

Given hidden features for every node, our network computes a probability distribution of named entity classes plus a special non-entity class by an output layer. For each node i with left sibling l and right sibling r, the probability distribution O_i is computed by an output layer:

$$O_i = \sigma((H_i \| H_l \| H_r)W_{out} + b_{out}) \qquad (3)$$

where $H_x = H_{bot,x} + H_{top,x}$, $x \in \{i, l, r\}$, and $\sigma(x) = (1 + e^{-x})^{-1}$. If a sibling does not exist, zero vectors are used as its hidden states. Should deep hidden layers be deployed, the last hidden layer is used.

Finally, the system makes predictions for a sentence by collecting the constituents whose most probable classes are named entity classes. However, nodes whose ancestors are already predicted as named entities are ignored to prevent predicting overlapping named entities.

4 Evaluation

We evaluate our system on OntoNotes 5.0 NER (Hovy et al., 2006; Pradhan et al., 2013) and analyze it with several ablation studies. The project sources are publicly available on `https://github.com/jacobvsdanniel/tf_rnn`.

Split	Token	NE	Constituent
Train	1,088,503	81,828	93.3 → 97.3
Validate	147,724	11,066	92.8 → 97.0
Test	152,728	11,257	92.9 → 97.2

Table 3: Dataset statistics for OntoNotes 5.0.

4.1 Training and Tuning

To train the model, we minimize the cross entropy loss of the softmax class probabilities in Equation 3 by the Adam optimizer (Kingma and Ba, 2014). Other details such as hyperparameters are documented in the supplemental materials as well as the public repository.

4.2 OntoNotes 5.0 NER

OntoNotes 5.0 annotates 18 types of named entities for diverse sources of texts. Like other previous work (Durrett and Klein, 2014; Chiu and Nichols, 2016), we use the format and the train-validate-test split provided by CoNLL-2012. In addition, both gold and auto parses are available.

Table 3 shows the dataset statistics. The last column shows the percentages of named entities that correspond to constituents of auto parses before and after binarization.

Table 1 and Table 2 compare our results with others on the whole dataset and different sources respectively. The sample mean, standard deviation, and sample count of BRNN-auto and Chiu and Nichols' model are 87.10, 0.14, 3 and 86.41, 0.22, 10 respectively. By one-tailed Welch's T-test, the former significantly surpasses the latter with 99% confidence level (0.000489 p-value).

4.3 Analysis of the Approach

The training and validation sets contain 1,236,227 tokens and 92,894 named entities, of which 90,371 correspond to some constituents of binarized auto parses. This backs our motivation that more than 97% named entities are linguistic constituents, and 52,729 of them are noun phrases.

Essentially, the constituent-based approach filters out the other 3% named entities that cross constituent boundaries (Figure 3), i.e. 3% loss of recall. We dig into this problem by analyzing a sequential labeling recurrent network (the sixth model in Table 1). The simple model performs reasonably well, but its non-constituent predictions are mostly false positive. In fact, it slightly improves if all non-constituent predictions are removed in post-processing, i.e., the precision gain of focusing on constituents is more significant than the recall loss. This is one advantage of our system over other sequential models, which try to learn and predict non-constituent named entities but do not perform well.

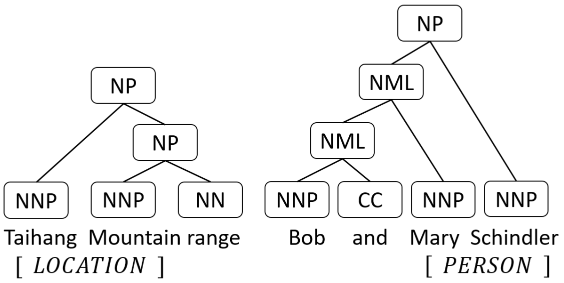


Figure 3: Two sample named entities that cross different branches of syntax parses.

In addition, to analyze the effects of constituency structures, we test our models with different qualities of parses (gold vs. auto in Table 1). The significant F1 differences suggest that structural linguistic information is crucial and can be learned by our model.

5 Conclusion

We have demonstrated a novel constituent-based BRNN-CNN for named entity recognition which successfully utilizes constituency structures and surpasses the current state-of-the-art on OntoNotes 5.0 NER. Instead of propagating information by word orders as normal recurrent networks, the model is able to recursively propagate structural linguistic information to every constituent. Experiments show that when a good parser is available, the approach will be a good alternative to traditional sequential labeling token-based NER.

Named entities that cross constituent boundaries are analyzed and we find out that a naïve sequential labeling model has difficulty predicting them without too many false positives. While avoiding them is one of the strengths of our model, generating more consistent parses to reduce this kind of named entities would be one possible direction for future research.

Acknowledgments

This work is supported by the 2016 Summer Internship Program of IIS, Academia Sinica.

References

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

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

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

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334. Association for Computational Linguistics.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs.LG]*.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of the Eighteenth Conference on Computational Language Learning*, pages 78–86.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 147–155, Stroudsburg, PA, USA. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013a. Parsing with compositional vector grammars. In *Proceedings of the ACL conference*.

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

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

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processingg*, pages 1556–1566.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Fast and Accurate Entity Recognition with Iterated Dilated Convolutions

Emma Strubell **Patrick Verga** **David Belanger** **Andrew McCallum**
College of Information and Computer Sciences
University of Massachusetts Amherst
{strubell, pat, belanger, mccallum}@cs.umass.edu

Abstract

Today when many practitioners run basic NLP on the entire web and large-volume traffic, faster methods are paramount to saving time and energy costs. Recent advances in GPU hardware have led to the emergence of bi-directional LSTMs as a standard method for obtaining per-token vector representations serving as input to labeling tasks such as NER (often followed by prediction in a linear-chain CRF). Though expressive and accurate, these models fail to fully exploit GPU parallelism, limiting their computational efficiency. This paper proposes a faster alternative to Bi-LSTMs for NER: Iterated Dilated Convolutional Neural Networks (ID-CNNs), which have better capacity than traditional CNNs for large context and structured prediction. Unlike LSTMs whose sequential processing on sentences of length N requires $O(N)$ time even in the face of parallelism, ID-CNNs permit fixed-depth convolutions to run in parallel across entire documents. We describe a distinct combination of network structure, parameter sharing and training procedures that enable dramatic 14-20x test-time speedups while retaining accuracy comparable to the Bi-LSTM-CRF. Moreover, ID-CNNs trained to aggregate context from the entire document are even more accurate while maintaining 8x faster test time speeds.

1 Introduction

In order to democratize large-scale NLP and information extraction while minimizing our environmental footprint, we require fast, resource-efficient methods for sequence tagging tasks such as part-of-speech tagging and named entity recognition (NER). Speed is not sufficient of course: they must also be expressive enough to tolerate the tremendous lexical variation in input data.

The massively parallel computation facilitated by GPU hardware has led to a surge of successful neural network architectures for sequence labeling (Ling et al., 2015; Ma and Hovy, 2016; Chiu and Nichols, 2016; Lample et al., 2016). While these models are expressive and accurate, they fail to fully exploit the parallelism opportunities of a GPU, and thus their speed is limited. Specifically, they employ either recurrent neural networks (RNNs) for feature extraction, or Viterbi inference in a structured output model, both of which require sequential computation across the length of the input.

Instead, parallelized runtime independent of the length of the sequence saves time and energy costs, maximizing GPU resource usage and minimizing the amount of time it takes to train and evaluate models. Convolutional neural networks (CNNs) provide exactly this property (Kim, 2014; Kalchbrenner et al., 2014). Rather than composing representations incrementally over each token in a sequence, they apply filters in parallel across the entire sequence at once. Their computational cost grows with the number of layers, but not the input size, up to the memory and threading limitations of the hardware. This provides, for example, audio generation models that can be trained in parallel (van den Oord et al., 2016).

Despite the clear computational advantages of CNNs, RNNs have become the standard method for composing deep representations of text. This is because a token encoded by a bidirectional RNN will incorporate evidence from the entire input sequence, but the CNN's representation is limited by

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2670–2680
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

the effective input width[1] of the network: the size of the input context which is observed, directly or indirectly, by the representation of a token at a given layer in the network. Specifically, in a network composed of a series of stacked convolutional layers of convolution width w, the number r of context tokens incorporated into a token's representation at a given layer l, is given by $r = l(w-1)+1$. The number of layers required to incorporate the entire input context grows linearly with the length of the sequence. To avoid this scaling, one could pool representations across the sequence, but this is not appropriate for sequence labeling, since it reduces the output resolution of the representation.

In response, this paper presents an application of *dilated convolutions* (Yu and Koltun, 2016) for sequence labeling (Figure 1). For dilated convolutions, the effective input width can grow exponentially with the depth, with no loss in resolution at each layer and with a modest number of parameters to estimate. Like typical CNN layers, dilated convolutions operate on a sliding window of context over the sequence, but unlike conventional convolutions, the context need not be consecutive; the dilated window skips over every dilation width d inputs. By stacking layers of dilated convolutions of exponentially increasing dilation width, we can expand the size of the effective input width to cover the entire length of most sequences using only a few layers: The size of the effective input width for a token at layer l is now given by $2^{l+1}-1$. More concretely, just four stacked dilated convolutions of width 3 produces token representations with a n effective input width of 31 tokens – longer than the average sentence length (23) in the Penn TreeBank.

Our overall *iterated dilated CNN* architecture (ID-CNN) repeatedly applies the same block of dilated convolutions to token-wise representations. This parameter sharing prevents overfitting and also provides opportunities to inject supervision on intermediate activations of the network. Similar to models that use logits produced by an RNN, the ID-CNN provides two methods for performing prediction: we can predict each token's label independently, or by running Viterbi inference in a chain structured graphical model.

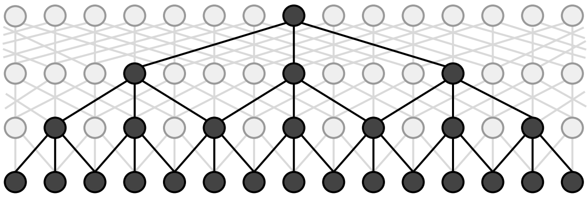


Figure 1: A dilated CNN block with maximum dilation width 4 and filter width 3. Neurons contributing to a single highlighted neuron in the last layer are also highlighted.

In experiments on CoNLL 2003 and OntoNotes 5.0 English NER, we demonstrate significant speed gains of our ID-CNNs over various recurrent models, while maintaining similar F1 performance. When performing prediction using independent classification, the ID-CNN consistently outperforms a bidirectional LSTM (Bi-LSTM), and performs on par with inference in a CRF with logits from a Bi-LSTM (Bi-LSTM-CRF). As an extractor of per-token logits for a CRF, our model out-performs the Bi-LSTM-CRF. We also apply ID-CNNs to entire documents, where independent token classification is as accurate as the Bi-LSTM-CRF while decoding almost 8× faster. The clear accuracy gains resulting from incorporating broader context suggest that these models could similarly benefit many other context-sensitive NLP tasks which have until now been limited by the computational complexity of existing context-rich models.[2]

2 Background

2.1 Conditional Probability Models for Tagging

Let $x = [x_1, \ldots, x_T]$ be our input text and $y = [y_1, \ldots, y_T]$ be per-token output tags. Let D be the domain size of each y_i. We predict the most likely y, given a conditional model $P(y|x)$.

This paper considers two factorizations of the conditional distribution. First, we have

$$P(y|x) = \prod_{t=1}^{T} P(y_t|F(x)), \qquad (1)$$

where the tags are conditionally independent given some features for x. Given these features, $O(D)$ prediction is simple and parallelizable across the

[1] What we call *effective input width* here is known as the *receptive field* in the vision literature, drawing an analogy to the visual receptive field of a neuron in the retina.

[2] Our implementation in TensorFlow (Abadi et al., 2015) is available at: https://github.com/iesl/dilated-cnn-ner

length of the sequence. However, feature extraction may not necessarily be parallelizable. For example, RNN-based features require iterative passes along the length of x.

We also consider a linear-chain CRF model that couples all of y together:

$$P(y|x) = \frac{1}{Z_x} \prod_{t=1}^{T} \psi_t(y_t|F(x))\psi_p(y_t, y_{t-1}), \quad (2)$$

where ψ_t is a local factor, ψ_p is a pairwise factor that scores consecutive tags, and Z_x is the partition function (Lafferty et al., 2001). To avoid overfitting, ψ_p does not depend on the timestep t or the input x in our experiments. Prediction in this model requires global search using the $O(D^2T)$ Viterbi algorithm.

CRF prediction explicitly reasons about interactions among neighboring output tags, whereas prediction in the first model compiles this reasoning into the feature extraction step (Liang et al., 2008). The suitability of such compilation depends on the properties and quantity of the data. While CRF prediction requires non-trivial search in output space, it can guarantee that certain output constraints, such as for IOB tagging (Ramshaw and Marcus, 1999), will always be satisfied. It may also have better sample complexity, as it imposes more prior knowledge about the structure of the interactions among the tags (London et al., 2016). However, it has worse computational complexity than independent prediction.

3 Dilated Convolutions

CNNs in NLP are typically one-dimensional, applied to a sequence of vectors representing tokens rather than to a two-dimensional grid of vectors representing pixels. In this setting, a convolutional neural network layer is equivalent to applying an affine transformation, W_c to a sliding window of width r tokens on either side of each token in the sequence. Here, and throughout the paper, we do not explicitly write the bias terms in affine transformations. The convolutional operator applied to each token x_t with output c_t is defined as:

$$c_t = W_c \bigoplus_{k=0}^{r} x_{t \pm k}, \quad (3)$$

where $\oplus$ is vector concatenation.

Dilated convolutions perform the same operation, except rather than transforming adjacent inputs, the convolution is defined over a wider effective input width by skipping over δ inputs at a time, where δ is the dilation width. We define the dilated convolution operator:

$$c_t = W_c \bigoplus_{k=0}^{r} x_{t \pm k\delta}. \quad (4)$$

A dilated convolution of width 1 is equivalent to a simple convolution. Using the same number of parameters as a simple convolution with the same radius (i.e. W_c has the same dimensionality), the $\delta > 1$ dilated convolution incorporates broader context into the representation of a token than a simple convolution.

3.1 Multi-Scale Context Aggregation

We can leverage the ability of dilated convolutions to incorporate global context without losing important local information by stacking dilated convolutions of increasing width. First described for pixel classification in computer vision, Yu and Koltun (2016) achieve state-of-the-art results on image segmentation benchmarks by stacking dilated convolutions with exponentially increasing rates of dilation, a technique they refer to as *multi-scale context aggregation*. By feeding the outputs of each dilated convolution as the input to the next, increasingly non-local information is incorporated into each pixel's representation. Performing a dilation-1 convolution in the first layer ensures that no pixels within the effective input width of any pixel are excluded. By doubling the dilation width at each layer, the size of the effective input width grows exponentially while the number of parameters grows only linearly with the number of layers, so a pixel representation quickly incorporates rich global evidence from the entire image.

4 Iterated Dilated CNNs

Stacked dilated CNNs can easily incorporate global information from a whole sentence or document. For example, with a radius of 1 and 4 layers of dilated convolutions, the effective input width of each token is width 31, which exceeds the average sentence length (23) in the Penn TreeBank corpus. With a radius of size 2 and 8 layers of dilated convolutions, the effective input width exceeds 1,000 tokens, long enough to encode a full newswire document.

Unfortunately, simply increasing the depth of stacked dilated CNNs causes considerable over-fitting in our experiments. In response, we present Iterated Dilated CNNs (ID-CNNs), which instead apply the same small stack of dilated convolutions multiple times, each iterate taking as input the result of the last application. Repeatedly employing the same parameters in a recurrent fashion provides both broad effective input width and desirable generalization capabilities. We also obtain significant accuracy gains with a training objective that strives for accurate labeling after each iterate, allowing follow-on iterations to observe and resolve dependency violations.

4.1 Model Architecture

The network takes as input a sequence of T vectors $\mathbf{x_t}$, and outputs a sequence of per-class scores $\mathbf{h_t}$, which serve either as the local conditional distributions of Eqn. (1) or the local factors ψ_t of Eqn. (2).

We denote the jth dilated convolutional layer of dilation width δ as $D_\delta^{(j)}$. The first layer in the network is a dilation-1 convolution $D_1^{(0)}$ that transforms the input to a representation $\mathbf{i_t}$:

$$\mathbf{i_t} = D_1^{(0)}\mathbf{x_t} \tag{5}$$

Next, L_c layers of dilated convolutions of exponentially increasing dilation width are applied to $\mathbf{i_t}$, folding in increasingly broader context into the embedded representation of $\mathbf{x_t}$ at each layer. Let $r()$ denote the ReLU activation function (Glorot et al., 2011). Beginning with $\mathbf{c_t}^{(0)} = \mathbf{i_t}$ we define the stack of layers with the following recurrence:

$$\mathbf{c_t}^{(j)} = r\left(D_{2^{L_c-1}}^{(j-1)}\mathbf{c_t}^{(j-1)}\right) \tag{6}$$

and add a final dilation-1 layer to the stack:

$$\mathbf{c_t}^{(L_c+1)} = r\left(D_1^{(L_c)}\mathbf{c_t}^{(L_c)}\right) \tag{7}$$

We refer to this stack of dilated convolutions as a *block* $B(\cdot)$, which has output resolution equal to its input resolution. To incorporate even broader context without over-fitting, we avoid making B deeper, and instead iteratively apply B L_b times, introducing no extra parameters. Starting with $\mathbf{b_t}^{(1)} = B(\mathbf{i_t})$:

$$\mathbf{b_t}^{(k)} = B\left(\mathbf{b_t}^{(k-1)}\right) \tag{8}$$

We apply a simple affine transformation W_o to this final representation to obtain per-class scores for each token $\mathbf{x_t}$:

$$\mathbf{h_t}^{(L_b)} = W_o\mathbf{b_t}^{(L_b)} \tag{9}$$

4.2 Training

Our main focus is to apply the ID-CNN an encoder to produce per-token logits for the first conditional model described in Sec. 2.1, where tags are conditionally independent given deep features, since this will enable prediction that is parallelizable across the length of the input sequence. Here, maximum likelihood training is straightforward because the likelihood decouples into the sum of the likelihoods of independent logistic regression problems for every tag, with natural parameters given by Eqn. (9):

$$\frac{1}{T}\sum_{t=1}^{T}\log P(y_t \mid \mathbf{h_t}^{(L_b)}) \tag{10}$$

We can also use the ID-CNN as logits for the CRF model (Eqn. (2)), where the partition function and its gradient are computed using the forward-backward algorithm.

We next present an alternative training method that helps bridge the gap between these two techniques. Sec. 2.1 identifies that the CRF has preferable sample complexity and accuracy since prediction directly reasons in the space of structured outputs. In response, we compile some of this reasoning in output space into ID-CNN feature extraction. Instead of explicit reasoning over output labels during inference, we train the network such that each block is predictive of output labels. Subsequent blocks learn to correct dependency violations of their predecessors, refining the final sequence prediction.

To do so, we first define predictions of the model after each of the L_b applications of the block. Let $\mathbf{h_t}^{(k)}$ be the result of applying the matrix W_o from (9) to $\mathbf{b_t}^{(k)}$, the output of block k. We minimize the average of the losses for each application of the block:

$$\frac{1}{L_b}\sum_{k=1}^{L_b}\frac{1}{T}\sum_{t=1}^{T}\log P(y_t \mid \mathbf{h_t}^{(k)}). \tag{11}$$

By rewarding accurate predictions after each application of the block, we learn a model where later blocks are used to refine initial predictions.

The loss also helps reduce the vanishing gradient problem (Hochreiter, 1998) for deep architectures. Such an approach has been applied in a variety of contexts for training very deep networks in computer vision (Romero et al., 2014; Szegedy et al., 2015; Lee et al., 2015; Gülçehre and Bengio, 2016), but not to our knowledge in NLP.

We apply dropout (Srivastava et al., 2014) to the raw inputs $\mathbf{x_t}$ and to each block's output $\mathbf{b_t}^{(b)}$ to help prevent overfitting. The version of dropout typically used in practice has the undesirable property that the randomized predictor used at train time differs from the fixed one used at test time. Ma et al. (2017) present *dropout with expectation-linear regularization*, which explicitly regularizes these two predictors to behave similarly. All of our best reported results include such regularization. This is the first investigation of the technique's effectiveness for NLP, including for RNNs. We encourage its further application.

5 Related work

The state-of-the art models for sequence labeling include an inference step that searches the space of possible output sequences of a chain-structured graphical model, or approximates this search with a beam (Collobert et al., 2011; Weiss et al., 2015; Lample et al., 2016; Ma and Hovy, 2016; Chiu and Nichols, 2016). These outperform similar systems that use the same features, but independent local predictions. On the other hand, the greedy *sequential prediction* (Daumé III et al., 2009) approach of Ratinov and Roth (2009), which employs lexicalized features, gazetteers, and word clusters, outperforms CRFs with similar features.

LSTMs (Hochreiter and Schmidhuber, 1997) were used for NER as early as the CoNLL shared task in 2003 (Hammerton, 2003; Tjong Kim Sang and De Meulder, 2003). More recently, a wide variety of neural network architectures for NER have been proposed. Collobert et al. (2011) employ a one-layer CNN with pre-trained word embeddings, capitalization and lexicon features, and CRF-based prediction. Huang et al. (2015) achieved state-of-the-art accuracy on part-of-speech, chunking and NER using a Bi-LSTM-CRF. Lample et al. (2016) proposed two models which incorporated Bi-LSTM-composed character embeddings alongside words: a Bi-LSTM-CRF, and a greedy stack LSTM which uses a simple shift-reduce grammar to compose words into labeled entities. Their Bi-LSTM-CRF obtained the state-of-the-art on four languages without word shape or lexicon features. Ma and Hovy (2016) use CNNs rather than LSTMs to compose characters in a Bi-LSTM-CRF, achieving state-of-the-art performance on part-of-speech tagging and CoNLL NER without lexicons. Chiu and Nichols (2016) evaluate a similar network but propose a novel method for encoding lexicon matches, presenting results on CoNLL and OntoNotes NER. Yang et al. (2016) use GRU-CRFs with GRU-composed character embeddings of words to train a single network on many tasks and languages.

In general, distributed representations for text can provide useful generalization capabilities for NER systems, since they can leverage unsupervised pre-training of distributed word representations (Turian et al., 2010; Collobert et al., 2011; Passos et al., 2014). Though our models would also likely benefit from additional features such as character representations and lexicons, we focus on simpler models which use word-embeddings alone, leaving more elaborate input representations to future work.

In these NER approaches, CNNs were used for low-level feature extraction that feeds into alternative architectures. Overall, end-to-end CNNs have mainly been used in NLP for sentence classification, where the output representation is lower resolution than that of the input Kim (2014); Kalchbrenner et al. (2014); Zhang et al. (2015); Toutanova et al. (2015). Lei et al. (2015) present a CNN variant where convolutions adaptively skip neighboring words. While the flexibility of this model is powerful, its adaptive behavior is not well-suited to GPU acceleration.

Our work draws on the use of dilated convolutions for image segmentation in the computer vision community (Yu and Koltun, 2016; Chen et al., 2015). Similar to our block, Yu and Koltun (2016) employ a *context-module* of stacked dilated convolutions of exponentially increasing dilation width. Dilated convolutions were recently applied to the task of speech generation (van den Oord et al., 2016), and concurrent with this work, Kalchbrenner et al. (2016) posted a pre-print describing the similar ByteNet network for machine translation that uses dilated convolutions in the encoder and decoder components. Our basic model architecture is similar to that of the ByteNet encoder, except that the inputs to our model are tokens and not bytes. Additionally, we present a novel loss and parameter sharing scheme to facilitate training models on much smaller datasets than those used by Kalchbrenner et al. (2016). We are the first to use dilated convolutions for sequence labeling.

The broad effective input width of the ID-CNN helps aggregate document-level context. Ratinov and Roth (2009) incorporate document context in their greedy model by adding features based on tagged entities within a large, fixed window of tokens. Prior work has also posed a structured model that couples predictions across the whole document (Bunescu and Mooney, 2004; Sutton and McCallum, 2004; Finkel et al., 2005).

6 Experimental Results

We describe experiments on two benchmark English named entity recognition datasets. On CoNLL-2003 English NER, our ID-CNN performs on par with a Bi-LSTM not only when used to produce per-token logits for structured inference, but the ID-CNN with greedy decoding also performs on-par with the Bi-LSTM-CRF while running at more than 14 times the speed. We also observe a performance boost in almost all models when broadening the context to incorporate entire documents, achieving an average F1 of 90.65 on CoNLL-2003, out-performing the sentence-level model while still decoding at nearly 8 times the speed of the Bi-LSTM-CRF.

6.1 Data and Evaluation

We evaluate using labeled data from the CoNLL-2003 shared task (Tjong Kim Sang and De Meulder, 2003) and OntoNotes 5.0 (Hovy et al., 2006; Pradhan et al., 2006). Following previous work, we use the same OntoNotes data split used for co-reference resolution in the CoNLL-2012 shared task (Pradhan et al., 2012). For both datasets, we convert the IOB boundary encoding to BILOU as previous work found this encoding to result in improved performance (Ratinov and Roth, 2009). As in previous work we evaluate the performance of our models using segment-level micro-averaged F1 score. Hyperparameters that resulted in the best performance on the validation set were selected via grid search. A more detailed description of the data, evaluation, optimization and data pre-processing can be found in the Appendix.

6.2 Baselines

We compare our **ID-CNN** against strong LSTM and CNN baselines: a **Bi-LSTM** with local decoding, and one with CRF decoding (**Bi-LSTM-CRF**). We also compare against a non-dilated CNN architecture with the same number of convolutional layers as our dilated network (**4-layer CNN**) and one with enough layers to incorporate an effective input width of the same size as that of the dilated network (**5-layer CNN**) to demonstrate that the dilated convolutions more effectively aggregate contextual information than simple convolutions (i.e. using fewer parameters). We also compare our document-level ID-CNNs to a baseline which does not share parameters between blocks (**noshare**) and one that computes loss only at the last block, rather than after every iterated block of dilated convolutions (**1-loss**).

We do not compare with deeper or more elaborate CNN architectures for a number of reasons: 1) Fast train and test performance are highly desirable for NLP practitioners, and deeper models require more computation time 2) more complicated models tend to over-fit on this relatively small dataset and 3) most accurate deep CNN architectures repeatedly up-sample and down-sample the inputs. We do not compare to stacked LSTMs for similar reasons — a single LSTM is already slower than a 4-layer CNN. Since our task is sequence labeling, we desire a model that maintains the token-level resolution of the input, making dilated convolutions an elegant solution.

6.3 CoNLL-2003 English NER

6.3.1 Sentence-level prediction

Table 1 lists F1 scores of models predicting with sentence-level context on CoNLL-2003. For models that we trained, we report F1 and standard deviation obtained by averaging over 10 random restarts. The Viterbi-decoding Bi-LSTM-CRF and ID-CNN-CRF and greedy ID-CNN obtain the highest average scores, with the ID-CNN-CRF outperforming the Bi-LSTM-CRF by 0.11 points of F1 on average, and the Bi-LSTM-CRF out-performing the greedy ID-CNN by 0.11 as well. Our greedy ID-CNN outperforms the Bi-LSTM and the 4-layer CNN, which uses the same number of parameters as the ID-CNN, and performs similarly to the 5-layer CNN which uses more parameters but covers the same effective input width. All CNN models out-perform the Bi-

Model	F1
Ratinov and Roth (2009)	86.82
Collobert et al. (2011)	86.96
Lample et al. (2016)	90.33
Bi-LSTM	89.34 ± 0.28
4-layer CNN	89.97 ± 0.20
5-layer CNN	90.23 ± 0.16
ID-CNN	90.32 ± 0.26
Collobert et al. (2011)	88.67
Passos et al. (2014)	90.05
Lample et al. (2016)	90.20
Bi-LSTM-CRF (re-impl)	90.43 ± 0.12
ID-CNN-CRF	**90.54 ± 0.18**

Table 1: F1 score of models observing sentence-level context. No models use character embeddings or lexicons. Top models are greedy, bottom models use Viterbi inference .

Model	Speed
Bi-LSTM-CRF	1×
Bi-LSTM	9.92×
ID-CNN-CRF	1.28×
5-layer CNN	12.38×
ID-CNN	14.10×

Table 2: Relative test-time speed of sentence models, using the fastest batch size for each model.[5]

Model	w/o DR	w/ DR
Bi-LSTM	88.89 ± 0.30	**89.34** ± 0.28
4-layer CNN	89.74 ± 0.23	**89.97** ± 0.20
5-layer CNN	89.93 ± 0.32	**90.23** ± 0.16
Bi-LSTM-CRF	90.01 ± 0.23	**90.43** ± 0.12
4-layer ID-CNN	89.65 ± 0.30	**90.32** ± 0.26

Table 3: Comparison of models trained with and without expectation-linear dropout regularization (DR). DR improves all models.

LSTM when paired with greedy decoding, suggesting that CNNs are better token encoders than Bi-LSTMs for independent logistic regression. When paired with Viterbi decoding, our ID-CNN performs on par with the Bi-LSTM, showing that the ID-CNN is also an effective token encoder for structured inference.

Our ID-CNN is not only a better token encoder than the Bi-LSTM but it is also faster. Table 2 lists relative decoding times on the CoNLL development set, compared to the Bi-LSTM-CRF. We report decoding times using the fastest batch size for each method.[3]

The ID-CNN model decodes nearly 50% faster than the Bi-LSTM. With Viterbi decoding, the gap closes somewhat but the ID-CNN-CRF still comes out ahead, about 30% faster than the Bi-LSTM-CRF. The most vast speed improvements come when comparing the greedy ID-CNN to the Bi-LSTM-CRF – our ID-CNN is more than 14 times faster than the Bi-LSTM-CRF at test time, with comparable accuracy. The 5-layer CNN, which observes the same effective input width as the ID-CNN but with more parameters, performs at about the same speed as the ID-CNN in our experiments. With a better implementation of dilated convolutions than currently included in TensorFlow, we would expect the ID-CNN to be notably faster than the 5-layer CNN.

We emphasize the importance of the dropout regularizer of Ma et al. (2017) in Table 3, where we observe increased F1 for every model trained with expectation-linear dropout regularization. Dropout is important for training neural network models that generalize well, especially on relatively small NLP datasets such as CoNLL-2003. We recommend this regularizer as a simple and helpful tool for practitioners training neural networks for NLP.

6.3.2 Document-level prediction

In Table 4 we show that adding document-level context improves every model on CoNLL-2003. Incorporating document-level context further improves our greedy ID-CNN model, attaining 90.65 average F1. We believe this model sees greater improvement with the addition of document-level context than the Bi-LSTM-CRF due to the ID-CNN learning a feature function better suited for representing broad context, in contrast with the Bi-LSTM which, though better than a simple RNN at encoding long memories of sequences, may reach its limit when provided with sequences more than 1,000 tokens long such as entire documents.

We also note that our combination of training objective (Eqn. 11) and tied parameters (Eqn.

[3] For each model, we tried batch sizes $b = 2^i$ with $i = 0...11$. At scale, speed should increase with batch size, as we could compose each batch of as many sentences of the same length as would fit in GPU memory, requiring no padding and giving CNNs and ID-CNNs even more of a speed advantage.

[5] Our ID-CNN could see up to 18× speed-up with a less naive implementation than is included in TensorFlow as of this writing.

Model	F1
4-layer ID-CNN (sent)	90.32 ± 0.26
Bi-LSTM-CRF (sent)	90.43 ± 0.12
4-layer CNN × 3	90.32 ± 0.32
5-layer CNN × 3	90.45 ± 0.21
Bi-LSTM	89.09 ± 0.19
Bi-LSTM-CRF	90.60 ± 0.19
ID-CNN	**90.65 ± 0.15**

Table 4: F1 score of models trained to predict document-at-a-time. Our greedy ID-CNN model performs as well as the Bi-LSTM-CRF.

Model	F1
ID-CNN noshare	89.81 ± 0.19
ID-CNN 1-loss	90.06 ± 0.19
ID-CNN	**90.65 ± 0.15**

Table 5: Comparing ID-CNNs with 1) backpropagating loss only from the final layer (**1-loss**) and 2) untied parameters across blocks (**noshare**)

8) more effectively learns to aggregate this broad context than a vanilla cross-entropy loss or deep CNN back-propagated from the final neural network layer. Table 5 compares models trained to incorporate entire document context using the document baselines described in Section 6.2.

In Table 6 we show that, in addition to being more accurate, our ID-CNN model is also much faster than the Bi-LSTM-CRF when incorporating context from entire documents, decoding at almost 8 times the speed. On these long sequences, it also tags at more than 4.5 times the speed of the greedy Bi-LSTM, demonstrative of the benefit of our ID-CNNs context-aggregating computation that does not depend on the length of the sequence.

6.4 OntoNotes 5.0 English NER

We observe similar patterns on OntoNotes as we do on CoNLL. Table 7 lists overall F1 scores of our models compared to those in the existing literature. The greedy Bi-LSTM out-performs the lexicalized greedy model of Ratinov and Roth (2009), and our ID-CNN out-performs the Bi-LSTM as well as the more complex model of Durrett and Klein (2014) which leverages the parallel co-reference annotation available in the OntoNotes corpus to predict named entities jointly with entity linking and co-reference. Our greedy model is out-performed by the Bi-LSTM-CRF reported in Chiu and Nichols (2016) as well as our own re-implementation, which appears to be the new state-of-the-art on this dataset.

Model	Speed
Bi-LSTM-CRF	1×
Bi-LSTM	4.60×
ID-CNN	7.96×

Table 6: Relative test-time speed of document models (fastest batch size for each model).

Model	F1	Speed
Ratinov and Roth (2009)[6]	83.45	
Durrett and Klein (2014)	84.04	
Chiu and Nichols (2016)	86.19 ± 0.25	
Bi-LSTM-CRF	86.99 ± 0.22	1×
Bi-LSTM-CRF-Doc	86.81 ± 0.18	1.32×
Bi-LSTM	83.76 ± 0.10	24.44×
ID-CNN-CRF (1 block)	86.84 ± 0.19	1.83×
ID-CNN-Doc (3 blocks)	85.76 ± 0.13	21.19×
ID-CNN (3 blocks)	85.27 ± 0.24	13.21×
ID-CNN (1 block)	84.28 ± 0.10	26.01×

Table 7: F1 score of sentence and document models on OntoNotes.

The gap between our greedy model and those using Viterbi decoding is wider than on CoNLL. We believe this is due to the more diverse set of entities in OntoNotes, which also tend to be much longer – the average length of a multi-token named entity segment in CoNLL is about one token shorter than in OntoNotes. These long entities benefit more from explicit structured constraints enforced in Viterbi decoding. Still, our ID-CNN outperforms all other greedy methods, achieving our goal of learning a better token encoder for structured prediction.

Incorporating greater context significantly boosts the score of our greedy model on OntoNotes, whereas the Bi-LSTM-CRF performs more poorly. In Table 7, we also list the F1 of our ID-CNN model and the Bi-LSTM-CRF model trained on entire document context. For the first time, we see the score decrease when more context is added to the Bi-LSTM-CRF model, though the ID-CNN, whose sentence model a lower score than that of the Bi-LSTM-CRF, sees an increase. We believe the decrease in the Bi-LSTM-CRF model occurs because of the

[6] Results as reported in Durrett and Klein (2014) as this data split did not exist at the time of publication.

nature of the OntoNotes dataset compared to CoNLL-2003: CoNLL-2003 contains a particularly high proportion of ambiguous entities,[7] perhaps leading to more benefit from document context that helps with disambiguation. In this scenario, adding the wider context may just add noise to the high-scoring Bi-LSTM-CRF model, whereas the less accurate dilated model can still benefit from the refined predictions of the iterated dilated convolutions.

7 Conclusion

We present iterated dilated convolutional neural networks, fast token encoders that efficiently aggregate broad context without losing resolution. These provide impressive speed improvements for sequence labeling, particularly when processing entire documents at a time. In the future we hope to extend this work to NLP tasks with richer structured output, such as parsing.

Acknowledgments

We thank Subhransu Maji and Luke Vilnis for helpful discussions, and Brendan O'Connor, Yoav Goldberg, the UMass NLP reading group and many anonymous reviewers for constructive comments on various drafts of the paper. We are also grateful to Guillaume Lample for sharing his pretrained word embeddings. This work was supported in part by the Center for Intelligent Information Retrieval, in part by DARPA under agreement number FA8750-13-2-0020, in part by Defense Advanced Research Agency (DARPA) contract number HR0011-15-2-0036, in part by the National Science Foundation (NSF) grant number DMR-1534431, and in part by the National Science Foundation (NSF) grant number IIS-1514053. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow.org*.

Razvan Bunescu and Raymond J. Mooney. 2004. Collective information extraction with relational markov networks. In *ACL*, pages 439–446.

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. 2015. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*.

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

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

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing and linking. *Transactions of the Association for Computational Linguistics*, 2:477–490.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, pages 363–370.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *AISTATS*.

Çalar Gülçehre and Yoshua Bengio. 2016. Knowledge matters: Importance of prior information for optimization. *Journal of Machine Learning Research*, 17(8):1–32.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL*, pages 172–175. Association for Computational Linguistics.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

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

[7] According to the ACL Wiki page on CoNLL-2003: "The corpus contains a very high ratio of metonymic references (city names standing for sport teams)"

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60.

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

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*.

Chen-Yu Lee, Saining Xie, Patrick W Gallagher, Zhengyou Zhang, and Zhuowen Tu. 2015. Deeply-supervised nets. In *AISTATS*, volume 2, page 5.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. *Empirical Methods in Natural Language Processing*.

Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proceedings of the 25th international conference on Machine learning*, pages 592–599. ACM.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. In *EMNLP*.

Ben London, Bert Huang, and Lise Getoor. 2016. Stability and generalization in structured prediction. *Journal of Machine Learning Research*, 17(222):1–52.

Xuezhe Ma, Yingkai Gaom, Zhiting Hu, Yaoliang Yu, Yuntian Deng, and Eduard Hovy. 2017. Dropout with expectation-linear regularization. In *ICLR*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, page 10641074.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *CoNLL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Bj orkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2006. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Proceedings of the Joint Conference on EMNLP and CoNLL: Shared Task*, pages 1–40.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Charles Sutton and Andrew McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. In *ICML Workshop on Statistical Relational Learning*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Annual Meeting of the Association for Computational Linguistics*.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. In *arXiv preprint arXiv:1603.06270*.

Fisher Yu and Vladlen Koltun. 2016. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28 (NIPS)*.

Entity Linking via Joint Encoding of Types, Descriptions, and Context

Nitish Gupta[*]
University of Pennsylvania
Philadelphia, PA
nitishg@seas.upenn.edu

Sameer Singh
University of California
Irvine, CA
sameer@uci.edu

Dan Roth[*]
University of Pennsylvania
Philadelphia, PA
danroth@seas.upenn.edu

Abstract

For accurate entity linking, we need to capture various information aspects of an entity, such as its description in a KB, contexts in which it is mentioned, and structured knowledge. Additionally, a linking system should work on texts from different domains without requiring domain-specific training data or hand-engineered features.

In this work we present a neural, modular entity linking system that learns a unified dense representation for each entity using multiple sources of information, such as its description, contexts around its mentions, and its fine-grained types. We show that the resulting entity linking system is effective at combining these sources, and performs competitively, sometimes out-performing current state-of-the-art systems across datasets, without requiring any domain-specific training data or hand-engineered features. We also show that our model can effectively "embed" entities that are new to the KB, and is able to link its mentions accurately.

1 Introduction

Entity linking, the task of identifying the real-world entity a mention in text refers to, provides the ability to ground text to existing knowledge bases, and thus supports multiple natural language understanding, and knowledge acquisition tasks.

A key challenge for successful entity linking is the need to capture semantic and background information at various levels of granularity. For example, to resolve the mention "India" in "**India** plays a match in England today" to the correct entity, `India_cricket_team`, one needs to use mention-level context to identify that the sentence refers to a sports team (using *plays* and *match*), use document-level context to identify the sport, and information about the entity to realize that `India_cricket_team` is a sports team and the string "India" may refer to it. The problem has been studied extensively by employing a variety of machine learning, and inference methods, including a pipeline of deterministic modules (Ling et al., 2015), simple classifiers (Cucerzan, 2007; Ratinov et al., 2011), graphical models (Durrett and Klein, 2014), classifiers augmented with ILP inference (Cheng and Roth, 2013), and more recently, neural approaches (He et al., 2013; Sun et al., 2015; Francis-Landau et al., 2016).

We present a neural approach to linking[1] that learns a dense unified representation of entities by encoding the semantic and background information from multiple sources – encyclopedic entity descriptions, entity-type information, and the contexts the entity occurs in – thus capturing different aspects of the "meaning" of an entity. Hence, we overcome the shortcomings of several existing models that do not capture all these aspects. For example, methods, such as Vinculum (Ling et al., 2015), do not make use of the local context of the mention ("plays" and "match") while others, such as Berkeley-CNN (Francis-Landau et al., 2016), do not take entity-types into account. Our proposed model uses compositional training to ensure that the learned entity representation captures the various information sources available to it, making it quite modular. Specifically, we introduce encoders for the different sources of information about the entity, and encourage the entity embedding to be similar to all of the encoded representations.

A key requirement for information extraction systems is their ability to work across texts from

[*] Work performed while these authors were at UIUC.

[1] The source code and the datasets are available at https://nitishgupta.github.io/neural-el

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2681–2690
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

various domains. Some methods (Francis-Landau et al., 2016; Nguyen et al., 2016; Hoffart et al., 2011) train parameters on domain-specific linked data, thus hampering their ability to generalize to new domains. By only making use of indirect supervision that is available in Wikipedia/Freebase, we refrain from using domain specific training data, and produce a domain-independent linking system. Our comprehensive evaluation on recent entity linking benchmarks reveals that the resulting entity linker compares favorably to state-of-the-art systems across datasets, even those that have hand-engineered features or use dataset-specific training. We hence show that our model not only leverages all the available information for each entity effectively, but is also robust to missing information, such as entities without links/description in Wikipedia or with incomplete entity types.

In the real-world, new entities are regularly added to the knowledge bases, thus, it is important for any entity linking system to be extendable to such entities, especially the ones that do not have any existing linked mentions. By the virtue of our model's modular nature, it can easily incorporate new entities not present during training. Specifically, we show that our model can perform accurate linking for new entities, without having to re-train the existing entity representations, only using their description and types.

2 Related Work

Existing approaches for entity linking differ in several ways, including the machine learning models, the types of training data, and the kinds of information used about the entities.

Many existing approaches use links and information from Wikipedia as the only source of supervision to build the entity linking system. These approaches use sparse entity and mention-context representations, such as, based on the Wikipedia categories (Cucerzan, 2007), weighted bag of words in the entity description and mention context (Kulkarni et al., 2009; Ratinov et al., 2011), hand crafted features based on partial string matches, punctuations in entity name (McNamee et al., 2009), etc. Heuristics (Mihalcea and Csomai, 2007) or linear classifiers (Bunescu and Pasca, 2006; Cucerzan, 2007; Ratinov et al., 2011; McNamee et al., 2009) are used over these features to rank entity candidates for linking. Recently, neural models have been proposed as a way to support better generalization over the sparse features; e.g., using feed-forward networks on bag-of-words of the entity context (He et al., 2013), or using entity-class information from KB (Sun et al., 2015).

Some models ignore the entity's description on Wikipedia, but rather, only rely on the context from links to learn entity representations (Lazic et al., 2015), or use a pipeline of existing annotators to filter entity candidates (Ling et al., 2015). Our model is similar to these approaches by only using information from Wikipedia; however, we do not use hand-crafted features, and use multiple sources of information such as local and document-level entity context, KB descriptions, and entity types, to learn explicit entity representation.

Few recent entity linking approaches (Hoffart et al., 2011; Durrett and Klein, 2014; Nguyen et al., 2016; Francis-Landau et al., 2016) use manually-annotated domain specific training data to learn the linking system. AIDA (Hoffart et al., 2011), for example, evaluate their system on test set from CoNLL-YAGO dataset but also train on the training data from the same dataset. Berkeley-CNN (Francis-Landau et al., 2016), that uses CNNs operating over different granularity of entity and mention contexts, also follows this training regime and trains separate models for each dataset. Such approaches can be prohibitive in many applications as it encourages the model to over-fit to the peculiarities of different datasets and domains.

Other forms of information, apart from descriptions, and context from linked data, are also utilized for linking. Many approaches perform joint inference over the linking decisions in a document (Milne and Witten, 2008; Ratinov et al., 2011; Hoffart et al., 2011; Globerson et al., 2016), identify mentions that do not link to any existing entity (`NIL`) (Bunescu and Pasca, 2006; Ratinov et al., 2011), and cluster `NIL`-mentions (Wick et al., 2013; Lazic et al., 2015) to discover new entities. Few approaches jointly model entity linking, and other related NLP tasks to improve linking, such as, coreference resolution (Hajishirzi et al., 2013), relational inference (Cheng and Roth, 2013), and joint coreference with typing (Durrett and Klein, 2014). In our model, we use fine-grained type information of the entity as an auxiliary distant supervision to improve mention-context representation but do not use intermediate typing decisions for linking.

Many approaches that learn entity embeddings for other applications have also been proposed,

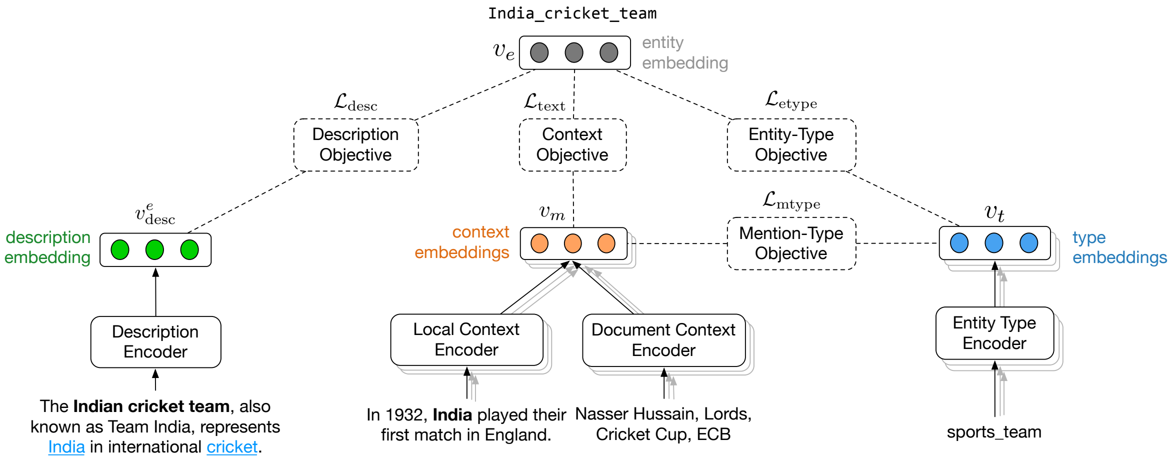


Figure 1: **Overview of the Model** (§ 3): Each entity has a Wikipedia description, linked mentions in Wikipedia (only one shown), and fine-grained types from Freebase (only one shown). We encode local and document-level mention contexts (§ 3.1), entity-description (§ 3.2), and fine-grained entity-types (§ 3.3 & § 3.4). Joint optimization (§ 3.5) over these provides the unified entity representations $\{v_e\}$.

such as, from the structured KB for KB completion (Bordes et al., 2011, 2013; Yang et al., 2014; Lin et al., 2015), or from both structured KBs, and text for relation extraction (Toutanova et al., 2016; Verga et al., 2016a). However, since it is not trivial for these models to incorporate new entities to the KB, few recent approaches alleviate this issue by representing entities as a composition of words in their names (Socher et al., 2013), relations they participate in (Verga et al., 2016b), or their types (Das et al., 2017), but do not use multiple sources of information jointly. In our work, we use structured knowledge (types) as well as unstructured knowledge (description and context) to learn entity embeddings for entity linking, and show that it extends to new entities.

3 Jointly Embedding Entity Information

Knowledge bases contain different kinds of information about entities such as textual description, linked mentions (in Wikipedia), and types (in Freebase). For accurate linking, it is often necessary to combine information from these various sources. Here, we describe our model that encodes information about the set of entities $\mathcal{E}$ using dense unified representation for linking ($v_e \in \mathbb{R}^d, \forall e \in \mathcal{E}$). In particular, we use existing mentions in Wikipedia to encode the context (§ 3.1), textual descriptions from Wikipedia to encode background information (§ 3.2), and fine-grained types from Freebase as structured topical knowledge (§ 3.3). Figure 1 provides an overview of our model.

3.1 Encoding the Mention Context, C

Consider the example mention in Figure 1 that contains two mentions, "India" and "England". In order to disambiguate "India" to the correct entity, a linking system would need to utilize both the local context (*played* and *match*), and the document context (to identify the sport). However, the model needs to represent context such that the semantics are preserved, e.g. "England" should not be linked to a sports team even though it shares the context with "India". In this section, we describe how we encode these two types of context, using a LSTM-based encoder to capture the lexical and syntactical local-context of a mention (v_m^{local}), and a feed-forward network to encode the document-level topical knowledge (v_m^{doc}), and combine them in a single representation for each mention (v_m).

Local-Context Encoder Given a mention m in the sentence $s = w_1, \ldots, m, \ldots, w_N$, we use LSTM encoders on the left ($w_1, \ldots, m$) and right ($m, \ldots, w_N$) contexts of the mention separately, and then combine it to form the local context representation of the mention (Fig. 2). More precisely, we formulate an LSTM as $h_i, s_i = \text{LSTM}(u_i, h_{i-1}, s_{i-1})$, $u_i \in \mathbb{R}^{d_w}$ is the input embedding of the i-th token in the sequence, and $h_{i-1}, s_{i-1} \in \mathbb{R}^l$ is the previous output and the cell state of the LSTM, respectively. The left-LSTM is applied to the sequence ($w_1, \ldots, m$) with the last output $\overrightarrow{h^l_m}$, while a different right-LSTM is ap-

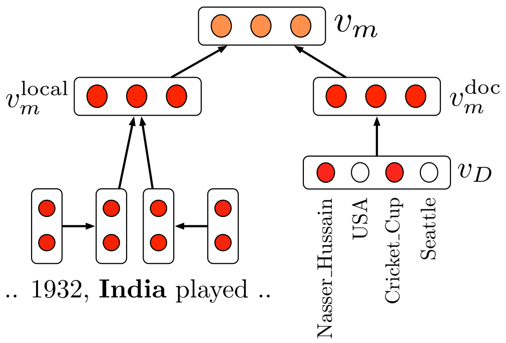


Figure 2: Overview of the mention context encoder

plied to the sequence $(w_N, \ldots, m)$[2] to produce $\overleftarrow{h^r_m}$. We concatenate these output $[\overrightarrow{h^l_m}, \overleftarrow{h^r_m}]$ and pass it through a single layer feed-forward network[3] to produce the local context representation of the mention (v_m^{local}), where $v_m^{\text{local}} \in \mathbb{R}^{D_m}$. Note that this encoder will produce different representations for different mentions in the same sentence.

Document-Context Encoder To represent the document context of a mention m, we use a bag-of-mention surfaces representation, $v_D \in \{0,1\}^{|V_G|}$, of the document, similar to Lazic et al. (2015). The vocabulary V_G consists of all mention surfaces seen in our training data, e.g. *USA*, *Nasser_Hussain*, *Pearl_Jam* etc. Such a representation helps capture the topical and entity coherence information in the document by utilizing co-occurrence between entity surface forms. This sparse vector v_D of bag-of-mention surfaces is compressed to a low-dimensional representation $v_m^{\text{doc}} \in \mathbb{R}^{D_m}$ using a single layer feed-forward network.

Mention-Context Encoder We combine the local (v_m^{local}) and document (v_m^{doc}) level context vectors by concatenating them, and passing them through a single-layer feed-forward network to obtain the mention context embedding $v_m \in \mathbb{R}^d$. In order to learn the entity representation v_e such that it encodes all of its mentions' contexts, we introduce an objective that encourages the context representation v_m to be similar to v_e (where mention m is a link to entity e), and dissimilar to other candidates[4]. Precisely, we maximize the probability of predicting the correct entity from the mention-context vector as $P_{\text{text}}(e|m) = \frac{\exp(v_m \cdot v_e)}{\sum\limits_{c_k \in C_m} \exp(v_m \cdot v_{c_k})}$, where C_m is the set of candidate entities. Given all the mentions in Wikipedia, we jointly optimize the entity representations, and the context encoders by maximizing the following log-likelihood:

$$\mathcal{L}_{\text{text}} = \frac{1}{M} \sum_{i=1}^{M} \log P_{\text{text}}(e_{m^{(i)}} | m^{(i)}) \qquad (1)$$

where $m^{(i)}$ is the i^{th} mention in the linked data, and $e_{m^{(i)}}$ is the entity the mention refers to.

3.2 Encoding Entity Description, D

The textual description about entities in Wikipedia can provide a useful source of background information about the entity, and thus has been used in many existing linking systems. Given the description as a sequence of words, we first embed each word to a d_w-dimensional vector resulting in a sequence of vectors $w_1, \ldots, w_n$. To encode this description as a fixed size vector, we use a Convolution Neural Network (CNN), similar to Francis-Landau et al. (2016), with global average pooling, to obtain $v^e_{\text{desc}} \in \mathbb{R}^d$.

In order for the entity representation v_e to encode its description, we use a similar objective as in the previous section § 3.1, i.e. we maximize the probability $P_{\text{desc}}(e|v^e_{\text{desc}})$, and learn the parameters by maximizing the log-likelihood $\mathcal{L}_{\text{desc}}$, defined similarly as (1).

3.3 Encoding Fine-Grained Types, E

Fine-grained types provide a source of structured information that is quite readily available, often more easily than the description or linked data (e.g. Freebase contains tens of millions of entities with types but Wikipedia only contains descriptions for a few million). These types have been shown to be quite useful for linking (Ling et al., 2015), since an accurate prediction of types from the mention, and its match with the entity types can often resolve many challenging ambiguities.

Here, we focus on being able to represent the different types at the entity level, leaving mention-level type information to the next section. Each entity has multiple types $T_e \subset \mathcal{T}$ from the type set $\mathcal{T}$ introduced by Ling and Weld (2012). We compute the probability $P(t|e)$ of type t being relevant to entity e as $\sigma(v_t \cdot v_e)$, where σ is the sigmoid function, $v_e \in \mathbb{R}^d$ is the entity representation, and $v_t \in \mathbb{R}^d$ is the embedding of type t in $\mathcal{T}$. We maximize the log-likelihood of the type information to

[2] We reverse the token sequence in the right context so that right-LSTM starts at the last token and ends at the mention.

[3] We use rectified linear unit (ReLU) as the non-linear activation throughout this paper.

[4] Details on candidate generation in Sec 4

jointly learn entity and type representations:

$$\mathcal{L}_{\text{etype}} = \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} \log \prod_{t \in T_e} P(t|e) \prod_{t' \notin T_e} (1 - P(t'|e))$$

3.4 Type-Aware Context Representation, T

Apart from being able to represent the types of the entities, it is also important for our linker to be able to represent the type information at the mention level. In the example in Fig. 1, although the mention “India” is prominently used to refer to the *country*, it is evident from the sentence that it refers to a *Sports Team*. The context-encoder captures this information in an unstructured manner, thus it will be useful for the encoder to directly utilize this supervision. This is a similar setup as Ling et al. (2015) and Shimaoka et al. (2017) that use noisy distant supervision to train a fine-grained type predictor for mentions.

In order for the context encoders, and type embeddings to directly inform each other, we introduce an objective $\mathcal{L}_{\text{mtype}}$ between every v_m and v_t if type t belongs to T_e for the entity e that m refers to. This objective is similar to $\mathcal{L}_{\text{etype}}$ from § 3.3.

3.5 Learning Unified Entity Representations

In the sections above we described different encoder models to capture entity-context information (local- and document-level), entity-description from a KB, and fine-grained types in a single entity representation vector. To learn the entity representations, and parameters of the encoders, we jointly maximize the total objective:

$$\{v_e\}, \Theta = \underset{\{v_e\},\Theta}{\operatorname{argmax}} \mathcal{L}_{\text{text}} + \mathcal{L}_{\text{desc}} + \mathcal{L}_{\text{etype}} + \mathcal{L}_{\text{mtype}}$$

where $\{v_e\}$ is the set of entity representations, and Θ is set of parameters for the different encoders. One advantage of having such a joint, modular objective is that it is robust to missing information, i.e. entities with missing mentions, types, or descriptions will still obtain accurate representations learned using other sources of information.

4 Entity Linking

Given a document, and mentions marked in it for disambiguation, we perform a two-step procedure to link them to an entity. First, we find a set of candidate entities, and their *prior* scores using a pre-computed dictionary. We then use our mention-context encoder to estimate the semantic similarity of each mention with the vector representations of each entity candidate, and combine the results from the two sources for making linking decisions.

A typical KB contains millions of entities, which makes it prohibitively expensive to compute a similarity score between each mention and all entities in the KB. Prior work has shown that, for a given mention, aggressively pruning the set of possible entities to a small subset hurts performance only negligibly, while making the linker extremely efficient. For each mention m, we generate a set of candidate entities $C_m = \{c_j\} \subset \mathcal{E}$ using CrossWikis (Spitkovsky and Chang, 2012), a dictionary computed from a Google crawl of the web that stores the frequency with which a mention links to a particular entity. To generate C_m we choose the top-30 entities for each mention string, and normalize this frequency across the chosen candidates to compute $P_{\text{prior}}(e|m)$. In the literature, such a dictionary is often built from the anchor links in Wikipedia (Ratinov et al., 2011; Hoffart et al., 2011) but Ling et al. (2015) show using CrossWikis gives improved prior scores and candidate recall.

For each mention m, we use our learned mention-context encoder from § 3.1 to encode the mention's context as v_m, and estimate the distribution over the candidates using $P_{\text{text}}(e|m)$. We treat these two pieces of evidence; pre-computed prior probability, and the context-based probability, as independent, disjunctive sources of signal, and thus combine them to compute $P(e|m)$ as:

$$\begin{aligned} P(e|m) &= P_{\text{prior}}(e|m) + P_{\text{text}}(e|m) \\ &\quad - (P_{\text{prior}}(e|m) * P_{\text{text}}(e|m)) \qquad (2) \\ \hat{e}_m &= \underset{e \in C_m}{\operatorname{argmax}}\ P(e|m) \qquad (3) \end{aligned}$$

where $\hat{e}_m$ is the predicted entity that the mention m should be disambiguated to.

5 Evaluation Setup

Here we provide a detailed description of how we train our models, benchmark datasets, linking systems we compare to, and the evaluation metrics.

Training Data Our primary source of information about the entities is Wikipedia (dump dated 2016/09/20). We use existing links in Wikipedia, with the anchors as mentions, and links as the true entity, as input to the context encoder (see § 3.1). As the description of each entity (§ 3.2), we use the first 100 tokens of the entity's Wikipedia page (same as Francis-Landau et al. (2016)). To obtain entity types (see § 3.3), we extract the types for each entity from Freebase and map them to the 112 fine-grained types introduced by Ling and Weld (2012). For context and description encoders, we use pre-trained 300-dimensional case-sensitive word embeddings by Pennington et al. (2014) as the first layer that is not updated during training.

Hyper-parameters We perform coarse-grained tuning of the hyper-parameters using a fraction of the training data. The vectors for the entities, types, contexts, and descriptions are of size $d = 200$. The size of the local context encoder LSTM hidden layer l, local context output, and the document-context encoder output D_m is set to $100 (= l = D_m)$. The document context vocabulary contains $|V_G| = 1.5$ million strings. We use dropout (Srivastava et al., 2014) with a probability of 0.4. Additionally, we use word-dropout where we replace a random subset of tokens (mention-strings) in the local (document) context with "unk" (rate of 0.4 and 0.6 for local and document context respectively). We use Adam (Kingma and Ba, 2014) for optimization, with learning rate 0.005 and mini-batches of size 1000.

Existing Approaches We compare our approach to the following five entity-linking models: (1) **Plato** (Lazic et al., 2015), an unsupervised generative model that uses indirect-supervision from Wikipedia and an additional corpus of 50 million unlabeled webpages, (2) **Wikifier** (Ratinov et al., 2011), an unsupervised linker that uses hand-crafted features to rank candidates, (3) **Vinculum** (Ling et al., 2015), a modular, unsupervised pipeline system, (4) **AIDA** (Hoffart et al., 2011), a supervised linker trained on CoNLL data and uses hand-crafted features, and (5) **BerkCNN** (Francis-Landau et al., 2016), a recent neural supervised approach that has variants that use hand-crafted features.

Evaluation Setup We evaluate our approach on the following four datasets: CoNLL-YAGO (Hoffart et al., 2011), ACE 2004 (NIST, 2004; Ratinov et al., 2011), ACE 2005 (NIST, 2005; Bentivogli et al., 2010), and Wikipedia (Ratinov et al., 2011). For each of these datasets, we use the standard test/development splits, but do not use any information from the training splits. End-to-end entity linking systems such as Vinculum and Wikifier perform an NER-style F1 evaluation where a prediction is only considered correct if the system mention boundaries match the gold annotation, and the predicted link is correct (we compare against these by extracting mentions with Stanford-NER). On the other hand, systems like Plato, AIDA, and Berkeley-CNN assume mentions are provided, and evaluate using the linking accuracy for gold-mentions. Further, the approaches we compare here (including ours) do not predict NIL entities for the datasets evaluated on.

	CoNLL		**ACE05**	**Wiki**
	Test	**Dev**		
Plato (Sup)	79.7	-	-	-
Plato (Semi-Sup)	86.4	-	-	-
*AIDA**	*81.8*	-	-	-
*BerkCNN:Sparse**	*74.9*	-	*83.6*	*81.5*
*BerkCNN:CNN**	*81.2*	*86.91*	*84.5*	*75.7*
*BerkCNN:Full**	*85.5*	-	*89.9*	*82.2*
Priors	68.5	70.9	81.1	78.1
Model C	81.4	83.4	83.7	86.1
Model CD	81.0	83.2	85.8	86.1
Model CT	82.3	83.9	86.5	88.2
Model CDT	82.5	85.6	86.8	88.0
Model CDTE	82.9	84.9	85.6	89.0

Table 1: **Entity Linking Performance:** Accuracy of existing systems, and variations of our model on gold mentions. The model using context information is labeled C, entity-description as D, context-typing as T, and entity-type encoding as E. Existing models marked in *Italics** train domain-specific linkers for each dataset. Our system performs competitively to these systems, and outperforms Plato (Sup) that uses the same indirect supervision.

6 Results

In this section we present various experiments to evaluate the performance of our proposed entity-linking system. Specifically, we focus on the following questions: (1) how effective is our model in combining different information on standard linking benchmarks, without requiring domain specific information (§ 6.1), (2) is our model able to accommodate unseen entities by using their types, or description, without re-training the entity representations (§ 6.2), and (3) how does the model perform on fine-grained mention typing, a task it is not directly trained for, compared to approaches designed for the task (§ 6.3). Further, Sec 6.4 presents examples to show the effect of encoding different kinds of information in a unified entity representation.

	F1	Accuracy
AIDA	77.8	-
Wikifier	85.1	-
Vinculum	88.5	-
Model C	88.9	93.1
Model CDT	89.8	93.9
Model CDTE	90.7	94.3

Table 2: **Results for ACE-2004**: F1 is calculated for predicted mentions, and accuracy on gold-mentions. Results for Wikifier and AIDA are from (Ling et al., 2015). All systems use the same mention extraction protocol showing the difference in F1 is due to linking performance.

6.1 Entity Linking

In Table 1 we present linking accuracy for our models that vary in the information they use. We see that the model that only encodes the context-information, Model C ($\mathcal{L} = \mathcal{L}_{\text{text}}$) consistently performs better than picking the entity with the highest prior probability from CrossWikis, indicating that the model is able to utilize the context across datasets. On incorporating the description with context (Model CD) we see improvement in the performance on ACE-2005, but slight decrease in CoNLL, suggesting the entity descriptions are not extremely useful for the latter (it contains rare entities, many short and incomplete sentences, and specific entities as annotations for metonymic mentions, as also observed by Ling et al. (2015)). On introducing the entity type-aware loss in Model CT to the context-only model, we see significantly improved results for all datasets, demonstrating that explicitly modeling fine-grained types helps learning a better context encoder and, in turn, type-aware entity representations. Combining descriptions with this model (Model CDT) shows further gains in accuracy indicating that our model is able to exploit complementary information from the two sources. Finally, on introducing explicit entity-type encoding, Model CDTE performs the best on two of the four datasets. As we will see in § 6.2, encoding entity-type information also allows our models to easily generalize to new entities.

On comparison to existing systems we see that all our variants outperform Plato's indirectly-supervised model trained on Wikipedia, which is the same information our Model C and CD use. Their semi-supervised model, that is additionally trained on 50 million web-pages, performs much better. In comparison to AIDA and Berkeley-CNN, that train separate models on respective datasets, we perform better than AIDA and Berkeley-CNN's *sparse* and neural model. On combining features from CNN to the *sparse* model, the Berkeley-CNN models for each dataset outperform our model, but are unlikely to generalize across the datasets[5].

Method	Accuracy
Random Guessing	16.7
Random Embeddings	34.0
Entity Description	65.1
Fine-Grained Types	73.7
Description + Types	79.5

Table 3: **Cold-Start Entities:** Linking new entities by using different information to learn their embeddings. Our model is able to jointly utilize description and type information better.

In Table 2 we present results for our models on ACE-2004. Our model outperforms the Wikifier and Vinculum systems that only use information from Wikipedia, and AIDA, by a significant margin, indicating its possible over-fitting to the CoNLL domain. Hence, it shows our model's ability to perform accurate linking across different datasets without using domain-specific information.

6.2 Cold-Start Entities

In realistic situations, new entities are regularly added to the knowledge base with little or no linked data for them. Hence, it is important for any information extraction system that learns entity representations to be easily extendable for such entities without needing to be re-trained. In this section, we consider the use of our approach to this setting.

In particular, for each such new entity, we need to determine their embedding using only their description and/or type information. For a new entity for which only the description is available, we directly set its embedding to be the output of the entity-description encoder without any need for learning. If only fine-grained types are available, we learn the new entity-embedding by optimizing the objective $\mathcal{L}_{\text{etype}}$. In case both description and types are available, we jointly maximize the similarity of the entity embedding with the output of the entity-description, and the type encoders (i.e. optimize $\mathcal{L}_{\text{desc}}$ and $\mathcal{L}_{\text{etype}}$). Note that we only learn the embeddings of each new entity, keeping all other

[5] Ling et al. (2015) show that AIDA is unable to perform well on datasets it has not been trained on.

Models	Acc.	Macro F1	Micro F1
FIGER	47.4	69.2	65.5
SSIR-LSTM	55.6	75.1	71.7
SSIR-Full	59.6	78.9	75.3
Our Model	57.7	72.8	72.1

Table 4: **Typing Prediction:** Performance on the FIGER (GOLD) dataset. Our performance is competitive with FIGER (Ling and Weld, 2012) and neural-LSTM model of Shimaoka et al. (2017). Their SSIR-Full model that uses a biLSTM layer, an attention layer, combined with hand-crafted features is state-of-art for this task.

parameters of our model (Model CDTE) fixed.

To evaluate this setting of new entities, we randomly select 1000 rare entities from Wikipedia that are not used during training. Among all mentions of these entities in Wikipedia, we only keep the mentions for which our candidate generation generates more than one candidate, resulting in 3791 mentions. On average, each mention had 6 candidate entities, and further, as priors are not available in this setting, we only rely on the context probability for linking, making this a challenging task.

We present the results of using different types of information about the entity for this data in Table 3. It is surprising that randomly initialized embeddings for these new entities perform better than random guessing, suggesting our model is sometimes able to eliminate the wrong candidates purely based on their learned embedding, i.e. an entity with a random embedding has a higher likelihood of being the correct entity. More importantly, we see that our model variants that utilize the available entity information are able to link much more accurately (47-60% error reduction). Further, using both description and types results in the best embeddings for these new entities ($\sim$ 80% accuracy).

6.3 Fine-Grained Typing

Since entity embeddings are trained to be both, context and type-aware, we evaluate whether they can be used to predict fine-grained types for mentions from context (using v_m and v_t). Compared to existing systems trained specifically for this task, embeddings from our approach (Model CDTE) performs competitively (see Table 4). In particular, our model performs better than the neural-LSTM model of Shimaoka et al. (2017), suggesting that our multi-task linking, and typing loss facilitates effective encoding of mention contexts.

12th Asian Nations Cup finals are hosted by **Lebanon** until this October 29. **Model CD:** Lebanon_football_team **Model CT:** Lebanon *(correct)* **Model CDTE:** Lebanon *(correct)*
Yugoslav midfielder **Petrovic** scored twice as PSV Eindhoven romped to a 6-0 win. **Model CD:** Zeljko_Petrovic *(correct)* **Model CT:** Vladimir_Petrovic **Model CDTE:** Zeljko_Petrovic *(correct)*
Ince was clambering over a wall at the Republican stadium during an under-21 clash. **Model CD:** Ince **Model CT:** Tom_Ince **Model CDTE:** Paul_Ince *(correct)*

Table 5: **Example predictions by our models:** Model CT (Ex.1) and CD (Ex.2) predict correctly when correct type prediction or background knowledge is sufficient, respectively. Only Model CDTE (Ex.3) predicts correctly when combination of context, types, and background knowledge is required.

6.4 Example Predictions

In Table 5 we show the prediction from different variants of our model for a few example mentions. In the first example, detecting the type of the mention is crucial, and thus we see both Model CT and CDTE are able to predict accurately. On the other hand, predicting the type of the mention is not especially useful in Example 2, and background factual knowledge from the entity description is needed (which models CD and CDTE are able to encode). Example 3 shows a challenging example where the appropriate combination of context, type prediction, and background knowledge is needed, that our Model CDTE is able to combine.

7 Conclusion

Motivated by the need to provide accurate entity-linking systems that are able to incorporate multiple sources of information, and do not require domain-specific datasets or hand-crafted features, we presented a novel neural approach to linking. We proposed a compositional training objective to learn unified entity embeddings that encode the variety of information available for each entity: its unstructured textual description, local and document contexts for its mentions, and sets of fine-grained types attached to it. The joint formulation allows the model to fruitfully combine the various sources of information, providing accurate linking on multiple datasets, generalization to new entities with missing linked data, and the use of entity embeddings for related tasks such as type prediction.

There are a number of avenues for future work. Further research will include encoding more structured knowledge about the entities, such as their relations to other entities, to make their representations semantically richer. We will investigate how we can use unstructured resources, such as the corpus of unlabeled webpages used by Plato, and noisy supervision from the Wikilinks corpus (Singh et al., 2012) in order to further improve the model. We will also evaluate our approach on substantially varied domains, such as discussion forums, and social media posts.

Acknowledgments

We would like to thank Mike Lewis, Xiao Ling, Ananya, Shyam Upadhyay, and the anonymous EMNLP reviewers for their valuable feedback. This work is supported in part by the US Defense Advanced Research Projects Agency (DARPA) under contract FA8750-13-2-0008, and in part by an Adobe Research faculty award. The views expressed are those of the authors and do not reflect the official policy or position of the Dept. of Defense, the U.S. Government, or Adobe.

References

Luisa Bentivogli, Pamela Forner, Claudio Giuliano, Alessandro Marchetti, Emanuele Pianta, and Kateryna Tymoshenko. 2010. Extending english ace 2005 corpus annotation with ground-truth links to wikipedia. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Razvan Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the European Chapter of the ACL (EACL)*.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 708–716.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the European Chapter of the ACL (EACL)*.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the Association for Computational Linguistics (TACL)*.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hannaneh Hajishirzi, Leila Zilles, Daniel S Weld, and Luke S Zettlemoyer. 2013. Joint coreference resolution and named-entity linking with multi-pass sieves. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Nevena Lazic, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2015. Plato: A selective context model for entity resolution. *Transactions of the Association for Computational Linguistics (TACL)*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Xiao Ling, Sameer Singh, and Daniel S Weld. 2015. Design challenges for entity linking. *Transactions of the Association for Computational Linguistics (TACL)*.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Paul McNamee, Mark Dredze, Adam Gerber, Nikesh Garera, Tim Finin, James Mayfield, Christine D Piatko, Delip Rao, David Yarowsky, and Markus Dreyer. 2009. Hltcoe approaches to knowledge base population at tac 2009. In *TAC*.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)*.

David Milne and Ian H. Witten. 2008. Learning to link with Wikipedia. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM)*, pages 509–518.

Thien Huu Nguyen, Nicolas Fauceglia, Mariano Rodriguez-Muro, Oktie Hassanzadeh, Alfio Massimiliano Gliozzo, and Mohammad Sadoghi. 2016. Joint learning of local and global features for entity linking via neural networks. In *Proceedings the International Conference on Computational Linguistics (COLING)*.

NIST. 2005. The ACE evaluation plan.

US NIST. 2004. The ace evaluation plan. *US National Institute for Standards and Technology (NIST)*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of the European Chapter of the ACL (EACL)*.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical report, University of Massachusetts, Amherst.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*.

Valentin I Spitkovsky and Angel X Chang. 2012. A cross-lingual dictionary for english wikipedia concepts. In *LREC*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Kristina Toutanova, Xi Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge bases and text. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016a. Multilingual relation extraction using compositional universal schema. In *Proceedings of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

Patrick Verga, Arvind Neelakantan, and Andrew McCallum. 2016b. Generalizing to unseen entities and entity pairs with row-less universal schema. In *Proceedings of the European Chapter of the ACL (EACL)*.

Michael Wick, Sameer Singh, Harshal Pandya, and Andrew McCallum. 2013. A joint model for discovering and linking entities. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

An Insight Extraction System on BioMedical Literature with Deep Neural Networks

Hua He[1], **Kris Ganjam**[2], **Navendu Jain**[2], **Jessica Lundin**[2], **Ryen White**[2] and **Jimmy Lin**[3]
[1] Department of Computer Science, University of Maryland College Park
huah@cs.umd.edu
[2] Microsoft
{krisgan,navendu,jelundin,ryenw}@microsoft.com
[3] Cheriton School of Computer Science, University of Waterloo
jimmylin@uwaterloo.ca

Abstract

Mining biomedical text offers an opportunity to automatically discover important facts and infer associations among them. As new scientific findings appear across a large collection of biomedical publications, our aim is to tap into this literature to automate biomedical knowledge extraction and identify important insights from them. Towards that goal, we develop a system with novel deep neural networks to extract insights on biomedical literature. Evaluation shows our system is able to provide insights with competitive accuracy of human acceptance and its relation extraction component outperforms previous work.

1 Introduction

Biomedical literature offers a rich set of knowledge sources to discover important facts and find associations among them. For instance, MEDLINE contains over 18 million references to articles published since 1946 and sourced from over 5500 journals worldwide (Simpson and Demner-Fushman, 2012). Two major processing tasks performed on the biomedical text are: (1) identify and classify biomedical entities (NER) into predefined categories such as proteins, genes, or diseases, and (2) infer pair-wise relationships among named entities e.g., protein-protein interaction (Poon et al., 2014), gene-protein, and medical problem-treatment.

This paper presents a system that processes biomedical text to extract two specific types of relationships among biomedical entities: (a) cause-effect and (b) correlation.

This system is motivated by the need to better automate biomedical knowledge extraction and identify important information from them, as new scientific findings appear across a large collection of publications. For instance, given user sleep patterns, existing biomedical research can be better utilized to provide *insights*: inform about potential *effect* (e.g., "*diabetes*", "*obesity*") due to the *cause* (e.g., "*sleep disorder*") and suggest appropriate treatment.

Since biomedical articles usually have title and abstract summarizing the contents of the full-text article, we focus on extracting the two relationship types from them. Unfortunately, mining this summary data still poses several key challenges. Similar to full-text, this data comprises unstructured text with domain-specific vocabulary, issues of synonymy (e.g., *"heart attack"* vs. *"myocardial infarction"*), acronyms, abbreviations and rapidly evolving terminology due to new scientific discoveries. While the titles are short and informative, they do not contain the key information that would be contained in the abstract.

Many of these challenges are also applicable for biomedical relation extraction. Further, identifying particular relation types is challenging because relations are expressed as discontinuous spans of text , and the relation types are typically application-specific. Finally, there is often little consensus on how to best annotate relation types resulting in lack of high quality annotated corpora for training.

In this study, we develop neural networks with novel similarity modeling for better causality/correlation relation extraction, as we map the extraction task into a representational similarity measurement task in the vector space. Our approach innovates in that it explicitly measures both relational and contextual similarity among representations of named entities, entity relations and contexts. Our system also provides a novel combination of recognizing named entities, predicting

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2691–2701
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

relationships (insights) between extracted entities, and ranking the output. We conduct human evaluations of the system to show it is able to extract insights with high human acceptance accuracy, and on a SemEval task evaluation its causality/correlation relation extraction compares favorably against previous state-of-the-art work.

1.1 Contributions

1. We build an end-to-end system to extract insights from biomedical literature.
2. We innovate in similarity measurement modeling with deep neural networks for better causality/correlation relation extraction.
3. Our human evaluation show our system can achieve competitive acceptance accuracy.

2 Related Work

Most previous work in BioNLP focused on extraction of biomedical concepts (Craven, 1999; Finkel et al., 2005; Poon and Vanderwende, 2010; Simpson and Demner-Fushman, 2012; Liu, 2016), such as drug or protein names. We also conduct relation extraction on general named entities, such as "*smoking*" or "*sleep quality*". Kabiljo et al. (2009) compared pattern-matching techniques against a baseline regular expression approach for gene/protein entity extraction. But existing tools for relation extraction are not as comprehensive as entity recognition tools.

Medical dictionaries and resources are heavily utilized by previous work. For instance, Chen et al. (2008) extracted disease-drug relation pairs with MedLEE (Friedman et al., 2004) system for clinical information extraction of EHR records. Liu et al. (2015) developed a text-mining system to search for associations among human diseases, genes, drugs, metabolites and toxins against large collections of text-rich biological databases. Previous research efforts also lead to semantic representation program SemRep (Rindflesh and Fiszman, 2003), which exploits biomedical domain knowledge and linguistic analysis of biomedical text. Other unconventional resource such as web query logs are also utilized (Paparrizos et al., 2016) to provide early warnings about the presence of devastating diseases.

Feature engineering was the dominant approach in most biomedical relation extraction work with machine learning techniques (Dogan et al., 2011; Yala et al., 2016); different sparse features were explored. For example, word n-gram features, knowledge-based features from medical dictionaries and word position features. Our work instead propose neural network models that do not require sparse features as in most previous work.

Algorithm 1 System Overview

```
 1: Input: Biomedical article title and abstract
 2: Preprocess the input texts
 3: for each sentence of the input do
 4:     Identify all possible named entities
 5:     for each named entity pair $(\vec{A}, \vec{B})$ do
 6:         if causality/correlation holds then
 7:             Extract and Score $(\vec{A}, \vec{B})$
 8:         end if
 9:     end for
10: end for
11: Rank all extracted $(\vec{A}, \vec{B})$ pairs
12: return top ranked entity pairs
```

Recent shift from feature engineering to model engineering with neural networks has significantly improved accuracy on many NLP tasks. Jagannatha and Yu (2016) adopted an LSTM model for medical entity detection given patient EHR records. There are recent work with the use of deep reinforcement learning on healthcare study (Li, 2017). Our approach is inspired by recent embedding learning work to jointly represent texts and knowledge base (Toutanova et al., 2015, 2016), previous work on embedding transfer learning (Bordes et al., 2013) and noise-contrastive estimation (Rao et al., 2016). Lastly our work models insight extraction as a similarity measurement problem, and is inspired by similarity measurement work (He et al., 2016; He and Lin, 2016) on pairwise word interaction modeling with deep neural networks.

3 System Overview

We provide a recipe to build a system for biomedical insight extraction and use it as a guide for the remainder of this paper (Algorithm 1).

To make our discussion concrete, we will use a sample biomedical article in Example 1. Given the text, at line 4 of Algorithm 1 we firstly look for all named entities using a shallow parser and public medical dictionaries (see details in Section 4). Many named entities could be found, for example, *"clinical study"*, *"sleep disturbances in middle-aged men"* and *"diabetes"*. Next given any pair of previously extracted entities within a sentence,

RESEARCH METHODS: *A group of 6,599 initially healthy, nondiabetic middle-aged men took part in a prospective, population-based study. The incidence of diabetes during a mean follow-up of 14.8 years was examined in relation to self-reported difficulties in falling asleep.*
RESULTS: *A total of 615 subjects reported either difficulties in falling asleep or use of hypnotics (seen as makers of sleep disturbances). Among those, 281 of the men developed diabetes during the follow-up period. The clinical study suggests sleep disturbances in middle-aged men are likely associated with diabetes.*

Example 1: Sample Text

at line 6 our neural network-based relation extractor checks if a valid causality/correlation relationship exists (Section 5). For example, our models can identify that the entity *"sleep disturbances in middle-aged men"* has a correlation relationship with *"diabetes"* but not with *"clinical study"*. Later each valid entity pair is scored via the ranking component at line 7 (Section 6). In the final step, the system returns top ranked insight(s) to users: "*sleep disturbances in middle-aged men* → *diabetes*" given this example.

Figure 1 presents the system which consists of three major neural network-based components: (1) a named entity extractor, (2) a causality/correlation relation extractor, and (3) an insight ranker. Our system reads in biomedical texts, then provides insights in the end. We primarily innovate in the relation extraction component. Next, we describe each of these components in detail.

4 Named Entity Extraction

Named entity extraction in biomedical domain is challenging due to the domain-specific and rapidly evolving terminology. For example, *"Diabetes mellitus type 1"*, *"Type 1 diabetes"*, *"IDDM"*, or *"juvenile diabetes"* all express the same concept. Given frequent evolution of entity naming for new drugs, diseases or abbreviations, this task becomes more complicated.

Most existing off-the-shelf biomedical entity recognizers narrowly focus on specific biomedical terms. Instead we aim to improve the system recall by extracting both specific biomedical concepts such as *"gene tmem230"* or *"prostate cancer"* as well as general noun phrases such as *"sleep quality"*, *"daily exercises"*, or *"men with diabetes"*. Thus the scope of the system is broader.

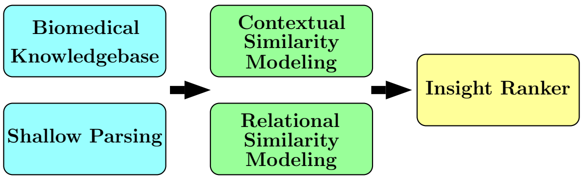

Biomedical Knowledgebase
Shallow Parsing
Contextual Similarity Modeling
Relational Similarity Modeling
Insight Ranker

Figure 1: Three major components of the system.

We design an entity extractor by using both an in-domain medical knowledge base for keyword matching, and a domain-independent neural network-based shallow parser for entity boundary detection. We present the procedure below:

1. We firstly use a large public dictionary, *Metathesaurus* of the Unified Medical Language System (UMLS) (Bodenreider, 2004) to obtain in-domain biomedical terms. UMLS *Metathesaurus* is a set of dictionaries providing large collections of biomedical vocabularies. We extract over 3.3 million of biomedical terms from UMLS, then utilize the *Aho Corasick* pattern matching algorithm to create a dictionary lookup tool. Our tool can efficiently locate all UMLS terms given input texts, since it has a linear complexity due to its trie tree data structure.
2. We also use a neural network-based shallow parser (Collobert et al., 2011) to identify boundaries of general noun phrases, which are not limited to biomedical terms. Usage of shallow parser is to improve system recall on named entity recognition.
3. Our named entity extraction component aims to locate all entities of input texts. The result list is an output concatenation of both step 2 and 3, and is later provided to the causality/correlation relation extraction component for further processing. If entity overlaps exist, only phrases with longest matching sequence are extracted.

Our insight extraction system adopts a coarse-to-fine design approach. First, we focus on improving recall for the entity extraction task. Then we show how the causality/correlation relation extraction component (Sec. 5) processes extracted named entities to achieve high precision.

5 Relation Extraction as Similarity Measurement

We first provide our model design intuition: if a causality/correlation relationship holds between two named entities, then representations of the two entities should be semantically similar and close to the representation of the relation in a low-dimensional vector space. Therefore we map the causality/correlation relation extraction into a similarity measurement task in the vector space.

Our novel approach learns representations of named entities $(\vec{A}, \vec{B})$, context words and the relation vector $\vec{R}$, then explicitly measures two aspects of the similarity: 1) **relational similarity** between entities and relation (Sec. 5.2); plus, 2) **contextual similarity** between entities and sentence context (Sec. 5.3).

The intent of our approach is to enforce such structure of the vector space: as the similarity among entities, relation and contexts gets stronger, a fit of all should be observed for better causality/correlation relation extraction. We develop two neural network models with such property; both are utilized in the relation extraction component of the system.

We define input sentence representation $S \in \mathbb{R}^{\ell \times d}$ to be a sequence of ℓ words, each with a d-dimensional word embedding vector. $x_t \in \mathbb{R}^d$ denotes the embedding vector of the t-th word $(t \in [1, l])$ in S. Model details are described in the following sections.

5.1 Context Modeling

Different words occurring in similar contexts should have a higher chance to contribute to similarity measurement and relation extraction. We use bidirectional LSTMs (BiLSTM) for context modeling as a basis for all following models.

LSTM (Hochreiter and Schmidhuber, 1997) is a special variant of Recurrent Neural Networks (Williams and Zipser, 1989). At time step t, given an input word x_t and previous LSTM hidden state h_{t-1}, $LSTM(x_t, h_{t-1})$ outputs current hidden state $h_t \in \mathbb{R}^{dim}$. BiLSTM consists of two LSTMs that run in parallel in opposite directions. The BiLSTM hidden state $h_t^{bi} \in \mathbb{R}^{2dim}$ is a concatenation of forward LSTM's h_t^{for} and backward LSTM's h_t^{back}, representing contexts of input word x_t in the sentence. We define *concat* operation and output sentence context representation $H^S \in \mathbb{R}^{\ell \times 2dim}$ below:

$$h_t = LSTM(x_t, h_{t-1}) \tag{1}$$

$$h_t^{bi} = concat(h_t^{for}, h_t^{back}) \tag{2}$$

$$H^S[t] = h_t^{bi} \tag{3}$$

Function 1 $SimiScore(\vec{A}, \vec{B}, \vec{R})$

```
$conC = concat(\vec{A}, \vec{B})$
$entityC = W^C \cdot conC$
$relationT = W^D \cdot \vec{R}$
$dist = W^{di} \cdot tanh(entityC + relationT)$
return $dist$
```

Context modeling with BiLSTM allows our following model components to be built over contexts rather than over individual words. Given named entity positions of the sentence, we get $\vec{A}$ and $\vec{B}$ from context H^S.

5.2 Relational Similarity Modeling

Relational similarity modeling focuses on interactions between named entities and relations in the vector space. When the named entity $\vec{A}$ goes through a transformation process induced by the relation $\vec{R}$, our intent of relational similarity modeling is to force the transformed entity to be translated to the other named entity $\vec{B}$ in the same vector space so that the relation $\vec{R}$ holds between the two named entities.

We show the following objective function of our relational similarity modeling:

$$\vec{A} + \vec{B} - \vec{R} \simeq 0 \tag{4}$$

To model the transformation process in Equation 4, we need to know how to measure the similarity of the triplet $(\vec{A}, \vec{B}, \vec{R})$. Therefore we develop a similarity measurement function $SimiScore(\vec{A}, \vec{B}, \vec{R})$ with learnable weights (W^*), the similarity function takes an input named entity pair of $(\vec{A}, \vec{B})$ and a relation $\vec{R}$, returns a similarity score $dist \in R^1$ representing how semantically close $(\vec{A}, \vec{B}, \vec{R})$ are, as in Function 1.

We utilize a ranking approach during training to incorporate the constraint of Equation 4 into the relational similarity model. Our goal is to learn a function $SimiScore(\cdot)$ so that the positive triplet $(\vec{A}, \vec{B}, \vec{R^+})$ is assigned a larger score than that of the negative triplet $(\vec{A}, \vec{B}, \vec{R^-})$:

$$SimiScore(\vec{A}, \vec{B}, \vec{R^+}) > SimiScore(\vec{A}, \vec{B}, \vec{R^-}) \tag{5}$$

where R^+ denotes the positive causality/correlation relation, R^- denotes a non-causality/non-correlation relation. The ranking approach maximizes the similarity score between the entity pair $(\vec{A}, \vec{B})$ and a positive relation $\vec{R}^+$ while minimizing the score with the negative $\vec{R}^-$,

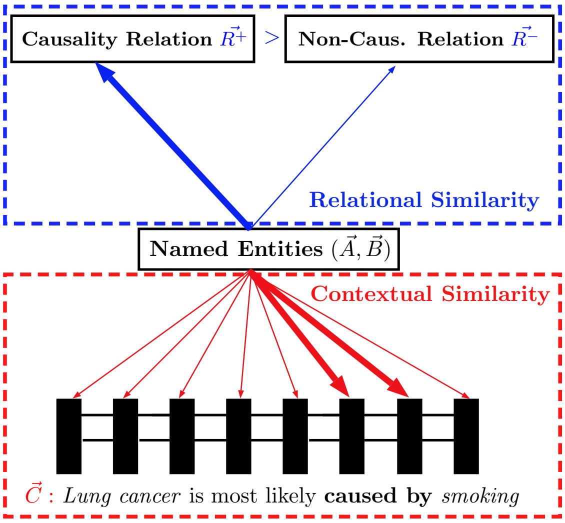

Causality Relation $\vec{R}^{+}$
>
Non-Caus. Relation $\vec{R}^{-}$
Relational Similarity
Named Entities $(\vec{A}, \vec{B})$
Contextual Similarity
$\vec{C}$: *Lung cancer* is most likely **caused by** *smoking*

Figure 2: Our causality/correlation relation extraction component models both relational similarity (blue) and contextual similarity (red). Thicker arrows indicate stronger similarity between named entities $(\vec{A}, \vec{B})$ and relation $\vec{R}$/sentence context.

thus ensuring that the positive connection is larger than the negative one as in Figure 2.

Our relational similarity model and the ranking training approach facilitate the transformation process of ($\vec{A}$, $\vec{B}$) and $\vec{R}$ in the vector space, which in the end leads to better constraint satisfaction of objective Equation 4.

The relational similarity model is placed on top of BiLSTM (Sec 5.1) as part of the system. We initialize named entities $\vec{A}/\vec{B}$ as h_A^{bi}/h_B^{bi} from the BiLSTM model, then initialize relation representations $\vec{R^+}/\vec{R^-}$ as random vectors. During training both $\vec{R^+}/\vec{R^-}$ are updated.

5.3 Contextual Similarity Modeling

Since not all words of a given title/abstract are created equal, important context words around named entities that can better contribute to the causality/correlation relation extraction deserve more model focus. We develop a contextual similarity model that can increase model weights onto important context words to better utilizing contextual information.

For example, given a sentence, *lung cancer is most likely caused by smoking*, the context words *caused by* are important clues to suggest there exists a causality/correlation relationship between the two named entities. Clue words that require model attentions usually include, e.g. *lead to, is associated with, because of*, while others are not obvious, such as *promote, reflect, reduce, make*.

Our system does not require a manually prepared list of clue words, but an attention mechanism (Bahdanau et al., 2014) is utilized to better identify them by conducting similarity measurement between context word representation h_t^{bi} (not including entity words) and extracted named entities $(\vec{A}, \vec{B})$ (from Sec. 4). Resulting similarity scores of words are accumulated in $atten \in \mathbb{R}^{\ell}$.

$$mix = W^a \cdot concat(\vec{A}, \vec{B}) \tag{6}$$

$$E[t] = dotProd(mix, h_t^{bi}), \forall t \in [1, l] \tag{7}$$

$$atten = softmax(E) \tag{8}$$

where we concatenate both entity representations $(\vec{A}, \vec{B})$, apply linear transformation with weights W^a to obtain a representation mix of both entities. We then use dot product $dotProd$ to measure the similarity between mix and each context word, finally normalize the attention weights $atten[:]$ with $softmax$. The weights of $atten$ indicate the importance of each context word with respect to both named entities.

The attention weights should better guide the focus of the model onto important context words of the sentence. That is, context words that are closer to entity representation mix should have better chances to be clue words. We define the attention re-weighted sentence representation $attenSen \in \mathbb{R}^{2dim}$:

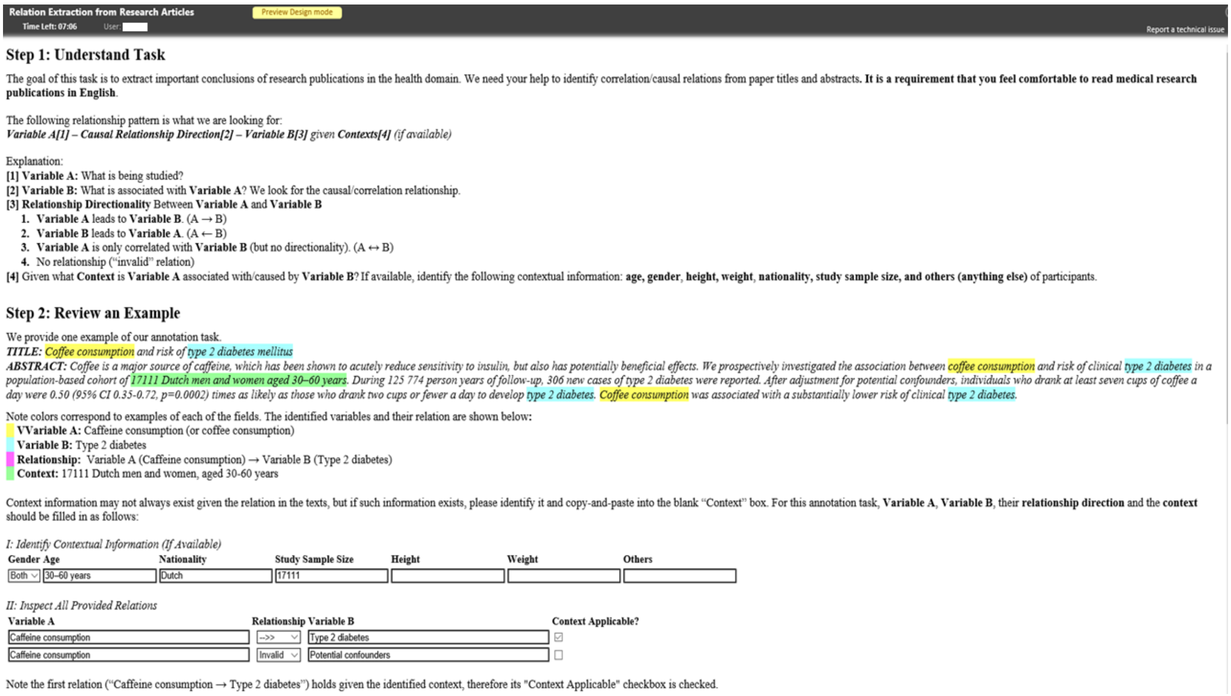
$$attenSen = atten \odot H^S \tag{9}$$

where $\odot$ represents element-wise multiplication.

Figure 2 illustrates an example where representation mix of named entities attends to context words one at a time. Important context clue words "*caused by*" should receive higher attention weights than irrelevant neighbor words.

The re-weighted sentence representation $attenSen$ is used together with entity representations $(\vec{A}, \vec{B})$ for final prediction.

In summary, both models described in this section focus on different aspects of similarity measurement in relation extraction: the contextual similarity model utilizes context information around named entities, while the relational similarity model focuses on enforcing a transformation constraint between entities and relation in the vector space. We adopt both models for better relation extraction, in the end only pairs of named entities that are recognized positively by either one of the models are passed to the next stage of the system.



Step 1: Understand Task

The goal of this task is to extract important conclusions of research publications in the health domain. We need your help to identify correlation/causal relations from paper titles and abstracts. **It is a requirement that you feel comfortable to read medical research publications in English**.

The following relationship pattern is what we are looking for:
Variable A[1] – Causal Relationship Direction[2] – Variable B[3] *given* ***Contexts[4]*** *(if available)*

Explanation:
[1] Variable A: What is being studied?
[2] Variable B: What is associated with **Variable A**? We look for the causal/correlation relationship.
[3] Relationship Directionality Between **Variable A** and **Variable B**

1. **Variable A** leads to **Variable B**. (A → B)
2. **Variable B** leads to **Variable A**. (A ← B)
3. **Variable A** is only correlated with **Variable B** (but no directionality). (A ↔ B)
4. No relationship (“invalid” relation)

[4] Given what **Context** is **Variable A** associated with/caused by **Variable B**? If available, identify the following contextual information: **age, gender**, **height, weight**, **nationality, study sample size, and others (anything else)** of participants.

Step 2: Review an Example

We provide one example of our annotation task.
TITLE: *Coffee consumption and risk of type 2 diabetes mellitus*
ABSTRACT: *Coffee is a major source of caffeine, which has been shown to acutely reduce sensitivity to insulin, but also has potentially beneficial effects. We prospectively investigated the association between coffee consumption and risk of clinical type 2 diabetes in a population-based cohort of 17111 Dutch men and women aged 30–60 years. During 125 774 person years of follow-up, 306 new cases of type 2 diabetes were reported. After adjustment for potential confounders, individuals who drank at least seven cups of coffee a day were 0.50 (95% CI 0.35-0.72, p=0.0002) times as likely as those who drank two cups or fewer a day to develop type 2 diabetes. Coffee consumption was associated with a substantially lower risk of clinical type 2 diabetes.*

Note colors correspond to examples of each of the fields. The identified variables and their relation are shown below:
🟨 **VVariable A:** Caffeine consumption (or coffee consumption)
🟦 **Variable B:** Type 2 diabetes
🟪 **Relationship:** Variable A (Caffeine consumption) → Variable B (Type 2 diabetes)
🟩 **Context:** 17111 Dutch men and women, aged 30-60 years

Context information may not always exist given the relation in the texts, but if such information exists, please identify it and copy-and-paste into the blank “Context” box. For this annotation task, **Variable A**, **Variable B**, their **relationship direction** and the **context** should be filled in as follows:

I: Identify Contextual Information (If Available)

Gender	Age	Nationality	Study Sample Size	Height	Weight	Others
Both	30–60 years	Dutch	17111			

II: Inspect All Provided Relations

Variable A	Relationship	Variable B	Context Applicable?
Caffeine consumption	-->>	Type 2 diabetes	☑
Caffeine consumption	Invalid	Potential confounders	☐

Note the first relation (“Caffeine consumption → Type 2 diabetes”) holds given the identified context, therefore its "Context Applicable" checkbox is checked.

Figure 3: Human annotation interface on UHRS platform. Annotators are required to identify and verify extracted entities and correlation/causality relations from the output of our system for evaluation.

6 Ranking of Extracted Insights

The last major component of our system is to rank extracted relations $(\vec{A}, \vec{B}, \vec{R})$ from the output of the relation extraction component, as there could be many extracted relations but not all of them are important enough as insights of the article. Importance scores of extracted relations are obtained by following a set of rules below:

1. We utilize the output classification probability $(\in [0, 1])$ of the relational similarity model as the base ranking score.
2. We use a multi-perspective convolutional neural network model (MPCNN) (He et al., 2015) to measure the similarity $(\in [0, 1])$ between the title of the article and extracted relation, since the MPCNN model has competitive performance on multiple benchmarks for textual similarity measurement. We compare title text with “$\vec{A}$ leads to $\vec{B}$” of an extracted relation, if the similarity score is over a threshold of 0.75, we increase the extracted relation’s ranking score by 15%. If the extracted relation is from the title text, we also boost its ranking score by 15% because of its location importance.

Once all extracted relations are scored, our system only returns the top ranked insights to users.

7 Experiment Setup

Datasets. Experiments are conducted on two datasets: our own dataset of medical/health publications annotated on Universal Human Relevance System (UHRS), a crowdsourcing platform for end-to-end system evaluation; and SemEval-2010 task 8 dataset for training and evaluation of our relation extraction component:

1. The first dataset consists of 100 publications from recent biomedical/health journals, which are then annotated on UHRS to evaluate our system. In order to ensure high-quality human annotations, Figure 3 provides an annotation interface on UHRS, which displays instructions, title/abstract texts of publications and a list of top ranked extracted insights from the system output. For fair evaluation the order of extracted insights is randomized then we ask expert annotators with suitable background to verify the correctness of each.
2. SemEval-2010 Task 8 (Hendrickx et al., 2009) defines 9 relation types between named entities: *Cause-Effect, Instrument-Agency, Product-Producer, Content-Container, Entity-Origin, Entity-Destination, Component-Whole, Member-Collection and Message-Topic*, and a tenth relation type *Other* when two named enti-

ties do not have the first 9 relations. SemEval-2010 dataset consists of $10,717$ sentences, with $8,000$ for training and $2,717$ for test. The dataset is human annotated, and each instance provides one sentence which includes two named entities and a relation type between the two entities.

Since our system focuses on extracting insights, we only use *Cause-Effect* subset of SemEval-2010 dataset as the positive training/testing examples and treat the remaining 9 categories data such as *Content-Container, Message-Topic* as negatives. We use this dataset for training and evaluating our relation extraction component (Sec. 5) only.

Training. Two loss functions are adopted to train relation extraction neural network models.

For contextual similarity model (Sec. 5.3), a hinge loss is used. The training objective is to minimize the following loss, summed over examples $\langle x, y_{gold} \rangle$:

$$loss_{contextSim}(w, x, y_{gold}) = \sum_{y' \neq y_{gold}} \max(0, 1 + f_w(x, y') - f_w(x, y_{gold})) \quad (10)$$

where input x represents an entity pair $(\vec{A}, \vec{B})$ plus its sentence context, y_{gold} is the ground truth label and y' is the model predicted label. Both y' and y_{gold} indicate the relation type with directionality (e.g. directional causality). w represents weights of contextual similarity model with BiLSTM, function $f_w(x, y')$ outputs the model predicted label value, function $f_w(x, y_{gold})$ outputs the model ground truth label value, and n is the number of training examples.

For relational similarity model (Sec. 5.2), a Bayesian Personalized Ranking (BPR) loss (Rendle et al., 2009) is used. The label of the relational similarity model is binary because the BPR loss ranks positive inputs above negative inputs, thereby requiring the supervision signal to distinguish positives from negatives. Due to BPR loss's ranking nature, each training instance of the relational similarity model include one positive input $(x, \vec{R^+})$ and one negative input $(x, \vec{R^-})$. Given a positive correlation/causality input $(\vec{R^+})$, we generate negative training examples by matching the input x with each of the negative relation labels $(\vec{R^-})$. BPR loss is shown to be better tailored for ranking tasks empirically (Verga et al., 2016):

$$loss_{relationSim}(w, x, \vec{R^+}, \vec{R^-}) = \sum_{\vec{R^-}} - \log(\sigma(f'_w(x, \vec{R^+}) - f'_w(x, \vec{R^-}))) \quad (11)$$

where σ is the sigmoid function, function $f'_w(x, \vec{R})$ represents the relational similarity model with BiLSTM, and outputs a similarity score for ranking purpose (Sec. 5.2).

In all experiments, we perform optimization using RMSProp (Tieleman and Hinton, 2012) with backpropagation (Bottou, 1998) and a learning rate fixed to 10^{-4} and a momentum parameter 0.9.

Settings and Preprocessing. We preprocess both datasets with Stanford CoreNLP toolkit (Manning et al., 2014). We tokenize, lowercase, sentence split and dependency parse all words of both datasets. We set LSTM hidden state $dim = 500$.

Two sets of $d = 300$-dimension word embeddings are utilized. The first one is 300-dimension GloVe word embeddings (Pennington et al., 2014) trained on 840 billion tokens; for better biomedical/health domain adaptation, we also train second word embeddings using the GloVe toolkit on biomedical research articles with over 1 billion tokens. We do not update word embeddings in all experiments.

During system deployment, we only initialize input words with the medical word embeddings if they do not exist in GloVe embeddings' vocabulary. We also concatenate embeddings of both input words and their head words on dependency trees as input for relation extraction models. We follow the task settings and compute F1-score with the official evaluation script only on *Cause-Effect* subset of SemEval-2010 data, then the best model based on F1 is selected for final system deployment. We set a distance limit and do not extract relations between two named entities if the distance is larger than 15.

8 Evaluation and Results

Human Evaluation of the Entire System. We firstly provide a full end-to-end evaluation of the system on UHRS with human annotators.

For each biomedical publication, top 10 candidate insights from the system are listed for further inspection. The annotators are required to understand the texts, carefully inspect each insight, finally either accept it if it is one of the article insights or simply reject it. The annotation

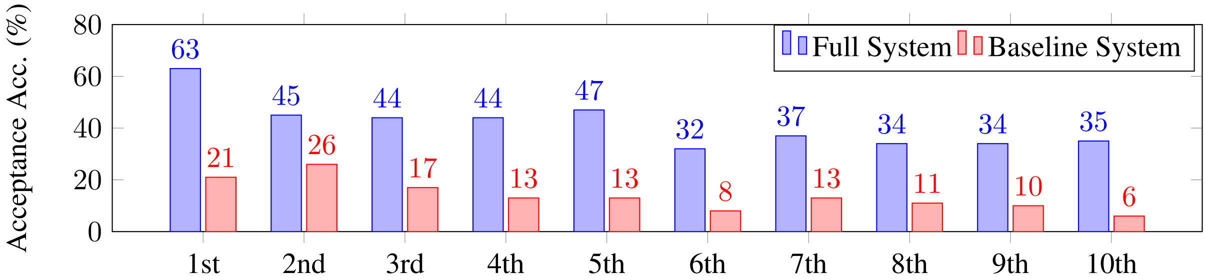


Figure 4: Human evaluation results of the full system and a baseline system on UHRS. We show the acceptance accuracy for each of the top ten positions given both systems' output lists. We primarily focus on the first 1 and 3 positions, namely *Precision*@1 and *Precision*@3.

System Ablation Study	*Precision*@1
Full System	63%
- Remove ReRanker (Sec. 6)	-5%
- Replace with BiGRU (Sec. 5)	-42%

Table 1: Ablation studies on the full system.

task requires understanding of biomedical/health publications and is non-trivial, therefore the system evaluation is completed by five expert annotators, who all hold postgraduate degrees and/or have biomedical background.

We also provide a baseline system, of which its relation extraction component is a bidirectional gated RNN model (BiGRU) (Cho et al., 2014). BiGRU model and the ranking component are major differences between our full and baseline system. Since typically only a limited number of key findings is presented in one article, we evaluate the system with an averaged acceptance accuracy at top 1 (*Precision*@1) and top 3 (*Precision*@3) positions of the output rank list, which represent on average the number of extracted insights accepted by annotators among the first 1 and 3 output.

Figure 4 shows annotation results with acceptance accuracies for each of the ten output positions given biomedical titles and articles. The *Precision*@3 of our full system is 50.6%, which is significantly better than the baseline system's 21.3%. For top 3 extracted insights on the list, our full system on average have 1.5 insights accepted by annotators. Furthermore, the acceptance accuracy *Precision*@1 of our system is 63% in comparison to that of the baseline system's 21%.

Table 1 shows the ablation study on the removal of the ranking component (Sec. 6) and the replacement of BiGRU model for causality/correlation relation extraction. We observe significant performance difference.

Model	F1 score⋆
Tymoshenko and Giuliano (2010)	82.30%
Tratz and Hovy (2010)	87.63%
Rink and Harabagiu (2010)	89.63%
BiGRU	89.89%
Miwa and Bansal (2016)	91.57%
Contextual similarity modeling	90.77%
Relational similarity modeling	**92.28%**

Table 2: Test results (F1 score) on the *Cause-Effect* subset(⋆) of SemEval-2010 dataset. Results are grouped as 1) Top 3 participating teams in SemEval-2010 competition; 2) Baseline BiGRU model; 3) Recent state-of-the-art treeLSTM model (Miwa and Bansal, 2016); 4) Our work.

Evaluation of Relation Extraction Component. We also evaluate the relation extraction component (Sec. 5) on *Cause-Effect* subset of SemEval-2010 dataset. Note our causality/correlation relation extraction component is *not* supposed to be a general purpose one, since our system only focuses on insight extraction of biomedical/health literature. We compare our relation extraction models against previous work on the *Cause-Effect* subset of the data, Table 2 shows our relational similarity model, without the use of sparse features or external resources such as WordNet, outperforms recent state-of-the-art treeLSTM model (Miwa and Bansal, 2016). It also shows BiGRU model is reasonably competitive on this dataset, which is why we use it in our baseline system for comparison purpose.

9 Result Analysis and Case Study

Visualization of Contextual Similarity Model. We show values of attention weights, *atten* of

Excess	oil	,	**dirt**	and	bacteria	cause	**acne**	.	
0	0	0	0	0	0	0.9962	0	0.0037	
The	**bombing**	resulted	in	the	**deaths**	of	1318	in	Hanoi
0	0	0.0005	0.9579	0.0415	0	0	0	0	0
Ambient	vanadium	pentoxide	**dust**	produces	**irritation**	of	the	eyes	...
0	0	0	0	0.99	0	0	0	0	...
Electron	**beam**	is	generated	by	an	explosive	**emission**	cathode	
0	0	0	0.0053	0.9946	0	0	0	0	

Table 3: Visualization of model attention weights $atten$ given four SemEval-2010 test sentences.

Equation 8 and 9 from within the contextual similarity model (Sec. 5.3). Given four sentences in the test set of SemEval-2010 data, the model predicts that all provided entity pairs (in bold) have the causality/correlation relation. From Table 3 we observe the model is able to do its expected job: it can recognize important clues words, such as "*result in*", "*produce*", "*generated by*" and "*cause*"; the model produces attention weights (each $\in [0, 1]$) to tell the importance of clue words for causality/correlation relation extraction. We also observe the model tends to focus more on prepositions of clue words, such as "*by*" of "*generated by*" and "*in*" of "*result in*", this is probably because we use head words as extra inputs (Sec. 7) to the model.

Case Study. We lastly provide case study of our system. We show two biomedical articles' titles and abstracts as examples, with only necessary omissions to remove irrelevant texts due to the space limit.

Given **Case 1**, our system outputs the top insight "*the slow negative shift of the DC potential → increased cortical excitability*" with a score of 0.71. Given **Case 2**, our system outputs top 3 insights: "*excessive drinking → skin cancer*" with a score of 0.55, "*excessive drinking → alcohol*" with a score of 0.43, and "*excessive drinking → sunburn*" with a score of 0.31. The above examples show that our system can provide reasonable insights from biomedical text.

10 Conclusion

We build an end-to-end system for insight extraction on biomedical literature. We develop novel similarity measurement modeling with deep neural networks to extract causation/correlation relations. Our evaluation shows the system is able to extract insights with competitive human acceptance accuracy and its relation extraction component compares favorably against previous work.

Case 1: *Scalp recorded direct current potential shifts associated with the transition to sleep in man.* ***Abstract:*** *Cortical direct current (DC) potentials are considered to reflect the state of cortical excitability which may change characteristically from wakefulness to sleep. The present experiments examined changes in the scalp recorded DC potential in 10 healthy humans ... It is reasonable to assume that the slow negative shift of the DC potential at the transition from wakefulness to sleep reflects increased cortical excitability.*

Case 2: *Alcohol consumption and self-reported sunburn: a cross-sectional, population-based survey.* ***Abstract:*** *Heavy drinking has been associated with several cancers, including melanoma and basal cell carcinoma. ... 299,658 adults reported their use of alcohol in the preceding month and a history of sunburn in the preceding year. Approximately 33.5% of respondents reported a sunburn within the past year. ... Excessive drinking is associated with higher rates of sunburn among American adults. The observed relationship typifies the high-risk behavior associated with excessive drinking and suggests one pathway linking alcohol use with skin cancer.*

Example 2: Case Study

Acknowledgments

We thank Yifeng Liu, Scott Wen-tau Yih for insightful discussions, and Max Ma for engineering support. We also thank three anonymous reviewers for highly helpful reviews.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Olivier Bodenreider. 2004. The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32:D267.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Léon Bottou. 1998. Online learning and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press.

Elizabeth S. Chen, George Hripcsak, Hua Xu, Marianthi Markatou, and Carol Friedman. 2008. Automated acquisition of diseasedrug knowledge from biomedical and clinical documents: An initial study. *Journal of the American Medical Informatics Association*, 15(1):87.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Mark Craven. 1999. Learning to extract relations from medline. In *AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 25–30.

Rezarta Islamaj Dogan, Aurélie Névéol, and Zhiyong Lu. 2011. A context-blocks model for identifying clinical relationships in patient records. *BMC Bioinformatics*, 12(S-3):S3.

Jenny Finkel, Shipra Dingare, Christopher D. Manning, Malvina Nissim, Beatrice Alex, and Claire Grover. 2005. Exploring the boundaries: gene and protein identification in biomedical text. *BMC Bioinformatics*.

Carol Friedman, Lyudmila Shagina, Yves A. Lussier, and George Hripcsak. 2004. Automated encoding of clinical documents based on natural language processing. *Journal of the American Medical Informatics Association*, 11:392–402.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948, San Diego, California. Association for Computational Linguistics.

Hua He, John Wieting, Kevin Gimpel, Jinfeng Rao, and Jimmy Lin. 2016. UMD-TTIC-UW at SemEval-2016 task 1: Attention-based multi-perspective convolutional neural networks for textual similarity measurement. In *SemEval*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

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

Abhyuday N. Jagannatha and Hong Yu. 2016. Bidirectional RNN for medical event detection in electronic health records. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 473–482, San Diego, California. Association for Computational Linguistics.

Renata Kabiljo, Andrew B. Clegg, and Adrian J. Shepherd. 2009. A realistic assessment of methods for extracting gene/protein interactions from free text. *BMC Bioinformatics*.

Yuxi Li. 2017. Deep reinforcement learning: An overview. *CoRR*, abs/1701.07274.

Yifeng Liu. 2016. Question answering for biomedicine. PhD Dissertation, University of Alberta.

Yifeng Liu, Yongjie Liang, and David Wishart. 2015. PolySearch2: a significantly improved text-mining system for discovering associations between human diseases, genes, drugs, metabolites, toxins and more. *Nucleic Acids Research*, 43(W1):W535–W542.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd*

Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 55–60.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

John Paparrizos, Ryen W. White, and Eric Horvitz. 2016. Detecting devastating diseases in search logs. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 559–568.

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

Hoifung Poon, Chris Quirk, Charlie DeZiel, and David Heckerman. 2014. Literome: PubMed-scale genomic knowledge base in the cloud. *Bioinformatics*, 30(19):2840.

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 813–821, Los Angeles, California. Association for Computational Linguistics.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1913–1916, New York, NY, USA. ACM.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States. AUAI Press.

Thomas C. Rindflesh and Marcelo Fiszman. 2003. The interaction of domain knowledge and linguistic structure in natural language processing: Interpreting hypernymic propositions in biomedical text. *J. of Biomedical Informatics*, 36(6):462–477.

Bryan Rink and Sanda Harabagiu. 2010. UTD: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 256–259, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew S. Simpson and Dina Demner-Fushman. 2012. Biomedical text mining: A survey of recent progress. In *Mining Text Data*, pages 465–517, Boston, MA. Springer US.

Tijmen Tieleman and Geoffrey E. Hinton. 2012. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. Coursera: Neural Networks for Machine Learning.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1499–1509.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Stephen Tratz and Eduard Hovy. 2010. ISI: Automatic classification of relations between nominals using a maximum entropy classifier. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 222–225, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kateryna Tymoshenko and Claudio Giuliano. 2010. FBK-IRST: Semantic relation extraction using cyc. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 214–217, Stroudsburg, PA, USA. Association for Computational Linguistics.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 886–896.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280.

Adam Yala, Regina Barzilay, Laura Salama, Molly Griffin, Grace Sollender, Aditya Bardia, Constance Lehman, Julliette M Buckley, Suzanne B Coopey, Fernanda Polubriaginof, Judy E Garber, Barbara L Smith, Michele A Gadd, Michelle C Specht, Thomas M Gudewicz, Anthony Guidi, Alphonse Taghian, and Kevin S Hughes. 2016. Using machine learning to parse breast pathology reports. *bioRxiv*.

Word Etymology as Native Language Interference

Vivi Nastase
University of Heidelberg
Heidelberg, Germany
nastase@cl.uni-heidelberg.de

Carlo Strapparava
Fondazione Bruno Kessler
Trento, Italy
strappa@fbk.eu

Abstract

We present experiments that show the influence of native language on lexical choice when producing text in another language – in this particular case English. We start from the premise that non-native English speakers will choose lexical items that are close to words in their native language. This leads us to an etymology-based representation of documents written by people whose mother tongue is an Indo-European language. Based on this representation we grow a language family tree, that matches closely the Indo-European language tree.

1 Introduction

In second-language writing, vestiges of the native language such as pronunciation, vocabulary and grammar are well-attested, and the phenomenon is called native language interference (Odlin, 1989). At the lexical level, the choice as well as the spelling can be indicative of the native language, through the choice of cognates, true or false friends – e.g. a writer with native language German may choose *bloom* cognate with *blume*, while a French one may choose *flower*, cognate with *fleur*. Misspellings – *cuestion* instead of *question* are also indicative, as the writer will tend to spell words close to the form from her original language (Nicolai et al., 2013).

In this paper we also look at native language interference starting from the lexical level, but abstract away from the actual word forms, and focus instead on the language of the etymological ancestors. The hypothesis we investigate is that the collective evidence of etymological ancestor languages are indicative of the language of the native speaker, and that this effect is sufficiently strong to allow us to rebuild an Indo-European language family tree. We use a corpus of essays written by English language learners, whose native language cover the languages from the Indo-European family. Etymological information is provided by an etymological dictionary extracted from Wiktionary.

Eng.: *flower*	Eng.: *bloom*	Eng.: *blossom*
↓	↓	↓
Middle Eng.: *flour*	Middle Eng.: *blome*	Middle Eng.: *blosme*
↓	↓	↓
Anglo Norm.: *flur*	Old Norse: *blōm*	Old Eng.: *blostm*
↓	↓	↓
Latin: *florem*	Proto Ger.: **blōmô*	Proto Ger.: **blōstama*
↓	↓	↓
Proto IE: $*b^hleh_3$	Proto IE: $*bleh_3$	Proto IE: $*b^hleh_3 - s-$

Figure 1: Examples of etymological ancestry from Wiktionary

The fact that word etymologies are included in the native-language interference phenomena could be used in various ways, i.a.: (i) to influence the selection of material for language learning, by focusing on vocabulary closer etymologically to the native language of the student, thus facilitating lexical retention; (ii) to reveal lexical choice errors caused by "etymological interference"; (iii) together with other interference phenomena, for the automatic corrections of language errors.

2 Related Work

English is a widespread common language for communication in a variety of fields – science, news, entertainment, politics, etc. A consequence is that numerous people learn English as a second (or indeed nth language). The study of native language interference with the learning of English can be used in multiple ways, including devising methods to make the learning easier and correcting language errors (Leacock et al., 2014; Gamon, 2010; Dahlmeier and Ng, 2011).

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2702–2707
Copenhagen, Denmark, September 7–11, 2017.

Massung and Zhai (2016) present an overview of approaches to the task of natural language identification (NLI). Various surface indicators hold clues about a speaker's native language, that make their way into language production in a non-native language. Nagata and Whittaker (2013),Nagata (2014) find that grammatical patterns from the native language seep into the production of English texts. Tsur and Rappoport (2007) verify the hypothesis that lexical choice of non-native speakers is influenced by the phonology of their native language, and Wong and Dras (2009) propose the idea that (grammatical) errors are also influenced by the native language. One could draw the inference that character n-grams then could be indicative of the native language, and this was shown to be the case by Ionescu et al. (2014).

The natural language identification task (Tetreault et al., 2013) attracted 29 participating teams, which used a variety of features to accomplish the NLI task as a classification exercise: n-grams of lexical tokens (words and POS tags), skip-grams, grammatical information (dependency parses, parse tree rules, preference for particular grammatical forms, e.g. active or passive voice), spelling errors.

Apart from morphological, lexical, grammatical features, words also have an etymological dimension. The language family tree itself is drawn based on the analysis of the evolution of languages. Language evolution includes, or starts with, word etymologies. Word etymologies have been under-used for tasks related to NLI. They have been used implicitly in work that investigates cognate interference (Nicolai et al., 2013), and explicitly by (Malmasi and Cahill, 2015) who use words with Old English and Latin etymologies as unigram features in building classifiers for the TOEFL11 dataset. Etymological information is obtained from the Etymological WordNet (de Melo and Weikum, 2010).

We also investigate here the impact of etymological information, but unlike previous work, we do not extract unigram/n-gram features for classification, but we look at the collective evidence captured by the etymological "fingerprint" for each document and set of essays.

3 Etymological fingerprints

To investigate the influence of etymological ancestor languages, we represent each essay through etymological features, based on which we also built language vectors for each Indo-European language represented in the corpus. Essay vectors are then used to test native language identification potential, and the language vectors are used to grow a language family tree.

3.1 Word etymologies

Dictionaries customarily include etymological information for their entries. Wikipedia's Wiktionary has amassed a considerable number of entries that joined this trend. The etymological information can, and indeed has been extracted and prepared for machine consumption (de Melo and Weikum, 2010): Etymological WordNet[1] contains 6,031,431 entries for 2,877,036 words (actually, morphemes) in 397 languages. Because we process essays written in English, we use only the entries that give etymological origins for English words – 240,656. Figure 1 shows as an example of the kind of information formalized in the Etymological WordNet the etymological ancestry for the words *flower, blossom, bloom*.

3.2 Document/collection modeling

Essay representation After tokenization, POS tagging and lemmatization, each essay in the dataset is represented as a vector of etymological ancestry proportions obtained through the following processing steps:

1. for each token that has an entry in the etymological dictionary, we replace it with the language of its etymological ancestor – e.g. *sight* will be replaced by *ang* (Old English), *vision* by *lat* (Latin) (Table 1 shows the number of words with etymological ancestors in the subsets corresponding to each language);

2. compute the proportion of each etymological language in the essay, and represent the essay as a vector of language proportions[2]. We experimented with using etymological information going back through several ancestry levels, but using the first level led to the best results.

$$e_i \sim \langle p_{il_1}, ..., p_{il_n} \rangle \quad \textit{where} \quad p_{il_k} = \frac{n_{il_k}}{N_{i,etym}}$$

[1] http://www1.icsi.berkeley.edu/~demelo/etymwn/

[2] Using automatically corrected typos (first option of ispell) did not change the results significantly.

n_{il_k} is the number of words with etymological ancestor l_k in the i-th essay, and $N_{i,etym}$ is the number of words with etymological information in essay i.

Language vectors For each subcollection corresponding to one (student native) language L_j, we build the language vectors by averaging over the essay vectors in the subcollection:

$$V_{L_j} = \langle p_{L_j l_1}, ..., p_{L_j l_m} \rangle$$

$$\textit{where} \quad p_{L_j l_k} = \frac{\sum\limits_{lang(e_i)=L_j} p_{il_k}}{|\{e_i | lang(e_i)=L_j\}|}$$

$p_{L_j l_k}$ is the proportion of etymological ancestor language l_k in all essays whose author has as native language L_j.

The essay and language vectors are filtered by removing etymological languages whose corresponding values in the language vectors are less than 10^{-4}.

4 Experiments

We investigate the strength of the etymological "fingerprint" of individual and collective essays written by non-native speakers of English, through two tasks – native language identification and language family tree construction. Towards this end, we work with a collection of essays written by contributors whose native language is an Indo-European language. The dataset is described in Section 4.1. For etymological information we rely on an etymological dictionary, described briefly in Section 3.1. Data modeling and the experiments conducted are described in Section 3.2.

4.1 Data

We used the ICLE dataset (Granger et al., 2009), consisting of English essays written by non-native English speakers. We filter out those that were written by people whose mother tongue is not from the Indo-European family (i.e. Chinese, Japanese, Turkish and Tswana). Table 1 shows a summary of the data statistics, including the number of words for which we have found ancestors in the etymological dictionary used. The corpus consists entirely of essays written by students. Two types of essay writing are present: argumentative essay writings and literature examination papers. Table 2 displays a list of topics in the corpus. The essays should be at least 500 words long and up to 1,000, and contain all the spelling mistakes made by their authors.

Following Nagata and Whittaker (2013), who also built the Indo-European family tree based on n-grams composed of function words and open-class parts of speech, essays that do not respect one of the following rules are filtered out: (i) the writer has only one native language, (ii) the writer has only one language at home; (iii) the two languages in (i) and (ii) are the same as the native language of the subcorpus to which the essay belongs. Table 1 shows a summary of the data statistics after filtering, including the number of words for which we have found ancestors in the etymological dictionary used.

Native language	*# essays*	*# tokens (with etym)*
Bulgarian	302	226,407 (149,151)
Czech	243	226,895 (148,391)
Dutch	263	264,981 (169,040)
French	347	256,749 (161,136)
German	437	259,967 (170,056)
Italian	392	253,798 (165,500)
Norwegian	317	238,403 (156,764)
Polish	365	263,223 (172,319)
Russian	276	259,510 (167,938)
Spanish	251	225,341 (139,565)
Swedish	355	224,948 (146,143)

Table 1: Statistics on the subset of ICLE dataset used.

1 Crime does not pay.
2 The prison system is outdated. No civilized society should punish its criminals: it should rehabilitate them.
3 Most university degrees are theoretical and do not prepare students for the real world. They are therefore of very little value.
4 A man/woman's financial reward should be commensurate with their contribution to the society they live in.
5 The role of censorship in Western society.
6 Marx once said that religion was the opium of the masses. If he was alive at the end of the 20th century, he would replace religion with television.
7 All armies should consist entirely of professional soldiers : there is no value in a system of military service.
8 The Gulf War has shown us that it is still a great thing to fight for one's country.
9 Feminists have done more harm to the cause of women than good.
10 In his novel Animal Farm, George Orwell wrote "All men are equal: but some are more equal than others". How true is this today?
11 In the words of the old song "Money is the root of all evil".
12 Europe.
13 In the 19th century, Victor Hugo said: ”How sad it is to think that nature is calling out but humanity refuses to pay heed. ”Do you think it is still true nowadays ?
14 Some people say that in our modern world, dominated by science technology and industrialization, there is no longer a place for dreaming and imagination. What is your opinion ?

Table 2: Topics in the ICLE dataset.

The suitability of the dataset above for NLI was questioned by Brooke and Hirst (2012). They have shown that the fact that the corpus consists of sets of essays on a number of topics causes an overestimation of the results of NLI when random splitting, particularly for groups of contributors that were presented with very different topics – e.g. students from Asia vs. students from Europe. We have analyzed the distribution of essays into topics using the essay titles, and observed that contributions from Europe (which are our focus) have similar distributions across the featured topics.

Growing the language tree To grow the language tree from the language vectors built from the English essays, we use a variation of the UPMGA – Unweighted Pair Group Method with Arithmetic Mean – algorithm. Starting with the language vectors V_{L_j}, we compute the distance between each pair of vectors using a distance metric algorithm. At each step we choose the closest pair (L_a, L_b) and combine them in a subtree, then combine their corresponding sub-collection of essays, and build the language vector for the "composite" language $L_{a,b}$, and compute its distance to the other language vectors.

4.2 Results

We test whether etymological information surfaces as native language interference that is detectible through the tasks of native language identification and reconstruction of the language family tree. Table 3 shows results on the multi-class classification of essays according to the native language of the author, in the form of F-score average results using SVM classification in 5-fold cross-validation (using Weka's SMO implementation[3] with polynomial kernel and default parameters). The baseline corresponds to the language distribution in the dataset. We use as additional comparison point another set of features used to reconstruct the language family tree – the (closed-class) word and POS 3grams Nagata and Whittaker (2013), such as *the NN of; a JJ NN; the JJ NN*. We build all such patterns for the data, and keep the top 1000 by overall frequency.

Adding etymological features that capture the distribution of etymological ancestors for each essay led to improved results for all languages, varying from a non-significant improvement of 0.2% point for Russian, to a significant and high 5.3% improvement for German. Using only words, the accuracy is 73.2%, which increases marginally to 73.7 when etymology information is added. Using a full complement of standard features – word, lemma and character ngrams (n=1..3) (built following (Lahiri and Mihalcea, 2013)) – gives an average accuracy (over 5 fold cross-validation) of 85.7%. Adding etymology does not lead to improvements when added to this set.

Language	Baseline	Etym.	Patt.	both
Bulgarian	8.52%	32.4	51.7	54.3
Czech	6.85%	21.9	53.4	54.4
Dutch	7.41%	11.7	50.4	51.1
French	9.78%	30.0	58.8	62.9
German	12.31%	45.4	47.4	52.7
Italian	11.04%	34.3	66.3	67.3
Norwegian	8.93%	35.5	57.0	59.3
Polish	10.28%	42.5	59.9	62.1
Rusian	7.78%	12.7	46.9	47.1
Spanish	7.07%	24.6	57.9	59.6
Swedish	10.00%	23.1	44.8	45.7
Accuracy		31.7	54.2	56.3

Table 3: 5-fold cross-validation F-scores and accuracy for language classification

Despite the rather low results when etymology is used on its own for language identification, the cumulative evidence leads to a language family tree that closely matches the gold standard (Figure 2). The tree on top is the gold standard cf. (Nagata and Whittaker, 2013; Crystal, 1997). The tree is grown by computing the euclidean distance between pairwise vectors, and then iteratively grouping together the closest vectors at each step as described in Section 4.1.

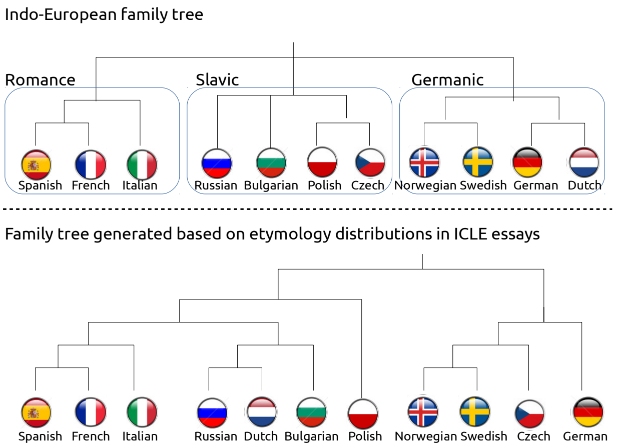


Figure 2: Language family trees – the gold standard and the automatically generated one

The two wrongly placed languages in our language family tree are Czech and Dutch. Czech

[3] http://www.cs.waikato.ac.nz/ml/weka/

is grouped with the Germanic languages. Historically, the country which is now the Czech Republic has been under German occupation for long periods of time. We propose the hypothesis that this has influenced the Czech language at the lexical level, and our etymological fingerprinting characterizes mostly the lexical aspects of language. We plan to verify this theory as etymological information for Czech and German becomes more readily available in sufficient quantities. We have not yet found an explanation for the grouping of Dutch with the Slavic languages. Like mentioned before, the language vectors we built rely exclusively on lexical information, and it is possible that Dutch's grammatical structure is what defines it best as being a part of the Germanic language family, as opposed to the lexical dimension.

5 Conclusion

In this paper we have shown an exploration of a novel indicator of native language interference in second language learning, particularly etymology. While cross-linguistically related words (cognates, false and true friends) have been part of the repertoire of features for native language identification and cross-language studies, we have focused here on the language of etymological ancestors of words, and in particular their distribution in documents. Experiments in recreating the Indo-European family tree have shown that the composition of a document in terms of etymological languages is indicative of the native language of the writer to the extent that fundamental characteristics of languages – typological relatedness between languages – emerge.

References

Julian Brooke and Graeme Hirst. 2012. Robust, lexicalized native language identification. In *Proceedings of COLING 2012*, pages 391–408, Mumbai, India. The COLING 2012 Organizing Committee.

David Crystal. 1997. *The Cambridge Encyclopedia of Language*. Cambridge University Press.

Daniel Dahlmeier and Hwee Tou Ng. 2011. Correcting semantic collocation errors with l1-induced paraphrases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 107–117, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Gamon. 2010. Using mostly native data to correct errors in learners' writing: A meta-classifier approach. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 163–171, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sylviane Granger, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2009. International Corpus of Learner English v2 (ICLE).

Radu Tudor Ionescu, Marius Popescu, and Aoife Cahill. 2014. Can characters reveal your native language? a language-independent approach to native language identification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1363–1373, Doha, Qatar. Association for Computational Linguistics.

Shibamouli Lahiri and Rada Mihalcea. 2013. Using n-gram and word network features for native language identification. In *Proc. of BEA@NAACL-HLT 2013*, pages 251–259.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2014. Automated grammatical error detection for language learners, 2nd edition. In Graeme Hirst, editor, *Synthesis Lectures on Human Language Technologies*. Morgan and Claypool.

Shervin Malmasi and Aoife Cahill. 2015. Measuring feature diversity in native language identification. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 49–55, Denver, Colorado. Association for Computational Linguistics.

Sean Massung and Chenkxiang Zhai. 2016. Non-native text analysis: A survey. *Natural Language Engineering*, 22(2):163186.

Gerard de Melo and Gerhard Weikum. 2010. Towards universal multilingual knowledge bases. In *Principles, Construction, and Applications of Multilingual Wordnets. Proceedings of the 5th Global WordNet Conference (GWC 2010)*, pages 149–156, New Delhi, India.

Ryo Nagata. 2014. Language family relationship preserved in non-native english. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1940–1949, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Ryo Nagata and Edward Whittaker. 2013. Reconstructing an indo-european family tree from non-native english texts. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1137–1147, Sofia, Bulgaria. Association for Computational Linguistics.

Garrett Nicolai, Bradley Hauer, Mohammad Salameh, Lei Yao, and Grzegorz Kondrak. 2013. Cognate and misspelling features for natural language identification. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 140–145, Atlanta, Georgia. Association for Computational Linguistics.

Terence Odlin. 1989. *Language Transfer*. Cambridge University Press.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 48–57, Atlanta, Georgia. Association for Computational Linguistics.

Oren Tsur and Ari Rappoport. 2007. Using classifier features for studying the effect of native language on the choice of written second language words. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, CACLA '07, pages 9–16, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sze-meng Jojo Wong and Mark Dras. 2009. Contrastive analysis and native language identification. In *Australasian Language Technology Association Workshop*, pages 53–61, Sydney, Australia.

A Simpler and More Generalizable Story Detector using Verb and Character Features

Joshua D. Eisenberg and **Mark A. Finlayson**
11200 S.W. 8th Street, ECS Building, Miami, FL 33141
School of Computing and Information Sciences
Florida International University
Miami, Florida
{jeise003, markaf}@fiu.edu

Abstract

Story detection is the task of determining whether or not a unit of text contains a story. Prior approaches achieved a maximum performance of 0.66 F_1, and did not generalize well across different corpora. We present a new state-of-the-art detector that achieves a maximum performance of 0.75 F_1 (a 14% improvement), with significantly greater generalizability than previous work. In particular, our detector achieves performance above 0.70 F_1 across a variety of combinations of lexically different corpora for training and testing, as well as dramatic improvements (up to 4,000%) in performance when trained on a small, disfluent data set. The new detector uses two basic types of features–ones related to events, and ones related to characters–totaling 283 specific features overall; previous detectors used tens of thousands of features, and so this detector represents a significant simplification along with increased performance.

1 Motivation

Understanding stories is a long-held goal of both artificial intelligence and natural language processing. Stories can be used for many interesting natural language processing tasks, and much can be learned from them, including concrete facts about specific events, people, and things; commonsense knowledge about the world; and cultural knowledge about the societies in which we live. Applying NLP directly to the large and growing number of stories available electronically, however, has been limited by our inability to efficiently separate story from non-story text. For the most part, studies of stories *per se* has relied on manual curation of story data sets (Mostafazadeh et al., 2016), which is, naturally, time-consuming, expensive, and doesn't scale. These human-driven methods pay no attention to the large number of stories generated daily in news, entertainment, and social media.

The goal of this work is to build and evaluate a high performing story detector that is both simple in design and generalizable across lexically different story corpora. Our definition of story can be found in §1.2, and is based on definitions used in prior work on story detection. Previous approaches to story detection have relied on tens of thousands of features (Ceran et al., 2012; Gordon and Swanson, 2009), and have used complicated pre-processing pipelines (Ceran et al., 2012). Moreover these prior systems, while clearly important advances, did not, arguably, include features that captured the "essence" of stories. Furthermore, these prior efforts had poor generalizability, i.e. when trained on one corpus, the detectors perform poorly when tested on a different corpus. Building on this prior work, we begin to address these shortcomings, presenting a new detector that has many orders of magnitude fewer features than used previously, significantly improved cross corpus performance, and higher F_1 on all training and testing combinations.

1.1 Task

Our goal is to design a system that can automatically decide whether or not a paragraph of text contains a story. We say a paragraph contains a story if any portion of it expresses a significant part of a story, including the characters and events involved in major plot points. Corpora used in prior work included Islamic Extremist texts (Ceran et al., 2012), and personal weblog posts (Gordon and Swanson, 2009), which were both annotated at

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2708–2715
Copenhagen, Denmark, September 7–11, 2017.

this level of granularity. In this paper we test combinations of new features on both of these corpora. Once we determined the best-performing feature set, we ran experiments using those features to evaluate its generalizability across corpora.

1.2 What is a Story?

Author E.M. Forster said "A story is a narrative of events arranged in their time sequence" (Forster, 2010). A more precise definition, of our own coinage, is that a narrative is a discourse presenting a coherent sequence of events which are causally related and purposely related, concern specific characters and times, and overall displays a level of organization beyond the commonsense coherence of the events themselves. In sum, a story is a series of events effected by animate actors. This reflects a general consensus among narratologists that there are at least two key elements to stories, namely, the plot (*fabula*) and the characters (*dramatis personae*) who move the plot forward (Abbott, 2008). While a story is more than just a plot carried out by characters–indeed, critical to 'storyness' is the connective tissue between these elements that can transport an audience to a different time and place–here we focus on these two core elements to effect better story detection.

1.3 Outline of the Paper

We begin by discussing prior work on story detection (§2). Then we introduce our new detector (§3), which relies on simple verb (§3.1) and character (§3.2) features. We test our detector on two corpora (§3.3)—one of blog posts and one of Islamist Extremist texts—using an SVM model to classify each paragraph as to whether or not it contains a story (§3.4). We conduct an array of experiments evaluating different combinations and variants of our features (§4). We also detail our use of undersampling for the majority class (§4.1), as well as our cross validation procedure (§4.2). We present both the results of the single corpus experiments (§4.3) and the cross-corpus and generalizability experiments (§4.4). We conclude with a list of contributions and discussion of future directions (§5).

2 Related Work

There have been three major contributions to the study of automatic story detection. In 2009, Gordan and Swanson developed a bag-of-words-based detector using blog data (Gordon and Swanson, 2009). They annotated a subset of paragraph-sized posts in the Spinn3r Blog corpus for the presence of stories, and used this data to train a confidence weighted linear classifier using all unigrams, bigrams, and trigrams from the data. Their best F_1 was 0.55. This was an important first step in story detection, and the annotated corpus of blog stories is an invaluable resource.

In 2012, Corman *et al.* developed a semantic-triplet-based detector using Islamist Extremist texts (Ceran et al., 2012). They annotated paragraphs of the CSC Islamic Extremist corpus for the presence of stories, and used this data to train an SVM with a variety of features including the top 20,000 *tf-idf* tokens, use of stative verbs, and agent-verb-patient triplets ("semantic triplets," discussed in more detail below in §3.1). Their best performing detector in that study achieved 0.63 F_1. The intent of the semantic triplet features was to encode the plot and the characters. These features were intended to capture the action of stories, but the specifics of the implementation was problematic: each unique agent-verb-patient triplet has its own element in the feature vector, and so this detector was sensitive primarily to the words that appeared in stories, not generalized actions or events.

Although Corman's detector has a higher F_1 than Gordon's, it was not clear which one was actually better; they were tested on different corpora. We compared the two detectors by reimplementing both, confirmed the correctness of the reimplementations, and running experiments where each detector was trained and tested on the corpora (Eisenberg et al., 2016). After these experiments, we showed that Corman's detector had better performance on the majority of experiments. Some of the results of these experiments are shown in Table 2. We also slightly improved the performance of Corman's detector to 0.66 F_1. In addition we reported results investigating the generalizability of the detectors; these results showed that neither the Gordon nor the Corman detectors generalized across corpora. We ascribed this problem to the fact that the features of each detector were closely tied to the literal words used, and did not attempt to generalize beyond those specific lexical items.

In terms of domain independence, we surveyed other discourse related tasks to see how generalization across domains has been achieved. For example, Braud *et al.* achieved domain independence in the identification of implicit relations between discourse units by training their system on both natural and synthetic data, weighting the influence of the two types (Braud and Denis, 2014). Jansen *et al.*, as another example, demonstrated domain independence on the task of non-factoid question answering by using both shallow and deep discourse structure, along with lexical features, to train their classifiers (Jansen et al., 2014). Thus, domain independence is certainly possible for discourse related tasks, but there does not yet seem to be a one-size-fits-all solution.

3 Developing the Detector

In contrast to focusing on specific lexical items, our implementation focuses on features which we believe capture more precisely the essence of stories, namely, features focusing on (a) events involving characters, and (b) the characters themselves.

3.1 Verb Features

Verbs are often used to express events. We use this fact to approximate event detection in a computationally efficient but still relatively accurate manner. The first part of each feature vector for a paragraph comprises 278 dimensions, where each element of this portion of the vector represents one of the 278 verb classes in VerbNet (Schuler, 2005). The value of each element depends on whether a verb from the associated verb class is used in the paragraph. Each element of the vector can have three values: the first value represents when a verb from the element's corresponding verb class is used in the paragraph and also involves a character as an argument of the verb. The second value represents when a verb from the verb class is used, but there are no characters involved. The third value represents the situation where no verbs from the verb class are used in the paragraph.

For clarity, we list the general steps of the verb feature extraction pipeline:

1. Split each paragraph into tokens, assign part of speech tags, and split the text into sentences, all using Stanford CoreNLP (Manning et al., 2014).
2. Parse each sentence with OpenNLP (Apache Foundation, 2017).
3. Label each predicate with its semantic roles using the SRL from the Story Workbench (Finlayson, 2008, 2011).
4. Disambiguate the Wordnet sense (Fellbaum, 1998) for each open-class word using the *It Makes Sense* WSD system (Zhong and Ng, 2010), using the Java WordNet Interface (JWI) to load and interact with WordNet (Finlayson, 2014).
5. Assign one of 278 VerbNet verb classes to each predicate, based on the assigned Wordnet sense, and using the jVerbnet library to interact with VerbNet. (Finlayson, 2012).
6. Determine whether the arguments of each predicate contains characters by using the Stanford Named Entity Recognizer (Finkel et al., 2005) and a gendered pronoun list.

We considered an argument to involve a character if it contained either (1) a gendered pronoun or (2) a named entity of type *Person* or *Organization*. We treated organizations as characters because they often fulfill that role in stories: for example, in the Extremist stories, organizations or groups like the *Islamic Emirate*, *Hamas*, or *the Jews* are agents or patients of important plot events. The verb features were encoded as a vector with length 278, each entry representing a different VerbNet verb class with three possible values: the verb class does not appear in the paragraph; the verb class appears but does not involve characters; or the verb class appears and a character is either an agent, patient, or both.

The verb features represent the types of events that occur in a paragraph, and whether or not characters are involved in those events. This is a generalized version of the semantic triplets that Corman *et al.* used for their story detector (Ceran et al., 2012), where they paired verbs with the specific tokens in the agent and patient arguments. The disadvantage of Corman's approach was that it led to phrases with similar meaning being mapped to different features: for example, the sentences "Bob played a solo" and "Mike improvised a melody" are mapped to different features by the semantic triplet based detector, even though the meaning of the sentences are almost the same: a character is performing music. On the other hand, in our approach, when we extract verb feature vectors from these sentences, both result in the same feature value, because the verbs *played* and *improvised* belong to the *performance* VerbNet class, and both verbs have a character in one of their arguments. This allow a generalized encoding of the types of action that occurs in a text.

3.2 Character Features

Our second focus is on character coreference chains. Characters, as discussed previously, are a key element of stories. A character must be present to drive the action of the story forward. We hypothesize that stories will contain longer coreference chains than non-stories. To encode this as a feature, we calculated the normalized length of the five longest coreference chains, and used those numbers as the character features. We computed these values as follows:

1. Extract coreference chains from each paragraph using Stanford CoreNLP coreference facility (Clark and Manning, 2016).
2. Filter out coreference chains that do not contain a character reference as defined in the Verb section above (a named entity of type *Person* or *Organization*, or a gendered pronoun).
3. Sort the chains within each paragraph with respect to the number of references in the chain.
4. Normalize the chain lengths by dividing the number of referring expression in each chain by the number of sentences in the paragraph.

These normalized chain lengths were used to construct a five-element feature vector for use by the SVM. We experimented with different numbers of longest chains, anywhere from the single longest to the ten longest chains. Testing on a development set of 200 Extremist paragraphs revealed using the five longest chains produced the best result.

3.3 Corpora

As noted, we used two corpora that were annotated by other researchers for the presence of stories at the paragraph level. The CSC Islamic Extremist Corpus comprises 24,009 paragraphs (Ceran et al., 2012), of which 3,300 were labeled as containing a story. These texts recount Afghani and Jihadi activities in the mid-2000's in a variety of location around the world. This corpus was originally used to train and test Corman's semantic-triplet-based story detector. The web blog texts come from the ICWSM 2009 Spinn3r Dataset (Burton et al., 2009). The full data set contains 44 million texts in many languages. Gordon and Swanson (2009) annotated a sample of 4,143 English texts from the full data set, 201 of which were identified as containing stories. This corpus was originally used to train and test Gordon's bag-of-words-based detector. Most of the texts in the blog corpus are no more than 250 characters, roughly a paragraph. The distribution of texts can be seen in Table 1.

Corpus	Story	Non-Story
Extremist	3,300	20,709
Blog	201	3,942

Table 1: Distribution of story paragraphs across the Extremist and blog corpora.

3.4 SVM Machine Learning

We used the Java implementation of LibSVM (Chang and Lin, 2011) to train an SVM classifier with our features. The hyper-parameters for the linear kernel were $\gamma = 0.5$, $\nu = 0.5$, and $c = 20$.

4 Experiments & Results

The results of our new experiments are shown in Table 3. We report precision, recall, and F_1 relative to the story and non-story classes. We performed experiments on three feature sets: the verb features alone (indicated by **Verb** in the table), character features alone (indicated by **Char**), and all features together (**Verb+Char**). We conducted experiments ranging over three corpora: the Extremist corpus (**Ext**), the blog corpus (**Web**), and the union of the two (**Comb**). These results may be compared with the previously best performing detector, namely, Corman's semantic triplet based detector (Ceran et al., 2012), as tested by us in prior work (Eisenberg et al., 2016), and shown in Table 2.

Training	Testing	Prec.	Recall	F_1
Ext	**Ext**	0.77	0.57	**0.66**
Ext	**Web**	0.23	0.37	0.28
Ext	**Comb**	0.43	0.41	0.32
Web	**Web**	0.66	0.31	0.43
Web	**Ext**	0.59	0.003	0.01
Web	**Comb**	0.59	0.01	0.01
Comb	**Ext**	0.62	0.51	0.43
Comb	**Web**	0.36	0.49	0.30
Comb	**Comb**	0.64	0.47	0.46

Table 2: Results for the Corman semantic triplet based detector as reported in (Eisenberg et al., 2016). These results are with respect to the story class.

				Not Story			Story	
Features	**Training**	**Testing**	Prec.	Recall	F_1	Prec.	Recall	F_1
Verb	**Ext**	**Ext**	0.73	0.81	0.77	0.78	0.70	0.74
Verb	**Web**	**Web**	0.69	0.75	0.72	0.73	0.66	0.69
Char	**Ext**	**Ext**	0.30	0.27	0.21	0.52	0.74	0.55
Char	**Web**	**Web**	0.67	0.68	0.67	0.67	0.65	0.65
Verb+Char	**Ext**	**Ext**	0.73	0.81	0.77	0.79	0.70	**0.74**
Verb+Char	**Ext**	**Web**	0.68	0.80	0.73	0.75	0.63	0.69
Verb+Char	**Ext**	**Comb**	0.70	0.77	0.73	0.75	0.67	0.71
Verb+Char	**Web**	**Web**	0.71	0.76	0.72	0.74	0.68	0.70
Verb+Char	**Web**	**Ext**	0.50	0.82	0.62	0.50	0.18	0.27
Verb+Char	**Web**	**Comb**	0.53	0.79	0.64	0.60	0.40	0.41
Verb+Char	**Comb**	**Ext**	0.74	0.81	0.77	0.79	0.71	**0.75**
Verb+Char	**Comb**	**Web**	0.68	0.74	0.70	0.72	0.64	0.67
Verb+Char	**Comb**	**Comb**	0.72	0.81	0.76	0.79	0.68	0.73

Table 3: Results of the new detectors as trained and tested on the Extremist (Ext), weblog (Web), or combined (Comb) corpora. The feature sets tested include the 278 verb class features (Verb), the normalized length of the five longest coreference chains (Char), and the combination of these two feature sets (Verb+Char). Undersampling is utilized in each of these experiments.

4.1 Undersampling

In each of the new experiments, we undersampled the non-story class before training (Japkowicz, 2000). Undersampling is a technique used to help supervised machine learning classifiers learn more about a class that has a significantly smaller number of examples relative to an alternative. In our case, non-story labels outnumbered story labels by a factor of 7 overall. Extremist story paragraphs are only 15.9% of the total annotated paragraphs in that set, and in the blog corpus stories were only 4.9% of the paragraphs. To prevent the detector from being over trained on non-story paragraphs, we thus reduced the size of the non-story training data to that of the story data, by randomly selecting a number of non-story texts equal to the number of story texts for training and testing.

4.2 Cross Validation

We used three versions of cross validation for the new experiments, one for each experimental condition: training and testing on a single corpus; training on a single corpus and testing on the combined corpus; or training on the combined corpus and testing on a single corpus. These procedures are the same as in our previous work (Eisenberg et al., 2016). We performed undersampling before cross validation, so when we are explaining how to divide up the story and non-story texts into cross validation folds, this refers to the full set of story texts and the set of non-story texts that was randomly selected to equal the number of story texts. For all experiments with cross validation, we use ten folds.

Train and Test on a Single Corpus: If the training and testing corpus is the same, divide up the stories into ten subsets of equal size, and the undersampled non-stories into ten subsets of equal size. For each fold of cross validation a different story set and non-story set (of the same index) are used as the testing set and the remaining nine are used for training.

Train on Combined, Test on Single: If the training is done on the combined corpus, and the test corpus is either the weblog or Extremist corpus, which we will refer to as the single test corpus, first divide the stories from the single test corpus into ten equal sized sets, and then divide up that corpus's non-stories into ten equal sets. For each fold of cross validation a different story set and non-story set (of the same index) from the single test corpus are used as the testing set and the remaining nine are used for training. The texts from the other corpus (the corpus that is not the single test corpus), are undersampled and added to all ten folds of training.

Train on Single, Test on Combined: If training is done on a single corpus, and the test corpus is the combined corpus, first divide the stories from the single training corpus into ten equal sized sets, and the undersampled non-stories from the single training corpus into ten equal sized sets. For each fold of cross validation a different story set and non-story set (of the same index) from the single training corpus are used as the testing set and the remaining nine are used for training. The texts from the other corpus (the corpus that is not the single training corpus), are undersampled and added to all ten folds of testing.

4.3 Single Corpus Experiments

For every experiment that used only a single corpus, the best feature set included both the verb and character features, achieving up to 0.74 F_1 when trained and tested on the Extremist corpus. This represents a new state-of-the-art, about 12.6% greater than the performance of Corman's detector when trained and tested on the same corpus (0.66 F_1).

When the detector uses only verb features it achieves an F_1 of 0.74 on the Extremist corpus, only 0.002 lower than the detector using all the features. Interestingly, the detector achieves 0.55 F_1 using only the five character features, which is respectful given such a small feature set. To put this in perspective, the Corman detector (Ceran et al., 2012) uses more than 20,000 features, and achieves an F_1 of 0.66. Thus we were able to achieve 83% of the performance of the Corman detector with 4,000 times fewer features.

When training and testing on the blog corpus, the detector using all the features achieved 0.70 F_1, a 74% increase from the Corman detector's 0.425 F_1. This is the best performing model on the blog corpus, from any experiment to date. The detector using only verb features achieves 0.74 F_1, which is only slightly worse than when both sets of features are used. When we trained using only the character features, the system achieves 0.65 F_1, which is still 54% higher than when the Corman detector is trained and tested on the blog corpus.

In the single corpus experiments, the detectors that we trained and tested on the Extremist paragraphs have higher performance than those trained on the web blogs, except for when we use only the five character features. A possible reason for this is the Stanford NER may not be recognizing the correct named entities in the Extremist texts, which contain many non-Western names, e.g., *Mujahidin*, *Okba ibn Nafi*, or *Wahid*. However, when we include the verb features, the detectors trained on the Extremist texts achieve better performance. We believe this is partially due to the greater number of stories in the Extremist corpus, and their increased grammatical fluency. The Extremist corpus is actually well written compared to the blog corpus, the latter of which contains numerous fragmentary and disjointed posts.

4.4 Cross Corpus Experiments

We show the generalizability of our best-performing detector (that including both verb and character features) by training it on one corpus and testing it on another.

When we trained the detector on the Extremist texts and tested on the blog texts, it scores a 0.68 F_1. This is 142% improvement over Corman's detector in the same setup (0.28 F_1), and is a higher F_1 than the previous state-of-the-art on any single corpus test. When we trained the detector on the Extremist corpus and tested on the combined corpus, it achieved 0.71 F_1, which is an 121% increase from Corman's detector in the equivalent setup.

For the detector trained on the blog corpus and tested on the Extremist corpus, the detector that uses both verbs and character features achieves an 0.27 F_1, which is a 2,600% increase over the Corman detector's 0.01 F_1 in this same setup. While 0.27 F_1 can by no means be called good performance, it is significantly better than the Corman detector's performance on this task, and so demonstrates significantly better generalizability. As seen in our experiments, detectors trained on only the blog corpus do not perform as well as detectors trained on the Extremist corpus. We suspect that this is partially due to the disfluent nature of the blog corpus, which includes many fragmentary sentences, grammatical errors, and slang, all of which are difficult for the NLP pipeline to handle.

Note that we performed no cross validation in the above experiments where we trained the detector on the Extremist corpus and tested on the blog corpus, or vice versa, because in these cases the training and testing sets have no intersection.

The cross corpus experiment with the largest percent increase is for the verb and character detector trained on the blog corpus and tested on the combined corpus. The new detector's F_1 is 0.41, a 4,000% increase from the Corman detector's 0.01 F_1 on this task. Although a 0.41 F_1 is also not good, this is a massive improvement over previous performance. This is further evidence that our verb and character feature based detector is significantly more generalizable than Corman's approach.

The remaining five cross corpus experiments involved the combined corpus. In this case, our detector out-performed Corman's detector. Of special note is the detector trained on the combined corpus and tested on the Extremist corpus. It achieved 0.75 F_1, which is 0.01 points of F_1 higher than our best single corpus detector, which was trained and tested on the Extremist corpus. This isn't a substantial increase in performance, but it suggests that information gleaned from the blog corpus does potentially–albeit marginally–help classification of the Extremist texts.

5 Conclusion

We have introduced a new story detection approach which uses simple verb and character features. This new detector outperforms the prior state-of-the-art in all tasks, sometimes by orders of magnitude. Further, we showed that our detector generalizes significantly better across lexically different corpora. We propose that this increase in performance and generalizability is due to the more general nature of our features, especially those related to verb classes. This approach has additional advantages, for example, the feature vector is fixed in size and does not grow in an unbounded fashion as new texts (with new verbs, agents, and patents) are added to the training data.

In future work we plan to develop richer character-based features. The current approach uses only normalized lengths of the five longest coreference chains, which leaves out important information about characters that could be useful to story detection. Indeed, our experiments showed that these character features only add a small amount of information above and beyond the verb features. However, when used alone, the character features still yield reasonable performance, which suggests that more meaningful character-based features could lead to story detectors with even better performance.

Acknowledgments This work was partially supported by National Institutes of Health (NIH) grant number 5R01GM105033-02. We thank W. Victor H. Yarlott for building a wrapper class to the *It Makes Sense* WSD. We also thank Reid Swanson especially, as well as Andrew Gordon, for working to retrieve, format, and providing their original annotations of the Weblog corpus. Thanks to Steve Corman for facilitating the transmission of the Extremist story data, which is covered by U.S. government FOUO rules.

References

H. Porter Abbott. 2008. *The Cambridge Introduction to Narrative*. Cambridge University Press, Cambridge, England.

Apache Foundation. 2017. OpenNLP v1.6.0, https://opennlp.apache.org, Last accessed on July 20, 2017.

Chloé Braud and Pascal Denis. 2014. Combining Natural and Artificial Examples to Improve Implicit Discourse Relation Identification. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 1694–1705, Dublin, Ireland.

Kevin Burton, Akshay Java, and Ian Soboroff. 2009. The ICWSM 2009 Spinn3r Dataset. In *Proceedings of the 3rd Annual Conference on Weblogs and Social Media (ICWSM 2009)*, San Jose, CA.

Betul Ceran, Ravi Karad, Steven Corman, and Hasan Davulcu. 2012. A Hybrid Model and Memory Based Story Classifier. In *Proceedings of the 3rd International Workshop on Computational Models of Narrative (CMN'12)*, pages 60–64, Istanbul, Turkey.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–27.

Kevin Clark and Christopher D. Manning. 2016. Deep Reinforcement Learning for Mention-Ranking Coreference Models. In *Proceedings of the 21st Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pages 2256–2262, Austin, Texas.

Joshua D Eisenberg, W Victor H Yarlott, and Mark A Finlayson. 2016. Comparing extant story classifiers: Results & new directions. In *Proceedings of the 7th International Workshop on Computational Models of Narrative (CMN'16)*, Paper 6, Krakow, Poland.

Christine Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL 2005)*, pages 363–370, Ann Arbor, MI.

Mark A. Finlayson. 2008. Collecting Semantics in the Wild: The Story Workbench. In *Proceedings of the AAAI Fall Symposium on Naturally Inspired Artificial Intelligence*, pages 46–53, Arlington, VA.

Mark A. Finlayson. 2011. The Story Workbench: An Extensible Semi-Automatic Text Annotation Tool. In *Proceedings of the 4th Workshop on Intelligent Narrative Technologies (INT4)*, pages 21–24, Stanford, CA.

Mark A. Finlayson. 2012. JVerbnet, v1.2.0, http://projects.csail.mit.edu/jverbnet, Last accessed on July 17, 2017.

Mark A. Finlayson. 2014. Java Libraries for Accessing the Princeton Wordnet: Comparison and Evaluation. In *Proceedings of the 7th International Global Wordnet Conference (GWC 2014)*, pages 78–85, Tartu, Estonia.

Edward M. Forster. 2010. *Aspects of the Novel*. RosettaBooks, New York City, New York.

Andrew Gordon and Reid Swanson. 2009. Identifying Personal Stories in Millions of Weblog Entries. In *Proceedings of the 3rd International Conference on Weblogs and Social Media (ICWSM 2009), Data Challenge Workshop*, pages 16–23, San Jose, CA.

Peter Jansen, Mihai Surdeanu, and Peter Clark. 2014. Discourse Complements Lexical Semantics for Non-factoid Answer Reranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), Long Papers*, pages 977–986, Baltimore, MD.

Nathalie Japkowicz. 2000. Learning from Imbalanced Data Sets: a Comparison of Various Strategies. In *AAAI Workshop on Learning from Imbalanced Data Sets*, pages 10–15, Austin, TX.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), System Demonstrations*, pages 55–60, Baltimore, MD.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2016)*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Karin Kipper Schuler. 2005. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania.

Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense: A Wide-Coverage Word Sense Disambiguation System for Free Text. In *Proceedings of the 48th Annual Meeting of the Assocation for Computational Linguistics (ACL 2010), System Demonstrations*, pages 78–83, Uppsala, Sweden.

Multi-modular domain-tailored OCR post-correction

Sarah Schulz and **Jonas Kuhn**
Institute for Natural Language Processing (IMS)
University of Stuttgart
firstname.lastname@ims.uni-stuttgart.de

Abstract

One of the main obstacles for many Digital Humanities projects is the low data availability. Texts have to be digitized in an expensive and time consuming process whereas Optical Character Recognition (OCR) post-correction is one of the time-critical factors. At the example of OCR post-correction, we show the adaptation of a generic system to solve a specific problem with little data. The system accounts for a diversity of errors encountered in OCRed texts coming from different time periods in the domain of literature. We show that the combination of different approaches, such as e.g. Statistical Machine Translation and spell checking, with the help of a ranking mechanism tremendously improves over single-handed approaches. Since we consider the accessibility of the resulting tool as a crucial part of Digital Humanities collaborations, we describe the workflow we suggest for efficient text recognition and subsequent automatic and manual post-correction.

1 Introduction

Humanities are no longer just the realm of scholars turning pages of thick books. As the worlds of humanists and computer scientists begin to intertwine, new methods to revisit known ground emerge and options to widen the scope of research questions are available. Moreover, the nature of language encountered in such research attracts the attention of the NLP community (Kao and Jurafsky (2015), Milli and Bamman (2016)). Yet, the basic requirement for the successful implementation of such projects often poses a stumbling block: large digital corpora comprising the textual material of interest are rare. Archives and individual scholars are in the process of improving this situation by applying Optical Character Recognition (OCR) to the physical resources. In the *Google Books*[1] project books are being digitized on a large scale. But even though collections of literary texts like *Project Gutenberg*[2] exist, these collections often lack the texts of interest to a specific question. As an example, we describe the compilation of a corpus of adaptations of Goethe's *Sorrows of the young Werther* which allows for the analysis of character networks throughout the publishing history of this work.

The success of OCR is highly dependent on the quality of the printed source text. Recognition errors, in turn, impact results of computer-aided research (Strange et al., 2014). Especially for older books set in hard-to-read fonts and with stained paper the output of OCR systems is not good enough to serve as a basis for Digital Humanities (DH) research. It needs to be post-corrected in a time-consuming and cost-intensive process.

We describe how we support and facilitate the manual post-correction process with the help of informed automatic post-correction. To account for the problem of relative data sparsity, we illustrate how a generic architecture agnostic to a specific domain can be adjusted to text specificities such as genre and font characteristics by including just small amounts of domain specific data. We suggest a system architecture (cf. Figure 1) with trainable modules which joins general and specific problem solving as required in many applications.

We show that the combination of modules via a ranking algorithm yields results far above the performance of single approaches.

[1] https://books.google.de/, 02.04.2017.
[2] http://www.gutenberg.org, 14.04.2017.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2716–2726
Copenhagen, Denmark, September 7–11, 2017.

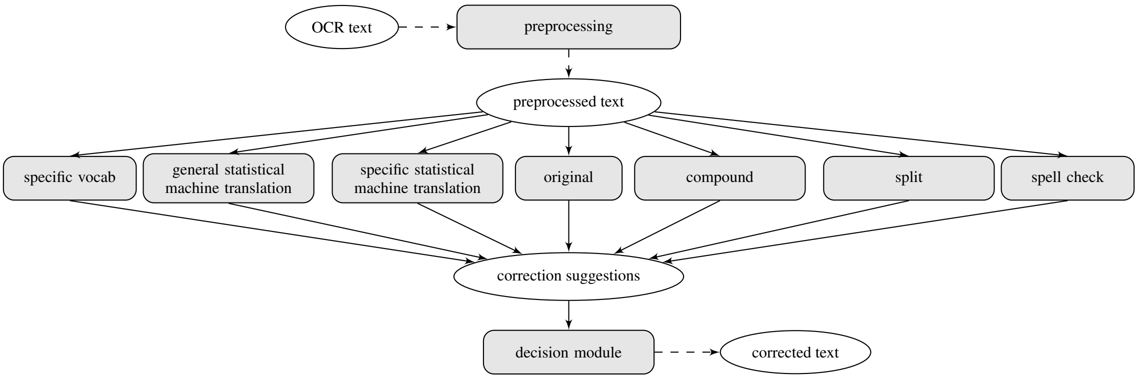


Figure 1: Multi-modular OCR post-correction system.

We discuss the point of departure for our research in Section 2 and introduce the data we base our system on in Section 4. In Section 5, we illustrate the most common errors and motivate our multi-modular, partly customized architecture. Section 6 gives an overview of techniques included in our system and the ranking algorithm. In Section 7, we discuss results, the limitations of automatic post-correction and the influence the amount of training data takes on the performance of such a system. Finally, Section 8 describes a way to efficiently integrate the results of our research into a digitization work-flow as we see the easy accessibility of computer aid as a central point in Digital Humanities collaborations.

2 Related work

There are two obvious ways to automatically improve quality of digitized text: optimization of OCR systems or automatic post-correction. Commonly, OCR utilizes just basic linguistic knowledge like character set of a language or reading direction. The focus lies on the image recognition aspect which is often done with artificial neural networks (cf. Graves et al. (2009), Desai (2010)).
Post-correction is focused on the correction of errors in the linguistic context. It thus allows for the purposeful inclusion of knowledge about the text at hand, e.g. genre-specific vocabulary. Nevertheless, post-correction has predominantly been tackled OCR system agnostic as outlined below. As an advantage, post-correction can also be applied when no scan or physical resource is available.
There have been attempts towards shared datasets for evaluation. Mihov et al. (2005) released a corpus covering four different kinds of OCRed text comprising German and Bulgarian. However, in 2017 the corpus was untraceable and no recent research relating to the data could be found.
OCR post-correction is applied in a diversity of fields in order to compile high-quality datasets. This is not merely reflected in the homogeneity of techniques but in the metric of evaluation as well. While accuracy has been widely used as evaluation measure in OCR post-correction research, Reynaert (2008a) advocates the use of precision and recall in order to improve transparency in evaluations. Dependent on the paradigm of the applied technique even evaluation measures like BLEU score can be found (cf. Afli et al. (2016)).
Since shared tasks are a good opportunity to establish certain standards and facilitate the comparability of techniques, the Competition on Post-OCR Text Correction[3] organized in the context of ICDAR 2017 could mark a milestone for more unified OCR post-correction research efforts.

Regarding techniques used for OCR post-correction, there are two main trends to be mentioned: statistical approaches utilizing error distributions inferred from training data and lexical approaches oriented towards the comparison of source words to a canonical form. Combinations of the two approaches are also available.
Techniques residing in this **statistical** domain have the advantage that they can model specific distributions of the target domain if training data is available. Tong and Evans (1996) approach post-correction as a statistical language modeling problem, taking context into account. Pérez-Cortes et al. (2000) employ stochastic finite-state automaton along with a modified version of the Viterbi Algorithm to perform a stochastic error correcting parsing. Extending the simpler stochastic context-sensitive models, Kolak and

[3] https://sites.google.com/view/icdar2017-postcorrectionocr/home, 3.07.2017.

Resnik (2002) apply the first noisy channel model, using edit distance from noisy to corrected text on character level. In order to train such a model, manually generated training data is required. Reynaert (2008b) suggests a corpus-based correction method, taking spelling variation (especially in historical text) into account. Abdulkader and Casey (2009) introduce an error estimator neural network that learns to assess error probabilities from ground truth data which in turn is then suggested for manual correction. This decreases the time needed for manual post-correction since correct words do not have to be considered as candidates for correction by the human corrector. Llobet et al. (2010) combine information from the OCR system output, the error distribution and the language as weighted finite-state transducers. Reffle and Ringlstetter (2013) use global as well as local error information to be able to fine-tune post-correction systems to historical documents. Related to the approach introduced by Pérez-Cortes et al. (2000), Afli et al. (2016) use statistical machine translation for error correction using the Moses toolkit on character level. Volk et al. (2010) merge the output of two OCR systems with the help of a language model to increase the quality of OCR text. The corpus of yearbooks of the Swiss Alpine Club which has been manually corrected via crowdsourcing (cf. Clematide et al. (2016)) is available from their website.

Lexical approaches often use rather generic distance measures between an erroneous word and a potential canonical lexical item. Strohmaier et al. (2003) investigate the influence of the coverage of a lexicon on the post-correction task. Considering the fact that writing in historical documents is often not standardized, the success of such approaches is limited. Moreover, systems based on lexicons rely on the availability of such resources. Historical stages of a language – which constitute the majority of texts in need for OCR post-correction – often lack such resources or provide incomplete lexicons which would drastically decrease performance of spell-checking-based systems. Ringlstetter et al. (2007) address this problem by suggesting a way to dynamically collect specialized lexicons for this task. Takahashi et al. (1990) apply spelling correction with preceding candidate word detection. Bassil and Alwani (2012) use Google's online spelling suggestions for as they draw on a huge lexicon based on contents gathered from all over the web. The **human component** as final authority has been mentioned in some of these projects. Visual support of the post-correction process has been emphasized by e.g. Vobl et al. (2014) who describe a system of iterative post-correction of OCRed historical text which is evaluated in an application-oriented way. They present the human corrector with an alignment of image and OCRed text and make batch correction of the same error in the entire document possible. They can show that the time needed by human correctors considerably decreases.

3 Evaluation metrics

We describe and evaluate our data by means of word error rate (WER) and character error rate (CER). The error rates are a commonly used metric in speech recognition and machine translation evaluation and can also be referred to as length normalized edit distance. They quantify the number of operations, namely the number of insertions, deletions and substitutions, that are needed to transform the suggested string into the manually corrected string and are computed as follows:

$$\text{WER} = \frac{\text{word insertions} + \text{word substitutions} + \text{word deletions}}{\text{\# words in the reference}}$$

$$\text{CER} = \frac{\text{char insertions} + \text{char substitutions} + \text{char deletions}}{\text{\# characters in the reference}}$$

4 Data

As mentioned in the introduction, errors found in OCRed texts are specific to time of origin, quality of scan and even the characteristics of a specific text. Our multi-modular architecture paves the way for a solution taking this into account by including general as well as specific modules. Thus, we suggest to include domain specific data as well as larger, more generic data sets in order to enhance coverage of vocabulary and possible error classes. The data described hereafter constitutes parallel corpora with OCR output and manually corrected text which we utilize for training statistical models.

4.1 The Werther corpus

Since our system is developed to help in the process of compiling a corpus comprising adaptations of Goethe's *The Sorrows Of Young Werther* throughout different text types and centuries, we

1	Berichtigung der Geschichte des jungen Werthers	H. von Breitenbach	1775
2	Schwacher jedoch wohlgemeynter Tritt vor dem Riss, neben oder hinter Herren Pastor Goeze, gegen die Leiden des jungen Werthers und dessen ruchlose Anhänger	anonymous	1775
3	Lorenz Konau	David Iversen	1776
4	Werther der Jude	Ludwig Jacobowski	1910
5	Eine rührende Erzählung aus geheimen Nachrichten von Venedig und Cadir (first letter)	Joseph Codardo und Rosaura Bianki	1778
6	Afterwerther oder Folgen jugendlicher Eifersucht	A. Henselt	1784
7	Der neue Werther oder Gefühl und Liebe	Karl P. Bonafont	1804
8	Leiden des modernen Werther	Max Kaufmann	1901

Table 1: Werther texts included in our corpus from different authors and times of origin.

collected texts from this target domain. To be able to train a specialized system, we manually corrected a small corpus of relevant texts (cf. Table 2). We use the output of Abbyy Fine Reader 7 for several Werther adaptations (Table 1) all based on scans of books with German Gothic lettering.

4.2 The Deutsches Textarchive (DTA) corpus

Even though manual OCR post-correction is a vital part of many projects, only very little detailed documentation of this process exists. *Das Deutsche Textarchiv* (The German Text Archive) (DTA) is one of the few projects providing detailed correction guidelines along with the scans and the text corrected within the project (Geyken et al., 2012). This allows the compilation of a comprehensive parallel corpus of OCR output and corrected text spanning a period of four centuries (17th to 20th) in German Gothic lettering. For OCR, we use the open source software *tesseract*[4] (Smith and Inc, 2007) which comes with recognition models for Gothic font.

4.3 Gutenberg data for language modeling

Since the output of our system is supposed to consist of well-formed German sentences, we need a method to assess the quality of the output language. This task is generally tackled by language modeling. We compiled a collection of 500 randomly chosen texts from Project Gutenberg[5] comprising 28,528,078 tokens. With its relative closeness to our target domain it constitutes the best approximation of a target language. The language model is trained with the KenLM toolkit (Heafield, 2011) with an order of 5 on token level and 10 on character level following De Clercq et al. (2013).

5 Why OCR post-correction is hard

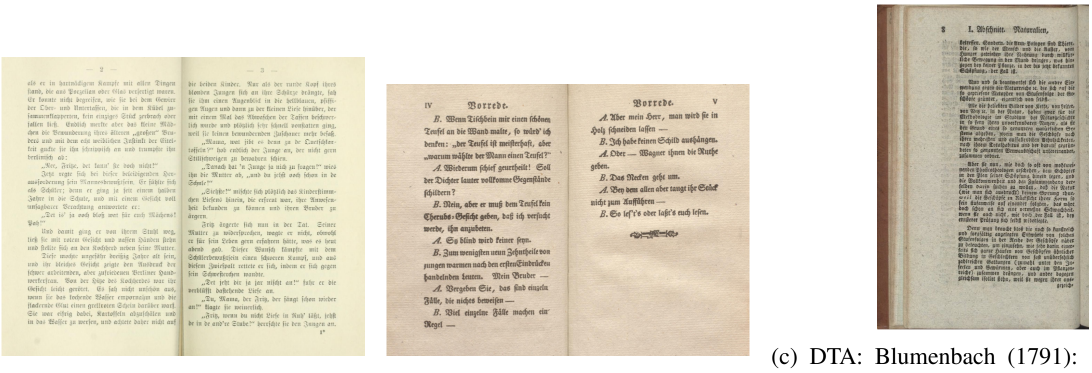
In tasks like the normalization of historical text (Bollmann et al., 2012) or social media, one can take advantage of regularities in the deviations from the standard form that appear throughout an entire genre or in case of social media e.g. dialect region (Eisenstein, 2013). Errors in OCR, however, depend on the font and quality of the scan as well as the time of origin which makes each text unique in its composition of features and errors.

In order to exemplify this claim, we analyzed three different samples: *Lorenz Konau* (1776), *Werther der Jude* (1910) and a sample from the DTA data. Figure 2 (a-c) illustrate the point that the quality of scan is crucial for the OCR success. Figure 2a shows a text from the 20th century where the type setting is rather regular and the distances between letters is uniform as opposed to Figure 2b. Figure 2c shows how the writing from the back of the page shines through and makes the script less readable. Thus, we observe a divergence in the frequency of certain character operations between those texts: the percentage of substitutions range between 74% for *Lorenz Konau* and 60% for *Werther der Jude* and 18% and 30% of insertions, respectively. The varying percentage of insertions might be due to the fact that some scans are more "washed out" than others. Successful insertion of missing characters, however, relies on the precondition that a system knows a lot of actual words and sentences in the respective language and cannot be resolved via e.g. character similarity like in the substitution from *l* to *t*.

Another factor that complicates the correction of a specific text is the number of errors per word. Words with an edit distance of one to the correct version are easier to correct those with more than one necessary operation. With respect to errors per word our corpus shows significant differences in error distributions. Especially in our DTA corpus the number of words with two or more character-level errors per word is considerably higher than those with one error. For *Werther*

[4] Considering the open source aspect of our resulting system, we decided to use the open source OCR software tesseract and move away from Abbyy some time after our project started: https://github.com/tesseract-ocr.

[5] Project Gutenberg. Retrieved January 21, 2017, from www.gutenberg.org.



(a) Werther der Jude (1910)

(b) Lorenz Konau (1776)

(c) DTA: Blumenbach (1791): Handbuch der Naturgeschichte

Figure 2: Scans of three different texts from our corpora. Emphasizes differences in quality of scan and differences in type setting, font and genre (e.g. drama).

der Jude (WER 10.0, CER 2.4) the number of errors in general is much lower than for *Konau* (WER: 34.7, CER: 10.9). These characteristics indicate that subcorpus-specific training of a system is promising.

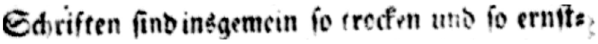
6 Specialized multi-modular post-correction

In order to account for the nature of errors that can occur in OCR text, we apply a variety of modules for post-correction. The system proceeds in two stages and is largely based on an architecture suggested by Schulz et al. (2016) for normalization of user-generated contents. In the first stage, a set of specialized modules (Section 6.1) suggest corrected versions for the tokenized[6] OCR text lines. Those modules can be context-independent (work on just one word at a time) or context-dependent (an entire text line is processed at a time). The second stage is the decision phase. After the collection of various suggestions per input token, these have to be ranked to enable a decision for the most probable output token given the context. We achieve this by assigning weights the different modules with the help of Minimal Error Rate Training (MERT) (Och, 2003).

6.1 Suggestion modules

In the following, we give an outline of techniques included into our system.

6.1.1 Word level suggestion modules



Schriften ſind insgemein ſo trocken und ſo ernſt;

Figure 3: Irregular type setting in German Gothic lettering. *sind* and *insgemein* are two separate words but yet written closely together.

- **Original**: the majority of words do not contain any kind of error, thus we want to have the initial token available in our suggestion pool
- **Spell checker**: spelling correction suggestion for misspelled words with hunspell[7]
- **Compounder**: merges two tokens into one token if it is evaluated as an existing word by hunspell
- **Word splitter**: splits two tokens into two words using compound-splitter module from the Moses toolkit (Koehn et al., 2007)
- **Text-Internal Vocabulary**: extracts high-frequent words from the input texts and suggests them as correction of words with small adjusted Levenshtein distance[8]

The compound and word split techniques react to the variance in manual typesetting, where the distances between letters vary. This means that the word boundary recognition becomes difficult (cf. Figure 3).

A problem related to the spell-checking approach is the limited coverage of the dictionary since it uses a modern German lexicon. Related to this is the difficulty of out-of-vocabulary words above average for literature text. Archaic words from e.g. the 17th century or named entities cannot be found in a dictionary and can therefore not be covered with any of the approaches mentioned above.

[6] Tokenizer of TreeTagger (Schmid, 1997).

[7] https://github.com/hunspell/hunspell.

[8] OCR-adjusted Levenshtein distance taking frequent substitution, insertion and deletion patterns learned from training data into account.

However, especially named entities are crucial for the automatic or semi-automatic analysis of narratives e.g. with the help of network analysis. Our Text-Internal Vocabulary technique is designed to find frequent words in the input text, following the assumption that errors would not be regular enough to distort those frequencies. We compile a list from those high-frequency words. Subsequently, erroneous words can be corrected calculating an OCR-adjusted Levenshtein distance. In this way misspelled words like *Loveuzo* could be resolved to *Lorenzo* if this name appears frequently. Since the ranking algorithm relies on a language model which will most probable not contain those suggestions, we insert the high-frequency words into the language modeling step.

6.1.2 Sentence level suggestion modules

As has been suggested by Afli et al. (2016), we include Phrase-based **Statistical Machine Translation** (SMT) into our system. We treat the post-correction as a translation problem translating from erroneous to correct text. Like in standard SMT, we train our models on a parallel corpus, the source language being the OCRed text and the target language being manually corrected text. We train models on token level as well as on character-level (unigram). This way, we aim at correcting frequently mis-recognized words along with frequent character-level errors. We train four different systems:

- token level
  - domain specific data (cf. Section 4.1)
  - general data (cf. Section 4.2)
- character level
  - domain specific data (cf. Section 4.1)
  - general data (cf. Section 4.2)

The models are trained with the Moses toolkit (Koehn et al., 2007). Moreover, we use a subsequent approach by forwarding the output of the character-based SMT model to the token-based SMT.

6.1.3 Additional feature

The information whether a word contains an error can help to avoid the incorrect alternation of an initially correct word (overcorrection). In order to deliver this information to the decision module without making a hard choice for each word, we include the information whether a word has been found either in combination with the word before or after in a corpus (cf. Section 4.3) into the decision process in form of a feature that will be weighted along with the other modules. This naive language modeling approach allows for a context-relevant decision of the correctness of a word.

6.2 Decision modules: the ranking mechanism

Since the recognition errors appearing in a text are hard to pre-classify by nature, we run all modules on each sentence of the input, returning suggestions for each word. Since the output of some of our modules are entire sentences, input sentence and output sentence have to be word-aligned in order to be able to make suggestions on word level. The word alignment between input and output sentence is done with the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970), an algorithm originally developed in bioinformatics.

From all corrected suggestions the most probable well-formed combination has to be chosen. To solve the combinatorial problem of deciding which suggestion is the most probable candidate for a word, the decision module makes use of the Moses decoder.

As in general SMT, the decoder makes use of a language model (cf. Section 4.3) and a phrase table. The phrase table is compiled from all input words along with all possible correction suggestions. In order to assign weights to the single modules and the language model, we tune on the phrase tables collected from a run on our dev$_{overall}$ set, following the assumption that suggestions of certain modules are more reliable than others and expect their feature weights to be higher after tuning.

7 Experiments

7.1 Experimental Setup

To guarantee diversity, we split each of texts 1-4 (cf. Table 1) into three parts and combined the respective parts: 80% train (train), 10% development (dev$_{SMT}$) and 10% test (test$_{init}$).

Test setup We introduce two different test scenarios. Even though both test sets are naturally compiled from unseen data, the first test set consists of a self-contained Werther adaptation introducing new named entities, originating from a different source and thus showing a different error

set	# tokens (OCR)	# tokens (corr)	WER	CER
train	70,159	68,608	15.7	5.5
train$_{ext}$	133,457	131,901	12.9	4.0
dev$_{SMT}$	12,464	12,304	13.9	3.5
dev$_{overall}$	13,663	13,396	16.75	4.6
test$_{init}$	17,443	17,367	9.4	2.5
test$_{unk}$	13,286	13,304	31.2	9.2

Table 2: Werther specific parallel corpus of OCR text and corrected text showing the number of tokens before and after post-correction along with WER and CER

set	# tokens (OCR)	# tokens (corr)	WER	CER
train	3,452,922	3,718,712	41.6	13.2
dev	663,376	836,974	30.4	9.1

Table 3: DTA parallel corpus of OCR text and corrected text showing the number of tokens before and after post-correction along with WER and CER

constitution. It constitutes an evaluation in which no initial manual correction as support for the automatic correction is included in the workflow. We henceforth call this unknown set *test$_{unk}$* (text 6).

In contrast, the second set contains parts of the same texts as the training, thus specific vocabulary might have been introduced already. The results for this test set give a first indication of the extent to which pre-informing the system with manually correcting parts of a text could assist the automatic correction process. Since this scenario can be described as a text-specific initiated post-correction, we henceforth refer to this test set as *test$_{init}$*.

We further on experiment with an extended training set train$_{ext}$ (train with texts 7 and 8) to assess the influence of the size of the specific training set on the overall performance. The sizes of the datasets before and after correction along with WER and CER are summarized in Table 2. The sizes for the general dataset before and after correction along with WER and CER are summarized in Table 3.

7.2 Evaluation

In the following we concentrate on the comparison of WER and CER before and after automatic post-correction. As a baseline for our system we chose the strongest single-handed module (SMT on character-level trained on Werther data).

training set	system	test$_{init}$		test$_{unk}$	
		WER	CER	WER	CER
	original text	23.5	15.1	36.7	30.0
train	baseline	22.0	13.2	26.6	26.3
	overall system	**4.7**	**8.0**	**15.4**	**19.6**
train$_{ext}$	baseline	21.1	11.7	24.0	20.4
	overall system	**4.4**	**7.2**	**15.2**	**16.4**

Table 4: WER and CER for both test sets before and after automatic post-correction for the system trained with the small training set (train) and the larger training set (train$_{ext}$). Baselines: the original text coming from the OCR system and the character-level SMT system trained on the Werther data.

Overall performance As indicated previously, our test sets differ with respect to their similarity to the training set. The results for both test scenarios for systems trained on our two training sets are summarized in Table 4. The results from test$_{init}$ and test$_{unk}$ show that our system performs considerably better than the baseline and can improve quality of the OCR output considerably.

For test$_{unk}$, the system improves the quality by almost 20 points of WER from 36.7 to 15.4 and over 10 points in CER from 30.0 to 19.6. For test$_{init}$, our system improves the quality of the text with a reduction of approximately 20 points of WER from 23.5 to 4.7 and 7 points in CER from 15.1 to 8.0. It is not surprising that the decrease in WER is stronger than the decrease in CER. This is due to the fact that many words contain more than one error and require more than one character level operation to get from the incorrect to the correct string.

Just slight improvement can be shown by adding training material to the Werther-specific parts of the system (cf. train$_{ext}$ row of Table 4). Merely the CER can be improved whereas the WER stays about the same. The improvement in test$_{unk}$ is higher than for test$_{init}$.

Module specific analysis Since a WER and CER evaluation is not expedient for all modules as they were designed to correct specific problems and not the entirety of them, we look into the specialized modules in terms of correct suggestions contributed to the suggestion pool and correct suggestions only suggested by one module (unique suggestions). As the system including the extended training set train$_{ext}$ delivered slightly better results, in the following we will describe the contribution of the single modules

	test_{init}			test_{unk}		
module	# overcorrected	# corrected	# unique correct	# overcorrected	# corrected	# unique correct
SMT Werther token	128	364	10	209	1,089	0
SMT Werther character	235	684	0	700	1,919	0
SMT Werther cascaded	273	697	2	728	1,933	4
SMT DTA token	2,179	229	8	1,627	893	19
SMT DTA character	4121	372	22	3,143	1,530	115
text-internal vocab	3,317	131	16	4,142	244	60
word split	594	3	0	720	45	2
spell check	1,329	219	15	2,819	731	40
compound	222	0	0	169	2	2
overall system	238	2171	-	675	2,642	-

Table 5: Number of overcorrected, corrected and uniquely corrected words per module out of 17,367 tokens in test_{init} (2,726 erroneous words) and 13,304 tokens in test_{unk} (4,141 erroneous words)

to the overall performance of this system (cf. Table 5). For test_{unk} the number of corrected tokens along with the number of overcorrections is higher than for test_{init} throughout all modules. Clearly, for test_{init} the Werther-specific modules are strongest. The more general modules prove useful for test_{unk}. The number of corrected words increases for the SMT module trained on DTA data on character-level. The usefulness of the module extracting specific words (text-internal vocab) as well as the general SMT model and the spell checker becomes evident in terms of unique suggestions contributed by those modules.

The analysis of the output of the individual modules and their contribution to the overall system uncovers an issue: those modules that produce a high number of incorrect suggestions, thus overcorrecting actually correct input tokens, are at the same time those modules that are the only ones producing correct suggestions for some of the incorrect input words. Consequently, those uniquely suggested corrections are not chosen in the decision modules due to an overall weak performance of this module. These suggestions are often crucial to the texts like the suggestions by the special vocabulary module which contain named entities or words specific to the time period. For our test_{unk} set, the text-internal vocabulary module yields around 60 unique suggestions, out of which 15 are names (Friedrich, Amalia) or words really specific to the text (*Auftrit* spelled with one t instead of two).

Challenges In the context of literature OCR post-correction is a challenging problem since the texts themselves can be considered *non-standard text*. The aim is not to bring the text at hand to an agreed upon standard form but to digitize exactly what was contained in the print version. This can be far from the standard form of a language. In one of our texts, we find a character speaking German with a strong dialect. Her speech contains a lot of words that are incorrect in standard German, however, the goal is it to preserve this "errors" in the digital version. Thus, correction merely on the basis of the OCR text without consulting the printed version or an image-digitized facsimile, can essentially never be perfect. It follows, that the integration of automatic post-correction techniques into the character recognition process could lead to further improvements.

7.3 Adaptability

Reusability as a key concept in NLP for DH originates in the time limitations given in such projects. Since DH projects do not evolve around the development of tools but the analysis performed with the help this tools in order to answer a specific question, the tools are expected to be delivered in an early phase of collaborative projects. From-scratch development easily exceeds this time limits. We show that our OCR post-correction system is modular enough to be adjusted to correct texts from other languages by training it for two other languages, English and French, with data released in the OCR post-correction competition organized in the context of ICDAR 2017[9]. The texts originate from the the last four centuries and come from different collections and therefore have been digitized using different OCR systems. The data is summarized in Table 6[10].

We adjust our system to the language by retraining the SMT models and including spell-checkers for the respective languages. Due to the modular architecture these adjustments can be made eas-

[9] https://sites.google.com/view/icdar2017-postcorrectionocr/home, 3.07.2017.

[10] The test set does not comply with the official shared task set since the manually corrected data is not yet available for the test set. We test on a combination of periodicals and monographs.

language	$train_{ocr}$	$train_{gold}$	$dev1_{ocr}$	$dev1_{gold}$	$dev2_{ocr}$	$dev2_{gold}$	$test_{ocr}$	$test_{gold}$
English	309,080	282,738	71,049	65,480	13,000	11,966	14,302	12,859
French	805,438	783,371	167,473	163,373	9,566	9,216	12,289	11,780

Table 6: Number of tokens in the English and French corpus provided by the competition on OCR-postcorrection.

ily and with a low expenditure of time. Since the datasets are compilation of a variety of texts, we use all modules but the domain-specific SMT models. We solely include one token-level and character-level SMT module for each language.

language	system	WER	CER
English	original	29.4	28.4
	SMT Cascaded	22.7	23.6
	overall system	22.1	24.5
French	original text	13.3	25.0
	SMT Cascaded	9.9	20.0
	overall system	8.7	21.5

Table 7: The results reported in word error rate (WER) and character error rate (CER) for the English and French test set.

The strongest unique module for these two languages is the subsequent combination of the character-level SMT and the token-level SMT models (Cascaded). For English it performs just slightly worse on WER and even outperforms the overall system on the CER. For French, the overall system is clearly stronger than the Cascaded SMT system with more than 1 percent improvement of WER but also performs worse in terms of CER by 1.5 percent. Generally, the OCR post-correction system achieves about 25% reduction of WER for English and over 30% reduction in WER for French.

8 Digitization workflow

We integrate the automatic OCR process with tesseract and our automatic post-correction system into a workflow which results in an hocr file, an XML format which is readable by PoCoTo (Vobl et al., 2014) a tool for supporting manual post-correction of OCRed text through alignment of image and digitized text. The upload of scans or images is provided online via a webapplication[11]. This shields the user from the technicalities of the correction process and provides them with the input for the PoCoTo tool.

The implementation of an easy-to-handle workflow is an often underemphasized aspect of DH. It needs to be intuitive enough to not absorb the time ion has been saved via automation. Since the final post-correction step requires that the human corrector compares the digitized version with the scan, presenting both next to each other is an ideal scenario. This functionality is one of the main strengths of PoCoTo, a visual correction tool, supporting manually initiated correction operations and batch correction of the same error.

9 Conclusion

We can show that the enhancement of a general, adaptable architecture by including small but specific data sets can improve results within a specific domain. Moreover, the combination of different techniques for of OCR post-correction is significantly superior to single techniques. Especially the integration of SMT models on token level and character level contributes to the overall success of the system. Due to the complexity of OCR post-correction, there cannot be a general solution. Even though the ranking algorithm achieves large improvement, further potential lies in the inclusion of fine-tuned language models since the decision process highly depends upon it. The intrinsic characteristic of literature as being *non-standard* complicates the task. However, techniques that focus on these features like our module that is specialized on extracting text-specific vocabulary show promising results for e.g. named entity correction.

10 Acknowledgements

We thank the German Ministry of Education and Research (BMBF) for supporting this research completed within the Center for Reflected Text Analytics (CRETA). Furthermore, we acknowledge the support of our colleagues at Deutsches Textarchiv.

[11] http://clarin05.ims.uni-stuttgart.de/ocr/, for access please contact the author.

References

Ahmad Abdulkader and Mathew R. Casey. 2009. Low Cost Correction of OCR Errors Using Learning in a Multi-Engine Environment. In *10th International Conference on Document Analysis and Recognition, ICDAR 2009, Barcelona, Spain, 26-29 July 2009*, pages 576–580.

Haithem Afli, Zhengwei Qiu, Andy Way, and Praic Sheridan. 2016. Using SMT for OCR Error Correction of Historical Texts. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Youssef Bassil and Mohammad Alwani. 2012. OCR Post-Processing Error Correction Algorithm Using Google's Online Spelling Suggestion. *Journal of Emerging Trends in Computing and Information Sciences*, 3(1):90–99.

Marcel Bollmann, Julia Krasselt, and Florian Petran. 2012. Manual and semi-automatic normalization of historical spelling Case studies from Early New High German. In *Proceedings of KONVENS 2012 (LThist 2012 workshop*, pages 342–350.

Simon Clematide, Lenz Furrer, and Martin Volk. 2016. Crowdsourcing an OCR Gold Standard for a German and French Heritage Corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*

Orphée De Clercq, Bart Desmet, Sarah Schulz, Els Lefever, and Veronique Hoste. 2013. Normalization of Dutch user-generated content. In *Proceedings of Recent Advances in Natural Language Processing*, pages 179–188. INCOMA.

Apurva A. Desai. 2010. Gujarati Handwritten Numeral Optical Character Reorganization Through Neural Network. *Pattern Recogn.*, 43(7):2582–2589.

Jacob Eisenstein. 2013. What to do about bad language on the Internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369, Atlanta, Georgia. Association for Computational Linguistics.

Alexander Geyken, Susanne Haaf, Bryan Jurish, Matthias Schulz, Christian Thomas, and Frank Wiegand. 2012. TEI und Textkorpora: Fehlerklassifikation und Qualittskontrolle vor, whrend und nach der Texterfassung im Deutschen Textarchiv. In *Jahrbuch für Computerphilologie*, page online.

Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. 2009. A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):855–868.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*.

Justine T. Kao and Dan Jurafsky. 2015. A computational analysis of poetic style: Imagism and its influence on modern professional and amateur poetry. *Linguistic Issues in Language Technology*, 12(3):1–31.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Okan Kolak and Philip Resnik. 2002. OCR Error Correction Using a Noisy Channel Model. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 257–262, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Rafael Llobet, Jose-Ramon Cerdan-Navarro, Juan-Carlos Perez-Cortes, and Joaquim Arlandis. 2010. OCR Post-processing Using Weighted Finite-State Transducers. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, ICPR '10, pages 2021–2024, Washington, DC, USA. IEEE Computer Society.

Stoyan Mihov, Klaus U. Schulz, Christoph Ringlstetter, Veselka Dojchinova, and Vanja Nakova. 2005. A Corpus for Comparative Evaluation of OCR Software and Postcorrection Techniques. In *Eighth International Conference on Document Analysis and Recognition (ICDAR 2005), 29 August - 1 September 2005, Seoul, Korea*, pages 162–166.

Smitha Milli and David Bamman. 2016. Beyond Canonical Texts: A Computational Analysis of Fanfiction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2048–2053.

Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA. Association for Computational Linguistics.

Juan Carlos Pérez-Cortes, Juan-Carlos Amengual, Joaquim Arlandis, and Rafael Llobet. 2000.

Stochastic Error-Correcting Parsing for OCR Post-Processing. In *15th International Conference on Pattern Recognition, ICPR'00, Barcelona, Spain, September 3-8, 2000*, pages 4405–4408.

Ulrich Reffle and Christoph Ringlstetter. 2013. Unsupervised Profiling of OCRed Historical Documents. *Pattern Recogn.*, 46(5):1346–1357.

Martin Reynaert. 2008a. All, and only, the Errors: more Complete and Consistent Spelling and OCR-Error Correction Evaluation. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Martin Reynaert. 2008b. Non-interactive OCR Post-correction for Giga-scale Digitization Projects. In *Proceedings of the 9th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing'08, pages 617–630, Berlin, Heidelberg. Springer-Verlag.

Christoph Ringlstetter, Klaus U. Schulz, and Stoyan Mihov. 2007. Adaptive Text Correction with Web-crawled Domain-dependent Dictionaries. *ACM Trans. Speech Lang. Process.*, 4(4).

Helmut Schmid. 1997. Probabilistic Part-of-Speech Tagging Using Decision Trees. In Daniel Jones and Harold Somers, editors, *New Methods in Language Processing*, Studies in Computational Linguistics, pages 154–164. UCL Press, London, GB.

Sarah Schulz, Guy De Pauw, Orphée De Clercq, Bart Desmet, Véronique Hoste, Walter Daelemans, and Lieve Macken. 2016. Multimodular Text Normalization of Dutch User-Generated Content. *ACM Trans. Intell. Syst. Technol.*, 7(4):61:1–61:22.

Ray Smith and Google Inc. 2007. An overview of the Tesseract OCR Engine. In *Proc. 9th IEEE Intl. Conf. on Document Analysis and Recognition (ICDAR)*, pages 629–633.

Carolyn Strange, Daniel McNamara, Josh Wodak, and Ian Wood. 2014. Mining for the Meanings of a Murder: The Impact of OCR Quality on the Use of Digitized Historical Newspapers. *Digital Humanities Quarterly*, 8(1).

Christian M. Strohmaier, Christoph Ringlstetter, Klaus U. Schulz, and Stoyan Mihov. 2003. Lexical Postcorrection of OCR-Results: The Web as a Dynamic Secondary Dictionary? In *7th International Conference on Document Analysis and Recognition (ICDAR 2003), 2-Volume Set, 3-6 August 2003, Edinburgh, Scotland, UK*, pages 1133–1137.

Hiroyasu Takahashi, Nobuyasu Itoh, Tomio Amano, and Akio Yamashita. 1990. A spelling correction method and its application to an OCR system. *Pattern Recognition*, 23(3-4):363–377.

Xiang Tong and David A. Evans. 1996. A Statistical Approach to Automatic OCR Error Correction in Context. In *Fourth Workshop on Very Large Corpora*, pages 88–100.

Thorsten Vobl, Annette Gotscharek, Uli Reffle, Christoph Ringlstetter, and Klaus U. Schulz. 2014. PoCoTo - an Open Source System for Efficient Interactive Postcorrection of OCRed Historical Texts. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, DATeCH '14, pages 57–61, New York, NY, USA. ACM.

Martin Volk, Torsten Marek, and Rico Sennrich. 2010. Reducing OCR errors by combining two OCR systems. In *Proceedings of the ECAI 2010 Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (LaTeCH 2010)*, pages 61–65.

Learning to Predict Charges for Criminal Cases with Legal Basis

Bingfeng Luo[1], **Yansong Feng**[*1], **Jianbo Xu**[2], **Xiang Zhang**[2] and **Dongyan Zhao**[1]
[1]Institute of Computer Science and Technology, Peking University, China
[2]Beijing Institute of Big Data Research, China
{bf_luo, fengyansong, xujb, xiang.zhang, zhaody}@pku.edu.cn

Abstract

The charge prediction task is to determine appropriate charges for a given case, which is helpful for legal assistant systems where the user input is fact description. We argue that relevant law articles play an important role in this task, and therefore propose an attention-based neural network method to jointly model the charge prediction task and the relevant article extraction task in a unified framework. The experimental results show that, besides providing legal basis, the relevant articles can also clearly improve the charge prediction results, and our full model can effectively predict appropriate charges for cases with different expression styles.

1 Introduction

The task of automatic charge prediction is to determine appropriate charges, such as *fraud*, *larceny* or *homicide*, for a case by analyzing its textual fact description. Such techniques are crucial for legal assistant systems, where users could find similar cases or possible penalties by describing a case with their own words. This is helpful for non-legal professionals to get to know the legal basis of their interested cases, e.g., cases they or their friends are involved in, since the massive legal materials and the lack of knowledge of legal jargons make it hard for outsiders to do it on their own.

However, predicting appropriate charges based on fact descriptions is not trivial: (1) The differences between two charges can be subtle, for example, in the context of criminal cases in China, distinguishing *intentional homicide* from *intentional injury* would require to determine, from the fact description, whether the defendant *intended to kill* the victim, or just *intended to hurt the victim, who, unfortunately died of severe injury*. (2) Multiple crimes may be involved in a single case, which means we need to conduct charge prediction in the multi-label classification paradigm. (3) Although we can expect an off-the-shelf classification model to learn to label a case with corresponding charges through massive training data, it is always more convincing to make the prediction with its involved law articles explicitly shown to the users, as legal basis to support the prediction. This issue is crucial in countries using *the civil law system*, e.g., China (except Hong Kong), where judgements are made based on statutory laws only. For example, in Fig. 1, a judgement document in China always includes relevant law articles (in the court view part) to support the decision. Even in countries using *the common law system*, e.g., the United States (except Louisiana), where the judgement is based mainly on decisions of previous cases, there are still some statutory laws that need to be followed when making decisions.

Existing attempts formulate the task of automatic charge prediction as a single-label classification problem, by either adopting a k-Nearest Neighbor (KNN) (Liu et al., 2004; Liu and Hsieh, 2006) as the classifier with shallow textual features, or manually designing key factors for specific charges to help text understanding (Lin et al., 2012), which make those works hard to scale to more types of charges. There are also works addressing a related task, finding the law articles that are involved in a given case. A simple solution is to convert this multi-label problem into a multi-class classification task by only considering a fixed set of article combinations (Liu and Liao, 2005; Liu and Hsieh, 2006), which can only be applied to a small set of articles and does not fit to real applications. Recent improvement takes a two-step approach by performing a preliminary classification first and then re-ranking the results with

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2727–2736
Copenhagen, Denmark, September 7–11, 2017.

word-level and article-level features (Liu et al., 2015). These efforts advance the applications of machine learning and natural language processing methods into legal assistance services, however, they are still in an early stage, e.g., relying on expert knowledge, using relatively simple classification paradigms, and shallow textual analysis. More importantly, related tasks, e.g., charge prediction and relevant article extraction, are treated independently, ignoring the fact that they could benefit from each other.

Recent advances in neural networks enable us to jointly model charge prediction and relevant article extraction in a unified framework, where the latent correspondence from the fact description about a case to its related law articles and further to its charges can be explicitly addressed by a two-stack attention mechanism. Specifically, we use a sentence-level and a document-level Bi-directional Gated Recurrent Units (Bi-GRU) (Bahdanau et al., 2015) with a stack of fact-side attention components to model the correlations among words and sentences, in order to capture the whole story as well as important details of the case. Given the analysis of the fact description, we accordingly learn a stack of article-side attention components to attentively select the most supportive law articles from the statutory laws to support our charge prediction, which is investigated in the multi-label paradigm.

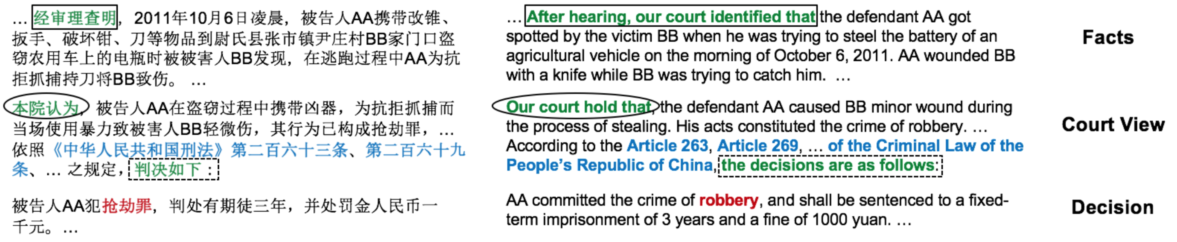
We evaluate our model in the context of predicting charges for criminal cases in China. We collect publicly available judgement documents from China's government website, from which we can automatically extract *fact descriptions*, *relevant law articles* and *the charges* using simple rules, as shown in Figure 1. Experimental results show that our neural network method can effectively predict appropriate charges for a given case, and also provide relevant law articles as legal basis to support the prediction. Our experiments also provide quantitive analysis about the effect of fact-side and article-side information on charge predicion, and confirm that, apart from providing legal basis, relevant articles also contain useful information that can help to improve charge prediction in the civil law system. We also examine our model on the news reports about criminal cases. Although trained on judgement documents, our model can still achieve promising performance on news data, showing a reasonable generalization ability over different expression styles.

2 Related Work

The charge prediction task aims at finding appropriate charges based on the facts of a case. Previous works consider this task in a multi-class classification framework, which takes the fact description as input and outputs a charge label. (Liu et al., 2004; Liu and Hsieh, 2006) use KNN to classify 12 and 6 criminal charges in Taiwan. However, except for the inferior scalability of the KNN method, their word-level and phrase-level features are too shallow to capture sufficient evidence to distinguish similar charges with subtle differences. (Lin et al., 2012) propose to make deeper understanding of a case by identifying charge-specific factors that are manually designed for 2 charges. This method also suffers from the scalability issue due to the human efforts required to design and annotate these factors for each pair of charges. Our method, however, employs Bi-GRU and a two-stack attention mechanism to make comprehensive understanding of a case without relying on explicit human annotations.

Within the civil law system, there are some works focusing on identifying applicable law articles for a given case. (Liu and Liao, 2005; Liu and Hsieh, 2006) convert this multi-label problem into a multi-class classification problem by only considering a fixed set of article combinations, which cannot scale well since the number of possible combinations will grow exponentially when a larger set of law articles are considered. (Liu et al., 2015) instead design a scalable two-step approach by first using Support Vector Machine (SVM) for preliminary article classification, and then re-ranking the results using word level features and co-occurence tendency among articles. We also use SVM to extract top k candidate articles, but further adopt Bi-GRU and article-side attention to better understand the texts and exploring the correlation among articles. More importantly, we optimize the article extraction task within our charge prediction framework, which not only provides another view to understand the facts, but also serves as legal basis to support the final decision.

Another related thread of work is to predict the overall outcome of a case. The target can be which party will the outcome side with (Aletras et al., 2016), or whether the present court will affirm or reverse the decision of a lower court (Katz et al.,



…经审理查明，2011年10月6日凌晨，被告人AA携带改锥、扳手、破坏钳、刀等物品到尉氏县张市镇尹庄村BB家门口盗窃农用车上的电瓶时被被害人BB发现，在逃跑过程中AA为抗拒抓捕持刀将BB致伤。 …	…After hearing, our court identified that the defendant AA got spotted by the victim BB when he was trying to steel the battery of an agricultural vehicle on the morning of October 6, 2011. AA wounded BB with a knife while BB was trying to catch him. …	**Facts**
本院认为，被告人AA在盗窃过程中携带凶器，为抗拒抓捕而当场使用暴力致被害人BB轻微伤，其行为已构成抢劫罪，…依照《中华人民共和国刑法》第二百六十三条、第二百六十九条、… 之规定，判决如下：	Our court hold that the defendant AA caused BB minor wound during the process of stealing. His acts constituted the crime of robbery. … According to the Article 263, Article 269, … of the Criminal Law of the People's Republic of China, the decisions are as follows:	**Court View**
被告人AA犯抢劫罪，判处有期徒三年，并处罚金人民币一千元。…	AA committed the crime of robbery, and shall be sentenced to a fixed-term imprisonment of 3 years and a fine of 1000 yuan. …	**Decision**

Figure 1: An example judgement document excerpt of a criminal case in our dataset. Names are anonymized as AA and BB. Rectangulars, ellipses and dashed rectangulars refer to the clauses that usually indicate the beginning of the facts part, the court view part and the decision part, respectively. Articles and charges are extracted with regular expressions and a charge list.

2016). Our work differs from them in that, instead of binary outcome (the latter one also contains an *other* class), we step further to focus on the detailed results of a case, i.e., the charges, where the output may contain multiple labels.

We also share similar spirit with the legal question answering task (Kim et al., 2014a), which aims at answering the yes/no questions in the Japanese legal bar exams, that we all believe that relevant law articles are important for decisions in the civil law system. Different from ours, this task requires participants to extract relevant Japanese Civil Code articles first, and then use them to answer the yes/no questions. The former phase is often treated as an information retrieval task, and the latter phase is considered as a textual entailment task (Kim et al., 2014b; Carvalho et al., 2016).

In the field of artificial intelligence and law, there are also works trying to find relevant cases given the input query (Raghav et al., 2016; Chen et al., 2013), which is crutial for decision making in the common law system. Rather than finding relevant cases, our work focuses on predicting specific charges, and we also emphasize the importance of law articles in decision making, which is important in the civil law system where the decisions are made based solely on statutory laws.

Our work is also related to the task of document classification, but mainly differs in that we also need to automatically identify applicable law articles to support and improve the charge prediction. Recently, various neural network (NN) architectures such as Convolutional Neural Network (CNN) (Kim, 2014) and Recurrent Neural Network (RNN) have been used for document embedding, which is further used for classification. (Tang et al., 2015) propose a two-layer scheme, RNN or CNN for sentence embedding, and another RNN for document embedding. (Yang et al., 2016) further use global context vectors to attentively distinguish informative words or sentences from non-informative ones during embedding, which we share similar spirit with. But, we take a more flexible and descriptive two-stack attention mechanism, one stack for fact embedding, and the other for article embedding which is dynamically generated for each instance according to the fact-side clues as extra guidance. Another difference is the multi-label nature of our task, where, rather than optimizing as multiple binary classification tasks (Nam et al., 2014), we convert the multi-label target to label distribution during training with cross entropy as loss function (Kurata et al., 2016), and use a threshold tuned on validation set to produce the final prediction, which performs better in our pilot experiments.

3 Data Preparation

Our data are collected from China Judgements Online[1], where the Chinese government has been publishing judgement documents since 2013. We randomly choose 50,000 documents for training, 5,000 for validation and 5,000 for testing. To ensure enough training data for each charge, we only classify the charges that appear more than 80 times in the training data, and treat documents with other charges as negative data. As for law articles, we consider those in the Criminal Law of the People's Republic of China. The resulting dataset contains 50 distinct charges, 321 distinct articles, averagely 383 words per fact description, 3.81 articles per case, and 3.56% cases with more than one charges.

An example judgement document is shown in Figure 1, where we highlight the indicator clauses that we used to divide a document into three pieces

[1] http://wenshu.court.gov.cn

and extract *fact description*, *articles*, and *charges* from each piece, respectively. We use a manually collected charge list to identify all the charges, and law articles are extracted by regular expressions[2]. The extracted charges and articles are considered as gold standard charges and articles for the corresponding fact description. We also masked all the charges in fact descriptions, since although rare, charge names sometimes may appear in the fact description part.

Currently, it is hard and expensive to match the facts related to different defendants with their corresponding charges. We therefore only consider the cases with one defendant, and leave the challenging multi-defendant cases for future work. Although this simplification may change the real-world charge distribution, it enables us to automatically build large scale high quality dataset without relying on annotations from legal practitioners.

4 Our Approach

As depicted in Fig. 2, our approach contains the following steps: (1) The input fact description is fed to a document encoder to generate the fact embedding $\mathbf{d}_f$, where $\mathbf{u}_{fw}$ and $\mathbf{u}_{fs}$ are global word-level and sentence-level context vectors used to attentively select informative words and sentences. (2) Concurrently, the fact description is also passed to an article extractor to find top k relevant law articles. (3) These articles are embedded by another document encoder, and passed to an article aggregator to attentively select supportive articles, and produce the aggregated article embedding $\mathbf{d}_a$. Specifically, three context vectors, i.e., $\mathbf{u}_{aw}$, $\mathbf{u}_{as}$ and $\mathbf{u}_{ad}$, are dynamically generated from $\mathbf{d}_f$, to produce attention values within the article document encoder and the article aggregator. (4) Finally, $\mathbf{d}_f$ and $\mathbf{d}_a$ are concatenated and passed to a softmax classifier to predict the charge distribution for the input case.

4.1 Document Encoder

Intuitively, a sentence is a sequence of words and a document is a sequence of sentences. The document embedding problem, therefore, can be converted to two sequence embedding problems (Tang et al., 2015; Yang et al., 2016). As shown in Fig. 3, we can first embed each sentence using a sentence-level sequence encoder, and then aggregate them with a document-level sequence encoder to produce the document embedding $\mathbf{d}$. While these two encoders can have different architectures, we use the same here for simplicity.

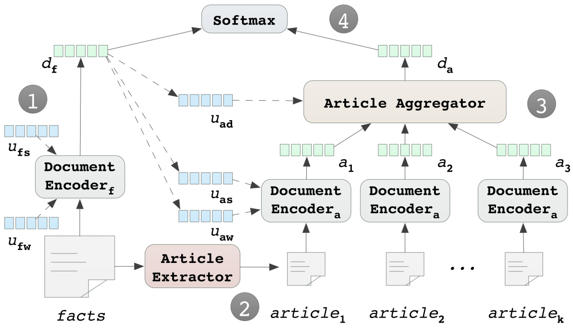


Figure 2: Overview of Our Model

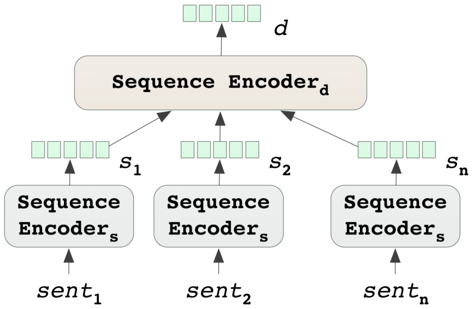


Figure 3: Document Encoder Framework

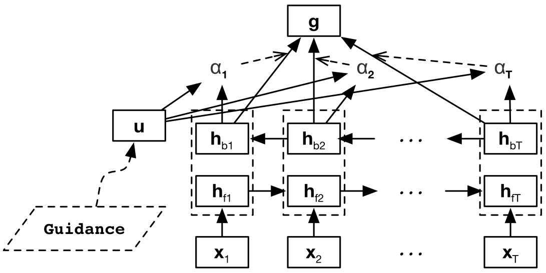


Figure 4: Attentive Sequence Encoder

Bi-GRU Sequence Encoder A challenge in building a sequence encoder is how to take the correlation among different elements into consideration. A promising solution is Bi-directional Gated Recurrent Units (Bi-GRU) (Bahdanau et al., 2015), which encodes the context of each element by using a gating mechanism to track the state of sequence. Specifically, Bi-GRU first uses a forward and a backward GRU (Cho et al., 2014), which is a kind of RNN, to encode the sequence in two opposite directions, and then concatenates the states of both GRUs to form its own states.

Given a sequence $[\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T]$ where $\mathbf{x}_t$ is

[2] The regular expression used to extract law articles: “第[、零〇一二两三四五六七八九十百千0-9]+条(之[一二两三四五六七八九十])?)”

the input embedding of element t, the state of Bi-GRU at position t is:

$$\mathbf{h}_t = [\mathbf{h}_{ft}, \mathbf{h}_{bt}] \tag{1}$$

where $\mathbf{h}_{ft}$ and $\mathbf{h}_{bt}$ are the states of the forward and backward GRU at position t. The final sequence embedding is either the concatenation of $\mathbf{h}_{fT}$ and $\mathbf{h}_{b1}$ or simply the average of $\mathbf{h}_t$.

Attentive Sequence Encoder However, directly using $[\mathbf{h}_{fT}, \mathbf{h}_{b1}]$ for sequence encoding often fails to capture all the information when the sequence is long, while using the average of $\mathbf{h}_t$ also has the drawback of treating useless elements equally with informative ones. Inspired by (Yang et al., 2016), we use a context vector to attentively aggregate the elements, but instead of using a global context vector, we allow the context vector to be dynamically generated when extra guidance is available (see Sec. 4.2).

As shown in Fig. 4, given the Bi-GRU state sequence $[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_T]$, our attentive sequence encoder calculates a sequence of attention values $[\alpha_1, \alpha_2, ..., \alpha_T]$, where $\alpha_t \in [0, 1]$ and $\sum_t \alpha_t = 1$. The final sequence embedding $\mathbf{g}$ is calculated by:

$$\mathbf{g} = \sum_{t=1}^{T} \alpha_t \mathbf{h}_t; \quad \alpha_t = \frac{exp(tanh(\mathbf{W}\mathbf{h}_t)^T\mathbf{u})}{\sum_t exp(tanh(\mathbf{W}\mathbf{h}_t)^T\mathbf{u})} \tag{2}$$

where $\mathbf{W}$ is a weight matrix, and $\mathbf{u}$ is the context vector to distinguish informative elements from non-informative ones.

By using this sequence encoder for fact embedding, the fact-side attention module actually contains two components, i.e., the word-level and sentence-level, using $\mathbf{u}_{fw}$ and $\mathbf{u}_{fs}$ as their global context vectors, respectively.

4.2 Using Law Articles

One of the challenges of using law articles to support charge prediction lies in the fact that statutory laws contain a large number of articles, which makes applying complex models to these articles directly time-consuming, and thus hard to scale. The multi-label nature of relevant article extraction also requires a model that can output multiple articles. We thus adopt a two-step approach, specifically, we first build a fast and easy-to-scale classifier to filter out a large fraction of irrelevant articles, and retain the top k articles. Then, we use neural networks to make comprehensive understanding of the top k articles, and further use the article-side attention module to select the most supportive ones for charge prediction.

Top k Article Extractor We treat the relevant article extraction task as multiple binary classifications. Specifically, we build a binary classifier for each article, focusing on its relevance to the input case, which results in 321 binary classifiers corresponding to the 321 distinct law articles appearing in our dataset. When more articles are considered, we can simply add more binary classifiers accordingly, with the existing classifiers untouched.

Similar to the preliminary classification phase of (Liu et al., 2015), we also use word-based SVM as our binary classifier, which is fast and performs well in text classification (Joachims, 2002; Wang and Manning, 2012). Specifically, we use bag-of-words TF-IDF features, chi-square for feature selection and linear kernel for binary classification.

Article Encoder Since each law article may contain multiple sentences, as shown in Fig. 2, we also use the document encoder described in Sec. 4.1 to produce an embedding $\mathbf{a}_j, j \in [1, k]$, for each article in the top k extracted articles. While using similar architecture, this article encoder differs from the fact encoder that, instead of using global context vectors, its word-level context vector $\mathbf{u}_{aw}$ and sentence-level context vector $\mathbf{u}_{as}$ are dynamically generated for each case according to its corresponding fact embedding $\mathbf{d}_f$:

$$\mathbf{u}_{aw} = \mathbf{W}_w\mathbf{d}_f + \mathbf{b}_w; \quad \mathbf{u}_{as} = \mathbf{W}_s\mathbf{d}_f + \mathbf{b}_s \tag{3}$$

where $\mathbf{W}_*$ is the weight matrix and $\mathbf{b}_*$ is the bias. The context vectors, $\mathbf{u}_{aw}$ and $\mathbf{u}_{as}$, are used to produce the word-level and sentence-level attention values, respectively. Through the dynamic context vectors, the fact embedding $\mathbf{d}_f$ actually guides our model to attend to informative words or sentences with respect to the facts of each case, rather than just selecting generally informative ones.

Attentive Article Aggregator The article aggregator aims to find supportive articles for charge prediction from the top k extractions, and accordingly produce an aggregated article embedding. Although the order of the top k extracted articles is not fully reliable, (Vinyals et al., 2016) suggests that it is still beneficial to use a bi-directional RNN to embed the context of each element even in a set, where the order does not exist. In our task, bi-directional RNN can help to utilize the co-occurrence tendency of relevant articles.

Specifically, we use the attentive sequence encoder in Sec. 4.1 to produce the aggregated article embedding $\mathbf{d}_a$. Again, to guide the attention with fact descriptions, we dynamically generate the article-level context vector $\mathbf{u}_{ad}$ by:

$$\mathbf{u}_{ad} = \mathbf{W}_d \mathbf{d}_f + \mathbf{b}_d \tag{4}$$

The attention values produced by the attentive sequence encoder can be seen as the relevance of each article to the input case, which can be used to rank and filter the top k articles. The results can be shown to users as legal basis for charge prediction.

4.3 The Output

To make the final charge prediction, we first concatenate the document embedding $\mathbf{d}_f$ and the aggregated article embedding $\mathbf{d}_a$, and feed them to two consecutive full connection layers to generate a new vector $\mathbf{d}'$, which is then passed to a softmax classifier to produce the predicted charge distribution. We use the validation set to determine a threshold τ, and consider all the charges with output probability higher than τ as positive predictions. The input to the first full connection layer can also be only $\mathbf{d}_f$ or $\mathbf{d}_a$, which means we use only fact or article to make the prediction.

The loss function for training is cross entropy:

$$Loss = -\sum_{i=1}^{N}\sum_{l=1}^{L} y_{il} log(o_{il}) \tag{5}$$

where N is the number of training data, L is the number of charges, y_{il} and o_{il} are the target and predicted probability of charge l for case i. The target charge distribution $\mathbf{y}_i$ is produced by setting positive labels to $\frac{1}{m_i}$ and negative ones to 0, where m_i is the number of positive labels in case i.

Supervised Article Attention We can also utilize the gold-standard law articles naturally in the judgement documents to supervise the article attention during training. Specifically, given the top k articles, we want the article attention distribution $\boldsymbol{\alpha} \in \mathbb{R}^k$ to simulate the target article distribution $\mathbf{t} \in \mathbb{R}^k$, where $t_j = \frac{1}{k'}$ if article j belongs to the gold-standard articles and $t_j = 0$ otherwise. Here k' is the number of gold-standard articles in the top k extractions. We, again, use cross entropy, and the loss function is:

$$Loss = -\sum_{i=1}^{N}(\sum_{l=1}^{L} y_{il} log(o_{il}) + \beta \sum_{j=1}^{k} t_{ij} log(\alpha_{ij})) \tag{6}$$

where β is the weight of the article attention loss.

5 Experiments

5.1 Experimental Setup

We use HanLP[3] for Chinese word segmentation and POS tagging. Word embeddings are trained using word2vec (Mikolov et al., 2013) on judgement documents, web pages from several legal forums and Baidu Encyclopedia. The resulting word embeddings contain 573,353 words, with 100 dimension. We randomly initialize a 50-d vector for each POS tag, which is concatenated with the word embedding as the final input. Each GRU in the Bi-GRU is of size 75, the two full connection layers are of size 200 and 150. The relevant article extractor generates top 20 articles, the weight of the article attention loss (β in Eq. 6) is 0.1, and prediction threshold τ is 0.4. We use Stochastic Gradient Descent (SGD) for training, with learning rate 0.1, and batch size 8.

We compare our full model with two variations: without article attention supervision and only using facts for charge prediction. The latter one is similar to the state-of-art document classification model (Yang et al., 2016), but adapted to the multi-label nature of our problem. We also implement an SVM model, which is effective and scales well in many fact-description-related tasks in the field of artificial intelligence and law (Liu et al., 2015; Aletras et al., 2016). Specifically, the SVM model takes bag-of-words TF-IDF features as input, and uses chi-square to select top 2,000 features.

5.2 Charge Prediction Results

The charge distribution is imbalanced, and the top 5 charges take more than 60% of the cases. Therefore, we evaluate the charge prediction task using precision, recall and F1, in both micro- and macro-level. The macro-precision/recall are calculated by averaging the precision and recall of each charge, and the micro-precision/recall are averaged over each prediction.

As shown in Table 1, the basic `SVM_fact` model, which only takes fact descriptions as input, indeed proves to be a strong baseline. By

[3] https://github.com/hankcs/HanLP

Model	Precision	Recall	F1
	(*Micro-/Macro-*)		
SVM_fact	**93.94**/79.53	77.66/49.54	85.03/61.05
SVM_art	82.12/42.90	61.23/39.56	70.15/41.16
SVM_fact_art	91.77/71.33	72.10/45.85	80.76/55.82
NN_fact	91.30/**83.32**	87.39/74.99	89.31/78.94
NN_art	90.09/81.50	86.10/69.62	88.05/75.10
NN_fact_art	90.79/83.07	88.42/75.73	89.59/79.23
NN_fact_supv_art	91.80/82.44	**88.67/78.62**	**90.21/80.48**
SVM_fact_gold_art	**98.97**/94.58	95.39/83.21	97.15/88.53
NN_fact_gold_art	98.78/**95.26**	**98.24/95.57**	**98.51/95.42**

Table 1: Charge prediction results. Left and right side of the slash refer to micro and macro statistics, respectively. `gold_art` refers to using gold standard articles mentioned in judgements (marked in blue in Fig. 1), which is the upper bound for article-related modules.

contrast, our corresponding neural network model (`NN_fact`), which also only uses facts for prediction, outperforms `SVM_fact` by about 4% in micro-F1. Since `NN_fact` benefits from the pretrained word embeddings, the two-level Bi-GRU architecture, and the fact-side attention module, it can attentively recognize informative expressions from the description and better capture the underlying correspondence from fact descriptions to appropriate charges, even when there is less overlap in the words used among cases with the same charge, or when there are limited data (i.e., infrequent charges). This may explain that NN models have more balanced performance over different charges, leading to more prominent improvements over SVM ones in macro metrics, which usually have a strong bias towards frequent charges.

When we use both facts and extracted relevant law articles (that are admittedly noisy), the SVM version (`SVM_fact_art`) drops by around 5% than `SVM_fact`, showing that the SVM model cannot benefit from the extracted, thus noisy, relevant articles in such a straightforward way. However, our NN version (`NN_fact_article`) can still learn from the noisy article extractions through attentively aggregating those extracted articles even without direct guidance, thus improves `NN_fact` by around 0.4%. Furthermore, if we use the gold standard articles during training as supervision for the article attention (our full model, `NN_fact_supv_art`), the results can be further improved, achieving 90.21% and 80.48% in micro- and macro-F1, respectively. The improvements made by using relevant law articles actually indicates the nature of the civil law system that judgements are made based on statutory laws.

However, if we only use the extracted relevant articles to make prediction (`SVM_art` and `NN_art`[4]), the performance becomes worse. Even with the proved-helpful attentive aggregator, the model performs worst among all NN variants (though still better than `SVM_fact`). This indicates that it is necessary to consider both facts and relevant law articles for charge prediction, and, the fact that `NN_fact` outperforms `NN_art` also indicates that although the judgments are made based on the statutory laws in the civil law system, the logic employed by the court when making decisions, to some extent, may be implicitly captured through massive fact-charges paris.

Now the question is: *how much improvement can we have if we can make full use of the relevant law articles within the civil law system?* Let us consider an ideal situation where we can access both fact descriptions and gold standard law articles during testing, which could be considered as an upper bound scenario. The SVM version (`SVM_fact_gold_art`) significantly outperforms `SVM_fact_art` by more than 30% in macro-F1. And the NN version (`NN_fact_gold_art`) outperforms `NN_fact_supv_art` by over 8%. These comparisons confirm again that law articles play an important role for automatic judgement prediction, but the extracted relevant articles inevitably contain noise, which should be properly handled, e.g., using an attentively aggregation mechanism to distill valuable evidence to support charge prediction.

Case Study We study the model outputs and find certain star-like confusion patterns among the charges. For example, *intentional injury* is often confused with multiple charges like *intentional homicide* (when the victim is dead, the difference is whether the defendant intends to kill or just hurt the victim) and *picking quarrels and provoking troubles* (there may also exist injuries here). These charges usually share some similar fact descriptions, e.g., how the injuries are caused, and since *intentional injury* appears more frequently than the others, `SVM_fact` thus outputs *intentional injury* in most situations, and fails to distinguish these charges. However, by using Bi-GRU and the attention mechanism, `NN_fact` can at-

[4] `NN_art` uses fact embeddings to attentively aggregate relevant articles, but only use the aggregated article embedding $\mathbf{d}_a$, without fact embedding $\mathbf{d}_f$, for charge prediction.

β	**Prec@1**	**MAP**	**Charge_F1**
0	60.94	61.61	89.59/79.23
0.01	81.06	78.00	89.77/79.48
0.1	87.90	83.39	**90.21/80.48**
0.5	91.44	86.95	89.93/79.67
1	**92.66**	**88.24**	89.83/78.66

Table 2: Refined Article Extraction Performance

tend to important details of the facts and significantly improves the performance on these charges. When the direct supervision for articles is available, NN_fact_supv_art can enhance the interaction between certain pairs of fact descriptions and law articles, which helps to capture the subtle differences among similar charges, and further improves the performance on these situations.

5.3 Article Extraction Results

We also evaluate our SVM article extractor, which achieves 77.60%, 88.96%, 94.21% and 96.53% recall regarding the top 5, 10, 20 and 30 articles, respectively. Although simple, the SVM extractor can obtain over 94% recall for top 20 articles, which is good enough for further refinement. However, the micro-F1 of the extractor is only 61.08% in the test set, which will lead to severe error propagation problem if we use the prediction results of the extractor directly. Therefore, we design the article attention mechanism to handle the noise in the top 20 articles.

Table 2 shows the re-ranking results of our article attention module (column 2-3) and the corresponding charge prediction performances (column 4), under different weights for article attention (β in Eq. 6). Prec@1 refers to top 1 precision, and MAP refers to mean average precision. We can see that, even if there is no supervision over the article attention ($\beta = 0$), our model still has reasonable performance on re-ranking the k articles. When the attention supervision is employed, the extraction quality improves significantly, and keeps increasing as β goes up. However, the charge prediction performance does not always increase with the article extraction quality, and the best performance is achieved when $\beta = 0.1$. This is not surprising, since there exists a tradeoff between the benefits of more accurate article extraction and the less model capacity left for charge classification due to the increased emphasis on the article extraction performance. The promising article extraction results also confirm the ability of our model to provide legal basis for the charge prediction.

Model	**Precision**	**Recall**	**F1**
SVM_fact	**100.00**	40.20	57.34
NN_fact	87.14	59.80	70.93
NN_fact_art	87.18	66.67	75.56
NN_fact_supv_art	90.00	**70.59**	**79.12**

Table 3: Performance (micro statistics) on News

5.4 Performance on News Data

There are usually clear differences between the expressions used by legal practitioners and people without legal background, thus it is important to see how our model will perform on fact descriptions written by non-legal professionals.

We create a news dataset by asking 3 law school students to annotate the appropriate charges for 100 social news reports about criminal cases from two news websites[5], with 262 words on average and 25 distinct charges. The κ value is 0.83, indicating good consistency. The annotators are asked to have a disscussion to achieve an aggreement on inconsistent annotations. The results are shown in Table 3, where we only report micro statistics due to the relatively small size of the dataset compared with the number of distinct charges.

We can see that, SVM_fact suffers a significant drop in F1 on the news data, confirming the gap between the expressions used by legal practitioners and non-legal professionals, given the BOW nature of SVM_fact. Although SVM_fact cannot generalize well, the patterns learned by SVM_fact are reliable in themselves, leading to a high precision. It is not surprising that our NN models also suffer from the expression differences, but due to the effectiveness of our NN architecture, with about 10%∼15% less absolute drop in F1, and NN_fact_supv_art can still achieve 79.12% in F1. For example, the word 暴打 (beat up) is seldom used in judgement documents, making it hard for SVM_fact to correctly utilize 暴打 as an indicator for injury related charges, but, our NN models can associate it with its near-synonymy 殴打 (hit), which is a formal expression in judgement documents. Furthermore, the clear improvements from NN_fact to NN_fact_art, and further to NN_fact_supv_art prove again the importance of relevant law articles in supporting the charge prediction, even in news domain. The reasonable performance on news data also shows that our method do have the ability to help non-legal professionals.

[5] http://news.cn and http://people.com.cn

6 Conclusion

In this paper, we propose an attention-based neural network framework that can jointly model the charge prediction task and the relevant article extraction task, where the weighted relevant articles can serve as legal basis to support the charge prediction. The experimental results on judgement documents of criminal cases in China show the effectiveness of our model on both charge prediction and relevant article extraction. The comparison of different variants of our model also indicates the importance of law articles in making judicial decisions in the civil law system. By experimenting on news data, we show that, although trained on judgement documents, our model also has reasonable generalization ability on fact descriptions written by non-legal professionals. While promising, our model still cannot explicitly handle multi-defendant cases, and there is also a clear gap between our model and the upper bound improvement that relevant articles can achieve. We will leave these challenges for future work.

Acknowledgement

This work is supported by the National High Technology R&D Program of China (2015AA015403); the National Natural Science Foundation of China (61672057, 61672058); KLSTSPI Key Lab. of Intelligent Press Media Technology.

References

Nikolaos Aletras, Dimitrios Tsarapatsanis, Daniel Preoţiuc-Pietro, and Vasileios Lampos. 2016. Predicting judicial decisions of the european court of human rights: A natural language processing perspective. *PeerJ Computer Science*, 2:e93.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Danilo S Carvalho, Minh-Tien Nguyen, Tran Xuan Chien, and Minh Le Nguyen. 2016. Lexical-morphological modeling for legal text analysis. *arXiv preprint arXiv:1609.00799*.

Yen-Liang Chen, Yi-Hung Liu, and Wu-Liang Ho. 2013. A text mining approach to assist the general public in the retrieval of legal documents. *Journal of the American Society for Information Science and Technology*, 64(2):280–290.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*, pages 1724–1734.

Thorsten Joachims. 2002. *Learning to classify text using support vector machines: Methods, theory and algorithms*. Kluwer Academic Publishers.

Daniel Martin Katz, II Bommarito, J Michael, and Josh Blackman. 2016. A general approach for predicting the behavior of the supreme court of the united states. *arXiv preprint arXiv:1612.03473*.

Mi-Young Kim, Randy Goebe, and Ken Satoh. 2014a. COLIEE-14. `http://webdocs.cs.ualberta.ca/~miyoung2/jurisin_task/index.html`.

Mi-Young Kim, Ying Xu, and Randy Goebel. 2014b. Legal question answering using ranking svm and syntactic/semantic similarity. In *JSAI International Symposium on Artificial Intelligence*, pages 244–258. Springer.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings EMNLP*, pages 1746–1751.

Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016. Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence. In *Proceedings of NAACL-HLT*, pages 521–526.

Wan-Chen Lin, Tsung-Ting Kuo, and Tung-Jia Chang. 2012. Exploiting machine learning models for chinese legal documents labeling, case classification, and sentencing prediction. *ROCLING XXIV (2012)*, page 140.

Chao-Lin Liu, Cheng-Tsung Chang, and Jim-How Ho. 2004. Case instance generation and refinement for case-based criminal summary judgments in chinese. *Journal of Information Science and Engineering*, 20(4):783–800.

Chao-Lin Liu and Chwen-Dar Hsieh. 2006. Exploring phrase-based classification of judicial documents for criminal charges in chinese. In *International Symposium on Methodologies for Intelligent Systems*, pages 681–690. Springer.

Chao-Lin Liu and Ting-Ming Liao. 2005. Classifying criminal charges in chinese for web-based legal services. In *Asia-Pacific Web Conference*, pages 64–75. Springer.

Yi-Hung Liu, Yen-Liang Chen, and Wu-Liang Ho. 2015. Predicting associated statutes for legal problems. *Information Processing & Management*, 51(1):194–211.

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

Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. 2014. Large-scale multi-label text classification—revisiting neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 437–452. Springer.

K Raghav, P Krishna Reddy, and V Balakista Reddy. 2016. Analyzing the extraction of relevant legal judgments using paragraph-level and citation information. *AI4J–Artificial Intelligence for Justice*, page 30.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Quantifying the Effects of Text Duplication on Semantic Models

Alexandra Schofield[1] **Laure Thompson**[1] **David Mimno**[2]
1 Department of Computer Science, Cornell University, Ithaca, NY
{xanda, laurejt}@cs.cornell.edu
2 Department of Information Science, Cornell University, Ithaca, NY
mimno@cornell.edu

Abstract

Duplicate documents are a pervasive problem in text datasets and can have a strong effect on unsupervised models. Methods to remove duplicate texts are typically heuristic or very expensive, so it is vital to know when and why they are needed. We measure the sensitivity of two latent semantic methods to the presence of different levels of document repetition. By artificially creating different forms of duplicate text we confirm several hypotheses about how repeated text impacts models. While a small amount of duplication is tolerable, substantial over-representation of subsets of the text may overwhelm meaningful topical patterns.

1 Introduction

Different discussions of the same subject tend to use similar words. Unsupervised models such as latent semantic analysis (LSA) (Deerwester et al., 1990) and latent Dirichlet allocation (LDA) (Blei et al., 2003) look for these statistical signatures of topicality in the form of repeated word co-occurrences. These methods have become increasingly popular because they are powerful and easy to apply to large unlabeled datasets. The apparent ease-of-use of LSA and LDA, however, makes it easy to overlook potential problems in text corpora. In this work, we focus on measuring the impact of one such issue: duplicate text.

Latent semantic methods look for patterns of repetition. But when text is repeated exactly, statistical methods that look for patterns may be diverted from more meaningful semantic groups: verbatim repetition looks, to the algorithm, more topical than actual topics. If not accounted for, repeated text can change measures of fitness to overvalue fit on repeated texts, or even "leak" held out data that is duplicated in the training data. At best, duplication may cause us to overestimate the expressiveness and reliability of models. At worst, models skewed by text duplication may invalidate any conclusions drawn from them, and, by extension, the method itself.

Text replication is a persistent and difficult problem in natural language corpora. In social media settings, partial duplication due to quotation and threading is ubiquitous. Of the 20k posts in the 20 Newsgroups corpus (Lang, 1995), 1151 are exact duplicates, and 25% of the remaining tokens are quoted text from other newsgroup messages.[1] In literary corpora, different versions of the same document may also conflict: text files for Hamlet may differ slightly due to publisher information, line numbers, editorial changes between Shakespeare's folios, and footnotes. Removing exactly identical duplicates of texts is possible through direct lexicographic matching, but for lexical near-duplicates and partial textual overlap, we may need more careful heuristics to detect duplicates, forcing researchers to make judgments about what text to remove and what to keep.

Evaluating what level of duplication is "safe" can therefore not only reduce the risk of false conclusions but also save great amounts of work spent identifying and removing duplication. In this work, we investigate the effect of text duplication on LSA and LDA by experimentally amplifying the magnitude of text duplication in a variety of corpora. We look both at how models shift to over-represent repeated text and how that shift affects the model representation of documents without repetition. To account for the variety of types of duplication, we look at exact du-

[1] Computed using scikit-learn's 20 Newsgroups API: http://scikit-learn.org/stable/datasets/twenty_newsgroups.html

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2737–2747
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

plication of whole documents as well as repetition of a text segment across many documents. Finally, we recommend what aspects of text deduplication one should focus on to successfully reduce negative effects, with different suggestions depending on the chosen model. [2]

2 Previous Work

Text duplication and reuse is a well-established problem in textual corpora. The web is filled with pages of near-duplicate content (Broder et al., 1997; Manku et al., 2007), journalistic reuse is common practice with the dissemination of information from news agencies to newspapers (Clough et al., 2002; Smith et al., 2013), and plagiarism is prevalent in student submissions (Clough et al., 2003). However, past work has focused on the *identification* of reuse instead of the *effects* that duplication has on semantic models.

The detection of text reuse relies on the ability to measure similarity between documents or passages. In general, these techniques measure the similarity of textual content, though other similarity metrics for reuse identification have been proposed (Bär et al., 2012). These measures can fall into two general groups: global and local. Global techniques measure the similarities of entire texts. These techniques are especially used for near-duplicate detection. A common approach of this form is fingerprinting (Potthast and Stein, 2008). This method involves transforming a document into a smaller representation (e.g. a set of n-grams) to measure similarity cheaply. Local techniques measure similarity at a finer granularity (e.g. paragraphs or sentences). In this setting, reuse may be mixed with text derived from other sources. These techniques often involve two steps: one aligning texts with some method (Lee, 2007; Smith et al., 2013)m and one scoring similarity of aligned sequences, e.g. based on cosine similarity of the bag-of-words vector. All of these techniques require choices of hyperparameters such as similarity threshold and n-gram size that affect what the technique considers duplicate text. Our work focuses on understanding what types of document deduplication are important so that practitioners can make better-informed choices about how to calibrate these models.

While it is possible to evaluate semantic models as features in a downstream supervised task, they are harder to evaluate intrinsically as unsupervised models of data exploration. For LDA, it is standard to consider held-out likelihood of a test set as a measure of model fit (Wallach et al., 2009b). One can also use human evaluations to judge the interpretability of topic summaries (Chang et al., 2009), though this measure can also be approximated with automated evaluations based on corpus statistics (Aletras and Stevenson, 2013; Lau et al., 2014; Mimno et al., 2011). One can also evaluate individual topics based on how much they diverge from corpus-wide distributional expectations (Al-Sumait et al., 2009).

Because LSA does not yield semantically meaningful dimensions, intrinsic approaches to evaluation are focused on the spatial aspects of the model's word embedding into the real domain. Word similarity tasks are perhaps the most common evaluation, which compare human "gold standard" judgments of word pair similarity to distances between the corresponding word vectors (Finkelstein et al., 2001; Bruni et al., 2012; Hill et al., 2016). However, the vagueness of definitions of "similarity" and the contextual dependency of similarity have cast doubt on these as gold standards of evaluation (Faruqui et al., 2016). Solving word analogies using vector arithmetic is also sometimes used to evaluate neural word embeddings, but LSA does not tend to produce this structure well (Pennington et al., 2014; Mikolov et al., 2013).

3 Theorized Impact

The fundamental problem with repeated text in a distributional semantic model is the over-representation of specific word co-occurrences to a model. To understand this, we consider the matrix factorization representation of these models. Borrowing notation from Arora et al. (2013), we consider a corpus with M documents and vocabulary size V over which we want to learn a K-dimensional representation of each document and vocabulary term. We can build an $M \times V$ matrix C to represent our corpus, where C_{di} is a function of the frequency of term i in document d. Both LSA and LDA represent factorizations of this matrix into two rank-K matrices, $C = WA^T$, where W is an $M \times K$ matrix and A is a $V \times K$ matrix. In the case of LSA, we apply tf-idf weighting

[2] Code for our experiments can be found at https://github.com/heraldicsandfox/semantic-text-duplication.

to C before producing a truncated singular value decomposition $C = U\Sigma A^T$, with Σ a diagonal matrix of dimension $K \times K$, and U and A column-orthogonal matrices. We can reduce this to the factorization above by multiplying Σ with one of the two outer matrix factors, e.g. $W = U\Sigma$. LDA performs a non-negative matrix factorization on a smoothed stochastic version of C, producing row-stochastic matrices W and A.

Duplicate text implies that more rows in C will contain a particular signature of word frequencies. This implies that a low-rank matrix factorization will increasingly devote representative power to this particular textual signature in order to minimize loss in its representation. We expect to observe two principal effects:

- As text is repeated more, to optimize model fit on the data, one or more topics/dimensions will converge to model the repeated text.
- Text that is not exactly or near-exactly repeated (or *singular* text) will be modeled less effectively both in terms of model fit and interpretability.

These effects are based on the incentive of the model to overfit repeated text: topics and dimensions modeling solely the repeated text will leave less representational power for the remaining text, and combinations of repeated and singular text will likely yield less coherent topics.

4 Evaluation Methods

We quantitatively examine several aspects of models with varying forms and degrees of duplication to determine the magnitude of the change produced by repeated text. It is important to note that our goal is simply to measure the difference between models, and not to make normative statements about the *quality* of topics. Indeed, many measures of topic quality such as word intrusion (Chang et al., 2009) and word co-occurrence (Newman et al., 2010; Mimno et al., 2011; Lau et al., 2014) may improve as a result of degenerate, single-document topics: most documents are internally coherent, so a single document's word distribution may appear to be a sensible topic.

Loss The first aspect is model loss. As stated in Section 3, as a segment of text is repeated more, we anticipate that the fit over documents containing repeated text will improve, while the fit over documents not containing repeated text will worsen. To evaluate this for LSA, we examine the Frobenius norm of the difference between the reconstruction WA^T and C for the rows corresponding to documents with and without repeated text. For LDA, we estimate the perplexity of both the training data and held-out data without repetitions from the same corpus.

Concentration Secondly, we examine component (e.g. topic/dimension) concentration. Repetition of a document amplifies the co-occurrence between the terms contained in the document. As this signal grows stronger, we expect models to begin "memorizing" these words. We anticipate that affected models will develop a simpler latent representation for the repeated document, one concentrated over a small number of components. For example, if a model is devoting topic k to a repeated document, then instances of that document should have a high proportion of topic k. Concentration measurements relate to loss, but focuses specifically on the document-component or document-topic patterns, while loss also includes information about the topic-word dynamics.

The effect of components converging to a single piece of repeated text should be easily observed by examining how close topics are to the unigram language model induced by the repeated text. If we repeat multiple documents independently, however, we may also expect to see distinct components correlated with disjoint subsets of the repeated texts. To account for this, we evaluate component concentration separately for documents with repeated text and without repeated text.

For LDA, we examine the *entropy* of document vectors. Information entropy represents the expectation of the representation length of a given outcome as a function of the probability distribution over outcomes:

$$E_d = \sum_k \theta_{dk} \log \theta_{dk}$$

where θ_{dk} is the probability of a token generated in document d having topic k. Entropy is inverse to concentration: the entropy of text should lower as the text is repeated more, as all of their topical mass would be concentrated in topics converging to modeling duplicate texts. Conversely, documents not containing repeated text may also have their entropy increase as text repetition increases, as topics will less adequately fit to the behavior of the singular documents.

In LSA, entropy is not as directly applicable: vectors in $W = U\Sigma$ can be arbitrarily real-numbered. However, we still want to access a similar basic concept, the amount a vector representation of a document is concentrated in a few dimensions. So, we examine the *absolute dispersion* of each row vector d in W:

$$D_d = \sum_k \frac{|d_k|}{\|d\|_1} \log \frac{|d_k|}{\|d\|_1}$$

where d_k is the kth component of d. Absolute dispersion measures the entropy of the L1-normalized masses of the vectors in W.

Expressivity The final aspect is expressivity of topics. If one topic converges to the unigram language model of repeated documents, the resulting model has effectively lost one topic worth of expressive power by focusing on overly-specific themes. Someone looking to learn generalized semantic corpus patterns from a topic model will therefore have one fewer topic of interest available. The frequency of terms in the repeated text may also overwhelm the most probable terms in many of the topics, again reducing the ability to interpret these topics or to understand their content through a summarized representation. While expressivity in the form of topic summaries makes little sense for LSA, using LDA models, we may examine topic summaries, obtained as the top s most probable terms of a topic where s is a fixed parameter. We may select the same number of terms s from the most probable in a unigram language model of the repeated text, and determine what proportion of the tokens obtained from concatenating topic similarities are the top terms of the repeated text language model.

5 Experimental Setup

Data We use two corpora: a sample of articles from the New York Times Annotated Corpus (Sandhaus, 2008) and a collection of Reuters newswires from the Spanish Language News Text Corpus (REUSL) (Graff, 1995). We choose news corpora because they provide well-curated text with repeated subjects but few exact document-level duplicates, though quotes and templated text may still cause text duplication. We can use these as a testbed for general duplication behaviors we see across a variety of corpora. Text is lower-cased and tokenized to only include tokens of three or more characters, allowing for contractions or hyphenations as single tokens. New York Times articles average 494.5 words in length, while Reuters newswires average 201.5 words.

To ensure our experiments are the only cause of exact duplication of text in our corpora, we use strict methods of text deduplication. When two or more documents have more than 70% unigram overlap, we remove all but the longest document. In addition, we delete 7-grams that appear in more than 10 documents based upon existing thresholds for plagiarism detection (Citron and Ginsparg, 2015). To account for stopwords, we remove all terms appearing in more than 80% of documents. Finally, we remove documents with fewer than 7 tokens after processing. We perform this process on a random sample of 30,000 documents from each corpus to ensure we may obtain a sample of 25,000 curated documents for each of our two corpora. We also produce 10% samples of these corpora, containing 2,500 documents each, to measure the effect of corpus size.

Text Duplication Treatments We use our deduplicated news corpora to construct datasets with artificial text duplication. We examine two different duplication scenarios: exact document duplication and template string duplication.

In *exact document duplication*, we randomly sample $p\%$ of the documents in the dataset and include c copies of each sampled document in our final corpus along with one copy each of the remaining documents, which we refer to as *singular* documents. To test the extremes of this effect, we also perform *single document* tests for large c with only one repeated document. From these synthetically duplicative corpora, we can determine whether effects are triggered by the sheer volume of duplicated text or if they are influenced by the diversity of the copied documents.

In *template string duplication*, rather than duplicating the sampled $p\%$ of documents, we prepend a fixed string to each document in the $p\%$ sample, producing what we refer to as *templated* documents or texts. As repeated text may be lexically similar or different from the non-repeated text of the corpus, we consider two different types of prepended string. The first is a randomly-sampled document from the deduplicated corpus but not included in the training set (*Sampled Template*), simulating repeated text that is lexically similar to the document content. The second is

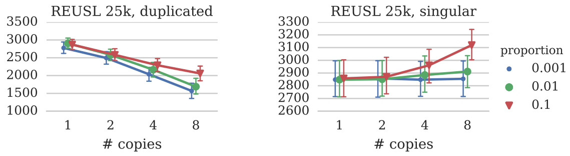


Figure 1: Training perplexity with LDA models trained on the REUSL 25k corpus with 80 topics. Perplexity decreases significantly for the duplicated documents with repetition, but the effect on singular documents is negligible with repeated proportion of the corpus smaller than 0.1.

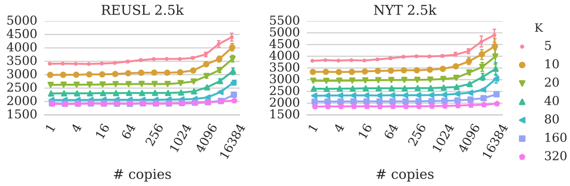


Figure 2: Perplexity from duplicating a single document remains largely unaffected for singular documents until the number of repetitions is $c = 4096$, when duplicate texts outnumber singular texts. There is also a subtle inflection point for smaller numbers of topics K at $c = 256$, approximately 1/10th of the corpus, but this effect is not visible with more topics.

the first 100 words of the classic Lorem Ipsum filler text (*Lorem Template*), simulating repeated text with little lexical overlap with the documents. Because we are investigating bag-of-words models, we do not worry about grammatical errors in the nearly-duplicated text, so the segmentation of this repeated prefix should not be a concern.

Training We analyze two types of semantic models: LSA and LDA. LSA models are trained using tf-idf weighting on word-document matrices using custom Python code.[3] LDA models are trained using Mallet (McCallum, 2002) with fixed hyperparameters $\alpha = 50/K$ and $\beta = 0.01$ for ease of comparison. To compute perplexity, we use log likelihood estimates from Mallet's built-in left-to-right estimation (Wallach et al., 2009a).

6 Results

Because of the exponential combination of different experimental settings available, it would be unfeasible to examine all our metrics for all data. Instead, we focus our analysis on specific examples that we believe demonstrate the effects seen in the rest of the corpus. We use smaller sets of 2.5k documents for examining the effect of heavy duplication and sets of 25k documents otherwise.

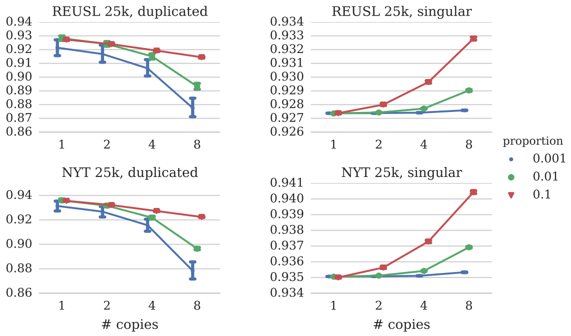


Figure 3: Model loss for LSA models with 80 components. Loss for duplicated documents decreases as the number of repetitions c increases. The frequency of replication affects loss at a much smaller scale for singular documents.

6.1 Loss

We begin with the case of exact document duplication. In Figure 1, the perplexity of LDA decreases substantially as documents are duplicated. This reduction is due to better fit to the duplicated documents. As fit improves in duplicated documents, however, we do not see a meaningfully worse fit for singular documents. These documents increase in perplexity, but the increase is not significant at low levels of duplication, such as when $c = 2$ or $p = 0.001$. In the single document case in Figure 2, this effect is emphasized: likelihood on singular documents remains level even with heavy repetition in short corpora. The sheer volume of duplicated text does not by itself damage model fit, likely because the duplicated text can be easily modeled by a single topic.

This effect is not solely due to LDA's specific probabilistic model. We see a similar pattern in LSA. In Figure 3, we see that loss for duplicated documents decreases as duplication increases. However, the amount of decay depends on the proportion of the corpus replicated: the smaller the proportion size, the more dramatic the decay. In contrast, the loss for singular documents increases only slightly with more copies, though more for higher proportions of duplication.

To gain a better understanding of how duplication affects LSA, we look at the effects of repeating a single document an extreme number of

[3] Code uses `scipy`, `numpy`, and `scikit-learn`.

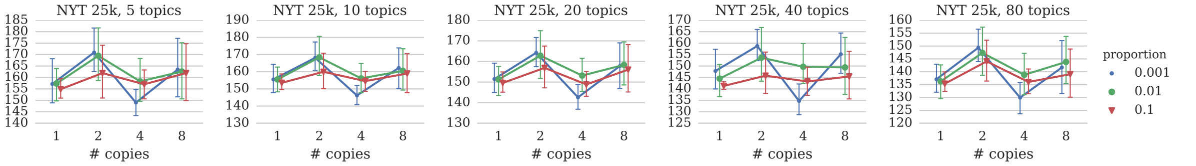


Figure 4: Held-out data perplexity (in thousands) for different the NYT 25k corpus with varying numbers of topics K. Increasing the proportion of repetition for exact duplicate documents does not increase test perplexity. With repeated corpus proportion $p = 0.001$, however, repeating documents exactly 4 times (but not 2 or 8 times) significantly improves perplexity, potentially because it induces a new topic to model it. Held-out data contained no repeated documents.

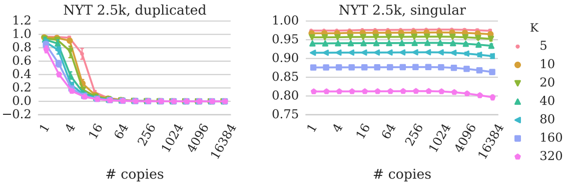


Figure 5: LSA model loss for the NYT 2.5k corpus with one duplicated document. Model loss for singular documents is again unaffected by repetition, while loss for the duplicated document quickly falls to zero as repetition increases. The fall in loss signals the start of model "memorization."

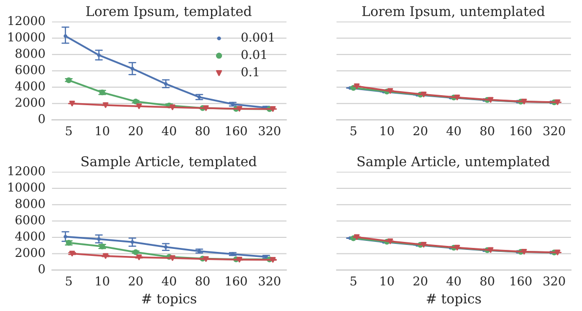


Figure 6: LDA training perplexity for REUSL 2.5k with different types of templated text repetition. The effect of duplication is prominent for small numbers of topics but diminishes with more topics to sufficiently model the missing text. With the fraction of the corpus that contains duplicates $p = 0.1$, the perplexity of template documents is below that of untemplated texts.

times. From Figure 5, we see that the loss of singular documents does not meaningfully change as the number of copies increases. We can observe the steep decline in loss for duplicated documents as the signal of when the top K components begin to "memorize" the duplicated document. The more components K, the fewer repetitions need for the overfitting to begin.

In the LDA case, we may also look at held out perplexity. Figure 4 shows that the fit for held-out test data is not generally significantly affected by increased repetition. There is a pattern within the data, in which repeating documents 4 times seems to produce better perplexity for singular documents than 2 or 8, significantly so for a small fraction of the corpus. A theory for this is that at a sufficient level of repetition, LDA fits the repeated text to its own topic instead of trying to conflate it with other document contents, producing better topics. However, additional repetition further saturates these topics and adds noise to the meaningful co-occurrence signal.

Figure 6 demonstrates that, as before, perplexity is significantly higher as template repetition increases when there is a small number of topics $K = 5$. However, as the number of topics increases, this disparity ceases to be significant. Interestingly, however, with high enough proportion p of documents containing templates, the perplexity drops below that of documents not containing the duplicates at all numbers of topics.

For LSA, templated repetition has no apparent effect on the loss of untemplated texts. However, the effect for templated texts is less straightforward. Figure 7 shows that for proportions $p = 0.1$ and $p = 0.01$ the loss of templated texts is smaller than for untemplated texts for all K component sizes. For proportion $p = 0.001$, though, templated loss is only smaller than untemplated loss when K is large, while templated loss is never significantly lower than untemplated loss for $p = 0.0001$. *Lorem Template* and *Sample Template* also exhibit different behaviors templated texts when $p = 0.001$ and K is large: loss is is significantly smaller for *Lorem Template* and has a larger drop in loss from $K = 80$ to $K = 160$.

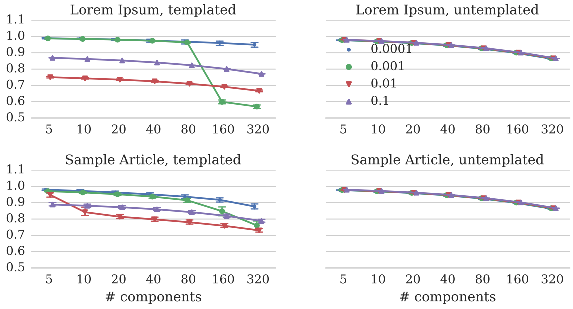


Figure 7: LSA loss of templated text for the REUSL 25k corpus. Higher levels of templating p result in smaller model loss for templated texts than for untemplated texts. For $p = 0.001$, templated loss becomes smaller than untemplated loss for $K = 160$ but more dramatically for the *Lorem Template*.

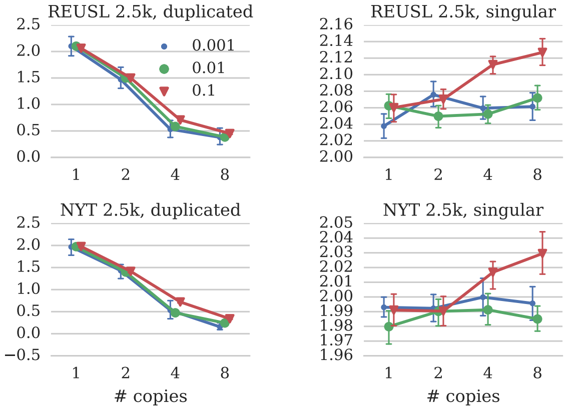


Figure 8: Entropy for LDA with 80 topics decreases for duplicated documents as the frequency of those documents increases, has little initial effect on the entropy of the singular documents.

This may indicate that LSA is able to more effectively model templated text when the templates have a distinctively different language model than the original documents.

6.2 Concentration

We expect the effect of duplication on entropy will be inversely correlated with its effect on model loss. As we increase the proportion of the corpus that is repeated, the model will devote more resources to duplicate text, leaving less modeling power for the remaining text. We therefore expect dispersion to increase with p for duplicate documents and decrease with p for singular documents. In Figure 8, the first effect clearly holds for LDA, but the second does not: there is a negligible change in entropy with the number of repetitions of documents. Figure 9 shows a subtler version of the same effect for LSA. Notably, the decrease in absolute dispersion for repeated documents is only visible in the short corpora.

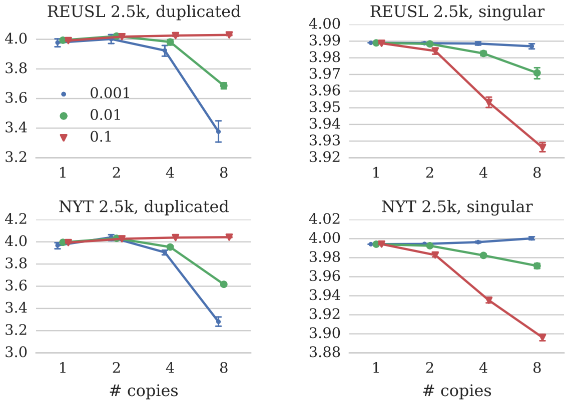


Figure 9: Absolute dispersion for LSA with 80 components increases slightly when first producing duplicates ($c = 2$), but falls off for smaller proportions of repetition $p = 0.01$ and $p = 0.001$ at higher frequencies. Increasing c has a comparatively small effect on the absolute dispersion of singular documents.

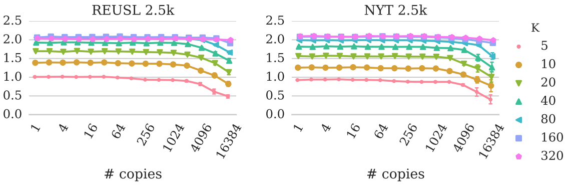


Figure 10: When a single document in the short corpora is repeated enough to comprise the majority of the corpus, the LDA entropy decreases over singular documents.

We can examine the extreme effects of the change in component concentration for singular documents by looking at its behavior in the *single document* treatment. In Figure 10, we see that while entropy remains level for repetitions comprising smaller portions of the corpus, eventually the entropy drops for both repeated and singular documents. This may be because most topics describe the repeated document, leaving few to model the remaining singular documents.

For LSA, absolute dispersion remains level for all repetitions tested for the *single document* treatment. This result highlights a key difference between LDA topics and LSA components: while changing the number of topics in LDA influences

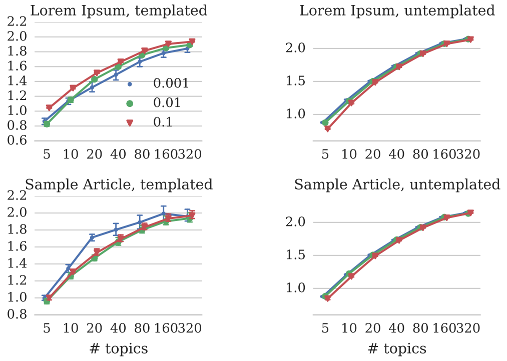


Figure 11: LDA entropy for the REUSL 25k corpus with *Sample Template* and *Lorem Template* treatments. With few topics, templated documents have lower entropy than untemplated documents, but with many topics, their entropy is higher. In the mid range of topics for *Lorem Template*, higher proportions of sampled text p produce higher entropy, but for *Sample Template*, lower p produces higher entropy.

the prior to raise or lower entropy over topics, the components of LSA are fixed. Increasing the number of components increases dimensionality, but never alters preexisting dimensions.

The effect is more subtle when templated text is repeated within documents. Figure 11 shows that with $K = 20$ LDA topics, if we apply the *Sample Template* to a small fraction of documents ($p = 0.001$), it produces a higher entropy than corpora with larger template inclusion proportion p. This is not surprising: though the template text and the original document are similar in style, with high probability they will still have different topics, which the model will have trouble fitting well without more observations. The *Lorem Template* has the reverse effect: the language is sufficiently disjoint from the content of the documents that few topics or even a single topic can model the repeated text fully, leading to low entropy. When the language model of duplicated text is disjoint from that of the text of interest, the template can be modeled by one or a few topics or components without significantly affecting other text.

6.3 Expressivity

Quantitative analyses of model fit and topic uncertainty are helpful in analyzing the effect of different settings, but do not necessarily tell us whether topics from corpora with repeated documents are useful. Analysis of expressivity helps us fill in some of the gaps in our explanations above as to what is happening at the individual topic level. In Figure 12, we see that for a moderate number of topics, increased repetition of documents impacts a substantial portion of the top-ranked words, or most probable terms of topics. The saturation effect has some relation to the number of repeated documents. With a single document repeated, as in Figure 13, as the number of topics increases, the ratio of top-ranked words belonging to the unigram language model drops. We also notice that with few topics, there is a clear "saturation point" where the topic begins to be represented more, which remains level until half the short corpora are represented by the duplicate document. The pattern overwhelmingly shows that single texts are easily fit by single topics.

In the case of the *Lorem Template* input, where little textual overlap exists between the template and original text, a few topics quickly fill in the repeated text, producing a limited effect on most topics. In Table 1, the number of topics containing "lorem" and "ipsum" remains small as the number of topics grows. Regardless of topic count of proportion, topics containing "lorem ipsum" are entirely broken Latin: the top probable terms of an example 320-topic model with $p = 0.1$ are *est justo donec iaculis sit ipsum quam lorem tristique sed amet eget pharetra curabitur fringilla non consequat mattis nec nascetur*, a direct sample of words from the template text.

7 Conclusion

The presence of duplicated strings, either documents or duplicated text within documents, is a serious but not insurmountable problem. Duplicate text can substantially alter the dimensions learned by distributional semantic models. The effect of duplication depends on several factors: the number of distinct repeated strings, the similarity of repeated strings to the rest of the corpus, and the

Proportion	5	10	20	40	80	160	320
0.001	0	0	0	0	0	0.2	0.8
0.01	0	0	0.2	0.56	1	1	1
0.1	1	1	1	1	1.78	2.3	3

Table 1: As the number of total topics increases, the average number of topics fitting the *Lorem Template* duplicate text remains stable, only rising above 1 with repeated proportion of the corpus $p = 0.1$ and at least 80 topics.

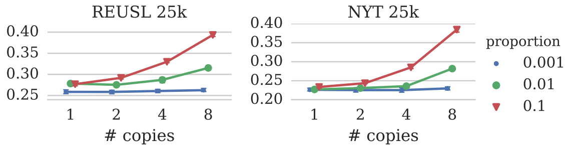


Figure 12: With 80-topic LDA models of our larger datasets, we see that increased repetition leads to significant increases in the amount of representation of repeated text in the top keys of topics.

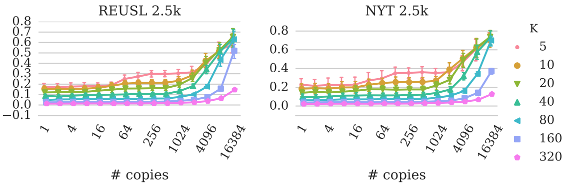


Figure 13: Top keys of LDA topics for only a single repeated document remain concentrated in only a few topics in models with $K > 5$, negligibly impacting the top keys of remaining topics.

number of repetitions. We find that different algorithms are affected in different ways, but that there are methods to alleviate the effect of duplication without exhaustively removing all duplicated documents. We provide the following specific conclusions and recommendations:

LDA accommodates low rates of document duplication for many documents. We find that with more frequent repetition, the algorithm is able to sequester repeated text into small numbers of topics if certain conditions hold. To handle this case, the model must have many topics available relative to the number of repeated strings, and the language of the repeated text must be sufficiently distinct. If these conditions are met, repeated text will affect a small number of topics that can be identified by their similarity to specific documents, or automatically based on lower than expected inter-document variability within a topic (Mimno and Blei, 2011) or distance from specific corpus-word or document-word distributions (AlSumait et al., 2009). We therefore suggest training a model first with slightly more topics than desired, then evaluating if there are any signs of repeated texts overwhelming several topics due to low coherence or corpus statistics. If no such indications occur, or if the duplication remains in one or two topics, then there is no need to modify the corpus or retrain the model, as the duplicate-capturing topic may be ignored.

LSA permits high rates of document duplication so long as few unique texts are repeated. Repeating one document will likely only affect one or a few components regardless of how many repetitions occur. However, if there are many different repeated documents, more components will be used to model them, which worsens the model fit more as the number of unique repeated texts increases. In this case, it may be preferable to look for near-duplicate documents more aggressively and worry less about exact duplicates. Unigram-count-based deduplication may be appropriate in this case, using a simple threshold of cosine similarity between the vectors of unigram counts between two documents to deduplicate.

Repeated text templates for LSA and LDA are sequestered by the model so long as they do not overlap heavily with topics of interest. In a topic model, it may be easy to identify the templated text based upon it appearing in one topic. However, if there is a concern that there is systematic use of text templates in documents (such as page headers or publication information) that may be too close to the language model, the n-gram removal approach inspired by Citron and Ginsparg (2015) is an expensive but straightforward way to ensure these strings are detected and deleted. The combination of unigram deduplication, n-gram deletion, and the inherent ability of semantic models to separate co-occurring text should reduce the negative effects of text duplication.

Acknowledgments

We would like to thank our reviewers for their helpful feedback and suggestions. We would also like to thank Jack Hessel, Måns Magnusson, and Ana Smith for their reviews and feedback. This material is based on work supported by the DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a; National Science Foundation grants #1526155, #1652536 (CAREER), and #DGE-1144153 (GRFP); and a faculty research fellowship from the Alfred P. Sloan Foundation.

References

Nikolaos Aletras and Mark Stevenson. 2013. Evaluating topic coherence using distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics*, pages 13–22.

Loulwah AlSumait, Daniel Barbará, James Gentle, and Carlotta Domeniconi. 2009. Topic significance ranking of LDA generative models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 67–82. Springer.

Sanjeev Arora, Rong Ge, Yonatan Halpern, David M Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. 2013. A practical algorithm for topic modeling with provable guarantees. In *Proceedings of the 30th International Conference on Machine Learning*, pages 280–288.

Daniel Bär, Torsten Zesch, and Irnya Gurevych. 2012. Text reuse detection using a composition of text similarity measures. In *Proceedings of the 24th International Conference on Computational Linguistics*, volume 1, pages 167–184.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Andrei Z Broder, Steven C Glassman, Mark S Manasse, and Geoffrey Zweig. 1997. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8-13):1157–1166.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 136–145. Association for Computational Linguistics.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems*, pages 288–296.

Daniel T Citron and Paul Ginsparg. 2015. Patterns of text reuse in a scientific corpus. *Proceedings of the National Academy of Sciences*, 112(1):25–30.

Paul Clough, Robert Gaizauskas, Scott SL Piao, and Yorick Wilks. 2002. Meter: Measuring text reuse. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 152–159.

Paul Clough et al. 2003. Old and new challenges in automatic plagiarism detection. In *National Plagiarism Advisory Service*. Http://ir.shef.ac.uk/cloughie/index.html.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, page 30.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on the World Wide Web*, pages 406–414. ACM.

Gustavo Graff, David amd Gallegos. 1995. Spanish news text. *Linguistic Data Consortium*, DVD: LDC95T9.

Felix Hill, Roi Reichart, and Anna Korhonen. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 530–539.

John Lee. 2007. A computational model of text reuse in ancient literary texts. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 472–479.

Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. 2007. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on the World Wide Web*, pages 141–150. ACM.

Andrew K McCallum. 2002. MALLET: a machine learning for language toolkit. Available at: http://mallet.cs.umass.edu.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

David Mimno and David Blei. 2011. Bayesian checking for topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 227–237. Association for Computational Linguistics.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In

Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 262–272. Association for Computational Linguistics.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Proceedings of the Meeting of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processin*, volume 14, pages 1532–1543.

Martin Potthast and Benno Stein. 2008. New issues in near-duplicate detection. In *Data Analysis, Machine Learning and Applications*, pages 601–609. Springer.

Evan Sandhaus. 2008. The New York Times annotated corpus. *Linguistic Data Consortium*, DVD: LDC2009T19.

David A Smith, Ryan Cordell, and Elizabeth Maddock Dillon. 2013. Infectious texts: Modeling text reuse in nineteenth-century newspapers. In *Proceedings of the IEEE International Conference on Big Data*, pages 86–94.

Hanna M Wallach, David Mimno, and Andrew K McCallum. 2009a. Rethinking LDA: Why priors matter. In *Advances in Neural Information Processing Systems*, pages 1973–1981.

Hanna M. Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009b. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1105–1112. ACM.

Identifying Semantically Deviating Outlier Documents*

Honglei Zhuang[1], **Chi Wang**[2], **Fangbo Tao**[1], **Lance Kaplan**[3] and **Jiawei Han**[1]
[1]Department of Computer Science, University of Illinois at Urbana-Champaign
[2]Microsoft Research, Redmond [3]US Army Research Lab
{hzhuang3, ftao2, hanj}@illinois.edu,
wang.chi@microsoft.com, lance.m.kaplan.civ@mail.mil

Abstract

A document outlier is a document that substantially deviates in semantics from the majority ones in a corpus. Automatic identification of document outliers can be valuable in many applications, such as screening health records for medical mistakes. In this paper, we study the problem of mining semantically deviating document outliers in a given corpus. We develop a generative model to identify frequent and characteristic semantic regions in the word embedding space to represent the given corpus, and a robust outlierness measure which is resistant to noisy content in documents. Experiments conducted on two real-world textual data sets show that our method can achieve an up to 135% improvement over baselines in terms of recall at top-1% of the outlier ranking.

1 Introduction

The technology today has made it unprecedentedly easy to collect and store documents in an increasing number of domains. Automatic text analysis (*e.g.* document clustering, summarization, topic modeling) becomes more useful and demanded as the corpus size grows. Some trending domains (*e.g.* health records) call for a new analytical task, *mining outlier documents*: given a corpus, identify a small number of documents which substantially deviate from the semantic focuses of the given corpus. Outlier documents can provide valuable insights or imply potential errors. For example, an outlier health record from records of the same disease could indicate a new variation of the disease if it has an abnormal symptom description, or a medical error if it has an abnormal treatment description. A previous study (Hauskrecht et al., 2013) uses structured data in health records to show the importance of this application, and points out that further improvement should be achieved by leveraging text data.

Existing work has studied a related albeit different task, novel document detection (Kasiviswanathan et al., 2012, 2013; Zhang et al., 2002, 2004), where one aims to identify from a document stream if a newly arriving document is novel or redundant. In other words, this task assumes all the previous documents are known to be "normal", and only checks if a new document is novel. In our task, no document is known to be normal, and there could be multiple outliers in the corpus. Outlier detection (Chandola et al., 2009; Hodge and Austin, 2004) is a popular topic in data mining but few focus on text data. A study (Guthrie, 2008) identifies anomalous text segments in a document, but mainly based on writing styles. We focus on studying semantically deviating documents.

The problem of detecting outlier documents has its unique challenges. First, different words or phrases may be used to indicate the same semantic meaning, which introduces lexical sparsity. Second, finding proper words or phrases to characterize the corpus is non-trivial. Semantically frequent words or phrases can still be too general or too vague. Third, a document can carry extremely

* Research was sponsored in part by the U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), National Science Foundation IIS-1320617, IIS 16-18481, and NSF IIS 17-04532, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov). The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies of the U.S. Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2748–2757
Copenhagen, Denmark, September 7–11, 2017.

rich and noisy signals, most of which are not helpful to determine whether it is an outlier.

We tackle the problem of mining outlier documents in the following steps. We leverage word embedding (Mikolov et al., 2013) to capture the semantic proximities between words and/or phrases, in order to solve the sparsity issue. Then we propose a generative model to identify semantic regions in the embedded space frequently mentioned by documents in the corpus. The model represents each semantic region with a von Mises-Fisher distribution. We also learn a concentration parameter for each region with our model, and develop a selection method to identify semantically specific regions which can better represent the corpus, and filter regions with largely uninformative words.

As the final step, we design a robust outlierness measure emphasizing only the words or phrases in a document relatively close to the semantic focuses identified, and eliminating the noises and redundant information.

The remaining of the paper is organized as follows. Section 2 introduces the preprocessing of data sets and clarifies the notations. Section 3 proposes the methodology to mine outlier documents. Section 4 describes the experiment setup, Section 5 presents the results and Section 6 concludes the paper.

2 Preliminaries

In this section, we formalize the problem and then briefly describe the preprocessing step.

2.1 Notations

The notations used in this study are introduced here. A *document* is represented as a sequence $d_i = (w_{i1}, w_{i2}, \cdots, w_{in_i})$, where each $w_{ij} \in \mathcal{V}$ represents a word or phrase from a given vocabulary $\mathcal{V}$ and n_i denotes the length of the d_i. We refer to a set of documents as a *corpus*, represented as $D = \{d_i\}_{i=1}^{|D|}$.

Notice that w_{ij} may refer to a unigram word or a multi-gram phrase. Although it is nontrivial to appropriately segment a document into a mixed sequence of words and phrases, it is not the focus of our paper. A recently developed phrase mining technique (Liu et al., 2015) is used to extract quality phrases and segment the documents.

Word embedding provides vectorized representations of words and phrases to capture their semantic proximity. We assume there is an effective word embedding technique (*e.g.* (Mikolov et al., 2013)), $f : \mathcal{V} \mapsto \mathbb{R}^{\nu}$, where f is the transforming function that takes a word or a phrase as input and projects it into a ν-dimensional vector as its distributed representation. The semantic proximity between two words or phrases w and w' can be preserved by the cosine similarity between their embedded vectors:

$$CosSim(f(w), f(w')) = \frac{f(w) \cdot f(w')}{\|f(w)\| \times \|f(w')\|}$$

Problem definition. This work studies how to effectively rank documents in a corpus based on how much they deviate from the semantic focuses of the corpus. Given a set of documents D, our objective is to design an outlierness measure $\Omega : D \mapsto \mathbb{R}$, such that documents with larger outlierness $\Omega(d)$ semantically deviate more from the majority of D.

2.2 Preprocessing

We perform several steps of preprocessing to derive the input representation of each document in a given corpus.

Phrase mining. SegPhrase, a recently developed phrase-mining method (Liu et al., 2015), is utilized to automatically identify quality phrases in a corpus. After being trained in one corpus, SegPhrase is also capable of segmenting unseen documents into chunks of phrases with mixed lengths. We train SegPhrase on an external corpus D_e to obtain the list of quality phrases. Then for each corpus D given for outlier detection, we employ the trained SegPhrase to chunk each document into a sequence of words and quality phrases.

Word embedding. We adopt word embedding as a preprocessing step to capture the semantic proximity between words/phrases. Instead of using the raw text, similar to (Liu et al., 2015), we use the sequence derived from SegPhrase as input to the word embedding algorithm. In particular, *word2vec* (Mikolov et al., 2013) is utilized in our experiments, but can be seamlessly replaced by any other embedding results.

We run the embedding algorithm based on the external corpus D_e, the same corpus used in phrase mining. As D_e is sufficiently large, there are only few words or phrases in D which never appear in D_e, and are simply discarded in the experiments.

Stop words removal. We remove stop words, as well as the words or phrases ranked high within a certain quantile in terms of document frequency[1] (DF) in the external corpus D_e. Such words or phrases usually carry background noise, and obstruct outlier detection.

3 Mining Outlier Documents

Our framework consists of the following steps. First, we leverage a generative model to identify semantic "regions" in the word embedding space frequently mentioned by documents in the given corpus. Second, we develop a selection method to further remove semantics regions that are too general to properly characterize the given corpus, and only keep regions both frequent and semantically specific, denoted as "semantic focuses". Finally, we calculate the outlierness measure for each document based on the mined semantic focuses. We design a robust outlierness measure which is less sensitive to noisy words or phrases in documents.

3.1 Embedded von Mises-Fisher Allocation

We start with a generative model to identify the frequent semantic regions in the word embedding space.

Since we use cosine similarity to capture the semantic proximities between two words or phrases, the magnitude of the embedding vector of each word can be omitted in this part. We use $\mathbf{x}_{ij} = f(w_{ij})/\|f(w_{ij})\|$ to represent the unit vector with the same direction as the embedded vector of w_{ij}, and use $\mathbf{X}$ to represent the collection of all $\mathbf{x}_{ij}$ where $1 \leq i \leq |D|$ and $1 \leq j \leq n_i$.

In order to characterize a semantic region in the embedded space, we introduce von Mises-Fisher (vMF) distribution. The von Mises-Fisher (vMF) distribution is prevalently adopted in directional statistics, which studies the distribution of normalized vectors on a spherical space. The probability density function of the vMF distribution is explicitly instantiated by the cosine similarity. It is an ideal distribution for our task because we use cosine similarity to measure the semantic proximity. Moreover, as we will see later, it empowers us to characterize how specific each semantic region is, which is helpful in further identification of semantic focuses for outlier detection.

We first introduce the formalization of the von Mises-Fisher distribution.

Von Mises-Fisher (vMF) distribution. A ν-dimensional unit random vector $\mathbf{x}$ (*i.e.* $\mathbf{x} \in \mathbb{R}^\nu$ and $\|\mathbf{x}\| = 1$) follows a von Mises-Fisher distribution $\text{vMF}(\cdot|\boldsymbol{\mu}, \kappa)$ if the probability density function follows:

$$p(\mathbf{x}) = C_\nu(\kappa) \exp\big(\kappa \boldsymbol{\mu}^\top \mathbf{x}\big)$$

where $C_\nu(\kappa) = \kappa^{\nu/2-1} \big/ (2\pi)^{\nu/2} I_{\nu/2-1}(\kappa)$; and $I_{\nu/2-1}(\cdot)$ is the modified Bessel function of the first kind; $(\nu/2 - 1)$ is the order.

The two parameters in the vMF distribution are the mean direction $\boldsymbol{\mu}$ and the concentration parameter κ respectively, where $\boldsymbol{\mu} \in \mathbb{R}^\nu$, $\|\boldsymbol{\mu}\| = 1$ and $\kappa > 0$. The distribution concentrated around the mean direction $\boldsymbol{\mu}$, and is more concentrated if the concentration parameter κ is larger.

Embedded von Mises-Fisher allocation. We propose a generative model by regarding each document as a bag of normalized embedded vectors, analogous to the bag-of-word representation of documents utilized in typical topic model (*e.g.*, LDA (Blei et al., 2001)). The major difference is that the data to be generated is now a bag-of-normalized-embedded-vectors for each document, and should be generated from a mixed vMF distribution instead of a mixed multinomial distribution.

A formalized description of the model is summarized as follows:

$$\begin{aligned}
\boldsymbol{\mu}_t &\sim \text{vMF}(\cdot|\boldsymbol{\mu}_0, C_0), && t = 1, 2, \cdots, T \\
\kappa_t &\sim \text{logNormal}(\cdot|m_0, \sigma_0^2), && t = 1, 2, \cdots, T \\
\boldsymbol{\pi}_i &\sim \text{Dirichlet}(\cdot|\boldsymbol{\alpha}), && i = 1, 2, \cdots, |D| \\
z_{ij} &\sim \text{Categorical}(\cdot|\pi_i), && j = 1, 2, \cdots, |d_i| \\
\mathbf{x}_{ij} &\sim \text{vMF}(\cdot|\boldsymbol{\mu}_{z_{ij}}, \kappa_{z_{ij}}), && j = 1, 2, \cdots, |d_i|
\end{aligned}$$

where $T > 0$ is an integer indicating the number of semantic regions, namely the number of vMF distributions in our mixture model.

We regularize the vMF parameters by the following prior distributions. We assume the mean direction $\boldsymbol{\mu}_t$ of each vMF distribution is generated from a prior vMF distribution $\text{vMF}(\cdot|\boldsymbol{\mu}_0, C_0)$, while the concentration parameter κ_t is generated from a log-normal prior $\text{logNormal}(\cdot|m_0, \sigma_0^2)$. A similar design is also adopted in (Gopal and Yang, 2014).

Parameter inference. We infer the parameters by Gibbs sampling. Because both the von Mises-Fisher distribution and the Dirichlet distribution

[1] Document frequency of a word (or phrase) is defined as number of documents where this word or phrase appears.

have conjugate priors, we can integrate out parameters $\boldsymbol{\mu}_t$ and $\boldsymbol{\pi}_i$ and develop a collapsed Gibbs sampler of z_{ij}:

$$
\begin{aligned}
&P(z_{ij} = t|\mathbf{Z}^{-ij}, \mathbf{X}, \boldsymbol{\kappa}; \boldsymbol{\alpha}, m_0, \sigma_0^2, \boldsymbol{\mu}_0, C_0) \\
\propto &\frac{(n_{it}^{-ij} + 1 + \boldsymbol{\alpha}^{(t)}) C_\nu(\kappa_t) C_\nu\big(\|C_0\boldsymbol{\mu}_0 + \kappa_t \mathbf{x}_{\cdot t}^{-ij}\|\big)}{C_\nu\bigg(\bigg\|C_0\boldsymbol{\mu}_0 + \kappa_t\big(\mathbf{x}_{\cdot t}^{-ij} + \mathbf{x}_{ij}\big)\bigg\|\bigg)}
\end{aligned}
$$

where $n_{it}^{-ij} = \sum_{j'}^{|d_i|} \delta(z_{ij'} = t) - \delta(z_{ij} = t)$ is the number of words in the i-th document being assigned to the t-th von Mises-Fisher distribution without taking w_{ij} into account; $\mathbf{x}_{\cdot t}^{-ij} = \sum_{i'}^{|D|} \sum_{j'}^{|d_i|} \mathbf{x}_{i'j'} \delta(z_{i'j'} = t) - \delta(z_{ij} = t)$ is the sum of word vectors assigned to semantic region t without counting w_{ij}. Here $\delta(\cdot)$ is the indicator function.

We can also derive a collapsed Gibbs sampler for concentration parameters κ_t's:

$$
\begin{aligned}
&P(\kappa_t|\mathbf{Z}, \mathbf{X}, \boldsymbol{\kappa}^{-t}; \boldsymbol{\alpha}, m_0, \sigma_0^2, \boldsymbol{\mu}_0, C_0) \\
\propto &\frac{C_\nu^{n_{\cdot t}}(\kappa_t)}{C_\nu\big(\|C_0\boldsymbol{\mu}_0 + \kappa_t \mathbf{x}_{\cdot t}\|\big)} \text{logNormal}(\kappa_t|m_0, \sigma_0^2)
\end{aligned}
$$

where $n_{\cdot t}$ is the number of words in semantic region t.

While sampling z_{ij} is relatively trivial, sampling κ_t is not straightforward. Similar difficulty is also mentioned in (Gopal and Yang, 2014). We employ a Metropolis-Hasting algorithm with another log-normal distribution centered at the current κ_t value as the proposal distribution.

After obtaining a sample from the posterior distribution of z_{ij}'s and κ_t's, we can easily obtain the MAP estimate of mean directions $\boldsymbol{\mu}_t$'s and the mixing distribution of each documents $\boldsymbol{\pi}_i$:

$$
\hat{\boldsymbol{\mu}}_t = \frac{C_0\boldsymbol{\mu}_0 + \kappa_t \mathbf{x}_{\cdot t}}{\|C_0\boldsymbol{\mu}_0 + \kappa_t \mathbf{x}_{\cdot t}\|}, \quad \hat{\boldsymbol{\pi}}_i = \frac{n_{it} + \boldsymbol{\alpha}^{(t)}}{n_{i\cdot} + \sum_t \boldsymbol{\alpha}^{(t)}}
$$

Discussions. We notice that there are some topic models (Das et al., 2015; Batmanghelich et al., 2016) proposed for similar data, where words are represented as embedding vectors. Our model is proposed independently for the purpose of identifying semantic focuses, which serves the task of outlier detection. Existing models may lack signals for the following outlier detection steps and hence cannot be directly plugged in. However, it is possible to adapt certain models to the outlier detection task.

3.2 Identifying Semantic Focuses

The semantic regions learned from the Embedded vMF Allocation model provide a set of candidates frequently mentioned by documents in the corpus. However, not all of them are semantic focuses of the corpus — some are too general to distinguish outlier and normal document.

We notice that uninformative semantic regions (*e.g.* a semantic region containing {"percent", "average", "compare", ...}) tend to have more scattered distribution over embedded vectors, possibly because of the diverse context of their usage. In contrast, corpus-specific semantic regions are more concentrated, (*e.g.* a semantic region containing {"drugs", "antidepressant", "prescription", ...}). Modeling semantic regions by vMF distributions provides us with a parsimonious signal to characterize how concentrated a semantic region is, *i.e.* the concentration parameter κ_t. This allows us to simply filter unqualified semantic regions with too small concentration parameters and obtain high-quality semantic focuses. Let a binary variable ϕ_t ($t = 1, 2, \cdots, T$) indicate whether the t-th vMF distribution is a semantic focus. Suppose a user specifies a threshold parameter $0 \leq \beta \leq 1$. We can determine ϕ_t by estimating the log-normal distribution that generates all κ_t's, $logNormal(\hat{m}, \hat{\sigma}^2)$, where

$$
\hat{m} = \frac{1}{T}\sum_t \log(\kappa_t), \ \hat{\sigma}^2 = \frac{1}{T}\sum_t \Big(\log(\kappa_t) - \hat{m}\Big)^2
$$

Set $\hat{F}_\kappa(\cdot)$ to be its cumulative distribution function. We assign $\phi_t = 1$ for semantic regions with $\kappa_t \geq F_\kappa^{-1}(\beta)$, and filter all the other semantic regions as $\phi_t = 0$.

Although parameter β needs to be set manually, our experiments suggest the performance is not quite sensitive to its value.

3.3 Document Outlierness

In this subsection, we start with a straightforward definition of outlierness based on the mined semantic focuses. Then we present several refinements to improve its robustness.

Baseline outlierness measure. A straightforward intuition is to assume outlier documents averagely have fewer words or phrases drawn from semantic focuses. To estimate this, we first need to calculate the probability of each word being drawn from the semantic focuses.

$$P(\phi_{z_{ij}} = 1 | \mathbf{x}_{ij}, \boldsymbol{\pi}_i) = \frac{\sum_t \phi_t \boldsymbol{\pi}_i^{(t)} \text{vMF}(\mathbf{x}_{ij} | \boldsymbol{\mu}_t, \kappa_t)}{\sum_t \boldsymbol{\pi}_i^{(t)} \text{vMF}(\mathbf{x}_{ij} | \boldsymbol{\mu}_t, \kappa_t)}$$

It is then possible to estimate the expected percentage of words *not* drawn from semantic focuses in each document as the outlierness:

$$\Omega_{\text{sf}}(d_i) = 1 - \frac{1}{|d_i|} \sum_{j=1}^{|d_i|} P(\phi_{z_{ij}} = 1 | \mathbf{x}_{ij}, \boldsymbol{\pi}_i) \quad (1)$$

However, due to the noisiness in text data, this assumption oversimplifies the characterization of outlier documents. In practice, we observe the following two issues: lexically general words/phrases, and noisy content in documents.

Penalizing lexically general words and phrases. Not all words or phrases close to semantic focuses are strong indicators of normal documents. General words (*e.g.* "science") can happen to be semantically close to a semantic focus, but are not as specific as most other words close to it (*e.g.* "medical research"). Therefore, we utilize a background corpus D_{bg} to calculate the specificity of the word. Assuming the actual mention of the word can be chosen from either the general background, or a corpus-specific vocabulary, we write down the probability that a word is corpus-specific to be:

$$P(\lambda_{ij} | w_{ij}) = \frac{nd(w_{ij})/|D|}{nd(w_{ij})/|D| + nd_{bg}(w_{ij})/|D_{bg}|}$$

where $nd(w) = |\{d_i | w \in d_i, d_i \in D\}|$ is the number of documents in D containing word w; $nd_{bg}(w) = |\{d_i | w \in d_i, d_i \in D_{bg}\}|$ is the number of documents containing word w in the background corpus D_{bg}; λ_{ij} is a binary random variable indicating whether w_{ij} is specific enough.

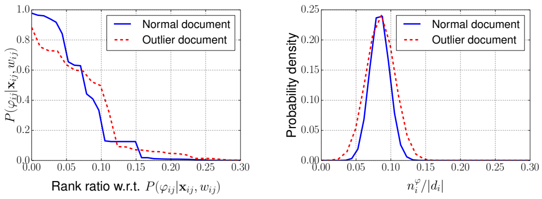
For each word, we define the word is *orthodox* if the word is not only semantically close to a semantic focus of the corpus, but also sufficiently specific. We then define the probability that a word or phrase w_{ij} in document d_i is orthodox as:

$$P(\varphi_{ij} | \mathbf{x}_{ij}, \boldsymbol{\pi}_i w_{ij}) = P(\phi_{z_{ij}} | \mathbf{x}_{ij}, \boldsymbol{\pi}_i) P(\lambda_{ij} | w_{ij})$$

where $\varphi_{ij} = 1$ indicates that w_{ij} (or equivalently $\mathbf{x}_{ij}$) is orthodox.

Now, we can define a second outlierness measure as the expected percentage of words that are *not* orthodox.

$$\Omega_{\text{e}}(d_i) = 1 - \frac{1}{|d_i|} \sum_{j=1}^{|d_i|} P(\varphi_{ij} | \mathbf{x}_{ij}, \boldsymbol{\pi}_i, w_{ij}) \quad (2)$$



(a) Ranked orthodox probability $P(\varphi_{ij} | \mathbf{x}_{ij}, \boldsymbol{\pi}_i, w_{ij})$ (b) Probability distribution of random variable $n_i^{\varphi}/|d_i|$

Figure 1: Comparison of a normal document and an outlier document in a news corpus ("Health" topic).

Noisy content in documents. We present the second issue of normal documents with an example. We compare a normal document in a corpus of New York Times news articles with tag "Health", to another document originally from another corpus, but with its outlierness calculated with regard to the semantic focuses of the "Health" corpus.

In Figure 1(a), we show the distribution of inferred orthodox probability $P(\varphi_{ij} = 1 | \mathbf{x}_{ij}, w_{ij})$ by ranking the words or phrases according to their probability value. We can observe that the outlier document barely has any words or phrases surely orthodox, while the normal document has 5% of words or phrases with a probability no less than 0.8 to be orthodox. However, if we simply take the average, these two documents become indistinguishable as the average is substantially dominated by the "tail" where most words or phrases in either documents are clearly not orthodox. Let n_i^{φ} be a random variable indicating the true number of orthodox words or phrases in document d_i. Since n_i^{φ} follows a Poisson-Binomial distribution, we can plot the probability distribution of n_i^{φ} normalized by the length of the document, as shown in Figure 1(b). It can be observed that the difference between the normalized expectation $\mathbb{E}[n_i^{\varphi}]/d_i$ of two documents is insignificant. Therefore, the measure described in Equation (2) will be unable to tell the difference between these two documents.

This example illustrates why the strategy of taking the average over the whole document can make mistakes, and also provides an important insight. As long as a document has a (potentially small) portion of words or phrases that are highly certain to be orthodox, it should not be considered as an outlier. Based on the above observation, we propose a third outlierness measure.

Orthodox quantile outlierness. We define a *quantile-based outlierness* definition to rank document outliers. Notice that the distribution of random variable n_i^{φ} follows a Poisson-Binomial distribution, which is the total number of success trials when one tosses a coin for each word or phrase in the document to determine whether it is orthodox with probability $P(\varphi_{ij}|\mathbf{x}_{ij}, w_{ij})$.

Moreover, we define the first $\frac{1}{1-\theta}$-quantile of the Poisson-Binomial distribution of n_i^{φ} as:

$$q_\theta(n_i^{\varphi}) = \sup_q \{q : P(n_i^{\varphi} \geq q) \geq \theta\} \qquad (3)$$

where $0 < \theta < 1$ is a given parameter close to 1. Intuitively, it measures the maximum lower bound of n_i^{φ} we can guarantee with confidence θ.

Based on Equation (3), we can give a formalized definition of our proposed outlierness:

$$\Omega_{\theta\text{-q}}(d_i) = 1 - \frac{q_\theta(n_i^{\varphi}) + 1}{|d_i| + 1} \qquad (4)$$

where the $\frac{1}{1-\theta}$-quantile is normalized by the document length with a smoothing constant. The cumulative probability distribution of a Poisson-Binomial distribution can be efficiently calculated by dynamic programming (Chen and Liu, 1997).

The advantage of the last proposed outlierness measure is that it emphasizes more on the highly orthodox words or phrases and eliminates the noise from a number of relatively uncertain ones.

4 Experiment Setup

4.1 Data Sets

New York Times News (NYT). We collected 41,959 news article published in 2013 from The New York Times API[2]. Each article is assigned with a unique label indicating in which section the article is published, such as Arts, Travel, Sports, and Health. There are totally 9 section labels in our collected data set. We treat papers in each section as a corpus D. Thereby we have a set of corpora $\mathcal{D} = \{D_s\}$, without overlapping documents. We also have an external news data set D_e crawled from Google news, with 51,114 news article published in 2015 without any label information.

ArnetMiner Paper Abstracts (ARNET). We employ abstracts of papers published in the field of computer science up to 2013, collected by ArnetMiner (Tang et al., 2008), and assign each paper into a field, according to Wikipedia[3]. We use papers from a set of domains to serve as an external corpus D_e, while papers in other domains form different corpora $\mathcal{D} = \{D_s\}$. Each domain (*e.g.*, data mining, computational biology, and computer graphics) forms a corpus D_s respectively. Again, notice that the corpora do not have overlapping documents with each other.

Table 1: Data set statistics.

Data set	Corpus D		External corpus D_e	
	Avg. $\lvert D \rvert$	Avg. $\lvert d \rvert$	$\lvert D_e \rvert$	Avg. $\lvert d \rvert$
NYT	4,662.11	592.66	52,114	471.63
ARNET	2,930.60	137.21	11,463	152.17

A summary is presented in Table 1.

Benchmark generation. Since we do not have true labels for outliers in a corpus, we use injection method to generate outlier detection benchmark. For each data set, we randomly select a corpus $D_s \in \mathcal{D}$ and mark all of its document as "normal documents". We then randomly select another corpus $D'_s \in \mathcal{D}, D'_s \neq D_s$, to inject ω documents from D'_s into D_s and mark them as outliers. We confine ω to be a small integer less than 1% of the size of $|D_s|$. More concretely, ω is an integer uniformly sampled from $(0, 0.01|D_s|]$.

For each data set, we randomly generate 10 outlier detection benchmarks, and evaluate the overall performance by the average performance on all the benchmarks.

4.2 Methods Evaluated

We compare the performances of the following methods.

Cosine similarity based. We characterize each document as a vector, and use the negative average cosine similarity between each document and the corpus as outlierness. We use two different ways to vectorize documents: TF-IDF weighted, and paragraph2vec (Le and Mikolov, 2014). The two methods are denoted as TFIDF-COS and P2V-COS respectively.

KL divergence based. We represent each document as a probability distribution, and the entire corpus as another probability distribution. Then we use the KL-divergence between each document and the entire corpus as the outlierness. We also

[2] `http://developer.nytimes.com/docs`

[3] `https://en.wikipedia.org/wiki/List_of_computer_science_conferences`

use two different ways to calculate the probability distribution. The first is to estimate the unigram distribution for each document and the entire corpus respectively, denoted as UNI-KL. The other is to first perform LDA on the entire corpus with 10 topics, and then infer topical allocation distribution of each document and the entire corpus. This method is represented as TM-KL.

Our method Our quantile based method is denoted as VMF-Q. We also provide two baselines derived from our own method as an ablation analysis. One method abandons the quantile based outlierness but use the expected orthodox percentage as Equation (2), denoted as VMF-E. The other method further removes the penalty on lexical general words and phrases, using Equation (1), denoted as VMF-SF.

4.3 Evaluation Measures

In most outlier detection applications, people are more concerned with recall. We measure the performance by *recall at a certain percentage*. More specifically, we compute the recall of outlier detection if the user checks a certain percentage r of the top-ranked documents in the output results. Since in our benchmark generation, the percentage of outliers does not exceed 1%. Therefore, the perfect results for any $r \geq 1\%$ should be 1.0.

We choose r to be 1%, 2%, and 5% respectively and evaluate different methods with recall at top-r (percentage). We also report the performance in terms of mean average precision (MAP).

4.4 Parameter Configurations

All benchmark data sets are preprocessed as described in Section 2. In the NYT data set we remove words or phrases within top 20% with respect to document frequency, while in the ARNET data set we remove the top 10%. The document frequency is calculated based on a background corpus D_{bg}, which is the same as the external corpus of NYT. Word embedding are trained on the external data set D_e using code of Mikolov *et al.* (Mikolov et al., 2013) with default parameter configurations, where the embedded vector length is set to 200. For paragraph2vec, we learn the length-100 vectors for each document along with the external data set to guarantee sufficient training data.

Table 2: Performance comparison of different outlier document detection methods. All results are shown as percents.

Data set	Method	MAP	Rcl@1%	Rcl@2%	Rcl@5%
NYT	TFIDF-COS	05.03	04.73	06.72	14.72
	P2V-COS	22.07	23.45	44.64	66.18
	UNI-KL	10.28	11.92	16.32	31.34
	TM-KL	14.51	16.50	16.50	24.67
	VMF-SF	33.70	31.03	44.45	62.60
	VMF-E	36.57	35.91	49.41	67.56
	VMF-Q	**41.88**	**56.99**	**63.29**	**79.23**
ARNET	TFIDF-COS	08.99	15.40	18.75	30.23
	P2V-COS	07.39	10.51	14.78	24.14
	UNI-KL	07.46	14.13	22.26	39.40
	TM-KL	10.09	12.04	15.37	20.24
	VMF-SF	10.69	12.05	22.58	44.51
	VMF-E	10.51	12.67	25.92	45.37
	VMF-Q	**19.74**	**22.40**	**34.40**	**53.87**

For the prior vMF distribution, we set $C_0 = 0.1$, a sufficiently small number so the prior distribution is close to a uniform distribution. $\boldsymbol{\mu}_0$ is set as a normalized all-1 vector. We also set $m_0 = log(100)$, and $\sigma^2 = 0.01$. The total number for Gibbs sampling is set to be 50 times of the total count of z_{ij}'s (*i.e.* $\eta = 50$). The number of vMF distributions T is set to 20 in the NYT data set and 10 in the ARNET data set respectively, due to the smaller sizes of corpora in the ARNET data set.

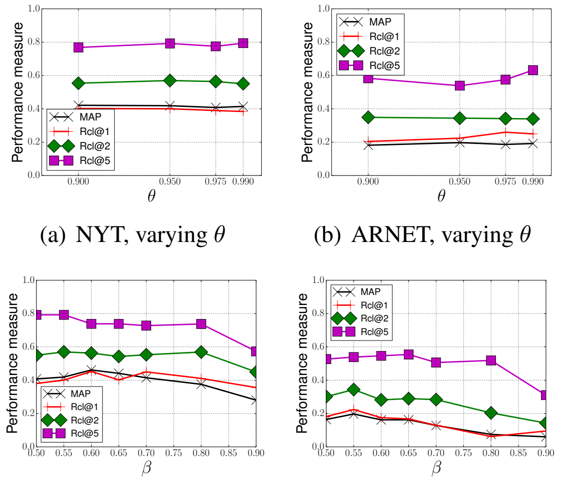
To determine semantic focuses, we set threshold parameter $\beta = 0.55$ for both data sets. The confidence parameter θ in outlierness calculation is set to 0.95 in both data sets. Our experiments later will show the performance is relatively robust to different configurations of both parameters.

5 Results

We present the experimental results in this section.

Performance comparison. Table 2 shows performance of different outlier document detection methods. It can be observed that our method outperforms all the baselines in both data sets. In both data sets, VMF-Q can achieve a 45% to 135% increase from baselines in terms of recall by examining the top 1% outliers. Generally, performances of most methods are lower in the ARNET data set comparing to NYT, potentially because the relatively short document lengths and more technical terminologies in ARNET.

Ablation analysis. Both refinements of the outlierness measure benefits the performance. Specifically, by changing the average based outlierness to quantile based outlierness, the recall@1% can be improved by 50-75%, and the recall@5% can also be improved by more than 17%.



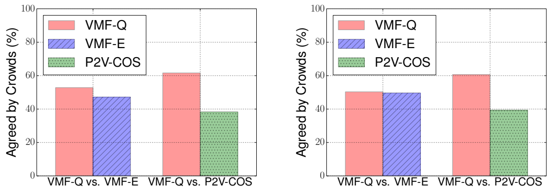
(a) NYT, varying θ

(b) ARNET, varying θ

(c) NYT, varying β

(d) ARNET, varying β

Figure 2: Performance of outlier document detection with different parameter configurations.



(a) Health

(b) Art

Figure 3: Crowd evaluation to compare different outlier detection methods on two corpora in NYT data set.

Sensitivity studies of parameters. We study if our proposed method is sensitive to the confidence parameter θ and filtering threshold parameter β. We compare the performance of VMF-Q by varying each parameter on both data sets. Figure 2(a) and 2(b) show that the performance is not very sensitive to different values of θ, as long as θ is sufficiently large (close to 1). Figure 2(c) and 2(d) show that the performance is relatively stable when β is between 0.5 and 0.7, but drops a little when β is set to larger value.

Human judgments. We compare VMF-Q to VMF-E and P2V-COS respectively by crowdsourcing, *without* artificially inserting "outliers". We conduct this experiments on two corpora in NYT data sets with topic "Health" and "Art" respectively. To compare two methods, we randomly select pairs of documents d_i and d_j such that both are ranked as top-10% outliers by at least one method, but their orders in the two rankings disagree. We conduct the experiments on CrowdFlower. Online crowd workers are given d_i and d_j as well as other documents in the corpus, and are asked to judge which one of d_i and d_j deviates more from the corpus. For each corpus, we select 200 pairs of documents.

Before taking the questions, each crowd worker needs to go through at least 10 "test questions" which we know the correct answer. These questions are constructed by taking one document from the corpus as d_i and another document not from the corpus as d_j. Therefore, the one not from the corpus should be the answer. A crowd worker needs to achieve no less than 80% of accuracy to be eligible to work on actual questions, and the accuracy needs to be maintained over 80% during the work, which is measured by "test questions" hidden in actual questions. Each question is answered by 3 workers. The final answer is determined by majority voting.

Figure 3 presents the results. On both corpora, there are significantly more workers tend to agree with VMF-Q comparing to P2V-COS, with significance level $\alpha = 0.05$. This further verifies that our method VMF-Q can achieve better performance than the P2V-COS baseline. On the other hand, on both data sets we can still observe more workers favoring VMF-Q than VMF-E, but the difference is not as large as the difference between VMF-Q and P2V-COS.

Case study. We also conduct a case study to show how our proposed method outperforms other baselines. Table 3 shows two pairs of documents in "Health" corpus of NYT data set. The left two columns show some comparing methods and their higher ranked outlier documents. The row of "Crowds" shows the outlier document chosen by human workers from the crowdsourcing platform, with a consensus of opinions from multiple workers.

In the first document pair, document A is about gun control policy and is substantially irrelevant to "Health" topic, while document B is about lung infection cases. Document A is a significant outlier, and VMF-Q and VMF-E also agree with our intuition. However, paragraph2vec (P2V) ranks document B higher, probably because it tries to summarize the entire document.

In the second document pair, document B is clearly *not* an outlier as the story is about a new book of AIDS. In comparison, document A dis-

Table 3: Case study of documents in "Health" corpus of NYT data set. We present several pairs of documents and how different methods rank the pair. The "Outlier" column indicates the document ranked higher in the outlier document ranking generated by the corresponding methods, and the row "Crowds" shows the ranking given by human evaluators.

Method	Outlier	Document A	Document B
P2V-COS	Doc B	*CHICAGO (AP) States with the most gun control laws have the fewest gun-related deaths, according to a study that suggests sheer quantity of measures might make a difference ...*	*A prominent Scottish bagpiping school has warned pipers around to world to clean their instruments regularly after one of its longtime members nearly died of a lung infection ...*
VMF-E	Doc A		
VMF-Q	Doc A		
Crowds	Doc A		
P2V-COS	Doc B	*ATLANTA There's more evidence that U.S. births may be leveling off after years of decline. The number of babies born last year only slipped a little, ...*	*Young men in a state prison for juveniles and professors of library science from the University of South Carolina have joined forces to fight AIDS with a graphic novel ...*
VMF-E	Doc B		
VMF-Q	Doc A		
Crowds	Doc A		

cussing U.S. population is an outlier. However, a great part of document B is about the content of the book, which confuses baselines P2V and VMF-E, as both methods tend to summarize the entire document and highly relevant words like "AIDS" are overwhelmed by the majority of the document. The only method that agrees with human annotators is VMF-Q.

6 Conclusion

In this paper, we propose a novel task of detecting document outliers from a given corpus. We propose a generative model to identify semantic focuses of a corpus, each represented as a vMF distribution in the embedded space. We also design a document outlierness measure. We experimentally verify the effectiveness of our methods. We hope this work provides insights for further studies on outlier document texts in specific domains, and in more challenging settings such as detecting outliers from crowdsourced data.

References

Kayhan Batmanghelich, Ardavan Saeedi, Karthik Narasimhan, and Samuel Gershman. 2016. Nonparametric spherical topic modeling with word embeddings. In *ACL*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2001. Latent dirichlet allocation. In *NIPS*, pages 601–608.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1–15:58.

Sean X Chen and Jun S Liu. 1997. Statistical applications of the poisson-binomial and conditional bernoulli distributions. *Statistica Sinica*, pages 875–892.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for topic models with word embeddings. In *ACL*, pages 795–804.

Siddharth Gopal and Yiming Yang. 2014. Von mises-fisher clustering models. In *ICML*, pages 154–162.

David Guthrie. 2008. *Unsupervised Detection of Anomalous Text*. Ph.D. thesis, University of Sheffield.

Milos Hauskrecht, Iyad Batal, Michal Valko, Shyam Visweswaran, Gregory F Cooper, and Gilles Clermont. 2013. Outlier detection for patient monitoring and alerting. *Journal of Biomedical Informatics*, 46(1):47–55.

Victoria J Hodge and Jim Austin. 2004. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126.

S. P. Kasiviswanathan, G. Cong, P. Melville, and R. D. Lawrence. 2013. Novel document detection for massive data streams using distributed dictionary learning. *IBM J. Res. Dev.*, 57(3-4):1:9–1:9.

Shiva P Kasiviswanathan, Huahua Wang, Arindam Banerjee, and Prem Melville. 2012. Online l1-dictionary learning with application to novel document detection. In *NIPS*, pages 2258–2266.

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

Jialu Liu, Jingbo Shang, Chi Wang, Xiang Ren, and Jiawei Han. 2015. Mining quality phrases from massive text corpora. In *SIGMOD*, pages 1729–1744. ACM.

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

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *KDD*, pages 990–998. ACM.

Jian Zhang, Zoubin Ghahramani, and Yiming Yang. 2004. A probabilistic model for online document clustering with application to novelty detection. In *NIPS*, pages 1617–1624.

Yi Zhang, Jamie Callan, and Thomas Minka. 2002. Novelty and redundancy detection in adaptive filtering. In *SIGIR*, pages 81–88. ACM.

Detecting and Explaining Causes From Text For a Time Series Event

Dongyeop Kang, Varun Gangal, Ang Lu, Zheng Chen, Eduard Hovy
Language Technology Institute
Carnegie Mellon University
{dongyeok,vgangal,alu1,zhengc1,hovy}@cs.cmu.edu

Abstract

Explaining underlying causes or effects about events is a challenging but valuable task. We define a novel problem of generating explanations of a time series event by (1) searching cause and effect relationships of the time series with textual data and (2) constructing a connecting chain between them to generate an explanation. To detect causal features from text, we propose a novel method based on the Granger causality of time series between features extracted from text such as N-grams, topics, sentiments, and their composition. The generation of the sequence of causal entities requires a commonsense causative knowledge base with efficient reasoning. To ensure good interpretability and appropriate lexical usage we combine symbolic and neural representations, using a neural reasoning algorithm trained on commonsense causal tuples to predict the next cause step. Our quantitative and human analysis show empirical evidence that our method successfully extracts meaningful causality relationships between time series with textual features and generates appropriate explanation between them.

1 Introduction

Producing true causal explanations requires deep understanding of the domain. This is beyond the capabilities of modern AI. However, it is possible to collect large amounts of causally related events, and, given powerful enough representational variability, to construct cause-effect chains by selecting individual pairs appropriately and linking them together. Our hypothesis is that chains composed of locally coherent pairs can suggest overall causation.

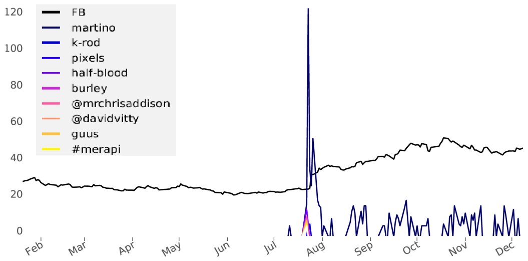


Figure 1: Example of causal features for Facebook's stock change in 2013. The causal features (e.g., *martino*, *k-rod*) rise before the Facebook's rapid stock rise in August.

In this paper, we view *causality* as (commonsense) cause-effect expressions that occur frequently in online text such as news articles or tweets. For example, "*greenhouse gases causes global warming*" is a sentence that provides an 'atomic' link that can be used in a larger chain. By connecting such causal facts in a sequence, the result can be regarded as a *causal explanation* between the two ends of the sequence (see Table 1 for examples).

This paper makes the following contributions:

- we define the problem of causal explanation generation,
- we detect causal features of a time series event (CSpikes) using Granger (Granger, 1988) method with features extracted from text such as N-grams, topics, sentiments, and their composition,
- we produce a large graph called CGraph of local cause-effect units derived from text and develop a method to produce causal explanations by selecting and linking appropriate units, using neural representations to enable unit matching and chaining.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2758–2767
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Table 1: Examples of generated causal explanation between some temporal causes and target companies' stock prices.

party $\xmapsto{cut}$ budget_cuts $\xmapsto{lower}$ budget_bill $\xmapsto{decreas}$ republicans $\xmapsto{caus}$ obama $\xmapsto{leadto}$ facebook_polls $\xmapsto{caus}$ **facebook's stock ↓**

The problem of causal explanation generation arises for systems that seek to determine causal factors for events of interest automatically. For given time series events such as companies' stock market prices, our system called CSPIKES detects events that are deemed causally related by time series analysis using Granger Causality regression (Granger, 1988). We consider a large amount of text and tweets related to each company, and produces for each company time series of values for hundreds of thousands of word n-grams, topic labels, sentiment values, etc. Figure 1 shows an example of causal features that temporally causes Facebook's stock rise in August.

However, it is difficult to understand how the statistically verified factors actually cause the changes, and whether there is a latent causal structure relating the two. This paper addresses the challenge of finding such latent causal structures, in the form of *causal explanations* that connect the given cause-effect pair. Table 1 shows example causal explanation that our system found between *party* and *Facebook's stock fall (↓)*.

To construct a general causal graph, we extract all potential causal expressions from a large corpus of text. We refer to this graph as CGRAPH. We use FrameNet (Baker et al., 1998) semantics to provide various causative expressions (verbs, relations, and patterns), which we apply to a resource of $183,253,995$ sentences of text and tweets. These expressions are considerably richer than previous rule-based patterns (Riaz and Girju, 2013; Kozareva, 2012). CGRAPH contains 5,025,636 causal edges.

Our experiment demonstrates that our causality detection algorithm outperforms other baseline methods for forecasting future time series values. Also, we tested the neural reasoner on the inference generation task using the BLEU score. Additionally, our human evaluation shows the relative effectiveness of neural reasoners in generating appropriate lexicons in explanations.

2 CSPIKES: Temporal Causality Detection from Textual Features

The objective of our model is, given a target time series y, to find the best set of textual features $F = \{f_1, ..., f_k\} \subseteq X$, that maximizes sum of causality over the features on y, where X is the set of all features. Note that each feature is itself a time series:

$$\arg\max_F \mathbf{C}(y, \Phi(X, y)) \tag{1}$$

where $\mathbf{C}(y, x)$ is a causality value function between y and x, and Φ is a linear composition function of features f. Φ needs target time series y as well because of our graph based feature selection algorithm described in the next sections.

We first introduce the basic principles of Granger causality in Section 2.1. Section 2.2 describes how to extract good source features $F = \{f_1, ..., f_k\}$ from text. Section 2.3 describes the causality function $\mathbf{C}$ and the feature composition function Φ.

2.1 Granger Causality

The essential assumption behind Granger causality is that a cause must occur before its effect, and can be used to predict the effect. Granger showed that given a target time series y (effect) and a source time series x (cause), *forecasting* future target value y_t with both past target and past source time series $E(y_t|y_{<t}, x_{<t})$ is significantly powerful than with only past target time series $E(y_t|y_{<t})$ (plain auto-regression), if x and y are indeed a cause-effect pair. First, we learn the parameters α and β to maximize the prediction expectation:

$$E(y_t|y_{<t}, x_{t-l}) = \sum_{j=1}^{m} \alpha_j y_{t-j} + \sum_{i=1}^{n} \beta_i x_{t-i} \tag{2}$$

where i and j are size of lags in the past observation. Given a pair of causes x and a target y, if β has magnitude significantly higher than zero (according to a confidence threshold), we can say that x causes y.

2.2 Feature Extraction from Text

Extracting meaningful features is a key component to detect causality. For example, to predict future trend of presidential election poll of *Donald Trump*, we need to consider his past poll data as well as people's reaction about his pledges such as *Immigration*, *Syria* etc. To extract such "good" features crawled from on-line media data, we propose three different types of features: F_{words}, F_{topic}, and F_{senti}.

F_{words} is time series of N-gram words that reflect popularity of the word over time in on-line media. For each word, the number of items (e.g., tweets, blogs and news) that contains the N-gram word is counted to get the day-by-day time series. For example, $x^{Michael\_Jordan} = [12, 51, ..]$ is a time series for a bi-gram word *Michael Jordan*. We filter out stationary words by using simple measures to estimate how dynamically the time series of each word changes over time. Some of the simple measures include Shannon entropy, mean, standard deviation, maximum slope, and number of rise and fall peaks.

F_{topic} is time series of latent topics with respect to the target time series. The latent topic is a group of semantically similar words as identified by a standard topic clustering method such as LDA (Blei et al., 2003). To obtain temporal trend of the latent topics, we choose the top ten frequent words in each topic and count their occurrence in the text to get the day-by-day time series. For example, $x^{healthcare}$ means how popular the topic *healthcare* that consists of *insurance*, *obamacare* etc, is through time.

F_{senti} is time series of sentiments (positive or negative) for each topic. The top ten frequent words in each topic are used as the keywords, and tweets, blogs and news that contain at least one of these keywords are chosen to calculate the sentiment score. The day-by-day sentiment series are then obtained by counting positive and negative words using OpinionFinder (Wilson et al., 2005), and normalized by the total number of the items that day.

2.3 Temporal Causality Detection

We define a causality function $\mathbf{C}$ for calculating causality score between target time series y and source time series x. The causality function $\mathbf{C}$ uses Granger causality (Granger, 1988) by fitting the two time series with a Vector AutoRegressive model with exogenous variables (VARX) (Hamilton, 1994): $y_t = \alpha y_{t-l} + \beta x_{t-l} + \epsilon_t$ where ϵ_t is a white Gaussian random vector at time t and l is a lag term. In our problem, the number of source time series x is not single so the prediction happens in the k multi-variate features $X = (f_1, ...f_k)$ so:

$$y_t = \alpha y_{t-l} + \boldsymbol{\beta}(f_{1,t-l} + ... + f_{k,t-l}) + \epsilon_t \quad (3)$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is the coefficient matrix of the target y and source X time series respectively, and ϵ is a residual (prediction error) for each time series. $\boldsymbol{\beta}$ means contributions of each lagged feature $f_{k,t-l}$ to the predicted value y_t. If the variance of $\boldsymbol{\beta_k}$ is reduced by the inclusion of the feature terms $f_{k,t-l} \in X$, then it is said that $f_{k,t-l}$ Granger-causes y.

Our causality function $\mathbf{C}$ is then $\mathbf{C}(y, f, l) = \Delta(\beta_{y,f,l})$ where Δ is change of variance by the feature f with lag l. The total Granger causality of target y is computed by summing the change of variance over all lags and all features:

$$\mathbf{C}(y, X) = \sum_{k,l} \mathbf{C}(y, f_k, l) \quad (4)$$

We compose best set of features Φ by choosing top k features with highest causality scores for each target y. In practice, due to large amount of computation for pairwise Granger calculation, we make a bipartite graph between features and targets, and address two practical problems: *noisiness* and *hidden edges*. We filter out noisy edges based on TFIDF and fill out missing values using non-negative matrix factorization (NMF) (Hoyer, 2004).

3 CGraph Construction

Formally, given source x and target y events that are causally related in time series, if we could find a sequence of cause-effect pairs $(x \mapsto e_1)$, $(e_1 \mapsto e_2)$, ... $(e_t \mapsto y)$, then $e_1 \mapsto e_2, ... \mapsto e_t$ might be a good causal explanation between x and y. Section 3 and 4 describe how to bridge the causal gap between given events (x, y) by (1) constructing a large general cause-effect graph (CGraph) from text, (2) linking the given events to their equivalent entities in the causal graph by finding the internal paths $(x \mapsto e_1, ...e_t \mapsto y)$ as causal explanations, using neural algorithms.

CGraph is a knowledge base graph where edges are directed and causally related between entities. To address less representational variability of rule based methods (Girju, 2003; Blanco et al., 2008; Sharp et al., 2016) in the causal graph construction, we used FrameNet (Baker et al., 1998) semantics. Using a semantic parser such

Table 2: Example (relation, cause, effect) tuples in different categories (manually labeled): *general*, *company*, *country*, and *people*. FrameNet labels related to causation are listed inside parentheses. The number of distinct relation types are 892.

	Relation	**Cause ↦ Effect**	
General	causes (Causation)	the virus (Cause)	aids (Effect)
	cause (Causation)	greenhouse gases (Cause)	global warming (Effect)
	forced (Causation)	the reality of world war ii (Cause)	the cancellation of the olympics (Effect)
Company	heats (Cause_temperature_change)	microsoft vague on windows (Item)	legislation battle (Agent)
	promotes (Cause_change_of_position_on_a_scale)	chrome (Item)	google (Agent)
	makes (Causation)	twitter (Cause)	love people you 've never met facebook (Effect)
Country	developing (Cause_to_make_progress)	north korea (Agent)	nuclear weapons (Project)
	improve (Cause_to_make_progress)	china (Agent)	its human rights record (Project)
	forced (Causation)	war with china (Cause)	the japanese to admit , in july 1938 (Effect)
People	attracts (Cause_motion)	obama (Agent)	more educated voters (Theme)
	draws (Cause_motion)	on america 's economic brains (Goal)	barack obama (Theme)
	made (Causation)	michael jordan (Cause)	about $ 33 million (Effect)

as SEMAFOR (Chen et al., 2010) that produces a FrameNet style analysis of semantic predicate-argument structures, we could obtain lexical tuples of causation in the sentence. Since our goal is to collect only causal relations, we extract total 36 causation related frames[1] from the parsed sentences.

Table 3: Number of sentences parsed, number of entities and tuples, and number of edges (*KB-KB*, *KBcross*) expanded by Freebase in CGRAPH.

# Sentences	# Entities	# Tuples	# *KB-KB*	# *KBcross*
183,253,995	5,623,924	5,025,636	470,250	151,752

To generate meaningful explanations, high coverage of the knowledge is necessary. We collect six years of tweets and NYT news articles from 1989 to 2007 (See Experiment section for details). In total, our corpus has 1.5 billion tweets and 11 million sentences from news articles. The Table 3 has the number of sentences processed and number of entities, relations, and tuples in the final CGRAPH.

Since the tuples extracted from text are very noisy [2], we constructed a large causal graph by linking the tuples with string match and filter out the noisy nodes and edges based on some graph statistics. We filter out nodes with very high degree that are mostly stop-words or auto-generated sentences. Too long or short sentences are also filtered out. Table 2 shows the (case, relation, effect) tuples with manually annotated categories such as *General*, *Company*, *Country*, and *People*.

4 Causal Reasoning

To generate a causal explanation using CGRAPH, we need traversing the graph for finding the path between given source and target events. This section describes how to efficiently traverse the graph by expanding entities with external knowledge base and how to find (or generate) appropriate causal paths to suggest an explanation using symbolic and neural reasoning algorithms.

4.1 Entity Expansion with Knowledge Base

A simple choice for traversing a graph are the traditional graph searching algorithms such as Breadth-First Search (BFS). However, the graph searching procedure is likely to be incomplete (*low recall*), because simple string match is insufficient to match an effect to all its related entities, as it misses out in the case where an entity is semantically related but has a lexically different name.

To address the *low recall* problem and generate better explanations, we propose the use of knowledge base to augment our text-based causal graph with real-world semantic knowledge. We use Freebase (Google, 2016) as the external knowledge base for this purpose. Among 1.9 billion edges in original Freebase dump, we collect its first and second hop neighbours for each target events.

While our CGRAPH is lexical in nature, Freebase entities appear as identifiers (MIDs). For entity linking between two knowledge graphs, we need to annotate Freebase entities with their lexical names by looking at the wiki URLs. We refer to the edges with freebase expansion as *KB-KB* edges, and link the *KB-KB* with our CGRAPH us-

[1] Causation, Cause_change, Causation_scenario, Cause_benefit_or_detriment, Cause_bodily_experience, etc.

[2] SEMAFOR has around 62% of accuracy on held-out set.

ing lexical matching, referring as *KBcross* edges (See Table 3 for the number of the edges).

4.2 Symbolic Reasoning

Simple traversal algorithms such as BFS are infeasible for traversing the CGRAPH due to the large number of nodes and edges. To reduce the search space k in $e_t \mapsto \{e^1_{t+1}, ...e^k_{t+1}\}$, we restricted our search by depth of paths, length of words in entity's name, and edge weight.

Algorithm 1 Backward Causal Inference. y is target event, d is depth of BFS, l is lag size, BFS_{back} is Breadth-First search for one depth in backward direction, and $\sum_l \mathbf{C}$ is sum of Granger causality over the lags.

```
$\mathbb{S} \leftarrow y, d = 0$
while ($\mathbb{S} = \varnothing$) or ($d > D_{max}$) do
    $\{e^1_{-d}, ...e^k_{-d}\} \leftarrow BFS_{back}(\mathbb{S})$
    $d = d + 1, \mathbb{S} \leftarrow \varnothing$
    for $j$ in $\{1, ..., k\}$ do
        if $\sum_l \mathbf{C}(y, e^j_{-d}, l) < \epsilon$ then $\mathbb{S} \leftarrow e^j_{-d}$
```

For more efficient inference, we propose a backward algorithm that searches potential causes (instead of effects) $\{e^1_t, ...e^k_t\} \hookleftarrow e_{t+1}$ starting from the target node $y = e_{t+1}$ using Breadth-first search (BFS). It keeps searching backward until the node e^j_i has less Granger confident causality with the target node y (See Algorithm 4 for causality calculation). This is only possible because our system has temporal causality measure between two time series events. See Algorithm 1 for detail.

4.3 Neural Reasoning

While symbolic inference is fast and straightforward, the sparsity of edges may make our inference semantically poor. To address the *lexical sparseness*, we propose a lexically relaxed reasoning using a neural network.

Inspired by recent success on alignment task such as machine translation (Bahdanau et al., 2014), our model learns the causal alignment between cause phrase and effect phrase for each type of relation between them. Rather than traversing the CGRAPH, our neural reasoner uses CGRAPH as a training resource. The encoder, a recurrent neural network such as LSTM (Hochreiter and Schmidhuber, 1997), takes the causal phrase while the decoder, another LSTM, takes the effectual phrase with their relation specific attention.

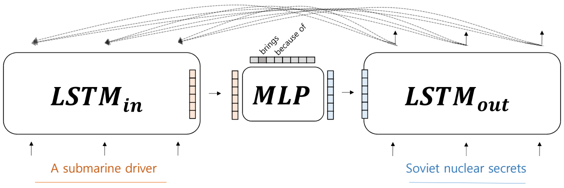


Figure 2: Our neural reasoner. The encoder takes causal phrases and decoder takes effect phrases by learning the causal alignment between them. The MLP layer in the middle takes different types of FrameNet relation and locally attend the cause to the effect w.r.t the relation (e.g., "because of", "led to", etc).

In original attention model (Bahdanau et al., 2014), the contextual vector c is computed by $c_i = a_{ij} * h_j$ where h_j is hidden state of causal sequence at time j and a_{ij} is soft attention weight, trained by feed forward network $a_{ij} = FF(h_j, s_{i-1})$ between input hidden state h_j and output hidden state s_{i-1}. The global attention matrix a, however, is easy to mix up all local alignment patterns of each relation.

For example, a tuple, *(north korea (Agent)* $\xmapsto[(Cause\_to\_make\_progress)]{developing}$ *nuclear weapons (Project))*, is different with another tuple, *(chrome (Item)* $\xmapsto[(Cause\_change\_of\_position)]{promotes}$ *google (Agent))* in terms of local type of causality. To deal with the *local attention*, we decomposed the attention weight a_{ij} by relation specific transformation in feed forward network:

$$a_{ij} = FF(h_j, s_{i-1}, r)$$

where FF has relation specific hidden layer and $r \in R$ is a type of relation in the distinct set of relations R in training corpus (See Figure 2).

Since training only with our causal graph may not be rich enough for dealing various lexical variation in text, we use pre-trained word embedding such as word2vec (Mikolov and Dean, 2013) trained on GoogleNews corpus[3] for initialization. For example, given a cause phrase *weapon equipped*, our model could generate multiple effect phrases with their likelihood: *(*$\xmapsto[0.54]{result}$*war)*, *(*$\xmapsto[0.12]{force}$*army reorganized)*, etc, even though there are no tuples exactly matched in CGRAPH.

[3] https://code.google.com/archive/p/word2vec/

Table 4: Examples of F_{words} with their temporal dynamics: Shannon entropy, mean, standard deviation, slope of peak, and number of peaks.

	entropy	mean	STD	max_slope	#-peaks
#lukewilliamss	0.72	22.01	18.12	6.12	31
happy_thanksgiving	0.40	61.24	945.95	3423.75	414
michael_jackson	0.46	141.93	701.97	389.19	585

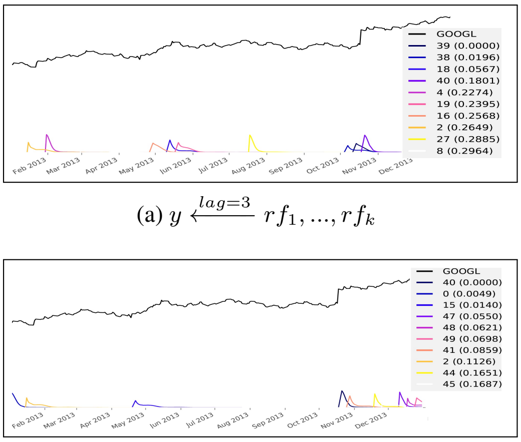
We trained our neural reasoner in either forward or backward direction. In prediction, decoder inferences by predicting effect (or cause) phrase in forward (or backward) direction. As described in the Algorithm 1, the backward inference continue predicting the previous causal phrases until it has high enough Granger confidence with the target event.

5 Experiment

Data. We collect on-line social media from tweets, news articles, and blogs. Our Twitter data has one million tweets per day from 2008 to 2013 that are crawled using Twitter's Garden Hose API. News and Blog dataset have been crawled from 2010 to 2013 using Google's news API. For target time series, we collect companies' stock prices in NASDAQ and NYSE from 2001 until present for 6,200 companies. For presidential election polls, we collect polling data of the 2012 presidential election from 6 different websites, including USA Today , Huffington Post, Reuters, etc.

Features. For N-gram word features F_{word},we choose the spiking words based on their temporal dynamics (See Table 4). For example, if a word is too frequent or the time series is too burst, the word should be filtered out because the trend is too general to be an event. We choose five types of temporal dynamics: Shannon entropy, mean, standard deviation, maximum slope of peak, and number of peaks; and delete words that have too low or high entropy, too low mean and deviation, or the number of peaks and its slope is less than a certain threshold. Also, we filter out words whose frequency is less than five. From the $1,677,583$ original words, we retain $21,120$ words as final candidates for F_{words} including uni-gram and bi-gram words.

For sentiment F_{senti} and topic F_{topic} features, we choose 50 topics generated for both politicians and companies separately using LDA, and then use top 10 words for each topic to calculate sentiment score for this topic. Then we can analyze the causality between sentiment series of a specific topic and collected time series.



(a) $y \xleftarrow{lag=3} rf_1, ..., rf_k$

(b) $y \xrightarrow{lag=3} rf_1, ..., rf_k$

Figure 3: Random causality analysis on **Googles**'s stock price change (y) and randomly generated features (rf) during 2013-01-01 to 2013-12-31. (a) shows how the random features rf cause the target y, while (b) shows how the target y causes the random features rf with lag size of 3 days. The color changes according to causality confidence to the target (blue is the strongest, and yellow is the weakest). The target time series has y scale of prices, while random features have y scale of causality degree $\mathbf{C}(y, rf) \subset [0, 1]$.

Tasks. To show validity of causality detector, first we conduct random analysis between target time series and randomly generated time series. Then, we tested forecasting stock prices and election poll values with or without the detected textual features to check effectiveness of our causal features. We evaluate our reasoning algorithm for generation ability compared to held-out cause-effect tuples using BLEU metric. Then, for some companies' time series, we describe some qualitative result of some interesting causal text features found with Granger causation and explanations generated by our reasoners between the target and the causal features. We also conducted human evaluation on the explanations.

5.1 Random Causality Analysis

To check whether our causality scoring function $\mathbf{C}$ detects the temporal causality well, we conduct a random analysis between target time series and randomly generated time series (See Figure 3). For Google's stock time series, we regularly move window size of 30 over the time and generate five days of time series with a random peak strength using a SpikeM model (Matsubara et al., 2012)[4]. The color of random time series rf changes from blue to yellow according to causality degree with the target $\mathbf{C}(y, rf)$. For example, blue is the strongest causality with target time series, while yellow is the weakest.

We observe that the strong causal (blue) features are detected just before (or after) the rapid rise of Google' stock price on middle October in (a) (or in (b)). With the lag size of three days, we observe that the strength of the random time series gradually decreases as it grows apart from the peak of target event. The random analysis shows that our causality function $\mathbf{C}$ appropriately finds cause or effect relation between two time series in regard of their strength and distance.

5.2 Forecasting with Textual Features

Table 5: Forecasting errors (RMSE) on **Stock** and **Poll** data with time series only (*SpikeM* and *LSTM*) and with time series plus text feature (*random*, *words*, *topics*, *sentiment*, and *composition*).

		Time Series		Time Series + Text				
	Step	SpikeM	LSTM	$\mathbf{C}_{rand}$	$\mathbf{C}_{words}$	$\mathbf{C}_{topics}$	$\mathbf{C}_{senti}$	$\mathbf{C}_{comp}$
Stock	1	102.13	6.80	3.63	2.97	3.01	3.34	$\underline{1.96}$
	3	99.8	7.51	4.47	4.22	4.65	4.87	$\underline{3.78}$
	5	97.99	7.79	5.32	$\underline{5.25}$	5.44	5.95	5.28
Poll	1	10.13	1.46	1.52	1.27	1.59	2.09	$\underline{1.11}$
	3	10.63	1.89	1.84	1.56	1.88	1.94	$\underline{1.49}$
	5	11.13	2.04	2.15	1.84	1.88	1.96	$\underline{1.82}$

We use time series forecasting task as an evaluation metric of whether our textual features are appropriately causing the target time series or not. Our feature composition function Φ is used to extract good causal features for forecasting. We test forecasting on stock price of companies (**Stock**) and predicting poll value for presidential election (**Poll**). For stock data, We collect daily closing stock prices during 2013 for ten IT companies[5]. For poll data, we choose ten candidate politicians [6] in the period of presidential election in 2012.

For each of stock and poll data, the future trend of target is predicted only with target's past time series or with target's past time series and past time series of textual features found by our system. Forecasting only with target's past time series uses *SpikeM* (Matsubara et al., 2012) that models a time series with small number of parameters and simple *LSTM* (Hochreiter and Schmidhuber, 1997; nne, 2015) based time series model. Forecasting with target and textual features' time series use Vector AutoRegressive model with exogenous variables (VARX) (Hamilton, 1994) from different composition function such as $\mathbf{C}_{random}$, $\mathbf{C}_{words}$, $\mathbf{C}_{topics}$, $\mathbf{C}_{senti}$, and $\mathbf{C}_{composition}$. Each composition function except $\mathbf{C}_{random}$ uses top ten textual features that causes each target time series. We also tested LSTM with past time series and textual features but VARX outperforms LSTM.

Table 6: Beam search results in neural reasoning. These examples could be filtered out by graph heuristics before generating final explanation though.

Cause$\mapsto$Effect in CGraph	Beam Predictions
the dollar's $\xmapsto{caus}$ against the yen	[1]$\xmapsto{caus}$ against the yen [2]$\xmapsto{caus}$ against the dollar [3]$\xmapsto{caus}$ against other currencies
without any exercise $\xmapsto{caus}$ news article	[1]$\xmapsto{leadto}$ a difference [2]$\xmapsto{caus}$ the risk [3]$\xmapsto{make}$ their weight

Table 5 shows root mean square error (RMSE) for forecasting with different step size (time steps to predict), different set of features, and different regression algorithms on stock and poll data. The forecasting error is summation of errors over moving a window (30 days) by 10 days over the period. Our $\mathbf{C}_{composition}$ method outperforms other time series only models and time series plus text models in both stock and poll data.

5.3 Generating Causality with Neural Reasoner

The reasoner needs to predict the next effect phrase (or previous cause phrase) so the model should be evaluated in terms of generation task. We used the BLEU (Papineni et al., 2002) metric to evaluate the predicted phrases on held out phrases in our CGraph . Since our CGraph has many edges, there may be many good paths (explanations), possibly making our prediction diverse. To evaluate such diversity in prediction, we used ranking-based BLEU method on the k set of

[4] SpikeM has specific parameters for modeling a time series such as peak strength, length, etc.

[5] Company symbols used: TSLA, MSFT, GOOGL, YHOO, FB, IBM, ORCL, AMZN, AAPL and HPO

[6] Name of politicians used: Santorum, Romney, Pual, Perry, Obama, Huntsman, Gingrich, Cain, Bachmann

Table 7: BLEU ranking. Additional word representation **+WE** and relation specific alignment **+REL** help the model learn the cause and effect generation task especially for diverse patterns.

	B@1	B@3A	B@5A
S2S	10.15	8.80	8.69
S2S + WE	11.86	10.78	10.04
S2S + WE + REL	12.42	12.28	11.53

predicted phrases by beam search. For example, $B@k$ means BLEU scores for generating k number of sentences and $B@kA$ means the average of them.

Table 6 shows some examples of our beam search results when $k = 3$. Given a cause phrase, the neural reasoner sometime predicts semantically similar phrases (e.g., *against the yen*, *against the dollar*), while it sometimes predicts very diverse phrases (e.g., *a different*, *the risk*).

Table 7 shows BLEU ranking results with different reasoning algorithms: **S2S** is a sequence to sequence learning trained on CGraph by default, **S2S+WE** adds word embedding initialization, and **S2S+REL+WE** adds relation specific attention. Initializing with pre-trained word embeddings (**+WE**) helps us improve on prediction. Our relation specific attention model outperforms the others, indicating that different type of relations have different alignment patterns.

5.4 Generating Explanation by Connecting

Evaluating whether a sequence of phrases is reasonable as an explanation is very challenging task. Unfortunately, due to lack of quantitative evaluation measures for the task, we conduct a human annotation experiment.

Table 8 shows example causal chains for the rise (↑) and fall (↓) of companies' stock price, continuously produced by two reasoners: *SYBM* is symbolic reasoner and *NEUR* is neural reasoner.

We also conduct a human assessment on the explanation chains produced by the two reasoners, asking people to choose more convincing explanation chains for each feature-target pair. Table 9 shows their relative preferences.

6 Related Work

Prior works on causality detection (Acharya, 2014; Anand, 2014; Qiu et al., 2012) in time series data (e.g., gene sequence, stock prices, temperature) mainly use Granger (Granger, 1988) ability for predicting future values of a time series using past values of its own and another time series. (Hlaváčková-Schindler et al., 2007) studies more theoretical investigation for measuring causal influence in multivariate time series based on the entropy and mutual information estimation. However, none of them attempts generating explanation on the temporal causality.

Previous works on text causality detection use syntactic patterns such as $X \xmapsto{verb} Y$, where the $verb$ is causative (Girju, 2003; Riaz and Girju, 2013; Kozareva, 2012; Do et al., 2011) with additional features (Blanco et al., 2008). (Kozareva, 2012) extracted cause-effect relations, where the pattern for bootstrapping has a form of $X^* \xmapsto[Z^*]{verb} Y$ from which terms X^* and Z^* was learned. The syntax based approaches, however, are not robust to semantic variation.

As a part of SemEval (Girju et al., 2007), (Mirza and Tonelli, 2016) also uses syntactic causative patterns (Mirza and Tonelli, 2014) and supervised classifier to achieve the state-of-the-art performance. Extracting the cause-effect tuples with such syntactic features or temporality (Bethard et al., 2008) would be our next step for better causal graph construction.

(Grivaz, 2010) conducts very insightful annotation study of what features are used in human reasoning on causation. Beyond the linguistic tests and causal chains for explaining causality in our work, other features such as counterfactuality, temporal order, and ontological asymmetry remain as our future direction to study.

Textual entailment also seeks a directional relation between two given text fragments (Dagan et al., 2006). Recently, (Rocktäschel et al., 2015) developed an attention-based neural network method, trained on large annotated pairs of textual entailment, for classifying the types of relations with decomposable attention (Parikh et al., 2016) or sequential tree structure (Chen et al., 2016). However, the dataset (Bowman et al., 2015) used for training entailment deals with just three categories, *contradiction*, *neutral*, and *entailment*, and focuses on relatively simple lexical and syntactic transformations (Kolesnyk et al., 2016). Our causal explanation generation task is also similar to *future scenario generation* (Hashimoto et al., 2014, 2015). Their scoring

Table 8: Example causal chains for explaining the rise (↑) and fall (↓) of companies' stock price. The temporally causal *feature* and *target* are linked through a sequence of predicted cause-effect tuples by different reasoning algorithms: a symbolic graph traverse algorithm *SYMB* and a neural causality reasoning model *NEUR*.

SYMB	**medals** $\xmapsto{match}$ gold_and_silver_medals $\xmapsto{swept}$ korea $\xmapsto{improving}$ relations $\xmapsto{widened}$ gap $\xmapsto{widens}$ **facebook** ↑
	excess $\xmapsto{match}$ excess_materialism $\xmapsto{cause}$ people_make_films $\xmapsto{make}$ money $\xmapsto{changed}$ twitter $\xmapsto{turned}$ **facebook** ↓
	clinton $\xmapsto{match}$ president_clinton $\xmapsto{raised}$ antitrust_case $\xmapsto{match}$ government's_antitrust_case_against_microsoft $\xmapsto{match}$ microsoft $\xmapsto{beats}$ **apple** ↓
NEUR	**google** $\xmapsto{forc}$ microsoft_to_buy_computer_company_dell_announces_recall_of_batteries $\xmapsto{cause}$ **microsoft** ↑
	the_deal $\xmapsto{make}$ money $\xmapsto{rais}$ at_warner_music_and_google_with_protest_videos_things $\xmapsto{caus}$ **google** ↓
	party $\xmapsto{cut}$ budget_cuts $\xmapsto{lower}$ budget_bill $\xmapsto{decreas}$ republicans $\xmapsto{caus}$ obama $\xmapsto{leadto}$ facebook_polls $\xmapsto{caus}$ **facebook** ↓
	company $\xmapsto{forc}$ to_stock_price $\xmapsto{leadto}$ investors $\xmapsto{increas}$ oracle_s_stock $\xmapsto{increas}$ **oracle** ↑

Table 9: Human evaluation on explanation chains generated by symbolic and neural reasoners.

Reasoners	SYMB	NEUR
Accuracy (%)	42.5	57.5

function uses heuristic filters and is not robust to lexical variation.

7 Conclusion

This paper defines the novel task of detecting and explaining causes from text for a time series. First, we detect causal features from online text. Then, we construct a large cause-effect graph using FrameNet semantics. By training our relation specific neural network on paths from this graph, our model generates causality with richer lexical variation. We could produce a chain of cause and effect pairs as an explanation which shows some appropriateness. Incorporating aspects such as time, location and other event properties remains a point for future work. In our following work, we collect a sequence of causal chains verified by domain experts for more solid evaluation of generating explanations.

References

2015. Neural network architecture for time series forecasting. https://github.com/hawk31/nnet-ts.

Saurav Acharya. 2014. Causal modeling and prediction over event streams.

Surya Pratap Singh Tanwar Anand, Mehndiratta. 2014. Web Metric Summarization using Causal Relationship Graph.

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

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

Steven Bethard, William J Corvey, Sara Klingenstein, and James H Martin. 2008. Building a corpus of temporal-causal structure. In *LREC*.

Eduardo Blanco, Nuria Castell, and Dan I Moldovan. 2008. Causal relation extraction. In *LREC*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *JMLR*.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Desai Chen, Nathan Schneider, Dipanjan Das, and Noah A Smith. 2010. Semafor: Frame argument resolution with log-linear models. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 264–267. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Enhancing and combining sequential and tree lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Quang Xuan Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 294–303. Association for Computational Linguistics.

Roxana Girju. 2003. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering-Volume 12*, pages 76–83. Association for Computational Linguistics.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. Semeval-2007 task 04: Classification of semantic relations between nominals. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 13–18. Association for Computational Linguistics.

Google. 2016. Freebase Data Dumps. https://developers.google.com/freebase/data.

Clive WJ Granger. 1988. Some recent development in a concept of causality. *Journal of econometrics*, 39(1):199–211.

Cécile Grivaz. 2010. Human judgements on causation in french texts. In *LREC*.

James Douglas Hamilton. 1994. *Time series analysis*, volume 2. Princeton university press Princeton.

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, and Jong-Hoon Oh. 2015. Generating event causality hypotheses through semantic relations. In *AAAI*, pages 2396–2403.

Chikara Hashimoto, Kentaro Torisawa, Julien Kloetzer, Motoki Sano, István Varga, Jong-Hoon Oh, and Yutaka Kidawara. 2014. Toward future scenario generation: Extracting event causality exploiting semantic relation, context, and association features. In *ACL (1)*, pages 987–997.

Katerina Hlaváčková-Schindler, Milan Paluš, Martin Vejmelka, and Joydeep Bhattacharya. 2007. Causality detection based on information-theoretic approaches in time series analysis. *Physics Reports*, 441(1):1–46.

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

Patrik O Hoyer. 2004. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469.

Vladyslav Kolesnyk, Tim Rocktäschel, and Sebastian Riedel. 2016. Generating natural language inference chains. *arXiv preprint arXiv:1606.01404*.

Zornitsa Kozareva. 2012. Cause-effect relation learning. In *Workshop Proceedings of TextGraphs-7 on Graph-based Methods for Natural Language Processing*, pages 39–43. Association for Computational Linguistics.

Yasuko Matsubara, Yasushi Sakurai, B. Aditya Prakash, Lei Li, and Christos Faloutsos. 2012. Rise and fall patterns of information diffusion: model and implications. In *KDD*, pages 6–14.

T Mikolov and J Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*.

Paramita Mirza and Sara Tonelli. 2014. An analysis of causality between events and its relation to temporal information. In *COLING*, pages 2097–2106.

Paramita Mirza and Sara Tonelli. 2016. Catena: Causal and temporal relation extraction from natural language texts. In *The 26th International Conference on Computational Linguistics*, pages 64–75.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Huida Qiu, Yan Liu, Niranjan A Subrahmanya, and Weichang Li. 2012. Granger causality for time-series anomaly detection. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1074–1079. IEEE.

Mehwish Riaz and Roxana Girju. 2013. Toward a better understanding of causality between verbal events: Extraction and analysis of the causal power of verb-verb associations. In *Proceedings of the annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*. Citeseer.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Rebecca Sharp, Mihai Surdeanu, Peter Jansen, Peter Clark, and Michael Hammond. 2016. Creating causal embeddings for question answering with minimal supervision. *arXiv preprint arXiv:1609.08097*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

A Novel Cascade Model for Learning Latent Similarity from Heterogeneous Sequential Data of MOOC

Zhuoxuan Jiang[1] and **Shanshan Feng**[2] and **Gao Cong**[2] and **Chunyan Miao**[3] and **Xiaoming Li**[1]
[1]School of Electronics Engineering and Computer Science, Peking University, China
[2]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[3]Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY),
Nanyang Technological University, Singapore
{jzhx,lxm}@pku.edu.cn, {sfeng003@e.,gaocong@,ascymiao@}ntu.edu.sg

Abstract

Recent years have witnessed the proliferation of Massive Open Online Courses (MOOCs). With massive learners being offered MOOCs, there is a demand that the forum contents within MOOCs need to be classified in order to facilitate both learners and instructors. Therefore we investigate a significant application, which is to associate forum threads to subtitles of video clips. This task can be regarded as a document ranking problem, and the key is how to learn a distinguishable text representation from word sequences and learners' behavior sequences. In this paper, we propose a novel cascade model, which can capture both the latent semantics and latent similarity by modeling MOOC data. Experimental results on two real-world datasets demonstrate that our textual representation outperforms state-of-the-art unsupervised counterparts for the application.

1 Introduction

With the rapid development of Massive Open Online Courses (MOOCs), more and more learners participate in MOOCs (Anderson et al., 2014). Due to the lack of effective management, most of the discussion forums within MOOCs are overloaded and in chaos (Huang et al., 2014). Therefore, a key problem is how to manage the forum contents.

To manage the forum contents, threads of forums can be regarded as documents and be classified to groups. There are several straightforward methods, such as defining sub-forums according to weeks and asking learners to tag threads. However their effectiveness is limited (Rossi and Gnawali, 2014), because learners have few incentives to tag threads. Recently, machine learning solutions have been proposed, e.g., content-related thread identification (Wise et al., 2016), confusion classification (Agrawal et al., 2015) and sentiment classification (Ramesh et al., 2015). However they are developed for specific research problems and cannot be applied to our problem. Moreover, they require labeling data which needs domain experts to label data for different courses.

We observe that the video clips of a MOOC would have many well-formed subtitles composed by instructors. Moreover, within MOOC settings, the course contents can be broken down to knowledge points, and each video clip just corresponds to a knowledge point. Consequently, we propose to fulfill the application, which is to associate threads to subtitles of video clips, i.e., thread-subtitle matching. By this way, the relevant videos to the threads can be recommended to learners, and the chaotic threads in discussion forums can also be well grouped.

However, it is challenging to identify the relevant video clips for threads without labeling data. To address this issue, we regard it as a document ranking problem based on the calculation of similarity between documents. The key problem of this task is to learn a textual representation, which can cluster similar documents and meanwhile distinguish irrelevant ones.

Intuitively, Bag-of-words model (BOW) can be utilized to calculate the similarity between threads and subtitles (Salton and Buckley, 1988). However, BOW cannot effectively capture semantics of words and documents. In addition, recently-studied semantic word embeddings, e.g., Word2Vec (Mikolov et al., 2013), can capture the semantics. Para2Vec (Le and Mikolov, 2014) can capture the similarity to some degree, but not explicitly model the latent similarity of documents.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2768–2773
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

Since the latent similarity is crucial to determine whether a document can be associated to the right target, in our task, the document representation is expected to preserve both the latent semantics and similarity.

In this paper, we leverage two kinds of sequential information: 1) word sequence of subtitles and forum contents, and 2) clickstream log of learning behaviors. Specifically, different from conventional representation learning tasks, e.g. Word2Vec and Para2Vec, we consider the clickstream data, which reflects the relationship between thread and video's subtitle. For instance, if a user watches a video and then clicks a thread in forums, the video would be relevant to the thread. In order to learn representations from the two types of data, we propose a novel cascade model.

Our basic idea is to jointly model three components: 1) word-word coherence, 2) document-document coherence, and 3) word-document coherence. The three components are cascaded for learning the low-dimensional word embeddings. Then the learned embeddings are used to calculate similarities between threads and subtitles.

To summarize, our contributions include:

- We study an application-oriented research problem, which is how to capture the latent similarity when learning text representation.
- We propose a novel cascade model to learn the document representation from heterogeneous sequential data: 1) word sequence and 2) learners' clickstream.
- We collect two real-world MOOC datasets and conduct thorough experiments. The results demonstrate that our proposed model outperforms the state-of-the-art unsupervised counterparts on the application.

2 Related Work

MOOC data has attracted extensive research attention and many interesting research problems have been studied. For example, dropout predicting (Qiu et al., 2016), sentiment analysis of learning gains (Ramesh et al., 2015), instructor intervention (Chaturvedi et al., 2014) and answer recommendation (Jenders et al., 2016), etc. Particularly, (Agrawal et al., 2015) considers a similar task as ours, which is to recommend video clips to threads. But its solution is designed for the specific task and needs labeling data. Our solution is an unsupervised learning method and the learned embeddings have other applications, e.g. thread retrieval.

How to represent text is a fundamental research problem in the field of information retrieval. Existing approaches can be generally classified into unsupervised methods and supervised methods (Tang et al., 2015). Although supervised embeddings can obtain good performance in specific tasks, such as using deep neural network (Mikolov et al., 2010; Kim, 2014), they need human efforts to get labels. Unsupervised word embeddings usually leverage various levels of textual information. For example, Word2Vec learns word embeddings based on word coherence. Para2Vec utilizes word and document coherence to learn their embeddings. Particularly, Hierarchical Document Vector (HDV) (Djuric et al., 2015) leverages both streaming documents and their contents to achieve better representation, which is similar to our proposed model. However, HDV regards the documents as the context of words, which cannot learn the latent similarity, since it fails to explicitly reflect the relationship between document and word. In order to model the heterogeneous MOOC data, we develop a cascade representation model. To our knowledge, (Jiang et al., 2017) also proposes an unsupervised learning model (called NOSE) for the task of thread-subtitle matching within MOOC settings. However, NOSE needs to build a heterogeneous textual network beforehand and may suffer from heterogeneous issue, which our model can avoid.

Recently, representation learning has been applied to many tasks, such as network embedding (Grover and Leskovec, 2016) and location embedding (Feng et al., 2017). In this paper, we focus on learning representation of words and documents in MOOCs.

3 Cascade Model

Based on our observation, we utilize two kinds of sequential information: 1) word sequences of subtitles and threads, and 2) clickstream of learning behaviors. In this paper, we regard the subtitles or threads consistently as *documents*. Particularly, we discover that the log of learners' clickstream, i.e., the click records of watching videos, reading threads and posting threads in a chronological order, can reflect the document-level latent semantics. An intuitive explanation is that a learner who jumps from videos to threads may look for further relevant information from forums when s/he is watching a video, or s/he wants to review the relevant videos when s/he reads a thread.

However, learning from the log of clickstream data merely guarantees that similar documents are close enough in the embedding space, while different documents cannot be scattered. To address this issue, we attempt to strengthen the relationship between words and their affiliated documents. Thus, words within the same documents would be gathered and otherwise scattered in the embedding space. Consequently, the latent similarity can be embodied by word embeddings.

Based on the aforementioned idea, we can model the data by three components: 1) latent semantics at word level, 2) latent similarity at document level, and 3) latent similarity between words and documents. To integrate all the three kinds of information into a uniform learning framework, we propose a novel cascade model, as shown in Fig. 1. $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ correspond to the log-likelihood of three components respectively. Formally, we aim at minimizing the log-likelihood function:

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 \quad (1)$$

Note that $\mathcal{L}_3$ not only learns the latent similarity, but also builds a connection between words and documents. In this way, our learned word embeddings can be adopted to our task without learning classifiers by labeling data.

3.1 Word-level Latent Semantics

As to the part of $\mathcal{L}_1$, corresponding to the red/bottom part of Fig. 1, we leverage the Word2Vec model to learn the semantics of words. In this paper, we take the Continuous bag-of-words (CBOW) architecture. The objective function is to minimize the log-likelihood:

$$\mathcal{L}_1 = -\sum_{t=1}^{T} \log \mathbb{P}(w_t | w_{t-c_w} : w_{t+c_w}) \quad (2)$$

where c_w is the context window length used in word sequence, and $w_{t-c_w} : w_{t+c_w}$ is the subsequence $(w_{t-c_w}, \ldots, w_{t+c_w})$ excluding w_t itself. The probability $\mathbb{P}(w_t | w_{t-c_w} : w_{t+c_w})$ is defined by the softmax function $\frac{\exp(\bar{\mathbf{v}}^T \mathbf{v}_{w_t})}{\sum_{w=1}^{W} \exp(\bar{\mathbf{v}}^T \mathbf{v}_w)}$, where $\mathbf{v}_{w_t}$ is the vector representation of word w_t, and $\bar{\mathbf{v}}$ is averaged vector representation of the subsequence. Two methods can be employed to calculating $\mathcal{L}_1$: hierarchical softmax and negative sampling (Mikolov et al., 2013).

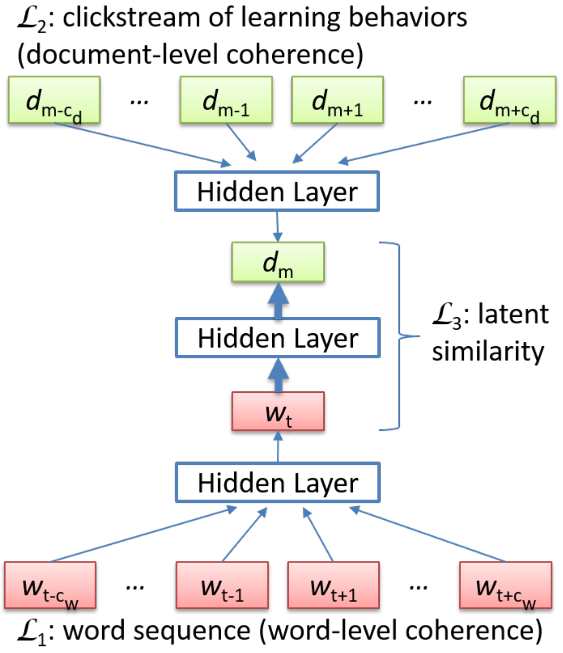

$\mathcal{L}_2$: clickstream of learning behaviors (document-level coherence)
d_{m-c_d}
...
d_{m-1}
d_{m+1}
...
d_{m+c_d}
Hidden Layer
d_m
Hidden Layer
w_t
$\mathcal{L}_3$: latent similarity
Hidden Layer
w_{t-c_w}
...
w_{t-1}
w_{t+1}
...
w_{t+c_w}
$\mathcal{L}_1$: word sequence (word-level coherence)

Figure 1: The architecture of proposed model which is cascaded by three parts: $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$.

3.2 Document-level Latent Similarity

Similar to $\mathcal{L}_1$, we adopt the CBOW architecture for calculating $\mathcal{L}_2$, as shown by the green/top part of Fig. 1. The objective function is to minimize the log-likelihood:

$$\mathcal{L}_2 = -\sum_{m=1}^{M} \log \mathbb{P}(d_m | d_{m-c_d} : d_{m+c_d}) \quad (3)$$

where M is the number of documents, c_d is the context window length used in clickstreams, and $d_{m-c_d} : d_{m+c_d}$ is the sub-sequence $(d_{m-c_d}, \ldots, d_{m+c_d})$ excluding d_m itself. The probability $\mathbb{P}(d_m | d_{m-c_d} : d_{m+c_d})$ is also the softmax function. Methods of hierarchical softmax and negative sampling can be employed to approximate the log-likelihood function.

3.3 Document-Word Latent Similarity

To learn the latent similarity, we make use of the relationship between words and documents, and then similar documents can be clustered, while different documents are scattered. Therefore, we propose the third component, $\mathcal{L}_3$, shown in the middle part of Fig. 1. Different from $\mathcal{L}_1$ and $\mathcal{L}_2$, we employ negative sampling of documents to calculate its approximation, because there are numerous threads in MOOC forums. Given a pair (w_t, d_m), representing that word w_t appears in document d_m, $\mathcal{L}_3$ is denoted as:

$$\mathcal{L}_3 = \sum_{w_t} \left(-\log \sigma(\mathbf{v}_{d_m}^T \mathbf{v}_{w_t}) + \sum_{c=1}^{C} \log \sigma(\mathbf{v}_{d_c}^T \mathbf{v}_{w_t}) \right) \tag{4}$$

where $\sigma(x)$ is the sigmoid function and C is the number of sampled negative documents.

3.4 Model Training

We adopt stochastic gradient descent (SGD) to minimize $\mathcal{L}$. As to the components of $\mathcal{L}_1$ and $\mathcal{L}_2$, we exploit the training methods proposed in (Mikolov et al., 2013) to the two kinds of sequences, i.e., words and documents, respectively. For training $\mathcal{L}_3$, given the pair (w_t, d_m), we calculate the gradients:

$$\frac{\partial \mathcal{L}_3}{\partial \mathbf{v}_{d_j}} = \Big(\sigma(\mathbf{v}_{d_j}^T \mathbf{v}_{w_t}) - \nVdash(j = m) \Big) \mathbf{v}_{w_t}, \tag{5}$$

$$\frac{\partial \mathcal{L}_3}{\partial \mathbf{v}_{w_t}} = \Big(\sigma(\mathbf{v}_{d_j}^T \mathbf{v}_{w_t}) - \nVdash(j = m) \Big) \mathbf{v}_{d_j}, \tag{6}$$

where d_j represents both the positive and negative samples, as $d_j \in \{d_i\} \cup \{d_c \sim P_n(w) | c = 1, \ldots, C\}$. $P_n(w)$ is the noise distribution and we set it as unigram distribution raised to 3/4th power, which is the same as Word2Vec. $\nVdash(x)$ is an indicator function defined as:

$$\nVdash(x) = \begin{cases} 1 & \text{if } x \text{ is true,} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

The time complexity of updating $\mathcal{L}$ is $O(T \log T + M \log M + TC)$ when using hierarchical softmax method for $\mathcal{L}_1$ and $\mathcal{L}_2$, or $O((2T + M)C)$ when using negative sampling method. Based on the complexity analysis, our cascade model is efficient enough and can be applied to MOOC datasets.

4 Experiment

Data Sets We collect the sequential data of two MOOCs from Coursera[1] and China University MOOC[2] respectively. The former is an interdiscipline course called *People and Network*, and the second is called *Introduction to MOOC*. From both courses, we collect subtitles of video clips, forum contents and learners' log of clickstream. Table 1 shows the statistical information of the two MOOCs.

For evaluation, we invite the teaching assistants (TAs) of respective courses to label test samples in advance. Note that our model is unsupervised. Therefore, labeled data (thread-subtitle matching pairs) are only used for evaluation, and we do not utilize dev dataset.

Experimental Setting We compare our embeddings with unsupervised rivals and the labels are only used for evaluation. To ensure fair comparison, we represent documents with their averaged word embeddings. Note that in the training phase, we represent each thread/subtitle with a vector, in order to make the words within a document clustered and close to each other. We evaluate the following methods.

- Bag-of-words(BOW): the classical text representation.

- Word2Vec: word embeddings which leverages word-level coherence and we adopt the CBOW architecture.

- Para2Vec: paragraph embeddings which considers document-level context information. We also adopt CBOW framework.

- Hierarchical Document Vector(HDV): the latest word embeddings with a hierarchical architecture for modeling streaming documents and their contents.

- Cascade Document Representation (CDR): our proposed model which captures both the latent semantics and latent similarity.

We use the hype-parameters recommended by previous literatures. For all the evaluated baselines, we use the same parameter setting. Thus it is fair to make comparison. The window size set in all baselines is 5 by default. The number of negative samples is empirically set as 5. The size of hidden layer is set as 100 for all the methods. We utilize the Precision@K (denoted by P@K) as metric. If the retrieved top-K subtitles hit at least one ground-truth label, we regard it as true; otherwise, it is false. In our experiments, we run 10 times and report the average result for each case.

[1] https://www.coursera.org, which is an educational technology company that offers MOOCs worldwide.

[2] http://www.icourse163.org, which is a leading MOOCs platform in China.

Course Name	#active users	#video clips	#threads	#posts	#words	#clicks
People and Network	10,807	60	219	1,206	121,142	31,096
Introduction to MOOC	3,949	19	557	7,177	480,495	45,642

Table 1: Statistics of two MOOC datasets.

Model	*People and Network*			*Introduction to MOOC*		
	P@1	P@3	P@5	P@1	P@3	P@5
BOW	0.437	**0.718**	0.806	0.449	0.811	0.909
Word2Vec	0.485	0.699	**0.816**	0.453	0.826	0.890
Para2Vec	0.408	0.612	0.728	0.504	0.823	0.894
HDV	0.466	0.621	0.777	0.496	0.819	0.913
CDR	**0.505**	0.689	0.786	**0.520**	**0.854**	**0.941**

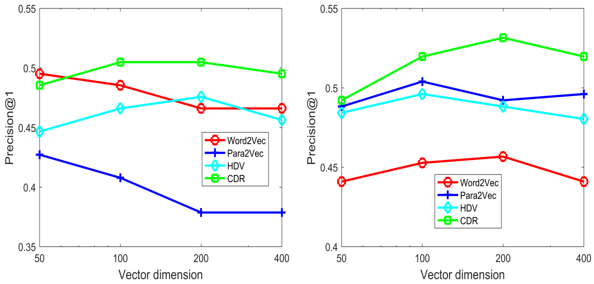
Table 2: Result of thread-subtitle matching.

Result Firstly we use all the data to learn word embeddings by models. Then the learned word vectors are utilized to calculate the similarity between threads and subtitles, and rank the subtitles. Table 2 reports the results of thread-subtitle matching. We can notice that there are some anomalies in P@3 and P@5 results. It may be for the reason of dataset. In the first MOOC (people and network), video subtitles contain relatively less words, and therefore it is hard to get effective representations. Overall, the proposed models can achieve better performance than baselines, and we highlight the Precision@1 results. Compared to HDV which also considers the streaming documents, our model is better at every task. This indicates our model can effectively capture the latent similarity.

We investigate the effect of number of dimensions, i.e., the size of the neural network's hidden layer. From Fig.2, we find that CDR can achieve better performance than baselines with various numbers of dimensions. In addition, the optimal results can be obtained when the dimension is set as 100 or 200 in both datasets.

5 Conclusion

In this paper, we propose an approach to solve a significant problem: how to learn distinguishable representations from word sequences in documents and clickstreams of learners. To model the heterogeneous data, we develop a cascade model which can jointly learn the latent semantics and latent similarity without labeling data. We conduct experiments on two real datasets, and the results demonstrate the effectiveness of our model.



(a) People and Network (b) Introduction to MOOC

Figure 2: P@1 of different dimensions.

Moreover, our model is not limited to MOOC data. For instance, we can adopt the proposed algorithm to streaming documents, e.g. webpage click streams, since our method can model the document-document sequences. We leave this as the future work.

Acknowledgments

This research is supported by the National Research Foundation, Prime Ministers Office, Singapore under its IDM Futures Funding Initiative, China NSFC with Grant No.61532001 and No.61472013, and China MOE-RCOE with Grant No.2016ZD201. Xiaoming Li is the corresponding author. We thank the anonymous reviewers for their insightful comments.

References

Akshay Agrawal, Jagadish Venkatraman, Shane Leonard, and Andreas Paepcke. 2015. Youedu: Addressing confusion in mooc discussion forums by recommending instructional video clips. In *EDM*, pages 297–304.

Ashton Anderson, Daniel P. Huttenlocher, Jon M. Kleinberg, and Jure Leskovec. 2014. Engaging with massive online courses. In *WWW*, pages 687–698.

Snigdha Chaturvedi, Dan Goldwasser, and Hal Daumé III. 2014. Predicting instructors intervention in mooc forums. In *ACL*, pages 1501–1511.

Nemanja Djuric, Hao Wu, Vladan Radosavljevic, Mihajlo Grbovic, and Narayan Bhamidipati. 2015. Hierarchical neural language models for joint representation of streaming documents and their content. In *WWW*, pages 248–255.

Shanshan Feng, Gao Cong, Bo An, and Yeow Meng Chee. 2017. Poi2vec: Geographical latent representation for predicting future visitors. In *AAAI*, pages 102–108.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864.

Jonathan Huang, Anirban Dasgupta, Arpita Ghosh, Jane Manning, and Marc Sanders. 2014. Superposter behavior in mooc forums. In *L@S*, pages 117–126.

Maximilian Jenders, Ralf Krestel, and Felix Naumann. 2016. Which answer is best?: Predicting accepted answers in mooc forums. In *WWW*, pages 679–684.

Zhuoxuan Jiang, Shanshan Feng, Weizheng Chen, Guangtao Wang, and Xiaoming Li. 2017. Unsupervised embedding for latent similarity by modeling heterogeneous mooc data. In *PAKDD*, pages 683–695.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

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

Tomas Mikolov, Martin Karafit, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

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

Jiezhong Qiu, Jie Tang, Tracy Xiao Liu, Jie Gong, Chenhui Zhang, Qian Zhang, and Yufei Xue. 2016. Modeling and predicting learning behavior in moocs. In *WSDM*, pages 93–102.

Arti Ramesh, Shachi H. Kumar, James R. Foulds, and Lise Getoor. 2015. Weakly supervised models of aspect-sentiment for online course discussion forums. In *ACL&IJCNLP*, pages 74–83.

Lorenzo A. Rossi and Omprakash Gnawali. 2014. Language independent analysis and classification of discussion threads in coursera mooc forums. In *IRI*, pages 654–661.

Gerard Salton and Chris Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523.

Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*, pages 1165–1174.

Alyssa Friend Wise, Yi Cui, and Jovita Vytasek. 2016. Bringing order to chaos in mooc discussion forums with content-related thread identification. In *LAK*, pages 188–197.

Identifying the Provision of Choices in Privacy Policy Text

Kanthashree Mysore Sathyendra
School of Computer Science
Carnegie Mellon University
ksathyen@andrew.cmu.edu

Shomir Wilson
EECS Department
University of Cincinnati
shomir.wilson@uc.edu

Florian Schaub
School of Information
University of Michigan
fschaub@umich.edu

Sebastian Zimmeck
School of Computer Science
Carnegie Mellon University
szimmeck@andrew.cmu.edu

Norman Sadeh
School of Computer Science
Carnegie Mellon University
sadeh@cs.cmu.edu

Abstract

Websites' and mobile apps' privacy policies, written in natural language, tend to be long and difficult to understand. Information privacy revolves around the fundamental principle of *notice and choice*, namely the idea that users should be able to make informed decisions about what information about them can be collected and how it can be used. Internet users want control over their privacy, but their choices are often hidden in long and convoluted privacy policy documents. Moreover, little (if any) prior work has been done to detect the provision of choices in text. We address this challenge of enabling user choice by automatically identifying and extracting pertinent choice language in privacy policies. In particular, we present a two-stage architecture of classification models to identify opt-out choices in privacy policy text, labelling common varieties of choices with a mean F1 score of 0.735. Our techniques enable the creation of systems to help Internet users to learn about their choices, thereby effectuating notice and choice and improving Internet privacy.

1 Introduction

Website privacy policies are long, verbose documents that are often difficult to understand. It has been shown that an average Internet user would require an impractical amount of time to read the privacy policies of online services that they use and would not properly understand them (McDonald and Cranor, 2008). Although Internet users are concerned about their privacy and would like to be informed about the privacy controls they can exercise, they are not willing or able to find these choices in policy text. Choices for privacy controls, which are the most actionable pieces of information in these documents, are frequently "hidden in plain sight" among other information. However, the nature of the text and the vocabulary used to present choices provide us with an opportunity to automatically identify choices, a goal that we focus upon in this paper.

We define a *choice instance* as a statement in a privacy policy that indicates that the user has discretion over aspects of their privacy. An example (which notably features a hyperlink) is the following:

> If you would like more information on how to opt out of information collection practices, go to `www.aboutads.info`.[1]

Some examples of choices offered to users include opt-outs or controls for the sharing of personal information with third parties, receiving targeted ads, or receiving promotional emails. Analyzing these choice instances in aggregate will help to understand how notice and choice is implemented in practice, which is of interest to legal scholars, policy makers and regulators. Furthermore, extracted choice options can be presented to users in more concise and usable notice formats (Schaub et al., 2015), such as a browser plug-in or a privacy based question answering system.

For this paper, we treat the identification of choice instances as a binary classification problem, in which we label each sentence in the privacy policy text as containing a choice instance or not. We use the OPP-115 Corpus (Wilson et al., 2016) for training and evaluation of our models.

[1] `http://www.nurse.com/privacy/` (last updated on July 13, 2015)

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2774–2779
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

We further annotate a second dataset[2] and develop a composite model architecture to automatically identify and label different types of opt-out choices offered in privacy policies. We primarily focus on extracting opt-out instances with hyperlinks because these are some the most common and useful choices described in privacy policies. Moreover, these choice expressions are actionable: the first step of the action to be taken (i.e., following a hyperlink) is clearly represented in the text of these instances.

The work presented in this paper has been conducted in the context of the 'Usable Privacy Policy' project, which combines crowdsourcing, machine learning and natural language processing to overcome the limitations of today's approach to 'notice and choice' in privacy (Sadeh et al., 2013).

2 Related Work

The Federal Trade Commission identifies "Notice and Choice" as one of the core principles of information privacy protection under the Fair Information Practice Principles (Federal Trade Commission, 2000). However, privacy policies, being long, complicated documents full of legal jargon, are sub-optimal for communicating information to individuals (Cranor, 2012; Cate, 2010; Schaub et al., 2015; Reidenberg et al., 2015). Antón et al. (2002) conducted a study in which they identified multiple privacy-related goals in accordance with Fair Information Practices, which included 'Choice/Consent' as one of the protection goals.

The potential for the application of NLP and information retrieval techniques to legal documents has been recognized by law practitioners (Mahler, 2015), with multiple efforts applying NLP techniques to legal documents. Bach et al. (2013) use a multi-layer sequence learning model and integer linear programming to learn logical structures of paragraphs in legal articles. Galgani et al. (2012) present a hybrid approach to summarization of legal documents, based on creating rules to combine different types of statistical information about text. Early work on automatically extracting annotations from privacy policies includes that of Ammar et al. (2012). Montemagni et al. (2010) investigate the peculiarities of the language in legal text with respect to that in ordinary text by applying shallow parsing. Ramanath et al. (2014) introduce an unsupervised model for the automatic alignment of privacy policies and show that Hidden Markov Models are more effective than clustering and topic models. Liu et al. (2016a) modelled the language of vagueness in privacy policies using deep neural networks.

Many of these efforts consider legal documents as a whole, and they focus less on identifying specific attributes of data practices such as choices. We focus on choices in the present work because of their potential to present Internet users with engaging, directly actionable information.

3 Approach

We used the OPP-115 Corpus to train and evaluate our models for identifying opt-out choices. The corpus consists of 115 website privacy policies and annotations (created by law students) for *data practices* that appear in them. A data practice is a statement about how a website user's personal information is collected, processed or shared. Each data practice consists of a selection of a category (i.e., a theme associated with the practice, such as "First Party Collection/Use"), a set of values for attributes specific to the category, and text spans from the policy associated with the value selections (Wilson et al., 2016). The attributes representing choice instances are present in multiple categories of data practices, namely "First Party Collection/Use," "Third Party Sharing/Collection," "User Access, Edit and Deletion," "Policy Change," and "User Choice/Control." The dataset contains annotations for different types of user choice instances, namely "opt-in," "opt-out," "opt-out link," "opt-out via contacting company," "deactivate account," "delete account (full)," and "delete account (partial)."

3.1 Dataset Refinement

We treated the problem of extracting choice instances as a binary classification problem where we labeled sentences from a privacy policy as containing a choice instance (positive) or not (negative). We focused specifically on opt-out choices, as they are among the most common choices offered to Internet users and because opting out is notoriously difficult for users (Leon et al., 2012). All sentences that contained an opt-out user choice (as specified by the OPP-115 annotations) were considered positive, and the rest were considered negative. This resulted in a gold standard set of

[2] Available for download at https://www.usableprivacy.org/data

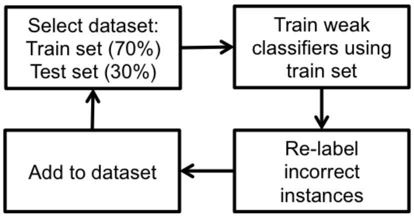

Select dataset:
Train set (70%)
Test set (30%)
Train weak classifiers using train set
Re-label incorrect instances
Add to dataset

Figure 1: Active learning with relabelling.

labeled sentences with 251 positive instances and approximately 12K negative instances.

Differences between our problem formulation and the OPP-115 annotation scheme led to the need for a few label adjustments. Opt-out text spans which crossed sentence boundaries resulted in positive labels for all involved sentences, although often only one of the sentences in a span was positive. Additionally, during the OPP-115 annotation procedure, the fact that hyperlinks were not shown to annotators meant that some choice instances were not correctly identified. This resulted in noisy labels in our derived dataset.

The unbalanced distribution of the opt-out labels allowed us to manually verify and correct labels in the positive class. However, correcting errors in the much larger negative class (of 12K instances) was a challenge, since comprehensive manual verification was infeasible. Instead, we adopted a semi-automated, iterative relabelling approach with active learning. We randomly divided the dataset into train (70%) and test (30%) sets. We trained a binary logistic regression classifier using bag of n-gram features on the training data, and then used it to classify the test data. This was essentially a weak classifier, since it was trained on noisy (unverified) data. We manually examined the false positives and false negatives as given by this model and relabelled incorrectly labelled instances, thus reducing noise in the dataset. Performing multiple iterations of this approach, each time with a different train and test set, resulted in a much cleaner dataset (Figure 1). Following this refinement, the model F1 scores improved and were also more accurate. For all our experiments thereon, we used this refined version of the dataset for training and evaluation.

3.2 Coarse-Grained Classification

We divided the dataset into train and test sets of 85 and 30 privacy policies, respectively. We experimented with a variety of features for *coarse-grained classification*, to separate positive and negative instances:

Stemmed Unigrams and Bigrams. We removed most stop words from the feature set, although some were retained for the modal verb and opt-out features (described below). Bigrams are important to capture pertinent phrases such as "opt out."

Relative Location in the Document. This was a ratio between the number of sentences appearing before the sentence instance and the total number of sentences in the privacy policy.

Topic Model Features. We represented the OPP-115 segment (roughly, a paragraph) containing the sentence instance as a topic distribution vector using latent Dirichlet allocation (Blei et al., 2003) and non-negative matrix factorization (Xu et al., 2003) with 8 and 10 topics, respectively. Previous work on vocabulary intersections of expert annotations and topic models for data practices in privacy policies (Liu et al., 2016b) inspired us to take this approach.

Modal Verbs and Opt-Out specific phrases. We observed vocabulary cues in positive instances that suggested a domain-independent "vocabulary of choice". Many positive instances were imperative sentences and contained modal words such as *may*, *might*, or *can*. We also identified key phrases in the training set such as *unsubscribe* and *opt-out* that were indicative of opt-out choices.

Syntactic Parse Tree Features. We obtained constituency parse trees for sentences using the Stanford Parser (Manning et al., 2014) and extracted production rules and non-terminals as features. We included the maximum depth and average depth of the parse tree as features, as these are indications of specificity.

We used logistic regression classification for the coarse-grained classification stage. Model hyperparameters were tuned based on 5-fold cross validation on the training set. The final parameters for the best performing model had the inverse L2 regularization constant set at C=1.3 and class-weights of 1.5 and 1 for positive and negative class, respectively.

3.3 Fine-Grained Classification

We also developed a *fine-grained* model to differentiate between varieties of opt-out instances. For training data, we annotated a set of 125 positive instances to assign two additional labels to each of them; these were *Party Offering Choice* and *Purpose*. Party Offering Choice could be one of *First*

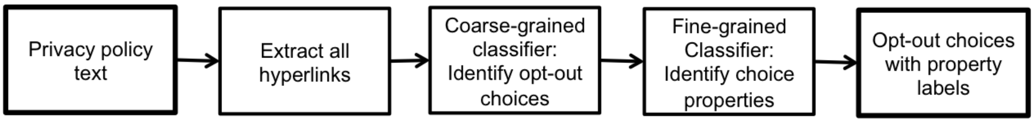

Privacy policy text
Extract all hyperlinks
Coarse-grained classifier: Identify opt-out choices
Fine-grained Classifier: Identify choice properties
Opt-out choices with property labels

Figure 2: Two-Tier Classification Model.

Annotation	# Instances
TH,AD	52
FI,CM	19
FI,AD	15
FI,SH	6
TH,AN	4
BR,CK	2
TH,SH	2
FI,CK	1
TH,CK	1

Table 1: Distribution of different annotation types.

Party (FI), *Third Party*, (TH), or *Browser* (BR). Purpose could be one of *Advertisement* (AD), *Data Sharing* (DS), *Communications* (CM), *Analytics* (AN) or *Cookies* (CK). Table 1 shows the distribution of these annotations. To predict these labels, we trained eight binary logistic regression classifiers, one for each of the preceding values. If multiple classifiers in a label set returned positive, we selected the prediction with the higher log likelihood. The features we used for these classifiers were:

Stemmed Unigrams and Bigrams. We collected bags of n-grams from the sentence under consideration and its containing segment.

Anchor Text. The anchor text of the hyperlink in the sentence.

Hyperlink URL Tokens. We split the URL by punctuation (such as '/' and '.') and extracted tokens.

Privacy Policy URL Tokens. We also extracted tokens from the policy URL as features.

URL Similarity Measure. We calculated the Jaccard index between the vocabulary of the policy URL and the hyperlink URL. This feature is used to identify whether the hyperlink was to a first-party page or a third-party page.

Figure 2 illustrates the overall architecture of our system. We first use the coarse-grained step to identify the presence of an opt-out instance, and then use the fine-grained step to ascertain key properties of an opt-out choice if one is present.

4 Results and Discussion

This work is one of the first efforts to automatically detect the provision of choices in text. For the coarse-grained task, we consider a simple baseline that labels sentences as positive if they contain one or more opt-out specific words, which come from a vocabulary set that we identified by examining positive instances in the training set. The F1 of the baseline was 0.554.

We performed ablation tests excluding one feature at a time from the coarse-grained classifier. The results of these tests are presented in Table 2 as precision, recall, and F1 scores for the positive class, i.e., the opt-out class. Using the F1 scores as the primary evaluation metric, it appears that all features help in classification. The unigram, topic distribution, nonterminal, and modal verb and opt-out phrase features contribute the most to performance. Including all the features results in an F1 score of 0.735. Ablation test without unigram features resulted in the lowest F1 score of 0.585, and by analyzing features with higher logistic regression weights, we found n-grams such as *unsubscribe* to have intuitively high weights. We also found the production rule "S→SBAR, VP" to have a high weight, indicating that presence of subordinate clauses (SBARs) help in classification.

For an additional practical evaluation, we created a second dataset of sentences from the privacy policies of the 180 most popular websites (as determined by Alexa rankings. We selected only those sentences that contained hyperlinks, since they are associated with particularly actionable choices in privacy policy text. We used our model (as trained on the OPP-115 Corpus) to label the 3,842 sentences in this set, and then manually verified the 124 positive predictions, observing perfect precision. Although we were unable to measure recall using this method, the high precision suggests the robustness of the model and the practical applicability of this approach to tools for Internet users.

The results for the opt-out type classification are shown in Table 3. Because of data sparsity, we show performance figures for only the top two most frequent label combinations. These results

Features/Models	Precision	Recall	F1
All	**0.862**	**0.641**	**0.735**
All - Unigrams	0.731	0.487	0.585
All - Bigrams	0.885	0.590	0.708
All - Rel. Location	0.889	0.615	0.727
All - Topic Models	0.852	0.590	0.697
All - Productions	0.957	0.564	0.710
All - Nonterminals	0.913	0.538	0.677
All - Max. Depth	0.857	0.615	0.716
All - Avg. Depth	0.857	0.615	0.716
Phrase Inclusion - Baseline	0.425	0.797	0.554
Paragraph Vec. - 50 Dimensions	0.667	0.211	0.320
Paragraph Vec. - 100 Dimensions	0.667	0.158	0.255

Table 2: Results of ablation tests for the coarse-grained classifier.

	Precision	Recall	F1
FI, CM	0.947	0.947	0.947
TH, AD	0.905	0.977	0.940

Table 3: Fine-grained classifier results.

also demonstrate a practical level of performance for Internet user-oriented tools.

5 Conclusion

We presented an approach to the problem of automatically identifying privacy choices in privacy policy text. Our experiments show that a two-stage supervised learning procedure is appropriate for this task. Our approach is to initially identify choices offered by the text and then to determine their properties. Using ablation tests, we showed that a mixture of feature types can improve upon the performance of a baseline bag-of-words model. Planned future work for this project will include the creation of a browser plug-in to present opt-out hyperlinks to Internet users.

6 Acknowledgements

This work has been supported by the National Science Foundation as part of the Usable Privacy Policy Project (www.usableprivacy.org) under Grant No. CNS 13-30596. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSF, or the US Government

References

Waleed Ammar, Shomir Wilson, Norman Sadeh, and Noah A Smith. 2012. Automatic categorization of privacy policies: A pilot study.

Annie I Antón, Julia Brande Earp, and Angela Reese. 2002. Analyzing website privacy requirements using a privacy goal taxonomy. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, pages 23–31. IEEE.

Ngo Xuan Bach, Nguyen Le Minh, Tran Thi Oanh, and Akira Shimazu. 2013. A two-phase framework for learning logical structures of paragraphs in legal articles. *ACM Transactions on Asian Language Information Processing (TALIP)*, 12(1):3.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Fred H Cate. 2010. The limits of notice and choice. *IEEE Security & Privacy*, 8(2):59–62.

Lorrie Faith Cranor. 2012. Necessary but not sufficient: Standardized mechanisms for privacy notice and choice. *J. on Telecomm. & High Tech. L.*, 10:273.

Federal Trade Commission. 2000. Privacy Online: A Report to Congress. Technical report, Federal Trade Commission.

Filippo Galgani, Paul Compton, and Achim Hoffmann. 2012. Combining different summarization techniques for legal text. In *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*, pages 115–123. Association for Computational Linguistics.

Pedro Leon, Blase Ur, Richard Shay, Yang Wang, Rebecca Balebako, and Lorrie Cranor. 2012. Why johnny can't opt out: a usability evaluation of tools to limit online behavioral advertising. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 589–598. ACM.

Fei Liu, Nicole Lee Fella, and Kexin Liao. 2016a. Modeling language vagueness in privacy policies using deep neural networks. In *2016 AAAI Fall Symposium Series*.

Frederick Liu, Shomir Wilson, Florian Schaub, and Norman Sadeh. 2016b. Analyzing vocabulary intersections of expert annotations and topic models for data practices in privacy policies. In *2016 AAAI Fall Symposium Series*.

Lars Mahler. 2015. What is nlp and why should lawyers care? `http://www.lawpracticetoday.org/article/nlp-lawyers/`.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Aleecia M McDonald and Lorrie Faith Cranor. 2008. Cost of reading privacy policies, the. *ISJLP*, 4:543.

Simonetta Montemagni, Wim Peters, and Daniela Tiscornia. 2010. *Semantic Processing of Legal Texts*. Springer.

Rohan Ramanath, Fei Liu, Norman Sadeh, and Noah A Smith. 2014. Unsupervised alignment of privacy policies using hidden markov models.

Joel R Reidenberg, Travis Breaux, Lorrie Faith Cranor, Brian French, Amanda Grannis, James T Graves, Fei Liu, Aleecia McDonald, Thomas B Norton, Rohan Ramanath, Cameron Russell, Norman Sadeh, and Florian Schaub. 2015. Disagreeable privacy policies: Mismatches between meaning and users' understanding. *Berkeley Tech. LJ*, 30:39.

Norman Sadeh, Alessandro Acquisti, Travis D Breaux, Lorrie Faith Cranor, Aleecia M McDonald, Joel R Reidenberg, Noah A Smith, Fei Liu, N Cameron Russell, Florian Schaub, et al. 2013. The usable privacy policy project. Technical report, Technical Report, CMU-ISR-13-119, Carnegie Mellon University.

Florian Schaub, Rebecca Balebako, Adam L. Durity, and Lorrie Faith Cranor. 2015. A design space for effective privacy notices. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, pages 1–17, Ottawa. USENIX Association.

S Wilson, F Schaub, A Dara, F Liu, S Cherivirala, P G Leon, M S Andersen, S Zimmeck, K Sathyendra, N C Russell, T B Norton, E Hovy, J R Reidenberg, and N Sadeh. 2016. The creation and analysis of a website privacy policy corpus. In *Annual Meeting of the Association for Computational Linguistics, Aug 2016*. ACL.

Wei Xu, Xin Liu, and Yihong Gong. 2003. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273. ACM.

An Empirical Analysis of Edit Importance between Document Versions

Tanya Goyal
tanyagoyal.93@gmail.com
Big Data Experience Lab
Adobe Research

Sachin Kelkar
sachinkel19@gmail.com
Indian Institute of Technology, Roorkee

Manas Agarwal
manasagarwal1993@gmail.com
Indian Institute of Technology, Roorkee

Jeenu Grover
groverjeenu@gmail.com
Indian Institute of Technology, Kharagpur

Abstract

In this paper, we present a novel approach to infer significance of various textual edits to documents. An author may make several edits to a document; each edit varies in its impact to the content of the document. While some edits are surface changes and introduce negligible change, other edits may change the content/tone of the document significantly. In this paper, we perform an analysis of the human perceptions of edit importance while reviewing documents from one version to the next. We identify linguistic features that influence edit importance and model it in a regression based setting. We show that the predicted importance by our approach is highly correlated with the human perceived importance, established by a Mechanical Turk study.

1 Introduction

In collaborative content authoring, multiple authors make changes to the same document, which results in the final version being significantly different from the base draft. Often there is a need to review the edits made to the original document, which can be a long and arduous task. Tools like Microsoft Word (mic) and Adobe Acrobat (ado) provide reviewers with a list of edits, in the form of insertions and deletions. While helpful, these tools do not differentiate between the different types of edits, or consider the varying impact of edits. For instance, change from numeric '18' to word 'eighteen' may be a minor change and less crucial for the author to review, as compared to an edit that alters the facts of the document. Thus, in our work, we focus on automatically inferring the impact/change introduced by edits, and predict the perceived importance of such edits by authors.

In this paper, we perform a linguistic analysis of how humans evaluate the significance of edits while reviewing documents. Our algorithm assigns scores to edits between two versions of a document, which indicate the significance of the specified edit as perceived by the reviewer. We demonstrate the efficacy of our approach by comparing our algorithm generated edit importance scores with the human perceived ground truth importance, established through a Mechanical Turk survey.

2 Related Work

There has been significant amount of work on defining the importance of a keyword or a sentence in the context of document summarization (Mihalcea and Tarau, 2004). Some prior work has also been done on inferring the type of edits between Wikipedia versions. Bronner et al. (Bronner and Monz, 2012) proposed a supervised approach to classify Wikipedia edits as factual or fluency. Daxenberger et al. (Daxenberger and Gurevych, 2013) propose an approach to classify these edits into a 21-category taxonomy. However, none of the the prior work studies the impact or significance of the edit to the content of the document. They do not take the context of the change into account, neither do they study how edits are perceived by reviewers and the significance associated to each edit type. To the best of our knowledge, there is no prior work that evaluates the importance of an edit between document versions as perceived by human reviewers, which is the novel contribution of our work here.

All authors have equal contribution in this paper.

This work was done as part of an internship at Adobe Research

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2780–2784
Copenhagen, Denmark, September 7–11, 2017.

3 Discussion on types of edits

Before trying to automatically infer the importance of individual edits, we first identified the broad categories of textual edits made by authors. Bronner et al. (Bronner and Monz, 2012) broadly classify text edits into two categories, namely, *Factual Edits* and *Fluency Edits*. Factual edits refer to those that modify, add or delete information in the document while fluency edits mainly deal with changes in writing styles or paraphrasing. To obtain finer granularity edit categories, we subclassified factual edits into *Information Modify, Information Delete* and *Information Insert*. To further classify fluency changes, we looked at linguistic literature (Honeck, 1971) and identified subcategories of paraphrase changes. Based on this, fluency edits were further classified into *Lexical Paraphrase* (change of textual elements by synonymous words/phrases/numbers) and *Transformational Paraphrase* (change in the structure of the sentence, e.g. active to passive voice).
For the purpose of this paper, we assume these edit categories to be exhaustive and consequently classify all changes as belonging to one of these.

4 Data and Annotation

Due to the unavailability of an appropriately annotated dataset, we performed an online survey on Amazon Mechanical Turk[1] to capture people's perception of edit importance. To achieve this, we used an available corpus of news articles[2]. We created newer versions for these articles by manually introducing multiple changes to each article. Fig 1 provides statistics for the types of edits (based on the discussion in the previous section) across this entire corpus of 52 article pairs. There are a total of 523 changed sentence pairs in the document corpus, and an average of 1.2 edits per sentence pair.

For annotation of edit importance, we asked Mechanical Turk workers to assign an importance score to each pair of changed sentences. We first provide each turker with the initial (original) version of a news article. After the turker finishes reading the article, he is presented with a list of sentences that were changed between the initial and the final version, along with the changed sentences. The worker classifies each of these sentence pairs as belonging to one of the following importance classes (a) Very Important, (b) Moderately Important, (c) Important, (d) Neutral, (e) Not necessary for review. To avoid introducing biases based on our own notions of edit importance, we provide only a brief description of the task and encourage annotators to follow their own intuitions of importance. Each sentence pair is annotated by 3 annotators and the final importance score is calculated as the mean of the three scores.

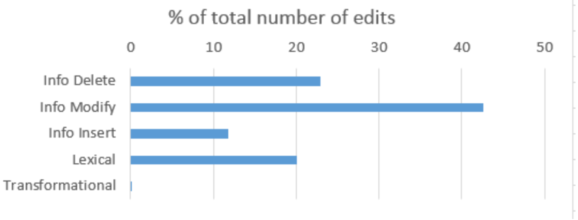


Figure 1: Number of edits of each type as a percentage of the total number of edits in the entire document corpus

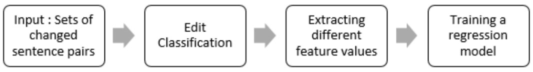

Input : Sets of changed sentence pairs
Edit Classification
Extracting different feature values
Training a regression model

Figure 2: Steps in training a supervised model to scores sentences

5 Solution Description

This section describes the methodology followed to obtain importance scores for text edits to documents. Fig 2 shows the overall workflow of the proposed approach.

The input to the algorithm is a corpus of documents D, which consists of pairs of documents (d, d') corresponding to the initial and final document versions respectively. We use the sentence alignment module proposed by Zhang et al. (Zhang and Litman, 2014) to obtain the mapped pairs of sentences (s, s'), where s represents a sentence in the first version d and s' is its modified variant in d'.

[1] https://www.mturk.com/mturk/

[2] http://literacynet.org/cnnsf/archives.html

5.1 Classification of Edits

The first step of the algorithm is to determine the number and types of the various edits between each sentence pair (s, s'). In this module, we assign an edit type label, as discussed in Section 3, to all edits between a pair of sentences.

Our analysis of the document corpus revealed that most text edits to sentences do not significantly change the structure of the sentence. Thus, a simple heuristic based approach can be used to identify edit types. We use the Stanford Parser to extract the POS tag sequences of the two sentences, with words backed off to their named entities wherever possible. We identify the longest common subsequence between these to obtain a word to word mapping for the sentence pair. If the ratio of the LCS and the mean of the sentence lengths (original and the modified) is above a threshold, we assume that the sentence structure is preserved. A simple word comparison between the similarly tagged words reveals instances of Information-Modify and Lexical changes; additions and deletions are identified as Information-Insert and Information-Delete respectively. In case the structure of the sentence is not preserved, the above heuristic fails, and we tag the sentence pair as Transformational Paraphrase. For such sentence pairs, we employ the method outlined by Bronner et al. (Bronner and Monz, 2012) and train a supervised classifier to differentiate between factual and fluency edits, without bothering about the subtype.

Following the outlined heuristic, we were able to correctly classify 92% of all edits in the document corpus, without using the supervised classifier.

5.2 Feature Extraction

Next, we extract linguistic features for supervised modeling of edit importance. We hypothesized that the importance of edits would be affected by both the nature of the edits, characterized by the aforementioned categories, as well as the relevance of the sentence to the content of the document. Thus, we chose features that capture both these aspects and have divided them into two groups, namely, change-related features and relevance-related features.

5.2.1 Change-related features

These set of features account for the factual differences between sentence pairs caused due to the edits. The complete list of such features is as follows:

- One-hot feature for type of edits identified in the Edit Classification module. We conjectured that different types of changes will have different perceived importance. For example, factual changes may be more important for the author to review compared to paraphrasing changes.
- One-hot feature for the POS tags and Named Entities whose count changes between the initial and the final version of the sentences. We also include one-hot features for those tags whose corresponding word changes between the two versions. These aim to capture the importance associated with deletion, insertion or modification of specific POS tags and Named Entities.
- One-hot features for the following dependency tuples that change between the two versions, with lexical items backed off to POS tags: (gov,typ, dep), (gov, typ), (typ, dep), (gov, dep).
- Count for the number of edits between the two versions.
- Absolute difference in the Flesch Kinkaid readability scores of the two sentences. We hypothesized that human perception of degree of change may be correlated with the change in ease of readability of content.

5.2.2 Relevance-related features

These features aim to score the sentences where the edit occurred. We conjectured that edit importance must also depend on the relevance of the underlying sentence to the content of the document. For instance, in an article about the monarchy in the United Kingdom, an edit that occurs in a sentence discussing the Queen may potentially be more important than one that provides generic facts about the country. The features we consider are :

- **TextRank Score**: We use the TextRank algorithm (Mihalcea and Tarau, 2004) to extract keywords from the document along with the PageRank score attached to them. Each sentence is scored based on the cumulative scores of all keywords that occur in it. Explicitly, the score of a particular sentence is calculated as:

$$Score(s) = \frac{\sum_{w \in W \cap S} KeywordScore(w)}{|S|} \tag{1}$$

where S is the set of words in the sentence and W is the set of keywords extracted by the TextRank algorithm.

- **Position of the sentence in the document**: The importance of sentence position has been studied in (Edmundson, 1969). We expect more important sentences to have a higher edit importance score attached to them.

We train a ridge regression model with the model parameters tuned using cross validation on the training data. We report the Spearman ρ correlation (Spearman, 1904) of the predicted edit importance scores with the human annotated scores on the test data.

6 Experiments and Results

In this section, we discuss the various experiments performed, and the results obtained. **Baselines**: To the best of our knowledge, our work is the first that attempts to infer importance/impact of text edits between document versions. Thus, we did not have established baselines to compare against. Instead we use the following features as baselines:

- **Sentence Order** - Sentences are ordered according to their position in the document, with the first sentence assigned most importance. This is also the order in which a reviewer would normally view edits.

- **Readability Score** - Sentence edit importance scores are calculated as being proportional to the change in their readability scores.

- **Text Rank** - We expect sentences with higher TextRank score to have higher edit importance attached to them.

Table 1 outlines the Spearman ρ correlation of our model and the above baselines with human judgments. We are able to achieve significant improvement over the baselines using the full set of features. An interesting observation was that sentence position correlates poorly with the human annotated importance scores. This indicates that the order/position of sentences has negligible effect on the perceived significance. Both readability score and TextRank have reasonable influence on edit importance, though neither of them is able to match the performance of the full set of features.

Approach	Spearman ρ
Sentence Position	0.067
Readability Score	0.306
Text Rank	0.208
Proposed Approach	0.979

Table 1: Spearman ρ of the predicted importance score with the human annotated importance scores.

Feature	Spearman ρ
Type of Change	0.47907
Readability Score	0.311989
Change in POS tags	0.978821
Change in NE	0.189176
Change in Dependency Tuples	0.96417
Sentence Position	0.06846
Text Rank	0.209058

Table 2: Performance of each feature group in isolation. Numbers reflect the performance (Spearman ρ) of the model when using only the specified feature group, relative to the performance when using all features.

Contribution of feature groups

In order to gain better insight into individual feature performance, we look more closely at the performance of each feature group in isolation. Table 2 shows the performance of the model when using only a specific feature group, relative to the performance when using all features. This provides us with a number of interesting insights. First, it is evident that change-related features contribute more to edit importance than relevance-related features.

According to our results, humans perceive change in number and types of POS tags to be the most significant indicator of edit importance. For further insight, we looked at the coefficient values of individual POS tags in the ridge regression model trained using only POS tags as features. Our investigations revealed that change in proper nouns, nouns, present participle verbs and modal are most highly correlated with edit importance. Contrary to our expectation, modification of named entities does not significantly influence edit importance. This may be due to the fact that named entity changes occur in only a small subset of sentences, and hence cannot be good predictors of edit importance when used as a feature by themselves.

7 Conclusion and Future Work

In this paper we present a novel approach to infer the importance of text edit between two document versions. We present an empirical analysis of the relevance of various linguistic features for the task of scoring edit importance and model it using a regression model. AMT is used to collect human annotated data for edit importance, and a comparison against those establish the superiority of our proposed approach over several baselines.

References

Adobe acrobat. https://acrobat.adobe.com/in/en/acrobat/pdf-reader.html. Accessed: 2017-07-05.

Microsoft word. https://products.office.com/en-in/word. Accessed: 2017-07-05.

Amit Bronner and Christof Monz. 2012. User edits classification using document revision histories. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 356–366. Association for Computational Linguistics.

Johannes Daxenberger and Iryna Gurevych. 2013. Automatically classifying edit categories in wikipedia revisions. In *EMNLP*, pages 578–589.

Harold P Edmundson. 1969. New methods in automatic extracting. *Journal of the ACM (JACM)*, 16(2):264–285.

Richard P Honeck. 1971. A study of paraphrases. *Journal of Verbal Learning and Verbal Behavior*, 10(4):367–381.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. Association for Computational Linguistics.

Charles Spearman. 1904. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101.

Fan Zhang and Diane Litman. 2014. Sentence-level rewriting detection. *Grantee Submission*.

Transition-Based Disfluency Detection using LSTMs

Shaolei Wang[1], **Wanxiang Che**[*1], **Yue Zhang**[2], **Meishan Zhang**[3] and **Ting Liu**[1]
[1]Center for Social Computing and Information Retrieval, Harbin Institute of Technology, China
[2]Singapore University of Technology and Design
[3]School of Computer Science and Technology, Heilongjiang University, China
{slwang, car, tliu}@ir.hit.edu.cn
yue_zhang@sutd.edu.sg, mason.zms@gmail.com

Abstract

We model the problem of disfluency detection using a transition-based framework, which incrementally constructs and labels the disfluency chunk of input sentences using a set of transition actions without syntax information. Compared with sequence labeling methods, it can capture non-local chunk-level features; compared with joint parsing and disfluency detection methods, it is free for noise in syntax. Experiments show that our model achieves state-of-the-art F-score on both the commonly used English Switchboard test set and a set of in-house annotated Chinese data.

1 Introduction

Disfluency detection is the task of recognizing non-fluent word sequences in spoken language transcripts (Zayats et al., 2016; Wang et al., 2016; Wu et al., 2015). As shown in Figure 1, standard annotation of disfluency structure (Shriberg, 1994) indicates the reparandum (words that are discarded, or corrected by the following words), the interruption point (+) marking the end of the reparandum, the associated repair, and an optional interregnum after the interruption point (filled pauses, discourse cue words, etc.).

Ignoring the interregnum, disfluencies can be categorized into three types: restarts, repetitions, and corrections, based on whether the repair is empty, the same as the reparandum or different, respectively. Table 1 gives a few examples. Interregnums are easy to detect as they often consist of fixed phrases (e.g. “uh”, “you know”). However, reparandums are more difficult to detect, because they can be in arbitrary form. Most previous disfluency detection work focuses on detecting reparandums.

$$\text{I want a flight [}\underbrace{\textit{to Boston}}_{\text{RM}} + \underbrace{\{\textit{um}\}}_{\text{IM}}\ \underbrace{\text{to Denver}}_{\text{RP}}\text{]}$$

Figure 1: Sentence with disfluencies annotated in English Switchboard corpus. RM=Reparandum, IM=Interregnum, RP=Repair. The preceding RM is corrected by the following RP.

Type	**Annotation**
repair	[I just + I] enjoy working
repair	[we want + {well} in our area we want] to
repetition	[it’s + {uh} it’s] almost like
restart	[we would like +] let’s go to the

Table 1: Different types of disfluencies.

The main challenges of detecting reparandums include that they vary in length, may occur in different locations, and are sometimes nested. For example, the longest reparandum in our training set has fifteen words. Hence, it is very important to capture long-range dependencies for disfluency detection. Since there is large parallelism between the reparandum chunk and the following repair chunk (for example, in Figure 1, the reparandum begins with *to* and ends before another occurrence of *to*), it is also useful to exploit chunk-level representation, which explicitly makes use of resulted infelicity disfluency chunks.

Common approaches take disfluency detection as a sequence labeling problem, where each sentential word is assigned with a label (Zayats et al., 2016; Hough and Schlangen, 2015; Qian and Liu, 2013; Georgila, 2009). These methods achieve good performance, but are not powerful enough to capture complicated disfluencies with longer spans or distances. Another drawback of these approaches is that they are unable to exploit chunk-

[*] Email corresponding.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2785–2794
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

level features. Semi-CRF (Ferguson et al., 2015) is used to alleviate this issue to some extent. Semi-CRF models still have their inefficiencies because they can only use the local chunk information limited by the markov assumption when decoding.

A different line of work (Rasooli and Tetreault, 2013; Honnibal and Johnson, 2014; Wu et al., 2015) adopts transition-based parsing models for disfluency detection. This line of work can be seen as a joint of disfluency detection and parsing. The main advantage of the joint models is that they can capture long-range dependency of disfluencies as well as chunk-level information. However, they introduce additional annotated syntactic structure, which is very expensive to produce, and can cause noise by significantly enlarging the output search space.

Inspired by the above observations, we investigate a transition-based model without syntactic information. Our model incrementally constructs and labels the disfluency chunks of input sentences using an algorithm similar to transition-based depency parsing. As shown in Figure 2, the model state consists of four components: (i) O, a conventional sequential LSTM (Hochreiter and Schmidhuber, 1997) to store the words that have been labeled as fluency. (ii) S, a stack LSTM to represent partial disfluency chunks, which captures chunk-level information. (iii) A, a conventional sequential LSTM to represent history of actions. (iiii) B, a Bi-LSTM to represent words that have not yet been processed. A sequence of transition actions are used to consume input tokens and construct the output from left to right. To reduce error propagation, we use beam-search (Collins and Roark, 2004) and scheduled sampling (Bengio et al., 2015), respectively.

We evaluate our model on the commonly used English Switchboard test set and a in-house annotated Chinese data set. Results show that our model outperforms previous state-of-the-art systems. The code is released[1].

2 Background

For a background, we briefly introduce transition-based parsing and its extention for joint disfluency detection. An arc-eager transition-based parsing system consists of a stack σ containing words being processed, a buffer β containing words to be processed and a memory A storing dependency

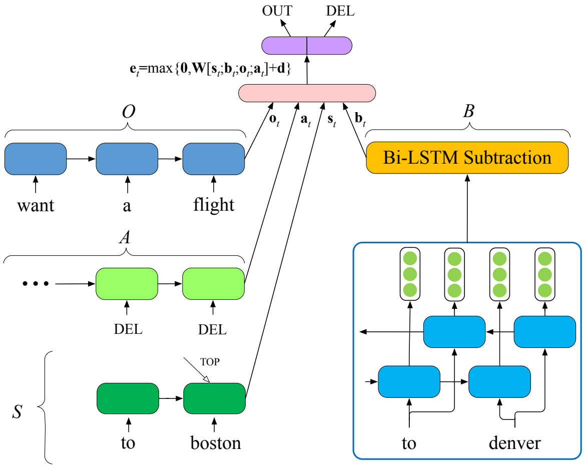


Figure 2: model state when processing the sentence “want a flight to boston to denver”.

arcs which have been generated. There are four types of transition actions (Nivre, 2008)

- *Shift* : Remove the front of the buffer and push it to the stack.
- *Reduce* : Pop the top of the stack.
- *LeftArc* : Pop the top of the stack, and link the popped word to the front of the buffer.
- *RightArc* : Link the front of the buffer to the top of the stack, remove the front of the buffer and push it to the stack.

Many neural network parsers have been constructed under this framework, such as (Dyer et al., 2015), who use different LSTM structure to represent information from σ to β.

For disfluency detection, the input is a sentence with disfluencies from automatic speech recognition (ASR). We denote the word sequence as $w_1^n = (w_1, ..., w_n)$. The output of the task is a sequence of binary tags denoted as $D_1^n = (d_1, ..., d_n)$, where each d_i corresponds to the word w_i, indicating whether w_i is a disfluent word or not. Hence the task can be modeled as searching for the best sequenc D^* given the stream of words w_1^n

$$D^* = argmax_D P(D_1^n | w_1^n)$$

Wu et al. (2015) proposes a statistical transition-based disfluency detection model, which performs disfluency detection and parsing jointly by augmenting the Shift-Reduce algorithm with a binary classifier transition (BCT) action:

[1] https://github.com/hitwsl/transition_disfluency

- *BCT* : Classify whether the current word is disfluent or not. If it is, remove it from the buffer, push it into the stack which is similar to *Shift* and then mark it as disfluent. Otherwise the original parser transition actions will be used.

Disfluency detection and parsing are jointly optimized

$$(D^*, T^*) = argmax_{D,T} \prod_{i=1}^{n} P(d_i|w_1^i, T_1^{i-1}) \\ \times P(T_1^i|w_1^i, T_1^{i-1}, d_i),$$

where T_1^i is the partial tree after word w_i is consumed, d_i is the disfluency tag of w_i. $P(T_1^i|.)$ is the parsing model and $P(d_1^i|.)$ is the disfluency model used to predict the disluency tags on the contexts of partial trees that have been built.

3 Our Transition-Based Model

The BCT model serves as a state-of-the-art transition-based baseline. However, it requires that the training data contains both syntax trees and disfluency annotations, which reduces the practicality of the algorithm. Also, BCT does not explicitly make use of resulting infelicity disfluency chunks. Being a discrete model, the performance relies heavily on manual feature engineering.

To address the constraints above, we apply a transition-based neural model for disfluency detection that does not use any syntax information. Our transition-based method incrementally constructs and labels the disfluency chunk of input sentences by performing a sequence of actions. The task is modeled as

$$(D^*, T^*) = argmax_{D,T} \prod_{i=1}^{n} P(d_i, T_1^i|w_1^i, T_1^{i-1}),$$

where T_1^i is the partial model state after word w_i is consumed. d_i is the disfluency tag of w_i.

3.1 Transition-Based Disfluency Detection

Our model incrementally constructs and labels the disfluency chunks of input sentences, where a state is represented by a tuple (O, S, A, B):

- *output* (O) : the *output* is used to represent the words that have been labeled as fluent.
- *stack* (S) : *stack* is used to represent the partially constructed disfluency chunk.
- *action* (A) : *action* is used to represent the complete history of actions taken by the transition system.
- *buffer* (B) : *buffer* is used to represent the sentences that have not yet been processed.

Given an input disfluent sentence, the *stack*, *output* and *action* are initially empty and the *buffer* contains all words of the sentence, a sequence of transition actions are used to consume words in the *buffer* and build the output sentence:

- OUT: which moves the first word in the *buffer* to the *output* and clears out the *stack* if it is not empty.
- DEL: which moves the first word in the *buffer* to the *stack*.

Search Algorithm

Based on the transition system, the decoder searches for an optimal action sequence for a given sentence. The system is initialized by pushing all the input words and their representations (of § 3.3) onto B in the reverse order, such that the first word is at the top of B, and S, O and A each contains an empty-stack token.

At each step, the system computes a composite representation of the model states (as determined by the current configurations of B, S, O, and A), which is used to predict an action to take. Decoding completes when B is empty (except for the empty-stack symbol), regardless of the state of S. Since each token in B is either moved directly to O or S every step, the total number of actions equals to the length of input sentence. Table 2 shows the sequence of operations required to process the sentence "want a flight to boston to denver".

As shown in Figure 2, the model state representation at time t, which is written as e_t, is defined as:

$$e_t = max\{0, W[s_t; b_t; o_t; a_t] + d\},$$

where W is a learned parameter matrix, s_t is the representation of S, b_t is the representation of B, o_t is the representation of O, a_t is the representation of A, d is a bias term. $(W[s_t; b_t; o_t; a_t] + d)$ is passed through a component-wise rectified linear unit (ReLU) for nonlinearity (Glorot et al., 2011).

Step	Action	Output	Stack	Buffer
0		[]	[]	[a, flight, to, boston, to, denver]
1	OUT	[a]	[]	[flight, to, boston, to, denver]
2	OUT	[a, flight]	[]	[to, boston, to, denver]
3	DEL	[a, flight]	[to]	[boston, to, denver]
4	DEL	[a, flight]	[to, boston]	[to, denver]
5	OUT	[a, flight, to]	[]	[denver]
6	OUT	[a, flight, to, denver]	[]	[]

Table 2: Segmentation process of *a flight to boston to denver*

Finally, the model state e_t is used to compute the probability of the action at time t as:

$$p(z_t|e_t) = \frac{exp(g_{z_t}^{\mathrm{T}} e_t + q_{z_t})}{\sum_{z' \in \mathcal{A}(S,B)} exp(g_{z'}^{\mathrm{T}} e_t + q_{z'})},$$

where g_z is a column vector representing the embedding of the transition action z, and q_z is a bias term for action z. The set $\mathcal{A}(S, B)$ represents the valid actions that may be taken given the current state. Since $e_t = f(s_t, b_t, a_t, o_t)$ encodes information about all previous decisions made by the transition system, the probability of any valid sequence of transition actions z conditioned on the input can be written as:

$$p(z|w) = \prod_{t=1}^{|z|} p(z_t|e_t)$$

We then have

$$\begin{aligned}(D^*, T^*) &= argmax_{D,T} \prod_{i=1}^{|z|} P(d_i, T_1^i | w_1^i, T_1^{i-1}) \\ &= argmax_{D,T} \prod_{t=1}^{|z|} p(z_t|e_t),\end{aligned}$$

where the disfluency detection task is merged into the transition-based system.

Beam Search

The mainly drawback of greedy search is error propagation. An incorrect action will have a negative influence to its subsequent actions, leading to an incorrect output sequence. One way to reduce error propagation is beam-search. Because the number of actions taken always equals to the number of input sentence for every valid path, it is straightforward to use beam search. We use beam-search for both training and testing. The early update strategy from Collins and Roark (2004) is applied for training. In particular, each training sequence is decoded, and we keep track of the location of the gold path in the beam. If the gold path falls out of the beam at step t, decoding process is stopped and parameter update is performed using the gold path as a positive example, and beam items as negative examples. We also use the global optimization method (Andor et al., 2016; Zhou et al., 2015) to train our beam-search model.

Scheduled Sampling

Scheduled sampling (Bengio et al., 2015) can also be used to reduce error propagation. The training goal of the greedy baseline is to maximize the likelihood of each action given the current model state, which means that the correct action is taken at each step. Doing inference, the action predicted by the model itself is taken instead. This discrepancy between training and inference can yield errors that accumulate quickly along the searching process. Scheduled sampling is used to solve the discrepancy by gently changing the training process from a fully guided scheme using the true previous action, towards a less guided scheme which mostly uses the predicting action instead. We take the action gaining higher $p(z_t|e_t)$ with a certain probability p, and a probability $(1-p)$ for the correct action when training.

3.2 State Representation

For better capturing non-local context information, we use LSTM structures to represent different components of each state, including *buffer*, *action*, *stack*, and *output*. In particular, we exploit LSTM-Minus (Wang and Chang, 2016) to model the *buffer* segment, conventional LSTM to model the *action* and *ouptut* segment, and stack LSTM (Dyer et al., 2015) to model the *stack* segments, which demonstrates highly effectively in parsing task.

Buffer Representation

In order to construct more informative representation, we use a Bi-LSTM to represent the *buffer* following the work of Wang and Chang (2016), where the subtraction between a unidirectional

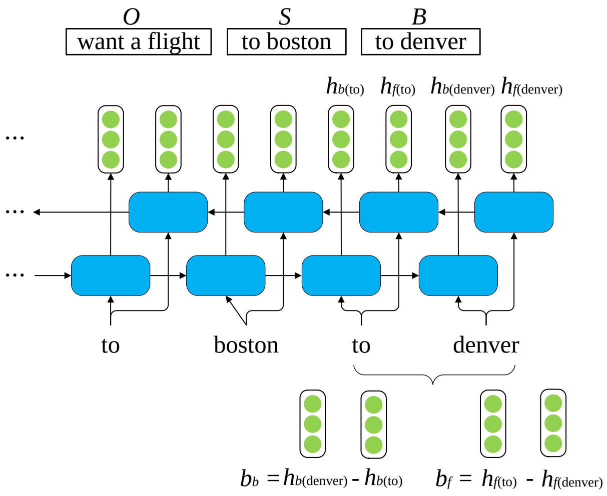

O
want a flight
S
to boston
B
to denver
$h_{b(\text{to})}$ $h_{f(\text{to})}$ $h_{b(\text{denver})}$ $h_{f(\text{denver})}$
...
...
...
to
boston
to
denver
$b_b = h_{b(\text{denver})} - h_{b(\text{to})}$
$b_f = h_{f(\text{to})} - h_{f(\text{denver})}$

Figure 3: Illustration for learning *buffer* representation based on a Bi-LSTM, $h_f(*)$ and $h_b(*)$ indicate the hidden vectors of forward and backward LSTM respectively.

LSTM hidden vectors is utilized to represent a segment's information. We perform a similar method in a Bi-LSTM to obtain the representation of the *buffer*. The forward and backward subtractions for the *buffer* can be described as $b_f = h_f(l) - h_f(f)$ and $b_b = h_b(f) - h_b(l)$, respectively, where $h_f(f)$ and $h_f(l)$ are the hidden vectors of the first and the last words in the forward LSTM, respectively. Similarly, $h_b(f)$ and $h_b(l)$ are the hidden vectors of the first and the last words in the backward LSTM, respectively. Then these subtractions are concatenated as the representation of the *buffer* $b_t = b_f \oplus b_b$. As illustrated in Figure 3, the forward and backward subtractions for *buffer* are $b_f = h_f(to) - h_f(denver)$ and $b_b = h_b(denver) - h_b(to)$, respectively. Here *to* is the first word in *buffer* and *denver* is the last. Then b_f and b_b are concatenated as the representation of *buffer*.

Action Representation

We represent an action a with an embedding $e_a(a)$ from a looking-up table E_a , and apply a conventional LSTM to represent the complete history of actions taken by the transition system. Once an action a is taken, the embedding $e_a(a)$ will be added to the right-most position of the LSTM.

Stack Representation

We use a stack LSTM (Dyer et al., 2015) to represent partial disfluency chunk. The stack LSTM tries to augment the conventional LSTM with a "stack pointer". For a conventional LSTM, new inputs are always added in the right-most position; but in a stack LSTM, the current location of the stack pointer determines which cell in the LSTM provides c_{t-1} and h_{t-1} when computing the new memory cell contents. In addition to adding elements to the end of the sequence, the stack LSTM provides a *pop* operation which moves the stack pointer to the previous element. Thus, the LSTM can be understood as a stack implemented so that contents are never overwritten, When the action OUT is taken, the *stack* is cleared by moving the stack pointer to the initial position. When the action DEL is taken, the representation of the *buffer* will be added directly to the stack LSTM.

Output Representation

We use a conventional LSTM to represent the *output*. When the action OUT is taken, the representation of the *buffer* will be added directly to the right-most position of the LSTM. Because the words in the *output* are a continuous subsequence of the final output sentence with disfluencies removed, the LSTM representation can be seen as a pseudo language model and thus has the ability to keep the generated sentence grammatical, which is very important for disfluency detection.

3.3 Token Embeddings

We use four vectors to represent each input token: a learned word embedding w; a fixed word embedding $\widetilde{w}$; a learned POS-tag embedding p; and a hand-crafted feature representation d. The four vectors are concatenated, transformed by a matrix V and fed to a rectified layer to learn a feature combination:

$$x = max\{0, V[\widetilde{w}; w; p; d] + b\},$$

where V means vector concatenation.

Following the work of Wang et al. (2016), we extract two types of hand-crafted discrete features (as shown in Table 3) for each token in a sentence, and incorporate them into our neural networks by translating them into a 0-1 vector d. The dimension of d is 78, which equals to the number of discrete features. For a token x_t, d_i fires if x_t matches the i-th pattern of the feature templates. The duplicate features indicate whether x_t has a *duplicated* word/POS-tag in certain distance. The *similarity* features indicate whether the surface string of x_t resembles its surrounding words.

duplicate features
$Duplicate(i, w_{i+k}), -15 \leq k \leq +15$ and $k \neq 0$: if w_i equals w_{i+k}, the value is 1, others 0
$Duplicate(p_i, p_{i+k}), -15 \leq k \leq +15$ and $k \neq 0$: if p_i equals p_{i+k}, the value is 1, others 0
$Duplicate(w_i w_{i+1}, w_{i+k} w_{i+k+1}), -4 \leq k \leq +4$ and $k \neq 0$: if $w_i w_{i+1}$ equals $w_{i+k} w_{i+k+1}$, the value is 1, others 0
$Duplicate(p_i p_{i+1}, p_{i+k} p_{i+k+1}), -4 \leq k \leq +4$ and $k \neq 0$: if $p_i p_{i+1}$ equals $p_{i+k} p_{i+k+1}$, the value is 1, others 0
similarity features
$fuzzyMatch(w_i, w_{i+k}), k \in \{-1, +1\}$: $similarity = 2 * num\_same\_letters/(len(w_i) + len(w_{i+k}))$. if $similarity > 0.8$, the value is 1, others 0

Table 3: Discrete features used in our transition-based neural networks. p-POS tag. w-word.

4 Experiments

4.1 Settings

Dataset. Our training data include the Switchboard portion of the English Penn Treebank (Marcus et al., 1993) and a in-house Chinese data set. For English, two annotation layers are provided: one for syntactic bracketing (MRG files), and the other for disfluencies (DPS files). The Switchboard annotation project was not fully completed. Because disfluency annotation is cheaper to produce, many of the DPS training files do not have matching MRG files. Only 619,236 of the 1,482,845 tokens in the DPS disfluency detection training data have gold-standard syntactic parses. To directly compare with transition-based parsing methods (Honnibal and Johnson, 2014; Wu et al., 2015), we also use the subcorpus of PARSED/MRG/SWBD. Following the experiment settings in Charniak and Johnson (2001), the training subcorpus contains directories 2 and 3 in PARSED/MRG/SWBD and directory 4 is split into test, development sets and others. Following Honnibal and Johnson (2014), we lower-case the text and remove all punctuations and partial words[2]. We also discard the ‘um’ and ‘uh’ tokens and merge ‘you know’ and ‘i mean’ into single tokens. Automatic POS-tags generated from pocket_crf (Qian and Liu, 2013) are used as POS-tag in our experiments.

For Chinese experiments, we collect 25k spoken sentences from meeting minutes, which are transcribed using the iflyrec toolkit[3], and annotate them with only disfluency annotations according to the guideline proposed by Meteer et al. (1995).

4.2 Neural Network Training

Pretrained Word Embeddings. Following Dyer et al. (2015) and Wang et al. (2016), we use a variant of the skip n-gram model introduced by Ling et al. (2015), named “structured skip n-gram”, to create word embeddings. The AFP portion of English Gigaword corpus (version 5) is used as the training corpus. Word embeddings for Chinese are trained on Chinese baike corpus. We use an embedding dimension of 100 for English, 300 for chinese.

Hyper-Parameters. Both the Bi-LSTMs and the stack LSTMs have two hidden layers and their dimensions are set to 100. Pretrained word embeddings have 100 dimensions and the learned word embeddings have also 100 dimensions. Pos-tag embeddings have 12 dimensions. The dimension of action embeddings is set to 20.

4.3 Performance On English Swtichboard

We build two baseline systems using CRF and Bi-LSTM, respectively. The hand-crafted discrete features of CRF refer to those in Ferguson et al. (2015). For the Bi-LSTM model, the token embedding is the same with our transition-based method. Table 4 shows the result of our model on both the development and test sets. Beam search improves the F-score form 87.1% to 87.5%, which is consistent with the finding of Buckman et al. (2016) on the LSTM parser of (Dyer et al., 2015) (improvements by about 0.3 point). Scheduled sampling achieves the same improvements compared to beam-search. Because of high training speed, we conduct subsequent experiments based on scheduled sampling.

We compare our transition-based neural model to five top performing systems. Our model outperforms the state-of-the-art, achieving a 87.5% F-

[2] words are recognized as partial words if they are tagged as ‘XX’ or end with ‘-’

[3] the iflyrec toolkit is available at http://www.iflyrec.com/

Method	Dev			Test		
	P	R	F1	P	R	F1
CRF	93.9	78.3	85.4	91.7	75.1	82.6
Bi-LSTM	94.1	79.3	86.1	91.7	80.6	85.8
greedy	91.4	83.7	87.4	91.1	83.3	87.1
+beam	93.6	83.6	88.3	92.8	82.7	**87.5**
+schedualed	92.3	84.3	88.1	91.1	84.1	**87.5**

Table 4: Experiment results on the development and test data of English Switchboard data.

Method	**P**	**R**	**F1**
Our	91.1	84.1	**87.5**
Attention-based (Wang et al., 2016)	91.6	82.3	86.7
Bi-LSTM (Zayats et al., 2016)	91.8	80.6	85.9
semi-CRF (Ferguson et al., 2015)	90.0	81.2	85.4
UBT (Wu et al., 2015)	90.3	80.5	85.1
M^3N (Qian and Liu, 2013)	-	-	84.1

Table 5: Comparison with previous state-of-the-art methods on the test set of English Switchboard.

score as shown in Table 5. It achieves 2.4 point improvements over UBT (Wu et al., 2015), which is the best syntax-based method for disfluency detection. The best performance by linear statistical sequence labeling methods is the semi-CRF method of Ferguson et al. (2015), achieving a 85.4% F-score leveraging prosodic features. Our model obtains a 2.1 point improvement compared to this. Our model also achieves 0.8 point improvement over the neural attention-based model (Wang et al., 2016), which regards the disfluency detection as a sequence-to-sequence problem. We attribute the success to the strong ability to learn global chunk-level features and the good state representation such as the stack-LSTM.

4.4 Result On DPS Corpus

As described in section 3.1, to directly compare with the transition-based parsing methods (Honnibal and Johnson, 2014; Wu et al., 2015), we only use MRG files, which are less than the DPS files. In fact, many methods, such as Qian and Liu (2013), have used all the DPS files as training data. We are curious about the performance of our system using all the DPS files. Following the experimental settings of Johnson and Charniak (2004), the corpus is split as follows: main training consisting of all sw[23]*.dps files, development training consisting of all sw4[5-9]*.dps files and test training consisting of all sw4[0-1]*.mrg files. Table 6 shows the result on the DPS files. The result of M^3N^* comes from our experiments with the toolkit[4] released by Qian and Liu (2013), which use the same data set and pre-processing. Our model achieves a 88.1% F-score by using more training data, obtaining 0.6 point improvement compared with the system training on MRG files. The performance is far better than the sequence labeling methods that use DPS files for training.

Method	**P**	**R**	**F1**
Our	93.1	83.5	**88.1**
Bi-LSTM	92.4	82.0	86.9
M^3N^* (Qian and Liu, 2013)	90.6	78.7	84.2
CRF	91.8	77.2	83.9

Table 6: Test result of our transition-based model using DPS files for training.

Method	Dev			Test		
	P	R	F1	P	R	F1
Our	68.9	40.4	50.9	77.2	37.7	**50.6**
Bi-LSTM	60.1	41.3	48.9	65.3	38.2	48.2
CRF	73.7	33.5	46.1	77.7	32.0	45.3

Table 7: performance on Chinese annotated data

4.5 Performance on Chinese

Table 7 shows the results of Chinese disfluency detection. Our model obtains a 2.4 point improvement compared with the baseline Bi-LSTM model and a 5.3 point compared with the baseline CRF model. The performance on Chinese is much lower than that on English. Apart from the smaller training set, the main reason is that the proportion of repair type disflueny is much higher.

5 Analysis

5.1 Ablation Tests

As described in section 3.1, the sate representation has four components. We explicitly compare the impact of different parts. As shown in Table 8, the F-score decreases most heavily without *stack*, which indicates that it is very necessary to capture chunk-level information for disfluency detection and our model can model it effectively. The results also show that *output*, which can be seen as a pseudo language model, has important influence on model performance. Seen from the result, history of actions represented in *action* is also useful for predicting at each step. The F-score decreases

[4] The toolkit is available at https://code.google.com/p/disfluency-detection/downloads.

Method	P	R	F1
ALL	91.1	84.1	**87.5**
- *stack*	93.5	80.6	86.5
- *action*	91.6	83.0	87.1
- *output*	89.0	84.4	86.7
- Bi-LSTM	93.6	81.4	87.1

Table 8: Results of feature ablation experiments on English Switchboard test data. "- Bi-LSTM" means using unidirectional LSTM for *buffer*

about 0.4 point, which shows that Bi-LSTM can capture more information compared to simple unidirectional LSTM.

5.2 Repetitions vs Non-repetitions

Repetition disfluencies are easier to detect and even some simple hand-crafted features can handle them well. Other types of reparandums such as repair are more complex (Zayats et al., 2016; Ostendorf and Hahn, 2013). In order to better understand model performances, we evaluate our model's ability to detect repetition vs. non-repetition (other) reparandum. The results are shown in Table 9. All the three models achieve high score on repetition reparandum. Our transition-based model is much better in predicting non-repetitions compared to CRF and Bi-LSTM. We conjecture that our transition-based structure can capture more of the reparandum/repair "rough copy" similarities by learning represention of both chunks and global state.

6 Related Work

Common approaches take disfluency detection as a sequence labeling problem, where each sentential word is assigned with a label (Georgila, 2009; Qian and Liu, 2013). These methods achieve good performance, but are not powerful enough to capture complicated disfluencies with longer spans or distances. Another drawback is that they have no ability to exploit chunk-level features. There are also works that try to use recurrent neural network (RNN), which can capture dependencies at any length in theory, on disfluency detection problem (Zayats et al., 2016; Hough and Schlangen, 2015). The RNN method treats sequence tagging as classification on each input token. Hence, it also has no power to exploit chunk-level features. Some works (Wang et al., 2016) regard the disfluency detection as a sequence-to-sequence problem and propose a neural attention-based model for it. The attention-based model can capture a global representation of the input sentence by using a RNN when encoding. It can strongly capture long-range dependencies and achieves good performance, but are also not powerful enough to capture chunk-level information. To capture chunk-level information, Ferguson et al. (2015) try to use semi-CRF for disfluency detection, and reports improved results. Semi-CRF models still have their inefficiencies because they can only use the local chunk information limited by the markov assumption when decoding.

Method	Repetitions	Non-repetitions	Either
CRF	93.8	61.4	82.6
Bi-LSTM	93.1	65.3	85.8
OUR	93.3	68.7	**87.5**

Table 9: F-score of different types of reparandums on English Switchboard test data.

Many syntax-based approaches (Lease and Johnson, 2006; Rasooli and Tetreault, 2013; Honnibal and Johnson, 2014; Wu et al., 2015) have been proposed which jointly perform dependency parsing and disfluency detection. The main advantage of joint models is that they can capture long-range dependency of disfluencies. However, it requires that the training data contains both syntax trees and disfluency annotations, which reduces the practicality of the algorithm. The performance relies heavily on manual feature engineering.

Transition-based framework has been widely exploited in a number of other NLP tasks, including syntactic parsing (Zhang and Nivre, 2011; Zhu et al., 2013), information extraction (Li and Ji, 2014) and joint syntactic models (Zhang et al., 2013, 2014).

Recently, deep learning methods have been widely used in many nature language processing tasks, such as name entity recognition (Lample et al., 2016), zero pronoun resolution (Yin et al., 2017) and word segmentation (Zhang et al., 2016). The effectiveness of neural features has also been studied for this framework (Zhou et al., 2015; Watanabe and Sumita, 2015; Andor et al., 2016). We apply the transition-based neural framework to disfluency detection, which to our knowledge has not been investigated before.

7 Conclusion

We introduced a transition-based model for disfluency detection, which does not use any syntax information, learning represention of both chunks and global contexts. Experiments showed that our model achieves the state-of-the-art F-scores on both the commonly used English Switchboard test set and a in-house annotated Chinese data set.

Acknowledgments

We thank the anonymous reviewers for their valuable suggestions. This work was supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61370164 and 61632011. The Chinese disfluency data is annotated by IFLYTEK CO.,LTD.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany. Association for Computational Linguistics.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Jacob Buckman, Miguel Ballesteros, and Chris Dyer. 2016. Transition-based dependency parsing with heuristic backtracking. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing; 2016 Nov 1-5; Austin, Texas, USA. Stroudsburg (USA): Association for Computational Linguistics (ACL); 2016. p. 2313-18*. ACL (Association for Computational Linguistics).

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–9. Association for Computational Linguistics.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 111. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

James Ferguson, Greg Durrett, and Dan Klein. 2015. Disfluency detection with a semi-markov model and prosodic features. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 257–262. Association for Computational Linguistics.

Kallirroi Georgila. 2009. Using integer linear programming for detecting speech disfluencies. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 109–112. Association for Computational Linguistics.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Aistats*, volume 15, page 275.

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

Matthew Honnibal and Mark Johnson. 2014. Joint incremental disfluency detection and dependency parsing. *Transactions of the Association for Computational Linguistics*, 2:131–142.

Julian Hough and David Schlangen. 2015. Recurrent neural networks for incremental disfluency detection. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Mark Johnson and Eugene Charniak. 2004. A tag-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 33. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Matthew Lease and Mark Johnson. 2006. Early deletion of fillers in processing conversational speech. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 73–76. Association for Computational Linguistics.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association* *for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Marie W Meteer, Ann A Taylor, Robert MacIntyre, and Rukmini Iyer. 1995. *Dysfluency annotation stylebook for the switchboard corpus*. University of Pennsylvania.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Mari Ostendorf and Sangyun Hahn. 2013. A sequential repetition model for improved disfluency detection. In *INTERSPEECH*, pages 2624–2628.

Xian Qian and Yang Liu. 2013. Disfluency detection using multi-step stacked learning. In *HLT-NAACL*, pages 820–825.

Mohammad Sadegh Rasooli and Joel R Tetreault. 2013. Joint parsing and disfluency detection in linear time. In *EMNLP*, pages 124–129.

Elizabeth Ellen Shriberg. 1994. *Preliminaries to a theory of speech disfluencies*. Ph.D. thesis, Citeseer.

Shaolei Wang, Wanxiang Che, and Ting Liu. 2016. A neural attention model for disfluency detection. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 278–287, Osaka, Japan. The COLING 2016 Organizing Committee.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional lstm. In *Proc. of ACL*, pages 2306–2315.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1169–1179, Beijing, China. Association for Computational Linguistics.

Shuangzhi Wu, Dongdong Zhang, Ming Zhou, and Tiejun Zhao. 2015. Efficient disfluency detection with transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 495–503. Association for Computational Linguistics.

Qingyu Yin, Yu Zhang, Weinan Zhang, and Ting Liu. 2017. A deep neural network for chinese zero pronoun resolution. In *Proceedings of the 26th International Conference on Artificial Intelligence*, IJCAI'17. AAAI Press.

Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency detection using a bidirectional lstm. *arXiv preprint arXiv:1604.03209*.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2013. Chinese parsing exploiting characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 125–134, Sofia, Bulgaria. Association for Computational Linguistics.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Character-level chinese dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1326–1336, Baltimore, Maryland. Association for Computational Linguistics.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-based neural word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 421–431, Berlin, Germany. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA. Association for Computational Linguistics.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1213–1222, Beijing, China. Association for Computational Linguistics.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443, Sofia, Bulgaria. Association for Computational Linguistics.

Neural Sequence-Labelling Models for Grammatical Error Correction

Helen Yannakoudakis, **Marek Rei**, **Øistein E. Andersen** and **Zheng Yuan**
The ALTA Institute
Computer Laboratory
University of Cambridge
{hy260,mr472,oa223,zy249}@cl.cam.ac.uk

Abstract

We propose an approach to N-best list re-ranking using neural sequence-labelling models. We train a compositional model for error *detection* that calculates the probability of each token in a sentence being correct or incorrect, utilising the full sentence as context. Using the error detection model, we then re-rank the N best hypotheses generated by statistical machine translation systems. Our approach achieves state-of-the-art results on error correction for three different datasets, and it has the additional advantage of only using a small set of easily computed features that require no linguistic input.

1 Introduction

Grammatical Error Correction (GEC) in non-native text attempts to automatically detect and correct errors that are typical of those found in learner writing. High precision and good coverage of learner errors is important in the development of GEC systems. Phrase-based Statistical Machine Translation (SMT) approaches to GEC have attracted considerable attention in recent years as they have been shown to achieve state-of-the-art results (Felice et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2016). Given an ungrammatical input sentence, the task is formulated as "translating" it to its grammatical counterpart. Using a parallel dataset of input sentences and their corrected counterparts, SMT systems are typically trained to correct all error types in text without requiring any further linguistic input. To further adapt SMT approaches to the task of GEC and tackle the paucity of error-annotated learner data, previous work has investigated a number of extensions, ranging from the addition of further features into the decoding process (Felice et al., 2014) via re-ranking the SMT decoder's output (Yuan et al., 2016) to neural-network adaptation components to SMT (Chollampatt et al., 2016a).

In this paper, we propose an approach to N-best list re-ranking using neural sequence-labelling models. N-best list re-ranking allows for fast experimentation since the decoding process remains unchanged and only needs to be performed once. Crucially, it can be applied to any GEC system that can produce multiple alternative hypotheses. More specifically, we train a neural compositional model for error *detection* that calculates the probability of each token in a sentence being correct or incorrect, utilising the full sentence as context. Using the error detection model, we then re-rank the N best hypotheses generated by the SMT system. Detection models can be more fine-tuned to finer nuances of grammaticality and acceptability, and therefore better able to distinguish between correct and incorrect versions of a sentence.

Our approach achieves state-of-the-art results on GEC for three different datasets, and it has the additional advantage of using only a small set of easily computed features that require no linguistic information, in contrast to previous work that has utilised a large set of features in a supervised setting (Hoang et al., 2016; Yuan et al., 2016).

2 Previous work

The first approaches to GEC primarily treat the task as a classification problem over vectors of contextual lexical and syntactic features extracted from a fixed window around the target token. A large body of work has investigated error-type-specific models, and in particular models targeting preposition and article errors, which are among the most frequent ones in non-native English learner writing (Chodorow et al., 2007; De Felice and Pul-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2795–2806
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

man, 2008; Han et al., 2010; Tetreault et al., 2010; Han et al., 2006; Tetreault and Chodorow, 2008; Gamon et al., 2008; Gamon, 2010; Rozovskaya and Roth, 2010; Rozovskaya et al., 2012; Dale and Kilgarriff, 2011; Leacock et al., 2014). Core components of one of the top systems in the CoNLL 2013 and 2014 shared tasks on GEC (Ng et al., 2013, 2014) include Averaged Perceptron classifiers, native-language error correction priors in Naive Bayes models, and joint inference frameworks capturing interactions between errors (e.g., noun number and verb agreement errors) (Rozovskaya et al., 2012, 2014, 2011; Rozovskaya and Roth, 2011). The power of the classification paradigm comes from its ability to generalise well to unseen examples, without necessarily requiring error-annotated learner data (Rozovskaya and Roth, 2016).

One of the first approaches to GEC as an SMT task is the one by Brockett et al. (2006), who generate artificial data based on hand-crafted rules to train a model that can correct countability errors. Dahlmeier and Ng (2011) focus on correcting collocation errors based on paraphrases extracted from parallel corpora, while Dahlmeier and Ng (2012a) are the first to investigate a discriminatively trained beam-search decoder for full-sentence correction, focusing on five different error types: spelling, articles, prepositions, punctuation insertion, and noun number. Yoshimoto et al. (2013) utilise SMT to tackle determiner and preposition errors, while Yuan and Felice (2013) use POS-factored, phrase-based SMT systems, trained on both learner and artificially generated data to tackle determiner, preposition, noun number, verb form, and subject–verb agreement errors. The SMT approach has better capacity to correct complex errors, and it only requires parallel corrected sentences as input.

Two state-of-the-art systems in the 2014 CoNLL shared task on correction of all errors regardless of type use SMT systems: Felice et al. (2014) use a hybrid approach that includes a rule-based and an SMT system augmented by a large web-based language model and combined with correction-type estimation to filter out error types with zero precision. Junczys-Dowmunt and Grundkiewicz (2016) investigate parameter tuning based on the MaxMatch (M^2) scorer, the shared-task evaluation metric (Dahlmeier and Ng, 2012b; Ng et al., 2014), and experiment with different optimisers and interactions of dense and sparse features.

Susanto et al. (2014) and Rozovskaya and Roth (2016) explore combinations of SMT systems and classifiers, the latter showing substantial improvements over the CoNLL state of the art. Chollampatt et al. (2016a) integrate a neural network joint model that has been adapted using native-language-specific learner text as a feature in SMT, while Chollampatt et al. (2016b) integrate a neural network global lexicon model and a neural network joint model to exploit continuous space representations of words rather than discrete ones, and learn non-linear mappings. Yuan and Briscoe (2016) present a Neural Machine Translation (NMT) model and propose an approach that tackles the rare-word problem in NMT.

Yuan et al. (2016) and Mizumoto and Matsumoto (2016) employ supervised discriminative methods to re-rank the SMT decoder's N-best list output based on language model and syntactic features respectively. Hoang et al. (2016) also exploit syntactic features in a supervised framework, but further extend their approach to generate new hypotheses. Our approach is similar in spirit, but differs in the following aspects: inspired by the work of Rei and Yannakoudakis (2016) who tackle error *detection* rather than *correction* within a neural network framework, we develop a neural sequence-labelling model for error *detection* to calculate the probability of each token in a sentence as being correct or incorrect; using the error detection model, we propose a small set of features that require no linguistic processing to re-rank the N best hypotheses. We evaluate our approach on three different GEC datasets and achieve state-of-the-art results, outperforming all previous approaches to GEC.

3 Datasets

We use the First Certificate in English (FCE) dataset (Yannakoudakis et al., 2011), and the NUS Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013) that was used in the CoNLL GEC shared tasks. Both datasets are annotated with the language errors committed and suggested corrections from expert annotators. The former consists of upper-intermediate learner texts written by speakers from a number of different native language backgrounds, while the latter consists of essays written by advanced undergraduate university students from an Asian language background. We use the public FCE train/test split, and the NUCLE train/test set used in CoNLL 2014 (the test set has been annotated by two different annotators).

We also use the publicly available Lang-8 corpus (Mizumoto et al., 2012; Tajiri et al., 2012) and the JHU FLuency-Extended GUG corpus (J-FLEG) (Napoles et al., 2017). Lang-8 contains learner English from lang-8.com, a language-learning social networking service, which has been corrected by native speakers. JFLEG is a newly released corpus for GEC evaluation that contains fluency edits to make the text more native-like in addition to correcting grammatical errors, and contains learner data from a range of proficiency levels.

We use Lang-8 and the FCE and CoNLL training sets to train our neural sequence-labelling model, and test correction performance on JFLEG, and the FCE and CoNLL test sets. For JFLEG, we use the 754 sentences on which Napoles et al. (2017) have already benchmarked four leading GEC systems. As our development set, we use a subset of the FCE training data.

4 Neural sequence labelling

We treat error detection as a sequence labelling task and assign a label to each token in the input sentence, indicating whether it is correct or incorrect in context. These binary gold labels can be automatically extracted from the manual error annotation available in our data (see Section 3). Similarly to Rei and Yannakoudakis (2016), we construct a bidirectional recurrent neural network for detecting writing errors. The system receives a series of tokens $[w_1 ... w_T]$ as input, and predicts a probability distribution over the possible labels for each token.

Every token w_t is first mapped to a token representation $\widetilde{x}_t$, which is also optimised during training. These embeddings are composed together into context-specific representations using a bidirectional LSTM (Hochreiter and Schmidhuber, 1997):

$$\overrightarrow{h_t} = \text{LSTM}(\widetilde{x}_t, \overrightarrow{h_{t-1}}) \tag{1}$$

$$\overleftarrow{h_t} = \text{LSTM}(\widetilde{x}_t, \overleftarrow{h_{t+1}}) \tag{2}$$

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{3}$$

where $\widetilde{x}_t$ is the token representation at position t, $\overrightarrow{h_t}$ is the hidden state of the forward-moving LSTM, $\overleftarrow{h_t}$ is the hidden state of the backward-moving LSTM, and h_t is the concatenation of both hidden states. A feedforward hidden layer with tanh activation is then used to map the representations from both directions into a more suitable combined space, and allow the model to learn higher-level features:

$$d_t = \tanh W_d h_t \tag{4}$$

where W_d is a weight matrix. Finally, a softmax output layer predicts the label distribution for each token, given the input sequence:

$$P(y_t | w_1 ... w_T) = \text{softmax}\, W_o d_t \tag{5}$$

where W_o is an output weight matrix.

We also make use of the character-level architecture proposed by Rei et al. (2016), allowing the model to learn morphological patterns and capture out-of-vocabulary words. Each individual character is mapped to a character embedding and a bidirectional LSTM is used to combine them together into a character-based token representation. This vector m, constructed only from individual characters, is then combined with the regular token embedding x_t using an adaptive gating mechanism:

$$z = \sigma\big(W_{z_1} \cdot \tanh(W_{z_2} x_t + W_{z_3} m)\big) \tag{6}$$

$$\widetilde{x}_t = z \cdot x_t + (1 - z) \cdot m \tag{7}$$

where W_{z_1}, W_{z_2} and W_{z_3} are weight matrices, z is a dynamically calculated gating vector, and $\widetilde{x}_t$ is the resulting token representation at position t.

We optimise the model by minimising cross-entropy between the predicted label distributions and the annotated labels. In addition to training the error detection objective, we make use of a multi-task loss function and train specific parts of the architecture as language models. This provides the model with a more informative loss function, while also encouraging it to learn more general compositional features and acting as a regulariser (Rei, 2017). First, two extra hidden layers are constructed:

$$\overrightarrow{m_t} = \tanh \overrightarrow{W}_m \overrightarrow{h_t} \tag{8}$$

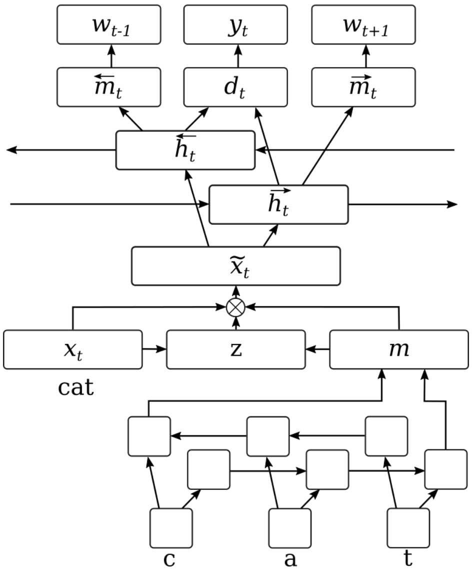


Figure 1: Error detection network architecture that is repeated for all the words in a sentence (illustration for the word "cat").

$$\overleftarrow{m_t} = \tanh \overleftarrow{W}_m \overleftarrow{h_t} \qquad (9)$$

where $\overrightarrow{W}_m$ and $\overleftarrow{W}_m$ are direction-specific weight matrices, used for connecting a forward or backward LSTM hidden state to a separate layer. The surrounding tokens are then predicted based on each hidden state using a softmax output layer:

$$P(w_{t+1}|w_1...w_t) = \text{softmax}\, \overrightarrow{W}_q \overrightarrow{m_t} \qquad (10)$$

$$P(w_{t-1}|w_t...w_T) = \text{softmax}\, \overleftarrow{W}_q \overleftarrow{m_t} \qquad (11)$$

During training, the following cost function is minimised, which combines the error detection loss function with the two language modeling objectives:

$$\begin{aligned} E = &- \sum_{t=1}^{T} \log P(y_t|w_t...w_T) \\ &- \gamma \sum_{t=1}^{T-1} \log P(w_{t+1}|w_1...w_t) \qquad (12) \\ &- \gamma \sum_{t=2}^{T} \log P(w_{t-1}|w_t...w_T) \end{aligned}$$

where γ is a weight that controls the importance of language modeling in relation to the error detection objective. Figure 1 shows the error detection network architecture.

4.1 Experimental settings

All digits in the text are replaced with the character '0'. Tokens that occur less than 2 times in the training data share an out-of-vocabulary (OOV) token embedding, whereas the character-level component still operates over the original tokens. The model hyperparameters are tuned based on $F_{0.5}$ on the FCE development set (Section 3) and γ is set to 0.1.[1] The model is optimised using Adam (Kingma and Ba, 2015), and training is stopped when $F_{0.5}$ does not improve on the development set over 5 epochs. Token representations have size 300 and are initialised with pre-trained word2vec embeddings trained on Google News (Mikolov et al., 2013). The character representations have size 50 and are initialised randomly. The LSTM hidden layers have size 200 for each direction.

4.2 Error detection performance

Rei and Yannakoudakis (2016)'s error detection framework uses token-level embeddings, bidirectional LSTMs for context representation, and a multi-layer architecture for learning more complex features. They train their model on the public FCE training set, and benchmark their results on the FCE and CoNLL test sets (Baseline LSTM_{FCE}). We also train and test our detection model on the same data and evaluate the effectiveness of our approach (LSTM_{FCE}). In Table 1, we can see that our architecture achieves a higher performance on both FCE and CoNLL, and particularly for FCE (7% higher $F_{0.5}$) and CoNLL test annotation 2 (around 2% higher $F_{0.5}$). When we use a larger training set that also includes the CoNLL training data and the public Lang-8 corpus (see Section 3), performance improves even further (LSTM), particularly for CoNLL test annotation 1 (at least 8% higher $F_{0.5}$ compared to LSTM_{FCE}). We use this model in the experiments reported in the following sections.

5 Statistical machine translation

SMT attempts to identify the 1-best correction hypothesis c^* of an input sentence s that maximises the following:

$$c^* = \arg\max_c p_{\text{LM}}(c)\, p(s|c) \qquad (13)$$

[1] Lower γ values tend to give better error detection results as this essentially prioritises the error detection objective.

	FCE test set			CoNLL test set annotation 1			CoNLL test annotation 2		
System	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Baseline LSTM_{FCE}	46.10	28.50	41.10	15.40	22.80	16.40	23.60	25.10	23.90
LSTM_{FCE}	58.88	28.92	48.48	17.68	19.07	17.86	27.62	21.18	25.88
LSTM	79.10	21.19	**51.14**	54.51	8.69	**26.53**	69.60	7.91	**27.18**

Table 1: Token-level error detection performance of our detection models (LSTM_{FCE} and LSTM) on FCE and the two CoNLL 2014 test set annotations. Baseline LSTM_{FCE} and LSTM_{FCE} are trained only on the public FCE training set.

A Language Model (LM) is used to estimate the correction hypothesis probability $p_{\text{LM}}(c)$ from a corpus of correct English, and a translation model to estimate the conditional $p(s|c)$ from a parallel corpus of corrected learner sentences. State-of-the-art SMT systems are phrase-based (Koehn et al., 2003) in that they use phrases as "translation" units and therefore allow many-to-many "translation" mappings. The translation model is decomposed into a phrase-translation probability model and a phrase re-ordering probability model, and the 1-best correction hypothesis is of the following log-linear form (Och and Ney, 2002):

$$c^* = \arg\max_c \exp \sum_{i=1}^{K} \lambda_i \, h_i(c, s) \tag{14}$$

where h represents a feature function (e.g., phrase-translation probability) and λ the feature weight.

In this work, we employ two SMT systems: Yuan et al. (2016)[2] and Junczys-Dowmunt and Grundkiewicz (2016). We apply our re-ranking approach to each SMT system's N-best list using features derived from the neural sequence-labelling model for error detection described in the previous section, improve each of the SMT systems, and achieve state-of-the-art results on all three GEC datasets: FCE, CoNLL and JFLEG.

5.1 N-best list re-ranking

For each SMT system, we generate the list of all the 10 best candidate hypotheses. We then use the following set of features (tuned on the FCE development set, see Section 3) to assign a score to each candidate, and determine a new ranking for each SMT model:

Sentence probability: Our error detection model outputs a probabilty indicating whether a token is likely to be correct or incorrect in context. We therefore use as a feature the overall sentence probability, calculated based on the probability of each of its tokens being correct: $\sum_w \log P(w)$

Levenshtein distance: We first use Levenshtein distance (LD) to identify which tokens in the original/source sentence have been corrected by the candidate hypothesis. We then identify the tokens that our detection model predicts as incorrect (i.e., the probability of being incorrect is greater than 0.5). These give us two different sets of annotations for the source sentence: tokens in the source sentence that the candidate hypothesis identifies as incorrect; and tokens in the source sentence that the error detection model identifies as incorrect. We then convert these annotations to binary sequences – i.e., 1 if the token is identified as incorrect, and 0 otherwise – and use as a feature the LD between those binary representations. More specifically, we would like to select the candidate sentence that has the smallest LD from the binary sequence created by the detection model: $\frac{1}{\text{LD}}$

True and false positives: Given the binary sequences described above, we also use as a feature the ratio of true positives (TP) to false positives (FP) by treating the error detection model as the "gold standard". Specifically, we count how many times the candidate hypothesis agrees or not with the detection model on the tokens identified as incorrect: $\frac{\text{TP}}{\text{FP}}$

We use a linear combination of the above three scores together with the overall score (i.e., original rank) given by the SMT system (we do not include any other SMT features) to re-rank each SMT system's 10-best list in an unsupervised way. The new 1-best correction hypothesis c^* is then the one that maximises:

$$c^* = \arg\max_c \sum_{i=1}^{K} \lambda_i \, h_i(c) \tag{15}$$

[2] Yuan et al. (2016) propose a supervised N-best list re-ranking approach; however, we only use their baseline SMT system.

	FCE test set				CoNLL test set				JFLEG			
	P	R	F$_{0.5}$	GLEU	P	R	F$_{0.5}$	GLEU	P	R	F$_{0.5}$	GLEU
Baseline												
CAMB16$_{\text{SMT}}$	63.27	31.95	52.90	70.15	45.39	21.82	37.33	64.90	65.56	29.12	52.44	46.10
Our work												
CAMB16$_{\text{SMT}}$ + LSTM	**65.03**	32.45	54.15	70.72	49.58	21.84	39.53	65.68	**65.86**	30.56	53.50	46.74
CAMB16$_{\text{SMT}}$ + LSTM$_{\text{camb}}$	64.25	**36.13**	**55.60**	**71.76**	**51.09**	**25.30**	**42.44**	**66.42**	65.41	**32.97**	**54.66**	**47.72**
Oracle	80.53	49.62	71.60	78.54	68.77	35.90	58.13	70.42	73.45	38.03	61.92	50.64
Baseline												
AMU16$_{\text{SMT (reported)}}$	–	–	–	–	**61.27**	27.98	49.49	–	–	–	43.20	41.70
AMU16$_{\text{SMT (replicated)}}$	**46.94**	13.75	31.66	63.73	61.15	27.84	49.34	68.23	**69.22**	18.56	44.77	41.98
Our work												
AMU16$_{\text{SMT (replicated)}}$ + LSTM	40.67	17.36	32.06	63.57	58.79	30.63	49.66	68.26	60.68	22.65	45.43	42.65
AMU16$_{\text{SMT (replicated)}}$ + LSTM$_{\text{camb}}$	43.34	**19.88**	**35.07**	**64.78**	59.88	**32.16**	**51.08**	**68.69**	64.12	**25.06**	**48.88**	**43.26**
Oracle	71.54	26.69	53.54	69.52	76.47	35.97	62.41	71.18	79.10	27.47	57.49	45.00
Other baselines												
VT16$_{\text{SMT}}$ + classifiers	–	–	–	–	60.17	25.64	47.40	–	–	–	–	–
NUS16$_{\text{SMT+NNJM}}$	–	–	–	–	–	–	44.27	–	–	–	52.70	46.30
NUS16$_{\text{SMT}}$ + re-ranker	–	–	–	–	50.35	23.84	41.19	–	–	–	–	–
CAMB16$_{\text{NMT}}$	–	–	53.49	71.16	–	–	39.90	65.59	–	–	50.80	47.20

Table 2: Using the neural sequence-labelling model for error detection ('+ LSTM' or '+ LSTM$_{\text{camb}}$') to re-rank the 10-best lists of two SMT systems – Yuan et al. (2016) (CAMB16$_{\text{SMT}}$) and Junczys-Dowmunt and Grundkiewicz (2016) (AMU16$_{\text{SMT}}$).

where h represents the score assigned to candidate hypothesis c according to feature i; λ is a parameter that controls the effect feature i has on the final ranking; and $K = 4$ as we have four different features (three features presented in this section, plus the original score output by the SMT system). λs are tuned on the FCE development set and are set to 1, except for the sentence probability feature which has $\lambda = 1.5$.[3]

6 Evaluation

We evaluate the effectiveness of our re-ranking approach on three different datasets: FCE, CoNLL 2014 and JFLEG. We report F$_{0.5}$ using the shared task's M^2 scorer (Dahlmeier and Ng, 2012b), and GLEU scores (Napoles et al., 2015). The latter is based on a variant of BLEU (Papineni et al., 2002) that is designed to reward correct edits and penalise ungrammatical ones. As mentioned in Section 5, we re-rank the 10-best lists of two SMT systems: Yuan et al. (2016) (CAMB16$_{\text{SMT}}$) and Junczys-Dowmunt and Grundkiewicz (2016) (AMU16$_{\text{SMT}}$). The results are presented in Table 2.

We replicate the AMU16$_{\text{SMT}}$ system to obtain the 10-best output, and report results using this version (AMU16$_{\text{SMT (replicated)}}$). Compared to the original results on CoNLL reported in their paper (AMU16$_{\text{SMT (reported)}}$), we obtain slightly lower performance.[4] We can see that AMU16$_{\text{SMT}}$ is the current state of the art on CoNLL, with an F$_{0.5}$ of 49.49. On the other hand, CAMB16$_{\text{SMT}}$ generalises better on FCE and JFLEG: 52.90 and 52.44 F$_{0.5}$ respectively. The lower performance of AMU16$_{\text{SMT}}$ can be attributed to the fact that it is tuned for the CoNLL shared task.

The current state of the art on FCE is a neural machine translation system, CAMB16$_{\text{NMT}}$ (Yuan and Briscoe, 2016), which is also the best model on JFLEG in terms of GLEU. The rest of the baselines we report are: Rozovskaya and Roth (2016), who explore combinations of SMT systems and classifiers (VT16$_{\text{SMT}}$ + classifiers); Chollampatt et al. (2016a), who integrate a neural network joint model that has been adapted using native-language-specific learner text as a feature in SMT (NUS16$_{\text{SMT+NNJM}}$); and Hoang et al. (2016), who perform supervised N-best list re-ranking using a large set of features, and further extend their approach to generate new hypotheses (NUS16$_{\text{SMT}}$ + re-ranker).[5]

[3] We experimented with a small set of values (from 0 to 2 with increments of .1), though not exhaustively.

[4] The differences are likely to be caused by different versions of the NLTK tokeniser and/or Moses.

[5] We note that Napoles et al. (2017) use an updated version of GLEU to evaluate AMU16$_{\text{SMT (reported)}}$, NUS16$_{\text{SMT+NNJM}}$ and CAMB16$_{\text{NMT}}$ on JFLEG. We therefore also use this updated version throughout all GLEU evaluations on JFLEG.

CAMB16$_{\text{SMT}}$ + LSTM$_{\text{camb}}$		
Ablated feature	F$_{0.5}$	GLEU
None	55.60	71.76
Sentence probability	54.13	70.65
Levenshtein distance	55.42	71.78
True/false positives	55.14	71.75

Table 3: Ablation tests on the FCE test set when removing one feature of the re-ranking system at a time.

When using our LSTM detection model to re-rank the 10-best list (+ LSTM), we can see that performance improves across all three datasets for both SMT systems. F$_{0.5}$ performance of CAMB16$_{\text{SMT}}$ on FCE improves from 52.90 to 54.15, on CoNLL from 37.33 to 39.53, and on JFLEG from 52.44 to 53.50 (the latter demonstrating that the detection model also helps with fluency edits). This improved result is also better than the state of the art CAMB16$_{\text{NMT}}$ on FCE.[6] When looking at AMU16$_{\text{SMT}}$, we can see that re-ranking (+ LSTM) further improves the best result on CoNLL from 49.34 (replicated) to 49.66 F$_{0.5}$, and there is a similar level of improvement for both FCE and JFLEG.

As a further experiment, we re-train our error detection model on the same training data as CAMB16$_{\text{SMT}}$ (+ LSTM$_{\text{camb}}$). More specifically, we use the Cambridge Learner Corpus (CLC) (Nicholls, 2003), a collection of learner texts of various proficiency levels, written in response to exam prompts and manually annotated with the errors committed (around 2M sentence pairs). In Table 2, we can see that the detection model further improves performance across all datasets and SMT systems. Compared to just doing SMT with CAMB16$_{\text{SMT}}$, re-ranking improves F$_{0.5}$ from 52.90 to 55.60 on FCE (performance increases further even though CAMB16$_{\text{SMT}}$'s training set includes a large set of FCE data), from 37.33 to 42.44 on CoNLL, and from 52.44 to 54.66 on JFLEG. The largest improvement is on CoNLL (5%), which is likely because CoNLL is not included in the training set. AMU16$_{\text{SMT (replicated)}}$ is specifically tuned for CoNLL; nevertheless, the detection model also improves F$_{0.5}$ on CoNLL from 49.34 to 51.08. Re-ranking using a small set of detection-based features produces state-of-the-art results on all three datasets (we note that CAMB16$_{\text{SMT}}$ generalises better across all).

We next run ablation tests to investigate the extent to which each feature contributes to performance. Results obtained on the FCE test set after excluding each of the features of the 'CAMB16$_{\text{SMT}}$ + LSTM$_{\text{camb}}$' re-ranking system are presented in Table 3. Overall, all features have a positive effect on performance, though the sentence probability feature does have the biggest impact: its removal is responsible for a 1.47 and 1.11 decrease of F$_{0.5}$ and GLEU respectively. A similar pattern is observed on the other datasets too.

6.1 Oracle

To calculate an upper bound per SMT system per dataset, we calculate character-level LD between each candidate hypothesis in the 10-best list and the gold corrected sentence. We then calculate an oracle score by selecting the candidate hypothesis that has the smallest LD. Essentially the oracle is telling us the maximum performance that can be obtained with the given 10-best list on each dataset. For datasets for which we have more than one annotation available, we select the oracle that gives the highest F$_{0.5}$.

In Table 2, we can see that, overall, CAMB16$_{\text{SMT}}$ has a higher oracle performance compared to AMU16$_{\text{SMT}}$. More specifically, the maximum attainable F$_{0.5}$ on FCE is 71.60, on CoNLL 58.13, and on JFLEG 61.92. This shows empirically that the 10-best list has great potential and should be exploited further. AMU16$_{\text{SMT}}$ has a lower oracle performance overall, though again this can be attributed to the fact that it is specifically tuned for CoNLL.

6.2 N-best list size

Next, we examine performance as the N-best list varies in size, ranging from 1 to 10 (Table 4). We observe a positive effect: the larger the size, the better the model for all datasets. F$_{0.5}$ does not seem to have reached a plateau with $n < 10$, which suggests that increasing the size of the list further can potentially lead to better results. We do, however, observe that large improvements are obtained when increasing the size from 1 to 3, suggesting that, most of the time, better alternatives are identified within the top 3 candidate hypotheses. This, however, is not the case for the oracle ($\text{F}_{0.5}^{\text{oracle}}$), which consistently increases as n gets larger.

[6] We note that CAMB16$_{\text{NMT}}$ outperforms the re-ranking approach by Yuan et al. (2016).

	CAMB16$_{\text{SMT}}$					
	FCE test set		CoNLL test set		JFLEG	
N-best list	$F_{0.5}$	$F_{0.5}^{\text{oracle}}$	$F_{0.5}$	$F_{0.5}^{\text{oracle}}$	$F_{0.5}$	$F_{0.5}^{\text{oracle}}$
1	52.90	52.90	37.33	37.33	52.44	52.44
2	54.28	62.10	39.87	47.03	53.04	55.64
3	54.96	65.93	41.00	51.12	53.24	57.69
4	55.18	68.05	41.83	53.13	53.47	58.93
5	55.33	69.47	42.12	54.80	54.01	59.66
6	55.43	70.24	42.49	55.53	54.18	60.34
7	55.48	70.74	42.60	56.54	54.45	61.14
8	55.47	71.00	42.63	57.04	54.47	61.28
9	55.51	71.27	**42.65**	57.23	54.64	61.55
10	**55.60**	71.60	42.44	58.13	**54.66**	61.92

Table 4: Re-ranking performance using LSTM$_{\text{camb}}$ as the N-best list varies in size from 1 to 10 for CAMB16$_{\text{SMT}}$ and its oracle.

Source	I work with children an the Computer help my Jop bat affeted to
CAMB16$_{SMT}$	I work with children and the Computer help my Jop bat affeted to
CAMB16$_{SMT}$ + LSTM$_{camb}$	I work with children and the **computer helps** my Jop bat affeted to
Reference	I work with children and the computer helps me in my job but affects it too
Source	It takes 25 minutes that is convenient to us
CAMB16$_{SMT}$	It takes 25 minutes that is convenient for us
CAMB16$_{SMT}$ + LSTM$_{camb}$	It takes 25 minutes **, which** is convenient for us
Reference	It takes 25 minutes , which is convenient for us
Source	I recommend to visit
CAMB16$_{SMT}$	I recommend you to visit
CAMB16$_{SMT}$ + LSTM$_{camb}$	I recommend **visiting**
Reference	I recommend visiting it
Source	Especially youngsters misuse this kind of invention
CAMB16$_{SMT}$	Especially youngsters misuse this kind of invention
CAMB16$_{SMT}$ + LSTM$_{camb}$	**In particular ,** youngsters misuse this kind of invention
Reference	Especially youngsters misuse this kind of invention

Table 5: Source sentences along with gold corrections (reference), 1-best candidates by CAMB16$_{\text{SMT}}$ and by CAMB16$_{\text{SMT}}$ + LSTM$_{\text{camb}}$.

6.3 Error type performance

In Table 5, we can see example source sentences, together with their corrected counterparts (reference), 1-best candidates by CAMB16$_{\text{SMT}}$ and 1-best candidates by CAMB16$_{\text{SMT}}$ + LSTM$_{\text{camb}}$. Re-ranking seems to fix errors such as subject–verb agreement ("the Computer help" to "the computer helps") and verb form ("I recommend you to visit" to "I recommend visiting"). In this section, we perform an analysis of performance per type to get a better understanding of where the strength of the re-ranking detection model comes from.

Until recently, GEC performance per error type was only analysed in terms of recall, as system output is not annotated. Recently, however, Bryant et al. (2017) proposed an approach to automatically annotating GEC output with error type information, which utilises a linguistically-enhanced alignment to automatically extract the edits between pairs of source sentences and their corrected counterparts, and a dataset-independent rule-based classifier to classify the edits into error types. Human evaluation showed that the predicted error types were rated as "Good" or "Acceptable" 95% of the time. We use their publicly available code to analyse per-error-type performance before and after re-ranking.

Table 6 presents the performance for a subset of error types that are affected the most before and after re-ranking CAMB16$_{\text{SMT}}$ on the FCE test set. The error types are interpreted as follows: **M**issing error; **R**eplace error; **U**nnecessary error. The largest improvement is observed in replacement errors referring to possessive nouns (R:NOUN:POSS) and verb agreement (R:VERB:SVA); and in unnecessary errors referring to adverbs (U:ADV), determiners (U:DET), pronouns (U:PRON), and verb tense (U:VERB:TENSE).

The LSTM architecture allows the network to learn advanced composition rules and remember dependencies over longer distances (e.g., R:VERB:SVA improves from 58.38 to 69.40). The network's language modelling objectives allow it to learn better and more general compositional features (e.g., U:ADV improves from 13.51 to 22.73), while the character-level architecture facilitates modelling of morphological patterns [e.g., replacement errors referring to verb form (R:VERB:FORM) improve from 53.62 to 58.06]. Between M, R, and U errors, the largest improvement is observed in U, for which there is at least 5% improvement in $F_{0.5}$.[7]

Overall, re-ranking improves $F_{0.5}$ across error types; however, there is a small subset that is

[7] U improves from 38.44 to 43.77; M from 43.43 to 45.40; R from 53.25 to 55.33.

	CAMB16$_{\text{SMT}}$	CAMB16$_{\text{SMT}}$ + LSTM$_{\text{camb}}$
Type	$F_{0.5}$	$F_{0.5}$
M:ADV	25.00	31.25
M:VERB	25.42	29.85
R:NOUN:NUM	56.60	62.50
R:NOUN:POSS	35.71	**55.56**
R:OTHER	34.99	38.75
R:PRON	26.88	33.33
R:VERB:FORM	53.62	58.06
R:VERB:SVA	58.38	**69.40**
R:VERB:TENSE	31.94	36.29
U:ADV	13.51	**22.73**
U:DET	46.27	**55.30**
U:NOUN	10.10	15.72
U:PREP	47.62	53.40
U:PRON	30.77	**39.33**
U:PUNCT	51.22	58.38
U:VERB:TENSE	28.41	**41.67**
M:PREP	43.69	39.43
M:VERB:FORM	50.00	**38.46**
R:ADJ	45.45	37.67
R:CONTR	50.00	**27.78**
R:WO	53.63	48.74

Table 6: Error-type performance before and after re-ranking on the FCE test set (largest impact highlighted in bold; bottom part of the table displays negative effects on performance).

negatively affected (Table 6, bottom part); for example, performance on missing errors referring to verb form (M:VERB:FORM) drops from 50.00 to 38.46, and on replace contraction errors (R:CONTR) from 50.00 to 27.78. Importantly, such an analysis allows us to examine the strengths and weaknesses of the models, which is key for the deployment of GEC systems.

7 Conclusion

To the best of our knowledge, no prior work has investigated the impact of detection models on correction performance. We proposed an approach to N-best list re-ranking using a neural sequence-labelling model that calculates the probability of each token in a sentence being correct or incorrect in context. Detection models can be more fine-tuned to finer nuances of grammaticality, and therefore better able to distinguish between correct and incorrect versions of a sentence. Using a linear combination of a small set of features derived from the detection model output, we re-ranked the N-best list of SMT systems and achieved state-of-the-art results on GEC on three different datasets. Our approach can be applied to any GEC system that produces multiple alternative hypotheses. Our results demonstrate the benefits of integrating detection approaches with correction systems, and how one can complement the other.

Acknowledgments

Special thanks to Christopher Bryant, Mariano Felice, and Ted Briscoe, as well as the anonymous reviewers for their valuable contributions at various stages.

References

Chris Brockett, William B Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 249–256. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Martin Chodorow, Joel R Tetreault, and Na-Rae Han. 2007. Detection of grammatical errors involving prepositions. In *Proceedings of the fourth ACL-SIGSEM workshop on prepositions*, pages 25–30. Association for Computational Linguistics.

Shamil Chollampatt, Duc Tam Hoang, and Hwee Tou Ng. 2016a. Adapting grammatical error correction based on the native language of writers with neural network joint models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1901–1911.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016b. Neural network translation models for grammatical error correction. *arXiv preprint arXiv:1606.00189*.

Daniel Dahlmeier and Hwee Tou Ng. 2011. Correcting semantic collocation errors with L1-induced paraphrases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 107–117. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. 2012a. A beam-search decoder for grammatical error correction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 568–578. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. 2012b. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 242–249. Association for Computational Linguistics.

Rachele De Felice and Stephen G Pulman. 2008. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 169–176. Association for Computational Linguistics.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the 18th Conference on Computational Natural Language Learning: Shared Task*, pages 15–24. Association for Computational Linguistics.

Michael Gamon. 2010. Using mostly native data to correct errors in learners' writing: a meta-classifier approach. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 163–171. Association for Computational Linguistics.

Michael Gamon, Jianfeng Gao, Chris Brockett, Alexandre Klementiev, William B Dolan, Dmitriy Belenko, and Lucy Vanderwende. 2008. Using contextual speller techniques and language modeling for ESL error correction. In *IJCNLP*, volume 8, pages 449–456.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Natural Language Engineering*, 12(1):115–129.

Na-Rae Han, Joel Tetreault, Soo-Hwa Lee, and Jin-Young Ha. 2010. Using error-annotated ESL data to develop an ESL error correction system. In *Proceedings of LREC. Emi Izumi, Kiyotaka Uchimoto and Hitoshi Isahara*.

Duc Tam Hoang, Shamil Chollampatt, and Hwee Tou Ng. 2016. Exploiting n-best hypotheses to improve an smt approach to grammatical error correction. *arXiv preprint arXiv:1606.00210*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. *arXiv preprint arXiv:1605.06353*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: a method for stochastic optimization. In *International Conference on Learning Representations*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54. Association for Computational Linguistics.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2014. Automated grammatical error detection for language learners, second edition. *Synthesis lectures on human language technologies*, 7(1):1–170.

Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR 2013)*.

Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yu Matsumoto. 2012. The effect of learner corpus size in grammatical error correction of ESL writings. In *24th International Conference on Computational Linguistics*, pages 863–872.

Tomoya Mizumoto and Yuji Matsumoto. 2016. Discriminative reranking for grammatical error correction with statistical machine translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 1133–1138.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 588–593.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. *arXiv preprint arXiv:1702.04066*.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12. Association for Computational Linguistics.

Diane Nicholls. 2003. The Cambridge Learner Corpus - error coding and analysis for lexicography and ELT. In *Proceedings of the Corpus Linguistics 2003 Conference*.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Marek Rei, Gamal K. O. Crichton, and Sampo Pyysalo. 2016. Attending to characters in neural sequence labeling models. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING-2016)*.

Marek Rei and Helen Yannakoudakis. 2016. Compositional sequence labeling models for error detection in learner writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, Dan Roth, and Nizar Habash. 2014. The Illinois-Columbia system in the CoNLL-2014 shared task. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 34–42. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2010. Generating confusion sets for context-sensitive error correction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 961–970. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2011. Algorithm selection and model adaptation for ESL correction tasks. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 924–933. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2016. Grammatical error correction: Machine translation and classifiers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2205–2215.

Alla Rozovskaya, Mark Sammons, Joshua Gioja, and Dan Roth. 2011. University of Illinois system in HOO text correction shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 263–266. Association for Computational Linguistics.

Alla Rozovskaya, Mark Sammons, and Dan Roth. 2012. The UI system in the HOO 2012 shared task on error correction. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 272–280. Association for Computational Linguistics.

Raymond Hendy Susanto, Peter Phandi, and Hwee Tou Ng. 2014. System combination for grammatical error correction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 951–962.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 198–202. Association for Computational Linguistics.

Joel Tetreault, Jennifer Foster, and Martin Chodorow. 2010. Using parse features for preposition selection and error detection. In *Proceedings of the ACL 2010 conference short papers*, pages 353–358. Association for Computational Linguistics.

Joel R Tetreault and Martin Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 865–872. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189. Association for Computational Linguistics.

Ippei Yoshimoto, Tomoya Kose, Kensuke Mitsuzawa, Keisuke Sakaguchi, Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2013. NAIST at 2013 CoNLL grammatical error correction shared task. In *CoNLL Shared Task*, pages 26–33.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 380–386.

Zheng Yuan, Ted Briscoe, and Mariano Felice. 2016. Candidate re-ranking for SMT-based grammatical error correction. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 256–266.

Zheng Yuan and Mariano Felice. 2013. Constrained grammatical error correction using statistical machine translation. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 52–61.

Adapting Sequence Models for Sentence Correction

Allen Schmaltz[*] **Yoon Kim** **Alexander M. Rush** **Stuart M. Shieber**

Harvard University
{schmaltz@fas,yoonkim@seas,srush@seas,shieber@seas}.harvard.edu
[*]Part of this work was completed while as an intern at Rakuten.

Abstract

In a controlled experiment of sequence-to-sequence approaches for the task of sentence correction, we find that character-based models are generally more effective than word-based models and models that encode subword information via convolutions, and that modeling the output data as a series of diffs improves effectiveness over standard approaches. Our strongest sequence-to-sequence model improves over our strongest phrase-based statistical machine translation model, with access to the same data, by 6 M^2 (0.5 GLEU) points. Additionally, in the data environment of the standard CoNLL-2014 setup, we demonstrate that modeling (and tuning against) diffs yields similar or better M^2 scores with simpler models and/or significantly less data than previous sequence-to-sequence approaches.

1 Introduction

The task of *sentence correction* is to convert a natural language sentence that may or may not have errors into a corrected version. The task is envisioned as a component of a learning tool or writing-assistant, and has seen increased interest since 2011 driven by a series of shared tasks (Dale and Kilgarriff, 2011; Dale et al., 2012; Ng et al., 2013, 2014).

Most recent work on language correction has focused on the data provided by the CoNLL-2014 shared task (Ng et al., 2014), a set of corrected essays by second-language learners. The CoNLL-2014 data consists of only around 60,000 sentences, and as such, competitive systems have made use of large amounts of corrected text without annotations, and in some cases lower-quality crowd-annotated data, in addition to the shared data. In this data environment, it has been suggested that statistical phrase-based machine translation (MT) with task-specific features is the state-of-the-art for the task (Junczys-Dowmunt and Grundkiewicz, 2016), outperforming word- and character-based sequence-to-sequence models (Yuan and Briscoe, 2016; Xie et al., 2016; Ji et al., 2017), phrase-based systems with neural features (Chollampatt et al., 2016b,a), re-ranking output from phrase-based systems (Hoang et al., 2016), and combining phrase-based systems with classifiers trained for hand-picked subsets of errors (Rozovskaya and Roth, 2016).

We revisit the comparison across translation approaches for the correction task in light of the Automated Evaluation of Scientific Writing (AESW) 2016 dataset, a correction dataset containing over 1 million sentences, holding constant the training data across approaches. The dataset was previously proposed for the distinct binary classification task of grammatical error identification.

Experiments demonstrate that pure character-level sequence-to-sequence models are more effective on AESW than word-based models and models that encode subword information via convolutions over characters, and that representing the output data as a series of *diffs* significantly increases effectiveness on this task. Our strongest character-level model achieves statistically significant improvements over our strongest phrase-based statistical machine translation model by 6 M^2 (0.5 GLEU) points, with additional gains when including domain information. Furthermore, in the partially crowd-sourced data environment of the standard CoNLL-2014 setup in which there are comparatively few professionally annotated sentences, we find that tuning against the tags marking the diffs yields similar or superior effectiveness relative to existing sequence-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2807–2813
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

to-sequence approaches despite using significantly less data, with or without using secondary models. All code is available at https://github.com/allenschmaltz/grammar.

2 Background and Methods

Task We follow recent work and treat the task of sentence correction as translation from a source sentence (the unedited sentence) into a target sentence (a corrected version in the same language as the source). We do not make a distinction between grammatical and stylistic corrections.

We assume a vocabulary $\mathcal{V}$ of natural language word types (some of which have orthographic errors). Given a sentence $\mathbf{s} = [s_1 \cdots s_I]$, where $s_i \in \mathcal{V}$ is the i-th token of the sentence of length I, we seek to predict the corrected target sentence $\mathbf{t} = [t_1 \cdots t_J]$, where $t_j \in \mathcal{V}$ is the j-th token of the corrected sentence of length J. We are given both $\mathbf{s}$ and $\mathbf{t}$ for supervised training in the standard setup. At test time, we are only given access to sequence $\mathbf{s}$. We learn to predict sequence $\mathbf{t}$ (which is often identical to $\mathbf{s}$).

Sequence-to-sequence We explore word and character variants of the sequence-to-sequence framework. We use a standard word-based model (WORD), similar to that of Luong et al. (2015), as well as a model that uses a convolutional neural network (CNN) and a highway network over characters (CHARCNN), based on the work of Kim et al. (2016), instead of word embeddings as the input to the encoder and decoder. With both of these models, predictions are made at the word level. We also consider the use of bidirectional versions of these encoders (+BI).

Our character-based model (CHAR+BI) follows the architecture of the WORD+BI model, but the input and output consist of characters rather than words. In this case, the input and output sequences are converted to a series of characters and whitespace delimiters. The output sequence is converted back to $\mathbf{t}$ prior to evaluation.

The WORD models encode and decode over a closed vocabulary (of the 50k most frequent words); the CHARCNN models encode over an open vocabulary and decode over a closed vocabulary; and the CHAR models encode and decode over an open vocabulary.

Our contribution is to investigate the impact of sequence-to-sequence approaches (including those not considered in previous work) in a series of controlled experiments, holding the data constant. In doing so, we demonstrate that on a large, professionally annotated dataset, the most effective sequence-to-sequence approach can significantly outperform a state-of-the-art SMT system without augmenting the sequence-to-sequence model with a secondary model to handle low-frequency words (Yuan and Briscoe, 2016) or an additional model to improve precision or intersecting a large language model (Xie et al., 2016). We also demonstrate improvements over these previous sequence-to-sequence approaches on the CoNLL-2014 data and competitive results with Ji et al. (2017), despite using significantly less data.

The work of Schmaltz et al. (2016) applies WORD and CHARCNN models to the distinct binary classification task of error identification.

Additional Approaches The standard formulation of the correction task is to model the output sequence as $\mathbf{t}$ above. Here, we also propose modeling the diffs between $\mathbf{s}$ and $\mathbf{t}$. The diffs are provided in-line within $\mathbf{t}$ and are described via tags marking the starts and ends of insertions and deletions, with replacements represented as deletion-insertion pairs, as in the following example selected from the training set: "Some key points are worth <del> emphasiz </del> <ins> emphasizing </ins> .". Here, "emphasiz" is replaced with "emphasizing". The models, including the CHAR model, treat each tag as a single, atomic token.

The diffs enable a means of tuning the model's propensity to generate corrections by modifying the probabilities generated by the decoder for the 4 diff tags, which we examine with the CoNLL data. We include four bias parameters associated with each diff tag, and run a grid search between 0 and 1.0 to set their values based on the tuning set.

It is possible for models with diffs to output invalid target sequences (for example, inserting a word without using a diff tag). To fix this, a deterministic post-processing step is performed (greedily from left to right) that returns to source any non-source tokens outside of insertion tags. Diffs are removed prior to evaluation. We indicate models that *do not* incorporate target diff annotation tags with the designator –DIFFS.

The AESW dataset provides the paragraph context and a journal domain (a classification of the document into one of nine subject categories) for each sentence.[1] For the sequence-to-sequence

[1] The paragraphs are shuffled for purposes of obfuscation,

Model	GLEU Dev	GLEU Test	M^2 Dev	M^2 Test
No Change	89.68	89.45	00.00	00.00
SMT–diffs+M^2	90.44	–	38.55	–
SMT–diffs+BLEU	90.90	–	37.66	–
Word+Bi–diffs	91.18	–	38.88	–
Char+Bi–diffs	91.28	–	40.11	–
SMT+BLEU	90.95	90.70	38.99	38.31
Word+Bi	91.34	91.05	43.61	42.78
CharCNN	91.23	90.96	42.02	41.21
Char+Bi	**91.46**	**91.22**	**44.67**	**44.62**
Word+dom	91.25	–	43.12	–
Word+Bi+dom	91.45	–	44.33	–
CharCNN+Bi+dom	91.15	–	40.79	–
CharCNN+dom	91.35	–	43.94	–
Char+Bi+dom	**91.64**	**91.39**	**47.25**	**46.72**

Table 1: AESW development/test set correction results. GLEU and M^2 differences on test are statistically significant via paired bootstrap resampling (Koehn, 2004; Graham et al., 2014) at the 0.05 level, resampling the full set 50 times.

models we propose modeling the input and output sequences with a special initial token representing the journal domain (+dom).[2]

3 Experiments

Data AESW (Daudaravicius, 2016; Daudaravicius et al., 2016) consists of sentences taken from academic articles annotated with corrections by professional editors used for the AESW shared task. The training set contains 1,182,491 sentences, of which 460,901 sentences have edits. We set aside a 9,947 sentence sample from the original development set for tuning (of which 3,797 contain edits), and use the remaining 137,446 sentences as the dev set[3] (of which 53,502 contain edits). The test set contains 146,478 sentences.

The primary focus of the present study is conducting controlled experiments on the AESW dataset, but we also investigate results on the CoNLL-2014 shared task data in light of recent neural results (Ji et al., 2017) and to serve as a baseline of comparison against existing sequence-to-sequence approaches (Yuan and Briscoe, 2016; Xie et al., 2016). We use the common sets of public data appearing in past work for training: the National University of Singapore (NUS) Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013) and the publicly available Lang-8 data (Tajiri et al., 2012; Mizumoto et al., 2012). The Lang-8 dataset of corrections is large[4] but is crowd-sourced[5] and is thus of a different nature than the professionally annotated AESW and NUCLE datasets. We use the revised CoNLL-2013 test set as a tuning/dev set and the CoNLL-2014 test set (without alternatives) for testing. We do not make use of the non-public Cambridge Learner Corpus (CLC) (Nicholls, 2003), which contains over 1.5 million sentence pairs.

Evaluation We follow past work and use the Generalized Language Understanding Evaluation (GLEU) (Napoles et al., 2016) and MaxMatch (M^2) metrics (Dahlmeier and Ng, 2012).

Parameters All our models, implemented with OpenNMT (Klein et al.), are 2-layer LSTMs with 750 hidden units. For the Word model, the word embedding size is also set to 750, while for the CharCNN and Char models we use a character embedding size of 25. The CharCNN model has a convolutional layer with 1000 filters of width 6 followed by max-pooling, which is fed into a 2-layer highway network. Additional training details are provided in Appendix A. For AESW, the Word+Bi model contains around 144 million parameters, the CharCNN+Bi model around 79 million parameters, and the Char+Bi model around 25 million parameters.

Statistical Machine Translation As a baseline of comparison, we experiment with a phrase-based machine translation approach (SMT) shown to be state-of-the-art for the CoNLL-2014 shared task data in previous work (Junczys-Dowmunt and Grundkiewicz, 2016), which adds task specific features and the M^2 metric as a scorer to the Moses statistical machine translation system. The SMT model follows the training, parameters, and dense and sparse task-specific features that generate state-of-the-art results for CoNLL-2014 shared task data, as implemented in publicly available code.[6] However, to compare models against the same training data, we remove language model features associated with external data.[7] We exper-

so document-level context is not available.

[2] Characteristics of the dataset preclude experiments with additional paragraph context features. (See Appendix A.)

[3] The dev set contains 13,562 unique deletion types, 29,952 insertion types, and 39,930 replacement types.

[4] about 1.4 million sentences after filtering

[5] derived from the Lang-8 language-learning website

[6] SRI International provided access to SRILM (Stolcke, 2002) for running Junczys-Dowmunt and Grundkiewicz (2016)

[7] We found that including the features and data associated with the large language models of Junczys-Dowmunt and Grundkiewicz (2016), created from Common Crawl text

iment with tuning against M^2 (+M^2) and BLEU (+BLEU). Models trained with diffs were only tuned with BLEU, since the tuning pipeline from previous work is not designed to handle removing such annotation tags prior to M^2 scoring.

4 Results and Analysis: AESW

Table 1 shows the full set of experimental results on the AESW development and test data.

The CHAR+BI+DOM model is stronger than the WORD+BI+DOM and CHARCNN+DOM models by 2.9 M^2 (0.2 GLEU) and 3.3 M^2 (0.3 GLEU), respectively. The sequence-to-sequence models were also more effective than the SMT models, as shown in Table 1. We find that training with target diffs is beneficial across all models, with an increase of about 5 M^2 points for the WORD+BI model, for example. Adding +DOM information slightly improves effectiveness across models.

We analyzed deletion, insertion, and replacement error types. Table 2 compares effectiveness across replacement errors. We found the CHARCNN+BI models were less effective than CHARCNN variants in terms of GLEU and M^2, and the strongest CHARCNN models were eclipsed by the WORD+BI models in terms of the GLEU and M^2 scores. However, Table 2 shows CHARCNN+DOM is stronger on lower frequency replacements than WORD models. The CHAR+BI+DOM model is relatively strong on article and punctuation replacements, as well as errors appearing with low frequency in the training set and overall across deletion and insertion error types, which are summarized in Table 3.

Errors never occurring in training The comparatively high Micro $F_{0.5}$ score (18.66) for the CHAR+BI+DOM model on replacement errors (Table 2) never occurring in training is a result of a high precision (92.65) coupled with a low recall (4.45). This suggests some limited capacity to generalize to items not seen in training. A selectively chosen example is the replacement from "discontinous" to "discontinuous", which never occurs in training. However, similar errors of low edit distance also occur once in the dev set and never in training, but the CHAR+BI+DOM model never correctly recovers many of these errors, and many of the correctly recovered errors are minor changes in capitalization or hyphenation.

Error frequency About 39% of the AESW training sentences have errors, and of those sentences, on average, 2.4 words are involved in changes in deletions, insertions, or replacements (i.e., the count of words occurring between diff tags) per sentence. In the NUCLE data, about 37% of the sentences have errors, of which on average, 5.3 words are involved in changes. On the AESW dev set, if we only consider the 9545 sentences in which 4 or more words are involved in a change (average of 5.8 words in changes per sentence), the CHAR+BI model is still more effective than SMT+BLEU, with a GLEU score of 67.21 vs. 65.34. The baseline GLEU score (No Change) is 60.86, reflecting the greater number of changes relative to the full dataset (cf. Table 1).

Re-annotation The AESW dataset only provides 1 annotation for each sentence, so we perform a small re-annotation of the data to gauge effectiveness in the presence of multiple annotations. We collected 3 outputs (source, gold, and generated sentences from the CHAR+BI+DOM model) for 200 randomly sampled sentences, reannotating to create 3 new references for each sentence. The GLEU scores for the 200 original source, CHAR+BI+DOM, and original gold sentences evaluated against the 3 new references were 79.79, 81.72, and 84.78, respectively, suggesting that there is still progress to be made on the task relative to human levels of annotation.

5 Results and Analysis: CoNLL

Table 4 shows the results on the CoNLL dev set, and Table 5 contains the final test results.

Since the CoNLL data does not contain enough data for training neural models, previous works add the crowd-sourced Lang-8 data; however, this data is not professionally annotated. Since the distribution of corrections differs between the dev/test and training sets, we need to tune the precision and recall.

As shown in Table 4, WORD+BI effectiveness increases significantly by tuning the weights[8] assigned to the diff tags on the CoNLL-2013 set[9].

filtered against the NUCLE corpus, *hurt* effectiveness for the phrase-based models. This is likely a reflection of the domain specific nature of the academic text and LaTeX holder symbols appearing in the text. Here, we conduct controlled experiments without introducing additional domain-specific monolingual data.

[8] In contrast, in early experiments on AESW, tuning yielded negligible improvements.

[9] The single model with highest M^2 score was then run on the test set. Here, a single set is used for tuning and dev.

Replacement Error Type (out of 39,930) – Frequency relative to training							
Model	Punctuation	Articles	Other				
			> 100	$[5, 100]$	$[2, 5)$	1	0
Raw frequency in dev	11507	1691	6788	8974	2271	1620	7079
Number of unique instances	371	367	215	2918	1510	1242	5819
SMT+BLEU	56.03	16.41	44.57	36.17	39.46	31.93	0.00
Word+Bi	56.13	18.58	55.38	44.33	18.79	6.38	0.77
Word+Bi+dom	56.87	19.16	59.02	44.57	19.70	4.42	2.01
CharCNN+dom	55.64	13.37	57.34	41.83	28.99	16.74	7.09
Char+Bi	58.71	28.40	55.34	44.59	28.98	24.48	14.14
Char+Bi+dom	58.93	27.64	59.32	46.08	32.82	26.48	18.66

Table 2: Micro $F_{0.5}$ scores on replacement errors on the dev set. Errors are grouped by 'Punctuation', 'Article', and 'Other'. 'Other' errors are further broken down based on frequency buckets on the training set, with errors grouped by the frequency in which they occur in the training set.

	Deletions	Insertions	Replacements
SMT+BLEU	46.56	31.48	42.21
Word+Bi	47.75	38.31	46.02
Word+Bi+dom	47.78	39.00	47.29
CharCNN+dom	48.30	39.57	46.24
Char+Bi	49.05	37.17	48.55
Char+Bi+dom	50.20	42.51	50.39

Table 3: Micro $F_{0.5}$ scores across error types

	Precision	Recall	$F_{0.5}$
Word+Bi–diffs	65.36	6.19	22.45
Word+Bi, before tuning	72.34	0.97	4.60
Word+Bi, after tuning	46.66	15.35	33.14

Table 4: M^2 scores on the CoNLL-2013 set.

Note that we are tuning the weights on this same CoNLL-2013 set. Without tuning, the model very rarely generates a change, albeit with a high precision. After tuning, it exceeds the effectiveness of Word+Bi–diffs. The comparatively low effectiveness of Word+Bi–diffs is consistent with past sequence-to-sequence approaches utilizing data augmentation, additional annotated data, and/or secondary models to achieve competitive levels of effectiveness.

Table 5 shows that Word+Bi is within 0.2 M^2 of Ji et al. (2017), despite using over 1 million fewer sentence pairs, and exceeds the M^2 scores of Xie et al. (2016) and Yuan and Briscoe (2016) without the secondary models of those systems. We hypothesize that further gains are possible utilizing the CLC data and moving to the character model. (The character model is omitted here due to the long training time of about 4 weeks.)

	Data	M^2
Yuan and Briscoe (2016)	CLC*	39.90
Xie et al. (2016)	NUCLE, Lang-8, Common Crawl LM	40.56
Ji et al. (2017)	NUCLE, Lang-8, CLC*	41.53
Word+Bi–diffs	NUCLE, Lang-8	35.73
Word+Bi	NUCLE, Lang-8	41.37

Table 5: M^2 scores on the CoNLL-2014 test set and data used for recent sequence-to-sequence based systems. Results for previous works are those reported by the original authors. *CLC is proprietary.

Notably, SMT systems (with LMs) are still more effective than reported sequence-to-sequence results, as in Ji et al. (2017), on CoNLL.[10]

6 Conclusion

Our experiments demonstrate that on a large, professionally annotated dataset, a sequence-to-sequence character-based model of diffs can lead to considerable effectiveness gains over a state-of-the-art SMT system with task-specific features, ceteris paribus. Furthermore, in the crowd-sourced environment of the CoNLL data, in which there are comparatively few professionally annotated sentences in training, modeling diffs enables a means of tuning that improves the effectiveness of sequence-to-sequence models for the task.

[10] For reference, the reported M^2 results of the carefully optimized SMT system of Junczys-Dowmunt and Grundkiewicz (2016) trained on NUCLE and Lang-8, with parameter vectors averaged over multiple runs, with a Wikipedia LM is 45.95 and adding a Common Crawl LM is 49.49. We leave to future work the intersection of a LM for the CoNLL environment and more generally, whether these patterns hold in the presence of additional monolingual data.

References

Shamil Chollampatt, Duc Tam Hoang, and Hwee Tou Ng. 2016a. Adapting grammatical error correction based on the native language of writers with neural network joint models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1901–1911, Austin, Texas. Association for Computational Linguistics.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016b. Neural network translation models for grammatical error correction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2768–2774. AAAI Press.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 568–572, Stroudsburg, PA, USA. Association for Computational Linguistics.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The nus corpus of learner english. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia. Association for Computational Linguistics.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. Hoo 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 54–62, Stroudsburg, PA, USA. Association for Computational Linguistics.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The hoo 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, ENLG '11, pages 242–249, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vidas Daudaravicius. 2016. Automated evaluation of scientific writing data set (version 1.2) [data file]. VTeX.

Vidas Daudaravicius, Rafael E. Banchs, Elena Volodina, and Courtney Napoles. 2016. A report on the automatic evaluation of scientific writing shared task. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 53–62, San Diego, CA. Association for Computational Linguistics.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2014. Randomized significance tests in machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 266–274, Baltimore, Maryland, USA. Association for Computational Linguistics.

Duc Tam Hoang, Shamil Chollampatt, and Hwee Tou Ng. 2016. Exploiting n-best hypotheses to improve an smt approach to grammatical error correction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2803–2809. AAAI Press.

J. Ji, Q. Wang, K. Toutanova, Y. Gong, S. Truong, and J. Gao. 2017. A Nested Attention Neural Hybrid Model for Grammatical Error Correction. *ArXiv e-prints*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556, Austin, Texas. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-Aware Neural Language Models. In *Proceedings of AAAI*.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2012. The effect of learner corpus size in grammatical error correction of ESL writings. In *Proceedings of COLING 2012: Posters*, pages 863–872, Mumbai, India. The COLING 2012 Organizing Committee.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2016. GLEU without tuning. *eprint arXiv:1605.02592 [cs.CL]*.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

Diane Nicholls. 2003. The cambridge learner corpus: Error coding and analysis for lexicography and ELT. In *Proceedings of the Corpus Linguistics 2003 conference*, pages 572–581.

Alla Rozovskaya and Dan Roth. 2016. Grammatical error correction: Machine translation and classifiers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2205–2215, Berlin, Germany. Association for Computational Linguistics.

Allen Schmaltz, Yoon Kim, Alexander M. Rush, and Stuart Shieber. 2016. Sentence-level grammatical error identification as sequence-to-sequence correction. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 242–251, San Diego, CA. Association for Computational Linguistics.

Andreas Stolcke. 2002. Srilm – an extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing*, volume 2, pages 901–904, Denver.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for esl learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 198–202, Jeju Island, Korea. Association for Computational Linguistics.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y. Ng. 2016. Neural language correction with character-based attention. *CoRR*, abs/1603.09727.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386, San Diego, California. Association for Computational Linguistics.

A Study of Style in Machine Translation: Controlling the Formality of Machine Translation Output

Xing Niu
Dept. of Computer Science
University of Maryland
College Park
xingniu@cs.umd.edu

Marianna Martindale
iSchool
University of Maryland
College Park
mmartind@umd.edu

Marine Carpuat
Dept. of Computer Science
University of Maryland
College Park
marine@cs.umd.edu

Abstract

Stylistic variations of language, such as formality, carry speakers' intention beyond literal meaning and should be conveyed adequately in translation. We propose to use lexical formality models to control the formality level of machine translation output. We demonstrate the effectiveness of our approach in empirical evaluations, as measured by automatic metrics and human assessments.

1 Introduction

Automatically analyzing and generating natural language requires capturing not only what is said, but also how to say it. Consider the sentences "anybody hurt?" and "is someone wounded?". The first one is less formal than the second one, and carries information beyond its literal meaning, such as the situation in which it might be used. Such differences in formality have been identified as an important dimension of style (Trudgill, 1992) or tone (Halliday, 1978) variation.

In this paper, we build on prior computational work that has focused on analyzing formality of texts (Lahiri and Lu, 2011; Brooke and Hirst, 2013; Pavlick and Nenkova, 2015; Pavlick and Tetreault, 2016) with a different aim: modeling formality for the purpose of controlling style in applications that generate language, with a focus on machine translation. Human translators translate a document for a specific audience (Nida and Taber Charles, 1969), and often ask what is the expected tone of the content when taking a new translation job. We design a machine translation system that operates under similar conditions and explicitly takes an expected level of formality as input. While ultimately we would like systems to preserve the formality of the source, this is a challenging task that requires not only automatically inferring the formality of the source, but also understanding how formality differs across languages and cultures. As a first step, we therefore limit our study to the scenario where the expected output formality is given to the MT system as an additional input.

We first select a formality model providing the most accurate scores on intrinsic formality datasets. We compare existing lexical formality models and novel variants based on inducing formality dimensions or subspaces in vector space models. We then turn to machine translation and show that a lexical formality model can have a positive impact when used to control the formality of machine translation output. When the expected formality matches the reference, we obtain improvement of translation quality evaluated by automatic metrics (BLEU). A human assessment also verified the effectiveness of our proposed system in generating translations at diverse levels of formality.

2 Formality-Sensitive MT

Our goal is to provide systems with the ability to generate language across a range of formality style. We propose a **Formality-Sensitive Machine Translation (FSMT)** scenario where the system takes two inputs: (1) text in the source language to be translated, and (2) a desired formality level capturing the intended audience of the translation. We propose to implement it as n-best re-ranking within a standard phrase-based MT architecture. Unlike domain adaptation approaches, which aim to produce domain-specific or potentially formality-specific systems, our goal is to obtain a single system trained on diverse data which can adaptively produce output for a range of styles.

We therefore introduce a formality-scoring fea-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2814–2819
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

ture for re-ranking. For each translation hypothesis h, given the formality level ℓ as a parameter:

$$f(h;\ell) = |\,\mathrm{Formality}(h) - \ell|$$

where $\mathrm{Formality}(h)$ is the sentence-level formality score for h. $f(h;\ell)$, along with standard model features, is fed into a standard re-ranking model. When training the re-ranking model, the parameter ℓ is set to the actual formality score of the reference translation for each instance. At test time, ℓ is provided by the user. The re-scoring weights help promote candidate sentences whose formality scores approach the expected level.

3 Formality Modeling

The FSMT system requires quantifying the formality level of a sentence. Following prior work, we define sentence-level formality based on lexical formality scores (Brooke et al., 2010; Pavlick and Nenkova, 2015). We conduct an empirical comparison of existing techniques that can be adapted as lexical formality models, and introduce a sentence-level formality scheme based on weighted average.

3.1 Lexical Formality

State-of-the-art lexical formality models (Brooke et al., 2010; Brooke and Hirst, 2014) are based on vector space models of word meaning, and a set of pre-selected seed words that are representative of formal and informal language.

SimDiff Brooke et al. (2010) proposed to score the formality of a word w by comparing its meaning to that of seed words of known formality using cosine similarity. Intuitively, w is more likely formal if it is semantically closer to formal seed words than to informal seed words. Formally, given a formal word set S_f and an informal word set S_i, SimDiff scores a word w by

$$\mathrm{score}(w) = \frac{1}{|S_f|}\sum_{v\in S_f}\cos(e_w,e_v) - \frac{1}{|S_i|}\sum_{v\in S_i}\cos(e_w,e_v)$$

Turning this difference into a formality score requires further manipulation. A neutral word r has to be manually selected to anchor the midpoint of the formality score range. In other words, the final formality score for r is enforced to be zero:

$$\mathrm{Formality}(w) = \frac{\mathrm{score}(w) - \mathrm{score}(r)}{\mathrm{normalizer}(w,r)}$$

The neutral word is typically selected from function words. We select “at” because it appears in nearly every document and appears with nearly equivalent probabilities in formal/informal corpora. Finally, a normalizer which is maximized among the whole vocabulary ensures that scores cover the entire $[-1,1]$ range.

Instead of using cosine diff as the score function $\mathrm{score}(w)$, other standard techniques can be also applied under this framework.

SVM As an alternative to the model proposed by Brooke and Hirst (2014), we propose to train an Support Vector Machine (SVM) model to find a hyperplane that separates formal and informal words and define the score function as the distance to the hyperplane.

Formality Subspace Another category of methods consists in identifying a subspace that captures formality within the original vector space. Lexical scores can then simply be obtained by projecting word representations onto the formality subspace. One example is training a Principal Component Analysis (**PCA**) model on word representations of all seeds. This method is based on the assumption that representative formal/informal words principally vary along the direction of formality. Alternatively, inspired by **Densifier** (Rothe et al., 2016), we can learn a subspace that aims at separating words in S_f vs. words in S_i and grouping words in the same set.

3.2 From Word to Sentence Formality

While previous work scored sentence by averaging word scores (Brooke and Hirst, 2014; Pavlick and Nenkova, 2015), we propose a weighted average scheme for word sequences W to downgrade the formality contribution of neutral words:

$$\mathrm{Formality}(W) = \frac{\sum_{w_i\in W}|\,\mathrm{Formality}(w_i)| \cdot \mathrm{Formality}(w_i)}{\sum_{w_i\in W}|\,\mathrm{Formality}(w_i)|}$$

3.3 Evaluation

Before evaluating our FSMT framework, we evaluate the formality models at the sentence level. Lahiri (2015) and Pavlick and Tetreault (2016) collected 5-way human scores for 11,263 sentences in the genres of blog, email, answers and news. Following Pavlick and Tetreault (2016), we averaged human scores for each sentence as the gold standard. As in prior work, the score quality was evaluated by the Spearman correlation.

A large mixed-topic corpus is required to train vector space models. As suggested by Brooke et al. (2010), we used the ICWSM 2009 Spinn3r dataset (English tier-1) which consists of about 1.6 billion words (Burton et al., 2009). We also compared the term-document association model Latent Semantic Analysis (LSA) (Deerwester et al., 1990) and the term-term association model word2vec (W2V) (Mikolov et al., 2013). We used the same 105 formal seeds and 138 informal seeds as Brooke et al. (2010).

Followed Brooke et al. (2010), to achieve best performance, we used a small dimensionality (10) for training LSA and word2vec. In practice, we normalized the LSA word vectors to make them have unit length for SVM and PCA, but did not applied it to word2vec. This suggests that the magnitude of LSA word vectors is harmful for formality modeling.

We also compared formality models based on word representations to a baseline that relies on unigram models to compare word statistics in corpora representative of formal vs. informal language (Pavlick and Nenkova, 2015). This method requires language examples of diverse formality. Conversational transcripts are generally considered as casual text, so we concatenated corpora such as Fisher (Cieri et al., 2004), Switchboard (Godfrey et al., 1992), SBCSAE (Bois et al., 2000-2005), CallHome[1], CallFriend[2], BOLT SMS/Chat (Song et al., 2014) and NPS Chatroom (Forsythand and Martell, 2007). As the formal counterpart, we extracted comparable size of text from Europarl (Koehn, 2005). This results in 30 Million tokens of formal corpora (1.1M segments) and 29 Million tokens of informal corpora (2.7M segments).

Table 1 shows that all models based on the vector space achieve similar performance in terms of Spearman's ρ (except SVM-W2V which yields lower performance). The baseline method based on unigram models was outperformed by 0.1+ point. So we select DENSIFIER-LSA as a representative for our FSMT system.

	LSA	W2V
SimDiff	0.660	0.654
SVM	0.657	0.585
PCA	0.656	0.663
DENSIFIER	0.664	0.644
baseline	0.540	

Table 1: Sentence-level formality quantifying evaluation (Spearman's ρ) among different models with different vector spaces.

4 Evaluation of the FSMT System

Set-up We evaluate this approach on a French to English translation task. Two parallel French-English corpora are used: (1) MultiUN (Eisele and Chen, 2010), which is extracted from the United Nations website, and can be considered to be formal text; (2) OpenSubtitles2016 (Lison and Tiedemann, 2016), which is extracted from movie and TV subtitles, covers a wider spectrum of styles, but overall tends to be informal since it primarily contains conversations. Each parallel corpus was split into a training set (100M English tokens), a tuning set (2.5K segments) and a test set (5K segments). Two corpora are then concatenated, such that training, tuning and test sets all contained a diversity of styles.

Moses (Koehn et al., 2007) is used to build our phrase-based MT system. We followed the standard training pipeline with default parameters.[3] Word alignments were generated using fast_align (Dyer et al., 2013), and symmetrized using the *grow-diag-final-and* heuristic. We used 4-gram language models, trained using KenLM (Heafield, 2011). Model weights were tuned using batch MIRA (Cherry and Foster, 2012).

We used constant size n=1000 for n-best lists in all experiments. The re-ranking is a log-linear model trained using batch MIRA. [4] We report results averaged over 5 random tuning re-starts to compensate for tuning noise (Clark et al., 2011).

FSMT In order to evaluate the impact of different input formality (e.g. low/neutral/high) on translation quality, ideally, we would like to have three human reference translations with different

[1] https://catalog.ldc.upenn.edu/LDC97S42

[2] https://talkbank.org/access/CABank/CallFriend/

[3] http://www.statmt.org/moses/?n=Moses.Baseline

[4] https://github.com/moses-smt/mosesdecoder/tree/master/scripts/nbest-rescore

Desired formality	Informal test set	Neutral test set	Formal test set
None (baseline)	39.74	40.17	**47.97**
low	**40.27**	39.65	47.76
neutral	38.70	**40.46**	47.84
high	37.58	39.53	**47.97**

Table 2: Translation quality (BLEU scores) on informal/neutral/formal sentence sets given different desired formality levels (-0.4, 0.0, 0.4). Best results with statistical significance are highlighted.

formality for each source sentence. Since such references are not available, we construct three sets of test data where instances are divided according to the formality level of the available reference translation. The formality distribution in the tuning set shows that 97% reference translations fall into the range of $[-0.6, 0.6]$. We therefore set three formality bins – informal $[-1, -0.2)$, neutral formality $[-0.2, 0.2]$, and formal $(0.2, 1]$ – and split the test set into these bins. We use DENSIFIER-LSA and training setting described above to translate the entire test set three times, with three different formality levels: low (-0.4), neutral (0) and high (0.4).

4.1 Automatic Evaluation

We first report standard automatic evaluation results using the BLEU score to compare FSMT output given different desired formality level on each bins (See Table 2).

The best BLEU scores for each formality level are obtained when the level of formality given as input to the MT system matches the nature of the text being translated, as can be seen in the scores along the diagonal in Table 2. Comparing with the baseline system, which produces the top translation from each n-best list, translation quality improves by +0.5 BLEU on informal text, +0.3 BLEU on neutral text, and remains constant on formal text. The impact increases with the distance to formal language increases. This can be explained by the fact that more formal sentences tend to be longer, and the impact of alternate lexical choice for a small number of words per sentence is smaller in longer sentences. In addition, the formal sentences are mostly drawn from UN data which is sufficiently different from the other genres in the heterogeneous training corpus that the informal examples do not affect baseline performance on formal data.

4.2 Human Assessment

Automatic evaluation is limited to comparing output to a single reference: lower BLEU scores conflate translation errors and stylistic mismatch. Therefore, we conduct a human study of the formality vs. the quality.

We conducted a manual evaluation of the output of our FSMT system taking low/high formality levels (-0.4/0.4) as parameters. 42 translation pairs were randomly selected and were annotated by 15 volunteers. For each pair of segments, the volunteers were asked to select the segment that would be more appropriate in a formal setting (e.g., a job interview) than in a casual setting (e.g., chatting with friends). A default option of "neither of them is more formal or hard to say" was also available.

By majority voting, 20 pairs were annotated as "N", indicating the two translations has no distinctions w.r.t. formality. For example, "A: how can they do this" vs. "B: how can they do that". Given that the translations were restricted to the n-best list, not all sentences could be translated into stylistically different language.

Of the remaining 21 pairs where annotators judged one output more formal than the other, in all but one case the translation produced by our FSMT system with high formality level parameter was judged to be more formal. Overall this indicates that our formality scoring and ranking procedure are effective.

To determine whether re-ranking based on formality might have a detrimental effect on quality, we also had annotators rate the fluency and adequacy of the segments. Inspired by Graham et al. (2013), annotators were first asked to assess fluency without a reference and separately adequacy with a reference. Both assessments used a sliding scale. Each segment was evaluated by an average of 7 annotators. After rescaling the ratings into the $[0, 1]$ range, we observed a 0.75 level of fluency for informal translations and 0.70 for formal ones. This slight difference fits our expectation that more casual language may feel more fluent while more formal language may feel more stilted. The adequacy ratings were 0.65 and 0.64 for informal and translations respectively, indicating that adjusting the level of formality had minimal effect on the adequacy of the result.

Some examples are listed in Table 3. Occa-

ℓ	**Examples**	**Comments**
-0.4	... and then he **ran away** .	–
0.4	... and then he **escaped** .	*annotated as more formal*
-0.4	**anybody hurt** ?	–
0.4	is **someone wounded** ?	*annotated as more formal*
-0.4	he **shot himself** in the middle of it .	–
0.4	he **committed suicide** in the middle of it .	*annotated as more formal*
-0.4	to move **things** forward .	–
0.4	**in order** to move **the process** forward.	*annotated as more formal*
-0.4	how do you do ?	*annotated as more formal*
0.4	how are you?	–
-0.4	oh , val , you should get the phone .	*missing words*
0.4	oh , val , you should have the phone (of pete) .	–
-0.4	**i believe** you've solved the case , lieutenant .	*additive words*
0.4	you solved the case , lieutenant .	–
REF	right by checkout .	
-0.4	right next to the **body** .	*incorrect word choice*
0.4	right next to the **fund** .	*incorrect word choice*

Table 3: Examples of variant translations to the same French source segment using low/high output formality levels (-0.4/0.4) as parameters. In general the variations lie on the direction of formality as expected, but occasionally translation errors occur.

sionally, the n-best list had no translation hypotheses with diverse formality, so the FSMT system dropped necessary words, appended inessential words, or selected improper or even incorrect words to fit the target formality level. In the case of 'how do you do', the translation that was meant to be more casual was rated more formal. Because the system measures formality on the lexical level, it was not able to recognize this idiomatically formal phrase made up of words that are not inherently formal. Despite these issues, most of the output were formality-variant translations of the same French source segment, as expected.

5 Conclusion

We presented a framework for formality-sensitive machine translation, where a system produces translations at a desired formality level. Our evaluation shows the effectiveness of this system in controlling language formality without loss in translation quality.

References

Du Bois, John W., Wallace L. Chafe, Charles Meyer, Sandra A. Thompson, Robert Englebretson, and Nii Martey. 2000-2005. Santa barbara corpus of spoken american english, parts 1-4.

Julian Brooke and Graeme Hirst. 2013. Hybrid models for lexical acquisition of correlated styles. In *IJCNLP*, pages 82–90. Asian Federation of Natural Language Processing / ACL.

Julian Brooke and Graeme Hirst. 2014. Supervised ranking of co-occurrence profiles for acquisition of continuous lexical attributes. In *COLING*, pages 2172–2183. ACL.

Julian Brooke, Tong Wang, and Graeme Hirst. 2010. Automatic acquisition of lexical formality. In *COLING (Posters)*, pages 90–98. Chinese Information Processing Society of China.

Kevin Burton, Akshay Java, and Ian Soboroff. 2009. The ICWSM 2009 Spinn3r dataset. In *Proceedings of the Third Annual Conference on Weblogs and Social Media (ICWSM 2009), San Jose, CA*.

Colin Cherry and George F. Foster. 2012. Batch tuning strategies for statistical machine translation. In *HLT-NAACL*, pages 427–436. The Association for Computational Linguistics.

Christopher Cieri, David Miller, and Kevin Walker. 2004. The fisher corpus: a resource for the next generations of speech-to-text. In *LREC*. European Language Resources Association.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *ACL (Short Papers)*, pages 176–181. The Association for Computer Linguistics.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *HLT-NAACL*, pages 644–648. The Association for Computational Linguistics.

Andreas Eisele and Yu Chen. 2010. Multiun: A multilingual corpus from united nation documents. In *LREC*. European Language Resources Association.

Eric N. Forsythand and Craig H. Martell. 2007. Lexical and discourse analysis of online chat dialog. In *ICSC*, pages 19–26. IEEE Computer Society.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of IEEE International Conference on Speech, and Signal Processing*, volume 1, pages 517–520. IEEE.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2013. Continuous measurement scales in human evaluation of machine translation. In *LAW@ACL*, pages 33–41. The Association for Computer Linguistics.

Michael AK Halliday. 1978. *Language as Social Semiotic: The Social Interpretation of Language and Meaning*. Edward Arnold, London.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*. The Association for Computational Linguistics.

Shibamouli Lahiri. 2015. SQUINKY! A corpus of sentence-level formality, informativeness, and implicature. *CoRR*, abs/1506.02306.

Shibamouli Lahiri and Xiaofei Lu. 2011. Interrater agreement on sentence formality. *CoRR*, abs/1109.0069.

Pierre Lison and Jörg Tiedemann. 2016. Opensubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *LREC*. European Language Resources Association (ELRA).

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

EugeneA. Nida and R Taber Charles. 1969. The theory and practice of translation. In *Helps for Translators*, volume 8. United Bible Societies.

Ellie Pavlick and Ani Nenkova. 2015. Inducing lexical style properties for paraphrase and genre differentiation. In *HLT-NAACL*, pages 218–224. The Association for Computational Linguistics.

Ellie Pavlick and Joel R. Tetreault. 2016. An empirical analysis of formality in online communication. *TACL*, 4:61–74.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *HLT-NAACL*, pages 767–777. The Association for Computational Linguistics.

Zhiyi Song, Stephanie Strassel, Haejoong Lee, Kevin Walker, Jonathan Wright, Jennifer Garland, Dana Fore, Brian Gainor, Preston Cabe, Thomas Thomas, Brendan Callahan, and Ann Sawyer. 2014. Collecting natural SMS and chat conversations in multiple languages: The BOLT phase 2 corpus. In *LREC*, pages 1699–1704. European Language Resources Association (ELRA).

Peter Trudgill. 1992. *Introducing Language and Society*. Penguin, London.

Sharp Models on Dull Hardware: Fast and Accurate Neural Machine Translation Decoding on the CPU

Jacob Devlin
jdevlin@microsoft.com
Microsoft Research

Abstract

Attentional sequence-to-sequence models have become the new standard for machine translation, but one challenge of such models is a significant increase in training and decoding cost compared to phrase-based systems. Here, we focus on efficient decoding, with a goal of achieving accuracy close the state-of-the-art in neural machine translation (NMT), while achieving CPU decoding speed/throughput close to that of a phrasal decoder.

We approach this problem from two angles: First, we describe several techniques for speeding up an NMT beam search decoder, which obtain a 4.4x speedup over a very efficient baseline decoder without changing the decoder output. Second, we propose a simple but powerful network architecture which uses an RNN (GRU/LSTM) layer at bottom, followed by a series of stacked fully-connected layers applied at every timestep. This architecture achieves similar accuracy to a deep recurrent model, at a small fraction of the training and decoding cost. By combining these techniques, our best system achieves a very competitive accuracy of 38.3 BLEU on WMT English-French NewsTest2014, while decoding at 100 words/sec on single-threaded CPU. We believe this is the best published accuracy/speed trade-off of an NMT system.

1 Introduction

Attentional sequence-to-sequence models have become the new standard for machine translation over the last two years, and with the unprecedented improvements in translation accuracy comes a new set of technical challenges. One of the biggest challenges is the high training and decoding costs of these neural machine translation (NMT) system, which is often at least an order of magnitude higher than a phrase-based system trained on the same data. For instance, phrasal MT systems were able achieve single-threaded decoding speeds of 100-500 words/sec on decade-old CPUs (Quirk and Moore, 2007), while Jean et al. (2015) reported single-threaded decoding speeds of 8-10 words/sec on a shallow NMT system. Wu et al. (2016) was able to reach CPU decoding speeds of 100 words/sec for a deep model, but used 44 CPU cores to do so. There has been recent work in speeding up decoding by reducing the search space (Kim and Rush, 2016), but little in computational improvements.

In this work, we consider a production scenario which requires low-latency, high-throughput NMT decoding. We focus on CPU-based decoders, since GPU/FPGA/ASIC-based decoders require specialized hardware deployment and logistical constraints such as batch processing. Efficient CPU decoders can also be used for on-device mobile translation. We focus on single-threaded decoding and single-sentence processing, since multiple threads can be used to reduce latency but not total throughput.

We approach this problem from two angles: In Section 4, we describe a number of techniques for improving the speed of the decoder, and obtain a 4.4x speedup over a highly efficient baseline. These speedups do not affect decoding results, so they can be applied universally. In Section 5, we describe a simple but powerful network architecture which uses a single RNN (GRU/LSTM) layer at the bottom with a large number of fully-connected (FC) layers on top, and obtains improvements similar to a deep RNN model at a fraction of the training and decoding cost.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2820–2825
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

2 Data Set

The data set we evaluate on in this work is WMT English-French NewsTest2014, which has 380M words of parallel training data and a 3003 sentence test set. The NewsTest2013 set is used for validation. In order to compare our architecture to past work, we train a word-based system without any data augmentation techniques. The network architecture is very similar to Bahdanau et al. (2014), and specific details of layer size/depth are provided in subsequent sections. We use an 80k source/target vocab and perform standard unk-replacement (Jean et al., 2015) on out-of-vocabulary words. Training is performed using an in-house toolkit.

3 Baseline Decoder

Our baseline decoder is a standard beam search decoder (Sutskever et al., 2014) with several straightforward performance optimizations:

- It is written in pure C++, with no heap allocation done during the core search.
- A candidate list is used to reduce the output softmax from 80k to ~500. We run word alignment (Brown et al., 1993) on the training and keep the top 20 context-free translations for each source word in the test sentence.
- The Intel MKL library is used for matrix multiplication, as it is the fastest floating point matrix multiplication library for CPUs.
- Early stopping is performed when the top partial hypothesis has a log-score of $\delta = 3.0$ worse than the best completed hypothesis.
- Batching of matrix multiplication is applied when possible. Since each sentence is decoded separately, we can only batch over the hypotheses in the beam as well as the input vectors on the source side.

4 Decoder Speed Improvements

This section describes a number of speedups that can be made to a CPU-based attentional sequence-to-sequence beam decoder. Crucially, none of these speedups affect the actual mathematical computation of the decoder, so they can be applied to any network architecture with a guarantee that they will not affect the results.[1]

The model used here is similar to the original implementation of Bahdanau et al. (2014). The exact target GRU equation is:

$$\begin{aligned}
d_{ij} &= \tanh(W_a h_{i-1} + V_a x_i) {\cdot} \tanh(U_a s_j) \\
\alpha_{ij} &= \frac{e^{d_{ij}}}{\sum_{j'} e^{d_{ij'}}} \\
c_i &= \sum_j \alpha_{ij} s_j \\
u_i &= \sigma(W_u h_{i-1} + V_u x_i + U_u c_i + b_u) \\
r_i &= \sigma(W_r h_{i-1} + V_r x_i + U_r c_i + b_r) \\
\hat{h}_i &= \sigma(r_i \odot (W_h h_{i-1}) + V_h x_i + U_h c_i + b_h) \\
h_i &= u_i h_{i-1} + (1 - u_i) \hat{h}_i
\end{aligned}$$

Where W_*, V_*, U_*, b_* are learned parameters, s_j is the hidden vector of the j^{th} source word, h_{i-1} is the previous target recurrent vector, x_i is the target input (e.g., embedding of previous word).

We also denote the various hyperparameters: b for the beam size, r for the recurrent hidden size, e is the embedding size, $|S|$ for the source sentence length, and $|T|$ for the target sentence length, $|E|$ is the vocab size.

4.1 16-Bit Matrix Multiplication

Although CPU-based matrix multiplication libraries are highly optimized, they typically only operate on 32/64-bit floats, even though DNNs can almost always operate on much lower precision without degredation of accuracy (Han et al., 2016). However, low-precision math (1-bit to 7-bit) is difficult to implement efficiently on the CPU, and even 8-bit math has limited support in terms of vectorized (SIMD) instruction sets. Here, we use 16-bit fixed-point integer math, since it has first-class SIMD support and requires minimal changes to training. Training is still performed with 32-bit floats, but we clip the weights to the range [-1.0, 1.0] the `relu` activation to [0.0, 10.0] to ensure that all values fit into 16-bits with high precision. A reference implementation of 16-bit multiplication in C++/SSE2 is provided in the supplementary material, with a thorough description of low-level details.[2]

A comparison between our 16-bit integer implementation and Intel MKL's 32-bit floating point multiplication is given in Figure 1. We can see that 16-bit multiplication is 2x-3x faster than 32-bit multiplication for batch sizes between 2 and 8, which is the typical range of the beam size b. We

[1] Some speedups apply quantization which leads to small random perturbations, but these change the BLEU score by less than 0.02.

[2] Included as ancillary file in Arxiv submission, on right side of submission page.

are able to achieve greater than a 2x speedup in certain cases because we pre-process the weight matrix offline to have optimal memory layout, which is a capability BLAS libraries do not have.

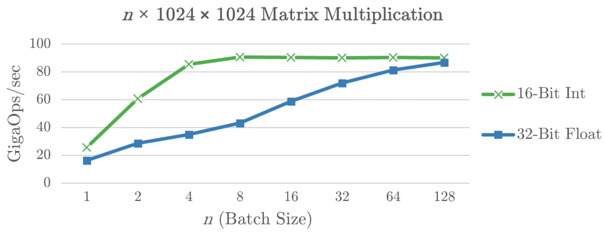


Figure 1: Single-threaded matrix multiplication using our 16-bit fixed-point vs. Intel MKL's 32-bit float, averaged over 10,000 multiplications. Both use the AVX2 instruction set.

4.2 Pre-Compute Embeddings

In the first hidden layer on the source and target sides, x_i corresponds to word embeddings. Since this is a closed set of values that are fixed after training, the vectors Vx_i can be pre-computed (Devlin et al., 2014) for each word in the vocabulary and stored in a lookup table. This can only be applied to the first hidden layer.

Pre-computation does increase the memory cost of the model, since we must store $r \times 3$ floats per word instead of e. However, if we only compute the k most frequently words (e.g., $k = 8,000$), this reduces the pre-computation memory by 90% but still results in 95%+ token coverage due to the Zipfian distribution of language.

4.3 Pre-Compute Attention

The attention context computation in the GRU can be re-factored as follows:

$$Uc_i = U(\sum_j \alpha_{ij} s_j) = \sum_j \alpha_{ij}(Us_j)$$

Crucially, the hidden vector representation s_j is only dependent on the source sentence, while a_{ij} is dependent on the target hypothesis. Therefore, the original computation Uc_i requires total $|T| \times b$ multiplications per sentence, but the re-factored version Us_j only requires total $|S|$ multiplications. The expectation over α must still be computed at each target timestep, but this is much less expensive than the multiplication by U.

4.4 SSE & Lookup Tables

For the element-wise vector functions use in the GRU, we can use vectorized instructions (SSE/AVX) for the `add` and `multiply` functions, and lookup tables for `sigmoid` and `tanh`. Reference implementations in C++ are provided in the supplementary material.

4.5 Merge Recurrent States

In the GRU equation, for the first target hidden layer, x_i represents the previously generated word, and h_{i-1} encodes the hypothesis up to *two* words before the current word. Therefore, if two partial hypotheses in the beam only differ by the last emitted word, their h_{i-1} vectors will be identical. Thus, we can perform matrix multiplication Wh_{i-1} only on the *unique* h_{i-1} vectors in the beam at each target timestep. For a beam size of $b = 6$, we measured that the ratio of unique h_{i-1} compared to total h_{i-1} is approximately 70%, averaged over several language pairs. This can only be applied to the first target hidden layer.

	Words/Sec.	**Speedup**
Type	**(Single-Threaded)**	**Factor**
Baseline	95	1.00x
+ 16-Bit Mult.	248	2.59x
+ Pre-Comp. Emb.	311	3.25x
+ Pre-Comp. Att.	342	3.57x
+ SSE & Lookup	386	4.06x
+ Merge Rec.	418	4.37x

Table 1: Decoding speeds on an Intel E5-2660 CPU, processing each sentence independently.

4.6 Speedup Results

Cumulative results from each of the preceding speedups are presented in Table 1, measured on WMT English-French NewsTest2014. The NMT architecture evaluated here uses 3-layer 512-dimensional bidirectional GRU for the source, and a 1-layer 1024-dimensional attentional GRU for the target. Each sentence is decoded independently with a beam of 6. Since these speedups are all mathematical identities excluding quantization noise, all outputs achieve 36.2 BLEU and are 99.9%+ identical.

The largest improvement is from 16-bit matrix multiplication, but all speedups contribute a significant amount. Overall, we are able to achieve a 4.4x speedup over a fast baseline decoder. Although the absolute speed is impressive, the model only uses one target layer and is several BLEU behind the SOTA, so the next goal is to maximize model accuracy while still achieving speeds greater than some target, such as 100 words/sec.

System	**BLEU**	**Words/Sec** **(Single-Threaded)**
Basic Phrase-Based MT (Schwenk, 2014)	33.1	-
SOTA Phrase-Based MT (Durrani et al., 2014)	37.0	-
RNN Search, 1-Layer Att. GRU, w/ Large Vocab (Jean et al., 2015)	34.6	†
Google NMT, 8-Layer Att. LSTM, Word-Based (Wu et al., 2016)	37.9	♭
Google NMT, 8-Layer Att. LSTM, WPM-32k (Wu et al., 2016)	39.0‡	♭
Convolutional Seq-to-Seq (Gehring et al., 2017)	40.5	-
Transformer Network (Vaswani et al., 2017)	41.0	-
(S1) Trg: 1024-AttGRU	36.2	418
(S2) Trg: 1024-AttGRU + 1024-GRU	36.8	242
(S3) Trg: 1024-AttGRU + 3-Layer 768-FC-Relu + 1024-FC-Tanh	37.1	271
(S4) Trg: 1024-AttGRU + 7-Layer 768-FC-Relu + 1024-FC-Tanh	37.4	229
(S5) Trg: 1024-AttGRU + 7-Layer 768-FC-Relu + 1024-GRU	37.6	157
(S6) Trg: 1024-AttGRU + 15-Layer 768-FC-Relu + 1024-FC-Tanh	37.3	163
(S7) Src: 8-Layer LSTM, Trg: 1024-AttLSTM + 7-Layer 1024-LSTM§	37.8	28
(E1) Ensemble of 2x Model (S4)	**38.3**	**102**
(E2) Ensemble of 3x Model (S4)	38.5	65

Table 2: Results on WMT English-French NewsTest2014. Models (S1)-(S6) use a 3-layer 512-dim bidirectional GRU for the source side. The CPU is an Intel Haswell E5-2660. † Reported as ~8 words/sec on one CPU core. ♭ Reported as ~100 words/sec, parallelized across 44 CPU cores. § Reproduction of Google NMT, Word-Based.

5 Model Improvements

In NMT, like in many other deep learning tasks, accuracy can be greatly improved by adding more hidden layers, but training and decoding time increase significantly (Luong et al., 2014; Zhou et al., 2016; Wu et al., 2016). Several past works have noted that convolutional neural networks (CNNs) are significantly less expensive than RNNs, and replaced the source and/or target side with a CNN-based architecture (Gehring et al., 2016; Kalchbrenner et al., 2016). However, these works have found it is difficult to replace the target side of the model with CNN layers while maintaining high accuracy. The use of a recurrent target is especially important to track attentional coverage and ensure fluency.

Here, we propose a mixed model which uses an RNN layer at the bottom to both capture full-sentence context and perform attention, followed by a series of fully-connected (FC) layers applied on top at each timestep. The FC layers can be interpreted as a CNN without overlapping stride. Since each FC layer consists of a single matrix multiplication, it is $1/6^{\text{th}}$ the cost of a GRU (or $1/8^{\text{th}}$ an LSTM). Additionally, several of the speedups from Section 4 can only be applied to the first layer, so there is strong incentive to only use a single target RNN.

To avoid vanishing gradients, we use ResNet-style skip connections (He et al., 2016). These allow very deep models to be trained from scratch and do not require any additional matrix multiplications, unlike highway networks (Srivastava et al., 2015). With 5 intermediate FC layers, target timestep i is computed as:

$$\begin{aligned}
h_i^B &= \text{AttGRU}(h_{i-1}^B, x_i, S) \\
h_i^1 &= \text{relu}(W^1 h_i^B) \\
h_i^2 &= \text{relu}(W^2 h_i^1) \\
h_i^3 &= \text{relu}(W^3 h_i^2 + h_i^1) \\
h_i^4 &= \text{relu}(W^4 h_i^3) \\
h_i^5 &= \text{relu}(W^5 h_i^4 + h_i^3) \\
h_i^T &= \tanh(W^T h_i^5) \textbf{ or } \text{GRU}(h_{i-1}^T, h_i^5) \\
y_i &= \text{softmax}(V h_i^T)
\end{aligned}$$

Where x_i is the target input embedding, S is the set of source hidden vectors used for attention, and V is the target output vocabulary matrix. The superscripts h^B and h^T simply denote the "bottom" and "top" hidden layers, while the numbered layers h^n represent the intermediate fully-connected layers.

We follow He et al. (2016) and only use skip connections on every other FC layer, but do not use batch normalization. The same pattern can be used for more FC layers, and the FC layers can be a different size than the bottom or top hidden layers. The top hidden layer can be an RNN or an FC layer. It is important to use `relu` activations (opposed to tanh) for ResNet-style skip connections. The GRUs still use tanh.

5.1 Model Results

Results using the mixed RNN+FC architecture are shown in Table 2, using all speedups. We have found that the benefit of using RNN+FC layers on the source is minimal, so we only perform ablation on the target. For the source, we use a 3-layer 512-dim bidi GRU in all models (S1)-(S6).

Model (S1) and (S2) are one and two layer baselines. Model (S4), which uses 7 intermediate FC layers, has similar decoding cost to (S2) while doubling the improvement over (S1) to 1.2 BLEU. We see minimal benefit from using a GRU on the top layer (S5) or using more FC layers (S6). In (E1) and (E2) we present 2 and 3 model ensembles of (S4), trained from scratch with different random seeds. We can see that the 2-model ensemble improves results by 0.9 BLEU, but the 3-model ensemble has little additional improvment. Although not presented here, we have found these improvement from decoder speedups and RNN+FC to be consistent across many language pairs.

All together, we were able to achieve a BLEU score of 38.3 while decoding at 100 words/sec on a single CPU core. As a point of comparison, Wu et al. (2016) achieves similar BLEU scores on this test set (37.9 to 38.9) and reports a CPU decoding speed of ~100 words/sec (0.2226 sents/sec), but parallelizes this decoding across 44 CPU cores. System (S7), which is our re-implementation of Wu et al. (2016), decodes at 28 words/sec on one CPU core, using all of the speedups described in Section 4. Zhou et al. (2016) has a similar computational cost to (S7), but we were not able to replicate those results in terms of accuracy.

Although we are comparing an ensemble to a single model, we can see ensemble (E1) is over 3x faster to decode than the single model (S7). Additionally, we have found that model (S4) is roughly 3x faster to train than (S7) using the same GPU resources, so (E1) is also 1.5x faster to train than a single model (S7).

References

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

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *ACL (1)*, pages 1370–1380. Citeseer.

Nadir Durrani, Barry Haddow, Philipp Koehn, and Kenneth Heafield. 2014. Edinburgh's phrase-based machine translation systems for wmt-14. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 97–104.

Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2016. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

Song Han, Huizi Mao, and William J. Dally. 2016. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *ICLR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. *CoRR*.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2014. Addressing the rare word problem in neural machine translation. *ACL 2015*.

Chris Quirk and Robert Moore. 2007. Faster beam-search decoding for phrasal statistical machine translation. *Machine Translation Summit XI*.

Holger Schwenk. 2014. http://www-lium.univ-lemans.fr/schwenk/cslm_joint_paper. [Online; accessed 03-September-2014].

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

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

Exploiting Cross-Sentence Context for Neural Machine Translation

Longyue Wang[†] **Zhaopeng Tu**[‡*] **Andy Way**[†] **Qun Liu**[†]
[†]ADAPT Centre, School of Computing, Dublin City University, Ireland
{longyue.wang, andy.way, qun.liu}@adaptcentre.ie
[‡]Tencent AI Lab, China
tuzhaopeng@gmail.com

Abstract

In translation, considering the document as a whole can help to resolve ambiguities and inconsistencies. In this paper, we propose a cross-sentence context-aware approach and investigate the influence of historical contextual information on the performance of neural machine translation (NMT). First, this history is summarized in a hierarchical way. We then integrate the historical representation into NMT in two strategies: 1) a warm-start of encoder and decoder states, and 2) an auxiliary context source for updating decoder states. Experimental results on a large Chinese-English translation task show that our approach significantly improves upon a strong attention-based NMT system by up to +2.1 BLEU points.

1 Introduction

Neural machine translation (NMT) has been rapidly developed in recent years (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015; Tu et al., 2016). The encoder-decoder architecture is widely employed, in which the encoder summarizes the source sentence into a vector representation, and the decoder generates the target sentence word by word from the vector representation. Using the encoder-decoder framework as well as gating and attention techniques, it has been shown that the performance of NMT has surpassed the performance of traditional statistical machine translation (SMT) on various language pairs (Luong et al., 2015).

The continuous vector representation of a symbol encodes multiple dimensions of similarity, equivalent to encoding more than one meaning of a word. Consequently, NMT needs to spend a substantial amount of its capacity in disambiguating source and target words based on the context defined by a source sentence (Choi et al., 2016). Consistency is another critical issue in document-level translation, where a repeated term should keep the same translation throughout the whole document (Xiao et al., 2011; Carpuat and Simard, 2012). Nevertheless, current NMT models still process a documents by translating each sentence alone, suffering from inconsistency and ambiguity arising from a single source sentence. These problems are difficult to alleviate using only limited intra-sentence context.

The cross-sentence context, or global context, has proven helpful to better capture the meaning or intention in sequential tasks such as query suggestion (Sordoni et al., 2015) and dialogue modeling (Vinyals and Le, 2015; Serban et al., 2016). The leverage of global context for NMT, however, has received relatively little attention from the research community.[1] In this paper, we propose a cross-sentence context-aware NMT model, which considers the influence of *previous source sentences* in the same document.[2]

Specifically, we employ a hierarchy of Recurrent Neural Networks (RNNs) to summarize the cross-sentence context from source-side previous sentences, which deploys an additional document-level RNN on top of the sentence-level RNN encoder (Sordoni et al., 2015). After obtaining the global context, we design several strategies to integrate it into NMT to translate the current sentence:

- *Initialization*, that uses the history represen-

[*] Corresponding Author: Zhaopeng Tu

[1] To the best of our knowledge, our work and Jean et al. (2017) are two independently early attempts to model cross-sentence context for NMT.

[2] In our preliminary experiments, considering target-side history inversely harms translation performance, since it suffers from serious error propagation problems.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2826–2831
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

tation as the initial state of the encoder, decoder, or both;

- *Auxiliary Context*, that uses the history representation as static cross-sentence context, which works together with the dynamic intra-sentence context produced by an attention model, to good effect.
- *Gating Auxiliary Context*, that adds a gate to Auxiliary Context, which decides the amount of global context used in generating the next target word at each step of decoding.

Experimental results show that the proposed *initialization* and *auxiliary context* (w/ or w/o gating) mechanisms significantly improve translation performance individually, and combining them achieves further improvement.

2 Approach

Given a source sentence $\mathbf{x}_m$ to be translated, we consider its K previous sentences in the same document as cross-sentence context $C = \{\mathbf{x}_{m-K}, ..., \mathbf{x}_{m-1}\}$. In this section, we first model C, which is then integrated into NMT.

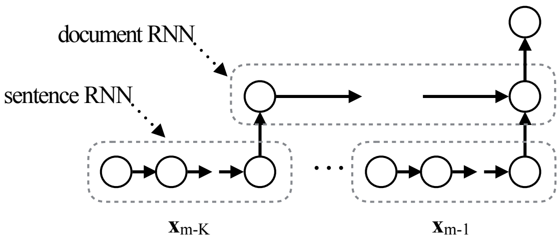


Figure 1: Summarizing global context with a hierarchical RNN ($\mathbf{x}_k$ is the k-th source sentence).

2.1 Summarizing Global Context

As shown in Figure 1, we summarize the representation of C in a hierarchical way:

Sentence RNN For a sentence $\mathbf{x}_k$ in C, the sentence RNN reads the corresponding words $\{x_{1,k}, ..., x_{n,k}, \dots, x_{N,k}\}$ sequentially and updates its hidden state:

$$h_{n,k} = f(h_{n-1,k}, x_{n,k}) \tag{1}$$

where $f(\cdot)$ is an activation function, and $h_{n,k}$ is the hidden state at time n. The last state $h_{N,k}$ stores order-sensitive information about all the words in $\mathbf{x}_k$, which is used to represent the summary of the whole sentence, i.e. $S_k \equiv h_{N,k}$. After processing each sentence in C, we can obtain all sentence-level representations, which will be fed into document RNN.

Document RNN It takes as input the sequence of the above sentence-level representations $\{S_1, ..., S_k, ..., S_K\}$ and computes the hidden state as:

$$h_k = f(h_{k-1}, S_k) \tag{2}$$

where h_k is the recurrent state at time k, which summarizes the previous sentences that have been processed to the position k. Similarly, we use the last hidden state to represent the summary of the global context, i.e. $D \equiv h_K$.

2.2 Integrating Global Context into NMT

We propose three strategies to integrate the history representation D into NMT:

Initialization We use D to initialize either NMT encoder, NMT decoder or both. For encoder, we use D as the initialization state rather than all-zero states as in the standard NMT (Bahdanau et al., 2015). For decoder, we rewrite the calculation of the initial hidden state $s_0 = \tanh(W_s h_N)$ as $s_0 = \tanh(W_s h_N + W_D D)$ where h_N is the last hidden state in encoder and $\{W_s, W_D\}$ are the corresponding weight metrices.

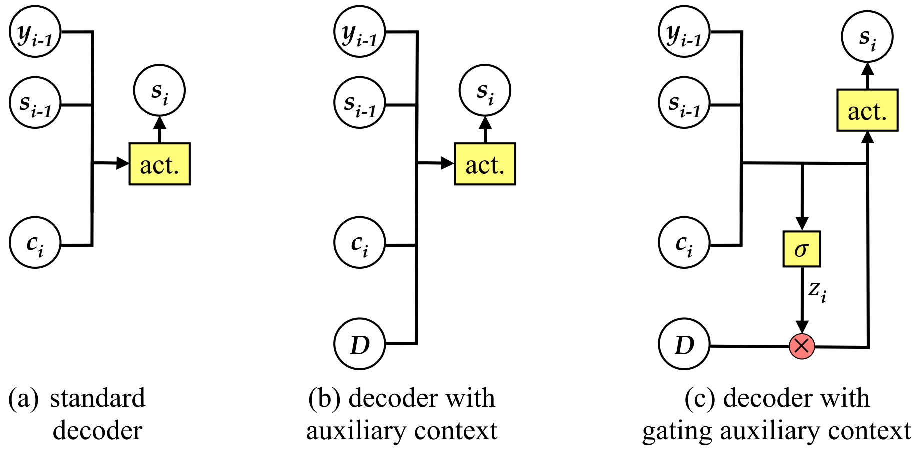
Auxiliary Context In standard NMT, as shown in Figure 2 (a), the decoder hidden state for time i is computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \tag{3}$$

where y_{i-1} is the most recently generated target word, and c_i is the intra-sentence context summarized by NMT encoder for time i. As shown in Figure 2 (b), *Auxiliary Context* method adds the representation of cross-sentence context D to jointly update the decoding state s_i:

$$s_i = f(s_{i-1}, y_{i-1}, c_i, D) \tag{4}$$

In this strategy, D serves as an auxiliary information source to better capture the meaning of the source sentence. Now the gated NMT decoder has four inputs rather than the original three ones. The concatenation $[c_i, D]$, which embeds both intra- and cross-sentence contexts, can be fed to the decoder as a single representation. We only need to modify the size of the corresponding parameter matrix for least modification effort.



(a) standard decoder

(b) decoder with auxiliary context

(c) decoder with gating auxiliary context

Figure 2: Architectures of NMT with auxiliary context integrations. *act.* is the decoder activation function, and σ is a sigmoid function.

Gating Auxiliary Context The starting point for this strategy is an observation: the need for information from the global context differs from step to step during generation of the target words. For example, global context is more in demand when generating target words for ambiguous source words, while less by others. To this end, we extend auxiliary context strategy by introducing a context gate (Tu et al., 2017a) to dynamically control the amount of information flowing from the auxiliary global context at each decoding step, as shown in Figure 2 (c).

Intuitively, at each decoding step i, the context gate looks at decoding environment (i.e., s_i, y_{i-1}, and c_i), and outputs a number between 0 and 1 for each element in D, where 1 denotes "completely transferring this" while 0 denotes "completely ignoring this". The global context vector D is then processed with an element-wise multiplication before being fed to the decoder activation layer.

Formally, the context gate consists of a sigmoid neural network layer and an element-wise multiplication operation. It assigns an element-wise weight to D, computed by

$$z_i = \sigma(U_z s_{i-1} + W_z y_{i-1} + C_z c_i) \qquad (5)$$

Here $\sigma(\cdot)$ is a logistic sigmoid function, and $\{W_z, U_z, C_z\}$ are the weight matrices, which are trained to learn when to exploit global context to maximize the overall translation performance. Note that z_i has the same dimensionality as D, and thus each element in the global context vector has its own weight. Accordingly, the decoder hidden state is updated by

$$s_i = f(s_{i-1}, y_{i-1}, c_i, z_i \otimes D) \qquad (6)$$

3 Experiments

3.1 Setup

We carried out experiments on Chinese–English translation task. As the document information is necessary when selecting the previous sentences, we collect all LDC corpora that contain document boundary. The training corpus consists of 1M sentence pairs extracted from LDC corpora[3] with 25.4M Chinese words and 32.4M English words. We chose the NIST05 (MT05) as our development set, and NIST06 (MT06) and NIST08 (MT08) as test sets. We used case-insensitive BLEU score (Papineni et al., 2002) as our evaluation metric, and sign-test (Collins et al., 2005) for calculating statistical significance.

We implemented our approach on top of an open source attention-based NMT model, Nematus[4] (Sennrich and Haddow, 2016; Sennrich et al., 2017). We limited the source and target vocabularies to the most frequent 35K words in Chinese and English, covering approximately 97.1% and 99.4% of the data in the two languages respectively. We trained each model on sentences of length up to 80 words in the training data with early stopping. The word embedding dimension was 600, the hidden layer size was 1000, and the batch size was 80. All our models considered the previous three sentences (i.e., $K = 3$) as cross-sentence context.

[3] The LDC corpora indexes are: 2003E07, 2003E14, 2004T07, 2005E83, 2005T06, 2006E24, 2006E34, 2006E85, 2006E92, 2007E87, 2007E101, 2007T09, 2008E40, 2008E56, 2009E16, 2009E95.

[4] Available at https://github.com/EdinburghNLP/nematus.

#	System	MT05	MT06	MT08	Ave.	△
1	MOSES	33.08	32.69	23.78	28.24	–
2	NEMATUS	34.35	35.75	25.39	30.57	–
3	+Init$_{\text{enc}}$	36.05	36.44†	26.65†	31.55	+0.98
4	+Init$_{\text{dec}}$	36.27	36.69†	27.11†	31.90	+1.33
5	+Init$_{\text{enc+dec}}$	36.34	36.82†	27.18†	32.00	+1.43
6	+Auxi	35.26	36.47†	26.12†	31.30	+0.73
7	+Gating Auxi	36.64	37.63†	26.85†	32.24	+1.67
8	+Init$_{\text{enc+dec}}$+Gating Auxi	**36.89**	**37.76**†	**27.57**†	**32.67**	+2.10

Table 1: Evaluation of translation quality. "Init" denotes Initialization of encoder ("enc"), decoder ("dec"), or both ("enc+dec"), and "Auxi" denotes Auxiliary Context. "†" indicates statistically significant difference ($P < 0.01$) from the baseline NEMATUS.

3.2 Results

Table 1 shows the translation performance in terms of BLEU score. Clearly, the proposed approaches significantly outperforms baseline in all cases.

Baseline (Rows 1-2) NEMATUS significantly outperforms Moses – a commonly used phrase-based SMT system (Koehn et al., 2007), by 2.3 BLEU points on average, indicating that it is a strong NMT baseline system. It is consistent with the results in (Tu et al., 2017b) (i.e., 26.93 vs. 29.41) on training corpora of similar scale.

Initialization Strategy (Rows 3-5) Init$_{\text{enc}}$ and Init$_{\text{dec}}$ improve translation performance by around +1.0 and +1.3 BLEU points individually, proving the effectiveness of warm-start with cross-sentence context. Combining them achieves a further improvement.

Auxiliary Context Strategies (Rows 6-7) The gating auxiliary context strategy achieves a significant improvement of around +1.0 BLEU point over its non-gating counterpart. This shows that, by acting as a critic, the introduced context gate learns to distinguish the different needs of the global context for generating target words.

Combining (Row 8) Finally, we combine the best variants from the initialization and auxiliary context strategies, and achieve the best performance, improving upon NEMATUS by +2.1 BLEU points. This indicates the two types of strategies are complementary to each other.

3.3 Analysis

We first investigate to what extent the mis-translated errors are fixed by the proposed system. We randomly select 15 documents (about 60 sentences) from the test sets. As shown in Table 2, we count how many related errors: i) are made by NMT (*Total*), and ii) fixed by our method (*Fixed*); as well as iii) newly generated (*New*). About *Ambiguity*, while we found that 38 words/phrases were translated into incorrect equivalents, 76% of them are corrected by our model. Similarly, we solved 75% of the *Inconsistency* errors including lexical, tense and definiteness (definite or indefinite articles) cases. However, we also observe that our system brings relative 21% new errors.

Errors	Ambiguity	Inconsistency	All
Total	38	32	70
Fixed	29	24	53
New	7	8	15

Table 2: Translation error statistics.

Hist.	这 不 等于 明着 提前 告诉 贪官 们 赶紧 转移 罪证 吗 ?
Input	能否 遏制 和 震慑 腐官 ?
Ref.	Can it inhibit and deter corrupt officials?
NMT	Can we contain and deter the *enemy*?
Our	Can it contain and deter the **corrupt officials**?

Table 3: Example translations. We italicize some *mis-translated* errors and highlight the **correct** ones in bold.

Case Study Table 3 shows an example. The word "腐官" (*corrupt officials*) is mis-translated as "*enemy*" by the baseline system. With the help of the similar word “贪官” in the previous sentence, our approach successfully correct this mistake. This demonstrates that cross-sentence context indeed helps resolve certain ambiguities.

4 Related Work

While our approach is built on top of hierarchical recurrent encoder-decoder (HRED) (Sordoni et al., 2015), there are several key differences which reflect how we have generalized from the original model. Sordoni et al. (2015) use HRED to summarize a single representation from both the current and previous sentences, which limits itself to (1) it is only applicable to encoder-decoder framework without attention model, (2) the representation can only be used to initialize decoder. In contrast, we use HRED to summarize the previous sentences alone, which provides additional cross-sentence context for NMT. Our approach is more flexible at (1) it is applicable to any encoder-decoder frameworks (e.g., with attention), (2) the cross-sentence context can be used to initialize either encoder, decoder or both.

While both our approach and Serban et al. (2016) use Auxiliary Context mechanism for incorporating cross-sentence context, there are two main differences: 1) we have separate parameters to better control the effects of the cross- and intra-sentence contexts, while they only have one parameter matrix to manage the single representation that encodes both contexts; 2) based on the intuition that not every target word generation requires equivalent cross-sentence context, we introduce a context gate (Tu et al., 2017a) to control the amount of information from it, while they don't.

At the same time, some researchers propose to use an additional set of an encoder and attention to model more information. For example, Jean et al. (2017) use it to encode and select part of the previous source sentence for generating each target word. Calixto et al. (2017) utilize global image features extracted using a pre-trained convolutional neural network and incorporate them in NMT. As additional attention leads to more computational cost, they can only incorporate limited information such as single preceding sentence in Jean et al. (2017). However, our architecture is free to this limitation, thus we use multiple preceding sentences (e.g. $K = 3$) in our experiments.

Our work is also related to multi-source (Zoph and Knight, 2016) and multi-target NMT (Dong et al., 2015), which incorporate additional source or target languages. They investigate one-to-many or many-to-one languages translation tasks by integrating additional encoders or decoders into encoder-decoder framework, and their experiments show promising results.

5 Conclusion and Future Work

We proposed two complementary approaches to integrating cross-sentence context: 1) a warm-start of encoder and decoder with global context representation, and 2) cross-sentence context serves as an auxiliary information source for updating decoder states, in which an introduced context gate plays an important role. We quantitatively and qualitatively demonstrated that the presented model significantly outperforms a strong attention-based NMT baseline system. We release the code for these experiments at `https://www.github.com/tuzhaopeng/LC-NMT`.

Our models benefit from larger contexts, and would be possibly further enhanced by other document level information, such as discourse relations. We propose to study such models for full length documents with more linguistic features in future work.

Acknowledgments

This work is supported by the Science Foundation of Ireland (SFI) ADAPT project (Grant No.:13/RC/2106). The authors also wish to thank the anonymous reviewers for many helpful comments with special thanks to Henry Elder for his generous help on proofreading of this manuscript.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*. San Diego, USA, pages 1–15.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017. Incorporating global visual features into attention-based neural machine translation. *arXiv preprint arXiv:1701.06521* .

Marine Carpuat and Michel Simard. 2012. The trouble with smt consistency. In *Proceedings of the 7th Workshop on Statistical Machine Translation*. Montreal, Quebec, Canada, pages 442–449.

Heeyoul Choi, Kyunghyun Cho, and Yoshua Bengio. 2016. Context-dependent word representation for neural machine translation. *arXiv preprint arXiv:1607.00578* .

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*. Ann Arbor, Michigan, pages 531–540.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Assocaition for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Beijing, China, pages 1723–1732.

Sebastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does neural machine translation benefit from larger context? *arXiv preprint arXiv:1704.05135* .

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1700–1709.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Prague, Czech Republic, pages 177–180.

Thang Luong, Hieu Pham, and D. Christopher Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, pages 1412–1421.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA, pages 311–318.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, et al. 2017. Nematus: a toolkit for neural machine translation. *arXiv preprint arXiv:1703.04357* .

Rico Sennrich and Barry Haddow. 2016. *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, Berlin, Germany, chapter Linguistic Input Features Improve Neural Machine Translation, pages 83–91.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. Phoenix, Arizona, pages 3776–3783.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*. Melbourne, Australia, pages 553–562.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 2014 Neural Information Processing Systems*. Montreal, Canada, pages 3104–3112.

Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. 2017a. Context gates for neural machine translation. *Transactions of the Association for Computational Linguistics* .

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017b. Neural machine translation with reconstruction. In *Proceedings of the 31th AAAI Conference on Artificial Intelligence (AAAI-17)*. San Francisco, California, USA, pages 3097–3103.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th annual meeting of the Association for Computational Linguistics*. Berlin, Germany, pages 76–85.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of the International Conference on Machine Learning, Deep Learning Workshop*. pages 1–8.

Tong Xiao, Jingbo Zhu, Shujie Yao, and Hao Zhang. 2011. Document-level consistency verification in machine translation. In *Machine Translation Summit*. Xiamen, China, volume 13, pages 131–138.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *arXiv preprint arXiv:1601.00710* .

Cross-Lingual Transfer Learning for POS Tagging without Cross-Lingual Resources

Joo-Kyung Kim[†], **Young-Bum Kim**[‡], **Ruhi Sarikaya**[‡], **Eric Fosler-Lussier**[†]
[†]The Ohio State University, Columbus, OH 43210, USA
[‡]Amazon Alexa, Seattle, WA 98121, USA

Abstract

Training a POS tagging model with cross-lingual transfer learning usually requires linguistic knowledge and resources about the relation between the source language and the target language. In this paper, we introduce a cross-lingual transfer learning model for POS tagging without ancillary resources such as parallel corpora. The proposed cross-lingual model utilizes a common BLSTM that enables knowledge transfer from other languages, and private BLSTMs for language-specific representations. The cross-lingual model is trained with language-adversarial training and bidirectional language modeling as auxiliary objectives to better represent language-general information while not losing the information about a specific target language. Evaluating on POS datasets from 14 languages in the Universal Dependencies corpus, we show that the proposed transfer learning model improves the POS tagging performance of the target languages without exploiting any linguistic knowledge between the source language and the target language.

1 Introduction

Bidirectional Long Short-Term Memory (BLSTM) based models (Graves and Schmidhuber, 2005), along with word embeddings and character embeddings, have shown competitive performance on Part-of-Speech (POS) tagging given sufficient amount of training examples (Ling et al., 2015; Lample et al., 2016; Plank et al., 2016; Yang et al., 2017).

Given insufficient training examples, we can improve the POS tagging performance by cross-lingual POS tagging, which exploits affluent POS tagging corpora from other source languages. This approach usually requires linguistic knowledge or resources about the relation between the source language and the target language such as parallel corpora (Täckström et al., 2013; Duong et al., 2013; Kim et al., 2015a; Zhang et al., 2016), morphological analyses (Hana et al., 2004), dictionaries (Wisniewski et al., 2014), and gaze features (Barrett et al., 2016).

Given no linguistic resources between the source language and the target language, transfer learning methods can be utilized instead. Transfer learning for cross-lingual cases is a type of transductive transfer learning, where the input domains of the source and the target are different (Pan and Yang, 2010) since each language has its own vocabulary space. When the input space is the same, lower layers of hierarchical models can be shared for knowledge transfer (Collobert et al., 2011; Kim et al., 2015b; Yang et al., 2017), but that approach is not directly applicable when the input spaces differ.

Yang et al. (2017) used shared character embeddings for different languages as a cross-lingual transfer method while using different word embeddings for different languages. Although the approach showed improved performance on Named Entity Recognition, it is limited to character-level representation transfer and it is not applicable for knowledge transfer between languages without overlapped alphabets.

In this work, we introduce a cross-lingual transfer learning model for POS tagging requiring no cross-lingual resources, where knowledge transfer is made in the BLSTM layers on top of word embeddings and character embeddings. Inspired by Kim et al. (2016)'s multi-task slot-filling model, our model utilizes a common BLSTM for representing language-generic information, which al-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2832–2838
Copenhagen, Denmark, September 7–11, 2017.

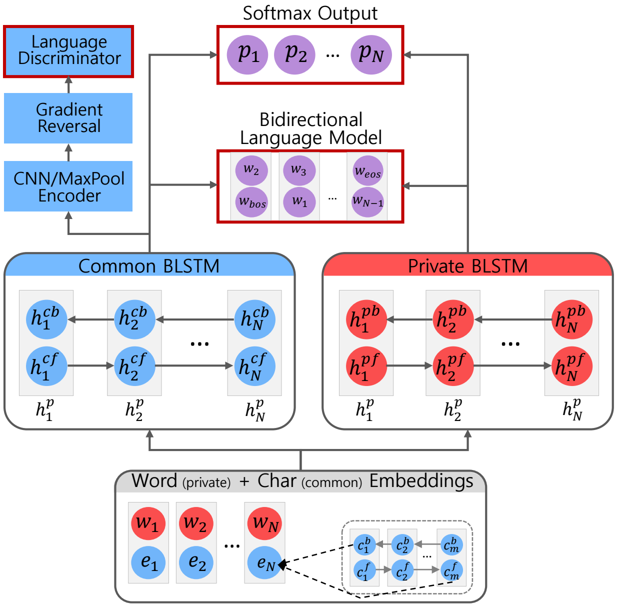

Softmax Output
p_1 p_2 ··· p_N
Language Discriminator
Gradient Reversal
CNN/MaxPool Encoder
Bidirectional Language Model
w_2 w_3 ··· w_{eos}
w_{bos} w_1 ··· w_{N-1}
Common BLSTM
h_1^{cb} h_2^{cb} ··· h_N^{cb}
h_1^{cf} h_2^{cf} h_N^{cf}
h_1^p h_2^p h_N^p
Private BLSTM
h_1^{pb} h_2^{pb} ··· h_N^{pb}
h_1^{pf} h_2^{pf} h_N^{pf}
h_1^p h_2^p h_N^p
Word (private) + Char (common) Embeddings
w_1 w_2 ··· w_N
e_1 e_2 e_N
c_1^b c_2^b ··· c_m^b
c_1^f c_2^f c_m^f

Figure 1: Model architecture: blue modules share parameters for all the languages and red modules have different parameters for different languages. w_i and e_i denote the i-th word vector and the i-th character vector composition, respectively. h_i^{cf}, h_i^{cb}, h_i^{pf}, and h_i^{pb} denote the i-th hidden outputs of the forward common LSTM, the backward common LSTM, the forward private LSTM, and the backward private LSTM, respectively. h_i^c and h_i^p denote the concatenated output of the common BLSTM and the private BLSTM, respectively. Violet circles represent target labels that are predicted with different parameters for different languages, where the inputs are output summation of the common BLSTM and the private BLSTM. The model is trained with three objectives denoted with red boxes.

lows knowledge transfer from other languages, and private BLSTMs for representing language-specific information. The common BLSTM is additionally encouraged to be language-agnostic with language-adversarial training (Chen et al., 2016) so that the language-general representations to be more compatible among different languages.

Evaluating on POS datasets from 14 different target languages with English as the source language in the Universal Dependencies corpus 1.4 (Nivre et al., 2016), the proposed model showed significantly better performance when the source language and the target language are in the same language family, and competitive performance when the language families are different.

2 Model

Cross-Lingual Training Figure 1 shows the overall architecture of the proposed model. The baseline POS tagging model is similar to Plank et al. (2016)'s model, and it corresponds to having only word+char embeddings, common BLSTM, and Softmax Output in Figure 1. Given an input word sequence, a BLSTM is used for the character sequence of each word, where the outputs of the ends of the character sequences from the forward LSTM and the backward LSTM are concatenated to the word vector of the current word to supplement the word representation. These serve as an input to a BLSTM, and an output layer are used for POS tag prediction.

For the cross-lingual transfer learning, the character embedding, the BLSTM with the character embedding (Yang et al., 2017),[1] and the common BLSTM are shared for all the given languages while word embeddings and private BLSTMs have different parameters for different languages.

The outputs of the common BLSTM and the private BLSTM of the current language are summed to be used as the input to the softmax layer to predict the POS tags of given word sequences. The loss function of the POS tagging can be formulate as:

$$\mathcal{L}_p = -\sum_{i=1}^{S}\sum_{j=1}^{N} p_{i,j} \log\left(\hat{p}_{i,j}\right), \qquad (1)$$

where S is the number of sentences in the current minibatch, N is the number of words in the current sentence, $p_{i,j}$ is the label of the j-th tag of the i-th sentence in the minibatch, and $\hat{p}_{i,j}$ is the predicted tag. In addition to this main objective, two more objectives for improving the transfer learning are described in the following subsections.

Language-Adversarial Training We encourage the outputs of the common BLSTM to be language-agnostic by using language-adversarial training (Chen et al., 2016) inspired by domain-adversarial training (Ganin et al., 2016; Bousmalis et al., 2016). First, we encode a BLSTM output sequence as a single vector using a CNN/MaxPool encoder, which is implemented the same as a CNN for text classification (Kim, 2014). The encoder is with three convolution filters whose sizes are 3, 4, and 5. For each filter, we pass the BLSTM output sequence as the input sequence and obtain a single vector from the filter output by using max pooling, and then $tanh$ activation function is used for transforming the vector. Then, the vector outputs of the three filters are concatenated and forwarded to the language discriminator through the gradient reversal layer. The discriminator is implemented

[1] We also tried isolated character-level modules but the overall performance was worse.

as a fully-connected neural network with a single hidden layer, whose activation function is Leaky ReLU (Maas et al., 2013), where we multiply 0.2 to negative input values as the outputs.

Since the gradient reversal layer is below the language classifier, the gradients minimizing language classification errors are passed back with opposed sign to the sentence encoder, which adversarially encourages the sentence encoder to be language-agnostic. The loss function of the language classifier is formulated as:

$$\mathcal{L}_a = -\sum_{i=1}^{S} l_i \log \hat{l_i}, \tag{2}$$

where S is the number of sentences, l_i is the language of the i-th sentence, and $\hat{l_i}$ is the softmax output of the tagging. Note that though the language classifier is optimized to minimize the language classification error, the gradient from the language classifier is negated so that the bottom layers are trained to be language-agnostic.

Bidirectional Language Modeling Rei (2017) showed the effectiveness of the bidirectional language modeling objective, where each time step of the forward LSTM outputs predicts the word of the next time step, and each of the backward LSTM outputs predicts the previous word. For example, if the current sentence is "I am happy", the forward LSTM predicts "am happy <eos>" and the backward LSTM predicts "<bos> I am". This objective encourages the BLSTM layers and the embedding layers to learn linguistically general-purpose representations, which are also useful for specific downstream tasks (Rei, 2017). We adopted the bidirectional language modeling objective, where the sum of the common BLSTM and the private BLSTM is used as the input to the language modeling module. It can be formulated as:

$$\begin{aligned}\mathcal{L}_l = -\sum_{i=1}^{S}\sum_{j=1}^{N} &\log\left(P\left(w_{j+1}|f_j\right)\right) + \\ &\log\left(P\left(w_{j-1}|b_j\right)\right),\end{aligned} \tag{3}$$

where f_j and b_j represent the j-th outputs of the forward direction and the backward direction, respectively, given the output sum of the common BLSTM and the private BLSTM.

All the three loss functions are added to be optimized altogether as:

$$\mathcal{L} = w_s\left(\mathcal{L}_p + \lambda\mathcal{L}_a + \lambda\mathcal{L}_l\right), \tag{4}$$

where λ is gradually increased from 0 to 1 as epoch increases so that the model is stably trained with auxiliary objectives (Ganin et al., 2016). w_s is used to give different weights to the source language and the target language. Since the source language has a larger train set and we are focusing on improving the performance of the target language, w_s is set to 1 when training the target language. For the source language, instead, it is set as the size of the target train set divided by the size of the source train set.

3 Experiments

For the evaluation, we used the POS datasets from 14 different languages in Universal Dependencies corpus 1.4 (Nivre et al., 2016). We used English as the source language, which is with 12,543 training sentences.[2] We chose datasets with 1k to 14k training sentences. The number of tag labels differs for each language from 15 to 18 though most of them are overlapped within the languages.

Table 1 shows the POS tagging accuracies of different transfer learning models when we limited the number of training sentences of the target languages to be the same as 1,280 for fair comparison among different languages. The remainder training examples of the target languages are still used for both language-adversarial training and bidirectional language modeling since the objectives do not require tag labels. Training with only the train sets in the target languages (c) showed 91.61% on average. When bidirectional language modeling objective is used (c, l), the accuracies were significantly increased to 92.82% on average. Therefore, we used the bidirectional language modeling for all the transfer learning evaluations.

With transfer learning, the three cases of using only the common BLSTM (c), using only the private BLSTMs (p), and using both (c, p) were evaluated. They showed better average accuracies than target only cases, but they showed mixed results. However, our proposed model ($c, p, l + a$), which utilizes both the common BLSTM with language-adversarial training and the private BLSTMs, showed the highest average score, 93.26%. For all the Germanic languages, where the source language also belongs to, the accuracies are significantly higher than those of

[2] The accuracies of English POS tagging are 94.01 and 94.33 for models without the bidirectional language modeling and with it, respectively.

		Target only		Source (English) → Target				
Language Family	Language	c	c,l	c,l	p,l	c,p,l	c,l+a	c,p,l+a
Germanic	Swedish	93.26	94.31	94.36	94.39	94.51	94.38	**94.63**
	Danish	92.13	93.41	93.34	93.76	94.05	93.74	**94.26**
	Dutch	83.24	84.73	85.20	84.92	84.85	84.99	**85.83**
	German	89.27	90.69	90.06	90.40	90.01	90.14	**90.71**
	Avg	89.47	90.78	90.74	90.87	90.86	90.82	**91.36**
Slavic	Slovenian	93.06	93.79	93.83	94.06	**94.20**	93.93	94.06
	Polish	91.30	91.30	91.69	**92.11**	91.86	91.77	**92.11**
	Slovak	86.53	89.56	90.11	89.88	89.98	**90.40**	90.01
	Bulgarian	93.45	95.27	95.33	95.50	95.52	95.25	**95.65**
	Avg	91.09	92.48	92.74	92.89	92.89	92.84	**92.95**
Romance	Romanian	93.20	94.09	**94.22**	94.17	94.05	93.91	94.20
	Portuguese	94.23	95.18	95.42	95.15	**95.55**	95.36	95.51
	Italian	93.80	**95.95**	95.79	95.61	95.84	95.70	95.92
	Spanish	91.94	93.34	93.34	93.31	93.29	92.94	**93.44**
	Avg	93.29	94.64	94.69	94.56	94.68	94.48	**94.77**
Indo-Iranian	Persian	93.91	94.63	94.68	94.79	94.78	94.49	**94.83**
Uralic	Hungarian	93.20	93.27	94.40	94.66	**94.69**	94.29	94.45
	Total Avg	91.61	92.82	92.98	93.05	93.08	92.95	**93.26**

Table 1: POS tagging accuracies (%) when setting the numbers of the tag-labeled training examples of the target languages to be the same as 1,280 (The remaining training examples are still used for the language modeling and the adversarial training.) c: using common BLSTM, p: using private BLSTMs, l: bidirectional language modeling objectives, a: language-adversarial training. (Underlined scores denote that the differences between the highest score of the other models and those scores are statistically insignificant with McNemar's χ^2 test with p-value $<$ 0.05.)

other transfer learning models. For the languages belonging to Slavic, Romance, or Indo-Iranian, our model shows competitive performance with the highest average accuracies among the compared models. Since languages in the same family are more likely to be similar and compatible, it is expected that the gain from the knowledge transfer to the languages in the same family to be higher than transferring to the languages in different families, which was shown in the results. This shows that utilizing both language-general representations that are encouraged to be more language-agnostic and language-specific representations effectively helps improve the POS tagging performance with transfer learning.

Table 2 shows the results when using 320 tag-labeled training sentences. In this case, transfer learning methods still show better accuracies than target-only approaches on average. However, the performance gain is weakened compared to using 1,280 labeled training sentences and there are some mixed results. In several cases, just utilizing private BLSTMs without the common BLSTM showed better accuracies than utilizing the common BLSTM.

When training with only 32 tag-labeled sentences, which is an extremely low-resourced setting, transfer learning methods still showed better accuracies than target-only methods on average. However, not using the common BLSTM in transfer learning models showed better performance than using it on average.[3] The main reason would be that we are not given a sufficient number of labeled training sentences to train both the common BLSTM and the private BLSTMs. In this case, just having private BLSTMs without the common BLSTM can show better performance. We also evaluated the opposite cases, which use all the tag-labeled training sentences in the target languages, and they showed mixed results. For example, the accuracy of German with the target only model is 93.31% while that of the proposed model is 93.04%. This is expected since transfer learning is effective when the target train set is small.

An extension of this work is utilizing multiple languages as the source languages. Since we have four languages for each of Germanic, Slavic, and Romance language families, we evaluated the performance of those languages using the other languages in the same families as the source languages expecting that languages in the same language family are more likely to be helpful each other. For the efficiency, we performed multi-task learning for multiple languages rather than differentiating the targets from sources. When we tried to use 1,280, 320, and 32 tag-labeled training sentences for each language in the multi-source settings, the results showed noticeably better per-

[3] The results in detail are shown in the first authors dissertation Kim (2017).

		Target only		Source (English) → Target				
Language Family	Language	p	p,l	p,l	c,l	p,c,l	c,l+a	p,c,l+a
Germanic	Swedish	87.43	90.49	**91.02**	90.45	90.48	90.72	90.70
	Danish	86.42	90.00	90.74	90.69	90.02	90.16	**90.79**
	Dutch	76.76	82.24	**82.61**	82.46	82.10	82.58	82.15
	German	86.25	88.95	89.10	88.69	88.93	88.08	**89.68**
	Avg	84.22	87.92	**88.37**	88.07	87.88	87.88	88.33
Slavic	Slovenian	87.02	89.97	90.29	90.00	90.32	89.58	**90.59**
	Polish	82.10	84.13	85.21	85.41	85.30	85.46	**85.50**
	Slovak	76.22	81.03	82.95	**83.40**	82.68	82.70	83.17
	Bulgarian	87.32	**92.81**	92.68	92.07	92.30	92.20	92.39
	Avg	83.16	86.98	87.78	87.72	87.65	87.48	**87.91**
Romance	Romanian	88.67	**91.44**	**91.44**	90.87	91.22	90.85	91.37
	Portuguese	90.66	93.73	93.55	93.90	93.81	93.58	**94.20**
	Italian	89.78	93.99	93.82	93.27	93.46	93.51	**94.00**
	Spanish	85.91	**91.07**	90.59	90.59	**91.07**	90.17	90.88
	Avg	88.76	92.56	92.35	92.16	92.39	92.03	**92.61**
Indo-Iranian	Persian	90.64	**92.40**	91.98	91.97	92.12	92.18	91.83
Uralic	Hungarian	89.14	90.65	91.45	91.48	90.91	**91.52**	90.72
	Total Avg	86.02	89.49	89.82	89.66	89.62	89.52	**89.86**

Table 2: POS tagging accuracies (%) with 320 tag-labeled training examples for each target language. All the training examples are still used for the other objectives.

formance than the results of using English as a single source language. Considering that utilizing 1,280*3=3,840, 320*3=960, or 32*3=96 tag labels from three other languages showed better results than using 12,543 English tag labels as the source, we can see that the knowledge transfer from multiple languages can be more helpful than that from single resource-rich source language. We also tried to use Wasserstein distance (Arjovsky et al., 2017) for the adversarial training in the multi-source settings, but there were no significant differences on average.[4]

Implementation Details All the models were optimized using ADAM (Kingma and Ba, 2015)[5] with minibatch size 32 for total 100 epochs and we picked the parameters showing the best accuracy on the development set to report the score on the test set. The dimensionalites of all the BLSTM related layers follow Plank et al. (2016)'s model. Each word vector is 128 dimensional and each character vector is 100 dimensional. They are randomly initialized with Xavier initialization (Glorot and Bengio, 2010). For stable training, we use gradient clipping, where the threshold is set to 5. The dimensionality of each hidden output of LSTMs is 100, and the hidden outputs of both forward LSTM and backward LSTM are concatenated, thereby the output of each BLSTM for each time step is 200. Therefore, the input to the common BLSTM and the private BLSTM is 128+200=328 dimensional. The inputs and the outputs of the BLSTMs are regularized with dropout rate 0.5 (Pham et al., 2014). For the consistent dropout usages, we let the dropout masks to be identical for all the time steps of each sentence (Gal and Ghahramani, 2016). For all the BLSTMs, forget biases are initialized with 1 (Jozefowicz et al., 2015) and the other biases are initialized with 0. Each convolution filter output for the sentence encoding is 64 dimensional, and the three filter outputs are concatenated to represent each sentence with a 192 dimensional vector.

4 Conclusion

We introduced a cross-lingual transfer learning model for POS tagging which uses separate BLSTMs for language-general and language-specific representations. Evaluating on 14 different languages, including the source language improved tagging accuracies in almost all the cases. Specifically, our model showed noticeably better performance when the source language and the target languages belong to the same language family, and competitively performed with the highest average accuracies for target languages in different families.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. All the experiments in this work were conducted with machines at Ohio Supercomputer Center (1987).

[4] The extended work in detail are shown in Kim (2017).

[5] learning rate=0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e - 8$.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. In *International Conference on Machine Learning (ICML)*.

Maria Barrett, Frank Keller, and Anders Søgaard. 2016. Cross-lingual transfer of correlations between parts of speech and gaze features. In *Proceedings of COLING*, pages 1330–1339.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 343–351.

Xilun Chen, Ben Athiwaratkun, Yu Sun, Kilian Weinberger, and Claire Cardie. 2016. Adversarial deep averaging networks for cross-lingual sentiment classification. In *arXiv:1606.01614*.

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

Long Duong, Paul Cook, Steven Bird, and Pavel Pecina. 2013. Simpler unsupervised POS tagging with bilingual projections. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 634–639.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29 (NIPS)*, pages 1019–1027.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Franois Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research (JMLR)*, 17:1–35.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610.

Jiri Hana, Anna Feldman, and Chris Brew. 2004. A resource-light approach to Russian morphology: Tagging Russian using Czech resources. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 222–229.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 2342–2350.

Joo-Kyung Kim. 2017. *Linguistic Knowledge Transfer for Enriching Vector Representations*. Ph.D. thesis, The Ohio State University.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Young-Bum Kim, Benjamin Snyder, and Ruhi Sarikaya. 2015a. Part-of-speech taggers for low-resource languages using CCA features. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1292–1302.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. Frustratingly easy neural domain adaptation. In *Proceedings of COLING*, pages 387–396.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015b. New transfer learning techniques for disparate label sets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 473–482.

Diederik P. Kingma and Jimmy Lei Ba. 2015. ADAM: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

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

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernández Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1520–1530.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML 2013 Workshop on Deep Learning for Audio, Speech, and Language Processing*.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Yevgeni Berzak, Riyaz Ahmad Bhat, Eckhard Bick, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Fabricio Chalub, Çar Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Claudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökrmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Jan Hajič, Linh Hà M, Dag Haug, Barbora Hladká, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Jessica Kenney, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lucia Lam, Phng Lê Hng, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Keiko Sophie Mori, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Lng Nguyn Th, Huyn Nguyn Th Minh, Vitaly Nikolaev, Hanna Nurmi, Petya Osenova, Robert Östling, Lilja Øvrelid, Valeria Paiva, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalnia, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Baiba Saulīte, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Carolyn Spadine, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jing Xian Wang, Jonathan North Washington, Mats Wirén, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2016. Universal dependencies 1.4. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University.

Ohio Supercomputer Center. 1987. Ohio supercomputer center. http://osc.edu/ark:/19495/f5s1ph73.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22(10):1345–1359.

Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 285–290.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 412–418.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics (TACL)*, 1:1–12.

Guillaume Wisniewski, Nicolas Pécheux, Souhir Gahbiche-Braham, and François Yvon. 2014. Cross-lingual part-of-speech tagging through ambiguous learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1779–1785.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *International Conference on Learning Representations (ICLR)*.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten pairs to tag – multilingual POS tagging via coarse mapping between embeddings. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 1307–1317.

Image Pivoting for Learning Multilingual Multimodal Representations

Spandana Gella **Rico Sennrich** **Frank Keller** **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
{spandana.gella, rico.sennrich}@ed.ac.uk
{keller,mlap}@inf.ed.ac.uk

Abstract

In this paper we propose a model to learn multimodal multilingual representations for matching images and sentences in different languages, with the aim of advancing multilingual versions of image search and image understanding. Our model learns a common representation for images and their descriptions in two different languages (which need not be parallel) by considering the image as a pivot between two languages. We introduce a new pairwise ranking loss function which can handle both symmetric and asymmetric similarity between the two modalities. We evaluate our models on image-description ranking for German and English, and on semantic textual similarity of image descriptions in English. In both cases we achieve state-of-the-art performance.

1 Introduction

In recent years there has been a significant amount of research in language and vision tasks which require the joint modeling of texts and images. Examples include text-based image retrieval, image description and visual question answering. An increasing number of large image description datasets has become available (Hodosh et al., 2013; Young et al., 2014; Lin et al., 2014) and various systems have been proposed to handle the image description task as a generation problem (Bernardi et al., 2016; Mao et al., 2015; Vinyals et al., 2015; Fang et al., 2015). There has also been a great deal of work on sentence-based image search or cross-modal retrieval where the objective is to learn a joint space for images and text (Hodosh et al., 2013; Frome et al., 2013; Karpathy et al., 2014; Kiros et al., 2015; Socher et al., 2014; Donahue et al., 2015).

Previous work on image description generation or learning a joint space for images and text has mostly focused on English due to the availability of English datasets. Recently there have been attempts to create image descriptions and models for other languages (Funaki and Nakayama, 2015; Elliott et al., 2016; Rajendran et al., 2016; Miyazaki and Shimizu, 2016; Specia et al., 2016; Li et al., 2016; Hitschler et al., 2016; Yoshikawa et al., 2017).

Most work on learning a joint space for images and their descriptions is based on Canonical Correlation Analysis (CCA) or neural variants of CCA over representations of image and its descriptions (Hodosh et al., 2013; Andrew et al., 2013; Yan and Mikolajczyk, 2015; Gong et al., 2014; Chandar et al., 2016). Besides CCA, a few others learn a visual-semantic or multimodal embedding space of image descriptions and representations by optimizing a ranking cost function (Kiros et al., 2015; Socher et al., 2014; Ma et al., 2015; Vendrov et al., 2016) or by aligning image regions (objects) and segments of the description (Karpathy et al., 2014; Plummer et al., 2015) in a common space. Recently Lin and Parikh (2016) have leveraged visual question answering models to encode images and descriptions into the same space.

However, all of this work is targeted at monolingual descriptions, i.e., mapping images and descriptions in a single language onto a joint embedding space. The idea of pivoting or bridging is not new and language pivoting is well explored for machine translation (Wu and Wang, 2007; Firat et al., 2016) and to learn multilingual multimodal representations (Rajendran et al., 2016; Calixto et al., 2017). Rajendran et al. (2016) propose a

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2839–2845
Copenhagen, Denmark, September 7–11, 2017.

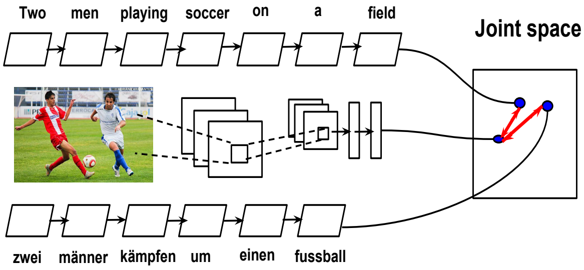


Figure 1: Our multilingual multimodal model with image as pivot

model to learn common representations between M views and assume there is parallel data available between a pivot view and the remaining $M-1$ views. Their multimodal experiments are based on English as the pivot and use large parallel corpora available between languages to learn their representations.

Related to our work Calixto et al. (2017) proposed a model for creating multilingual multimodal embeddings. Our work is different from theirs in that we choose the image as the pivot and use a different similarity function. We also propose a single model for learning representations of images and multiple languages, whereas their model is language-specific.

In this paper, we learn multimodal representations in multiple languages, i.e., our model yields a joint space for images and text in multiple languages using the image as a pivot between languages. We propose a new objective function in a multitask learning setting and jointly optimize the mappings between images and text in two different languages.

2 Dataset

We experiment with the Multi30k dataset, a multilingual extension of Flickr30k corpus (Young et al., 2014) consisting of English and German image descriptions (Elliott et al., 2016). The Multi30K dataset has 29k, 1k and 1k images in the train, validation and test splits respectively, and contains two types of multilingual annotations: (i) a corpus of one English description per image and its translation into German; and (ii) a corpus of five independently collected English and German descriptions per image. We use the independently collected English and German descriptions to train our models. Note that these descriptions are not translations of each other, i.e., they are not parallel, although they describe the same image.

3 Problem Formulation

Given an image i and its descriptions c_1 and c_2 in two different languages our aim is to learn a model which maps i, c_1 and c_2 onto same common space $\mathbb{R}^N$ (where N is the dimensionality of the embedding space) such that the image and its gold-standard descriptions in both languages are mapped close to each other (as shown in Figure 1). Our model consists of the embedding functions f_i and f_c to encode images and descriptions and a scoring function S to compute the similarity between a description–image pair.

In the following we describe two models: (i) the PIVOT model that uses the image as pivot between the description in both the languages; (ii) the PARALLEL model that further forces the image descriptions in both languages to be closer to each other in the joint space. We build two variants of PIVOT and PARALLEL with different similarity functions S to learn the joint space.

3.1 Multilingual Multimodal Representation Models

In both PIVOT and PARALLEL we use a deep convolutional neural network architecture (CNN) to represent the image i denoted by $f_i(i) = W_i \cdot CNN(i)$ where W_i is a learned weight matrix and $CNN(i)$ is the image vector representation. For each language we define a recurrent neural network encoder $f_c(c_k) = GRU(c_k)$ with gated recurrent units (GRU) activations to encode the description c_k.

In PIVOT, we use monolingual corpora from multiple languages of sentences aligned with images to learn the joint space. The intuition of this model is that an image is a universal representation across all languages, and if we constrain a sentence representation to be closer to image, sentences in different languages may also come closer. Accordingly we design a loss function as follows:

$$loss_{pivot} = \sum_k \left[\sum_{(c_k,i)} \left(\sum_{c'_k} \max\{0, \alpha - S(c_k,i) + S(c'_k,i)\} + \sum_{i'} \max\{0, \alpha - S(c_k,i) + S(c_k,i')\} \right) \right] \quad (1)$$

where k stands for each language. This loss function encourages the similarity $S(c_k, i)$ between gold-standard description c_k and image i to be greater than any other irrelevant description c'_k by a margin α. A similar loss function is useful for learning multimodal embeddings in a single language (Kiros et al., 2015). For each minibatch, we obtain invalid descriptions by selecting descriptions of other images except the current image of interest and vice-versa.

In PARALLEL, in addition to making an image similar to a description, we make multiple descriptions of the same image in different languages similar to each other, based on the assumption that these descriptions, although not parallel, share some commonalities. Accordingly we enhance the previous loss function with an additional term:

$$\begin{aligned} loss_{para} = loss_{pivot} + \sum_{(c_1,c_2)} \Bigg(\sum_{c'_1} \max\{0, \alpha - S(c_1,c_2) \\ + S(c'_1,c_2)\} + \sum_{c'_2} \max\{0, \alpha - S(c_1,c_2) + S(c_1,c'_2)\} \Bigg) \end{aligned} \quad (2)$$

Note that we are iterating over all pairs of descriptions (c_1, c_2), and maximizing the similarity between descriptions of the same image and at the same time minimizing the similarity between descriptions of different images.

We learn models using two similarity functions: symmetric and asymmetric. For the former we use cosine similarity and for the latter we use the metric of Vendrov et al. (2016) which is useful for learning embeddings that maintain an order, e.g., dog and cat are more closer to pet than animal while being distinct. Such ordering is shown to be useful in building effective multimodal space of images and texts. An analogy in our setting would be two descriptions of an image are closer to the image while at the same time preserving the identity of each (which is useful when sentences describe two different aspects of the image). The similarity metric is defined as:

$$S(a,b) = -||max(0, b-a)||^2 \quad (3)$$

where a and b are embeddings of image and description.

We call the symmetric similarity variants of our models as PIVOT-SYM and PARALLEL-SYM, and the asymmetric variants PIVOT-ASYM and PARALLEL-ASYM.

4 Experiments and Results

We test our model on the tasks of image-description ranking and semantic textual similarity. We work with each language separately. Since we learn embeddings for images and languages in the same semantic space, our hope is that the training data for each modality or language acts complementary data for the another modality or language, and thus helps us learn better embeddings.

Experiment Setup We sampled minibatches of size 64 images and their descriptions, and drew all negative samples from the minibatch. We trained using the Adam optimizer with learning rate 0.001, and early stopping on the validation set. Following Vendrov et al. (2016) we set the dimensionality of the embedding space and the GRU hidden layer N to 1024 for both English and German. We set the dimensionality of the learned word embeddings to 300 for both languages, and the margin α to 0.05 and 0.2, respectively, to learn asymmetric and symmetric similarity-based embeddings.[1] We keep all hyperparameters constant across all models. We used the L2 norm to mitigate over-fitting (Kiros et al., 2015). We tokenize and truecase both English and German descriptions using the Moses Decoder scripts.[2]

To extract image features, we used a convolutional neural network model trained on 1.2M images of 1000 class ILSVRC 2012 object classification dataset, a subset of ImageNet (Russakovsky et al., 2015). Specifically, we used VGG 19-layer CNN architecture and extracted the activations of the penultimate fully connected layer to obtain features for all images in the dataset (Simonyan and Zisserman, 2015). We use average features from 10 crops of the re-scaled images.[3]

Baselines As baselines we use monolingual models, i.e., models trained on each language separately. Specifically, we use Visual Semantic Embeddings (VSE) of Kiros et al. (2015) and Order Embeddings (OE) of Vendrov et al. (2016). We

[1] We constrain the embeddings of descriptions and images to have non-negative entries when using asymmetric similarity by taking their absolute value.

[2] https://github.com/moses-smt/mosesdecoder/tree/master/scripts

[3] We rescale images so that the smallest side is 256 pixels wide, we take 224 × 224 crops from the corners, center, and their horizontal reflections to get 10 crops for the image.

System	Text to Image				Image to Text			
	R@1	R@5	R@10	Mr	R@1	R@5	R@10	Mr
VSE (Kiros et al., 2015)	23.3	53.6	65.8	5	31.6	60.4	72.7	3
OE (Vendrov et al., 2016)	25.8	**56.5**	67.8	4	**34.8**	**63.7**	74.8	3
PIVOT-SYM	23.5	53.4	65.8	5	31.6	61.2	73.8	3
PARALLEL-SYM	24.7	53.9	65.7	5	31.7	62.4	74.1	3
PIVOT-ASYM	26.2	56.4	**68.4**	4	33.8	62.8	**75.2**	3
PARALLEL-ASYM	**27.1**	56.2	66.9	4	31.5	61.4	74.7	3

Table 1: Image-description ranking results of English on Flickr30k test data.

System	Text to Image				Image to Text			
	R@1	R@5	R@10	Mr	R@1	R@5	R@10	Mr
VSE (Kiros et al., 2015)	20.3	47.2	60.1	6	29.3	58.1	71.8	4
OE (Vendrov et al., 2016)	21.0	48.5	60.4	6	26.8	57.5	70.9	4
PIVOT-SYM	20.3	46.4	59.2	6	26.9	56.6	70.0	4
PARALLEL-SYM	20.9	46.9	59.3	6	28.2	57.7	71.3	4
PIVOT-ASYM	**22.5**	49.3	61.7	6	28.2	**61.9**	**73.4**	**3**
PARALLEL-ASYM	21.8	**50.5**	**62.3**	**5**	**30.2**	60.4	72.8	**3**

Table 2: Image-description ranking results of German on Flickr30k test data.

Image	Descriptions	Image Rank		
		OE	PIVOT	PARALLEL
	2 Menschen auf der Straße mit Megafon	141	37	6
	two people in blue shirts are outside with a bullhorn	85	7	3
	ein Verkäufer mit weißem Hut und blauem Hemd , verkauft Kartoffeln oder ähnliches an Männer und Frauen	36	1	3
	at an outdoor market , a small group of people stoop to buy potatoes from a street vendor , who has his goods laid out on the ground	24	2	2

Table 3: The rank of the gold-standard image when using each German and English descriptions as a query on models trained using asymmetric similarity.

use a publicly available implementation to train both VSE and OE.[4]

4.1 Image-Description Ranking Results

To evaluate the multimodal multilingual embeddings, we report results on an image-description ranking task. Given a query in the form of a description or an image, the task its to retrieve all images or descriptions sorted based on the relevance. We use the standard ranking evaluation metrics of recall at position k (R@K, where higher is better) and median rank (Mr, where lower is better) to evaluate our models. We report results for both English and German descriptions. Note that we have one single model for both languages.

In Tables 1 and 2 we present the ranking results of the baseline models of Kiros et al. (2015) and Vendrov et al. (2016) and our proposed PIVOT and PARALLEL models. We do not compare our image-description ranking results with Calixto et al. (2017) since they report results on half of validation set of Multi30k whereas our results are on the publicly available test set of Multi30k. For English, PIVOT with asymmetric similarity is either competitive or better than monolingual models and symmetric similarity, especially in the R@10 category it obtains state-of-the-art. For German, both PIVOT and PARALLEL with the asymmetric scoring function outperform monolingual models and symmetric similarity. We also observe that the German ranking experiments benefit the most from the multilingual signal. A reason for this could be that the German description corpus has many singleton words (more than 50% of the vocabulary) and English description mapping might have helped in learning better semantic embeddings. These results suggest that the multilingual signal could be used to learn better multimodal embeddings, irrespective of the language. Our results also show that the asymmetric scoring function can help learn better embeddings. In Table 3 we present a few examples where PIVOT-ASYM and PARALLEL-ASYM models performed better on both the languages compared to baseline order embedding model even using descriptions of very different lengths as queries.

4.2 Semantic Textual Similarity Results

In the semantic textual similarity task (STS), we use the textual embeddings from our model to compute the similarity between a pair of sen-

[4] https://github.com/ivendrov/order-embedding

Model	VF	2012	2014	2015
Shared Task Baseline	–	29.9	51.3	60.4
STS Best System	–	87.3	83.4	86.4
GRAN (Wieting et al., 2017)	–	83.7	84.5	85.0
MLMME (Calixto et al., 2017)	VGG19	–	72.7	79.7
VSE (Kiros et al., 2015)	VGG19	80.6	82.7	89.6
OE (Vendrov et al., 2016)	VGG19	82.2	84.1	90.8
PIVOT-SYM	VGG19	80.5	81.8	89.2
PARALLEL-SYM	VGG19	82.0	81.4	90.4
PIVOT-ASYM	VGG19	83.1	83.8	90.3
PARALLEL-ASYM	VGG19	**84.6**	**84.5**	**91.5**

Table 4: Results on Semantic Textual Similarity Image datasets (Pearson's $r \times 100$). Our systems that performed better than best reported shared task scores are in **bold**.

tences (image descriptions in this case). We evaluate on video task from STS-2012 and image tasks from STS-2014, STS-2015 (Agirre et al. 2012, Agirre et al. 2014, Agirre et al. 2015). The video descriptions in the STS-2012 task are from the MSR video description corpus (Chen and Dolan, 2011) and the image descriptions in STS-2014 and 2015 are from UIUC PASCAL dataset (Rashtchian et al., 2010).

In Table 4, we present the Pearson correlation coefficients of our model predicted scores with the gold-standard similarity scores provided as part of the STS image/video description tasks. We compare with the best reported scores for the STS shared tasks, achieved by MLMME (Calixto et al., 2017), paraphrastic sentence embeddings (Wieting et al., 2017), visual semantic embeddings (Kiros et al., 2015), and order embeddings (Vendrov et al., 2016). The shared task baseline is computed based on word overlap and is high for both the 2014 and the 2015 dataset, indicating that there is substantial lexical overlap between the STS image description datasets. Our models outperform both the baseline system and the best system submitted to the shared task. For the 2012 video paraphrase corpus, our multilingual methods performed better than the monolingual methods showing that similarity across paraphrases can be learned using multilingual signals. Similarly, Wieting et al. (2017) have reported to learn better paraphrastic sentence embeddings with multilingual signals. Overall, we observe that models learned using the asymmetric scoring function outperform the state-of-the-art on these datasets, suggesting that multilingual sharing is beneficial. Although the task has nothing to do German, because our models can make use of datasets from different languages, we were able to train on significantly larger training dataset of approximately 145k descriptions. Calixto et al. (2017) also train on a larger dataset like ours, but could not exploit this to their advantage. In Table 5 we present the example sentences with the highest and lowest difference between gold-standard and predicted semantic textual similarity scores using our best performing PARALLEL-ASYM model.

S1	S2	GT	Pred
Black bird standing on concrete.	Blue bird standing on green grass.	1.0	4.2
Two zebras are playing.	Zebras are socializing.	4.2	1.2
Three goats are being rounded up by a dog.	Three goats are chased by a dog	4.6	4.5
A man is folding paper.	A woman is slicing a pepper.	0.6	0.6

Table 5: Example sentences with gold-standard semantic textual similarity score and the predicted score using our best performing PARALLEL-ASYM model.

5 Conclusions

We proposed a new model that jointly learns multilingual multimodal representations using the image as a pivot between languages. We introduced new objective functions that can exploit similarities between images and descriptions across languages. We obtained state-of-the-art results on two tasks: image-description ranking and semantic textual similarity. Our results suggest that exploiting multilingual and multimodal resources can help in learning better semantic representations.

Acknowledgments

This work greatly benefited from discussions with Siva Reddy and Desmond Elliot. The authors would like to thank the anonymous reviewers for their helpful comments. The authors gratefully acknowledge the support of the European Research Council (Lapata: award number 681760).

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 81–91. Association for Computational Linguistics.

Eneko Agirre, Carmen Baneab, Claire Cardiec, Daniel Cerd, Mona Diabe, Aitor Gonzalez-Agirrea, Weiwei Guof, Inigo Lopez-Gazpioa, Montse Maritxalara, Rada Mihalceab, et al. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 252–263.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 385–393. Association for Computational Linguistics.

Galen Andrew, Raman Arora, Jeff A. Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1247–1255.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *Journal of Artifical Intelligence Research*, 55:409–442.

Iacer Calixto, Qun Liu, and Nick Campbell. 2017. Multilingual multi-modal embeddings for natural language processing. *arXiv preprint arXiv:1702.01101*.

Sarath Chandar, Mitesh M. Khapra, Hugo Larochelle, and Balaraman Ravindran. 2016. Correlational neural networks. *Neural Computation*, 28(2):257–285.

David L Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 190–200. Association for Computational Linguistics.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. 2016. Multi30k: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language, hosted by the 54th Annual Meeting of the Association for Computational Linguistics, VL@ACL 2016, August 12, Berlin, Germany*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. 2015. From captions to visual concepts and back. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1473–1482.

Orhan Firat, Baskaran Sankaran, Yaser Al-Onaizan, Fatos T. Yarman-Vural, and Kyunghyun Cho. 2016. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 268–277.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129.

Ruka Funaki and Hideki Nakayama. 2015. Image-mediated learning for zero-shot cross-lingual document retrieval. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal*.

Yunchao Gong, Liwei Wang, Micah Hodosh, Julia Hockenmaier, and Svetlana Lazebnik. 2014. Improving image-sentence embeddings using large weakly annotated photo collections. In *European Conference on Computer Vision*, pages 529–545. Springer.

Julian Hitschler, Shigehiko Schamoni, and Stefan Riezler. 2016. Multimodal pivots for image caption translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

Andrej Karpathy, Armand Joulin, and Fei Fei F Li. 2014. Deep fragment embeddings for bidirectional image sentence mapping. In *Advances in neural information processing systems*, pages 1889–1897.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. 2015. Unifying visual-semantic embeddings with multimodal neural language models. *Transactions of the Association for Computational Linguistics*.

Xirong Li, Weiyu Lan, Jianfeng Dong, and Hailong Liu. 2016. Adding chinese captions to images. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 271–275. ACM.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, pages 740–755.

Xiao Lin and Devi Parikh. 2016. Leveraging visual question answering for image-caption ranking. In *European Conference on Computer Vision*, pages 261–277. Springer.

Lin Ma, Zhengdong Lu, Lifeng Shang, and Hang Li. 2015. Multimodal convolutional neural networks for matching image and sentence. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2623–2631.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-rnn). *International Conference on Learning Representations*.

Takashi Miyazaki and Nobuyuki Shimizu. 2016. Cross-lingual image caption generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1780–1790. Association for Computational Linguistics.

Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. 2015. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2641–2649.

Janarthanan Rajendran, Mitesh M. Khapra, Sarath Chandar, and Balaraman Ravindran. 2016. Bridge correlational neural networks for multilingual multimodal representation learning. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 171–181.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 139–147. Association for Computational Linguistics.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of Association of Computational Linguistics*, 2:207–218.

Lucia Specia, Stella Frank, Khalil Simaan, and Desmond Elliott. 2016. A shared task on multimodal machine translation and crosslingual image description. In *Proceedings of the First Conference on Machine Translation, Berlin, Germany. Association for Computational Linguistics*.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. *International Conference on Learning Representations*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3156–3164.

John Wieting, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning paraphrastic sentence embeddings from back-translated bitext. *arXiv preprint arXiv:1706.01847*.

Hua Wu and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. *Machine Translation*, 21(3):165–181.

Fei Yan and Krystian Mikolajczyk. 2015. Deep correlation for matching images and text. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3441–3450.

Yuya Yoshikawa, Yutaro Shigeto, and Akikazu Takeuchi. 2017. STAIR captions: Constructing a large-scale japanese image caption dataset. *CoRR*, abs/1705.00823.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Neural Machine Translation with Source Dependency Representation

Kehai Chen[1]*, **Rui Wang**[2]†, **Masao Utiyama**[2], **Lemao Liu**[3],
Akihiro Tamura[4], **Eiichiro Sumita**[2] and **Tiejun Zhao**[1]
[1]Machine Intelligence & Translation Laboratory, Harbin Institute of Technology
[2]ASTREC, National Institute of Information and Communications Technology (NICT)
[3]Tencent AI Lab
[4]Graduate School of Science and Engineering, Ehime University
{khchen and tjzhao}@hit.edu.cn
{wangrui, mutiyama and eiichiro.sumita}@nict.go.jp
lemaoliu@gmail.com and tamura@cs.ehime-u.ac.jp

Abstract

Source dependency information has been successfully introduced into statistical machine translation. However, there are only a few preliminary attempts for Neural Machine Translation (NMT), such as concatenating representations of source word and its dependency label together. In this paper, we propose a novel attentional NMT with source dependency representation to improve translation performance of NMT, especially on long sentences. Empirical results on NIST Chinese-to-English translation task show that our method achieves 1.6 BLEU improvements on average over a strong NMT system.

1 Introduction

Neural Machine Translation (NMT) (Kalchbrenner and Blunsom, 2013; Bahdanau et al., 2014; Sutskever et al., 2014) relies heavily on source representations, which encode implicitly semantic information of source words by neural networks (Mikolov et al., 2013a,b). Recently, several research works have been proposed to learn richer source representation, such as multi-source information (Zoph and Knight, 2016; Firat et al., 2016), and particularly source syntactic information (Eriguchi et al., 2016; Li et al., 2017; Huadong et al., 2017; Eriguchi et al., 2017), thus improving the performance of NMT.

In this paper, we enhance source representations by dependency information, which can capture source long-distance dependency constraints for word prediction. Actually, source dependency information has been shown greatly effective in Statistical Machine Translation (SMT) (Garmash and Monz, 2014; Kazemi et al., 2015; Hadiwinoto et al., 2016; Chen et al., 2017; Hadiwinoto and Ng, 2017). In NMT, there has been a quite recent preliminary exploration (Sennrich and Haddow, 2016), in which vector representations of source word and its dependency label are simply concatenated as source input, achieving state-of-the-art performance in NMT (Bojar et al., 2016).

In this paper, we propose a novel NMT with source dependency representation to improve translation performance. Compared with the simple approach of vector concatenation, we learn the Source Dependency Representation (**SDR**) to compute dependency context vectors and alignment matrices in a more sophisticated manner, which has the potential to make full use of source dependency information. To this end, we create a dependency unit for each source word to capture long-distance dependency constraints. Then we design an *Encoder* with convolutional architecture to jointly learn SDRs (Section 3) and source dependency annotations, thus computing dependency context vectors and hidden states by a novel double-context based *Decoder* for word prediction (Section 4). Empirical results on NIST Chinese-to-English translation task show that the proposed approach achieves significant gains over the method by Sennrich and Haddow (2016), and thus delivers substantial improvements over the standard attentional NMT (Section 5).

2 Background

An NMT model consists of an *Encoder* process and a *Decoder* process, and hence it is often called *Encoder-Decoder* model (Sutskever et al., 2014; Bahdanau et al., 2014). Typically, each unit of source input $x_j \in (x_1, \ldots, x_J)$ is firstly embedded as a vector V_{x_j}, and then represented as

[*] Kehai Chen was an internship research fellow at NICT when conducting this work.

[†] Corresponding author.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2846–2852
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

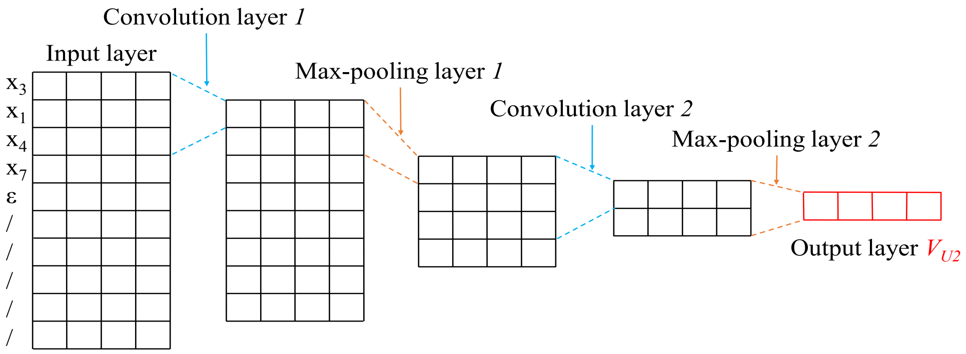


Figure 1: The CNN architecture for learning SRD.

an annotation vector h_j by

$$h_j = f_{enc}(V_{x_j}, h_{j-1}), \quad (1)$$

where f_{enc} is a bidirectional Recurrent Neural Network (**RNN**) (Bahdanau et al., 2014). These annotation vectors $H = (h_1, \dots, h_J)$ are used to generate the target word in the *Decoder*.

An RNN *Decoder* is used to compute the target word y_i probability by a softmax layer g:

$$p(y_i|y_{<i}, x) = g(\hat{y}_{i-1}, s_i, c_i), \quad (2)$$

where $\hat{y}_{i-1}$ is the previously emitted word, and s_i is an RNN hidden state for the current time step:

$$s_i = \varphi(\hat{y}_{i-1}, s_{i-1}, c_i), \quad (3)$$

and the context vector c_i is computed as a weighted sum of these source annotations h_j:

$$c_i = \sum_{j=1}^{J} \alpha_{ij} h_j, \quad (4)$$

where the normalized alignment weight α_{ij} is computed by

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{J} exp(e_{ik})}, \quad (5)$$

where e_{ij} is an alignment which indicates how well the inputs around position j and the output at the position i match:

$$e_{ij} = f(s_{i-1}, h_j). \quad (6)$$

where f is a feedforward neural network.

3 Source Dependency Representation

In order to capture source long-distance dependency constraints, we extract a dependency unit U_j for each source word x_j from dependency tree, inspired by a dependency-based bilingual composition sequence for SMT (Chen et al., 2017). The extracted U_j is defined as the following:

$$U_j = \langle PA_{x_j}, SI_{x_j}, CH_{x_j} \rangle, \quad (7)$$

where PA_{x_j}, SI_{x_j}, CH_{x_j} denote the parent, siblings and children words of source word x_j in a dependency tree. Take x_2 in Figure 2 as an example, the blue solid box U_2 denotes its dependency unit: $PA_{x_2} = \langle x_3 \rangle$, $SI_{x_2} = \langle x_1, x_4, x_7 \rangle$ and $CH_{x_2} = \langle \varepsilon \rangle$ (no child), that is, $U_2 = \langle x_3, x_1, x_4, x_7, \varepsilon \rangle$.

We design a simplified neural network following Chen et al. (2017)'s Convolutional Neural Network (**CNN**) method, to learn the SDR for each source dependency unit U_j, as shown in Figure 1. Our neural network consists of an input layer, two convolutional layers, two pooling layers and an output layer:

- **Input layer**: the input layer takes words of a dependency unit U_j in the form of embedding vectors $n \times d$, where n is the number of words in a dependency unit and d is vector dimension of each word. In our experiments, we set n to 10,[1] and d is 620. For dependency units shorter than 10, we perform "/" padding at the ending of U_j. For example, the padded U_2 is $\langle x_3, x_1, x_4, x_7, \varepsilon, /, /, /, /, / \rangle$.

[1] We find that 99% of all the source dependency units contain no more than 10 words. So if the length is more than 10, the extra words are abandoned; if the length is less than 10, the rest positions are padded with "/".

- **Convolutional layer**: the first convolution consists of one $3\times d$ convolution kernels (the stride is 1) to output an $(n\text{-}2)\times d$ matrix; the second convolution consists of one $3\times d$ convolution kernels to output a $\frac{n-2}{2}\times d$ matrix.

- **Max-Pooling layer**: the first pooling layer performs row-wise max over the two consecutive rows to output a $\frac{n-2}{4}\times d$ matrix; the second pooling layer performs row-wise max over the two consecutive rows to output a $\frac{n-2}{8}\times d$ matrix.

- **Output layer**: the output layer performs row-wise average based on the output of the second pooling layer to learn a compact d-dimension vector V_{U_j} for U_j. In our experiment, the output of the output layer is $1 \times d$-dimension vector.

It should be noted that the dependency unit is similar to the source dependency feature of Sennrich and Haddow (2016) and the SDR is the same to the source-side representation of Chen et al. (2017). In comparison with Sennrich and Haddow (2016), who concatenate the source dependency labels and word together to enhance the *Encoder* of NMT, we adapt a separate attention mechanism together with a CNN dependency *Encoder*. Compared with Chen et al. (2017), which expands the famous neural network joint model (Devlin et al., 2014) with source dependency information to improve the phrase pair translation probability estimation for SMT, we focus on source dependency information to enhance attention probability estimation and to learn corresponding dependency context and RNN hidden state for improving translation.

4 NMT with SDR

In this section, we propose two novel NMT models **SDRNMT**-*1* and **SDRNMT**-*2*, both of which can make use of source dependency information SDR to enhance *Encoder* and *Decoder* of NMT.

4.1 SDRNMT-*1*

Compared with standard attentional NMT, the *Encoder* of SDRNMT-*1* model consists of a convolutional architecture and an bidirectional RNN, as shown in Figure 2. Therefore, the proposed *Encoder* can not only learn compositional representations for dependency units but also greatly tackle the sparsity issues associated with large dependency units.

Motivated by (Sennrich and Haddow, 2016), we concatenate the V_{x_j} and V_{U_j} as input of the *Encoder*, as shown in the black dotted box in Figure 2. Source annotation vectors are learned based on the concatenated representation with dependency information:

$$h_j = f_{enc}(V_{x_j} : V_{U_j}, h_{j-1}), \tag{8}$$

where ":" denotes the operation of vectors concatenation. Finally, these learned annotation vectors are as the input of the standard NMT *Decoder* to jointly learn alignment and translation. The only difference between our method and (Sennrich and Haddow, 2016)'s method is that they only use dependency label representation instead of V_{U_j}.

4.2 SDRNMT-*2*

In SDRNMT-*1*, a single annotation, learned over concatenating word representation and SDR, is used to compute the context vector and the RNN hidden state for the current time step. To relieve more translation performance for NMT from the SDR, we propose a *double-context* mechanism, as shown in Figure 3.

First, the *Encoder* of SDRNMT-*2* consists of two independent annotations h_j and d_j:

$$\begin{aligned} h_j &= f_{enc}(V_{x_j}, h_{j-1}), \\ d_j &= f_{enc}(V_{U_j}, d_{j-1}), \end{aligned} \tag{9}$$

where $H = [h_1, \cdots, h_J]$ and $D = [d_1, \cdots, d_J]$ encode source sequential and long-distance dependency information, respectively.

The *Decoder* learns the corresponding alignment matrices and context vectors over the H and D, respectively. That is, according to eq.(6), given the previous hidden state s^s_{i-1} and s^d_{i-1}, the current alignments $e^s_{i,j}$ and $e^d_{i,j}$ are computed over source annotation vectors h_j and d_j, respectively:

$$\begin{aligned} e^s_{i,j} &= f(s^s_{i-1} + h_j), \\ e^d_{i,j} &= f(s^d_{i-1} + d_j). \end{aligned} \tag{10}$$

According to eq.(5), we further compute the current alignment $\tilde{\alpha}$:

$$\tilde{\alpha}_{i,j} = \frac{exp(\lambda e^s_{i,j} + (1-\lambda)e^d_{i,j})}{\sum_{j=1}^{J} exp(\lambda e^s_{i,j} + (1-\lambda)e^d_{i,j})}, \tag{11}$$

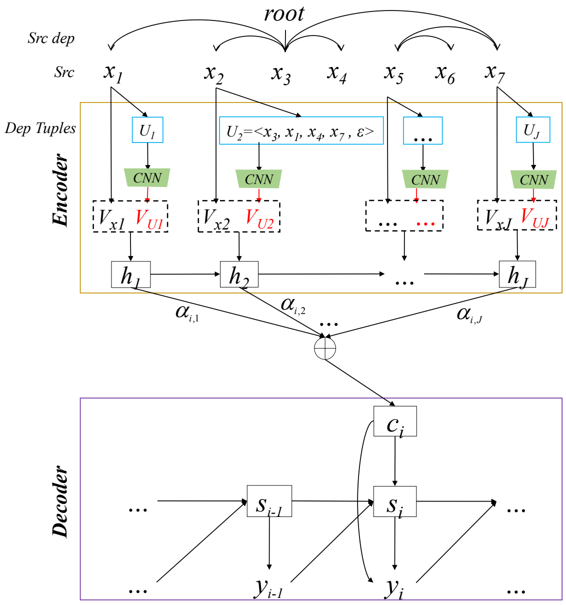


Figure 2: SDRNMT-*1* for the *i*-th time step.

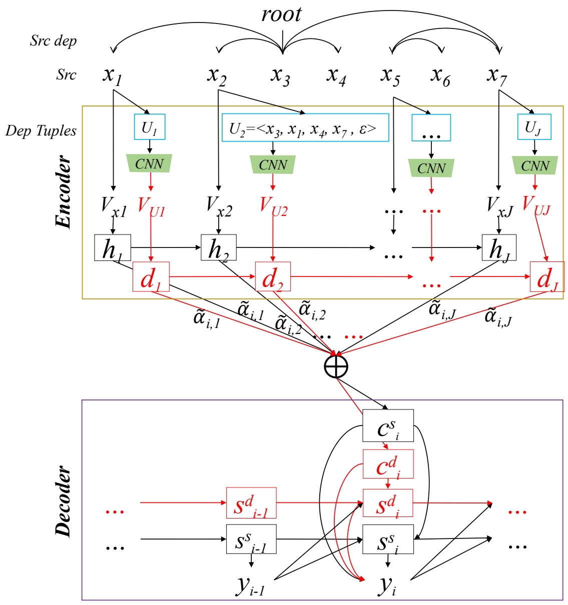


Figure 3: SDRNMT-*2* for the *i*-th time step.

where λ is a hyperparameter[2] to control the importance of H and D. Note that compared with the original alignment model only depending on the sequential annotation vectors H, the alignment weight $\tilde{\alpha}_{i,j}$ jointly compute statistic over source sequential annotation vectors H and dependency annotation vectors D.

The current context vector c_i^s and c_i^d are compute by eq.(4), respectively:

$$c_i^s = \sum_{j=1}^{J} \tilde{\alpha}_{i,j} h_j, \text{and } c_i^d = \sum_{j=1}^{J} \tilde{\alpha}_{i,j} d_j. \quad (12)$$

The current hidden state s_i^s and s_i^d are computed by eq.(3), respectively:

$$\begin{aligned} s_i^s &= \varphi(s_{i-1}^s, y_{i-1}, c_i^s), \\ s_i^d &= \varphi(s_{i-1}^d, y_{i-1}, c_i^d). \end{aligned} \quad (13)$$

Finally, according to eq.(2), the probabilities for the next target word are computed using two hidden states s_i^s and s_i^d, the previously emitted word $\hat{y}_{i-1}$, the current sequential context vector c_i^s and dependency context vector c_{d_i}:

$$p(y_i | y_{<i}, x, T) = g(\hat{y}_{i-1}, s_i^s, s_i^d, c_i^s, c_i^d). \quad (14)$$

5 Experiment

5.1 Setting up

We carry out experiments on Chinese-to-English translation. The training dataset consists of 1.42M sentence pairs extract from LDC corpora.[3] We use the Stanford dependency parser (Chang et al., 2009) to generate the dependency tree for Chinese. We choose the NIST 2002 (MT02) and the NIST 2003-2008 (MT03-08) datasets as the validation set and test sets, respectively. Case-insensitive *4*-gram NIST BLEU score (Papineni et al., 2002) is used as an evaluation metric, and *signtest* (Collins et al., 2005) is as statistical significance test.

The baseline systems include the standard Phrase-Based Statistical Machine Translation (**PBSMT**) implemented in Moses (Koehn et al., 2007) and the standard Attentional NMT (**AttNMT**) (Bahdanau et al., 2014), where only source word representation is utilized. We also compare with a state-of-the-art syntax enhanced NMT method (Sennrich and Haddow, 2016). For a fair comparison, we only utilize dependency information for (Sennrich and Haddow, 2016), called **Sennrich-deponly**. We try our best to re-implement the baseline methods on Nematus toolkit [4] (Sennrich et al., 2017).

For all NMT systems, we limit the source and target vocabularies to 30K, and the maximum sentence length is *80*. The word embedding dimension is *620*,[5] and the hidden layer dimension

[2] λ can be tuned according to a subset FBIS of training data and be set as *0.6* in the experiments.

[3] LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08, and LDC2005T06.

[4] https://github.com/EdinburghNLP/nematus

[5] For SDRNMT-*1*, the *360* dimensions are from V_{x_j} and the *260* dimensions are from V_{U_j}.

System	Dev (MT02)	MT03	MT04	MT05	MT06	MT08	AVG
PBSMT	33.15	31.02	33.78	30.33	29.62	23.53	29.66
AttNMT	36.31	34.02	37.11	32.86	32.54	25.44	32.40
Sennrich-deponly	36.68	34.51	38.09	33.37	32.96	26.96	32.98
SDRNMT-*1*	36.88	34.98*	38.14	34.61	33.58	27.06	33.32
SDRNMT-2	**37.34**	**35.91****	**38.73***	**34.18****	**33.76****	**27.64***	**34.04**

Table 1: Results on NIST Chinese-to-English Translation Task. "*" indicates statistically significant better than "Sennrich-deponly" at p-value $<$ 0.05 and "**" at p-value $<$ 0.01. **AVG** = average BLEU scores for test sets.

is *1000*, and all the layers use the dropout training technique (Hinton et al., 2012). We shuffle training set before training and the mini-batch size is *80*. Training is conducted on a single Tesla P100 GPU. All NMT models train for *15* epochs using ADADELTA (Zeiler, 2012), and the train time is *6* days, which is *25%* slower than the standard NMT.

5.2 Results and Analyses

Table 1 shows the translation performances on test sets measured in BLEU score. The AttNMT significantly outperforms PBSMT by *2.74* BLEU points on average, indicating that it is a strong baseline NMT system. The baseline Sennrich-deponly improves the performance over the AttNMT by *0.58* BLEU points on average. This indicates that the proposed source dependency constraint is beneficial for improving the performance of NMT.

Moreover, SDRNMT-*1* gains improvements of *0.92* and *0.34* BLEU points on average than the AttNMT and Sennrich-deponly. These show that the proposed SDR can more effectively capture source dependency information than vector concatenation. Especially, the proposed SDRNMT-*2* outperforms the AttNMT and Sennrich-deponly on average by *1.64* and *1.03* BLEU points. These verify that the proposed double-context method is effective for word prediction.

5.3 Effect of Translating Long Sentences

We follow (Bahdanau et al., 2014) to group sentences of similar lengths all the test sets (MT03-08), for example, "*40*" indicates that the length of sentences is between 30 and 40, and compute a BLEU score per group. As demonstrated in Figure 4, the proposed models outperform other baseline systems, especially in translating long sentences. These results show that the proposed models can effective encode long-distance dependencies to improve translation.

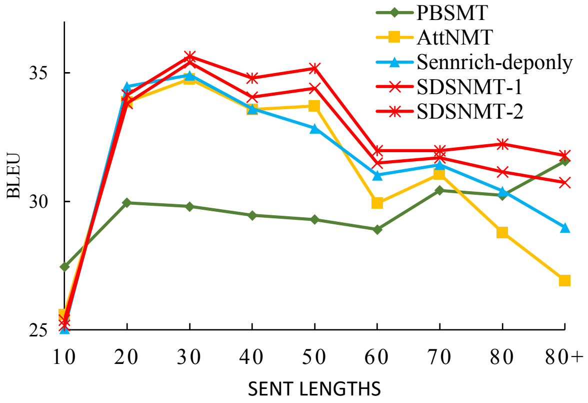


Figure 4: Translation qualities for different sentence lengths.

6 Conclusion and Future Work

In this paper, we explored the source dependency information to improve the performance of NMT. We proposed a novel attentional NMT with source dependency representation to capture source long-distance dependencies. In the future, we will try to exploit a general framework for utilizing richer syntax knowledge.

Acknowledgments

We are grateful to the anonymous reviewers for their insightful comments and suggestions. This work is partially supported by the program "Promotion of Global Communications Plan: Research, Development, and Social Demonstration of Multilingual Speech Translation Technology" of MIC, Japan. Tiejun Zhao is supported by the National Natural Science Foundation of China (NSFC) via grant 91520204 and National High Technology Research & Development Program of China (863 program) via grant 2015AA015405.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D. Manning. 2009. Discriminative reordering with Chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation at NAACL HLT 2009*, pages 51–59, Boulder, Colorado. Association for Computational Linguistics.

Kehai Chen, Tiejun Zhao, Muyun Yang, and Lemao Liu. 2017. Translation prediction with source dependency-based context representation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3166–3172, California, USA. AAAI Press.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 531–540, Ann Arbor, Michigan. Association for Computational Linguistics.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland. Association for Computational Linguistics.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 823–833, Berlin, Germany. Association for Computational Linguistics.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California. Association for Computational Linguistics.

Ekaterina Garmash and Christof Monz. 2014. Dependency-based bilingual language models for reordering in statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1689–1700, Doha, Qatar. Association for Computational Linguistics.

Christian Hadiwinoto, Yang Liu, and Hwee Tou Ng. 2016. To swap or not to swap? exploiting dependency word pairs for reordering in statistical machine translation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2943–2949, Arizona, USA. AAAI Press.

Christian Hadiwinoto and Hwee Tou Ng. 2017. A dependency-based neural reordering model for statistical machine translation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, California, USA. AAAI Press.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*.

Chen Huadong, Huang Shujian, Chiang David, and Chen Jiajun. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Arefeh Kazemi, Antonio Toral, Andy Way, Amirhassan Monadjemi, and Mohammadali Nematbakhsh. 2015. Dependency-based reordering model for constituent pairs in hierarchical smt. In *Proceedings of the 18th Annual Conference of the European Association for Machine Translation*, pages 43–50, Antalya, Turkey. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Junhui Li, Xiong Deyi, Tu Zhaopeng, Zhu Muhua, and Zhou Guodong. 2017. Modeling source syntax for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 3111–3119, USA. Curran Associates Inc.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain. Association for Computational Linguistics.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California. Association for Computational Linguistics.

Visual Denotations for Recognizing Textual Entailment

Dan Han[1]
dan.han@aist.go.jp

Pascual Martínez-Gómez[1]
pascual.mg@aist.go.jp

Koji Mineshima[2]
mineshima.koji@ocha.ac.jp

[1]Artificial Intelligence Research Center, AIST
[2]Ochanomizu University
Tokyo, Japan

Abstract

In the logic approach to Recognizing Textual Entailment, identifying phrase-to-phrase semantic relations is still an unsolved problem. Resources such as the Paraphrase Database offer limited coverage despite their large size whereas unsupervised distributional models of meaning often fail to recognize phrasal entailments. We propose to map phrases to their visual denotations and compare their meaning in terms of their images. We show that our approach is effective in the task of Recognizing Textual Entailment when combined with specific linguistic and logic features.

1 Introduction and Related Work

Recognizing Textual Entailment (RTE) is a challenging task that was described as *the best way of testing an NLP system's semantic capacity* (Cooper et al., 1994). In this task, given a text T and a hypothesis H, the objective is to recognize whether T implies H (yes), whether T contradicts H (no) or otherwise (unk). For example, given:

(T) Some men walk in the tall and green grass.
(H) Some people walk in the field.

the system needs to recognize that T implies H (yes). Although humans can easily solve these problems, machines face great difficulties (Dagan et al., 2013). RTE has been approached from different perspectives, ranging from purely statistical systems (Lai and Hockenmaier, 2014; Zhao et al., 2014) to purely logical (Bos et al., 2004; Abzianidze, 2015; Mineshima et al., 2015) and hybrid systems (Beltagy et al., 2013).

We evaluate our idea on top of a logic system since they generally offer a high precision and interpretability, which is useful to our purposes. In this approach, there are two main challenges. The first challenge is to model the logical semantic composition of sentences guided by the syntax and logical words (e.g. *most*, *not*, *some*, *every*). The second challenge is to introduce lexical knowledge that describes the relationship between words or phrases (e.g. *men* → *people*, *tall and green grass* → *field*).

Whereas the relationship *men* → *people* can be found in high precision ontological resources such as WordNet (Miller, 1995), phrasal relations such as *tall and green grass* → *field* are not available in databases such as the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013) despite their large size. Moreover, although unsupervised distributional similarity models have an infinite domain (given a compositional function on words), they often fail to identify entailments (e.g. *guitar* has a high similarity to *piano* but they do not entail each other). To address these issues, Roller et al. (2014) investigated supervised methods to identify word-to-word hypernym relations given word vectors whereas Beltagy et al. (2016) proposed a mechanism to extract phrase pairs from T and H and train a classifier to identify paraphrases in unseen T-H problems. Our approach is largely inspired by their work and our intention is to increase the performance of these phrase and sentence level entailment classifiers using multimodal features.

Our assumption is that *the same concept expressed using different phrase forms is mapped to similar visual representations* since humans tend to ground the meaning of phrases into the same visual denotation. In a similar line, Kiela and Bottou (2014) proposed a simple yet effective concatenation of pre-trained distributed word representations and visual features, whereas Izadinia et al. (2015) suggests a tighter parametric integration using a set of hand annotated phrasal entail-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2853–2859
Copenhagen, Denmark, September 7–11, 2017.

ment relations; however, their work was limited to recognizing word or phrase relations, ignoring the additional challenges that come in RTE which we show is critical. Young et al. (2014) and Lai and Hockenmaier (2014) did tackle sentence-level RTE using visual denotations. However, their approach is only applicable to those RTE problems whose words or phrases appear in the FLICKR30K corpus, which is a considerable limitation. Lai and Hockenmaier (2017) extended the approach to also recognize unseen phrasal semantic relations using a neural network augmented with conditional probabilities estimated from visual denotations. Instead, our approach is much simpler and similarly effective.

Our contribution is a method to judge phrase-to-phrase semantic relations using an asymmetric similarity scoring function between their sets of visual denotations. We identify the conditions in which this function contributes to sentence-level RTE and show empirically its benefit. Our approach is simpler than previous methods and it does not require annotated phrase relations. Moreover, this approach is not limited to specific corpora or evaluation datasets and it is potentially language independent.

2 Methodology

We formulate our framework in terms of a classifier $g_\theta : \mathcal{T} \times \mathcal{H} \rightarrow \{\text{yes}, \text{no}, \text{unk}\}$ that outputs an entailment judgment for any text $T \in \mathcal{T}$ and hypothesis $H \in \mathcal{H}$. There are three key issues in designing an effective classifier that uses visual denotations: i) to discern when it is appropriate to use visual denotations to recognize phrasal entailments, ii) to extract candidate phrase pairs and iii) to map those phrases into visual denotations[1] and measure their semantic similarity in terms of their associated images.

Textual and Logic Features The first issue is to understand the linguistic and logic limitations of visual denotations in recognizing phrasal entailments. From our observations, the linguistic phenomena that make visual denotations ineffective are word-to-word verb relations (e.g. *laughing* and *crying*) since their associated images may depict different actions with similar entities (e.g. pictures of *a baby crying* are similar to those of *a baby laughing*); antonym relations between any word in a phrase pair (e.g. similar images for *big car* and *small vehicle*); and words that denote people of different gender (e.g. *boy* versus *lady*, *man* versus *woman*) as they often display high visual similarity compared to other entities. The logic phenomena we identified signal sentences with small differences in critical words, phrases or structures, as in the presence of negations (e.g. images of *no cat* still display cats), passive-active constructions and subject-object case mismatches (e.g. images of *boy eats apple* and *apple eats boy* are similar) between T and H.

These logic phenomena can be easily detected from logic formulas with the aid of the variable unification during the theorem proving process. For instance, using event semantics (Davidson, 1967; Parsons, 1990), an active sentence *a boy eats an apple* and its corresponding passive sentence *an apple is eaten by a boy* can be compositionally mapped to the same logical formula, i.e., $\exists e \exists x \exists y (\textbf{boy}(x) \wedge \textbf{apple}(y) \wedge \textbf{eat}(e) \wedge (\textbf{subj}(e) = x) \wedge (\textbf{obj}(e) = y))$, while *a boy eats an apple* and *an apple eats a boy* are mapped to different formulas. When trying to prove the formula corresponding to H from the formula corresponding to T, one needs to unify the variables contained in these formulas, so that the non-logical predicates such as **boy**, **apple** and **eat** in T and H are aligned by taking into account logical signals.

Extract candidate phrase pairs between T and H The second issue is to find candidate phrase pairs between T and H for which we compare their visual denotations. In our running example (see Figure 1), a desirable candidate phrase pair would be *tall and green grass* and *field*. We use a tree mapping algorithm (Martínez-Gómez and Miyao, 2016) that finds node correspondences between the syntactic trees of T and H. The search is carried out bottom-up, guided by an ensemble of cost functions. This ensemble rewards word or phrase correspondences that are equal or if a linguistic relationship (i.e. synonymy, hypernymy, etc.) holds between them according to WordNet. This tree mapping implicitly defines hierarchical phrase pair correspondences between T and H. We only select those phrase pairs for which both phrases have less than 6 words. We believe that discerning the entailment relation between longer phrases should be left to the logic prover and the compositional mechanism of meaning.

[1] We approximate the visual denotations of a phrase by obtaining the images associated to that phrase.

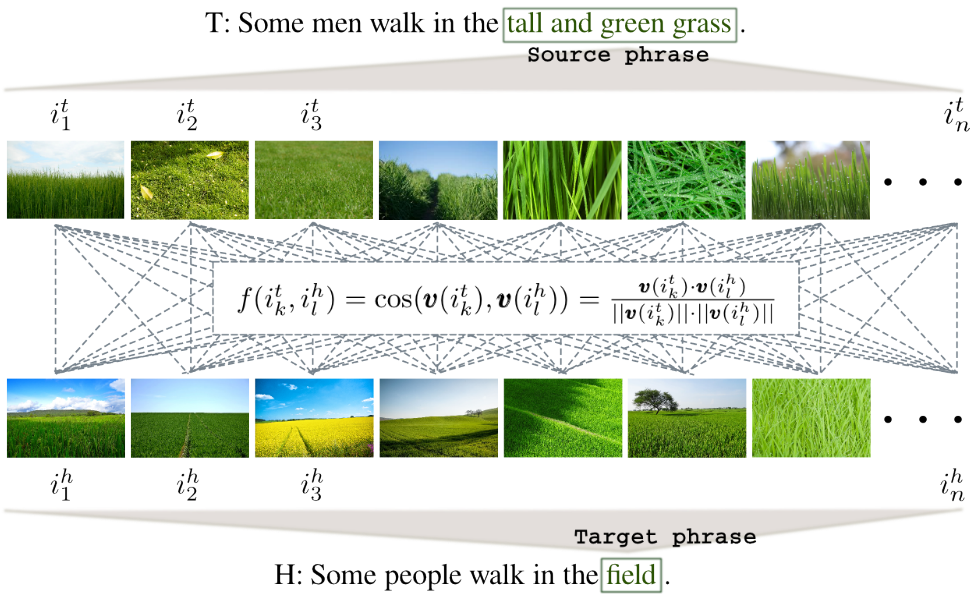

T: Some men walk in the tall and green grass.
Source phrase
i_1^t
i_2^t
i_3^t
i_n^t
$f(i_k^t, i_l^h) = \cos(\boldsymbol{v}(i_k^t), \boldsymbol{v}(i_l^h)) = \frac{\boldsymbol{v}(i_k^t)\cdot\boldsymbol{v}(i_l^h)}{||\boldsymbol{v}(i_k^t)||\cdot||\boldsymbol{v}(i_l^h)||}$
i_1^h
i_2^h
i_3^h
i_n^h
Target phrase
H: Some people walk in the field.

Figure 1: Phrase-image mappings for the phrase pair *tall and green grass* and *field* in one RTE problem.

Visual Features At this stage it remains to measure the semantic relation between the candidate phrase pairs (extracted with the tree mapping algorithm described above) using their visual denotations (see Figure 1 for a schematic diagram[2]). For this purpose, we select the phrase pairs (t, h) with highest and lowest *similarity score*. We define the similarity score as the average cosine similarity between the *best* image correspondences. That is:

$$\text{score}(t, h) = \frac{1}{|I_h|} \sum_{i_l^h \in I_h} \max_{i_k^t \in I_t} f(i_k^t, i_l^h) \quad (1)$$

where $I_t = \{i_1^t, \ldots, i_n^t\}$ are the n images associated with phrase t from T and $I_h = \{i_1^h, \ldots, i_n^h\}$ are the n images for phrase h from H, for $1 \leq l, k \leq n$. Note the asymmetry in Eq. 1 which captures semantic subsumptions (a picture of *river* is among the pictures of *body of water*). The function f returns the cosine similarity between two images:

$$f(i_k^t, i_l^h) = \cos(\mathbf{v}(i_k^t), \mathbf{v}(i_l^h)) = \frac{\mathbf{v}(i_k^t) \cdot \mathbf{v}(i_l^h)}{||\mathbf{v}(i_k^t)|| \cdot ||\mathbf{v}(i_l^h)||} \quad (2)$$

where $\mathbf{v}(i)$ is the vector representation of an image i. We obtain these vector representations concatenating the activations of the first 7 layers of GoogLeNet (Szegedy et al., 2015) as it is common practice (Kiela and Bottou, 2014).

Given the phrases with the highest and lowest similarity score,[3] we extract four features from each pair. The first feature is the similarity score itself. The other three features capture statistics of the relationship $f(I_t \times I_h)$ between the two sets of visual denotations I_t and I_h. This relationship $f(I_t \times I_h)$ is defined as the the matrix of image cosine similarities:

$$f(I_t \times I_h) = \begin{bmatrix} f(i_1^t, i_1^h) & f(i_1^t, i_2^h) & \cdots & f(i_1^t, i_n^h) \\ f(i_2^t, i_1^h) & f(i_2^t, i_2^h) & \cdots & f(i_2^t, i_n^h) \\ \vdots & \vdots & \ddots & \vdots \\ f(i_n^t, i_1^h) & f(i_n^t, i_2^h) & \cdots & f(i_n^t, i_n^h) \end{bmatrix} \quad (3)$$

Specifically, these three features are:

- $\max f(I_t \times I_h)$ returns the cosine similarity between the two most similar images. This feature is robust against polysemic phrases (at least one image associated to *pupil* is similar to at least one image associated to *student*) and hypernymy.
- average$f(I_t \times I_h)$ returns the average similarity across all image pairs and aims to measure the visual denotation overlap between both phrases in the pair.
- $\min f(I_t \times I_h)$ returns the similarity between the two most different images and gives a notion of how different the meanings of the two phrases can be.

[2] Due to copyright, images in this paper are a subset of Google Image Search results for which we have a publishing license. Nevertheless, they are faithful representatives.

[3] If there are no candidate phrase pairs, the T-H problem is ignored. If there is only one phrase pair, such a pair is used as the pair with highest and lowest score.

All features above are concatenated into a feature vector which is paired with the T-H entailment gold label to train the classifier.

3 Experiments

Our system is independent from the logic backend but we use `ccg2lambda` (Martínez-Gómez et al., 2016)[4] for its high precision and capabilities to solve word-to-word divergences using WordNet and VerbOcean (Chklovski and Pantel, 2004).

We evaluate our system on the SemEval-2014 version of the SICK dataset (Marelli et al., 2014) with train/trial/test splits of $4,500/500/4,927$ T-H pairs and a yes/no/unk label distribution of $.29/.15/.56$. We chose SICK for its relatively limited vocabulary ($2,409$ words) and short sentences. The average T and H sentence length was 10.6 where 3.6 to 3.8 words appeared in T and not in H or vice versa. We used scipy's Random Forests (Breiman, 2001) as our entailment classifier with 500 trees and feature value standardization, trained and evaluated on those T-H pairs for which `ccg2lambda` outputs *unknown* (around 71% of the problems).

Using the tree mapping algorithm,[5] we obtained an average of 9.8 phrase pairs per T-H problem. We obtained $n = 30$ images for every phrase using Google Image Search API which we consider as our visual denotations. The images and their vector representations were obtained between Sept. 2016 and Feb. 2017 using the image miner and the feature extractor of Kiela (2016).[6]

Our main baseline is `ccg2lambda` when using only WordNet and VerbOcean to account for word-to-word lexical divergences. `ccg2lambda` is augmented with a classifier `c` that uses either text and logic features `t` or image features from 10, 20, or 30 images (`10i`, `20i` or `30i`). On the training data (Table 1), `ccg2lambda` obtains an accuracy of 82.89%. Using our classifier with all features, we carried out 10 runs of a 10-fold cross-validation on the training data and we obtained an accuracy (standard deviation) of 84.14 (0.06), 84.30 (0.14) and 84.28 (0.11) when using 10, 20 and 30 images, respectively. Thus, no significant differences in accuracy were observed for different numbers of images. When using only text and logic features (`c-t`), the accuracy dropped to 76.60 (0.03); when using only image features (`c-20i`), the accuracy dropped to 82.85%. These results show that using visual denotations to recognize phrasal entailments contributes to improvements in accuracy and that the interaction with text and logic features produces further gains.

System	Accuracy	Std.
`ccg2lambda`	82.89	–
`ccg2lambda, c-t-10i`	84.14	0.06
`ccg2lambda, c-t-20i`	**84.30**	0.14
`ccg2lambda, c-t-30i`	84.28	0.11
`ccg2lambda, c-t`	76.60	0.03
`ccg2lambda, c-20i`	82.85	0.08

Table 1: Results (accuracy and standard deviation) of the classifier `c` in a cross-validation on the training split of SICK dataset using text and logic features `t` for `10i`, `20i` and `30i` images.

System	Prec.	Rec.	Acc.
`ccg2lambda` + images	90.24	71.08	**84.29**
`ccg2lambda`, only text	96.95	62.65	83.13
L&H, text + images	–	–	82.70
L&H, only text	–	–	81.50
Illinois-LH, 2014	81.56	81.87	84.57
Yin & Schütze, 2017	–	–	87.10
Baseline (majority)	–	–	56.69

Table 2: Results on the test split of SICK dataset using precision, recall and accuracy. The system "`ccg2lambda` + images" uses text and logics features and 20 images per phrase: `c-t-20i`.

On the test data, we obtained 1.1% higher accuracy (84.29 versus 83.13) over the `ccg2lambda` baseline with a standard deviation of 0.07% over 10 runs (see Table 2) when using the setting `c-t-20i`. As a comparison, Lai and Hockenmaier (2017) obtain a similar accuracy increase when using visual denotations (1.2%) with a substantially more complex approach that requires training on the SNLI dataset (Bowman et al., 2015), a much larger corpus.

The best SemEval-2014 system obtained an accuracy of 84.57 (Lai and Hockenmaier, 2014) and other heavily engineered, finely-tuned systems (Beltagy et al., 2016; Yin and Schütze, 2017) reported up to 3% points of accuracy improvement since then. Thus, our results are still below the state of the art.

[4] https://github.com/mynlp/ccg2lambda

[5] https://github.com/pasmargo/t2t-qa

[6] https://github.com/douwekiela/mmfeat

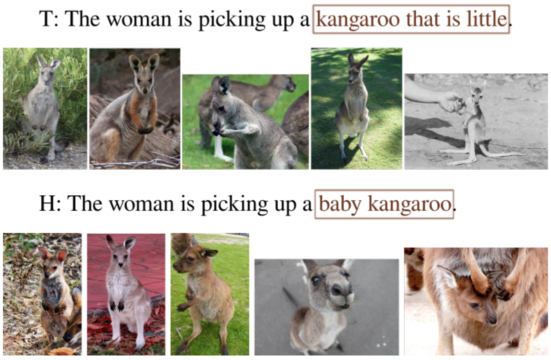

T: The woman is picking up a kangaroo that is little.
H: The woman is picking up a baby kangaroo.

Figure 2: True positive, ID: 4012; gold: yes.

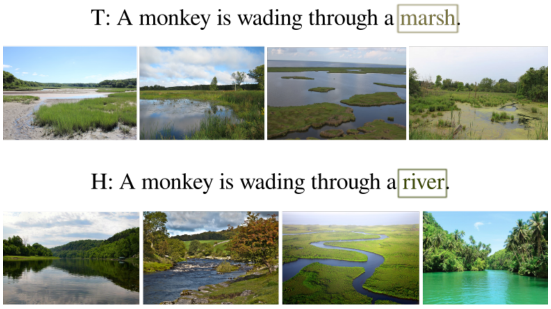

T: A monkey is wading through a marsh.
H: A monkey is wading through a river.

Figure 3: False positive, ID: 1215; gold: unk.

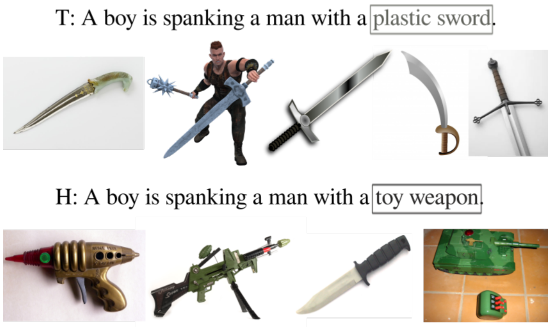

T: A boy is spanking a man with a plastic sword.
H: A boy is spanking a man with a toy weapon.

Figure 4: False negative, ID: 1318; gold: yes.

4 Error analysis

We had an average of 126 true positives (gold label yes, system label yes) and 81 false positives (gold label unk, system label yes) in our cross-validation over the training data. Figure 2 shows an example of a true positive where the tree mapping algorithm extracted the phrase pair *kangaroo that is little* and *baby kangaroo*. The image similarity features showed a high score causing the classifier to correctly produce the judgment yes. Figure 3 shows a false positive where the extracted phrase pair was *marsh* and *river* and for which the image similarity is unfortunately high. These cases are common when comparing people (*boy* and *man*) or scenery (such as *beach* and *desert*).

Figure 4 shows a false negative (gold label yes, system label unk) where the candidate phrase pair was *plastic sword* and *toy weapon*. In this case, there was only one image with a plastic sword within the images associated to *toy weapon* which may have caused the cosine similarities to be low.

5 Discussion and Conclusion

In this paper we have evaluated our method on the SICK dataset which was originally created from image captions. For that reason, the proportion of concepts with good visual denotations might be higher than in typically occurring RTE problems. Our future work is to assess the applicability of our approach into other RTE problems such as the RTE challenges, SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2017) datasets and further investigate what syntactic or semantic units can be best represented using visual denotations.

Another issue is the use of a commercial image search API as a black box to retrieve images. These search engines may include heuristics that map similar phrases or keywords into the same canonical form and that are difficult to control experimentally. However, we believe that our approach is still valid for a variety of image search mechanisms and it is generally useful to resolve lexical ambiguity at a high coverage.

We identified the conditions in which visual denotations are effective for sentence-level RTE and devised a simple scoring function to assess phrasal semantic subsumption, which may serve as the basis for more elaborated strategies. Our system is independent on the semantic parser but the entailment recognition mechanism requires a theorem prover that displays remaining sub-goals. The system and instructions are available at `https://github.com/mynlp/ccg2lambda`

Acknowledgments

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO). This project is also supported by JSPS KAKENHI Grant Number 17K12747, partially funded by Microsoft Research Asia and JST CREST Grant Number JPMJCR1301, Japan. We thank Ola Vikholt and the anonymous reviewers for their valuable comments.

References

Lasha Abzianidze. 2015. A tableau prover for natural logic and language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2492–2502, Lisbon, Portugal. Association for Computational Linguistics.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets markov: Deep semantics with probabilistic logical form. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 11–21, Atlanta, Georgia, USA. Association for Computational Linguistics.

Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. 2016. Representing meaning with a combination of logical and distributional models. *Computational Linguistics*, 42(4):763–808.

Johan Bos, Stephen Clark, Mark Steedman, James R Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th international conference on Computational Linguistics*, pages 104–111. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the web for fine-grained semantic verb relations. In *Proceedings of EMNLP 2004*, pages 33–40, Barcelona, Spain. Association for Computational Linguistics.

Robin Cooper, Richard Crouch, Jan van Eijck, Chris Fox, Josef van Genabith, Jan Jaspers, Hans Kamp, Manfred Pinkal, Massimo Poesio, Stephen Pulman, et al. 1994. FraCaS–a framework for computational semantics. *Deliverable*, D6.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing textual entailment: Models and applications*, volume 6. Morgan & Claypool Publishers.

Donald Davidson. 1967. The logical form of action sentences. In Nicholas Rescher, editor, *The Logic of Decision and Action*. University of Pittsburgh Press.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.

Hamid Izadinia, Fereshteh Sadeghi, Santosh K. Divvala, Hannaneh Hajishirzi, Yejin Choi, and Ali Farhadi. 2015. Segment-phrase table for semantic segmentation, visual entailment and paraphrasing. In *The IEEE International Conference on Computer Vision (ICCV)*.

Douwe Kiela. 2016. Mmfeat: A toolkit for extracting multi-modal features. In *Proceedings of ACL-2016 System Demonstrations*, pages 55–60, Berlin, Germany. Association for Computational Linguistics.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 36–45, Doha, Qatar. Association for Computational Linguistics.

Alice Lai and Julia Hockenmaier. 2014. Illinois-LH: A denotational and distributional approach to semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 329–334, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Alice Lai and Julia Hockenmaier. 2017. Learning to predict denotational probabilities for modeling entailment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 721–730, Valencia, Spain. Association for Computational Linguistics.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC2014*, pages 216–223.

Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2016. ccg2lambda: A compositional semantics system. In *Proceedings of ACL-2016 System Demonstrations*, pages 85–90, Berlin, Germany. Association for Computational Linguistics.

Pascual Martínez-Gómez and Yusuke Miyao. 2016. Rule extraction for tree-to-tree transducers by cost minimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 12–22, Austin, Texas. Association for Computational Linguistics.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2061, Lisbon, Portugal. Association for Computational Linguistics.

Terence Parsons. 1990. *Events in the Semantics of English: A Study in Subatomic Semantics*. The MIT Press.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1025–1036, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *CoRR*, abs/1704.05426.

Wenpeng Yin and Hinrich Schütze. 2017. Task-specific attentive pooling of phrase alignments contributes to sentence matching. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 699–709, Valencia, Spain. Association for Computational Linguistics.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Jiang Zhao, Man Lan, and Tiantian Zhu. 2014. ECNU: Expression- and message-level sentiment orientation classification in twitter using multiple effective features. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 259–264, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Sequence Effects in Crowdsourced Annotations

Nitika Mathur **Timothy Baldwin** **Trevor Cohn**
School of Computing and Information Systems
The University of Melbourne
Victoria 3010, Australia

nmathur@student.unimelb.edu.au
{tbaldwin,tcohn}@unimelb.edu.au

Abstract

Manual data annotation is a vital component of NLP research. When designing annotation tasks, properties of the annotation interface can lead to unintentional artefacts in the resulting dataset, biasing the evaluation. In this paper, we explore sequence effects where annotations of an item are affected by the preceding items. Having assigned one label to an instance, the annotator may be less (or more) likely to assign the same label to the next. During rating tasks, seeing a low quality item may affect the score given to the next item either positively or negatively. We see clear evidence of both types of effects using auto-correlation studies over three different crowdsourced datasets. We then recommend a simple way to minimise sequence effects.

1 Introduction

NLP research relies heavily on annotated datasets for training and evaluation. The design of the annotation task can influence the decisions made by annotators in subtle ways: besides the actual features of the instance being annotated, annotators are also influenced by factors such as the user interface, wording of the question, and familiarity with the task or domain.

When collecting NLP annotations, care is usually taken to ensure that the annotations are of high quality, through careful design of label sets, annotation guidelines and training of annotators (Hovy et al., 2006), methods for aggregating annotations (Passonneau and Carpenter, 2014), and intuitive user interfaces (Stenetorp et al., 2012).

Crowdsourcing has emerged as a cheaper, faster alternative to expert NLP annotations (Snow et al., 2008; Callison-Burch and Dredze, 2010; Graham et al., 2017), although it entails additional effort to filter out unskilled or opportunistic workers, e.g. through the collection of redundant repeated judgements for each instance, or including some trap questions with known answers (Callison-Burch and Dredze, 2010; Hoßfeld et al., 2014). In most annotation exercises, the order of presentation of instances is randomised to remove bias due to similarities in topic, style and vocabulary (Koehn and Monz, 2006; Bojar et al., 2016).

When crowdsourcing judgements, the normal practise (as used in the datasets we analyse) is for the item ordering to be randomised in creating a “HIT” (i.e. a single collection of items presented to a crowdworker for judgement), and then to have each HIT annotated by multiple workers, for quality control purposes. The order of items is generally fixed across all annotators of an individual HIT (Snow et al., 2008; Graham et al., 2017).

In this paper, we show that worker scores are affected by sequence bias, whereby the order of presentation can affect individuals’ assessment of an item. Since all workers see the instances in the same order, this affects any other inferences made from the data, including aggregated assessment or inferences about individual annotators (such as their overall quality or individual thresholds).

Possible explanations for sequence effects include:

Gambler’s fallacy: Once annotators have developed an idea of the distribution of scores/labels, they can come to expect even small sequences to follow the distribution. In particular, in binary annotation tasks, if they expect that True (1) and False (0) items are equally likely, then they believe the sequence 00000 (100% False and 0% True) is less likely than the sequence 01010 (50% False and 50% True). So if they assign 0 to an item,

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2860–2865
Copenhagen, Denmark, September 7–11, 2017.

they may approach the next item with a prior belief that it is more likely to be a 1 than a 0. Chen et al. (2016) showed evidence for the gambler's fallacy in decisions of loan officers, asylum judges, and baseball umpires.

Sequential contrast effects: A high quality item may raise the bar for the next item. On the other hand, a bad item may make the next item seem better in comparison (Kenrick and Gutierres, 1980; Hartzmark and Shue, to appear)

Assimilation and anchoring: The annotator uses their score of the previous item as an anchor, and adjusts the score of the current item from this anchor, based on perceived similarities and differences with the previous item. If they focus on similarities between the previous and current instance, the annotations show an assimilation effect (Geiselman et al., 1984; Damisch et al., 2006). Anchoring effects may decrease as people gain experience and expertise in the task (Wilson et al., 1996).

2 Methodology

We test whether the annotation of an instance is correlated with the annotation on previous instances, conditioned on control variables such as the gold standard (i.e. expert annotations[1]), based on the following linear model:

$$\mathrm{Y}_{i,t} = \beta_0 + \beta_1 \mathrm{Y}_{i,t-1} + \beta_2 \mathrm{Gold} + \eta \qquad (1)$$

where $\mathrm{Y}_{i,t}$ is the annotation given by an annotator i to an instance t, and η is white Gaussian noise with zero mean. We use linear regression for continuous data and logistic regression for binary data.[2] If there is no dependence between consecutive instances, and annotators assign labels/scores based only on the aspects of the current instance, then the data can be explained from the gold score (learning a positive β_2 value) and bias term (β_0), with β_1 set to zero. When we use the ground truth as a control, if β_1 is non-zero, it is evidence of mistakes being made by annotators due to sequential bias. A positive value of β_1 can be explained by priming or anchoring, and a negative value with sequential contrast effects or the gambler's fallacy. Accordingly, we test the statistical significance of the $\beta_1 \neq 0$ to determine whether sequencing effects are present in crowdsourced text corpora.

Task	All	Good	Moderate
RTE	-0.102	-0.169^{*}	-0.192^{**}
TEMPORAL	0.198	-0.567^{***}	-0.511^{***}

Table 1: Autocorrelation coefficient β_1 for RTE and TEMPORAL data. Stars denote statistical significance: * = 0.05, ** = 0.01, and *** = 0.001.

3 Experiments

We analyse several influential datasets that have been constructed through crowdsourcing, including both binary and continuous annotation tasks: recognising textual entailment, event ordering, affective text analysis, and machine translation evaluation.

3.1 Recognising Textual Entailment (RTE) and Event Temporal Ordering

First, we examine the recognising textual entailment ("RTE") and event temporal ordering ("TEMPORAL") datasets from Snow et al. (2008). In the RTE task, annotators are presented with two sentences, and are asked to judge whether the second text can be inferred from the first. With the TEMPORAL dataset, they are shown two sentences describing events, and asked to indicate which of the two events occurred first. Both datasets include both expert annotations and crowdsourced annotations constructed using Amazon Mechanical Turk ("MTurk"). On MTurk, each RTE HIT contains 20 instances, and each TEMPORAL HIT contains 10 instances, which the workers see in sequential order. For both tasks, each HIT was annotated by 10 workers.

Results We use logistic regression on worker labels against labels on the previous instance in the current HIT, with the expert judgements as a control variable. We also add an additional control, namely the percentage of True labels assigned by the worker overall, which accounts for the overall annotator bias. To calculate this, we use scores by the worker excluding the current score, to avoid giving the model any information about the current instance.

As shown in Table 1, over all workers ("All"), we find a small negative autocorrelation for both the RTE and TEMPORAL tasks. One possibility

[1] For the Machine Translation dataset described in Section 3.3, we use the mean of at least fifteen crowd workers as a proxy for expert annotations.

[2] η is not included in the case of logistic regression

is that this is biased by opportunistic workers who assign the same label to all instances in the HIT, for which we would not expect any sequential bias effects. When we exclude these workers ("Moderate"), the autocorrelation increases, and is highly statistically significant. We also show results for workers with at least 60% accuracy when compared to expert annotations ("Good"), and observe a similar effect.

3.2 Affective text analysis

In the affective text analysis task ("AFFECTIVE"), annotators are asked to rate news headlines for anger, disgust, fear, joy, sadness, and surprise on a continuous scale of 0–100. Besides these emotions, they are asked to rate sentences for (emotive) valence, i.e., how strongly negative or positive they are (-100 to $+100$). In this dataset, there are 100 headlines divided into 10 HITs, with 10 workers annotating each HIT (Snow et al., 2008). We test for autocorrelation of scores of each aspect individually, controlling for the expert scores and worker correlation with the expert scores. We also look separately at datasets of good and bad workers, based on whether the correlation with the expert annotations is greater than 0.5.

Results For individual emotions, we do not observe any significant autocorrelation ($p \geq 0.05$). As there are only 1000 annotations per emotion, we also look at results when combining data for all aspects. Though we find a statistically significant negative autocorrelation for scores of the full dataset, this disappears when we filter out bad workers (Table 2). Given the difficulty of this very subjective task, it is likely that many of workers considered 'bad' might have simply found this task too difficult or arbitrary, and thus become more prone to sequence effects.

3.3 Machine Translation Adequacy

When evaluating machine translation ("MT"), we tend to focus on adequacy: the extent to which the meaning of the reference translation is captured in the MT output. In the method of Graham et al. (2015) — the current best-practise, as adopted by WMT (Bojar et al., 2016) — annotators are asked to judge the adequacy of translations using a 100-point sliding scale which is initialised at the mid point. There are 3 marks on the scale dividing it into 4 quarters to aid workers with internal calibration. They are given no other instructions or guidelines.

	All	Good	Bad
β_1	-0.03^{*}	-0.01	-0.04^{*}
β_2	0.45^{***}	0.66^{***}	0.23^{***}

Table 2: Autocorrelation coefficient β_1 for the AFFECTIVE dataset.

In this paper, we base our analysis on the adequacy dataset of Graham et al. (2015), on Spanish-English newswire data from WMT 2013 (Bojar et al., 2013). The dataset consists of 12 HITS of 100 sentence pairs each; each HIT is annotated by at least 15 workers.

HITs are designed to include quality control items to filter out poor quality scores. In addition to 70 MT system translations, each HIT contains degraded versions of 10 of these translations, 10 reference translations by a human expert corresponding to 10 of these translations, and repeats of another 10 translations. Good workers are assumed to give high scores to the references, similar scores to the pair of repeats, and high scores to the MT system translations when compared to corresponding degraded translations. Workers who submitted scores of clearly bad quality were rejected. For the remaining workers, the Wilcoxon rank-sum test is used to test whether the score difference between the repeat judgements is less than the score difference between translations and the corresponding degraded versions. We divide these workers into "good" and "moderate" based on the threshold of $p < 0.05$.

To eliminate differences due to different internal scales, every individual worker's scores are standardised by subtracting the mean and dividing by the standard deviation of their scores. Following Graham et al. (2015), we use the average of standardised scores of at least 15 good workers as the ground truth.

We refer to the final dataset as "MT_{adeq}".

Results As this is a (practically) continuous output, we use a linear regression model, whereby the current score is predicted based on the previous score, with the mean of all worker scores as control. We also controlled for worker correlation with mean score, and position of the sentence in the HIT, but these were not significant and did not affect the autocorrelation. As seen in Table 3, we see a small but significant positive autocorrelation for good workers. The bias is much stronger with

	Good	Moderate	Bad
β_1	0.030^{***}	0.037^{***}	0.192^{***}
β_2	0.741	0.661	0.256
N items	48216	24696	17738

Table 3: MT$_{adeq}$ dataset: Autocorrelation coefficient β_1, showing sequence bias of good, moderate and bad workers.

Position	Good	Moderate	Bad
1st Tertile	0.044^{***}	0.063^{***}	0.179^{***}
2nd Tertile	0.032^{***}	0.034^{***}	0.173^{***}
3rd Tertile	0.015^{**}	0.014^{*}	0.225^{***}

Table 4: MT$_{adeq}$ dataset: Regression coefficient β_1 of adequacy scores with the previous score. We also show results for translations in the first, second or third tertile based on the position of the sentence of the HIT

bad (rejected) workers.

An interesting question is whether the bias changes as workers annotate more data, which could be ascribed to learning through the task, calibrating their internal scales, or becoming fatigued on a monotonous task. Each HIT consists of 100 sentences, and we divide the dataset into 3 equal groups based on the position of sentence in the HIT. As shown in Table 4, for good and moderate workers, the bias is stronger in the first group of sentences annotated, decreases in the second, and is much smaller in the last. This could be because workers are familiarising themselves with the task earlier on, and calibrating their scale. There is no such trend with bad quality scores, possibly because the workers are not putting in sufficient effort to produce accurate scores.

Next we assess the impact of the bias in the worst case situation. We discretize scores into low, middle and high based on equal-frequency binning, and divide the dataset into 3 groups based on the score assigned to the previous sentence. As shown in Table 5 we can see that the sentences in the "low" partition and the "high" partition have a difference of 0.18, which is highly significant;[3] moreover, this difference is likely to be sufficiently large to alter the rankings of systems in an evaluation. The bias remains even when we increase the number of workers and use the average score, as all workers scored the translations in the same order. This shows that the mean is also affected by sequence bias.

N	All	Low	Middle	High	H − L
1	0.01	−0.09	0.05	0.08	0.18^{***}
5	0.00	−0.05	−0.02	0.08	0.14^{***}
10	−0.00	−0.05	−0.04	0.09	0.13^{***}
15	−0.00	−0.05	−0.02	0.07	0.12^{***}

Table 5: MT$_{adeq}$ dataset: Translations following a low quality translation receive a lower score than those following a good translation: "All" is the mean score of all sentences in the dataset, where each sentence score is calculated as the average of N (standardised) worker scores. "Low", "Middle", and "High" are mean scores of sentences where the previous sentence annotated is of low, medium and high quality, resp. "H − L" is the difference between the average high and low scores.

Thus, it is theoretically possible to exploit sequence bias to artificially deflate (or inflate) a specific system's computed score by ordering a HIT such that the system's output is seen consistently immediately after a bad (or good) output.

4 Discussion and Conclusions

We have shown significant sequence effects across several independent crowdsourced datasets: a negative autocorrelation in the RTE and TEMPORAL datasets, and a positive autocorrelation in the MT$_{adeq}$ dataset. The negative autocorrelation can be attributed either to sequential contrast effects or the gambler's fallacy. These effects were not significant for the AFFECTIVE dataset, perhaps due to the nature of the annotation task, whereby annotations of one emotion are separated by six other annotations, thus limiting the potential for sequencing effects. It is also possible that the dataset is too small to obtain statistical significance.

MT judgements are subjective, and when people are asked to rate them on a continuous scale, they need time to calibrate their scale. We show that the sequential bias decreases for better workers as they annotate more sentences in the HIT, indicating a learning effect. Since the ordering of the systems is random, system scores obtained by averaging scores of all sentences translated by the system would be unbiased, assuming a sufficiently large sample of sentences. Thus we do not expect sequential bias to have a marked effect on system rankings or other macro-level conclusions on the basis of this data. However, the scores of in-

[3] $p < 0.001$ using Welch's two-sample t-test

dividual translations remain biased, which augurs poorly for the use of these annotations at the sentence level, such as when used in error analysis or for training automatic metrics.

Sequence problems can be easily addressed by adequate randomisation — providing each individual worker with a separate dataset that has been randomised, such that no two workers see the same ordered data. In this way sequence bias effects can be considered as independent noise sources, rather than a systematic bias, and consequently the aggregate results over several workers will remain unbiased.

This study has shown that sequence bias is real, and can distort evaluation and annotation exercises with crowd-workers. We limited our scope to binary and continuous responses, however it is likely that sequence effects are prevalent for multinomial and structured outputs, e.g., in discourse and parsing, where priming is known to have a significant effect (Reitter et al., 2006). Another important question for future work is whether sequence bias is detectable in expert annotators, not just crowd workers.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This work was supported in part by the Australian Research Council.

References

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with Amazon's Mechanical Turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 1–12, Los Angeles, USA.

Daniel Chen, Tobias J. Moskowitz, and Kelly Shue. 2016. Decision-making under the gambler's fallacy: Evidence from asylum judges, loan officers, and baseball umpires. *The Quarterly Journal of Economics*.

Lysann Damisch, Thomas Mussweiler, and Henning Plessner. 2006. Olympic medals as fruits of comparison? Assimilation and contrast in sequential performance judgments. *Journal of Experimental Psychology: Applied*, 12(3):166.

R Edward Geiselman, Nancy A Haight, and Lori G Kimata. 1984. Context effects on the perceived physical attractiveness of faces. *Journal of Experimental Social Psychology*, 20(5):409–424.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2017. Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering*, 23(1):330.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2015. Accurate evaluation of segment-level machine translation metrics. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*, pages 1183–1191, Denver, USA.

Samuel M. Hartzmark and Kelly Shue. to appear. A tough act to follow: Contrast effects in financial markets. *Journal of Finance*.

Tobias Hoßfeld, Matthias Hirth, Judith Redi, Filippo Mazza, Pavel Korshunov, Babak Naderi, Michael Seufert, Bruno Gardlo, Sebastian Egger, and Christian Keimel. 2014. Best practices and recommendations for crowdsourced QoE — lessons learned from the Qualinet Task Force "Crowdsourcing". Lessons learned from the Qualinet Task Force "Crowdsourcing" COST Action IC1003 European Network on Quality of Experience in Multimedia Systems and Services (QUALINET).

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York, USA.

Douglas T. Kenrick and Sara E. Gutierres. 1980. Contrast effects and judgments of physical attractiveness: When beauty becomes a social problem. *Journal of Personality and Social Psychology*, 38(1):131.

Philipp Koehn and Christof Monz. 2006. Manual and automatic evaluation of machine translation between European languages. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 102–121, New York, USA.

J. Rebecca Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association of Computational Linguistics*, 2(1):311–326.

David Reitter, Frank Keller, and Johanna D Moore. 2006. Computational modelling of structural priming in dialogue. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 121–124, New York, USA.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast — but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, USA.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. BRAT: A web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France.

Timothy D. Wilson, Christopher E. Houston, Kathryn M. Etling, and Nancy Brekke. 1996. A new look at anchoring effects: basic anchoring and its antecedents. *Journal of Experimental Psychology: General*, 125(4):387.

No Need to *Pay Attention*: Simple Recurrent Neural Networks Work! (for Answering "Simple" Questions)

Ferhan Ture and **Oliver Jojic**
Comcast Labs, Washington, DC 20005
{ferhan_ture, oliver_jojic}@cable.comcast.com

Abstract

First-order factoid question answering assumes that the question can be answered by a single fact in a knowledge base (KB). While this does not seem like a challenging task, many recent attempts that apply either complex linguistic reasoning or deep neural networks achieve 65%–76% accuracy on benchmark sets. Our approach formulates the task as two machine learning problems: detecting the entities in the question, and classifying the question as one of the relation types in the KB. We train a recurrent neural network to solve each problem. On the SimpleQuestions dataset, our approach yields substantial improvements over previously published results — even neural networks based on much more complex architectures. The simplicity of our approach also has practical advantages, such as efficiency and modularity, that are valuable especially in an industry setting. In fact, we present a preliminary analysis of the performance of our model on real queries from Comcast's X1 entertainment platform with millions of users every day.

1 Introduction

First-order factoid question answering (QA) assumes that the question can be answered by a single fact in a knowledge base (KB). For example, "How old is Tom Hanks" is about the [age] of [Tom Hanks]. Also referred to as *simple* questions by Bordes et al. (2015), recent attempts that apply either complex linguistic reasoning or attention-based complex neural network architectures achieve up to 76% accuracy on benchmark sets (Golub and He, 2016; Yin et al., 2016). While it is tempting to study QA systems that can handle more complicated questions, it is hard to reach reasonably high precision for unrestricted questions. For more than a decade, successful industry applications of QA have focused on first-order questions. This bears the question: are users even interested in asking questions beyond first-order (or are these use cases more suitable for interactive dialogue)? Based on voice logs from a major entertainment platform with millions of users every day, Comcast X1, we find that most existing use cases of QA fall into the first-order category.

Our strategy is to tailor our approach to first-order QA by making strong assumptions about the problem structure. In particular, we assume that the answer to a first-order question is a *single* property of a *single* entity in the KB, and decompose the task into two subproblems: (a) detecting entities in the question and (b) classifying the question as one of the relation types in the KB. We simply train a vanilla recurrent neural network (RNN) to solve each subproblem (Elman, 1990). Despite its simplicity, our approach (RNN-QA) achieves the highest reported accuracy on the SimpleQuestions dataset. While recent literature has focused on building more complex neural network architectures with attention mechanisms, attempting to generalize to broader QA, we enforce stricter assumptions on the problem structure, thereby reducing complexity. This also means that our solution is efficient, another critical requirement for real-time QA applications. In fact, we present a performance analysis of RNN-QA on Comcast's X1 entertainment system, used by millions of customers every day.

2 Related work

If knowledge is presented in a structured form (e.g., knowledge base (KB)), the standard ap-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2866–2872
Copenhagen, Denmark, September 7–11, 2017.

proach to QA is to transform the question and knowledge into a compatible form, and perform reasoning to determine which fact in the KB answers a given question. Examples of this approach include pattern-based question analyzers (Buscaldi et al., 2010), combination of syntactic parsing and semantic role labeling (Bilotti et al., 2007, 2010), as well as lambda calculus (Berant et al., 2013) and combinatory categorical grammars (CCG) (Reddy et al., 2014). A downside of these approaches is the reliance on linguistic resources/heuristics, making them language- and/or domain-specific. Even though Reddy et al. (2014) claim that their approach requires less supervision than prior work, it still relies on many English-specific heuristics and hand-crafted features. Also, their most accurate model uses a corpus of paraphrases to generalize to linguistic diversity. Linguistic parsers can also be too slow for real-time applications.

In contrast, an RNN can detect entities in the question with high accuracy and low latency. The only required resources are word embeddings and a set of questions with entity words tagged. The former can be easily trained for any language/domain in an unsupervised fashion, given a large text corpus without annotations (Mikolov et al., 2013; Pennington et al., 2014). The latter is a relatively simple annotation task that exists for many languages and domains, and it can also be synthetically generated. Many researchers have explored similar techniques for general NLP tasks (Collobert et al., 2011), such as named entity recognition (Lu et al., 2015; Hammerton, 2003), sequence labeling (Graves, 2008; Chung et al., 2014), part-of-speech tagging (Huang et al., 2015; Wang et al., 2015), chunking (Huang et al., 2015).

Deep learning techniques have been studied extensively for constructing parallel neural networks for modeling a joint probability distribution for question-answer pairs (Hsu et al., 2016; Yang et al., 2014; He et al., 2015; Mueller and Thyagarajan, 2016) and re-ranking answers output by a retrieval engine (Rao et al., 2016; Yang et al., 2016). These more complex approaches might be needed for general-purpose QA and sentence similarity, where one cannot make assumptions about the structure of the input or knowledge. However, as noted in Section 1, first-order factoid questions can be represented by an entity and a relation type, and the answer is usually stored in a structured knowledge base. Dong et al. (2015) similarly assume that the answer to a question is at most two hops away from the target entity. However, they do not propose how to obtain the target entity, since it is provided as part of their dataset. Bordes et al. (2014) take advantage of the KB structure by projecting entities, relations, and subgraphs into the same latent space. In addition to finding the target entity, the other key information to first-order QA is the relation type corresponding to the question. Many researchers have shown that classifying the question into one of the predefined types (e.g., based on patterns (Zhang and Lee, 2003) or support vector machines (Buscaldi et al., 2010)) improves QA accuracy.

3 Approach

(a) From Question to Structured Query. Our approach relies on a *knowledge base*, containing a large set of *facts*, each one representing a binary [subject, relation, object] relationship. Since we assume *first-order questions*, the answer can be retrieved from a single fact. For instance, "How old is Sarah Michelle Gellar?" can be answered by the fact:

```
[Sarah Michelle Gellar,bornOn,4/14/1977]
```

The main idea is to dissect a first-order factoid natural-language question by converting it into a structured query: {entity "Sarah Michelle Gellar", relation: `bornOn`}. The process can be modularized into two machine learning problems, namely ***entity detection*** and ***relation prediction***. In the former, the objective is to *tag* each question word as either entity or not. In the latter, the objective is to *classify* the question into one of the K relation types. We modeled both using an RNN.

We use a standard RNN architecture: Each word in the question passes through an embedding lookup layer E, projecting the one-hot vector into a d-dimensional vector x_t. A recurrent layer combines this *input representation* with the hidden layer representation from the *previous word* and applies a non-linear transformation to compute the hidden layer representation for the *current word*. The hidden representation of the final recurrent layer is projected to the output space of k dimensions and normalized into a probability distribution via soft-max.

In *relation prediction*, the question is classified into one of the 1837 classes (i.e., relation types in Freebase). In the *entity detection* task, each word is classified as either *entity* or *context* (i.e., $k = 2$).

Given a new question, we run the two RNN models to construct the structured query. Once every question word is classified as entity (denoted by `E`) or context (denoted by `C`), we can extract entity phrase(s) by grouping consecutive entity words. For example, for question "How old is Michelle Gellar", the output of entity detection is [`C C C E E`], from which we can extract a single entity "Michelle Gellar". The output of relation prediction is `bornOn`. The inferred structured query q becomes the following:
{*entityText*: "michelle gellar", *relation*: `bornOn`}

(b) Entity Linking. The textual reference to the entity (entityText in q) needs to be linked to an actual entity node in our KB. In order to achieve that, we build an *inverted index* I_{entity} that maps all n-grams of an entity ($n \in \{1, 2, 3\}$) to the entity's alias text (e.g., name or title), each with an associated $TF\text{-}IDF$ score. We also map the exact text ($n = \infty$) to be able to prioritize exact matches.

Following our running example, let us demonstrate how we construct I_{entity}. Let us assume there is a node e_i in our KB that refers to the actress "Sarah Michelle Gellar". The alias of this entity node is the name, which has three unigrams ("sarah", "michelle", "gellar"), two bigrams ("sarah michelle", "michelle gellar") and a single trigram (i.e., the entire name). Each one of these n-grams gets indexed in I_{entity} with $TF\text{-}IDF$ weights. Here is how the weights would be computed for unigram "sarah" and bigram "michelle gellar" ($\Rightarrow$ denotes mapping):

$$I_{\text{entity}}(\text{“sarah”}) \Rightarrow \{\text{node} : e_i,$$
$$\text{score} : TF\text{-}IDF(\text{“sarah”}, \text{“sarah michelle gellar”})\}$$
$$I_{\text{entity}}(\text{“michelle gellar”}) \Rightarrow \{\text{node} : e_i,$$
$$\text{score} : TF\text{-}IDF(\text{“michelle gellar”}, \text{“sarah michelle gellar”})\}$$

This is performed for every n-gram ($n \in \{1, 2, 3, \infty\}$) of every entity node in the KB. Assuming there is an entity node, say e_j, for the actress "Sarah Jessica Parker", we would end up creating a second mapping from unigram "sarah":

$$I_{\text{entity}}(\text{“sarah”}) \Rightarrow \{\text{node} : e_j,$$
$$\text{score} : TF\text{-}IDF(\text{“sarah”}, \text{“sarah jessica parker”})\}$$

In other words, "sarah" would be linked to both e_i and e_j, with corresponding $TF\text{-}IDF$ weights.

Once the index I_{entity} is built, we can link *entityText* from the structured query (e.g., "michelle gellar") to the intended entity in the KB (e.g., e_i). Starting with $n = \infty$, we iterate over n-grams of *entityText* and query I_{entity}, which returns all matching entities in the KB with associated $TF\text{-}IDF$ relevance scores. For each n-gram, retrieved entities are appended to the candidate set C. We continue this process with decreasing value of n (i.e., $n \in \{\infty, 3, 2, 1\}$)

Early termination happens if C is non-empty and n is less than or equal to the number of tokens in *entityText*. The latter criterion is to avoid cases where we find an exact match but there are also partial matches that might be more relevant: For "jurassic park", for $n = \infty$, we get an exact match to the original movie "Jurassic Park". But we would also like to retrieve "Jurassic Park II" as a candidate entity, which is only possible if we keep processing until $n = 2$.

(c) Answer Selection. Once we have a list of candidate entities C, we use each candidate node e_{cand} as a starting point to reach candidate answers.

A *graph reachability index* I_{reach} is built for mapping each entity node e to all nodes e' that are reachable, with the associated path $p(e, e')$. For the purpose of the current approach, we limit our search to a single hop away, but this index can be easily expanded to support a wider search.

$$I_{reach}(e_i) \Rightarrow$$
$$\{\text{node:}e_{i_1}, \text{text:The Grudge}, \text{path:}[\texttt{actedIn}]\}$$
$$\{\text{node:}e_{i_2}, \text{text:4/14/1977}, \text{path:}[\texttt{bornOn}]\}$$
$$\{\text{node:}e_{i_3}, \text{text:F. Prinze}, \text{path:}[\texttt{marriedTo}]\}$$

We use I_{reach} to retrieve all nodes e' that are reachable from e_{cand}, where the path from is consistent with the predicted relation r (i.e., $r \in p(e_{\text{cand}}, e')$). These are added to the candidate answer set A. For example, in the example above, node e_{i_2} would have been added to the answer set A, since the path [`bornOn`] matches the predicted relation in the structured query. After repeating this process for each entity in C, the highest-scored node in A is our best answer to the question.

4 Experimental Setup

Data. Evaluation of RNN-QA was carried out on SimpleQuestions, which uses a subset of Freebase containing 17.8M million facts, 4M unique entities, and 7523 relation types. Indexes I_{entity} and I_{reach} are built based on this knowledge base.

SimpleQuestions was built by (Bordes et al., 2014) to serve as a larger and more diverse factoid QA dataset.[1] Freebase facts are sampled in a way to ensure a diverse set of questions, then given to human annotators to create questions from, and get labeled with corresponding entity and relation type. There are a total of 1837 unique relation types that appear in SimpleQuestions.

Training. We fixed the embedding layer based on the pre-trained 300-dimensional Google News embedding,[2] since the data size is too small for training embeddings. Out-of-vocabulary words were assigned to a random vector (sampled from uniform distribution). Parameters were learned via stochastic gradient descent, using categorical cross-entropy as objective. In order to handle variable-length input, we limit the input to N tokens and prepend a special pad word if input has fewer.[3] We tried a variety of configurations for the RNN: four choices for the type of RNN layer (GRU or LSTM, bidirectional or not); depth from 1 to 3; and drop-out ratio from 0 to 0.5, yielding a total of 48 possible configurations. For each possible setting, we trained the model on the training portion and used the validation portion to avoid over-fitting. After running all 48 experiments, the most optimal setting was selected by micro-averaged F-score of predicted entities (entity detection) or accuracy (relation prediction) on the validation set. We concluded that the optimal model is a 2-layer bidirectional LSTM (BiLSTM2) for entity detection and BiGRU2 for relation prediction. Drop-out was 10% in both cases.

5 Results

End-to-End QA. For evaluation, we apply the relation prediction and entity detection models on each test question, yielding a structured query $q = \{entityText\colon t_e, relation\colon r\}$ (Section 3a). Entity linking gives us a list of candidate entity nodes (Section 3b). For each candidate entity e_{cand}, we can limit our relation choices to the set of unique relation types that some candidate entity e_{cand} is associated with. This helps eliminate the artificial ambiguity due to overlapping relation types as well as the spurious ambiguity due to redundancies in a typical knowledge base. Even though there are 1837 relation types in Freebase, the number of relation types that we need to consider per question (on average) drops to 36. The highest-scored answer node is selected by finding the highest scored entity node e that has an outward edge of type r (Section 3c). We follow Bordes et al. (2015) in comparing the predicted entity-relation pair to the ground truth. A question is counted as correct if and only if the entity we select (i.e., e) and the relation we predict (i.e, r) match the ground truth.

Table 1 summarizes end-to-end experimental results. We use the best models based on validation set accuracy and compare it to three prior approaches: a specialized network architecture that explicitly memorizes facts (Bordes et al., 2015), a network that learns how to convolve sequence of characters in the question (Golub and He, 2016), and a complex network with attention mechanisms to learn most important parts of the question (Yin et al., 2016). Our approach outperforms the state of the art in accuracy (i.e., precision at top 1) by 11.9 points (15.6% relative).

Model	P@1
Memory Network (2015)	63.9
Char-level CNN (2016)	70.9
Attentive max-pooling (2016)	76.4
RNN-QA (best models)	**88.3**
naive ED	58.9
naive RP	4.1
naive ED and RP	3.7

Table 1: Top-1 accuracy on test portion of SimpleQuestions. Ablation study on last three rows.

Last three rows quantify the impact of each component via an ***ablation study***, in which we replace either entity detection (ED) or relation prediction (RP) models with a naive baseline: (i) we assign the relation that appears most frequently in training data (i.e., `bornOn`), and/or (ii) we tag the entire question as an entity (and then perform the n-gram entity linking). Results confirm that RP is absolutely critical, since both datasets include a diverse and well-balanced set of relation types. When we applied the naive ED baseline, our results drop significantly, but they are still comparable to prior results. Given that most prior work do not use the network to detect entities, we can

[1] 75910/10845/21687 question-answer pairs for training/validation/test is an order of magnitude larger than comparable datasets. Vocabulary size is 55K as opposed to around 3K for WebQuestions (Berant et al., 2013).

[2] `word2vec.googlecode.com`

[3] Input length (N) was set to 36, the maximum number of tokens across training and validation splits.

deduce that our RNN-based entity detection is the reason our approach performs so well.

Error Analysis. In order to better understand the weaknesses of our approach, we performed a *blame analysis*: Among 2537 errors in the test set, 15% can be blamed on entity detection — the relation type was correctly predicted, but the detected entity did not match the ground truth. The reverse is true for 48% cases.[4] We manually labeled a sample of 50 instances from each blame scenario. When entity detection is to blame, 20% was due to spelling inconsistencies between question and KB, which can be resolved with better text normalization during indexing (e.g., "la kings" refers to "Los Angeles Kings"). We found 16% of the detected entities to be correct, even though it was not the same as the ground truth (e.g., either "New York" or "New York City" is correct in "what can do in new york?"); 18% are inherently ambiguous and need clarification (e.g., "where bin laden got killed?" might mean "Osama" or "Salem"). When blame is on relation prediction, we found that the predicted relation is reasonable (albeit different than ground truth) 29% of the time (e.g., "what was nikola tesla known for" can be classified as `profession` or `notable_for`).

RNN-QA in Practice. In addition to matching the state of the art in effectiveness, we also claimed that our simple architecture provides an efficient and modular solution. We demonstrate this by applying our model (without any modifications) to the entertainment domain and deploying it to the Comcast X1 platform serving millions of customers every day. Training data was generated synthetically based on an internal entertainment KB. For evaluation, 295 unique question-answer pairs were randomly sampled from real usage logs of the platform.

We can draw two important conclusions from Table 2: First of all, we find that almost all of the user-generated natural-language questions (278/295∼95%) are first-order questions, supporting the significance of first-order QA as a task. Second, we show that even if we simply use an open-sourced deep learning toolkit (`keras.io`) for implementation and limit the computational resources to 2 CPU cores per thread, RNN-QA answers 75% of questions correctly with very reasonable latency.

Error	Count
Correct	220
Incorrect entity	16
Incorrect relation	42
Not first-order question	17
Total Latency	76±16 ms

Table 2: Evaluation of RNN-QA on real questions from X platform.

6 Conclusions and Future work

We described a simple yet effective approach for QA, focusing primarily on first-order factual questions. Although we understand the benefit of exploring task-agnostic approaches that aim to capture semantics in a more general way (e.g., (Kumar et al., 2015)), it is also important to acknowledge that there is no "one-size-fits-all" solution as of yet.

One of the main novelties of our work is to decompose the task into two subproblems, entity detection and relation prediction, and provide solutions for each in the form of a RNN. In both cases, we have found that bidirectional networks are beneficial, and that two layers are sufficiently deep to balance the model's ability to fit versus its ability to generalize.

While an ablation study revealed the importance of both entity detection and relation prediction, we are hoping to further study the degree of which improvements in either component affect QA accuracy. Drop-out was tuned to 10% based on validation accuracy. While we have not implemented attention directly on our model, we can compare our results side by side on the same benchmark task against prior work with complex attention mechanisms (e.g., (Yin et al., 2016)). Given the proven strength of attention mechanisms, we were surprised to find our simple approach to be clearly superior on SimpleQuestions.

Even though deep learning has opened the potential for more generic solutions, we believe that taking advantage of problem structure yields a more accurate and efficient solution. While first-order QA might seem like a solved problem, there is clearly still room for improvement. By revealing that 95% of real use cases fit into this paradigm, we hope to convince the reader that this is a valuable problem that requires more *attention*.

[4] In remaining 37% incorrect answers, both models fail, so the blame is shared.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL.

Matthew W Bilotti, Jonathan Elsas, Jaime Carbonell, and Eric Nyberg. 2010. Rank Learning for Factoid Question Answering with Linguistic and Semantic Constraints. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 459–468, New York, NY, USA. ACM.

Matthew W Bilotti, Paul Ogilvie, Jamie Callan, and Eric Nyberg. 2007. Structured Retrieval for Question Answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 351–358, New York, NY, USA. ACM.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075.

Davide Buscaldi, Paolo Rosso, José Manuel Gómez-Soriano, and Emilio Sanchis. 2010. Answering Questions with an N-gram Based Passage Retrieval Engine. *J. Intell. Inf. Syst.*, 34(2):113–134.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 260–269. The Association for Computer Linguistics.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

David Golub and Xiaodong He. 2016. Character-level question answering with attention. *arXiv preprint arXiv:1604.00727*.

Alex Graves. 2008. *Supervised sequence labelling with recurrent neural networks*. Ph.D. thesis, Technical University Munich.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 172–175. Association for Computational Linguistics.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1576–1586. The Association for Computational Linguistics.

Wei-Ning Hsu, Yu Zhang, and James R. Glass. 2016. Recurrent neural network encoder with attention for community question answering. *CoRR*, abs/1603.07044.

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

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285.

Zefu Lu, Lei Li, and Wei Xu. 2015. Twisted recurrent network for named entity recognition. In *Bay Area Machine Learning Symposium*.

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

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2786–2792. AAAI Press.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1913–1916. ACM.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *TACL*, 2:377–392.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *arXiv preprint arXiv:1510.06168*.

Liu Yang, Qingyao Ai, Jiafeng Guo, and W Bruce Croft. 2016. anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 287–296. ACM.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *EMNLP*, pages 645–650.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. *arXiv preprint arXiv:1606.03391*.

Dell Zhang and Wee Sun Lee. 2003. Question Classification Using Support Vector Machines. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 26–32, New York, NY, USA. ACM.

The strange geometry of skip-gram with negative sampling

David Mimno and **Laure Thompson**
Cornell University
mimno@cornell.edu, laurejt@cs.cornell.edu

Abstract

Despite their ubiquity, word embeddings trained with skip-gram negative sampling (SGNS) remain poorly understood. We find that vector positions are not simply determined by semantic similarity, but rather occupy a narrow cone, diametrically opposed to the context vectors. We show that this geometric concentration depends on the ratio of positive to negative examples, and that it is neither theoretically nor empirically inherent in related embedding algorithms.

1 Introduction

It is generally assumed that the geometry of word embeddings is determined by semantic relatedness. Vectors are assumed to be distributed throughout a K-dimensional space, with specific regions devoted to specific concepts. We find that vectors trained with the skip-gram with negative sampling (SGNS) algorithm (Mikolov et al., 2013) are not only influenced by semantics but are also strongly influenced by the negative sampling objective. In fact, far from spanning the possible space, they exist only in a narrow cone in $\mathbb{R}^K$. Nevertheless, SGNS vectors have become a foundational tool in NLP and perform as well or better than numerous methods with similar objectives (Turian et al., 2010; Dhillon et al., 2012; Pennington et al., 2014; Luo et al., 2015) with respect to evaluations of intrinsic and extrinsic quality (Schnabel et al., 2015).

SGNS works by training two sets of embeddings: the “official” word embeddings and a second set of *context* embeddings, with one K-dimensional vector in each set for each word in the vocabulary. The objective tries to make the word vector and context vector closer for a pair of words that actually occur together than for randomly sampled “negative” words. Following training, the word vectors are typically saved; the context vectors are deleted. Any difference between these two sets of vectors is puzzling, since the sliding window used in training is symmetrical: a word and its context word reverse roles almost immediately. Indeed, the superficially similar GloVe algorithm (Pennington et al., 2014) also defines word and context vectors and by default returns the mean of these two vectors.

Previous work has analyzed what the algorithm might be doing in theory, as an approximation to a matrix factorization (Levy and Goldberg, 2014). Other work has considered the empirical effects of some of the more arbitrary-seeming algorithmic choices (Levy et al., 2015). But we still have relatively little understanding of how the algorithm actually determines parameter values.

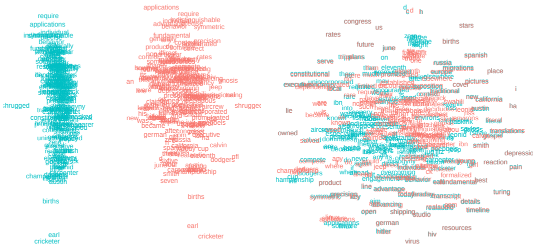


Figure 1: SGNS word vectors and their context vectors projected using PCA (left) and t-SNE (right). t-SNE provides a more readable layout, but masks the divergence between word and context vectors.

In this work we measure geometric properties of SGNS-trained word vectors and their context vectors. Although the word vectors appear to span K-dimensional space, we find that the SGNS objective results in vectors that are narrowly clustered in a single orthant, and can be made non-negative without significant loss. Figure 1 shows two visualizations of SGNS vectors and context vectors. The context vectors mirror the “official” word vectors, with the angle between vectors effectively controlled by the number of negative samples. We

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2873–2878
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

show that this effect is due to negative sampling and not the general embedding objective. We note that this relationship between vectors is effectively hidden by the commonly-used t-SNE projection (van der Maaten and Hinton, 2008).

2 Word embeddings with SGNS

The SGNS algorithm defines two sets of parameters, K-dimensional word vectors $\boldsymbol{w}_i$ and context vectors $\boldsymbol{c}_i$ for each word i. We define a weight between a word i and a context word j as $\sigma_{ij} = \frac{\exp(w_i^T c_j)}{1+\exp(w_i^T c_j)}$. For each observed pair i, j we sample S "negative" context words from a modified unigram distribution $p(w)^{0.75}$. The stochastic gradient update for one parameter w_{ik} is then

$$\frac{d\ell}{dw_{ik}} = (1 - \sigma_{ij})c_{jk} + \sum_{s=1}^{S} -\sigma_{is}c_{sk}, \qquad (1)$$

suppressing for clarity a learning rate parameter λ. A symmetrical update is performed for the context word parameters $\boldsymbol{c}_j$ and $\boldsymbol{c}_s$, substituting $\boldsymbol{w}_i$ for $\boldsymbol{c}$. This update has been shown to be equivalent to the gradient of a factorization of a pointwise mutual information matrix (Levy and Goldberg, 2014).

The impact of the update is to push the vector $\boldsymbol{w}_i$ closer to the context vector of the observed context word $\boldsymbol{c}_j$ and away from the context vectors of the negatively sampled words. The amount of change at any given update is dependent on the degree to which the current model predicts the "correct" source of the context word, whether from the real data distribution or the negative sampling distribution. If the model is infinitely certain that the real word is real ($\sigma_{ij} = 1.0$) and the fake words are fake ($\sigma_{is} = 0.0 \; \forall s$), it will make no change to the current parameters.

3 Results

We first present a series of empirical observations based on vectors trained from a corpus of Wikipedia articles that is commonly distributed with word embedding implementations.[1] We then evaluate the sensitivity of these properties to different algorithmic parameters. We make no assertion that these are optimal (or even particularly good) vectors, only that they are representative of the properties of the algorithm.

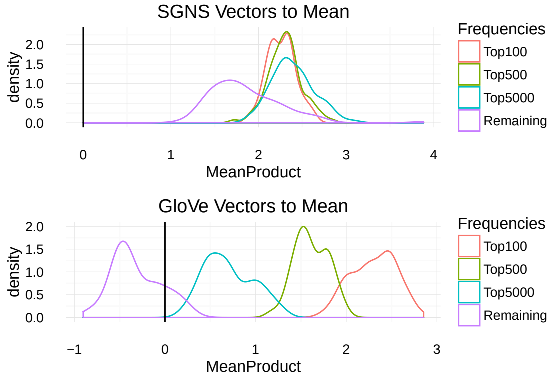


Figure 2: SGNS-trained vectors mostly point in the same direction, towards a mean vector $\hat{\boldsymbol{w}}$.

To determine whether observed properties are due to SGNS specifically or to embeddings in general, we compare SGNS-trained vectors to vectors trained by the GloVe algorithm (Pennington et al., 2014). The choice of GloVe as a comparison is due to its popularity and superficial similarity to SGNS.[2] We begin by examining one set of embeddings from each algorithm, both with $K = 50$ dimensions, a vocabulary of $\approx$ 70k words, and context window 5. We then evaluate sensitivity to negative samples, window size, and dimension.

Embeddings are sensitive to word frequency (Hellrich and Hahn, 2016). Following Zipf's law, words in natural language tend to sort into ranges of frequent words (the majority of tokens) and rare words (the majority of types), with a large class of intermediate-frequency terms in the middle. As a result, the large majority of interactions are between frequent terms or between frequent and infrequent terms. Interactions between infrequent terms are rare, no matter how large the corpus. We define four categories of words by ranked frequency: the top 100 words (ultra-high frequency), the 100–500th ranked words (high frequency), the 500–5000th ranked words (moderate frequency) and the remaining (low frequency) words.

SGNS vectors are arranged along a primary axis. Our first observation is that SGNS-trained vectors all point in roughly the same direction. We can define a mean vector $\bar{w}$ by averaging the vectors of the complete vocabulary $\boldsymbol{w}$. We sample a balanced set of 400 total words with 100 each from the four frequency categories. Figure 2 shows the

[1] http://mattmahoney.net/dc/text8.zip

[2] We make no attempt in this work to compare the *quality* of SGNS and GloVe vectors, nor should the omission of other algorithms be attributed to anything but space constraints.

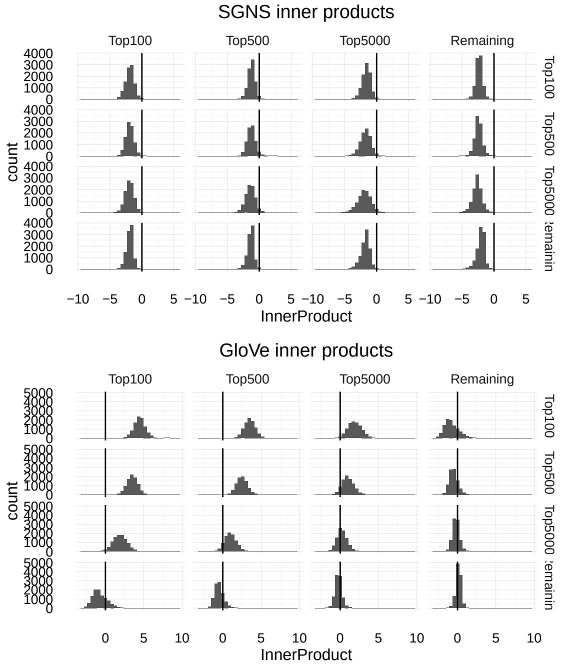


Figure 3: Almost all combinations of words have negative inner products for SGNS, unlike GloVe.

distribution of inner products between these 400 sampled words and their mean vector $\hat{\boldsymbol{w}}$. All vectors have a large, positive inner product with the mean, indicating that they are not evenly dispersed through the space. Furthermore, the frequency category of words has relatively little effect on the inner product, with the exception of the rare words, which have slightly less positive inner products. As a comparison, the vectors trained by GloVe show a clear relationship with word frequency, with low-frequency words opposing the frequency-balanced mean vector.

This result does not depend on a specific mean vector. Using the *global* mean vector rather than the frequency-balanced mean vector reverses the order of frequency categories within each plot, but does not change their overall shape. SGNS vector inner products are all positive, with low-frequency words the most positive. GloVe inner products become positive for low-frequency words and negative for high-frequency words.

The inner product between vectors is used by the algorithm during training, but in practice vectors are often normalized to have unit length before use. It is possible that the apparent pattern shown in Figure 2 may be an artifact of differing average lengths between words of different frequencies. After normalizing SGNS vectors to length 1.0, the lowest and highest frequency words are most similar to the mean vector, with the moderate-frequency words showing the greatest deviation. Normalization does not change the relative order for GloVe vectors.

SGNS vectors point away from context vectors. It is possible that vectors could have a positive inner product with the mean vector but be mutually orthogonal. Figure 3 shows the distribution of inner products $\boldsymbol{w}_i^T \boldsymbol{c}_j$ for pairs of words divided by frequency for SGNS and GloVe. Almost all interactions have similar, negative inner products for SGNS, while GloVe interactions are sensitive to frequency and vary more widely. We note that the high-frequency words in GloVe appear to form a cohesive cluster between themselves (positive inner products) that points away from the lower frequency words (negative inner products), while infrequent words are more dispersed and have no clear pattern relative to each other.

SGNS vectors are mostly non-negative. Not only do SGNS vectors occupy a narrow region of embedding space, it appears that the vectors can be rotated to fall mostly within the positive orthant. For each column of the matrix of vectors $\boldsymbol{w}$ we can compute the dimension-wise mean $\bar{w}_k$. Multiplying $\boldsymbol{w}$ by a diagonal matrix of the signs of the means $diag(sign(\bar{w}_k))$ flips each dimension so that its mean is positive. Figure 4 shows the resulting positive-mean histogram for 12 of the 50 dimensions trained by SGNS (the remaining dimensions are similar). Some dimensions have medians close to 0.0, but most skew positive.

Indeed, it is possible to simply drop all remaining negative values without radically changing the properties of the vectors. Embeddings are often evaluated based on word similarity prediction (Schnabel et al., 2015). Using only positive entries, Spearman rank correlation drops from 0.283 to .276 on the SIMLEX word similarity task and from 0.556 to 0.542 on the MEN task. Subtracting the global mean vector has similarly little impact, reducing SIMLEX correlation to 0.271 and increasing MEN correlation to 0.575. This property may help explain why sparse (Faruqui et al., 2015) and non-negative (Luo et al., 2015) embeddings do not lose significant performance.

SGNS context vectors point away from the word vectors. What then is the geometry of the *context* vectors $\boldsymbol{c}$? The two sets of vectors appear to present a noisy mirror image of each other. Figure 5 shows the distribution of inner products between

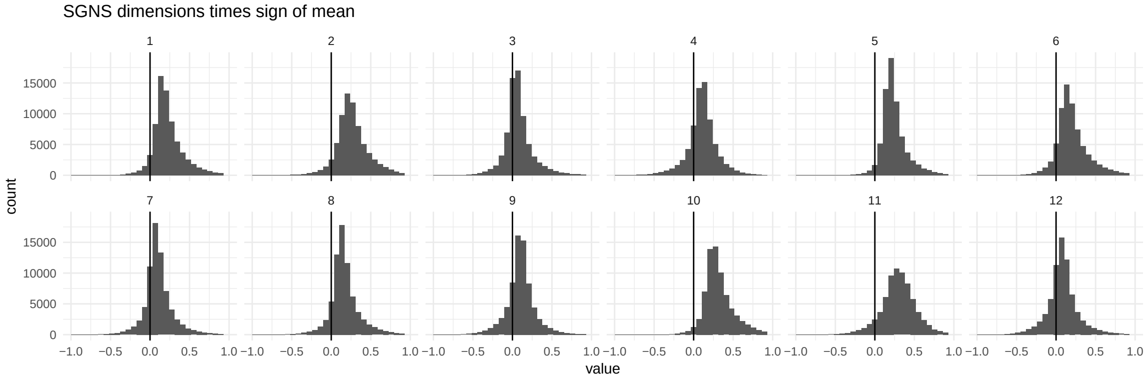


Figure 4: Most latent dimensions show significant skew. Each panel shows a histogram of values for one of the first 12 latent dimensions, after multiplication by the sign of the mean for that dimension.

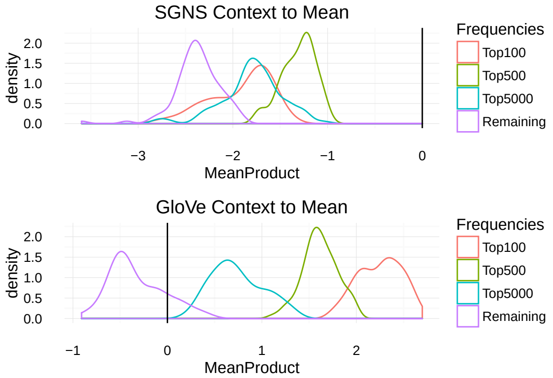


Figure 5: SGNS context vectors point *away* from the mean vector $\hat{\boldsymbol{w}}$, GloVe context vectors do not.

the context vectors and the same mean vector $\hat{\boldsymbol{w}}$ used in Figure 2. These inner products are negative, indicating that the context vectors point in the opposite direction from the word vectors. In contrast, the GloVe context vectors have essentially the same relationship to the mean of the word vectors as the word vectors themselves. This property explains why it is common to output the mean of $\boldsymbol{w}_i$ and $\boldsymbol{c}_i$ for each word for GloVe but not for SGNS: in Glove these two vectors are essentially noisy copies of one another, while in SGNS the two vectors are pointing almost in the opposite direction.

Positive and negative weights come to equilibrium. Eq. 1 balances two terms, a positive interaction term $1.0 - \sigma_{ij}$ between a word and a context word and negative interaction terms $0.0 - \sigma_{is}$ between a word and one of S randomly sampled words. These terms can be viewed as a "label" minus an expectation, as in the gradient for logistic regression. Since there is no $1/S$ term to balance the number of random samples, one might expect that the "power" of the sampled context terms might overwhelm the true interaction term. In practice, these samples appear to find an equilibrium that effectively balances out the number of random samples after a short burn-in phase. We recorded a moving average of positive and negative weights for an ultra-frequent word (*the*) and a moderately frequent word (*tuesday*). In both cases, the mean of the values for positive samples starts at 0.5 and for the negative samples at -0.5. The positive values converge toward $S = 5$ times the mean of the values for negative samples: 0.581 vs. -0.182 for *the* and 0.693 vs. -0.138 for *tuesday*.

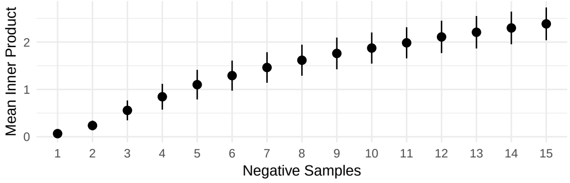


Figure 6: The number of negative samples affects the inner product between vectors and the mean vector. Results are indistinguishable across 10 initializations for each value.

The negative objective is optimized when each model vector points away from the context vectors. The positive objective, in contrast, is maximized when word and context vectors for related words are pointing in the same direction. The negative force acts to repel the vectors, the positive force acts to pull them together.

During the crucial early phases of the algorithm, negative samples have more weight than positive samples: when inner products are near zero, both types of samples will have values of σ_{ij} and σ_{is} close to 0.5, so negative samples will "count" S times more than positive. The early phases of the algorithm will focus on pushing the two sets of vectors apart into separate regions of the latent space. Once vectors and context vectors separate, inner products will become negative, so σ_{ij} and σ_{is} will move closer to 0.0.

The balance between positive and negative samples consistently affects the geometry of the vectors, and is not sensitive to random initialization. We varied the number of negative samples from $S = 1$ to $S = 15$, and ran 10 trials for each value with different random initializations. As shown in Figure 6, as we increase S, the average inner product between vectors and the mean vector within each model increases.

SGNS vectors are concentrated and point away from their context vectors, and changing the number of negative samples appears to affect this property. We now consider whether other factors could also cause this behavior.

Effect of window size Both SGNS and GloVe operate over word co-occurrences within a sliding window centered around each token in the corpus. This window size parameter has an effect on the semantics of vectors, so it is important to consider whether it has an effect on the geometry of vectors. Simply setting an equal window size for SGNS and GloVe does not, however, guarantee that the two algorithms are seeing equivalent data, because each pair is weighted linearly by token distance in SGNS and by $1/distance$ in GloVe. Figure 7 shows average inner products for each frequency with the global mean vector for 10 trials each at window size 5, 10, 15, 20 with $K = 50$. Increasing window size leads to greater divergence between high- and low-frequency words for word and context vectors, but does not change their pattern. GloVe results are similarly unchanged.

Effect of vector size As with window size, the dimensionality K of the word vectors can affect their ability to represent semantic relationships. Figure 8 shows an increase in inner product with the global mean as we increase K (10 trials each, window size 15), but the effect is small relative to that of the number of negative samples S. GloVe inner products change by less than 0.05.

4 Conclusion

SGNS vectors encode semantic relatedness, but their arrangement is much more strongly influenced by the negative sampling objective than is usually assumed. We find that vectors lie on a narrow primary axis that is effectively non-negative. Users should not interpret relationships between vectors without recognizing this geometric context.

In this work we have deliberately restricted ourselves to describing the geometric properties of vectors. We see several areas for further work. First, there are likely to be theoretical reasons why the observed concentration of SGNS vectors in a narrow cone does not appear to affect performance relative to algorithms that do not have this property. Second, measuring the interplay between positive and negative objectives may provide insight into algorithmic choices that are now poorly understood, such as the effect of reducing the occurrence of frequent words in the corpus and the sampling distribution of negative examples. Finally, we suggest that in addition to theoretical analysis, more work should be done to understand the actual working of algorithms on real data.

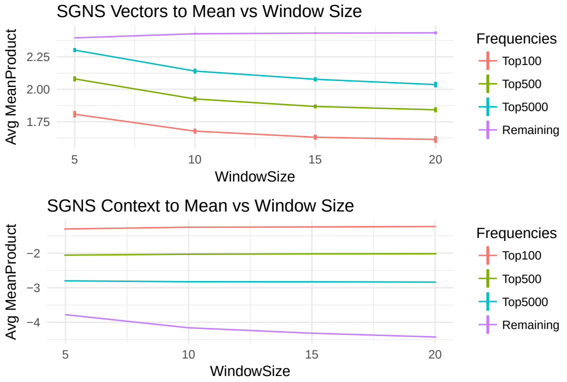


Figure 7: SGNS word and context vectors face in opposite directions regardless of window size.

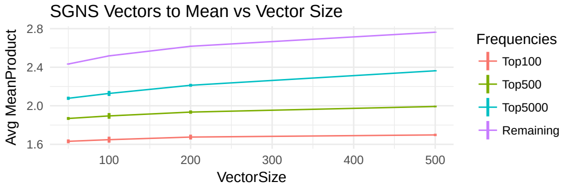


Figure 8: As vector size increases SGNS vectors shift toward the mean vector $\bar{\boldsymbol{w}}$. (GloVe vectors change by < 0.05.)

Acknowledgements

Anonymous reviewers helped make this paper much better. This work was supported by NSF #1526155, #1652536, and #DGE-1144153; and the Alfred P. Sloan Foundation.

References

Paramveer S. Dhillon, Jordan Rodu, Dean P. Foster, and Lyle H. Ungar. 2012. Two step CCA: A new spectral method for estimating vector models of words. In *ICML*, pages 1551–1558.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. In *ACL*.

Johannes Hellrich and Udo Hahn. 2016. Bad company—neighborhoods in neural embedding spaces considered harmful. In *COLING*.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

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

Hongyin Luo, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2015. Online learning of interpretable word embeddings. In *EMNLP*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing high-dimensional data using t-SNE. *JMLR*, 9:2579–2605.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic Regularities in Continuous Space Word Representations. *HLT-NAACL*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

Tobias Schnabel, Igor Labutov, David M. Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *EMNLP*, pages 298–307.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *ACL*, pages 384–394.

Natural Language Processing with Small Feed-Forward Networks

Jan A. Botha Emily Pitler Ji Ma Anton Bakalov
Alex Salcianu David Weiss Ryan McDonald Slav Petrov
Google Inc.
Mountain View, CA
{jabot,epitler,maji,abakalov,salcianu,djweiss,ryanmcd,slav}@google.com

Abstract

We show that small and shallow feed-forward neural networks can achieve near state-of-the-art results on a range of unstructured and structured language processing tasks while being considerably cheaper in memory and computational requirements than deep recurrent models. Motivated by resource-constrained environments like mobile phones, we showcase simple techniques for obtaining such small neural network models, and investigate different tradeoffs when deciding how to allocate a small memory budget.

1 Introduction

Deep and recurrent neural networks with large network capacity have become increasingly accurate for challenging language processing tasks. For example, machine translation models have been able to attain impressive accuracies, with models that use hundreds of millions (Bahdanau et al., 2014; Wu et al., 2016) or billions (Shazeer et al., 2017) of parameters. These models, however, may not be feasible in all computational settings. In particular, models running on mobile devices are often constrained in terms of memory and computation.

Long Short-Term Memory (LSTM) models (Hochreiter and Schmidhuber, 1997) have achieved good results with small memory footprints by using character-based input representations: e.g., the part-of-speech tagging models of Gillick et al. (2016) have only roughly 900,000 parameters. Latency, however, can still be an issue with LSTMs, due to the large number of matrix multiplications they require (eight per LSTM cell): Kim and Rush (2016) report speeds of only 8.8 words/second when running a two-layer LSTM translation system on an Android phone.

Feed-forward neural networks have the potential to be much faster. In this paper, we show that small feed-forward networks can achieve results at or near the state-of-the-art on a variety of natural language processing tasks, with an order of magnitude speedup over an LSTM-based approach.

We begin by introducing the network model structure and the character-based representations we use throughout all tasks (§2). The four tasks that we address are: language identification (Lang-ID), part-of-speech (POS) tagging, word segmentation, and preordering for translation. In order to use feed-forward networks for structured prediction tasks, we use transition systems (Titov and Henderson, 2007, 2010) with feature embeddings as proposed by Chen and Manning (2014), and introduce two novel transition systems for the last two tasks. We focus on *budgeted* models and ablate four techniques (one on each task) for improving accuracy for a given memory budget:

1. Quantization: Using more dimensions and less precision (Lang-ID: §3.1).
2. Word clusters: Reducing the network size to allow for word clusters and derived features (POS tagging: §3.2).
3. Selected features: Adding explicit feature conjunctions (segmentation: §3.3).
4. Pipelines: Introducing another task in a pipeline and allocating parameters to the auxiliary task instead (preordering: §3.4).

We achieve results at or near state-of-the-art with small ($<$ 3 MB) models on all four tasks.

2 Small Feed-Forward Network Models

The network architectures are designed to limit the memory and runtime of the model. Figure 1 illustrates the model architecture:

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2879–2885
Copenhagen, Denmark, September 7–11, 2017.

1. Discrete features are organized into groups (e.g., $\mathbf{E}_{bigrams}$), with one embedding matrix $\mathbf{E}_g \in \mathbb{R}^{V_g \times D_g}$ per group.
2. Embeddings of features extracted for each group are reshaped into a single vector and concatenated to define the output of the embedding layer as $\mathbf{h}_0 = [\mathbf{X}_g \mathbf{E}_g \mid \forall g]$.
3. A single hidden layer, $\mathbf{h}_1$, with M rectified linear units (Nair and Hinton, 2010) is fully connected to $\mathbf{h}_0$.
4. A softmax function models the probability of an output class y: $P(y) \propto \exp(\beta_y^T \mathbf{h}_1 + b_y)$, where $\beta_y \in \mathbb{R}^M$ and b_y are the weight vector and bias, respectively.

Memory needs are dominated by the embedding matrix sizes ($\sum_g V_g D_g$, where V_g and D_g are the vocabulary sizes and dimensions respectively for each feature group g), while runtime is strongly influenced by the hidden layer dimensions.

Hashed Character *n*-grams Previous applications of this network structure used (pretrained) word embeddings to represent words (Chen and Manning, 2014; Weiss et al., 2015). However, for word embeddings to be effective, they usually need to cover large vocabularies (100,000+) and dimensions (50+). Inspired by the success of character-based representations (Ling et al., 2015), we use features defined over character *n*-grams instead of relying on word embeddings, and learn their embeddings from scratch.

We use a distinct feature group g for each n-gram length N, and control the size V_g directly by applying *random feature mixing* (Ganchev and Dredze, 2008). That is, we define the feature value v for an n-gram string x as $v = \mathcal{H}(x) \mod V_g$, where $\mathcal{H}$ is a well-behaved hash function. Typical values for V_g are in the 100-5000 range, which is far smaller than the exponential number of unique raw n-grams. A consequence of these small feature vocabularies is that we can also use small feature embeddings, typically D_g=16.

Quantization A commonly used strategy for compressing neural networks is quantization, using less precision to store parameters (Han et al., 2015). We compress the embedding weights (the vast majority of the parameters for these shallow models) by storing scale factors for each embedding (details in the supplementary material). In §3.1, we contrast devoting model size to higher precision and lower dimensionality versus lower precision and more network dimensions.

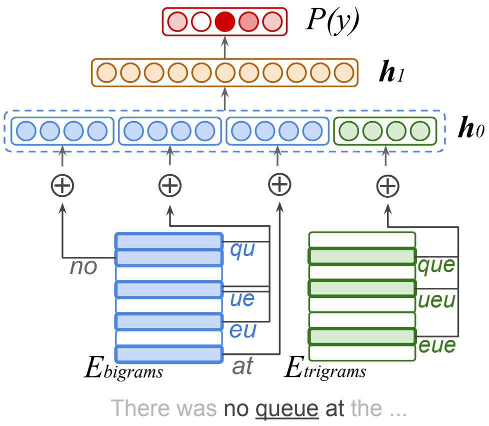


Figure 1: An example network structure for a model using bigrams of the previous, current and next word, and trigrams of the current word. Does not illustrate hashing.

Training Our objective function combines the cross-entropy loss for model predictions relative to the ground truth with L2 regularization of the biases and hidden layer weights. For optimization, we use mini-batched averaged stochastic gradient descent with momentum (Bottou, 2010; Hinton, 2012) and exponentially decaying learning rates. The mini-batch size is fixed to 32 and we perform a grid search for the other hyperparameters, tuning against the task-specific evaluation metric on held-out data, with early stopping. Full feature templates and optimal hyperparameter settings are given in the supplementary material.

3 Experiments

We experiment with small feed-forward networks for four diverse NLP tasks: language identification, part-of-speech tagging, word segmentation, and preordering for statistical machine translation.

Evaluation Metrics In addition to standard task-specific quality metrics, our evaluations also consider model size and computational cost. We skirt implementation details by calculating size as the number of kilobytes (1KB=1024 bytes) needed to represent all model parameters and resources. We approximate the computational cost as the number of floating-point operations (FLOPs) performed for one forward pass through the network given an embedding vector $\mathbf{h}_0$. This cost is dominated by the matrix multiplications to compute (unscaled) activation unit values, hence our metric excludes the non-linearities and softmax normalization, but still accounts for the final layer logits. To ground this metric, we also provide indicative absolute speeds for each task, as measured on a modern workstation CPU (3.50GHz Intel Xeon E5-1650 v3).

3.1 Language Identification

Recent shared tasks on code-switching (Molina et al., 2016) and dialects (Malmasi et al., 2016) have generated renewed interest in language identification. We restrict our focus to single language identification across diverse languages, and compare to the work of Baldwin and Lui (2010) on predicting the language of Wikipedia text in 66 languages. For this task, we obtain the input $\mathbf{h}_0$ by separately *averaging* the embeddings for each n-gram length ($N = [1, 4]$), as summation did not produce good results.

Table 1 shows that we outperform the low-memory nearest-prototype model of Baldwin and Lui (2010). Their nearest neighbor model is the most accurate but its memory scales linearly with the size of the training data.

Moreover, we can apply quantization to the embedding matrix without hurting prediction accuracy: it is better to use less precision for each dimension, but to use more dimensions. Our subsequent models all use quantization. There is no noticeable variation in processing speed when performing dequantization on-the-fly at inference time. Our 16-dim Lang-ID model runs at 4450 documents/second (5.6 MB of text per second) on the preprocessed Wikipedia dataset.

Relationship to Compact Language Detector These techniques back the open-source Compact Language Detector v3 (CLD3)[1] that runs in Google Chrome browsers.[2] Our experimental Lang-ID model uses the same overall architecture as CLD3, but uses a simpler feature set, less involved preprocessing, and covers fewer languages.

3.2 POS Tagging

We apply our model as an unstructured classifier to predict a POS tag for each token independently, and compare its performance to that of the byte-to-span (BTS) model (Gillick et al., 2016). BTS is a 4-layer LSTM network that maps a sequence of bytes to a sequence of labeled spans, such as tokens and their POS tags. Both approaches limit model size by using small input vocabularies: byte values in the case of BTS, and hashed character n-grams and (optionally) cluster ids in our case.

Model	Micro F1	Size
Baldwin and Lui (2010): NN	90.2	-
Baldwin and Lui (2010): NP	87.0	-
Small FF, 6 dim	87.3	334 KB
Small FF, 16 dim	88.0	800 KB
Small FF, 16 dim, *quantized*	88.0	302 KB

Table 1: Language Identification. Quantization allows trading numerical precision for larger embeddings. The two models from Baldwin and Lui (2010) are the nearest neighbor (NN) and nearest prototype (NP) approaches.

Model	Acc.	Wts.	MB	Ops.
Gillick et al. (2016)	95.06	900k	-	6.63m
Small FF	94.76	241k	0.6	0.27m
+Clusters	95.56	261k	1.0	0.31m
$\frac{1}{2}$ Dim.	95.39	143k	0.7	0.18m

Table 2: POS tagging. Embedded word clusters improves accuracy and allows the use of smaller embedding dimensions.

Bloom Mapped Word Clusters It is well known that word clusters can be powerful features in linear models for a variety of tasks (Koo et al., 2008; Turian et al., 2010). Here, we show that they can also be useful in neural network models. However, naively introducing word cluster features drastically increases the amount of memory required, as a word-to-cluster mapping file with hundreds of thousands of entries can be several megabytes on its own.[3] By representing word clusters with a Bloom map (Talbot and Talbot, 2008), a key-value based generalization of Bloom filters, we can reduce the space required by a factor of ∼15 and use 300KB to (approximately) represent the clusters for 250,000 word types.

In order to compare against the monolingual setting of Gillick et al. (2016), we train models for the same set of 13 languages from the Universal Dependency treebanks v1.1 (Nivre et al., 2016) corpus, using the standard predefined splits.

As shown in Table 2, our best models are 0.3% more accuate on average across all languages than the BTS monolingual models, while using 6x fewer parameters and 36x fewer FLOPs. The cluster features play an important role, providing a 15% relative reduction in error over our vanilla model, but also increase the overall size. Halv-

[1] github.com/google/cld3

[2] As of the date of this writing in 2017.

[3] For example, the commonly used English clusters from the BLLIP corpus is over 7 MB – people.csail.mit.edu/maestro/papers/bllip-clusters.gz

Transition	
SPLIT	$([\sigma],[i\|\beta]) \rightarrow ([\sigma\|i],[\beta])$
MERGE	$([\sigma],[i\|\beta]) \rightarrow ([\sigma],[\beta])$

Table 3: Segmentation Transition system. Initially all characters are on the buffer β and the stack σ is empty: $([],[c_1c_2...c_n])$. In the final state the buffer is empty and the stack contains the first character for each word.

ing all feature embedding dimensions (except for the cluster features) still gives a 12% reduction in error and trims the overall size back to 1.1x the vanilla model, staying well under 1MB in total. This halved model configuration has a throughput of 46k tokens/second, on average.

Two potential advantages of BTS are that it does not require tokenized input and has a more accurate multilingual version, achieving 95.85% accuracy. From a *memory* perspective, one multilingual BTS model will take less space than separate FF models. However, from a *runtime* perspective, a pipeline of our models doing language identification, word segmentation, and then POS tagging would still be faster than a single instance of the deep LSTM BTS model, by about 12x in our FLOPs estimate.[4]

3.3 Segmentation

Word segmentation is critical for processing Asian languages where words are not explicitly separated by spaces. Recently, neural networks have significantly improved segmentation accuracy (Zhang et al., 2016; Cai and Zhao, 2016; Liu et al., 2016; Yang et al., 2017; Kong et al., 2015). We use a structured model based on the transition system in Table 3, and similar to the one proposed by Zhang and Clark (2007). We conduct the segmentation experiments on the Chinese Treebank 6.0 with the recommended data splits. No external resources or pretrained embeddings are used. Hashing was detrimental to quality in our preliminary experiments, hence we do not use it for this task. To learn an embedding for unknown characters, we cast characters occurring only once in the training set to a special symbol.

Selected Features Because we are not using hashing here, we need to be careful about the size of the input vocabulary. The neural network with its non-linearity is in theory able to learn bigrams by conjoining unigrams, but it has been shown that explicitly using character bigram features leads to better accuracy (Zhang et al., 2016; Pei et al., 2014). Zhang et al. (2016) suggests that embedding manually specified feature conjunctions further improves accuracy ('Zhang et al. (2016)-combo' in Table 4). However, such embeddings could easily lead to a model size explosion and thus are not considered in this work.

Model	Accuracy	Size
Zhang et al. (2016)	95.01	–
Zhang et al. (2016)-combo	95.95	–
Small FF, 64 dim	94.24	846KB
Small FF, 256 dim	94.16	3.2MB
Small FF, 64 dim, *bigrams*	95.18	2.0MB

Table 4: Segmentation results. Explicit bigrams are useful.

The results in Table 4 show that spending our memory budget on small bigram embeddings is more effective than on larger character embeddings, in terms of both accuracy and model size. Our model featuring bigrams runs at 110KB of text per second, or 39k tokens/second.

3.4 Preordering

Preordering source-side words into the target-side word order is a useful preprocessing task for statistical machine translation (Xia and McCord, 2004; Collins et al., 2005; Nakagawa, 2015; de Gispert et al., 2015). We propose a novel transition system for this task (Table 5), so that we can repeatedly apply a small network to produce these permutations. Inspired by a non-projective parsing transition system (Nivre, 2009), the system uses a SWAP action to permute spans. The system is sound for permutations: any derivation will end with all of the input words in a permuted order, and complete: all permutations are reachable (use SHIFT and SWAP operations to perform a bubble sort, then APPEND $n-1$ times to form a single span). For training and evaluation, we use the English-Japanese manual word alignments from Nakagawa (2015).

Transition	Precondition
APPEND	$([\sigma\|i\|j],[\beta]) \rightarrow ([\sigma\|[ij]],[\beta])$
SHIFT	$([\sigma],[i\|\beta]) \rightarrow ([\sigma\|i],[\beta])$
SWAP	$([\sigma\|i\|j],[\beta]) \rightarrow [\sigma\|j],[i\|\beta]); i < j$

Table 5: Preordering Transition system. Initially all words are part of singleton spans on the buffer: $([],[[w_1][w_2]...[w_n]])$. In the final state the buffer is empty and the stack contains a single span.

[4] Our calculation of BTS FLOPs is very conservative and favorable to BTS, as detailed in the supplementary material.

Model	FRS	Size
Nakagawa (2015)	81.6	-
Small FF	75.2	0.5MB
Small FF + POS tags	81.3	1.3MB
Small FF + Tagger input fts.	76.6	3.7MB

Table 6: Preordering results for English → Japanese. *FRS* (in $[0, 100]$) is the fuzzy reordering score (Talbot et al., 2011).

Pipelines For preordering, we experiment with either spending all of our memory budget on reordering, or spending some of the memory budget on features over predicted POS tags, which also requires an additional neural network to predict these tags. Full feature templates are in the supplementary material. As the POS tagger network uses features based on a three word window around the token, another possibility is to add all of the features that would have affected the POS tag of a token to the reorderer directly.

Table 6 shows results with or without using the predicted POS tags in the preorderer, as well as including the features used by the tagger in the reorderer directly and only training the downstream task. The preorderer that includes a separate network for POS tagging and then extracts features over the predicted tags is more accurate and smaller than the model that includes all the features that contribute to a POS tag in the reorderer directly. This pipeline processes 7k tokens/second when taking pretokenized text as input, with the POS tagger accounting for 23% of the computation time.

4 Conclusions

This paper shows that small feed-forward networks are sufficient to achieve useful accuracies on a variety of tasks. In resource-constrained environments, speed and memory are important metrics to optimize as well as accuracies. While large and deep recurrent models are likely to be the most accurate whenever they can be afforded, feed-foward networks can provide better value in terms of runtime and memory, and should be considered a strong baseline.

Acknowledgments

We thank Kuzman Ganchev, Fernando Pereira, and the anonymous reviewers for their useful comments.

References

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

Timothy Baldwin and Marco Lui. 2010. Language identification: The long and the short of the matter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 229–237. Association for Computational Linguistics.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for Chinese. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 409–420, Berlin, Germany. Association for Computational Linguistics.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 740–750.

M. Collins, P. Koehn, and I. Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*, pages 531–540.

Kuzman Ganchev and Mark Dredze. 2008. Small statistical models by random feature mixing. In *Proceedings of the ACL-08- HLT Workshop on Mobile Language Processing*, pages 18–19. Association for Computational Linguistics.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of NAACL-HLT*, pages 1296–1306, San Diego, USA. Association for Computational Linguistics.

Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. 2015. Fast and accurate preordering for SMT using neural networks. In *Proceedings of NAACL*, pages 1012–1017.

Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Geoffrey E. Hinton. 2012. A practical guide to training restricted Boltzmann machines. In *Neural Networks: Tricks of the Trade (2nd ed.)*, Lecture Notes in Computer Science, pages 599–619. Springer.

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

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2015. Segmental recurrent neural networks. *CoRR*, abs/1511.06018.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530. Association for Computational Linguistics.

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, and Ting Liu. 2016. Exploring segment representations for neural segmentation models. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2880–2886.

Shervin Malmasi, Marcos Zampieri, Nikola Ljubešić, Preslav Nakov, Ahmed Ali, and Jörg Tiedemann. 2016. Discriminating between similar languages and Arabic dialect identification: A report on the third DSL shared task. In *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*, pages 1–14, Osaka, Japan. The COLING 2016 Organizing Committee.

Giovanni Molina, Fahad AlGhamdi, Mahmoud Ghoneim, Abdelati Hawwari, Nicolas Rey-Villamizar, Mona Diab, and Thamar Solorio. 2016. Overview for the second shared task on language identification in code-switched data. In *Proceedings of the Second Workshop on Computational Approaches to Code Switching*, pages 40–49, Austin, Texas. Association for Computational Linguistics.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814.

Tetsuji Nakagawa. 2015. Efficient top-down BTG parsing for machine translation preordering. In *Proceedings of ACL*, pages 208–218.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for Chinese word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 293–303, Baltimore, Maryland. Association for Computational Linguistics.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

David Talbot, Hideto Kazawa, Hiroshi Ichikawa, Jason Katz-Brown, Masakazu Seno, and Franz J. Och. 2011. A lightweight evaluation framework for machine translation reordering. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 12–21, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Talbot and John Talbot. 2008. Bloom maps. In *Proceedings of the Meeting on Analytic Algorithmics and Combinatorics*, pages 203–212. Society for Industrial and Applied Mathematics.

Ivan Titov and James Henderson. 2007. Fast and robust multilingual dependency parsing with a generative latent variable model. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 947–951.

Ivan Titov and James Henderson. 2010. A latent variable model for generative dependency parsing. In Harry Bunt, Paola Merlo, and Joakim Nivre, editors, *Trends in Parsing Technology: Dependency Parsing, Domain Adaptation, and Deep Parsing*, pages 35–55. Springer Netherlands, Dordrecht.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word Representations: A Simple and General Method for Semi-supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 323–333.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of COLING*, page 508.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. *CoRR*, abs/1704.08960.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Transition-based neural word segmentation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 421–431, Berlin, Germany. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 840–847, Prague, Czech Republic. Association for Computational Linguistics.

Deep Multi-Task Learning for Aspect Term Extraction with Memory Interaction[*]

Xin Li and **Wai Lam**
Key Laboratory on High Confidence Software Technologies (Sub-Lab, CUHK),
Ministry of Education, and
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong, Hong Kong
{lixin, wlam}@se.cuhk.edu.hk

Abstract

We propose a novel LSTM-based deep multi-task learning framework for aspect term extraction from user review sentences. Two LSTMs equipped with extended memories and neural memory operations are designed for jointly handling the extraction tasks of aspects and opinions via memory interactions. Sentimental sentence constraint is also added for more accurate prediction via another LSTM. Experiment results over two benchmark datasets demonstrate the effectiveness of our framework.

1 Introduction

The aspect-based sentiment analysis (ABSA) task is to identify opinions expressed towards specific entities such as *laptop* or attributes of entities such as *price* (Liu, 2012a). This task involves three subtasks: Aspect Term Extraction (ATE), Aspect Polarity Detection and Aspect Category Detection. As a fundamental subtask in ABSA, the goal of the ATE task is to identify opinionated aspect expressions. One of most important characteristics is that opinion words can provide indicative clues for aspect detection since opinion words should co-occur with aspect words. Most publicly available datasets contain the gold standard annotations for opinionated aspects, but the ground truth of the corresponding opinion words is not commonly provided. Some works tackling the ATE task ignore the consideration of opinion words and just focus on aspect term modeling and learning (Jin et al., 2009; Jakob and Gurevych, 2010; Toh and Wang, 2014; Chernyshevich, 2014; Manek et al., 2017; San Vicente et al., 2015; Liu et al., 2015; Poria et al., 2016; Toh and Su, 2016; Yin et al., 2016). They fail to leverage opinion information which is supposed to be useful clues.

Some works tackling the ATE task consider opinion information (Hu and Liu, 2004a,b; Popescu and Etzioni, 2005; Zhuang et al., 2006; Qiu et al., 2011; Liu et al., 2012b, 2013a,b, 2014) in an unsupervised or partially supervised manner. Qiu et al. (2011) proposed Double Propagation (DP) to collectively extract aspect terms and opinion words based on information propagation over a dependency graph. One drawback is that it heavily relies on the dependency parser, which is prone to generate mistakes when applying on informal online reviews. Liu et al. (2014) modeled relation between aspects and opinions by constructing a bipartite heterogenous graph. It cannot perform well without a high-quality phrase chunker and POS tagger reducing its flexibility. As unsupervised or partially supervised frameworks cannot take the full advantages of aspect annotations commonly found in the training data, the above methods lead to deficiency in leveraging the data. Recently, Wang et al. (2016) considered relation between opinion words and aspect words in a supervised model named RNCRF. However, RNCRF tends to suffer from parsing errors since the structure of the recursive network hinges on the dependency parse tree. CMLA (Wang et al., 2017a) used a multilayer neural model where each layer consists of aspect attention and opinion attention. However CMLA merely employs standard GRU without extended memories.

We propose MIN (Memory Interaction Network), a novel LSTM-based deep multi-task learning framework for the ATE task. Two LSTMs with extended memory are designed for handling

[*]The work described in this paper is substantially supported by a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14203414). We thank Lidong Bing and Piji Li for their helpful comments on this draft and the anonymous reviewers for their valuable feedback.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2886–2892
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

the extraction tasks of aspects and opinions. The aspect-opinion relationship is established based on neural memory interactions between aspect extraction and opinion extraction where the global indicator score of opinion terms and local positional relevance between aspects and opinions are considered. To ensure that aspects are from sentimental sentences, MIN employs a third LSTM for sentimental sentence classification facilitating more accurate aspect term extraction. Experiment results over two benchmark datasets show that our framework achieves superior performance.

2 Model

2.1 Overview

Let an input review sentence with T word tokens and the corresponding distributed representations be w $= \{w_1, ..., w_T\}$ and x $= \{x_1, ..., x_T\}$ respectively. The ATE task is treated as a sequence labeling task with BIO tagging scheme and the set of aspect tags for the word w_t is $y_t^A \in \{B, I, O\}$, where B, I, O represent beginning of, inside and outside of the aspect span respectively. Commonly found training data contains gold annotations for aspect terms and opinionated sentences, but the gold standard of opinion words are usually not available.

In our multi-task learning framework, three tasks are involved: (1) aspect term extraction (ATE), (2) opinion word extraction and (3) sentimental sentence classification. We design a task-specific LSTM, namely, A-LSTM, O-LSTM and S-LSTM, for tackling each of the above tasks respectively. The first component of our proposed framework consists of A-LSTM and O-LSTM where we equip LSTMs with extended operational memories and some operations are defined over the memories for task-level memory interactions. The second component is to determine if a review sentence is sentimental. This is achieved by employing a vanilla LSTM, namely, S-LSTM.

2.2 Model Description

The first component of our framework MIN is composed of A-LSTM and O-LSTM. Both LSTMs have extended memories for task-level memory interactions. A-LSTM involves a large aspect memory $H_t^A \in \mathbb{R}^{n_m \times dim_h^A}$ and an opinion summary vector $m_t^O \in \mathbb{R}^{dim_h^O}$ where H_t^A contains n_m pieces of aspect hidden states of dimension dim_h^A and m_t^O is distilled from H_t^O. As for O-LSTM, similarly, an opinion memory $H_t^O \in \mathbb{R}^{n_m \times dim_h^O}$ and an aspect-specific summary vector $m_t^A \in \mathbb{R}^{dim_h^A}$ are included.

We use the aspect term annotations in the training data for training A-LSTM. As there is no ground truth available for opinion words in the training data, sentiment lexicon and high-precision dependency rules are introduced to find potential opinion words. Commonly used opinion words can be found in some general sentiment lexicons. To find opinion words, not in sentiment lexicons, in a sentence, we build a small rule set $\mathcal{R}$ composed of dependency relations with high confidence, e.g., $amod$, $nsubj$, and determine if w_t directly depends on the gold aspect word through the dependencies in $\mathcal{R}$. If so, w_t will be treated as a potential opinion word. Then such opinion words are used as training data for O-LSTM.

In the memory-enhanced A-LSTM and O-LSTM, we manually design three kinds of operations: (1) **READ** to select n_m pieces of aspect (opinion) hidden states from the past memories and build H_t^A (H_t^O); (2) **DIGEST** to distill an aspect (opinion)-specific summary m_t^A (m_t^O) from H_t^A (H_t^O) where influences of opinion terms and relative positions of inputs are considered; (3) **INTERACT** to perform interaction between A-LSTM and O-LSTM using the task specific summaries (i.e., m_t^A and m_t^O).

Consider the work flow of A-LSTM for aspect term extraction. Since opinion words and aspect terms should co-occur, the goal of A-LSTM participating in memory interactions is to acquire opinion summaries from O-LSTM (i.e., m_t^O) for better aspect prediction. First of all, MIN will **READ** n_m pieces of opinion memories which are most related to w_t from O-LSTM. Syntax structure could be used but syntactic parsers are not effective for processing short informal review sentences. Therefore, MIN selects memory segments temporally related to w_t. Precisely, the opinion memory at the time step t is $H_t^O = [h_{t-1}^O; ...; h_{t-n_m}^O]$ where h_{t-i}^O is the $(t-i)$-th hidden state from O-LSTM. Since the linear context contains most of the parent nodes and the child nodes of w_t on the dependency parse tree, treating the corresponding memory segments as relevant segments to w_t is reasonable.

Then MIN will **DIGEST** the collected opinion memories H_t^O in the A-LSTM. As different memory segments are not of equal importance for the current decision and the same segment in different memories (i.e., different H_t^O) also makes a difference, MIN leverages two kind of weights to summarize the collected content. The first weight is the indicator score of being opinion terms denoted as $v^I \in \mathbb{R}^{n_m}$, which is used to measure how much opinion information the word w_{t-i} ($i = 1, .., n_m$) holds. We adopt Euclidean distance between distributed representations of w_{t-i} and opinion words. It is obvious that computing the distance between x_{t-i} and each opinion word is expensive. Thus, we run an off-the-shelf clustering algorithm over opinion words in the training set and then use the produced n_c centroids to estimate the indicator score v_i^I of w_{t-i} being an opinion word:

$$v_i^I = \sum_{j=1}^{n_c} \frac{1}{||x_{t-i} - c_j||_2} \tag{1}$$

where x_{t-i} is the distributed representation of w_{t-i} and c_j is the centroid vector representation of j-th cluster. This weighting scheme ensures that w_{t-i} is assigned a high score as long as x_{t-i} is close to a particular centroid. The aspect decision of w_t is also affected by relative position between w_{t-i} and w_t. Thus, MIN employs the second weight v^P to explicitly model their positional relevance and the initial weight for the i-th segment v_i^P is calculated as below:

$$v_i^P = \frac{n_m - i + 1}{\sum_{k=1}^{n_m} k} \tag{2}$$

where n_m is the number of hidden state in H_t^O. This position-aware weight enables that the closer the word w_{t-i} is to the current input, the more the corresponding memory segment will contribute to the current decision. To better capture the local positional relevance, we make the initialized v^P as learnable parameters. Combining the above two weights helps to utilize each active memory segment according to the importance for prediction and m_t^O, the summary of H_t^O is generated:

$$m_t^O = (H_t^O)^\top \left(\frac{v^I \odot v^P}{||v^I||_2}\right) \tag{3}$$

where $\odot$ denotes element-wise multiplication and $|| * ||_2$ is Euclidean norm of vectors. From Equation 3, m_t^O is dominated by the associated memory segment of w_{t-i} that obtains the high combined weights.

In the last operation **INTERACT**, A-LSTM communicates with O-LSTM by acquiring m_t^O from O-LSTM and incorporating the summary into the memory update. The update process is as follows:

$$\begin{aligned}
i_t^A &= \sigma(W_i^A x_t + U_i^A [H_t^A[1] : m_t^O]) + b_i^A) \\
f_t^A &= \sigma(W_f^A x_t + U_f^A [H_t^A[1] : m_t^O]) + b_f^A) \\
\hat{c}_t^A &= \tanh(W_c^A x_t + U_c^A [H_t^A[1] : m_t^O]) + b_c^A) \\
o_t^A &= \sigma(W_o^A x_t + U_o^A [H_t^A[1] : m_t^O]) + b_o^A) \\
c_t^A &= i_t^A \odot \hat{c}_t^A + c_{t-1}^A \odot f_t^A \\
h_t^A &= \tanh(c_t^A) \odot o_t^A
\end{aligned} \tag{4}$$

where W_*^A, U_*^A and b_*^A are weight parameters of the A-LSTM and σ is the *sigmoid* activation function. [:] denotes vector concatenation operation. m_t^O can be seen as the summary of the opinion indicator in the left context of w_t and $H_t^A[1]$ is the most immediate hidden memory of A-LSTM. MIN blends the opinion summary from O-LSTM with the memory from A-LSTM. The co-occurrence relation between aspects and opinion words is modeled by such "memory fusion" strategy. Since opinion words can appear on both sides of w_t, memory segments corresponding to the right context (i.e., "future" memory) should be included. Hence, we conduct bi-directional training for A-LSTM.

The work flow of memory interaction and the update process of the internal memories in O-LSTM are kept same with those in A-LSTM except the **DIGEST** operation. Specifically, we set m_t^A, the task-specific summary of A-LSTM, as h_t^A.

The second component of MIN is a generic LSTM called S-LSTM for discriminating sentimental sentences and non-sentimental sentences. The design and the process of the memory update in this component are similar to that in Jozefowicz et al. (2015). In sentences not conveying any sentimental meanings, some words like *food*, *service* tend to be misclassified as aspect terms since they are commonly used in user reviews. To avoid this kind of error, we add a constraint that an aspect term should come from sentimental sentence. Specifically, S-LSTM learns the sentimental representation h_T^S of the sentence and then feeds it in aspect prediction as a soft constraint:

$$P(y_t^A | x_t) = \text{softmax}(W_{fc}^A([h_t^A : h_T^S])) \tag{5}$$

where W_{fc}^A denotes the weight matrix of the fully-connected softmax layer.

On the whole, our proposed MIN framework has three LSTMs and each of them is differentiable. Thus, our MIN framework can be efficiently trained with gradient descent. For A-LSTM and O-LSTM, we use the token-level cross-entropy error between the predicted distribution $P(y_t^{\mathcal{T}}|x_t)$ and the gold standard distribution $P(y_t^{\mathcal{T},g}|x_t)$ as the loss function ($\mathcal{T} \in \{A, O\}$):

$$Loss(\mathcal{T}) = -\frac{1}{N*T}\sum_{i=1}^{N}\sum_{t=1}^{T} P(Y_{i,t}^{\mathcal{T},g}|X_{i,t}) \odot \log[P(Y_{i,t}^{\mathcal{T}}|X_{i,t})] \tag{6}$$

For S-LSTM, sentence-level cross entropy error are employed to calculate the corresponding loss:

$$Loss(S) = -\frac{1}{N}\sum_{i=1}^{N} P(Y_i^{S,g}|X_i) \odot \log[P(Y_i^{S}|X_i)] \tag{7}$$

Then, losses from different LSTMs are combined to form the training objective of the MIN framework:

$$J(\theta) = Loss(A) + Loss(O) + Loss(S) \tag{8}$$

.

	#TRAIN/#TEST Sentences	#TRAIN/#TEST Aspects
D_1	3045/800	2358/654
D_2	2000/676	1743/622

Table 1: Statistics of datasets.

3 Experiment

3.1 Dataset

We conduct experiments on two benchmark datasets from SemEval ABSA challenge (Pontiki et al., 2014, 2016) as shown in Table 1. D_1 (SemEval 2014) contains reviews from the laptop domain and D_2 (SemEval 2016) contains reviews from the restaurant domain. In these datasets, aspect terms have been labeled and sentences containing at least one golden truth aspect are regarded as sentimental sentences. As gold standard annotations for opinion words are not provided, we select words with strong subjectivity from MPQA[1] as potential opinion words. Apart from the common opinion words in the sentiment lexicon, we also treat words, which directly depend on gold standard aspect terms through high-precision dependency rules, as opinion words.

3.2 Experiment Design

To evaluate the proposed MIN framework, we perform comparison with the following two groups of methods:

(1) *CRF* based methods:

- **CRF**: Conditional Random Fields with basic feature templates[2] and word embeddings.
- **Semi-CRF**: First-order semi-Markov conditional random fields (Sarawagi et al., 2004) and the feature template in Cuong et al. (2014) is adopted.
- **IHS_RD** (Chernyshevich, 2014), **NLANGP** (Toh and Su, 2016): Best systems in ATE subtask in SemEval ABSA challenges (Pontiki et al., 2014, 2016).
- **DLIREC** (Toh and Wang, 2014), **AUEB** (Xenos et al., 2016): Top-ranked CRF-based systems in ATE subtask in SemEval ABSA challenges (Pontiki et al., 2014, 2016).
- **WDEmb** (Yin et al., 2016): Enhanced CRF with word embeddings, linear context embeddings and dependency path embeddings.

(2) *Neural Network* based methods

- **LSTM**: Vanilla bi-directional LSTM with pre-trained word embeddings[3].
- **RNCRF** (Wang et al., 2016): Dependency Tree based Recursive Neural Network with CRF extractor[4].

For datasets in the restaurant domain, we train word embeddings of dimension 200 with word2vec (Mikolov et al., 2013) on Yelp reviews[5]. For those in laptop domain, we use pre-trained $glove.840B.300d$[6].

[1] http://mpqa.cs.pitt.edu/

[2] http://sklearn-crfsuite.readthedocs.io/en/latest/

[3] As we use our own implementation of LSTM, the reported results are different from those in (Liu et al., 2015)

[4] Specifically, we list the result of RNCRF over D_1 without opinion annotations for fair comparison. As no result is provided for RNCRF-no-opinion over D_2, we report the corresponding performance of the full model. See their following works (Wang et al., 2017a,b). Also, CMLA (Wang et al., 2017a) reports better results than RNCRF but we do not compare with it. The reason is that CMLA introduces the gold standard opinion labels in the training data while such labels are not available for our experiments

[5] https://www.yelp.com/dataset_challenge

[6] https://nlp.stanford.edu/projects/glove/

	D_1	D_2
CRF	74.01%	69.56%
Semi-CRF	68.75%	66.35%
IHS_RD	74.55%	-
DLIREC	73.78%	-
NLANGP	-	72.34%
AUEB	-	70.44%
WDEmb	75.16%	-
LSTM	75.25%	71.26%
RNCRF	77.26%	69.74%
Our Work	**77.58%**	**73.44%**

Table 2: Experiment results

The hyper-parameters are selected via ten-fold cross validation. The dimension of hidden representations are 100, 20, 40 for A-LSTM, O-LSTM and S-LSTM respectively. The dropout rate for O-LSTM and S-LSTM is 0.4. The size of the aspect (opinion) memory n_m is 4. The batch size is set to 32. As for initialization of network parameters, we adopt the strategy that the initial weights are sampled from the uniform distribution (Glorot and Bengio, 2010). We employ ADAM (Kingma and Ba, 2014) as optimizer and the default settings of ADAM are used.

To better reveal the capability of the proposed MIN, we train 5 models with the same group of hyper-parameters and report the average F_1 score over the testing set.

3.3 Results and Analysis

Table 2 depicts experiment results. Compared to the best systems in SemEval challenge, MIN achieves 3.0% and 1.1% absolute gains on D_1 and D_2 respectively. Besides, our MIN outperforms WDEmb, a strong CRF-based system benefiting from several kinds of useful word embeddings, by 2.1% on D_1. With memory interactions and consideration of sentimental sentence, our MIN boosts the performance of vanilla bi-directional LSTM (+2.0% and +1.7% respectively). It validates the effectiveness of the manually designed memory operations and the proposed memory interaction mechanism. MIN also outperforms the state-of-the-art RNCRF on each dataset suggesting that memory interactions can be an alternative strategy instead of syntactic parsing. To further study the impact of each element in MIN, we conduct ablation experiments. As shown in Table 3, removing bi-directionality decreases the extraction performances (-2.0% and -1.0%). The soft sentimental constraint proves to be useful since MIN is 1.5% and 1.0% superior than the framework without S-LSTM on D_1 and D_2 respectively. O-LSTM brings in the largest performance gains on D_2 compared with ablated framework (i.e., MIN without O-LSTM), verifying our postulation that aspect-opinion "interaction" is more effective than only considering aspect terms. We also observe that the contribution of O-LSTM is less significant than that of bi-directionality on D_1 (+1.6% vs +2.0%). This is reasonable since using opinion words as adjective modifiers placed after the aspects is common in English.

	D_1	D_2
MIN without bi-directionality	75.59%	71.87%
MIN without S-LSTM	76.04%	72.55%
MIN without O-LSTM	75.97%	71.80%
MIN	**77.58%**	**73.44%**

Table 3: Ablation experiment results.

4 Conclusions

We propose Memory Interaction Network (MIN), a multi-task learning framework, to detect aspect terms from the online user reviews. Compared with previous studies, our MIN has following features:

- Co-occurrence pattern between aspects and opinions is captured via memory interactions, where the neural memory operations are designed to summarize task-level information and perform interactions.
- A novel LSTM unit with extended memories is developed for memory interactions.

References

Maryna Chernyshevich. 2014. Ihs r&d belarus: Cross-domain extraction of product features using crf. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 309–313.

Nguyen Viet Cuong, Nan Ye, Wee Sun Lee, and Hai Leong Chieu. 2014. Conditional random field with high-order dependencies for sequence labeling and segmentation. *Journal of Machine Learning Research*, 15(1):981–1009.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*, pages 249–256.

Minqing Hu and Bing Liu. 2004a. Mining and summarizing customer reviews. In *Proceedings of KDD*, pages 168–177.

Minqing Hu and Bing Liu. 2004b. Mining opinion features in customer reviews. In *Proceedings of AAAI*, pages 755–760.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single and cross-domain setting with conditional random fields. In *Proceedings of EMNLP*, pages 1035–1045.

Wei Jin, Hung Hay Ho, and Rohini K Srihari. 2009. A novel lexicalized hmm-based learning framework for web opinion mining. In *Proceedings of ICML*, pages 465–472.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of ICML*, pages 2342–2350.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Bing Liu. 2012a. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Kang Liu, Heng Li Xu, Yang Liu, and Jun Zhao. 2013a. Opinion target extraction using partially-supervised word alignment model. In *Proceedings of IJCAI*, pages 2134–2140.

Kang Liu, Liheng Xu, and Jun Zhao. 2012b. Opinion target extraction using word-based translation model. In *Proceedings of EMNLP/CoNLL*, pages 1346–1356.

Kang Liu, Liheng Xu, and Jun Zhao. 2013b. Syntactic patterns versus word alignment: Extracting opinion targets from online reviews. In *Proceedings of ACL*, pages 1754–1763. Association for Computational Linguistics.

Kang Liu, Liheng Xu, and Jun Zhao. 2014. Extracting opinion targets and opinion words from online reviews with graph co-ranking. In *Proceedings of ACL*, pages 314–324. Association for Computational Linguistics.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of EMNLP*, pages 1433–1443.

Asha S Manek, P Deepa Shenoy, M Chandra Mohan, and KR Venugopal. 2017. Aspect term extraction for sentiment analysis in large movie reviews using gini index feature selection method and svm classifier. *World Wide Web Journal*, 20(2):135–154.

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

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphee De Clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of EMNLP*, pages 339–346.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2016. Aspect extraction for opinion mining with a deep convolutional neural network. *Knowledge-Based Systems*, 108:42–49.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27.

Iñaki San Vicente, Xabier Saralegi, and Rodrigo Agerri. 2015. Elixa: A modular and flexible absa platform. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 748–752.

Sunita Sarawagi, William W Cohen, et al. 2004. Semi-markov conditional random fields for information extraction. In *Proceedings of NIPS*, pages 1185–1192.

Zhiqiang Toh and Jian Su. 2016. Nlangp at semeval-2016 task 5: Improving aspect based sentiment analysis using neural network features. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 282–288.

Zhiqiang Toh and Wenting Wang. 2014. Dlirec: Aspect term extraction and term polarity classification system. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 235–240.

Wenya Wang, Sinno Jialin Pan, and Daniel Dahlmeier. 2017b. Multi-task coupled attentions for category-specific aspect and opinion terms co-extraction. *arXiv preprint arXiv:1702.01776*.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016. Recursive neural conditional random fields for aspect-based sentiment analysis. In *Proceedings of EMNLP*, pages 616–626. Association for Computational Linguistics.

Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017a. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. In *Proceedings of AAAI*, pages 3316–3322.

Dionysios Xenos, Panagiotis Theodorakakos, John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2016. Aueb-absa at semeval-2016 task 5: Ensembles of classifiers and embeddings for aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 312–317.

Yichun Yin, Furu Wei, Li Dong, Kaimeng Xu, Ming Zhang, and Ming Zhou. 2016. Unsupervised word and dependency path embeddings for aspect term extraction. In *Proceedings of IJCAI*, pages 2979–2985.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proceedings of CIKM*, pages 43–50.

Analogs of Linguistic Structure in Deep Representations

Jacob Andreas and **Dan Klein**
Computer Science Division
University of California, Berkeley
`jda,klein@cs.berkeley.edu`

Abstract

We investigate the compositional structure of message vectors computed by a deep network trained on a communication game. By comparing truth-conditional representations of encoder-produced message vectors to human-produced referring expressions, we are able to identify aligned (vector, utterance) pairs with the same meaning. We then search for structured relationships among these aligned pairs to discover simple vector space transformations corresponding to negation, conjunction, and disjunction. Our results suggest that neural representations are capable of spontaneously developing a "syntax" with functional analogues to qualitative properties of natural language.[1]

1 Introduction

The past year has seen a renewal of interest in end-to-end learning of communication strategies between pairs of agents represented with deep networks (Wagner et al., 2003). Approaches of this kind make it possible to learn decentralized policies from scratch (Foerster et al., 2016; Sukhbaatar et al., 2016), with multiple agents coordinating via learned communication protocol. More generally, any encoder–decoder model (Sutskever et al., 2014) can be viewed as implementing an analogous communication protocol, with the input encoding playing the role of a message in an artificial "language" shared by the encoder and decoder (Yu et al., 2016). Earlier work has found that under suitable conditions, these protocols acquire simple interpretable lexical (Dircks and Stoness, 1999; Lazaridou et al., 2016) and sequential structure (Mordatch and Abbeel, 2017), even without natural language training data.

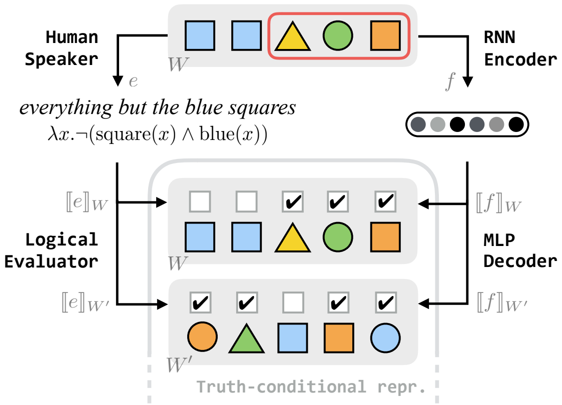


Figure 1: Overview of our task. Given a dataset of referring expression games, example human expressions, and their associated logical forms, we compute explicit denotations both for the original task and in other possible tasks—giving rise to a truth-conditional representation of the natural language. We train a recurrent encoder–decoder model to solve the same tasks directly, and use the decoder to generate comparable truth-conditional representations of neural encodings.

One of the distinguishing features of natural language is compositionality: the existence of operations like negation and coordination that can be applied to utterances with predictable effects on meaning. RNN models trained for natural language processing tasks have been found to learn representations that encode some of this compositional structure—for example, sentence representations for machine translation encode explicit features for certain syntactic phenomena (Shi et al., 2016) and represent some semantic relationships translationally (Levy et al., 2014). It is thus natural to ask whether these "language-like" structures also arise spontaneously in models trained directly from an environment signal. Rather than using language as a form of supervision, we propose to use it as a *probe*—exploiting post-hoc statistical correspondences between natural language descriptions and neural encodings to discover regular structure in representation space.

To do this, we need to find (vector, string) pairs with matching semantics, which requires first aligning unpaired examples of human–human

[1] Code and data are available at `http://github.com/jacobandreas/rnn-syn`.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2893–2897
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

communication with network hidden states. This is similar to the problem of "translating" RNN representations recently investigated in Andreas et al. (2017). Here we build on that approach in order to perform a detailed analysis of *compositional* structure in learned "languages". We investigate a communication game previously studied by FitzGerald et al. (2013), and make two discoveries: in a model trained without any access to language data,

1. The strategies employed by human speakers in a given communicative context are surprisingly good predictors of RNN behavior in the same context: humans and RNNs send messages whose interpretations agree on nearly 90% of object-level decisions, even outside the contexts in which they were produced.

2. Interpretable language-like structure naturally arises in the space of representations. We identify geometric regularities corresponding to negation, conjunction, and disjunction, and show that it is possible to linearly transform representations in ways that approximately correspond to these logical operations.

2 Task

We focus our evaluation on a communication game due to FitzGerald et al. (2013) (Figure 1, top). In this game, the *speaker* observes (1) a *world* W of 1–20 objects labeled with with attributes and (2) a designated *target* subset X of objects in the world. The *listener* observes only W, and the speaker's goal is to communicate a representation of X that enables the listener to accurately reconstruct it. The GENX dataset collected for this purpose contains 4170 human-generated natural-language referring expressions and corresponding logical forms for 273 instances of this game. Because these human-generated expressions have all been pre-annotated, we treat language and logic interchangeably and refer to both with the symbol e. We write $e(W)$ for the expression generated by a human for a particular world W, and $[\![e]\!]_W$ for the result of evaluating the logical form e against W.

We are interested in using language data of this kind to analyze the behavior of a deep model trained to play the same game. We focus our analysis on a standard RNN encoder–decoder, with the encoder playing the role of the speaker and the decoder playing the role of the listener. The encoder is a single-layer RNN with GRU cells (Cho et al., 2014) that consumes both the input world and target labeling and outputs a 64-dimensional hidden representation. We write $f(W)$ for the output of this encoder model on a world W. To make predictions, this representation is passed to a decoder implemented as a multilayer perceptron. The decoder makes an independent labeling decision about every object in W (taking as input both f and a feature representation of a particular object W_i). We write $[\![f]\!]_W$ for the full vector of decoder outputs on W. We train the model maximize classification accuracy on randomly-generated scenes and target sets of the same form as in the GENX dataset.

3 Approach

We are not concerned with the RNN model's raw performance on this task (it achieves nearly perfect accuracy). Instead, our goal is to explore what kinds of messages the model computes in order to achieve this accuracy—and specifically whether these messages contain high-level semantics and low-level structure similar to the referring expressions produced by humans. But how do we judge semantic equivalence between natural language and vector representations? Here, as in Andreas et al. (2017), we adopt an approach inspired by formal semantics, and represent the meaning of messages via their *truth conditions* (Figure 1).

For every problem instance W in the dataset, we have access to one or more human messages $e(W)$ as well as the RNN encoding $f(W)$. The truth-conditional account of meaning suggests that we should judge e and f to be equivalent if they designate the same set of of objects in the world (Davidson, 1967). But it is not enough to compare their predictions solely in the context where they were generated—testing if $[\![e]\!]_W = [\![f]\!]_W$—because any pair of models that achieve perfect accuracy on the referring expression task will make the same predictions in this initial context, regardless of the meaning conveyed.

Instead, we sample a collection of alternative worlds $\{W_i\}$ observed elsewhere in the dataset, and compute a tabular meaning representation *rep*$(e) = \{[\![e]\!]_{W_i}\}$ by evaluating e in each world W_i. We similarly compute *rep*$(f) = \{[\![f]\!]_{W_i}\}$, allowing the learned decoder model to play the role of logical evaluation for message vectors. For

	Theory	Objects	Worlds	Tables
All	Random	0.50	0.00	0.00
	Literal	0.74	0.27	0.05
	Human	0.92	0.63	0.35

Table 1: Agreement with predicted model behavior for the high-level semantic correspondence task, computed for objects (single entries in tabular representation), worlds (rows), and full tables. Referring expressions e generated by humans in a single communicative context are highly predictive of how learned representations f will be interpreted by the decoder across multiple contexts.

logically equivalent messages, these tabular representations are guaranteed to be identical, so the sampling procedure can be viewed as an approximate test of equivalence. It additionally allows us to compute softer notions of equivalence by measuring agreement on individual worlds or objects.

4 Interpreting the meaning of messages

We begin with the simplest question we can answer with this tool: how often do the messages generated by the encoder model have the same meaning as messages generated by humans for the same context? Again, our goal is not to evaluate the performance of the RNN model, but instead our ability to understand its behavior. Does it send messages with human-like semantics? Is it more explicit? Or does it behave in a way indistinguishable from a random classifier?

For each scene in the GENX test set, we compute the model-generated message f and its tabular representation $rep(f)$, and measure the extent to which this agrees with representations produced by three "theories" of model behavior (Figure 2): (1) a **random** theory that accepts or rejects objects with uniform probability, (2) a **literal** theory that predicts membership only for objects that exactly match some object in the original target set, and (3) a **human** theory that predicts according to the most frequent logical form associated with natural language descriptions of the target set (as described in the preceding section). We evaluate agreement at the level of individual objects, worlds, and full tabular meaning representations.

Results are shown in Table 1. Model behavior is well explained by human decisions in the same context: object-level decisions can be predicted with close to 90% accuracy based on human judgments alone, and a third of message pairs agree exactly in every sampled scene, providing strong evidence that they carry the same semantics.

These results suggest that the model has learned a communication strategy that is at least superficially language-like: it admits representations of the same kinds of communicative abstractions that humans use, and makes use of these abstractions with some frequency. But this is purely a statement about the high-level behavior of the model, and not about the structure of the space of representations. Our primary goal is to determine whether this behavior is achieved using low-level structural regularities in vector space that can themselves be associated with aspects of natural language communication.

5 Interpreting the structure of messages

For this we turn to a focused investigation of three specific logical constructions used in natural language: a unary operation (negation) and two binary operations (conjunction and disjunction). All are used in the training data, with a variety of scopes (e.g. *all green objects that are not a triangle*, *all the pieces that are not tan arches*).

Because humans often find it useful to specify the target set by exclusion rather than inclusion, we first hypothesize that the RNN language might find it useful to incorporate some mechanism cor-

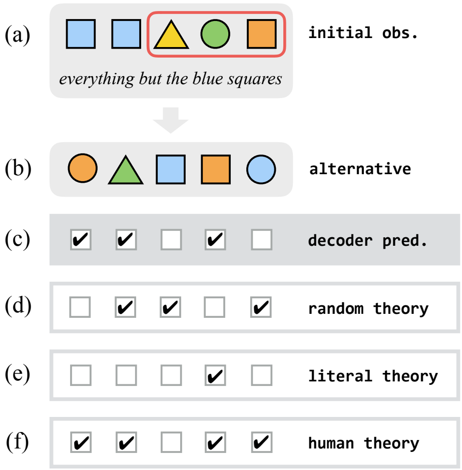


Figure 2: Evaluating theories of model behavior. First, the encoder is run on an initial world (a), producing a representation whose meaning we would like to understand (see Figure 1). We then observe the behavior of the decoder holding this representation fixed but replacing the underlying world representation with alternatives like (b). We compare the true decoder output to a number of *theories* of its behavior. The random theory (d) outputs a random decision for every object. The literal theory (e) predicts that the decoder will output a positive label only on those objects that exactly match some object in the initial observation. The human theory (f) assigns labels according to the logical semantics of the utterance produced by a human presented with the initial observation.

	Theory	Objects	Worlds	Tables
Neg.	Random	0.50	0.00	0.00
	Literal	0.50	0.12	0.03
	Negation	0.97	0.81	0.45
Disj.	Random	0.50	0.00	0.00
	Literal	0.58	0.09	0.01
	Disjunction	0.92	0.54	0.19
Conj.	Random	0.50	0.00	0.00
	Literal	0.81	0.19	0.01
	Conjunction	0.90	0.56	0.37

Table 2: Agreement with predicted model behavior for negation, conjunction, and disjunction tasks (top to bottom). Evaluation is performed on transformed message vectors as described in Section 5. We discover a robust linear transformation of message vectors corresponding to negation, as well as evidence of structured representations of binary operations.

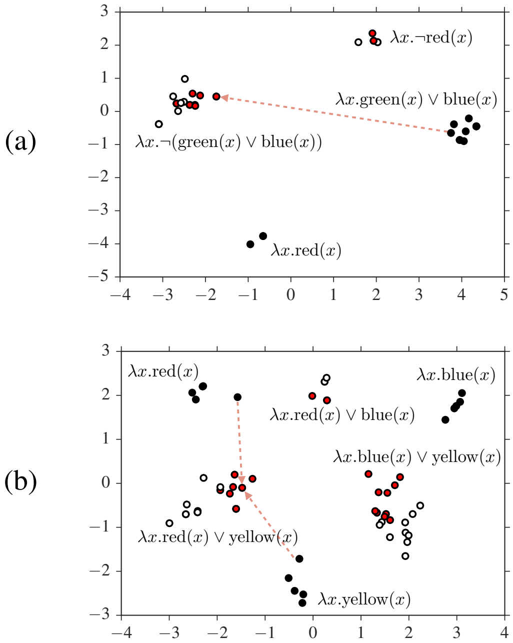


Figure 3: Principal components of structured message transformations discovered by our experiments. (a) Negation: black and white dots show raw message vectors denotationally equivalent to the provided logical cluster label (Section 3). Red dots show the result of transforming black dots with the estimated negation operation N. (b) The corresponding experiment for disjunction using the transformation M.

responding to negation, and that messages can be predictably "negated" in vector space. To test this hypothesis, we first collect examples of the form (e, f, e', f'), where $e' = \neg e$, $\textit{rep}(e) = \textit{rep}(f)$, and $\textit{rep}(e') = \textit{rep}(f')$. In other words, we find pairs of pairs of RNN representations f and f' for which the natural language messages (e, e') serve as a denotational *certificate* that f' behaves as a negation of f. If the learned model does not have any kind of primitive notion of negation, we expect that it will not be possible to find any kind of predictable relationship between pairs (f, f'). (As an extreme example, we could imagine every possible prediction rule being associated with a different point in the representation space, with the correspondence between position and behavior essentially random.) Conversely, if there is a first-class notion of negation, we should be able to select an arbitrary representation vector f with an associated referring expression e, apply some transformation N to f, and be able to predict *a priori* how the decoder model will interpret the representation Nf—i.e. in correspondence with $\neg e$.

Here we make the strong assumption that the negation operation is not only predictable but *linear*. Previous work has found that linear operators are powerful enough to capture many hierarchical and relational structures (Paccanaro and Hinton, 2002; Bordes et al., 2014). Using examples (f, f') collected from the training set as described above, we compute the least-squares estimate $\hat{N} = \arg\min_N \sum ||Nf - f'||_2^2$. To evaluate, we collect example representations from the test set that are equivalent to known logical forms, and measure how frequently model behaviors $\textit{rep}(Nf)$ agree with the logical predictions $\textit{rep}(\neg e)$—in other words, how often the linear operator N actually corresponds to logical negation. Results are shown in the top portion of Table 2. Correspondence with the logical form is quite high, resulting in 97% agreement at the level of individual objects and 45% agreement on full representations. We conclude that the estimated linear operator $\hat{N}$ is analogous to negation in natural language. Indeed, the behavior of this operator is readily visible in Figure 3: predicted negated forms (in red) lie close in vector space to their true values, and negation corresponds roughly to mirroring across a central point.

In our final experiment, we explore whether the same kinds of linear maps can be learned for the binary operations of conjunction and disjunction. As in the previous section, we collect examples from the training data of representations whose denotations are known to correspond to groups of logical forms in the desired relationship—in this case tuples (e, f, e', f', e'', f''), where $\textit{rep}(e) = \textit{rep}(f)$, $\textit{rep}(e') = \textit{rep}(f')$, $\textit{rep}(e'') = \textit{rep}(f'')$ and either $e'' = e \wedge e'$ (conjunction) or $e'' = e \vee e'$ (disjunction). Since we expect that our operator will be symmetric in its arguments, we solve for $\hat{M} = \arg\min_M \sum ||Mf + Mf' - f''||_2^2$.

Results are shown in the bottom portions of Table 2. Correspondence between the behavior predicted by the contextual logical form and the model's actual behavior is less tight than for negation. At the same time, the estimated operators are clearly capturing some structure: in the case of disjunction, for example, model interpretations are correctly modeled by the logical form 92% of the time at the object level and 19% of the time at the denotation level. This suggests that the operations of conjunction and disjunction do have some functional counterparts in the RNN language, but that these functions are not everywhere well approximated as linear.

6 Conclusions

Building on earlier tools for identifying neural codes with natural language strings, we have presented a technique for exploring compositional structure in a space of vector-valued representations. Our analysis of an encoder–decoder model trained on a reference game identified a number of language-like properties in the model's representation space, including transformations corresponding to negation, disjunction, and conjunction. One major question left open by this analysis is what happens when multiple transformations are applied hierarchically, and future work might focus on extending the techniques in this paper to explore recursive structure. We believe our experiments so far highlight the usefulness of a denotational perspective from formal semantics when interpreting the behavior of deep models.

References

Jacob Andreas, Anca Dragan, and Dan Klein. 2017. Translating neuralese. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676* .

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

Donald Davidson. 1967. Truth and meaning. *Synthese* 17(1):304–323.

Christopher Dircks and Scott Stoness. 1999. Effective lexicon change in the absence of population flux. *Advances in Artificial Life* pages 720–724.

Nicholas FitzGerald, Yoav Artzi, and Luke Zettlemoyer. 2013. Learning distributions over logical forms for referring expression generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Jakob Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*. pages 2137–2145.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2016. Towards multi-agent communication-based language learning. *arXiv preprint arXiv:1605.07133* .

Omer Levy, Yoav Goldberg, and Israel Ramat-Gan. 2014. Linguistic regularities in sparse and explicit word representations. pages 171–180.

Igor Mordatch and Pieter Abbeel. 2017. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908* .

Alberto Paccanaro and Jefferey Hinton. 2002. Learning hierarchical structures with linear relational embedding. In *Advances in Neural Information Processing Systems*. Vancouver, BC, Canada, volume 14, page 857.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Sainbayar Sukhbaatar, Rob Fergus, et al. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*. pages 2244–2252.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. pages 3104–3112.

Kyle Wagner, James A Reggia, Juan Uriagereka, and Gerald S Wilkinson. 2003. Progress in the simulation of emergent communication and language. *Adaptive Behavior* 11(1):37–69.

Licheng Yu, Hao Tan, Mohit Bansal, and Tamara L Berg. 2016. A joint speaker-listener-reinforcer model for referring expressions. *arXiv preprint arXiv:1612.09542* .

A Simple Regularization-based Algorithm for Learning Cross-Domain Word Embeddings

Wei Yang
University of Waterloo
w85yang@uwaterloo.ca

Wei Lu
Singapore University
of Technology and Design
luwei@sutd.edu.sg

Vincent W. Zheng
Advanced Digital Sciences Center
vincent.zheng
@adsc.com.sg

Abstract

Learning word embeddings has received a significant amount of attention recently. Often, word embeddings are learned in an unsupervised manner from a large collection of text. The genre of the text typically plays an important role in the effectiveness of the resulting embeddings. How to effectively train word embedding models using data from different domains remains a problem that is underexplored. In this paper, we present a simple yet effective method for learning word embeddings based on text from different domains. We demonstrate the effectiveness of our approach through extensive experiments on various down-stream NLP tasks.

1 Introduction

Recently, the learning of distributed representations for natural language words (or word embeddings) has received a significant amount of attention (Mnih and Hinton, 2007; Turian et al., 2010; Mikolov et al., 2013a,b,c; Pennington et al., 2014). Such representations were shown to be able to capture syntactic and semantic level information associated with words (Mikolov et al., 2013a). Word embeddings were shown effective in tasks such as named entity recognition (Sienčnik, 2015), sentiment analysis (Li and Lu, 2017) and syntactic parsing (Durrett and Klein, 2015). One common assumption made by most of the embedding methods is that, the text corpus is from one single domain; e.g., articles from bioinformatics. However, in practice, there are often text corpora from multiple domains; e.g., we may have text collections from broadcast news or Web blogs, whose words are not necessarily limited to bioinformatics. Can these corpora from different domains help learn better word embeddings, so as to improve the downstream NLP applications in a target domain like bioinformatics? Our answer is yes, because despite the domain differences, these additional domains do introduce more text data converying useful information (i.e., more words, more word co-occurrences), which can be helpful for consolidating the word embeddings in the target bioinformatics domain.

In this paper, we propose a simple and easy-to-implement approach for learning cross-domain word embeddings. Our model can be seen as a regularized *skip-gram* model (Mikolov et al., 2013a,b), where the source domain information is selectively incorporated for learning the target domain word embeddings in a principled manner.

2 Related Work

Learning a continuous representation for words has been studied for quite a while (Hinton et al., 1986). Many earlier word embedding methods employed the computationally expensive neural network architectures (Collobert and Weston, 2008; Mikolov et al., 2013c). Recently, an efficient method for learning word representations, namely the skip-gram model (Mikolov et al., 2013a,b) was proposed and implemented in the widely used word2vec toolkit. It tries to use the current word to predict the surrounding context words, where the prediction is defined over the embeddings of these words. As a result, it learns the word embeddings by maximizing the likelihood of predictions.

Domain adaptation is an important research topic (Pan et al., 2013), and it has been considered in many NLP tasks. For example, domain adaptation is studied for sentiment classification (Glorot et al., 2011) and parsing (McClosky et al., 2010), just to name a few. However, there is very

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2898–2904
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

little work on domain adaptation for word embedding learning. One major reason preventing people from using text corpora from different domains for word embedding learning is the lack of guidance on which kind of information is worth learning from the source domain(s) for the target domain. In order to address this problem, some pioneering work has looked into this problem. For example, Bollegala et al. (2015) considered those frequent words in the source domain and the target domain as the "pivots". Then it tried to use the pivots to predict the surrounding "non-pivots", meanwhile ensuring the pivots to have the same embedding across two domains. Embeddings learned from such an approach were shown to be able to improve the performance on a cross-domain sentiment classification task. However, this model fails to learn embeddings for many words which are neither pivots nor non-pivots, which could be crucial for some downstream tasks such as named entity recognition.

3 Our Approach

Let us first state the objective of the skip-gram model (Mikolov et al., 2013a) as follows:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}} &= \sum_{(w,c)\in\mathcal{D}} \#(w,c)\Big(\log\sigma(\mathbf{w}\cdot\mathbf{c}) \\ &+ \sum_{i=1}^{k}\mathbb{E}_{c'_i\sim P(w)}[\log\sigma(-\mathbf{w}\cdot\mathbf{c'}_i)]\Big)\end{aligned} \quad (1)$$

where $\mathcal{D}$ refers to the complete text corpus from which we learn the word embeddings. The word w is the current word, c is the context word, and $\#(w,c)$ is the number of times they co-occur in $\mathcal{D}$. We use $\mathbf{w}$ and $\mathbf{c}$ to denote the vector representations for w and c, respectively. The function $\sigma(\cdot)$ is the sigmoid function. The word c'_i is a "negative sample" sampled from the distribution $P(w)$ – typically chosen as the unigram distribution $U(w)$ raised to the $3/4$rd power (Mikolov et al., 2013b).

In our approach, we first learn for each word w an embedding $\mathbf{w}_s$ from the source domain $\mathcal{D}_s$. Next we learn the target domain embeddings as follows:

$$\mathcal{L}'_{\mathcal{D}_t} = \mathcal{L}_{\mathcal{D}_t} + \sum_{w\in\mathcal{D}_t\cap\mathcal{D}_s} \alpha_w \cdot ||\mathbf{w}_t - \mathbf{w}_s||^2 \quad (2)$$

where $\mathcal{D}_t$ refers to the target domain, and $\mathbf{w}_t$ is the target domain representation for w. Such an regularized objective can still be optimized using standard stochastic gradient descent. Note that in the above formula, the regularization term only considers words that appear in both source and target domain, ignoring words that only appear in either the source or the target domain only.

Our approach is inspired by the recent regularization-based domain adaptation framework (Lu et al., 2016). Here, α_w measures the amount of transfer across the two domains when learning the representation for word w. If it is large, it means we require the embeddings of word w in the two domains to be similar. We define α_w as follows:

$$\alpha_w = \sigma(\lambda\cdot\phi(w)) \quad (3)$$

where λ is a hyper-parameter to decide the scaling factor of the significance function $\phi(\cdot)$, which allows the user to control the degree of "knowledge transfer" from source domain to target domain.

How do we define the significance function $\phi(w)$ that controls the amount of transfer for the word w? We first define the frequency of the word w in the dataset $\mathcal{D}$ as $f_{\mathcal{D}}(w)$, the number of times the word w appears in the domain $\mathcal{D}$. Based on this we can define the *normalized* frequency for the word w as follows:

$$\mathcal{F}_{\mathcal{D}}(w) = \frac{f_{\mathcal{D}}(w)}{\max_{w'\in\mathcal{D}_k} f_{\mathcal{D}}(w')} \quad (4)$$

where $\mathcal{D}_k \subset \mathcal{D}$ consists of all except for the top k most frequent words from $\mathcal{D}$[1].

We define the function $\phi(\cdot)$ based on the following metric that is motivated by the well-known Sørensen-Dice coefficient (Sørensen, 1948; Dice, 1945) commonly used for measuring similarities:

$$\phi(w) = \frac{2\cdot\mathcal{F}_{\mathcal{D}_s}(w)\cdot\mathcal{F}_{\mathcal{D}_t}(w)}{\mathcal{F}_{\mathcal{D}_s}(w)+\mathcal{F}_{\mathcal{D}_t}(w)} \quad (5)$$

Why does such a definition make sense? We note that the value of $\phi(w)$ would be high only if both both $\mathcal{F}_{\mathcal{D}_s}(w)$ and $\mathcal{F}_{\mathcal{D}_t}(w)$ are high – in this case the word w is a frequent word across different domains. Intuitively, these are likely those words whose semantics do not change across the two domains, and we should be confident about making their embeddings similar in the two domains. On the other hand, domain-specific words

[1] In all our experiments, we empirically set k to 20.

	Enwik9	PubMed	Gigaword (En)	Yelp	IMDB	Tweets (En)	Tweets (Es)	Eswiki
# tokens	124.3M	124.9M	135.6M	38.9M	29.0M	162.8M	69.4M	102.8M
# sents	–	5,000,000	5,400,000	2,376,079	1,230,465	16,185,356	6,785,697	3,684,670

Table 1: Statistics for datasets used for embedding learning in all experiments.

tend to be more frequent in one domain than the other. In this case, the resulting $\phi(w)$ will also have a lower score, indicating a smaller amount of transfer across the two domains. While other user-defined significance functions are also possible, in this work we simply adopt such a function based on the above simple observations. We will validate our assumptions with experiments in the next section.

4 Experiments

We present extensive evaluations to assess the effectiveness of our approach. Following recent advice by Nayak et al. (2016) and Faruqui et al. (2016), to assess the quality of the learned word embeddings, we considered employing the learned word embeddings as continuous features in several down-stream NLP tasks, including entity recognition, sentiment classification, and targeted sentiment analysis.

We have used various datasets from different domains for learning cross-domain word embeddings under different tasks. We list the data statistics in Table 1.

4.1 Baseline Methods

We consider the following baseline methods when assessing the effectiveness of our approach.

- Discrete: only discrete features (such as bag of words, POS tags, word n-grams and POS tag n-grams, depending on the actual down-stream task) were considered. All following systems include both these base features and the respective additional features.

- Source: we train word embeddings from the source domain as additional features.

- Target: we train word embeddings from the target domain as additional features.

- All: we combined the data from two domains to form a single dataset for learning word embeddings as additional features.

- Concat: we simply concatenate the learned embeddings from both source and target domains as additional features.

Method	GENIA			ACE		
	P	R	F_1	P	R	F_1
Discrete	71.1	63.9	67.3	64.5	52.3	57.7
Source	71.1	62.3	66.4	63.5	57.3	60.3
Target	71.6	64.5	67.9	63.3	57.1	60.0
All	71.2	61.8	66.1	**64.6**	57.2	60.7
Concat	71.5	64.1	67.6	63.5	57.7	60.5
DARep	71.4	61.5	66.1	62.4	54.5	58.2
This work	**72.4**	**65.4**	**68.7**	64.5	**58.9**	**61.6**

Table 2: Results on entity recognition.

- DARep: we use the previous approach of Bollegala et al. (2015) for learning cross-lingual word representations as additional features.

4.2 Entity Recognition

Our first experiment was conducted on entity recognition (Tjong Kim Sang and De Meulder, 2003; Florian et al., 2004), where the task is to extract semantically meaning entities and their mentions from the text.

For this task, we built a standard entity recognition model using conditional random fields (Lafferty et al., 2001). We used the standard features which are commonly used for different methods, including word unigrams and bigrams, bag-of-words features, POS tag window features, POS tag unigrams and bigram features. We conducted two sets of experiments on two different datasets. The first dataset is the GENIA dataset (Ohta et al., 2002), a popular dataset used in bioinformatics, and the second is the ACE-2005 dataset (Walker et al., 2006), which is a standard dataset used for various information extraction tasks.

For the GENIA dataset which consists of 10,946 sentences, we used Enwik9 as the source domain and PubMed as the target domain for learning word embeddings. We set the dimension of word representations as 50.

For the experiments on ACE, we selected the BN subset of ACE2005, which consists of 4,460 CNN headline news and share a similar domain with Gigaword. We used Enwik9 as the source domain and Gigaword as the target domain. We followed a procedure similar to GENIA for experiments.

To tune our hyperparameter λ, we first split the last 10% of the training set as the development

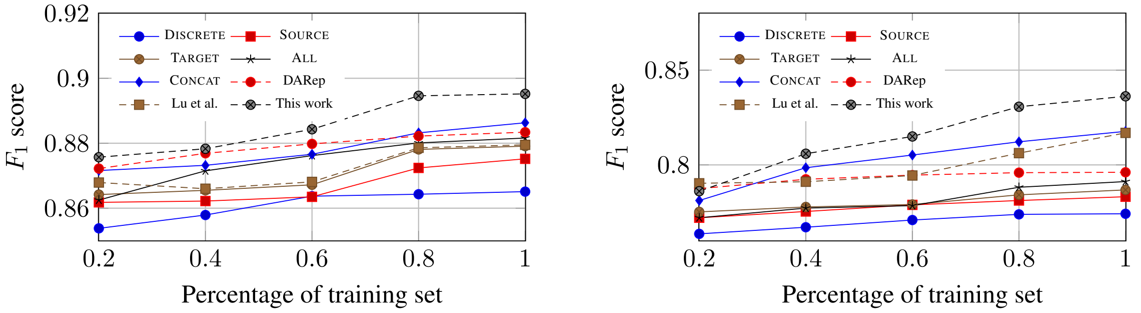


Figure 1: Results on sentiment classification. Left: Yelp (source) to IMDB (target). Right: IMDB (source) to Yelp (target).

portion. We then trained a model using the remaining 90% as the training portion and used the development portion for development of the hyperparameter λ. After development, we re-trained the models using the original training set[2].

We report the results in Table 2. From the results we can observe that the embeddings learned using our algorithm can lead to improved performance when used in this particular down-stream NLP task. We note that in such a task, many entities consist of domain-specific terms, therefore learning good representations for such words can be crucial. As we have discussed earlier, our regularization method enables our model to differentiate domain-specific words from words which are more general in the learning process. We believe this mechanism can lead to improved learning of representations for both types of words.

4.3 Sentiment Classification

The second task we consider is sentiment classification, which is essentially a text classification task, where the goal is to assign each text document a class label indicating its sentiment polarity (Pang et al., 2002; Liu, 2012).

This is also the only task presented in the previous DARep work by Bollegala et al. (2015). As such, we largely followed Bollegala et al. (2015) for experiments. Instead of using the dataset they used which only consists of 2,000 reviews, we considered two much larger datasets – IMDB and Yelp 2014 – for such a task, which was previously used in a sentiment classification task (Tang et al., 2015). IMDB dataset (Diao et al., 2014) is crawled from the movie review site IMDB[3] which consists of 84,919 reviews. Yelp 2014 dataset consists of 231,163 online reviews provided by the Yelp Dataset Challenge[4].

Following Bollegala et al. (2015), for this task we simply learned the word embeddings from the training portion of the review datasets themselves only. No external data was used for learning word embeddings. As Bollegala et al. (2015) only evaluated on a small dataset in their paper for such a task, to understand the effect of varying the amount of training data, we also tried to train our model on datasets with different sizes. We conducted two sets of experiments: we first used the Yelp dataset as the source domain and IMDB as the target domain, and then we switched these two datasets in our second set of experiments. Figure 1 shows the F_1 measures for different word embeddings when different amounts of training data were used. We also compared with the previous approach for domain adaptation (Lu et al., 2016) which only employs discrete features. We can observe that when the dataset becomes large, our learned word embeddings are shown to be more effective than all other approaches. When the complete training set is used, our model significantly outperforms DARep ($p < 0.05$ for both directions with bootstrap resampling test (Koehn, 2004)). DARep appears to be effective when the training dataset is small. However, as the training set size increases, there is no significant improvement for such an approach. As we can also observe from the figure, our approach consistently gives better results than baseline approaches (except for the second experiment when 20% of the data was used). Furthermore, when the amount of training data increases, the differences between our approach and other approaches generally become larger.

Such experiments show that our model works

[2] We selected the optimal value for the hyper-parameter λ from the set $\lambda \in \{0.1, 1, 5, 10, 20, 30, 50\}$ for all experiments in this paper.

[3] http://www.imdb.com

[4] https://www.yelp.com/dataset_challenge

Model	English			Spanish		
	P.	*R.*	F_1	*P.*	*R.*	F_1
Discrete	44.8	37.0	40.5	46.0	39.8	42.7
Source	44.1	36.3	39.8	46.1	40.5	43.1
Target	46.5	39.1	42.5	46.5	40.8	43.4
All	45.4	37.0	40.8	46.4	40.7	43.3
Concat	46.7	39.3	42.7	**46.6**	41.0	43.6
DARep	46.2	39.8	42.8	46.2	40.9	43.4
This work	**46.9**	**39.9**	**43.1**	**46.6**	**41.4**	**43.9**

Table 3: Results on targeted sentiment analysis.

well when different amounts of data are available, and our approach appears to be more competitive when a large amount of data is available.

4.4 Targeted Sentiment Analysis

We also conducted experiments on targeted sentiment analysis (Mitchell et al., 2013) – the task of jointly recognizing entities and their sentiment information. We used the state-of-the-art system for targeted sentiment analysis by Li and Lu (2017) whose code is publicly available [5], and used the data from (Mitchell et al., 2013) which consists of 7,105 Spanish tweets and 2,350 English tweets, with named entities and their sentiment information annotated. Note that the model of Li and Lu (2017) is a structured prediction model that involves latent variables. The experiments here therefore allow us to assess the effectiveness of our approach on such a setup involving latent variables. We follow Li and Lu (2017) and report precision ($P.$), recall ($R.$) and F1-measure (F_1) for such a targeted sentiment analysis task, where the prediction is regarded as correct if and only if both the entity's boundary and its sentiment information are correct. Also, unlike previous experiments, which are conducted on English only, these experiments additionally allow us to assess our approach's effectiveness when a different language other than English is considered.

For the English task, we used Enwik9 as the source domain for learning word embeddings, and our crawled English tweets as the target domain. For the Spanish task, we used Eswiki as the source domain, and we also crawled Spanish tweets as the target domain. See Table 1 for the statistics. Similar to the experiments conducted for entity recognition, we split the first 80% of the data for training, the next 10% for development and the last 10% for evaluation. We tuned the hyper-parameter λ using the development set and re-trained the embeddings on the dataset combining the training and the development set, which are then used in final evaluations. Results are reported in Table 3, which show our approach is able to achieve the best results across two datasets in such a task, and outperforms DARep ($p < 0.05$). Interestingly, the concatenation approach appears to be competitive in this task, especially for the Spanish dataset, which appears to be better than the DARep approach. However, we note such an approach does not capture any information transfer across different domains in the learning process. In contrast, our approach learns embeddings for the target domain by capturing useful cross-domain information and therefore can lead to improved modeling of embeddings that are shown more helpful for this specific down-stream task.

5 Conclusion and Future Work

In this paper, we presented a simple yet effective algorithm for learning cross-domain word embeddings. Motivated by the recent regularization-based domain adaptation framework (Lu et al., 2016), the algorithm performs learning by augmenting the skip-gram objective with a simple regularization term. Our work can be easily extended to multi-domain scenarios. The method is also flexible, allowing different user-defined metrics to be incorporated for defining the function controlling the amount of domain transfer.

Future work includes performing further investigations to better understand and to visualize what types of information has been transferred across domains and how such information influence different types of down-stream NLP tasks. It is also important to understand how such an approach will work on other types of models such as neural networks based NLP models. Our code is available at `http://statnlp.org/research/lr/`.

Acknowledgments

We thank all the reviewers for their useful feedback to the earlier draft of this paper. This work was done when the first author was visiting Singapore University of Technology and Design. We thank the support of Human-centered Cyber-physical Systems Programme at Advanced Digital Sciences Center from Singapores Agency for Science, Technology and Research (A*STAR). This work is supported by MOE Tier 1 grant SUTDT12015008.

[5] Available at http://statnlp.org/research/st/.

References

Danushka Bollegala, Takanori Maehara, and Ken ichi Kawarabayashi. 2015. Unsupervised cross-domain word representation learning. In *Proc. of ACL-IJCNLP*, pages 730–740.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*, pages 160–167. ACM.

Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proc. of KDD*, pages 193–202.

Lee R Dice. 1945. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. *arXiv preprint arXiv:1507.03641*.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint arXiv:1605.02276*.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N. Nicolov, and S. Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proc. of NAACL/HLT*, pages 1–8.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proc. of ICML*, pages 513–520.

G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. 1986. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Distributed Representations, pages 77–109. MIT Press, Cambridge, MA, USA.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP*, pages 388–395.

John Lafferty, Andrew McCallum, Fernando Pereira, et al. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, pages 282–289.

Hao Li and Wei Lu. 2017. Learning latent sentiment scopes for entity-level sentiment analysis. In *Proc. of AAAI*, pages 3482–3489.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1).

Wei Lu, Hai Leong Chieu, and Jonathan Löfgren. 2016. A general regularization framework for domain adaptation. In *Proc. of EMNLP*, pages 950–954.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Proc. of NAACL-HLT*, pages 28–36.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proc. of NAACL-HLT*, pages 746–751.

Margaret Mitchell, Jacqueline Aguilar, Theresa Wilson, and Benjamin Van Durme. 2013. Open domain targeted sentiment. In *Proc. of EMNLP*, pages 1643–1654.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proc. of ICML*, pages 641–648.

Neha Nayak, Gabor Angeli, and Christopher D Manning. 2016. Evaluating word embeddings using a representative suite of practical tasks. *ACL 2016*, page 19.

Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. 2002. The genia corpus: An annotated research abstract corpus in molecular biology domain. In *Proc. of the second international conference on Human Language Technology Research*, pages 82–86.

Sinno Jialin Pan, Zhiqiang Toh, and Jian Su. 2013. Transfer joint embedding for cross-domain named entity recognition. *ACM Transactions on Information Systems (TOIS)*, 31(2):7.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proc. of EMNLP*, pages 79–86.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*, volume 14, pages 1532–1543.

Scharolta Katharina Sienčnik. 2015. Adapting word2vec to named entity recognition. In *Proc. of NODALIDA*, 109, pages 239–243. Linköping University Electronic Press.

Thorvald Sørensen. 1948. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. *Biol. Skr.*, 5:1–34.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Proc. of ACL-IJCNLP*, pages 1014–1023.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc. of HLT-NAACL*, pages 142–147.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57.

Learning what to read: Focused machine reading

Enrique Noriega-Atala
Marco A. Valenzuela-Escárcega

Clayton T. Morrison
Mihai Surdeanu

University of Arizona
Tucson, Arizona, USA

{enoriega,marcov,claytonm,msurdeanu}@email.arizona.edu

Abstract

Recent efforts in bioinformatics have achieved tremendous progress in the machine reading of biomedical literature, and the assembly of the extracted biochemical interactions into large-scale models such as protein signaling pathways. However, batch machine reading of literature at today's scale (PubMed alone indexes over 1 million papers per year) is unfeasible due to both cost and processing overhead. In this work, we introduce a focused reading approach to guide the machine reading of biomedical literature towards *what* literature should be read to answer a biomedical query as efficiently as possible. We introduce a family of algorithms for focused reading, including an intuitive, strong baseline, and a second approach which uses a reinforcement learning (RL) framework that learns when to explore (widen the search) or exploit (narrow it). We demonstrate that the RL approach is capable of answering more queries than the baseline, while being more efficient, i.e., reading fewer documents.

1 Introduction

The millions of academic papers in the biomedical domain contain a vast amount of information that may lead to new hypotheses for disease treatment. However, scientists are faced with a problem of "undiscovered public knowledge," as they struggle to read and assimilate all of this information (Swanson, 1986). Furthermore, the literature is growing at an exponential rate (Pautasso, 2012); PubMed[1] has been adding more than a million papers per year since 2011. We have surpassed our ability to keep up with and integrate these findings through manual reading alone.

Large ongoing efforts, such as the BioNLP task community (Nédellec et al., 2013; Kim et al., 2012, 2009) and the DARPA Big Mechanism Program (Cohen, 2015), are making progress in advancing methods for machine reading and assembly of extracted biochemical interactions into large-scale models. However, to date, these methods rely either on the manual selection of relevant documents, or on the processing of large batches of documents that may or may not be relevant to the model being constructed.

Batch machine reading of literature at this scale poses a new, growing set of problems. First, access to some documents is costly. The PubMedCentral (PMC) Open Access Subset[2] (OA) is estimated[3] to comprise 20%[4] of the total literature; the remaining full-text documents are only available through paid access. Second, while there have been great advances in quality, machine reading is still not solved. Updates to our readers requires reprocessing the documents. For large document corpora, this quickly becomes the chief bottleneck in information extraction for model construction and analysis. Finally, even if we could cache all reading results, the search for connections between concepts within the extracted results should not be done blindly. At least in the biology domain, the many connections between biological entities and processes leads to a very high branching factor, making blind search for paths intractable.

To effectively read at this scale, we need to incorporate methods for *focused reading*: develop the ability to pose queries about concepts of interest and perform targeted, incremental search

[1] http://www.ncbi.nlm.nih.gov/pubmed

[2] https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/

[3] https://tinyurl.com/bachman-oa

[4] This includes 5% from PMC author manuscripts.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2905–2910
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

through the literature for connections between concepts while minimizing reading documents that are likely irrelevant.

In this paper we present what we believe is the first algorithm for focused reading. We make the following contributions:
(1) Present a general framework for a family of possible focused reading algorithms along with a baseline instance.
(2) Cast the design of focused reading algorithms in a reinforcement learning (RL) setting, where the machine decides if it should explore (i.e., cast a wider net) or exploit (i.e., focus reading on a specific topic).
(3) Evaluate our focused reading policies in terms of search efficiency and quality of information extracted. The evaluation demonstrates the effectiveness of the RL method: this approach found more information than the strong baseline we propose, while reading fewer documents.

2 Related Work

The past few years have seen a large body of work on information extraction (IE), particularly in the biomedical domain. This work is too vast to be comprehensively discussed here. We refer the interested reader to the BioNLP community (Nédellec et al., 2013; Kim et al., 2012, 2009, inter alia) for a starting point. However, most of this work focuses on *how* to read, not on *what* to read given a goal. To our knowledge, we are the first to focus on the latter task.

Reinforcement learning has been used to achieve state of the art performance in several natural language processing (NLP) and information retrieval (IR) tasks. For example, RL has been used to guide IR and filter irrelevant web content (Seo and Zhang, 2000; Zhang and Seo, 2001). More recently, RL has been combined with deep learning with great success, e.g., for improving coreference resolution (Clark and Manning, 2016). Finally, RL has been used to improve the efficiency of IE by learning how to incrementally reconcile new information and help choose what to look for next (Narasimhan et al., 2016), a task close to ours. This serves as an inspiration for the work we present here, but with a critical difference: Narasimhan et al. (2016) focus on slot filling using a pre-existing template. This makes both the information integration and stopping criteria well-defined. On the other hand, in our focused reading domain, we do not know ahead of time which new pieces of information are necessarily relevant and must be taken in context.

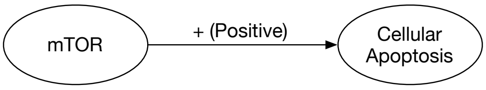


Figure 1: Example of a graph edge encoding the relation extracted from the text: *mTOR triggers cellular apoptosis*.

3 Focused Reading

Here we consider focused reading for the biomedical domain, and we focus on binary promotion/inhibition interactions between biochemical entities. In this setting, the machine reading (or IE) component constructs a directed graph, where vertices represent *participants* in an interaction (e.g., protein, gene, or a biological process), and edges represent directed activation interactions. Edge labels indicate whether the controller entity has a *positive* (promoting) or *negative* (inhibitory) influence on the controlled participant. Figure 1 shows an example edge in this graph.

We use REACH[5], an open source IE system (Valenzuela-Escárcega et al., 2015), to extract interactions from unstructured biomedical text and construct the graph above. We couple this IE system with a Lucene[6] index of biomedical publications to retrieve papers based on queries about participant mentions in the text (as discussed below).

Importantly, we essentially use IE as a black box (thus, our method could potentially work with any IE system), and focus on strategies that guide *what* the IE system reads for a complex information need. In particular, we consider the common scenario where a biologist (or other model-building process) queries the literature on:

> How does one participant (source) affect another (destination), where the connection is typically indirect?

This type of queries is common in biology, where such direct/indirect interactions are observed in experiments, but the explanation of why these dependencies exist is unclear.

Algorithm 1 outlines the general focused reading algorithm for this task. In the algorithm,

[5] https://github.com/clulab/reach
[6] https://lucene.apache.org

S, D, A, and B represent individual participants, where S and D are the **s**ource and **d**estination entities in the initial user query. G is the interaction graph that is iteratively constructed during the focused reading procedure, with V being the set of vertices (biochemical entities), and E the set of edges (promotion/inhibition interactions). Σ is the strategy that chooses which two entities/vertices to be used in the next information retrieval iteration. Q is a Lucene query automatically constructed in each iteration to retrieve new papers to read.

Algorithm 1 Focused reading framework

```
 1: procedure FOCUSEDREADING($S, D$)
 2:     $G \leftarrow \{\{S, D\}, \emptyset\}$
 3:     repeat
 4:         $\Sigma \leftarrow$ ENDPOINTSTRATEGY($G$)
 5:         $(A, B) \leftarrow$ CHOOSEENDPOINTS($\Sigma, G$)
 6:         $Q \leftarrow$ CHOOSEQUERY($A, B, G$)
 7:         $(V, E) \leftarrow$ LUCENE+REACH($Q$)
 8:         EXPAND($V, E, G$)
 9:     until ISCONNECTED($S, D$) OR STOPCONDITIONMET($G$)
10: end procedure
```

The algorithm initializes the search graph as containing the two unconnected participants as vertices: $\{S, D\}$ (line 2). The algorithm then enters into its central loop (lines 3 through 9). The loop terminates when one or more directed paths connecting S to D are found, or when a stopping condition is met: either G has not changed since the previous run through the loop, or after exceeding some number of iterations through the loop (in this work, ten).

At each pass through the loop the algorithm grows the search graph as follows:

1. The graph G is initialized with two nodes, the source and destination in the user's information need, and no edges (because we have not read any papers yet).
2. Given the current graph, choose a strategy, Σ, for selecting which entities to query next: *exploration* or *exploitation* (line 4). In general, exploration aims to widen the search space by adding many more nodes to the graph, whereas exploitation aims to narrow the search by focusing on entities in a specific region of the graph.
3. Using strategy Σ, choose the next entities to attempt to link: (A, B) (line 5).
4. Choose a query, Q: again, using *exploration* or *exploitation*, following the same intuition as with the entity choice strategy (line 6). Here exploration queries retrieve a wider range of documents, while exploitation queries are more restrictive.
5. Run the Lucene query to retrieve papers and process the papers using the IE system. The result of this call is a set of interactions, similar to that in Figure 1 (line 7).
6. Add the new interaction participant entities (vertices V) and directed influences (edges E) to the search graph (line 8).
7. If the source and destination entities are connected in G, stop: the user's information need has been addressed. Otherwise, continue from step 2.

The central loop performs a bidirectional search in which each iteration expands the search horizon outward from S and D. Algorithm 1 represents a family of possible focused reading algorithms, differentiated by how each of the functions in the main loop are implemented. In this work, ISCONNECTED stops after a single path is found, but a variant could consider finding multiple paths, paths of some length, or incorporate other criteria about the properties of the path. We next consider particular choices for the inner loop functions.

4 Baseline Algorithm and Evaluation

The main functions that affect the search behavior of Algorithm 1 are ENDPOINTSTRATEGY and CHOOSEQUERY. Here we describe a *baseline* focused reading implementation in which ENDPOINTSTRATEGY and CHOOSEQUERY aim to find any path between S and D as quickly as possible.

For ENDPOINTSTRATEGY, we follow the intuition that some participants in a biological graph tend to be connected to more participants than others, and therefore more likely to yield interactions providing paths between participants in general. Our heuristic is therefore to choose new participants to query that currently have the most inward and outgoing edges (i.e., highest vertex degree) in the current state of G (disallowing choosing an entity pair used in a previous query).

Now that we have our candidate participants (A, B), our next step is to formulate how we will use these participants to retrieve new papers. Here we consider two classes of query: (1) we restrict our query to only retrieve papers that simultaneously mention both A and B, therefore more likely retrieving a paper with a direct link between A and B (*exploit*), or (2) we retrieve papers that mention

	Baseline	**RL Query Policy**	
# IR queries	573	433	25% decrease
Unique papers read	26,197	19,883	24% decrease
# Paths recovered (out of 289)	189 (65%)	198 (68%)	3% increase

Table 1: Results of the baseline and RL Query Policy for the focused reading of biomedical literature.

either A or B, therefore generally retrieving more papers that will introduce more new participants (*explore*). For our baseline, where we are trying to find a path between S and D as quickly as possible, we implement a greedy ChooseQuery: first try the conjunctive exploitation query; if no documents are retrieved, then "relax" the search to the disjunctive exploration query.

To evaluate the baseline, we constructed a data set based on a collection of papers seeded by a set of 132 entities that come from the University of Pittsburgh DyCE[7] model, a biomolecular model of pancreatic cancer (Telmer et al., 2017). Using these entities, we retrieved 70,719 papers that mention them. We processed all papers using REACH, extracting all of the interactions mentioned, and converted them into a single graph. The resulting graph consisted of approximately 80,000 vertices, 115,000 edges, and had an average (undirected) vertex degree of 24. We will refer to this graph as the *REACH graph*, as it represents what *can* be retrieved by REACH from the set of 70K papers. Next, we identified which pairs of the original 132 entities are connected by directed paths in DyCE. A total of 789 pairs were found. We used 289 of these entity pairs as testing queries (i.e., generating queries that aim to explain how a given pair is connected according to the literature). The other 500 pairs were held out to train the RL method described below.

We ran this baseline focused reading algorithm on each of the 289 pairs of participants, in each case attempting to recover a directed path from one to the other. The results are summarized in the middle column of Table 1. By issuing 573 queries, the baseline read 26,197 papers out of the total 70,719 papers (37% of the corpus), in order to recover 189 of the 289 paths (65%).

5 Reinforcement Learning for Focussed Reading

We analyzed the baseline's behavior in the evaluation to identify the conditions under which it failed to find paths. From this, we found that some of the failures could be avoided had we used a different strategy for ChooseQuery, i.e., the baseline chose to exploit when it should have explored more. The conditions for making different choices depend on the current state of G, and earlier query behavior can affect later query opportunities, making this an iterative decision making problem and a natural fit for a RL formulation.

Inspired by this observation, we consider RL for finding a better policy for ChooseQuery. We'll refer to an instance of the focused reading algorithm with a learned ChooseQuery policy as the *RL Query Policy*. All other focus reading functionality is the same as in the baseline. For actions, we consider a simple binary action choice: exploit (conjunctive query) or explore (disjunctive query). We represent the state of the search using a set of features that include: (f1) the current iteration of the search; (f2) the number of times a participant has been used in previous queries; (f3) whether the participants are chosen from the same connected component in G; (f4) the vertex degree of participants; and (f5) the search iteration in which a participant was introduced. With the goal of recovering paths as quickly as possible, we provide a reward of $+1$ if the algorithm successfully finds a path, a reward of -1 if the search fails to find a path, and assess a "living reward" of -0.05 for each step during the search, to encourage trying to finish the search as quickly as possible.

We trained the RL Query Policy using the SARSA (Sutton and Barto, 1998) RL algorithm. As the number of unique states is large, we used a linear approximation of the q-function. Once the policy converged during training, we then fixed the linear estimate of the q-function and used this as a fixed policy for selecting queries. We trained the RL Query Policy on the separate set of 500 entity pairs, and evaluated it on the same data set of 289 participant pairs used to evaluate the baseline. Table 1 summaries the results of both the baseline and the RL Query Policy. The Query Policy resulted in a 25% decrease in the number of queries that were run, leading to a 24% drop in the number of papers that were read, while at the same time *increasing* the number of paths recovered by 3%. We tested the statistical significance of the

[7] **Dy**namic **C**ell **E**nvironment model of pancreatic cancer.

	All features	− Iteration number (f1)	− Query counts (f2)	− Same component (f3)	− Ranks (f4)	− Particip. intro. (f5)
Paths found	198	199	200	201	**202**	196
Papers read	19,883	20,918	20,531	20,463	27,708	**17,936**
Queries made	433	484	484	467	469	**403**

Table 2: Ablation test on the features used to represent the RL state.

	Empty query results	Ungrounded participant(s)	Low yield from IE
Error cause	12	4	2

Table 3: Error analysis on 18 queries that failed under the RL algorithm.

difference in results between the baseline and RL policy by performing a bootstrap resampling test. Our hypotheses were that the policy reads fewer papers, makes fewer queries and finds more paths. The resulting estimated p-values for fewer papers and fewer queries was found to be near 0, and < 0.003 for finding more paths. An ablation study of the state features found that features (f2) and (f5) had the largest impact on number of papers read; both model the history of the reading task (see the next section for details). This highlights that the RL model is indeed learning to model the entire iterative process.

6 Analysis

Feature Ablation Test: We performed an ablation test on the features that encode the RL state. The results are summarized in Table 2. Similar to Section 5, we grouped the features into five different groups, and we measured the impact of removing one feature group at a time. Overall, the amount of paths found doesn't have a significant amount of variance, but the efficiency of the search (amount of papers read and number of queries made) depends on several feature groups. For example, features (f1), (f2), and (f4) have a large effect on both the number of papers read and the number of queries made. Removing the feature (f5) actually reduces the number of papers read by approximately 2K with a minimal reduction in the number of paths found, which suggests that this task could benefit from feature selection.

RL Policy Error Analysis: Lastly, we analyzed the execution trace of eighteen (20% of the errors) of the searches that failed to find a path under RL. The results are summarized in Table 3. The table shows that the main source of failures is receiving *no results* from the information retrieval query, i.e., when the IR system returns zero documents for the chosen query. This is typically caused by over-constrained queries. The second most common source of failures was *ungrounded participants*, i.e., when at least one of the selected participants that form the query could not be linked to our protein knowledge base. This is generally caused by mistakes in our NER sequence model, and also tends to yield no results from the IR component. Finally, the *low yield from IE* situation appears when the the information produced through machine reading in one iteration is scarce and adds no new components to the interaction graph, again resulting in a stop condition.

7 Discussion and future work

We introduced a framework for the focused reading of biomedical literature, which is necessary to handle the data overload that plagues even machine reading approaches. We have presented a generic focused reading algorithm, an intuitive, strong baseline algorithm that instantiates it, and formulated an RL approach that learns how to efficiently query the paper repository that feeds the machine reading component. We showed that the RL-based focused reading is more efficient than the baseline (i.e., it reads 24% fewer papers), while answering 7% more queries.

There are many exciting directions in which to take this work. First, more of the focused reading algorithm can be subject to RL, with the CHOOSEENDPOINTS policy being the clear next candidate. Second, we can expand focused reading to efficiently search for multiple paths between S and D. Finally, we will incorporate additional biological constraints (e.g., focus on pathways that exist in specific species) into the search itself.

Acknowledgments

This work was partially funded by the DARPA Big Mechanism program under ARO contract W911NF-14-1-0395.

Dr. Mihai Surdeanu discloses a financial interest in Lum.ai. This interest has been disclosed to the University of Arizona Institutional Review Committee and is being managed in accordance with its conflict of interest policies.

References

Kevin Clark and Christopher D Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667* .

Paul R. Cohen. 2015. DARPA's Big Mechanism program. *Physical Biology* 12(4):045008.

Jin-Dong Kim, Ngan Nguyen, Yue Wang, Jun'ichi Tsujii, Toshihisa Takagi, and Akinori Yonezawa. 2012. The genia event and protein coreference tasks of the bionlp shared task 2011. *BMC bioinformatics* 13(11):1.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*. Association for Computational Linguistics, pages 1–9.

Karthik Narasimhan, Adam Yala, and Regina Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*.

Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. 2013. Overview of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*. pages 1–7.

Marco Pautasso. 2012. Publication growth in biological sub-fields: patterns, predictability and sustainability. *Sustainability* 4(12):3234–3247.

Young-Woo Seo and Byoung-Tak Zhang. 2000. A reinforcement learning agent for personalized information filtering. In *Proceedings of the 5th international conference on Intelligent user interfaces*. ACM, pages 248–251.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. MIT press Cambridge.

Don R Swanson. 1986. Undiscovered public knowledge. *The Library Quarterly* 56(2):103–118.

C. A. Telmer, K. Sayed, A. A. Butchy, Kaltenmeir, Michael Lotze, and N. Miskov-Zivanov. 2017. Manuscript in preparation.

Marco A. Valenzuela-Escárcega, Gustave Hahn-Powell, Thomas Hicks, and Mihai Surdeanu. 2015. A domain-independent rule-based framework for event extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Assian Federation of Natural Language Processing: Software Demonstrations (ACL-IJCNLP)*.

Byoung-Tak Zhang and Young-Woo Seo. 2001. Personalized web-document filtering using reinforcement learning. *Applied Artificial Intelligence* 15(7):665–685.

DOC: Deep Open Classification of Text Documents

Lei Shu, Hu Xu, Bing Liu
Department of Computer Science
University of Illinois at Chicago
{lshu3, hxu48, liub}@uic.edu

Abstract

Traditional supervised learning makes the *closed-world* assumption that the classes appeared in the test data must have appeared in training. This also applies to text learning or text classification. As learning is used increasingly in dynamic open environments where some new/test documents may not belong to any of the training classes, identifying these novel documents during classification presents an important problem. This problem is called *open-world classification* or *open classification*. This paper proposes a novel deep learning based approach. It outperforms existing state-of-the-art techniques dramatically.

1 Introduction

A key assumption made by classic supervised text classification (or learning) is that classes appeared in the test data must have appeared in training, called the *closed-world* assumption (Fei and Liu, 2016; Chen and Liu, 2016). Although this assumption holds in many applications, it is violated in many others, especially in dynamic or open environments. For example, in social media, a classifier built with past topics or classes may not be effective in classifying future data because new topics appear constantly in social media (Fei et al., 2016). This is clearly true in other domains too, e.g., self-driving cars, where new objects may appear in the scene all the time.

Ideally, in the text domain, the classifier should classify incoming documents to the right existing classes used in training and also detect those documents that don't belong to any of the existing classes. This problem is called *open world classification* or *open classification* (Fei and Liu, 2016). Such a classifier is aware *what it does and does not know*. This paper proposes a novel technique to solve this problem.

Problem Definition: Given the training data $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ is the i-th document, and $y_i \in \{l_1, l_2, \ldots, l_m\} = \mathcal{Y}$ is $\mathbf{x}_i$'s class label, we want to build a model $f(\mathbf{x})$ that can classify each test instance $\mathbf{x}$ to one of the m training or *seen* classes in $\mathcal{Y}$ or reject it to indicate that it does not belong to any of the m training or seen classes, i.e., *unseen*. In other words, we want to build a $(m+1)$-class classifier $f(\mathbf{x})$ with the classes $\mathcal{C} = \{l_1, l_2, \ldots, l_m, \textit{rejection}\}$.

There are some prior approaches for open classification. One-class SVM (Schölkopf et al., 2001; Tax and Duin, 2004) is the earliest approach. However, as no negative training data is used, one-class classifiers work poorly. Fei and Liu (2016) proposed a Center-Based Similarity (CBS) space learning method (Fei and Liu, 2015). This method first computes a center for each class and transforms each document to a vector of similarities to the center. A binary classifier is then built using the transformed data for each class. The decision surface is like a "ball" encircling each class. Everything outside the ball is considered not belonging to the class. Our proposed method outperforms this method greatly. Fei et al. (2016) further added the capability of incrementally or cumulatively learning new classes, which connects this work to *lifelong learning* (Chen and Liu, 2016) because without the ability to identify novel or new things and learn them, a system will never be able to learn by itself continually.

In computer vision, Scheirer et al. (2013) studied the problem of recognizing unseen images that the system was not trained for by reducing *open space risk*. The basic idea is that a classifier should not cover too much open space where there are few or no training data. They proposed to reduce the half-space of a binary SVM classifier

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2911–2916
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

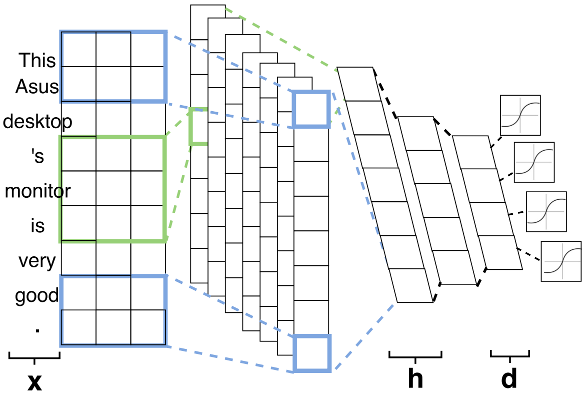


Figure 1: Overall Network of DOC

with a positive region bounded by two parallel hyperplanes. Similar works were also done in a probability setting by Scheirer et al. (2014) and Jain et al. (2014). Both approaches use probability threshold, but choosing thresholds need prior knowledge, which is a weakness of the methods. Dalvi et al. (2013) proposed a multi-class semi-supervised method based on the EM algorithm. It has been shown that these methods are poorer than the method in (Fei and Liu, 2016).

The work closest to ours is that in (Bendale and Boult, 2016), which leverages an algorithm called *OpenMax* to add the rejection capability by utilizing the logits that are trained via closed-world softmax function. One weak assumption of OpenMax is that examples with equally likely logits are more likely from the unseen or rejection class, which can be examples that are hard to classify. Another weakness is that it requires validation examples from the unseen/rejection class to tune the hyperparameters. Our method doesn't make these weak assumptions and performs markedly better.

Our proposed method, called DOC (*Deep Open Classification*), uses deep learning (Goodfellow et al., 2016; Kim, 2014). Unlike traditional classifiers, DOC builds a multi-class classifier with a 1-vs-rest final layer of sigmoids rather than softmax to reduce the open space risk. It reduces the open space risk further for rejection by tightening the decision boundaries of sigmoid functions with Gaussian fitting. Experimental results show that DOC dramatically outperforms state-of-the-art existing approaches from both text classification and image classification domains.

2 The Proposed DOC Architecture

DOC uses CNN (Collobert et al., 2011; Kim, 2014) as its base and augments it with a 1-vs-rest final sigmoid layer and Gaussian fitting for classification. Note: other existing deep models like RNN (Williams and Zipser, 1989; Schuster and Paliwal, 1997) and LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 2002) can also be adopted as the base. Similar to RNN, CNN also works on embedded sequential data (using 1D convolution on text instead of 2D convolution on images). We choose CNN because OpenMax uses CNN and CNN performs well on text (Kim, 2014), which enables a fairer comparison with OpenMax.

2.1 CNN and Feed Forward Layers of DOC

The proposed DOC system (given in Fig. 1) is a variant of the CNN architecture (Collobert et al., 2011) for text classification (Kim, 2014)[1]. The first layer embeds words in document $\mathbf{x}$ into dense vectors. The second layer performs convolution over dense vectors using different filters of varied sizes (see Sec. 3.4). Next, the max-over-time pooling layer selects the maximum values from the results of the convolution layer to form a k-dimension feature vector $\boldsymbol{h}$. Then we reduce $\boldsymbol{h}$ to a m-dimension vector $\boldsymbol{d} = d_{1:m}$ (m is the number of training/seen classes) via 2 fully connected layers and one intermediate ReLU activation layer:

$$\boldsymbol{d} = \boldsymbol{W}'(\text{ReLU}(\boldsymbol{W}\boldsymbol{h} + \boldsymbol{b})) + \boldsymbol{b}', \qquad (1)$$

where $\boldsymbol{W} \in \mathbb{R}^{r\times k}$, $\boldsymbol{b} \in \mathbb{R}^r$, $\boldsymbol{W}' \in \mathbb{R}^{m\times r}$, and $\boldsymbol{b}' \in \mathbb{R}^m$ are trainable weights; r is the output dimension of the first fully connected layer. The output layer of DOC is a 1-vs-rest layer applied to $d_{1:m}$, which allows rejection. We describe it next.

2.2 1-vs-Rest Layer of DOC

Traditional multi-class classifiers (Goodfellow et al., 2016; Bendale and Boult, 2016) typically use softmax as the final output layer, which does not have the rejection capability since the probability of prediction for each class is normalized across all training/seen classes. Instead, we build a 1-vs-rest layer containing m sigmoid functions for m seen classes. For the i-th sigmoid function corresponding to class l_i, DOC takes all examples with $y = l_i$ as positive examples and all the rest examples $y \neq l_i$ as negative examples.

The model is trained with the objective of summation of all log loss of the m sigmoid functions

[1] `https://github.com/alexander-rakhlin/CNN-for-Sentence-Classification-in-Keras`

on the training data D.

$$\text{Loss} = \sum_{i=1}^{m}\sum_{j=1}^{n} -\mathbb{I}(y_j = l_i)\log p(y_j = l_i) \\ -\mathbb{I}(y_j \neq l_i)\log(1 - p(y_j = l_i)), \qquad (2)$$

where $\mathbb{I}$ is the indicator function and $p(y_j = l_i) = \text{Sigmoid}(d_{j,i})$ is the probability output from ith sigmoid function on the jth document's ith-dimension of $\boldsymbol{d}$.

During testing, we reinterpret the prediction of m sigmoid functions to allow rejection, as shown in Eq. 3. For the i-th sigmoid function, we check if the predicted probability $\text{Sigmoid}(d_i)$ is less than a threshold t_i belonging to class l_i. If all predicted probabilities are less than their corresponding thresholds for an example, the example is rejected; otherwise, its predicted class is the one with the highest probability. Formally, we have

$$\hat{y} = \begin{cases} \textit{reject}, \text{ if } \text{Sigmoid}(d_i) < t_i, \forall l_i \in \mathcal{Y}; \\ \arg\max_{l_i \in \mathcal{Y}} \text{Sigmoid}(d_i), \text{ otherwise.} \end{cases} \qquad (3)$$

Note that although multi-label classification (Huang et al., 2013; Zhang and Zhou, 2006; Tsoumakas and Katakis, 2006) may also leverage multiple sigmoid functions, Eq. 3 forbids multiple predicted labels for the same example, which is allowed in multi-label classification. DOC is also related to multi-task learning (Huang et al., 2013; Caruana, 1998), where each label l_i is related to a 1-vs-rest binary classification task with shared representations from CNN and fully connected layers. However, Eq. 3 performs classification and rejection based on the outputs of these binary classification tasks.

Comparison with OpenMax: OpenMax builds on the traditional closed-world multi-class classifier (softmax layer). It reduces the open space for each seen class, which is weak for rejecting unseen classes. DOC's 1-vs-rest sigmoid layer provides a reasonable representation of all other classes (the rest of seen classes and unseen classes), and enables the 1 class forms a good boundary. Sec. 3.5 shows that this basic DOC is already much better than OpenMax. Below, we improve DOC further by tightening the decision boundaries more.

2.3 Reducing Open Space Risk Further

Sigmoid function usually uses the default probability threshold of $t_i = 0.5$ for classification of each class i. But this threshold does not consider potential open space risks from unseen (rejection) class data. We can improve the boundary by increasing t_i. We use Fig. 2 to illustrate. The x-axis represents d_i and y-axis is the predicted probability $p(y = l_i|d_i)$. The sigmoid function tries to push positive examples (belonging to the i-th class) and negative examples (belonging to the other seen classes) away from the y-axis via a high gain around $d_i = 0$, which serves as the default decision boundary for d_i with $t_i = 0.5$. As demonstrated by those 3 circles on the right-hand side of the y-axis, during testing, unseen class examples (circles) can easily fill in the gap between the y-axis and those dense positive (+) examples, which may reduce the recall of rejection and the precision of the i-th seen class prediction. Obviously, a better decision boundary is at $d_i = T$, where the decision boundary more closely "wrap" those dense positive examples with the probability threshold $t_i \gg 0.5$.

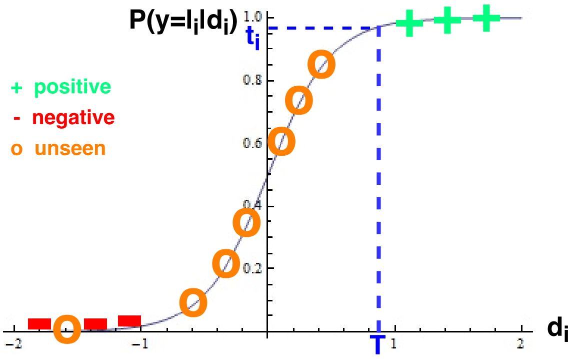


Figure 2: Open space risk of sigmoid function and desired decision boundary $d_i = T$ and probability threshold t_i.

To obtain a better t_i for each seen class i-th, we use the idea of outlier detection in statistics:

1. Assume the predicted probabilities $p(y = l_i|\mathbf{x}_j, y_j = l_i)$ of all training data of each class i follow one half of the Gaussian distribution (with mean $\mu_i = 1$), e.g., the three positive points in Fig. 2 projected to the y-axis (we don't need d_i). We then artificially create the other half of the Gaussian distributed points (≥ 1): for each existing point $p(y = l_i|\mathbf{x}_j, y_j = l_i)$, we create a mirror point $1 + (1 - p(y = l_i|\mathbf{x}_j, y_j = l_i)$ (not a probability) mirrored on the mean of 1.

2. Estimate the standard deviation σ_i using both the existing points and the created points.

3. In statistics, if a value/point is a certain number (α) of standard deviations away from the mean, it is considered an outlier. We thus set the probability threshold $t_i = \max(0.5, 1 - \alpha\sigma_i)$. The commonly used number for α is 3, which also works well in our experiments.

Note that due to Gaussian fitting, different class l_i can have a different classification threshold t_i.

3 Experimental Evaluation

3.1 Datasets

We perform evaluation using two publicly available datasets, which are exactly the same datasets used in (Fei and Liu, 2016).

(1) **20 Newsgroups**[2] (Rennie, 2008): The 20 newsgroups data set contains 20 non-overlapping classes. Each class has about 1000 documents.

(2) **50-class reviews** (Chen and Liu, 2014): The dataset has Amazon reviews of 50 classes of products. Each class has 1000 reviews. Although product reviews are used, we do not do sentiment classification. We still perform topic-based classification. That is, given a review, the system decides what class of product the review is about.

For every dataset, we keep a 20000 frequent word vocabulary. Each document is fixed to 2000-word length (cutting or padding when necessary).

3.2 Test Settings and Evaluation Metrics

For a fair comparison, we use exactly the same settings as in (Fei and Liu, 2016). For each class in each dataset, we randomly sampled 60% of documents for training, 10% for validation and 30% for testing. Fei and Liu (2016) did not use a validation set, but the test data is the same 30%. We use the validation set to avoid overfitting. For open-world evaluation, we hold out some classes (as unseen) in training and mix them back during testing. We vary the number of training classes and use 25%, 50%, 75%, or 100% classes for training and all classes for testing. Here using 100% classes for training is the same as the traditional closed-world classification. Taking 20 newsgroups as an example, for 25% classes, we use 5 classes (we randomly choose 5 classes from 20 classes for 10 times and average the results, as in (Fei and Liu, 2016)) for training and all 20 classes for testing (15 classes are unseen in training). We use macro F_1-score over $5 + 1$ classes (1 for rejection) for evaluation. Please note that examples from unseen classes are dropped in the validation set.

Table 1: Macro-F_1 scores for 20 newsgroups

% of seen classes	**25%**	**50%**	**75%**	**100%**
cbsSVM	59.3	70.1	72.0	85.2
OpenMax	35.7	59.9	76.2	91.9
DOC ($t = 0.5$)	75.9	84.0	87.4	92.6
DOC	82.3	85.2	86.2	92.6

Table 2: Macro-F_1 scores for 50-class reviews

% of seen classes	**25%**	**50%**	**75%**	**100%**
cbsSVM	55.7	61.5	58.6	63.4
OpenMax	41.6	57.0	64.2	69.2
DOC ($t = 0.5$)	51.1	63.6	66.2	69.8
DOC	61.2	64.8	66.6	69.8

3.3 Baselines

We compare DOC with two state-of-the-art methods published in 2016 and one DOC variant.

cbsSVM: This is the latest method published in NLP (Fei and Liu, 2016). It uses SVM to build 1-vs-rest CBS classifiers for multiclass text classification with rejection option. The results of this system are taken from (Fei and Liu, 2016).

OpenMax: This is the latest method from computer vision (Bendale and Boult, 2016). Since it is a CNN-based method for image classification, we adapt it for text classification by using CNN with a softmax output layer, and adopt the OpenMax layer[3] for open text classification. When all classes are seen (100%), the result from softmax is reported since OpenMax layer always performs rejection. We use default hyperparameter values of OpenMax (Weibull tail size is set to 20).

DOC($t = 0.5$): This is the basic DOC ($t = 0.5$). Gaussian fitting isn't used to choose each t_i.

Note that (Fei and Liu, 2016) compared with several other baselines. We don't compare with them as it was shown that cbsSVM was superior.

3.4 Hyperparameter Setting

We use word vectors pre-trained from Google News[4] (3 million words and 300 dimensions). For the CNN layers, 3 filter sizes are used $[3, 4, 5]$. For each filter size, 150 filters are applied. The dimension r of the first fully connected layer is 250.

[2] http://qwone.com/~jason/20Newsgroups/

[3] https://github.com/abhijitbendale/OSDN

[4] https://code.google.com/archive/p/word2vec/

3.5 Result Analysis

The results of 20 newsgroups and 50-class reviews are given in Tables 1 and 2, respectively. From the tables, we can make the following observations:

1. DOC is markedly better than OpenMax and cbsSVM in macro-F_1 scores for both datasets in the 25%, 50%, and 75% settings. For the 25% and 50% settings (most test examples are from unseen classes), DOC is dramatically better. Even for 100% of traditional closed-world classification, it is consistently better too. DOC($t = 0.5$) is better too.

2. For the 25% and 50% settings, DOC is also markedly better than DOC($t = 0.5$), which shows that Gaussian fitting finds a better probability threshold than $t = 0.5$ when many unseen classes are present. In the 75% setting (most test examples are from seen classes), DOC($t = 0.5$) is slightly better for 20 newsgroups but worse for 50-class reviews. DOC sacrifices some recall of seen class examples for better precision, while $t = 0.5$ sacrifices the precision of seen classes for better recall. DOC($t = 0.5$) is also worse than cbsSVM for 25% setting for 50-class reviews. It is thus not as robust as DOC.

3. For the 25% and 50% settings, cbsSVM is also markedly better than OpenMax.

4 Conclusion

This paper proposed a novel deep learning based method, called DOC, for open text classification. Using the same text datasets and experiment settings, we showed that DOC performs dramatically better than the state-of-the-art methods from both the text and image classification domains. We also believe that DOC is applicable to images.

In our future work, we plan to improve the cumulative or incremental learning method in (Fei et al., 2016) to learn new classes without training on all past and new classes of data from scratch. This will enable the system to learn by self to achieve continual or lifelong learning (Chen and Liu, 2016). We also plan to improve model performance during testing (Shu et al., 2017).

Acknowledgments

This work was supported in part by grants from National Science Foundation (NSF) under grant no. IIS-1407927 and IIS-1650900.

References

Abhijit Bendale and Terrance E Boult. 2016. Towards open set deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 1563–1572.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, Springer, pages 95–133.

Zhiyuan Chen and Bing Liu. 2014. Mining topics in documents: standing on the shoulders of big data. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 1116–1125.

Zhiyuan Chen and Bing Liu. 2016. *Lifelong Machine Learning*. Morgan & Claypool Publishers.

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

Bhavana Dalvi, William W Cohen, and Jamie Callan. 2013. Exploratory learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pages 128–143.

Geli Fei and Bing Liu. 2015. Social media text classification under negative covariate shift. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2015)*.

Geli Fei and Bing Liu. 2016. Breaking the closed world assumption in text classification. In *Proceedings of NAACL-HLT*. pages 506–514.

Geli Fei, Shuai Wang, and Bing Liu. 2016. Learning cumulatively to become more knowledgeable. In *Proceedings of SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2016)*.

Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. 2002. Learning precise timing with lstm recurrent networks. *Journal of machine learning research* 3(Aug):115–143.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

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

Yan Huang, Wei Wang, Liang Wang, and Tieniu Tan. 2013. Multi-task deep neural network for multi-label learning. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*. IEEE, pages 2897–2900.

Lalit P Jain, Walter J Scheirer, and Terrance E Boult. 2014. Multi-class open set recognition using probability of inclusion. In *European Conference on Computer Vision*. Springer, pages 393–409.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Jason Rennie. 2008. 20 newsgroup dataset.

Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. 2013. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(7):1757–1772.

Walter J Scheirer, Lalit P Jain, and Terrance E Boult. 2014. Probability models for open set recognition. *IEEE transactions on pattern analysis and machine intelligence* 36(11):2317–2324.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural computation* 13(7):1443–1471.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Lei Shu, Hu Xu, and Bing Liu. 2017. Lifelong learning crf for supervised aspect extraction. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-2017, short paper)*.

David MJ Tax and Robert PW Duin. 2004. Support vector data description. *Machine learning* 54(1):45–66.

Grigorios Tsoumakas and Ioannis Katakis. 2006. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining* 3(3).

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2):270–280.

Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering* 18(10):1338–1351.

CharManteau: Character Embedding Models For Portmanteau Creation

Varun Gangal [*], **Harsh Jhamtani** [*], **Graham Neubig, Eduard Hovy, Eric Nyberg**
Language Technologies Institute,
Carnegie Mellon University
{vgangal,jharsh,gneubig,hovy,ehn}@cs.cmu.edu

Abstract

Portmanteaus are a word formation phenomenon where two words are combined to form a new word. We propose character-level neural sequence-to-sequence (S2S) methods for the task of portmanteau generation that are end-to-end-trainable, language independent, and do not explicitly use additional phonetic information. We propose a noisy-channel-style model, which allows for the incorporation of unsupervised word lists, improving performance over a standard source-to-target model. This model is made possible by an exhaustive candidate generation strategy specifically enabled by the features of the portmanteau task. Experiments find our approach superior to a state-of-the-art FST-based baseline with respect to ground truth accuracy and human evaluation.

1 Introduction

Portmanteaus (or lexical blends Algeo (1977)) are novel words formed from parts of multiple root words in order to refer to a new concept which can't otherwise be expressed concisely. Portmanteaus have become frequent in modern-day social media, news reports and advertising, one popular example being *Brexit* (Britain + Exit). Petri (2012). These are found not only in English but many other languages such as Bahasa Indonesia Dardjowidjojo (1979), Modern Hebrew Bat-El (1996); Berman (1989) and Spanish Piñeros (2004). Their short length makes them ideal for headlines and brandnames (Gabler, 2015). Unlike better-defined morphological phenomenon such as inflection and derivation, portmanteau generation is difficult to capture using a set of rules. For instance, Shaw et al. (2014) state that the composition of the portmanteau from its root words depends on several factors, two important ones being maintaining prosody and retaining character segments from the root words, especially the head. An existing work by Deri and Knight (2015) aims to solve the problem of predicting portmanteau using a multi-tape FST model, which is data-driven, unlike prior approaches. Their methods rely on a grapheme to phoneme converter, which takes into account the phonetic features of the language, but may not be available or accurate for non-dictionary words, or low resource languages.

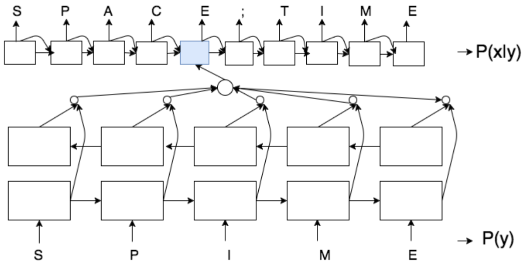


Figure 1: A sketch of our Backward, noisy-channel model. The attentional S2S model with bidirectional encoder gives $P(x|y)$ and next-character model gives $P(y)$, where y (*spime*) is the portmanteau and $\boldsymbol{x} = \text{concat}(\boldsymbol{x}^{(1)}, \text{“;”}, \boldsymbol{x}^{(2)})$ are the concatenated root words (*space* and *time*).

Prior works, such as Faruqui et al. (2016), have demonstrated the efficacy of neural approaches for morphological tasks such as inflection. We hypothesize that such neural methods can (1) provide a simpler and more integrated end-to-end framework than multiple FSTs used in the previous work, and (2) automatically capture features such as phonetic similarity through the use of character embeddings, removing the need for explicit

[*] denotes equal contribution

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2917–2922
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

grapheme-to-phoneme prediction. To test these hypotheses, in this paper, we propose a neural S2S model to predict portmanteaus given the two root words, specifically making 3 major contributions:

- We propose an S2S model that attends to the two input words to generate portmanteaus, and an additional improvement that leverages noisy-channel-style modelling to incorporate a language model over the vocabulary of words (§2).
- Instead of using the model to directly predict output character-by-character, we use the features of portmanteaus to exhaustively generate candidates, making scoring using the noisy channel model possible (§3).
- We curate and share a new and larger dataset of 1624 portmanteaus (§4).

In experiments (§5), our model performs better than the baseline Deri and Knight (2015) on both objective and subjective measures, demonstrating that such methods can be used effectively in a morphological task.

2 Proposed Models

This section describes our neural models.

2.1 Forward Architecture

Under our first proposed architecture, the input sequence $\boldsymbol{x} = \text{concat}(\boldsymbol{x}^{(1)}, \text{“;”}, \boldsymbol{x}^{(2)})$, while the output sequence is the portmanteau $\boldsymbol{y}$. The model learns the distribution $P(\boldsymbol{y}|\boldsymbol{x})$.

The network architecture we use is an attentional S2S model (Bahdanau et al., 2014). We use a bidirectional encoder, which is known to work well for S2S problems with similar token order, which is true in our case. Let $\overrightarrow{LSTM}$ and $\overleftarrow{LSTM}$ represent the forward and reverse encoder; $e_{enc}()$ and $e_{dec}()$ represent the character embedding functions used by encoder and decoder The following equations describe the model:

$$
\begin{aligned}
h_0^{\overrightarrow{enc}} &= \overrightarrow{0}, h_{|x|}^{\overleftarrow{enc}} = \overrightarrow{0} \\
h_t^{\overrightarrow{enc}} &= \overrightarrow{LSTM}(h_{t-1}^{enc}, e_{enc}(x_t)) \\
h_t^{\overleftarrow{enc}} &= \overleftarrow{LSTM}(h_{t+1}^{enc}, e_{enc}(x_t)) \\
h_t^{enc} &= h_t^{\overrightarrow{enc}} + h_t^{\overleftarrow{enc}} \\
h_0^{dec} &= h_{|x|}^{enc} \\
h_t^{dec} &= LSTM(h_{t-1}^{dec}, [\text{concat}(e_{dec}(y_{t-1}), c_{t-1})]) \\
p_t &= \text{softmax}(W_{hs}[\text{concat}(h_t^{dec}, c_t)] + b_s)
\end{aligned}
$$

The context vector c_t is computed using dot-product attention over encoder states. We choose dot-product attention because it doesn't add extra parameters, which is important in a low-data scenario such as portmanteau generation.

$$
\begin{aligned}
a_i^t &= dot(h_t^{dec}, h_i^{enc}), \alpha^t = \text{softmax}(a^t) \\
c_t &= \sum_{i=1}^{i=|x|} \alpha_i^t h_i^{enc}
\end{aligned}
$$

In addition to capturing the fact that portmanteaus of two English words typically sound *English-like*, and to compensate for the fact that available portmanteau data will be small, we pretrain the character embeddings on English language words. We use character embeddings learnt using an LSTM language model over words in an English dictionary,[1] where each word is a sequence of characters, and the model will predict next character in sequence conditioned on previous characters in the sequence.

2.2 Backward Architecture

The second proposed model uses Bayes's rule to reverse the probabilities $P(\boldsymbol{y}|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y})}{P(\boldsymbol{x})}$ to get $\text{argmax}_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}) = \text{argmax}_{\boldsymbol{y}} P(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y})$. Thus, we have a reverse model of the probability $P(\boldsymbol{x}|\boldsymbol{y})$ that the given root words were generated from the portmanteau and a character language model model $P(\boldsymbol{y})$. This is a probability distribution over all character sequences $y \in A^*$, where A is the alphabet of the language. This way of factorizing the probability is also known as a *noisy channel model*, which has recently also been shown to be effective for neural MT (Hoang et al. (2017), Yu et al. (2016)). Such a model offers two advantages

1. The reverse direction model (or alignment model) gives higher probability to those portmanteaus from which one can discern the root words easily, which is one feature of good portmanteaus.
2. The character language model $P(\boldsymbol{y})$ can be trained on a large vocabulary of words in the language. The likelihood of a word y is factorized as $P(y) = \Pi_{i=1}^{i=|y|} P(y_i|y_1^{i-1})$, where $y_j^i = y_i, y_{i+1} \dots y_j$, and we train a LSTM to maximize this likelihood.

[1] Specifically in our experiments, 134K words from the CMU dictionary (Weide, 1998).

3 Making Predictions

Given these models, we must make predictions, which we do by two methods

Greedy Decoding: In most neural sequence-to-sequence models, we perform auto-regressive greedy decoding, selecting the next character greedily based on the probability distribution for the next character at current time step. We refer to this decoding strategy as GREEDY.

Exhaustive Generation: Many portmanteaus were observed to be concatenation of a prefix of the first word and a suffix of the second. We therefore generate all candidate outputs which follow this rule. Thereafter we score these candidates with the decoder and output the one with the maximum score. We refer to this decoding strategy as SCORE.

Given that our training data is small in size, we expect ensembling (Breiman, 1996) to help reduce model variance and improve performance. In this paper, we ensemble our models wherever mentioned by training multiple models on 80% subsamples of the training data, and averaging log probability scores across the ensemble at test-time.

4 Dataset

The existing dataset by Deri and Knight (2015) contains 401 portmanteau examples from Wikipedia. We refer to this dataset as D_{Wiki}. Besides being small for detailed evaluation, D_{Wiki} is biased by being from just one source. We manually collect D_{Large}, a dataset of 1624 distinct English portmanteaus from following sources:

- Urban Dictionary[2]
- Wikipedia
- Wiktionary
- BCU's Neologism Lists from '94 to '12.

Naturally, $D_{Wiki} \subset D_{Large}$. We define $D_{Blind} = D_{Large} - D_{Wiki}$ as the dataset of 1223 examples not from Wikipedia. We observed that 84.7% of the words in D_{Large} can be generated by concatenating prefix of first word with a suffix of the second.

Model	Attn	Ens	Init	Search	Matches	Distance
BASELINE	-	-	-	-	**45.39%**	1.59
FORWARD	✓	×	×	GREEDY	22.00%	1.98
	✓	×	✓	GREEDY	28.00%	1.90
	✓	×	×	BEAM	13.25%	2.47
	✓	×	✓	BEAM	15.25%	2.37
	✓	×	×	SCORE	30.25%	1.64
	✓	×	✓	SCORE	32.88%	1.53
	✓	✓	✓	SCORE	42.25%	1.33
	✓	✓	×	SCORE	41.25%	1.34
	×	×	✓	SCORE	6.75%	3.78
	×	×	×	SCORE	6.50%	3.76
BACKWARD	✓	×	×	SCORE	37.00%	1.53
	✓	×	✓	SCORE	42.25%	1.35
	✓	✓	✓	SCORE	**48.75%**	**1.12**
	✓	✓	×	SCORE	**46.50%**	**1.24**
	×	×	✓	SCORE	5.00%	3.95
	×	×	×	SCORE	4.75%	3.98

Table 1: 10-Fold Cross-Validation results, D_{Wiki}. *Attn, Ens, Init* denote attention, ensembling, and initializing character embeddings respectively.

5 Experiments

In this section, we show results comparing various configurations of our model to the baseline FST model of Deri and Knight (2015) (BASELINE). Models are evaluated using exact-matches (*Matches*) and average Levenshtein edit-distance (*Distance*) w.r.t ground truth.

5.1 Objective Evaluation Results

In *Experiment 1*, we follow the same setup as Deri and Knight (2015). D_{Wiki} is split into 10 folds. Each fold model uses 8 folds for training, 1 for validation, and 1 for test. The average (10 fold cross-validation style approach) performance metrics on the test fold are then evaluated. *Table 1* shows the results of *Experiment 1* for various model configurations. We get the BASELINE numbers from Deri and Knight (2015). Our best model obtains 48.75% *Matches* and 1.12 *Distance*, compared to 45.39% *Matches* and 1.59 *Distance* using BASELINE.

For *Experiment 2*, we seek to compare our best approaches from *Experiment 1* to the BASELINE on a large, held-out dataset. Each model is trained on D_{Wiki} and tested on D_{Blind}. BASELINE was similarly trained only on D_{Wiki} , making it a fair comparison. Table 2 shows the results[3]. Our best model gets *Distance* of 1.96 as compared to 2.32 from BASELINE.

We observe that the *Backward* architecture performs better than *Forward* architecture, confirming our hypothesis in §2.2. In addition, ablation results confirm the importance of attention, and

[2] Not all neologisms are portmanteaus, so we manually choose those which are for our dataset.

[3] For BASELINE (Deri and Knight, 2015), we use their trained model from `http://leps.isi.edu/fst/step-all.php`

Model	Attn	Ens	Init	Search	Matches	Distance
BASELINE	-	-	-	-	**31.56%**	2.32
FORWARD	✓	×	✓	SCORE	25.26%	2.13
	✓	×	×	SCORE	24.93%	2.32
	✓	✓	×	SCORE	31.23%	1.98
	✓	✓	✓	SCORE	28.94%	2.04
BACKWARD	✓	×	✓	SCORE	25.75%	2.14
	✓	×	×	SCORE	25.26%	2.17
	✓	✓	×	SCORE	**31.72%**	**1.96**
	✓	✓	✓	SCORE	**32.78%**	**1.96**

Table 2: Results on D_{Blind} (1223 Examples). In general, BACKWARD architecture performs better than FORWARD architecture.

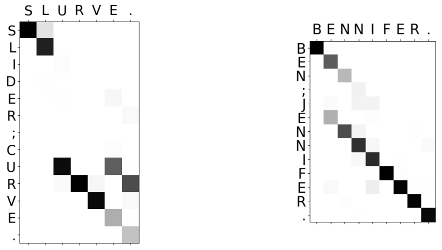


Figure 2: Attention matrices while generating *slurve* from *slider;curve*, and *bennifer* from *ben;jennifer* respectively, using *Forward* model. **;** and **.** are separator and stop characters. Darker cells are higher-valued

initializing the word embeddings. We believe this is because portmanteaus have high fidelity towards their root word characters and its critical that the model can observe all root sequence characters, which attention manages to do as shown in Fig. 2.

5.1.1 Performance on Uncovered Examples

The set of candidates generated before scoring in the approximate SCORE decoding approach sometimes do not cover the ground truth. This holds true for 229 out of 1223 examples in D_{Blind}. We compare the FORWARD approach along with a GREEDY decoding strategy to the BASELINE approach for these examples.

Both FORWARD+GREEDY and the BASELINE get 0 *Matches* on these examples. The *Distance* for these examples is 4.52 for BASELINE and 4.09 for FORWARD+GREEDY. Hence, we see that one of our approaches (FORWARD+GREEDY) outperforms BASELINE even for these examples.

5.2 Significance Tests

Since our dataset is still small relatively small (1223 examples), it is essential to verify whether BACKWARD is indeed statistically significantly better than BASELINE in terms of *Matches*.

Input	FORWARD	BACKWARD	GROUND TRUTH
shopping;marathon	shopparathon	shoathon	shopathon
fashion;fascism	fashism	fashism	fashism
wiki;etiquette	wikiquette	wiquette	wikiquette
clown;president	clowident	clownsident	clownsident

Table 3: Example outputs from different models (Refer to appendix for more examples)

Judgement	Percentage of total
Much Better (1)	29.06
Better (2)	29.06
Worse (3)	25.11
Much Worse (4)	16.74

Table 4: AMT annotator judgements on whether our system's proposed portmanteau is better or worse compared to the baseline

In order to do this, we use a paired bootstrap[4] comparison (Koehn, 2004) between BACKWARD and BASELINE in terms of *Matches*. BACKWARD is found to be better (gets more *Matches*) than BASELINE in 99.9% ($p = 0.999$) of the subsets.

Similarly, BACKWARD has a lower *Distance* than BASELINE by a margin of 0.2 in 99.5% ($p = 0.995$) of the subsets.

5.3 Subjective Evaluation and Analysis

On inspecting outputs, we observed that often output from our system seemed good in spite of high edit distance from ground truth. Such aspect of an output *seeming good* is not captured satisfactorily by measures like edit distance. To compare the errors made by our model to the baseline, we designed and conducted a human evaluation task on AMT.[5] In the survey, we show human annotators outputs from our system and that of the baseline. We ask them to judge which alternative is *better* overall based on following criteria: 1. It is a good shorthand for two original words 2. It sounds better. We requested annotation on a scale of 1-4. To avoid ordering bias, we shuffled the order of two portmanteau between our system and that of baseline. We restrict annotators to be from Anglophone countries, have HIT Approval Rate > 80% and pay 0.40$ per HIT (5 Questions per HIT).

As seen in Table 4, output from our system was labelled better by humans as compared to the baseline 58.12% of the time. Table 3 shows outputs from different models for a few examples.

[4] We average across $M = 1000$ randomly chosen subsets of D_{Blind}, each of size $N = 611$ ($\approx 1223/2$)

[5] We avoid ground truth comparison because annotators can be biased to ground truth due to its existing popularity.

6 Related Work

Özbal and Strapparava (2012) generate new words to describe a product given its category and properties. However, their method is limited to handcrafted rules as compared to our data driven approach. Also, their focus is on brand names. Hiranandani et al. (2017) have proposed an approach to recommend brand names based on brand/product description. However, they consider only a limited number of features like memorability and readability. Smith et al. (2014) devise an approach to generate portmanteaus, which requires user-defined weights for attributes like *sounding good*. Generating a portmanteau from two root words can be viewed as a S2S problem. Recently, neural approaches have been used for S2S problems (Sutskever et al., 2014) such as MT. Ling et al. (2015) and Chung et al. (2016) have shown that character-level neural sequence models work as well as word-level ones for language modelling and MT. Zoph and Knight (2016) propose S2S models for multi-source MT, which have multi-sequence inputs, similar to our case.

7 Conclusion

We have proposed an end-to-end neural system to model portmanteau generation. Our experiments show the efficacy of proposed system in predicting portmanteaus given the root words. We conclude that pre-training character embeddings on the English vocabulary helps the model. Through human evaluation we show that our model's predictions are superior to the baseline. We have also released our dataset and code[6] to encourage further research on the phenomenon of portmanteaus. We also release an online demo [7] where our trained model can be queried for portmanteau suggestions. An obvious extension to our work is to try similar models on multiple languages.

Acknowledgements

We thank Dongyeop Kang, David Mortensen, Qinlan Shen and anonymous reviewers for their valuable comments. This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the DEFT program.

[6] https://github.com/vgtomahawk/Charmanteau-CamReady

[7] http://tinyurl.com/y9x6mvy

References

John Algeo. 1977. Blends, a structural and systemic view. *American speech*, 52(1/2):47–64.

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

Outi Bat-El. 1996. Selecting the best of the worst: the grammar of Hebrew blends. *Phonology*, 13(03):283–328.

Ruth Berman. 1989. The role of blends in Modern Hebrew word-formation. *Studia linguistica et orientalia memoriae Haim Blanc dedicata. Wiesbaden: Harrassowitz*, pages 45–61.

Leo Breiman. 1996. Bagging predictors. *Machine learning*, 24(2):123–140.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv:1603.06147*.

Soenjono Dardjowidjojo. 1979. Acronymic Patterns in Indonesian. *Pacific Linguistics Series C*, 45:143–160.

Aliya Deri and Kevin Knight. 2015. How to make a frenemy: Multitape FSTs for portmanteau generation. In *Proceedings of NAACL-HLT*, pages 206–210.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological Inflection Generation using Character Sequence to Sequence Learning. In *Proceedings of NAACL-HLT*, pages 634–643.

Neal Gabler. 2015. The Weird Science of Naming New Products. *New York Times - http://tinyurl.com/lmlq7ex*.

Gaurush Hiranandani, Pranav Maneriker, and Harsh Jhamtani. 2017. Generating appealing brand names. *arXiv preprint arXiv:1706.09335*.

Cong Duy Vu Hoang, Gholamreza Haffari, and Trevor Cohn. 2017. Decoding as Continuous Optimization in Neural Machine Translation. *arXiv:1701.02854*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*, pages 388–395.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *arXiv:1511.04586*.

Gözde Özbal and Carlo Strapparava. 2012. A computational approach to the automation of creative naming. In *Proceedings of ACL*, pages 703–711. Association for Computational Linguistics.

Alexandra Petri. 2012. Say No to Portmanteaus. *Washington Post* - `http://tinyurl.com/kvmep2t`.

Carlos-Eduardo Piñeros. 2004. The creation of portmanteaus in the extragrammatical morphology of Spanish. *Probus*, 16(2):203–240.

Katherine E Shaw, Andrew M White, Elliott Moreton, and Fabian Monrose. 2014. Emergent faithfulness to morphological and semantic heads in lexical blends. In *Proceedings of the Annual Meetings on Phonology*, volume 1.

Michael R Smith, Ryan S Hintze, and Dan Ventura. 2014. Nehovah: A neologism creator nomen ipsum. In *Proceedings of the International Conference on Computational Creativity*, pages 173–181.

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

R Weide. 1998. The CMU pronunciation dictionary, release 0.6. *Carnegie Mellon University*.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomas Kocisky. 2016. The Neural Noisy Channel. *arXiv:1611.02554*.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. *arXiv:1601.00710*.

Using Automated Metaphor Identification to Aid in Detection and Prediction of First-Episode Schizophrenia

E. Darío Gutiérrez[1] **Philip R. Corlett**[2] **Cheryl M. Corcoran**[3] **Guillermo A. Cecchi**[1]
[1] IBM Research
[2] Yale University
[3] New York State Psychiatric Institute at Columbia University
edg@icsi.berkeley.edu philip.corlett@yale.edu
cheryl.corcoran@nyspi.columbia.edu guillermo.cecchi@ibm.com

Abstract

The diagnosis of serious mental health conditions such as schizophrenia is based on the judgment of clinicians whose training takes many years and cannot be easily formalized into objective measures. However, clinical research suggests there are disturbances in aspects of the language use of patients with schizophrenia, which opens a door for the use of NLP tools in schizophrenia diagnosis and prognosis. Using metaphor-identification and sentiment-analysis algorithms to automatically generate features, we create a classifier that, with high accuracy, can predict which patients will develop (or currently suffer from) schizophrenia. To our knowledge, this study is the first to demonstrate the utility of automated metaphor identification algorithms for detection or prediction of disease.

1 Introduction

Schizophrenia is a severe mental disorder that has a devastating impact on those who suffer from it, as well as on their families and communities. Schizophrenia is characterized by psychotic behaviors (hallucinations, delusions, thought disorders, movement disorders), flat affect and anhedonia, and trouble with focusing and executive functioning, among other symptoms (American Psychiatric Association, 2013). It afflicts over 21 million people worldwide, and is associated with a 100-150 percent increase in early mortality (Goff et al., 2005; World Health Organization, 2016; Simeone et al., 2015). As a result, diagnosis and treatment of schizophrenia has important public health consequences. Unfortunately, practitioners who are qualified to diagnose and treat serious mental health issues such as schizophrenia are in chronically short supply, and their accumulated knowledge cannot be easily formalized into reproducible metrics (Patel et al., 2007).

However, clinical research into the symptoms and mechanisms of schizophrenia suggests that disturbances in language use, and especially in metaphor use and affect, characterize schizophrenia. This suggests that automated NLP methods may have the potential to help in diagnosis and prognosis of schizophrenia. In this paper, we work from open-ended transcripts of patients interviewed by non-specialists. We then apply NLP algorithms for metaphor identification and sentiment analysis to automatically generate features for a classifier that, with high accuracy, can predict which patients will develop schizophrenia and which patients would currently be diagnosed with schizophrenia by psychiatrists.

2 Background & Related work

2.1 NLP and Computational Psychiatry

Several recent studies have proven that NLP text-analysis techniques can be successfully applied to predict mental illness. Vincze et al. (2016) use linguistic and demographic features to predict whether a speech transcript was produced by an individual with mild cognitive impairment or by a healthy control. To our knowledge, Elvevåg et al. (2007) were the first to use automated NLP methods to predict whether or not patients suffered from schizophrenia. The technical specifics of their method are unclear, as the paper was intended for a clinical audience, but they use a k-nearest neighbors algorithm in a feature space made up of n-gram features and distributional semantic features to classify 26 schizophrenia patients and 25 healthy controls. They achieve classification accuracy of 78.4% on this task. Mota et al. (2012) employ a graph-based method to classify transcripts taken from interviews with eight patients with

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2923–2930
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

schizophrenia, eight healthy controls, and eight manic patients, achieving both precision and recall of 0.875. Bedi et al. (2015) apply semantic coherence measures and measures based on part-of-speech tags to predict whether 34 youths at risk of psychosis would have a psychotic episode within 2.5 years of being interviewed (five of whom did transition within the study period). They correctly classify 100% of participants.

2.2 Metaphor, Affect and Schizophrenia

Mental-health clinicians have long had intuitions that schizophrenia patients differ from healthy individuals in their use of metaphor. A survey by Kuperberg (2010) of over 50 years of observations in the schizophrenia literature concludes that schizophrenia patients "may use common words in an idiosyncratic or bizarre manner." Particularly colorful (and metaphorical) examples of bizarre speech recorded by Andreasen (1986) include patients who referred to watches as "time vessels" and to gloves as "hand shoes."

Billow et al. (1997) carried out the first experimental exploration of this phenomenon. They measure the metaphor production of patients with schizophrenia and healthy controls during free responses to a structured interview. They find that patients with schizophrenia produce comparable rates of felicitous, coherent metaphors as healthy controls, but produce deviant metaphorical speech with significantly greater frequency.

It is not clear what could account for these differences in metaphor production, but neuroscientific studies of patients with schizophrenia offer some clues. Research shows that schizophrenia is associated with dysfunction of the amygdala, a brain structure responsible for regulating emotion (Rasetti et al., 2009). Other work demonstrates impairments in emotion perception and production in patients with schizophrenia (Vaskinn et al., 2008) and even demonstrates that face emotion recognition deficits are a predictor of psychosis onset (Corcoran et al., 2015). Based on these findings, and recognizing the important role that metaphor plays in emotional language (see (Kövecses, 2003)), Elvevåg et al. (2011) hypothesize that metaphor production disturbances in patients with schizophrenia are deeply tied to "emotional" language (i.e., language with high affective polarity). However, it should be noted in this regard that most work on metaphor processing has focused on cortical regions involved (Chen et al., 2008; Schmidt et al., 2010; Benedek et al., 2014).

2.3 Sentiment Analysis & Metaphor Detection Algorithms

Sentiment analysis is a natural-language processing task that involves determining, for given text, whether the text conveys a positive or negative sentiment, and how positive or negative the sentiment is. The book by Liu (2015) gives a comprehensive overview of sentiment analysis.

Metaphor detection is the task of determining whether a given word, phrase, or passage is being used metaphorically or literally. It is an emerging field in NLP, with research still in relatively early stages. A variety of different machine-learning and statistical methods have been applied to the task, including clustering (Birke and Sarkar, 2006; Shutova et al., 2010; Li and Sporleder, 2010; Shutova and Sun, 2013); topic models (Bethard et al., 2009; Li et al., 2010; Heintz et al., 2013); topical structure and imageability analysis (Strzalkowski et al., 2013); semantic similarity graphs (Sporleder and Li, 2009), and feature-based classifiers (Gedigian et al., 2006; Li and Sporleder, 2009; Turney et al., 2011; Dunn, 2013a,b; Hovy et al., 2013; Mohler et al., 2013; Neuman et al., 2013; Tsvetkov et al., 2013, 2014; Klebanov et al.). Metaphor detection methods differ in how they define the task of metaphor detection–for instance, some algorithms seek to determine whether a phrase (such as *sweet victory*) is metaphorical (Krishnakumaran and Zhu, 2007; Turney et al., 2011; Tsvetkov et al., 2014; Bracewell et al., 2014; Gutiérrez et al., 2016), while others attempt to tag metaphoricity at the level of the utterance (Dunn, 2013a), or at the level of individual tokens in running text (Klebanov et al.; Schulder and Hovy, 2014; Do Dinh and Gurevych, 2016). For a recent review, see Shutova (2015). For our purposes, we decided that token-level metaphor detection offered the most appropriate level of granularity, and we chose the algorithm of (Do Dinh and Gurevych, 2016) because of its state-of-the-art performance at this task at the time we began this project.

3 Data

3.1 First-Episode Schizophrenia Transcripts

Our main data set[1] consists of interviews with 17 patients who have suffered a first episode of schizophrenia (denoted by 1EP+) and 15 healthy

[1] Patient data are confidential and can only be used via a Data-Sharing Agreement with authors Corcoran and Corlett; please contact these authors for more information.

controls (denoted by 1EP-). Healthy controls were obtained from the same source population as patients with schizophrenia in the metropolitan New York City region, using web-based advertising on Craigslist, as well as by posting of flyers in and around the region. Participants engaged in open-ended interviews lasting approximately one hour, during which they were encouraged to express themselves narratively. Participants were queried on four topics, for which interviewers provided clarifying questions if they were not spontaneously discussed. The four discussion topics, as well as details of interviewer training and participant selection criteria, are discussed in more detail in the supplementary materials as well as in (Ben-David et al., 2014). Independent transcribers transcribed the interviews. Participants were matched for socioeconomic characteristics and education level. The average age of the 1EP- cohort was 35, and the average age of the 1EP+ cohort was 39. However, the 1EP- cohort was 47% male, while the 1EP+ cohort was 76% male. We refer to this data set as 1EP.

3.2 Prodromal Psychosis Transcripts

We use the data set introduced by Bedi et al. (2015) of transcripts from 34 youths at clinical high risk (CHR) for psychosis, based on the Structured Interview for Prodromal Syndromes (Miller et al., 2003). Demographic details are provided in Bedi et al. (2015). There were no significant differences for age, gender, ethnicity or medication usage between CHR converters vs. CHR non-converters. Notably, all CHR participants were ascertained using gold-standard clinical measures for which the researchers obtained excellent inter-rater reliability with other CHR programs in North America. Open-ended baseline interviews were collected from the participants using the same protocol as above. Participants were then assessed quarterly for 2.5 years to determine whether they had transitioned to psychosis. Five of the participants suffered a first episode of psychosis within the assessment period (denoted by CHR+); the remainder did not (denoted by CHR-).

4 Experiments

The review of the literature in §2.2 suggests that a constellation of disturbances in metaphor use and extremeness/lability of sentiment may characterize schizophrenia. In order to assess whether these phenomena can truly distinguish patients with schizophrenia from healthy controls or to predict future schizophrenic episodes, we produce five features. Four of these features are derived from sentiment scores produced by a sentiment analysis algorithm, and one is derived from metaphor tags produced by a metaphor identification algorithm.

4.1 Feature Set

Metaphoricity We hope to detect the alteration in metaphor production observed in patients with schizophrenia by Billow et al. (1997) using an automated metaphor detection algorithm that tag word tokens as metaphorical or not. We adapt the token-level metaphor identification algorithm of Do Dinh and Gurevych (2016) to our task. In particular, we use a multilayer perceptron (MLP) architecture with three layers. The input layer is comprised of the concatenation of the word embeddings for each token and the two tokens before and after (not including non-content tokens, and padded with a randomly created embedding at sentence beginnings and endings). The vector for each token is composed of the word's 300-dimensional Word2Vec skip-gram negative sampling word embedding [2], concatenated with a one-hot binary vector that indicates the token's part of speech. The hidden layer has ten fully connected hidden units with the hyperbolic tangent activation function. The output node classifies a token as literal or metaphorical using the softmax activation function.

Training is accomplished by minimizing a cross-entropy objective using stochastic gradient descent; the learning rate is decremented linearly during each epoch, for a maximum of 100 epochs. As in Do Dinh and Gurevych (2016), the MLP is trained on the VU Amsterdam Metaphor Corpus (VUAMC), a subset of the BNC where each token has been annotated as metaphorical or not (Steen et al., 2010), using cross-validation with an 80%-20% train-test split to optimize the regularization and learning rate parameters.

We then measure the percentage of all tokens labeled metaphorical by the metaphor identification algorithm in each transcript, denoting it by Met. We present an example text tagged by this algorithm in figure 1. Notably, the algorithm mistakenly tags the adverbially used preposition *up* in *ended up* as metaphorical; Do Dinh and Gurevych

[2] http://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/

> We ended **up** going to different high schools ...and then at home we also ran in different social **circles** and things like that.

Figure 1: Sample sentence from one of the transcripts in the 1ep data set. Tokens in **bold** was tagged metaphorical by the token-level metaphor detection algorithm.

(2016) cite this as one of the common failure modes of their algorithm, along with failure to detect metaphors that are only clearly metaphorical from a large amount of surrounding context.

Sentiment We posit that the sentiment scores produced by automated sentiment analysis algorithms should be able to detect disturbances in the production of emotional language, particularly in regard to metaphor. To this end, we create two features that summarize the distribution of sentiment scores in each transcript. In order to obtain token- and phrase-level sentiment scores, we use the implementation of the Recursive Neural Tensor Network sentiment analysis algorithm (Socher et al., 2012) that is included in the Stanford CoreNLP toolkit, with default settings. This implementation comes pre-trained on the Stanford Sentiment Treebank. Tokens are tagged on an integer scale from 1 (Very Negative) to 5 (Very Positive). For each transcript, we take the percentage of all token-level sentiment scores that were either extremely positive (score of 1) or extremely negative (score of 5), which we denote by SentTok and similarly compute the percentage of all phrase-level sentiment scores, which we denote by SentPhr. We also compute sentiment coherence as

$$\frac{1}{N}\sum_{i=1}^{N}|s_i - s_{i-1}|$$

where the s_i denotes either the sentiment score for token i (to compute CohTok), or the sentiment score for phrase i (to compute CohPhr).

4.2 Classification Algorithms

For all algorithms and data sets, we present results produced by leave-one-out cross-validation because of the small number of transcripts available. We use a radial-basis-function support-vector classifier and a convex-hull classifier to classify transcripts based on the variables above. The convex-hull classifier was previously used by Bedi et al. (2015). A test point is classified as originating from a CHR- participant if it lies within the convex-hull of all the CHR- data points in the training set; otherwise, it is classified as CHR+. The intuition behind the convex-hull approach is that individuals that eventually develop psychosis do not necessarily do so following a unique path to conversion, and moreover psychosis itself cannot be considered a well-defined single condition (Binbay et al., 2012); thus it is reasonable to hypothesize that the "breakdown" of mental abilities may occur along different trajectories for individual CHR+ patients.

5 Results & Discussion

5.1 Statistical Analysis

As predicted, we find that the metaphor identification algorithm does indeed tag a significantly higher proportion of the tokens in the transcripts of patients with schizophrenia as metaphorical (6.3%) than in the healthy controls' transcripts (5.2%); ($t = 3.76$, $p < .001$). No significant difference was found between the other variables of interest between patients with schizophrenia and healthy controls. No significant difference was found between males and females in metaphor use frequency ($t = 1.105$, $p = 0.28$).

5.2 Classification Performance

First-Episode Schizophrenia Transcripts Table 1 shows the performance of classifiers that individually use each of the five features §4.1 as predictors, as well as the classifier that uses all five in tandem (All)[3]. Baseline represents the results of a simple majority classifier (because 18 of the 33 transcripts belonged to patients with schizophrenia, this entails classifying all transcripts as belonging to patients with schizophrenia). Because the 1EP set was not balanced for gender or age, we also present the results of classifying men as having schizophrenia and women as not having schizophrenia (Gender) as well as the results of training a classifier on age (Age). Bedi and Mota represent the classification results attained by applying the features/method of Bedi et al. (2015) and Mota et al. (2012), respectively. Using all of the features to train the support-vector classifier performed better than using any of the features individually. The accuracy of the classifier based on all the features was significantly better than baseline (Fisher's exact test, $p < .005$). Notably, our features outperformed the features

[3] The All classifier does not use gender or age features.

suggested by Bedi et al. (2015) and by Mota et al. (2012).

Prodromal Psychosis Transcripts On the prodromal transcripts, a classifier trained on all the features once again outperformed classifiers on any of the features individually, which performed at or near baseline. Interestingly, the convex-hull classifier outperformed the support vector classifier on this data. The convex-hull classifier trained on all five features correctly identified the outcome of 33 of the 34 CHR patients (97.1% accuracy). The sole patient who was misclassified belonged to the CHR+ group. This is comparable to the 100% accuracy of the Bedi et al. (2015) method and superior to the 79.4% accuracy of the Mota et al. (2012) method.

In order to explore the relationship between the two data sets, we also applied the best classifier trained on the 1EP data to the prodromal data. Interestingly, the 1EP classifier tagged 29 of the 34 CHR patients as patients with schizophrenia, including all five patients in the CHR+ group. The hypersensitivity of the 1EP classifier when applied to the prodromal data suggests that the cues that discriminate between patients with first-episode schizophrenia and healthy controls tend to place CHR patients into the same category as patients with first-episode schizophrenia. It is worth noting that the classifier tagged all of the CHR+ patients as 1EP+. We believe this indicates that our method would be useful as a tool meant to channel limited attention and resources toward patients with particularly high risk (above and beyond the criteria that currently flag a patient as being CHR).

6 Conclusion

To our knowledge, this study is the first to demonstrate the utility of automated metaphor identification algorithms in a public-health setting, and particularly for the prediction or detection of schizophrenia. Our algorithm's performance on the task of schizophrenia diagnosis from transcripts outperforms the two existing methods detailed in existing literature.

Table 1: Classification performance on 1ep set.

Variables	F-score	Accuracy
Baseline	0.703	0.563
Gender	0.703	0.656
Age	0.629	0.594
CohTok	0.694	0.531
CohPhr	0.703	0.656
Met	0.789	0.750
SentTok	0.732	0.688
SentPhr	0.718	0.656
All	**0.848**	**0.844**
Bedi	0.744	0.688
Mota	0.732	0.688

Our results also contribute to clinical knowledge of the nature of language-use abnormalities in schizophrenia, as they support previous research which finds that those suffering from schizophrenia produce more metaphors in free speech than healthy controls. Previously it was only possible to measure such disturbances by labor-intensive and subjective hand-coding of transcripts for metaphoricity, or by the assessment of expert clinicians, whose time is limited. This work breaks new ground by showing that such disturbances can be measured in an automated and reproducible fashion, using features generated via machine learning.

Our work is somewhat constrained by the small sample size available to us. As our data comes from a vulnerable population, obtaining a larger data set is challenging, but essential for future work. In fact, two of the authors are in the process of collecting data from a total of 120 CHR individuals. This would enable a more thorough investigation of a larger and more sophisticated suite of linguistic features, and especially a more fine-grained analysis of the interaction of metaphor and emotional language in schizophrenia.

Acknowledgments

This work was funded in part via NIMH grants R01MH107558 and R03MH108933. The authors have no conflicts to declare.

References

American Psychiatric Association. 2013. *Diagnostic and Statistical Manual of Mental Disorders (DSM-5)*. American Psychiatric Association.

Nancy C. Andreasen. 1986. The scale for assessment of thought, language and communication (TLC). *Schizophrenia Bulletin* 12:473–482.

Gillinder Bedi, Facundo Carrillo, Guillermo A. Cecchi, Diego Fernández Slezak, Mariano Sigman, Natália B. Mota, Sidarta Ribeiro, Daniel C. Javitt, Mauro Copelli, and Cheryl M. Corcoran. 2015. Automated analysis of free speech predicts psychosis onset in high-risk youths. *npj Schizophrenia* 1:15030.

S. Ben-David, M. Birnbaum, M. Eilenberg, J. DeVylder, K. Gill, J. Schienle, N. Azimov, E.P. Lukens,

L. Davidson, and C.N. Corcoran. 2014. The subjective experience of youths at clinical high risk for psychosis: a qualitative study. *Psychiatric Services* pages 1499–1501.

Mathias Benedek, Roger Beaty, Emanuel Jauk, Karl Koschutnig, Andreas Fink, Paul J. Silvia, Beate Dunst, and Aljoscha C. Neubauer. 2014. The neural basis of figurative language production. *NeuroImage* 90:99–106.

Steven Bethard, Vicky Tzuyin Lai, and James H. Martin. 2009. Topic model analysis of metaphor frequency for psycholinguistic stimuli. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*. Association for Computational Linguistics, pages 9–16.

Richard M Billow, Jeffrey Rossman, Nona Lewis, Deberah Goldman, and Charles Raps. 1997. Observing expressive and deviant language in schizophrenia. *Metaphor and Symbol* 12(3):205–216.

T. Binbay, M. Drukker, H. Elbi, F.A. Tank, F. zknay, H. Onay, N. Zal, J. van Os, and K. Alptekin. 2012. Testing the psychosis continuum: differential impact of genetic and nongenetic risk factors and comorbid psychopathology across the entire spectrum of psychosis. *Schizophrenia Bulletin* 38:992–1002.

Julia Birke and Anoop Sarkar. 2006. A clustering approach for nearly unsupervised recognition of nonliteral language. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*. pages 329–336.

David B. Bracewell, Marc T. Tomlinson, Michael Mohler, and Bryan Rink. 2014. A tiered approach to the recognition of metaphor. In *International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*. pages 403–414.

Evan Chen, Page Widick, and Anjan Chatterjee. 2008. Functional-anatomical organization of predicate metaphor processing. *Brain and Language* 107:194–202.

C.M. Corcoran, J.G. Keilp, J. Kayser, C. Klim, P.D. Butler, G.E. Bruder, R.C. Gur, and D.C. Javitt. 2015. Emotion recognition deficits as predictors of transition in individuals at clinical high risk for schizophrenia: a neurodevelopmental perspective. *Psychological Medicine* 45:2959–2973.

Erik-Lân Do Dinh and Iryna Gurevych. 2016. Token-level metaphor detection using neural networks. In *Proceedings of the Fourth Workshop on Metaphor in NLP*. Association for Computational Linguistics, San Diego, California, pages 28–33.

Jonathan Dunn. 2013a. Evaluating the premises and results of four metaphor identification systems. In *Computational Linguistics and Intelligent Text Processing*, Springer, pages 471–486.

Jonathan Dunn. 2013b. What metaphor identification systems can tell us about metaphor-in-language. In *Proceedings of the First Workshop on Metaphor in NLP*. pages 1–10.

Brita Elvevåg, Peter W. Foltz, Daniel R. Weinberger, and Terry E. Goldberg. 2007. Quantifying incoherence in speech: An automated methodology and novel application to schizophrenia. *Schizophrenia Research* 93(1):304–316.

Brita Elvevåg, Kim Helsen, Marc De Hert, Kim Sweers, and Gert Storms. 2011. Metaphor interpretation and use: a window into semantics in schizophrenia. *Schizophrenia Research* 133(1):205–211.

Matt Gedigian, John Bryant, Srini Narayanan, and Branimir Ciric. 2006. Catching metaphors. In *Proceedings of the Third Workshop on Scalable Natural Language Understanding*. Association for Computational Linguistics, New York, pages 41–48.

D. C. Goff, C. Cather, A.E. Evins, D.C. Henderson, O. Freudenreich, P.M. Copeland, and F.M. Sacks. 2005. Medical morbidity and mortality in schizophrenia: guidelines for psychiatrists. *Journal of Clinical Psychiatry* 66:183–194.

E. Darío Gutiérrez, Ekaterina Shutova, Tyler Marghetis, and Benjamin K. Bergen. 2016. Literal and metaphorical senses in compositional distributional semantic models. In *Annual Meeting of the Association for Computational Linguistics*. pages 183–193.

Ilana Heintz, Ryan Gabbard, Mahesh Srinivasan, David Barner, Donald S Black, Marjorie Freedman, and Ralph Weischedel. 2013. Automatic extraction of linguistic metaphor with lda topic modeling. In *Proceedings of the First Workshop on Metaphor in NLP*. pages 58–66.

Dirk Hovy, Shashank Srivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huiying Li, Whitney Sanders, and Eduard Hovy. 2013. Identifying metaphorical word use with tree kernels. In *Proceedings of the First Workshop on Metaphor in NLP*. pages 52–57.

Beata Beigman Klebanov, Chee Wee Leong, Michael Heilman, and Michael Flor. ???? Different texts, same metaphors: unigrams and beyond. In *Proceedings of the Second Workshop on Metaphor in NLP*. Baltimore, MD, USA, pages 11–17.

Zoltán Kövecses. 2003. *Metaphor and emotion: Language, culture, and body in human feeling*. Cambridge University Press, Cambridge, UK.

Saisuresh Krishnakumaran and Xiaojin Zhu. 2007. Hunting elusive metaphors using lexical resources. In *Proceedings of the Workshop on Computational approaches to Figurative Language*. Association for Computational Linguistics, pages 13–20.

Gina R. Kuperberg. 2010. Language in schizophrenia part 1: an introduction. *Language and Linguistics Compass* 4(8):576–589.

Linlin Li, Benjamin Roth, and Caroline Sporleder. 2010. Topic models for word sense disambiguation and token-based idiom detection. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 1138–1147.

Linlin Li and Caroline Sporleder. 2009. Classifier combination for contextual idiom detection without labelled data. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, pages 315–323.

Linlin Li and Caroline Sporleder. 2010. Using Gaussian mixture models to detect figurative language in context. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 297–300.

Bing Liu. 2015. *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press, Cambridge, UK.

Tandy J. Miller, Thomas H. McGlashan, Joanna L. Rosen, Kristen Cadenhead, Joseph Ventura, William McFarlane, Diana O. Perkins, Godfrey D. Pearlson, and Scott W. Woods. 2003. Prodromal assessment with the structured interview for prodromal syndromes and the scale of prodromal symptoms: predictive validity, interrater reliability, and training to reliability. *Schizophrenia Bulletin* 29:703–715.

Michael Mohler, David Bracewell, David Hinote, and Marc Tomlinson. 2013. Semantic signatures for example-based linguistic metaphor detection. In *Proceedings of the First Workshop on Metaphor in NLP*. pages 27–35.

Natalia B Mota, Nivaldo AP Vasconcelos, Nathalia Lemos, Ana C Pieretti, Osame Kinouchi, Guillermo A Cecchi, Mauro Copelli, and Sidarta Ribeiro. 2012. Speech graphs provide a quantitative measure of thought disorder in psychosis. *PloS one* 7(4):e34928.

Yair Neuman, Dan Assaf, Yohai Cohen, Mark Last, Shlomo Argamon, Newton Howard, and Ophir Frieder. 2013. Metaphor identification in large texts corpora. *PLoS ONE* 8:e62343.

Vikram Patel, Alan J Flisher, Sarah Hetrick, and Patrick McGorry. 2007. Mental health of young people: a global public-health challenge. *The Lancet* 369(9569):1302–1313.

Roberta Rasetti, Venkata S. Mattay, Lisa M. Wiedholz, Bhaskar S. Kolachana, Ahmad R. Hariri, Joseph H. Callicott, Andreas Meyer-Lindenberg, and Daniel R. Weinberger. 2009. Evidence that altered amygdala activity in schizophrenia is related to clinical state and not genetic risk. *American Journal of Psychiatry* 166(2):216–225.

Gwenda L. Schmidt, Alexander Kranjec, Eileen R. Cardillo, and Anjan Chatterjee. 2010. Beyond laterality:a critical assement of research on the neural basis of metaphor. *Journal of the International Neuropsychological Society* 16:1–5.

Marc Schulder and Eduard Hovy. 2014. Metaphor detection through term relevance. In *Proceedings of the Second Workshop on Metaphor in NLP*. Baltimore, MD, USA, pages 18–26.

Ekaterina Shutova and Lin Sun. 2013. Unsupervised metaphor identification using hierarchical graph factorization clustering. In *Human Language Technologies: The 2013 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. pages 978–988.

Ekaterina Shutova, Lin Sun, and Anna Korhonen. 2010. Metaphor identification using verb and noun clustering. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, pages 1002–1010.

Ekatrina Shutova. 2015. Design and evaluation of metaphor processing systems. volume Forthcoming.

Jason C Simeone, Alexandra J Ward, Philip Rotella, Jenna Collins, and Ricarda Windisch. 2015. An evaluation of variation in published estimates of schizophrenia prevalence from 1990-2013: a systematic literature review. *BMC psychiatry* 15(1):193.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pages 1201–1211.

Caroline Sporleder and Linlin Li. 2009. Unsupervised recognition of literal and non-literal use of idiomatic expressions. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 754–762.

Gerard J. Steen, Aletta G. Dorst, J. Berenike Herrmann, Anna Kaal, Tina Krennmayr, and Trijntje Pasma. 2010. *A method for linguistic metaphor identification: From MIP to MIPVU*, volume 14. John Benjamins Publishing, Amsterdam/Philadelphia.

Tomek Strzalkowski, George A. Broadwell, Sarah Taylor, Laurie Feldman, Boris Yamrom, Samira Shaikh, Ting Liu, Kit Cho, Umit Boz, Ignacio Cases, and Kyle Elliot. 2013. Robust extraction of metaphors from novel data. In *Proceedings of the First Workshop on Metaphor in NLP*. Association for Computational Linguistics, Atlanta, Georgia, pages 67–76.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. pages 248–258.

Yulia Tsvetkov, Elena Mukomel, and Anatole Gershman. 2013. Cross-lingual metaphor detection using common semantic features.

Peter D. Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the 2011 Conference on the Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '11, pages 680–690.

Anja Vaskinn, Kjetil Sundet, Svein Friis, Carmen Simonsen, Astrid B Birkenaes, Halldora JONsdOTtir, Petter Andreas Ringen, and Ole A Andreassen. 2008. Emotion perception and learning potential: mediators between neurocognition and social problem-solving in schizophrenia? *Journal of the International Neuropsychological Society* 14(02):279–288.

Veronika Vincze, Gábor Gosztolya, László Tóth, Ildikó Hoffmann, and Gréta Szatlóczki. 2016. Detecting mild cognitive impairment by exploiting linguistic information from transcripts. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Berlin, Germany, pages 181–187.

World Health Organization. 2016. Schizophrenia fact sheet. http://www.who.int/mediacentre/factsheets/fs397/en/.

Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking

Hannah Rashkin[†] **Eunsol Choi**[†] **Jin Yea Jang**[‡]
Svitlana Volkova[‡] **Yejin Choi**[†]
[†]Paul G. Allen School of Computer Science & Engineering, University of Washington
{hrashkin,eunsol,yejin}@cs.washington.edu
[‡]Data Sciences and Analytics, Pacific Northwest National Laboratory
{jinyea.jang,svitlana.volkova}@pnnl.gov

Abstract

We present an analytic study on the language of news media in the context of political fact-checking and fake news detection. We compare the language of real news with that of satire, hoaxes, and propaganda to find linguistic characteristics of untrustworthy text. To probe the feasibility of automatic political fact-checking, we also present a case study based on PolitiFact.com using their factuality judgments on a 6-point scale. Experiments show that while media fact-checking remains to be an open research question, stylistic cues can help determine the truthfulness of text.

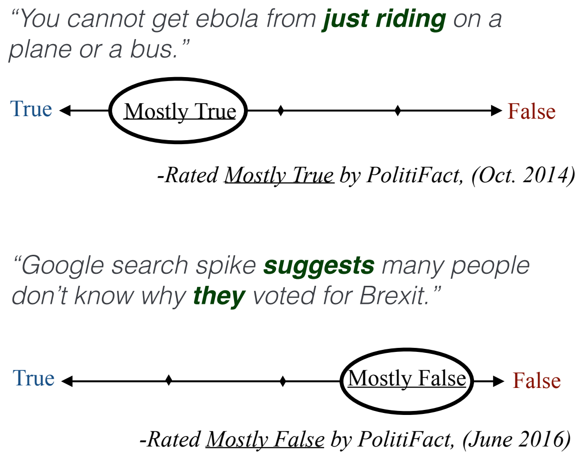

"You cannot get ebola from **just riding** on a plane or a bus."
True
Mostly True
False
-Rated Mostly True by PolitiFact, (Oct. 2014)
"Google search spike **suggests** many people don't know why **they** voted for Brexit."
True
Mostly False
False
-Rated Mostly False by PolitiFact, (June 2016)

Figure 1: Example statements rated by PolitiFact as *mostly true* and *mostly false*. Misleading phrasing - bolded in green - was one reason for the in-between ratings.

1 Introduction

Words in news media and political discourse have a considerable power in shaping people's beliefs and opinions. As a result, their truthfulness is often compromised to maximize impact. Recently, fake news has captured worldwide interest, and the number of organized efforts dedicated solely to fact-checking has almost tripled since 2014.[1] Organizations, such as PolitiFact.com, actively investigate and rate the veracity of comments made by public figures, journalists, and organizations.

Figure 1 shows example quotes rated for truthfulness by PolitiFact. Per their analysis, one component of the two statements' ratings is the misleading phrasing (bolded in green in the figure). For instance, in the first example, the statement is true as stated, though only because the speaker hedged their meaning with the quantifier *just*. In the second example, two correlated events – Brexit and Google search trends – are presented ambiguously as if they were directly linked.

Importantly, like above examples, most fact-checked statements on PolitiFact are rated as neither entirely true nor entirely false. Analysis indicates that falsehoods often arise from subtle differences in phrasing rather than outright fabrication (Rubin et al., 2015). Compared to most prior work on deception literature that focused on binary categorization of truth and deception, political fact-checking poses a new challenge as it involves a graded notion of truthfulness.

While political fact-checking generally focuses on examining the accuracy of a single quoted statement by a public figure, the reliability of general news stories is also a concern (Connolly et al., 2016; Perrott, 2016). Figure 2 illustrates news types categorized along two dimensions: the *intent* of the authors (desire to deceive) and the *content* of the articles (true, mixed, false).

[1] https://www.poynter.org/2017/there-are-now-114-fact-checking-initiatives-in-47-countries/450477/

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2931–2937
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

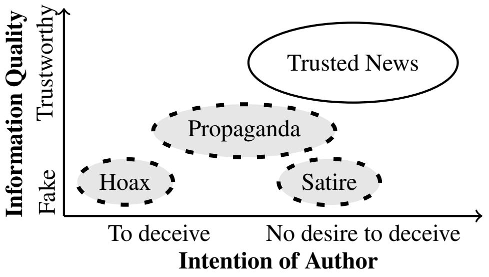


Figure 2: Types of news articles categorized based on their intent and information quality.

In this paper, we present an analytic study characterizing the language of political quotes and news media written with varying intents and degrees of truth. We also investigate graded deception detection, determining the truthfulness on a 6-point scale using the political fact-checking database available at PolitiFact.[2]

2 Fake News Analysis

News Corpus with Varying Reliability To analyze linguistic patterns across different types of articles, we sampled standard trusted news articles from the English Gigaword corpus and crawled articles from seven different unreliable news sites of differing types. Table 1 displays sources identified under each type according to US News & World Report.[3] These news types include:

- *Satire*: mimics real news but still cues the reader that it is not meant to be taken seriously
- *Hoax*: convinces readers of the validity of a paranoia-fueled story
- *Propaganda*: misleads readers so that they believe a particular political/social agenda

Unlike hoaxes and propaganda, satire is intended to be notably different from real news so that audiences will recognize the humorous intent. Hoaxes and satire are more likely to invent stories, while propaganda frequently combines truths, falsehoods, and ambiguities to confound readers.

To characterize differences between news types, we applied various lexical resources to trusted and fake news articles. We draw lexical resources from prior works in communication theory and stylistic analysis in computational linguistics. We tokenize the text with NLTK (Bird et al., 2009) and compute per-document count for each lexicon, and report averages per article of each type.

News Type	Source	# of Doc	# Tokens per Doc.
Trusted	Gigaword News	13,995	541
Satire	The Onion	14,170	350
	The Borowitz Report	627	250
	Clickhole	188	303
Hoax	American News	6,914	204
	DC Gazette	5,133	582
Propaganda	The Natural News	15,580	857
	Activist Report	17,869	1,169

Table 1: News articles used for analysis in Section 2.

First among these lexicons is the Linguistic Inquiry and Word Count (LIWC), a lexicon widely used in social science studies (Pennebaker et al., 2015). In addition, we estimate the use of strongly and weakly subjective words with a sentiment lexicon (Wilson et al., 2005). Subjective words can be used to dramatize or sensationalize a news story. We also use lexicons for hedging from (Hyland, 2015) because hedging can indicate vague, obscuring language. Lastly, we introduce intensifying lexicons that we crawled from Wiktionary based on a hypothesis that fake news articles try to enliven stories to attract readers. We compiled five lists from Wiktionary of words that imply a degree a dramatization (comparatives, superlatives, action adverbs, manner adverbs, and modal adverbs) and measured their presence.

Discussion Table 2 summarizes the ratio of averages between unreliable news and truthful news for a handful of the measured features. Ratios greater than one denote features more prominent in fake news, and ratios less than one denote features more prominent in truthful news. The ratios between unreliable/reliable news reported are statistically significant ($p < 0.01$) with Welsch t-test after Bonferroni correction.

Our results show that first-person and second-person pronouns are used more in less reliable or deceptive news types. This contrasts studies in other domains (Newman et al., 2003), which found fewer self-references in people telling lies about their personal opinions. Unlike that domain, news writers are trying to appear indifferent. Editors at trustworthy sources are possibly more

[2] All resources created for this paper including corpus of news articles from unreliable sources, collection of Politifact ratings, and compiled Wiktionary lexicons have been made publicly available at `homes.cs.washington.edu/~hrashkin/factcheck.html`

[3] www.usnews.com/news/national-news/articles/2016-11-14/avoid-these-fake-news-sites-at-all-costs

Lexicon Markers	Ratio	Example		Max
		Source	Text	
Swear (LIWC)	7.00	Borowitz Report	... Ms. Rand, who has been **damned** to eternal torment ...	S
2nd pers (You)	6.73	DC Gazette	**You** would instinctively justify and rationalize **your** ...	P
Modal Adverb	2.63	American News	... investigation of Hillary Clinton was **inevitably** linked ...	S
Action Adverb	2.18	Activist News	... if one **foolishly** assumes the US State Department ...	S
1st pers singular (I)	2.06	Activist Post	**I** think its against the law of the land to finance riots ...	S
Manner Adverb	1.87	Natural News	... consequences of **deliberately** engineering extinction.	S
Sexual (LIWC)	1.80	The Onion	... added that his daughter better not be **pregnant**.	S
See (LIWC)	1.52	Clickhole	New Yorkers ... can bask in the **beautiful image** ...	H
Negation(LIWC)	1.51	American News	There is **nothing** that outrages liberals more than ...	H
Strong subjective	1.51	Clickhole	He has one of the most **brilliant** minds in basketball.	H
Hedge (Hyland, 2015)	1.19	DC Gazette	As the Communist Party USA website **claims**...	H
Superlatives	1.17	Activist News	Fresh water is the single **most** important natural resource	P
Weak subjective	1.13	American News	... he made that **very clear** in his response to her.	P
Number (LIWC)	0.43	Xinhua News	... **7 million** foreign tourists coming to the country in **2010**	S
Hear (LIWC)	0.50	AFP	The prime minister also **spoke** about the commission ...	S
Money (LIWC)	0.57	NYTimes	He has proposed to lift the state **sales tax** on groceries	P
Assertive	0.84	NYTimes	Hofstra has **guaranteed** scholarships to the current players.	P
Comparitives	0.86	Assoc. Press	... from fossil fuels to **greener** sources of energy	P

Table 2: Linguistic features and their relationship with fake news. The ratio refers to how frequently it appears in fake news articles compared to the trusted ones. We list linguistic phenomena more pronounced in the fake news first, and then those that appear less in the fake news. Examples illustrate sample texts from news articles containing the lexicon words. All reported ratios are statistically significant. The last column (MAX) lists compares which type of fake news most prominently used words from that lexicon (P = propaganda, S = satire, H = hoax)

rigorous about removing language that seems too personal, which is one reason why this result differs from other lie detection domains. This finding instead corroborates previous work in written domains found by Ott et al. (2011) and Rayson et al. (2001), who found that such pronouns were indicative of imaginative writing. Perhaps imaginative storytelling domains is a closer match to detecting unreliable news than lie detection on opinions.

Our results also show that words that can be used to exaggerate – subjectives, superlatives, and modal adverbs – are all used more by fake news. Words used to offer concrete figures – comparatives, money, and numbers – appear more in truthful news. This also builds on previous findings by Ott et al. (2011) on the difference between superlative/comparative usage.

Trusted sources are more likely to use assertive words and less likely to use hedging words, indicating that they are less vague about describing events, as well. This relates to psychology theories (Buller and Burgoon, 1996) that deceivers show more "uncertainty and vagueness" and "indirect forms of expression". Similarly, the trusted sources use the *hear* category words more often, possibly indicating that they are citing primary sources more often.

The last column in Table 2 shows the fake news type that uses the corresponding lexicon most prominently. We found that one distinctive feature of satire compared to other types of untrusted news is its prominent use of adverbs. Hoax stories tend to use fewer superlatives and comparatives. In contrast, compared to other types of fake news, propaganda uses relatively more assertive verbs and superlatives.

News Reliability Prediction We study the feasibility of predicting the reliability of the news article into four categories: trusted, satire, hoax, or propaganda. We split our collected articles into balanced training (20k total articles from the Onion, American News, The Activist, and the Gigaword news excluding 'APW', 'WPB' sources) and test sets (3k articles from the remaining sources). Because articles in the training and test set come from different sources, the models must classify articles without relying on author-specific cues. We also use 20% of the training articles as an in-domain development set. We trained a Max-Entropy classifier with L2 regularization on n-gram tf-idf feature vectors (up to trigrams).[4]

The model achieves F1 scores of 65% on the out-of-domain test set (Table 3). This is a promising result as it is much higher than random, but still leaves room for improvement compared to the

[4] N-gram tfidf vectors have acted as competitive means of cross-domain text-classification. Zhang et al. (2015) found that for data sets smaller than a million examples, this was the best model, outperforming neural models.

Data	Sources	Random	MaxEnt
Dev	in-domain	0.26	0.91
Test	out-of-domain	0.26	0.65

Table 3: F1 scores of 4-way classification of news reliability.

performance on the development set consisting of articles from in-domain sources.

We examined the 50 highest weighted n-gram features in the MaxEnt classifier for each class. The highest weighted n-grams for trusted news were often specific places (e.g., “washington”) or times (“on monday”). Many of the highest weighted from satire were vaguely facetious hearsay (“reportedly”, “confirmed”). For hoax articles, heavily weighted features included divisive topics (“liberals”, “trump”) and dramatic cues (“breaking”). Heavily weighted features for propaganda tend towards abstract generalities (“truth”, “freedom”) as well as specific issues (“vaccines”, “syria”). Interestingly, “youtube” and “video” are highly weighted for the propaganda and hoax classes respectively; indicating that they often rely on video clips as sources.

3 Predicting Truthfulness

Politifact Data Related to the issue of identifying the truthfulness of a news article is the fact-checking of individual statements made by public figures. Misleading statements can also have a variety of intents and levels of reliability depending on whom is making the statement.

PolitiFact[5] is a site led by Tampa Bay Times journalists who actively fact-check suspicious statements. One unique quality of PolitiFact is that each quote is evaluated on a 6-point scale of truthfulness ranging from “True” (factual) to “Pants-on-Fire False” (absurdly false). This scale allows for distinction between categories like mostly true (the facts are correct but presented in an incomplete manner) or mostly false (the facts are not correct but are connected to a small kernel of truth).

We collected labelled statements from PolitiFact and its spin-off sites (PunditFact, etc.) (10,483 statements in total). We analyze a subset of 4,366 statements that are direct quotes by the original speaker. The distributions of ratings on the PolitiFact scale for this subset are shown in Table 4. Most statements are labeled as neither completely true nor false.

	More True			More False		
	True	Mostly True	Half True	Mostly False	False	Pants-on-fire
6-class	20%	21%	21%	14%	17%	7%
2-class	62%			38%		

Table 4: PolitiFact label distribution. PolitiFact uses a 6-point scale ranging from: True, Mostly True, Half-true, Mostly False, False, and Pants-on-fire False.

We formulate a fine-grained truthfulness prediction task with Politifact data. We split quotes into training/development/test set of {2575, 712, 1074} statements, respectively, so that all of each speaker’s quotes are in a single set. Given a statement, the model returns a rating for how reliable the statement is (Politifact ratings are used as gold labels). We ran the experiment in two settings, one considering all 6 classes and the other considering only 2 (treating the top three truthful ratings as true and the lower three as false).

Model We trained an LSTM model (Hochreiter and Schmidhuber, 1997) that takes the sequence of words as the input and predicts the Politifact rating. We also compared this model with Maximum Entropy (MaxEnt) and Naive Bayes models, frequently used for text categorization.

For input to the MaxEnt and Naive Bayes models, we tried two variants: one with the word tf-idf vectors as input, and one with the LIWC measurements concatenated to the tf-idf vectors. For the LSTM model, we used word sequences as input and also a version where LSTM output is concatenated with LIWC feature vectors before undergoing the activation layer. The LSTM word embeddings are initialized with 100-dim embeddings from GLOVE (Pennington et al., 2014) and fine-tuned during training. The LSTM was implemented with Theano and Keras with 300-dim hidden state and a batch size of 64. Training was done with ADAM to minimize categorical cross-entropy loss over 10 epochs.

Classifier Results Table 5 summarizes the performance on the development set. We report macro averaged F1 score in all tables. The LSTM outperforms the other models when only using text as input; however the other two models improve substantially with adding LIWC features, particu-

[5] www.politifact.com/

	2-CLASS		6-CLASS	
	text	+ LIWC	text	+ LIWC
Majority Baseline	.39	-	.06	-
Naive Bayes	.44	.58	.16	.21
MaxEnt	.55	.58	.20	.21
LSTM	.58	.57	.21	.22

Table 5: Model performance on the Politifact validation set.

MODEL	FEATURE	2-CLASS	6-CLASS
Majority Baseline		.39	.06
Naive Bayes	text + LIWC	.56	.17
MaxEnt	text + LIWC	.55	.22
LSTM	text + LIWC	.52	.19
LSTM	text	.56	.20

Table 6: Model performance on the Politifact test set.

larly in the case of the multinomial naive Bayes model. In contrast, the LIWC features do not improve the neural model much, indicating that some of this lexical information is perhaps redundant to what the model was already learning from text.

We report results on the test set in Table 6. We again find that LIWC features improves MaxEnt and NB models to perform similarly to the LSTM model. As in the dev. set results, the LIWC features do not improve the LSTM's performance, and even seem to hurt the performance slightly.

4 Related Work

Deception Detection Psycholinguistic work in interpersonal deception theory (Buller and Burgoon, 1996) has postulated that certain speech patterns can be signs of a speaker trying to purposefully obscure the truth. Hedge words and other vague qualifiers (Choi et al., 2012; Recasens et al., 2013), for example, may add indirectness to a statement that obscures its meaning.

Linguistic aspects deception detection has been well-studied in a variety of NLP applications (Ott et al., 2011; Mihalcea and Strapparava, 2009; Jindal and Liu, 2008; Girlea et al., 2016; Zhou et al., 2004). In these applications, people purposefully tell lies to receive an extrinsic payoff. In our study, we compare varying types of unreliable news source, created with differing intents and levels of veracity.

Fact-Checking and Fake News There is research in political science exploring how effective fact-checking is at improving people's awareness (Lord et al., 1979; Thorson, 2016; Nyhan and Reifler, 2015). Prior computational works (Vlachos and Riedel, 2014; Ciampaglia et al., 2015) have proposed fact-checking through entailment from knowledge bases. Our work takes a more linguistic approach, performing lexical analysis over varying types of falsehood.

Biyani et al. (2016) examine the unique linguistic styles found in clickbait articles, and Kumar et al. (2016) also characterize hoax documents on Wikipedia. The differentiation between these fake news types is also proposed in previous work (Rubin et al., 2015). Our paper extends this work by offering a quantitative study of linguistic differences found in articles of different types of fake news, and build predictive models for graded deception across multiple domains – PolitiFact and news articles. More recent work (Wang, 2017) has also investigated PolitiFact data though they investigated meta-data features for prediction whereas our investigation is focused on linguistic analysis through stylistic lexicons.

5 Conclusion

We examine truthfulness and its contributing linguistic attributes across multiple domains e.g., online news sources and public statements. We perform multiple prediction tasks on fact-checked statements of varying levels of truth (graded deception) as well as a deeper linguistic comparison of differing types of fake news e.g., propaganda, satire and hoaxes. We have shown that fact-checking is indeed a challenging task but that various lexical features can contribute to our understanding of the differences between more reliable and less reliable digital news sources.

6 Acknowledgements

We would like to thank anonymous reviewers for providing insightful feedback. The research described in this paper was conducted under the Laboratory Directed Research and Development Program at Pacific Northwest National Laboratory, a multiprogram national laboratory operated by Battelle for the U.S. Department of Energy, the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1256082, in part by NSF grants IIS-1408287, IIS-1714566, and gifts by Google and Facebook.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Prakhar Biyani, Kostas Tsioutsiouliklis, and John Blackmer. 2016. "8 amazing secrets for getting more clicks": Detecting clickbaits in news streams using article informality. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, pages 94–100.

David B. Buller and Judee K. Burgoon. 1996. Interpersonal deception theory. *Communication Theory* 6(3):203–242. https://doi.org/10.1111/j.1468-2885.1996.tb00127.x.

Eunsol Choi, Chenhao Tan, Lillian Lee, Cristian Danescu-Niculescu-Mizil, and Jennifer Spindel. 2012. Hedge detection as a lens on framing in the gmo debates: A position paper. In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*. Association for Computational Linguistics, pages 70–79.

Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M. Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. 2015. Computational fact checking from knowledge networks. *PLOS ONE* 10(6):e0128193. https://doi.org/10.1371/journal.pone.0128193.

Kate Connolly, Angelique Chrisafis, Poppy McPherson, Stephanie Kirchgaessner, Benjamin Haas, Dominic Phillips, Elle Hunt, and Michael Safi. 2016. Fake news: An insidious trend that's fast becoming a global problem. https://www.theguardian.com/media/2016/dec/02/fake-news-facebook-us-election-around-the-world. Accessed: 2017-01-30.

Codruta Girlea, Roxana Girju, and Eyal Amir. 2016. Psycholinguistic features for deceptive role detection in werewolf. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 417–422.

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

Ken Hyland. 2015. Metadiscourse. In *The International Encyclopedia of Language and Social Interaction*, John Wiley & Sons, Inc., pages 1–11.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, pages 219–230.

Srijan Kumar, Robert West, and Jure Leskovec. 2016. Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pages 591–602.

Charles G Lord, Lee Ross, and Mark R Lepper. 1979. Biased assimilation and attitude polarization: The effects of prior theories on subsequently considered evidence. *Journal of Personality and Social Psychology* 37(11):2098–2109.

Rada Mihalcea and Carlo Strapparava. 2009. The lie detector: Explorations in the automatic recognition of deceptive language. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Association for Computational Linguistics, pages 309–312.

Matthew L Newman, James W Pennebaker, Diane S Berry, and Jane M Richards. 2003. Lying words: Predicting deception from linguistic styles. *Personality and social psychology bulletin* 29(5):665–675.

Brendan Nyhan and Jason Reifler. 2015. The effect of fact-checking on elites: A field experiment on US state legislators. *American Journal of Political Science* 59(3):628–640.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 309–319.

James W. Pennebaker, Roger J. Booth, Ryan L. Boyd Boyd, and Martha E. Francis. 2015. *Linguistic Inquiry and Word Count: LIWC2015*. Pennebaker-Conglomerates, Austin, TX. www.liwc.net.

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

Kathryn Perrott. 2016. 'Fake news' on social media influenced US election voters, experts say. http://www.abc.net.au/news/2016-11-14/fake-news-would-have-influenced-us-election-experts-say/8024660. Accessed: 2017-01-30.

Paul Rayson, Andrew Wilson, and Geoffrey Leech. 2001. Grammatical word class variation within the british national corpus sampler. *Language and Computers* 36(1):295–306.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pages 1650–1659.

Victoria L. Rubin, Yimin Chen, and Niall J. Conroy. 2015. Deception detection for news: Three types of fakes. *Proceedings of the Association for Information Science and Technology* 52(1):1–4.

Emily Thorson. 2016. Belief echoes: The persistent effects of corrected misinformation. *Political Communication* 33(3):460–480.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*. Association for Computational Linguistics, pages 18–22.

William Yang Wang. 2017. "Liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the Association for Computational Linguistics Short Papers*. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 347–354.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*. pages 649–657.

Lina Zhou, Judee K. Burgoon, Jay F. Nunamaker, and Doug Twitchell. 2004. Automating linguistics-based cues for detecting deception in text-based asynchronous computer-mediated communications. *Group Decision and Negotiation* 13(1):81–106.

Topic-Based Agreement and Disagreement in US Electoral Manifestos

Stefano Menini[1,3], Federico Nanni[2], Simone Paolo Ponzetto[2], Sara Tonelli[1]
[1]Fondazione Bruno Kessler, Trento, Italy
[2]University of Mannheim, Germany
[3]University of Trento, Italy
{menini, satonelli}@fbk.eu
{federico, simone}@informatik.uni-mannheim.de

Abstract

We present a topic-based analysis of agreement and disagreement in political manifestos, which relies on a new method for topic detection based on key concept clustering. Our approach outperforms both standard techniques like LDA and a state-of-the-art graph-based method, and provides promising initial results for this new task in computational social science.

1 Introduction

During the last decade, the adoption of natural language processing (NLP) techniques for the study of political phenomena has gained considerable momentum (Grimmer and Stewart, 2013), arguably because of both the availability of parliamentary proceedings (van Aggelen et al., 2017), electoral manifestos (Volkens et al., 2011) and campaign debates (Woolley and Peters, 2008), and the interest of the computational social science (CSS) community in the potential of text mining methods for advancing political science research (Lazer et al., 2009).

Previous work focused on the automatic detection of sentiment expressions in political news (Young and Soroka, 2012), the identification of ideological proportions (Sim et al., 2013) and the scaling on a left-right spectrum of politicians' speeches (Slapin and Proksch, 2008). More recently, researchers looked at topic-centered approaches to provide finer-grained analyses, including segmentation methods for topic-labeled manifestos (Glavaš et al., 2016), supporting manual coders in identifying coarse-grained political topics (Zirn et al., 2016), as well as topic-based and cross-lingual political scaling (Nanni et al., 2016; Glavaš et al., 2017).

Measuring Agreement. Automatically measuring the level of agreement in political documents (Gottipati et al., 2013; Menini and Tonelli, 2016) has the potential of supporting political analyses such as the comparisons between campaign strategies (Burton et al., 2015), the study of promises kept and broken after elections (Naurin, 2011), the formation of coalitions (Debus, 2009) and the interactions between government and opposition (Hix and Noury, 2016). However, previous work relies on the availability of pre-defined topics, including supervised methods (Galley et al., 2004; Hillard et al., 2003), approaches leveraging collaboratively generated resources (Gottipati et al., 2013; Awadallah et al., 2012) or pairwise agreement detection from political debates (Menini and Tonelli, 2016).

Our Contributions. *a) New task:* Given a collection of political documents such as, e.g., electoral manifestos, we look at ways to perform an automatic, topic-based agreement-disagreement classification. *b) New approach:* We first segment the texts into coarse-grained domains. Next, coarse domains are used to extract a fine-grained list of topic-based points of view which, in turn, are used to perform classification. We achieve this by developing a novel approach for topic detection on the basis of key concept clustering techniques: this is shown to outperform not only LDA-based Topic Modeling – the *de facto* standard approach for this task in CSS (Grimmer and Stewart, 2013) – but also established unsupervised (k-means) and state-of-the-art graph-based clustering techniques. *c) Experimental study and resources:* We use manifestos from the Comparative Manifesto Project (Volkens et al., 2011). As in previous works (Zirn et al., 2016), we focus on a subset consisting of six U.S. manifestos (Republican and Democrat) from the 2004, 2008 and 2012 elections. We show

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2938–2944
Copenhagen, Denmark, September 7–11, 2017.

that our method leads to promising results when measuring the topic-based agreement between the party manifestos, thus indicating the overall feasibility of the task. Additionally, we release all code and annotations related to this paper to foster further work from the research community.

2 System Overview

We present a new system for measuring the topic-based agreement of political manifestos. Our approach consists of four main steps: i) macro-domain detection, e.g. *foreign policy, economy, welfare*, ii) key concept extraction, iii) topic detection as key concept clustering, e.g., *energy consumption, new energy solution, petroleum dependence* for the topic *green economy*, and iv) pairwise, topic-based agreement detection.

The central component of our pipeline is a new approach for fine-grained topic detection in political contents based on key concept clustering. This is because, among existing methods, supervised approaches cannot be applied here due to the scarce availability of in-domain labeled data, as well as the already remarked high complexity of the annotation process (Benoit et al., 2016). Moreover, the application of unsupervised topic detection techniques like LDA has been shown during prototyping to produce low-quality topics that are rather coarse (cf. the results in Section 3).

Similar to LDA-based approaches, we view each topic as a cluster of words or phrases. However, given that we are in a domain with topics built around rather specific lexical cues, we do not rely on the entire vocabulary of the documents. Instead, we build clusters that are made up of semantically similar key concepts extracted from the documents themselves, including both single and multiwords (i.e. keywords and keyphrases). In the next paragraph we present an overview of each component of our system.

1) Domain Detection. We are given as input sentences from a political manifesto. The first step of our work is to classify them into the seven macro-domains defined by the Manifesto Project, namely *external relations*, *freedom and democracy*, *political system*, *economy*, *welfare and quality of life*, *fabric of society*, *social groups*. To achieve this goal, we use ClassyMan, a system developed in a previous work (Zirn et al., 2016), which predicts the domains and domain shifts between pairs of adjacent sentences.

2) Key concept Extraction. Next, for each domain we process each sentence using Keyphrase Digger (KD) (Moretti et al., 2015). KD is a rule-based (hence domain-agnostic) system for key concept extraction that combines statistical measures with linguistic information, and which has shown competitive performance on the SemEval 2010 benchmark (Kim et al., 2010). We set the tool to extract lemmatized key concepts up to three tokens. For each key concept, we compute its tf-idf, considering each domain as a different document. The result is a list of key concepts for each domain, with a score representing their relevance to the domain.

3) Key concept Clustering. Starting from the flat lists of key concepts extracted by KD, we adopt a recursive procedure to merge them into meaningful clusters. First, we build a distributional semantic vector for each key concept by averaging the embeddings of each word in the key concept (we use the GloVe embeddings from Pennington et al. (2014) with 50 dimensions, pre-trained on Wikipedia). Next, we build a semantic graph representation where *a)* each node consists of a key concept, *b)* the weight of each edge is the cosine distance between their respective embedding vectors and *c)* edges are directed, pointing to the node of the key concept with the higher tf-idf. For ties, we create multiple edges. To direct the nodes we adopt tf-idf, since we want to weigh the key concepts according to the relevance for the macro-domain we are processing. This allows us to obtain well-defined groups within the domains.

To reduce the number of weak edges, we set a cosine similarity threshold of 0.8,[1] and we set an edge between two multi-word keyphrases if they have at least one word in common (e.g. *ethnic minority*, *black minority*).

Finally, we obtain clusters of semantically related key concepts from the graph as follows: *a)* we extract all groups of key concepts with an edge directed to the same node and create a first set of clusters. Then, *b)* clusters sharing at least 50% of the key concepts are merged. Next, *c)* the clusters purity is improved by removing the less relevant key concepts. These are identified as those key concepts whose cosine distance is more than 1.5 times the standard deviation from the centroid of

[1] We evaluated the clustering output with different thresholds ranging from 0.6 to 0.9. The value of 0.8 is the one leading to the best accuracy. In Table 1 and 2 we report the results using this threshold.

the cluster. At the end of this process, we obtain for each domain a set of clusters or *topics*, made of semantically related key concepts. The number of clusters is determined dynamically during the process and does not need to be defined a priori.

4) Statement Extraction. We use the clusters of key concepts to identify pairs of statements, related to the same topic, from the Republican (R) and Democratic (D) manifestos. For each topic, we first collect the statements from D and R manifestos having among their key concepts one of the key concepts defining the topic and then we pair groups of three statements from D with groups of three statements from R. We use groups of three statements because it allows us *i)* to obtain a sufficient number of pairs to perform automatic classification, and *ii)* to improve the quality of the manual annotations. We noticed during an initial evaluation that annotators focus easier on groups of 3 sentences rather than larger groups and that, on the other hand, using less than 3 sentences decreases the chances to obtain at least two statements in agreement/disagreement within a pair.

5) Agreement Classification. The last step is the automatic classification of agreement and disagreement between Republicans and Democrats. To classify pairs of statements, we rely on a supervised machine learning approach with the set of features used in Menini and Tonelli (2016), in which a similar task is addressed. The classification relies on features related to surface information such as lexical overlap and negation, to the semantic content of the statements (e.g. sentiment) and to their relation (e.g. entailment).

3 Evaluation

3.1 Topic Extraction

Having a set of manifestos annotated with coarse-grained domains – using *ClassyMan*, which achieved a micro F1-Score of 0.78 across the seven macro-topics in a 10-fold cross validation setting – the central step of our pipeline is to detect clusters of key concepts representing fine-grained topics in each macro domain. To do that, we adopt the method described above, that we call here *Key Concept Clusters*. We examine its performance in comparison with two types of baselines.

LDA Baselines. We first employ vanilla LDA, a common approach for topic detection in CSS (Grimmer and Stewart, 2013), relying on the assumption that tokens often co-occurring together in a corpus belong to the same topic. For this task, we use the Mallet topic model package.[2] Given the fact that our method for key concept clustering identifies on average 30 topics per domain, we create a corpus for each domain with all its sentences and we run LDA with 10,000 iterations to obtain 30 topics. We test LDA by considering all the tokens in the corpus (*Vanilla LDA*) and only the extracted key concepts (*Key concept LDA*).

Clustering Baselines. The second type of baseline adopts the same representation of key concepts used in our approach, i.e., we represent candidate phrases by averaging the embeddings of their constituent words. We test two different clustering approaches to group them into topics: the first uses *K-means* (with 30 clusters). The second (*Graph-based*) builds a fully-connected semantic relatedness graph by measuring the cosine similarity between all pairs of key concepts: topic clustering is then obtained by finding all maximal cliques in the graph using the Bron-Kerbosch algorithm.

Evaluation. In order to assess the overall quality of the topics produced by each approach, we adopt the word-intrusion post-hoc evaluation method (Chang et al., 2009) using the platform presented in Lauscher et al. (2016). For each approach, we randomly pick 100 topics and for each topic we keep two sets of key concepts, respectively the four and eight top-relevant elements of the cluster.[3] Then, we add to these four/eight words a new word from another topic (i.e. the intruder), and we shuffle the obtained five/nine words. Finally, we ask three political science experts to identify the intruder. The more the topics are coherent, the easier the intruder is detected. While this type of post-hoc evaluation is extremely time-consuming – no less than 45 minutes of work for annotator for each produced ranking, thus hindering the experimental assessment of, for instance, the role of different numbers of topics for each baseline – it is necessary given the already remarked limits of existing gold standards manually-created for the task (Mikhaylov et al., 2012; King et al., 2017).

Results. As shown in Table 1, our system outperforms the other methods with an accuracy of 0.86 in the word-intrusion task with four key con-

[2] http://mallet.cs.umass.edu/

[3] For LDA, Mallet provides already ranked results. For the other approaches, the most relevant key concepts are the key concepts closest to the centroid of the cluster.

Method	Acc.@4	Acc.@8
Vanilla-LDA	0.22	0.35
Key concept-LDA	0.29	0.36
Graph-based Clusters	0.46	0.44
k-means Clusters	0.72	0.67
Key concept Clusters	0.86	0.67

Table 1: Topics evaluation: accuracy in word intrusion task. The table reports the accuracy values on the first 4 and 8 key concepts in the clusters.

Method	Kappa@4	Kappa@8
Vanilla LDA	0.32	0.46
Key concept LDA	0.50	0.40
Graph-based Clusters	0.39	0.32
k-means Clusters	0.65	0.61
Key concept Clusters	0.79	0.62

Table 2: Inter-annotator agreement (IAA) evaluation (Fleiss' kappa) in the word intrusion task. The table reports the IIA on the first 4 and 8 key concepts in the clusters.

cepts in each cluster, while it decreases to 0.67 if we extend the evaluation to include eight key concepts. Besides, inter-annotator agreement (Fleiss' kappa), reported in Table 2, varies a lot across the different methods. In particular, the agreement in the intrusion task with four key concepts is higher for clusters generated with our method (0.79), while it is very low using LDA (0.32). This confirms the findings by Chang et al. (2009) that LDA topics are often difficult to interpret.

If we extend the evaluation to the first eight elements of each cluster, we notice that the difference between the agreement with our pipeline (0.62) and LDA (0.46) decreases. This shows that, with *key concept clusters*, increasing the number of key concepts in a topic affects their interpretation, although there is still an improvement with respect to the other approaches.

Final Tuning. We next tune clustering to classify fine-grained topics as in agreement or disagreement. Tuning is performed as to maximize clustering accuracy while obtaining a sufficient number of topics shared by both Democrats and Republicans. Since a cosine similarity threshold of 0.8 in the clustering process leads to clusters that are too specific, often addressed only by one of the two parties, we reduce the threshold to 0.7, so that the topics are likely to be covered by both manifestos. In addition, we want to compare the agreement focusing on small clusters, composed by a maximum of 10 key concepts. To obtain them, we iterate the clustering process over the key concepts of larger clusters, progressively increasing the cosine similarity threshold until there are no groups larger than 10 key concepts. We reach this goal with a threshold of 0.85. Using these settings, the accuracy (Acc. @4) of the clusters decreases to 0.74, but we obtain clusters that allow us to extract a total of 351 pairs covering 87 fine-grained topics. Table 3 shows some of the clusters extracted.

3.2 Agreement Classification

Data Annotation. The statements in the pairs have been annotated by three scholars of political science in terms of agreement, disagreement or none of the two. The annotation results in 158 pairs in disagreement, 135 in agreement and 58 neither in agreement nor in disagreement, with an inter-annotator agreement (IAA) of 0.64 (Fleiss' Kappa). Note that only in three cases the annotators claimed that the meaning of a sentence pair did not match with the topic detected with our approach. This additional finding highlights again the quality of our method for topic detection based on key concept clustering.

Agreement Classification. Agreement classification is carried out using Support Vector Machine (SVM) tested in two configurations. In the first setting, we train and test the classifier with 10-fold cross validation over the manually annotated pairs from the political manifestos. In the second configuration, we explore instead a cross-domain approach: we train the SVM on the 1960 Elections dataset from Menini and Tonelli (2016) and use all the pairs in our gold standard of political manifestos as test set. This experiment is aimed at assessing the impact of training on comparable data are from the same domain (i.e., transcript of political speeches vs. manifestos).

The results of both configurations are shown in Table 4, where they are compared to a random baseline. The results show that the set of features used suits our task, classifying the data with an accuracy comparable to the performance of human annotators, if we consider IAA as an upper bound for the task. We achieve nevertheless results that are in a lower range than Menini and Tonelli (2016), thus suggesting that agreement and disagreement is harder to detect in political mani-

Macro-domain: External Relations
japan, korea, missile, north_korea, south_korea, weapon_north_korea
extremism, renounce_terrorism, nuclear_terrorism, proliferation, security, terrorism
Macro-domain: Freedom and Democracy
culture, freedom, ideology, religion, society, tolerance, tradition
democracy, discrimination, first_amendment_rights, freedom, issue, law, rights_of_citizenship
Macro-domain: Political System
budget, budget_act, cost, cut, deficit, shortfall, tax
congressional_republican, election, republican, republican_platform, romney, vote
Macro-domain: Economy
alternative_fuel, electricity, fuel, gas, transportation_fuel
bailout, credit, loan, mortgage, payment, savings
Macro-domain: Welfare and Quality of Life
ailment, chronic, disease, health, illness, obesity, treatment_of_disease
global_energy_forum, industry, new_energy_solution, new_global_energy, solar_energy_generation
Macro-domain: Fabric of Society
crime, criminal, high-profile_criminal_conviction, prosecution
religious_freedom, religious, religious_discrimination
Macro-domain: Social Groups
agricultural agricultural_america, agricultural_production, agriculture, farm, rural, rural_america
hispanic, latino, latino_population

Table 3: Examples of key concept clusters extracted for each macro-domain.

Classification	Accuracy
Random Baseline	0.54
10-fold cross validation	0.66
1960 Elections training	0.61

Table 4: Results on agreement classification.

festos than in speeches. Finally the accuracy of the classifier in the cross-domain setting is lower than the one obtained with in-domain cross-validation, but still comparable with that of human annotators.

4 Conclusion

In this paper, we presented a system for supporting automatic topic-bases analyses of agreement and disagreement in political manifestos. This approach goes beyond established approaches for the task, which are either too coarse-grained or rely intensively on manual annotations.

Our method can provide insights into agreement and disagreement between parties, covering several topics of internal and foreign policy. By examining the results, we find an overall cross-party agreement of 46% regarding the discussed issues. However, this agreement varies substantially if we consider the different macro-domains. For example, while we notice a strong disagreement over the domain *political system*, especially for what concerns the responsibilities of previous administrations, other domains, such as *external relations*, present a more balanced ratio of agreement and disagreement between Republicans and Democrats. The possibility of measuring agreement at a finer level (topics) that is offered by our approach, shows, for example, that between 2004 and 2012 two opposite positions have been defined regarding the Middle East. On the contrary, there has been a general agreement on the role of the U.S. concerning the relations with Europe.

In the future, we hope that the pipeline presented in this paper will support political science researchers in studying topics such as party polarization through the analysis and comparison of electoral manifestos, parliamentary proceedings and campaign speeches. On the computational side, we will to extend our approach to cross-lingual data, in order to enable computer-assisted political analysis across different languages.

Downloads. The code for topic detection as key concept clustering process is available at https://dh.fbk.eu/technologies/keyphrase-clustering.

References

Astrid van Aggelen, Laura Hollink, Max Kemman, Martijn Kleppe, and Henri Beunders. 2017. The debates of the european parliament as linked open data. *Semantic Web*, 8(2):271–281.

Rawia Awadallah, Maya Ramanath, and Gerhard Weikum. 2012. Polaricq: Polarity classification of political quotations. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 1945–1949.

Kenneth Benoit, Drew Conway, Benjamin E Lauderdale, Michael Laver, and Slava Mikhaylov. 2016. Crowd-sourced text analysis: reproducible and agile production of political data. *American Political Science Review*, 110(02):278–295.

Michael John Burton, William J. Miller, and Daniel M. Shea. 2015. *Campaign Craft: The Strategies, Tactics, and Art of Political Campaign Management: The Strategies, Tactics, and Art of Political Campaign Management*. New York: Praeger.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems*, pages 288–296.

Marc Debus. 2009. Pre-electoral commitments and government formation. *Public Choice*, 138(1-2):45.

Michel Galley, Kathleen McKeown, Julia Hirschberg, and Elizabeth Shriberg. 2004. Identifying agreement and disagreement in conversational speech: Use of bayesian networks to model pragmatic dependencies. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 669–676.

Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. 2016. Unsupervised text segmentation using semantic relatedness graphs. In *Proceedings of the 5th Joint Conference on Lexical and Computational Semantics*, pages 125–130.

Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. 2017. Unsupervised cross-lingual scaling of political texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 688–693.

Swapna Gottipati, Minghui Qiu, Yanchuan Sim, Jing Jiang, and Noah A. Smith. 2013. Learning topics and positions from Debatepedia. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1858–1868.

Justin Grimmer and Brandon M Stewart. 2013. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, 21(3):267–297.

Dustin Hillard, Mari Ostendorf, and Elizabeth Shriberg. 2003. Detection of agreement vs. disagreement in meetings: Training with unlabeled data. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (Short Papers)*, pages 34–36.

Simon Hix and Abdul Noury. 2016. Government-opposition or left-right? the institutional determinants of voting in legislatures. *Political Science Research and Methods*, 4(02):249–273.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26.

Gary King, Patrick Lam, and Margaret Roberts. 2017. Computer-assisted keyword and document set discovery from unstructured text. *American Journal of Political Science*.

Anne Lauscher, Federico Nanni, Pablo Ruiz Fabo, and Simone Paolo Ponzetto. 2016. Entities as topic labels: combining entity linking and labeled lda to improve topic interpretability and evaluability. *IJ-Col: Italian journal of computational linguistics*, 2(2):67–88.

David Lazer, Alex Sandy Pentland, Lada Adamic, Sinan Aral, Albert Laszlo Barabasi, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, et al. 2009. Life in the network: the coming age of computational social science. *Science (New York, NY)*, 323(5915):721.

Stefano Menini and Sara Tonelli. 2016. Agreement and disagreement: Comparison of points of view in the political domain. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 2461–2470.

Slava Mikhaylov, Michael Laver, and Kenneth R Benoit. 2012. Coder reliability and misclassification in the human coding of party manifestos. *Political Analysis*, 20(1):78–91.

Giovanni Moretti, Rachele Sprugnoli, and Sara Tonelli. 2015. Digging in the dirt: Extracting keyphrases from texts with KD. In *Proceedings of the 2nd Italian Conference on Computational Linguistics*.

Federico Nanni, Cäcilia Zirn, Goran Glavaš, Jason Eichorst, and Simone Paolo Ponzetto. 2016. TopFish: Topic-based analysis of political position in US electoral campaigns. In *Proceedings of the International Conference on the Advances in Computational Analysis of Political Text*, pages 61–67.

Elin Naurin. 2011. *Election Promises, Party Behaviour and Voter Perceptions*. Palgrave Macmillan.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–43.

Yanchuan Sim, Brice Acree, Justin H Gross, and Noah A. Smith. 2013. Measuring ideological proportions in political speeches. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 91–101.

Jonathan B Slapin and Sven-Oliver Proksch. 2008. A scaling model for estimating time-series party positions from texts. *American Journal of Political Science*, 52(3):705–722.

Andrea Volkens, Onawa Lacewell, Pola Lehmann, Sven Regel, Henrike Schultze, and Annika Werner. 2011. The manifesto data collection. *Manifesto Project (MRG/CMP/MARPOR), Berlin: Wissenschaftszentrum Berlin für Sozialforschung (WZB)*.

John T Woolley and Gerhard Peters. 2008. The American presidency project.

Lori Young and Stuart Soroka. 2012. Affective news: The automated coding of sentiment in political texts. *Political Communication*, 29(2):205–231.

Căcilia Zirn, Goran Glavaš, Federico Nanni, Jason Eichorts, and Heiner Stuckenschmidt. 2016. Classifying topics and detecting topic shifts in political manifestos. In *Proceedings of the International Conference on the Advances in Computational Analysis of Political Text*, pages 88–93.

Zipporah: a Fast and Scalable Data Cleaning System for Noisy Web-Crawled Parallel Corpora

Hainan Xu **Philipp Koehn**
Department of Compute Science,
Center for Language and Speech Processing,
Johns Hopkins University, U.S.A., 21218
hxu31@jhu.edu phi@jhu.edu

Abstract

We introduce Zipporah, a fast and scalable data cleaning system. We propose a novel type of bag-of-words translation feature, and train logistic regression models to classify good data and synthetic noisy data in the proposed feature space. The trained model is used to score parallel sentences in the data pool for selection. As shown in experiments, Zipporah selects a high-quality parallel corpus from a large, mixed quality data pool. In particular, for one noisy dataset, Zipporah achieves a 2.1 BLEU score improvement with using 1/5 of the data over using the entire corpus.

1 Introduction

Statistical machine translation (SMT) systems require the use of parallel corpora for training the internal model parameters. Data quality is vital for the performance of the SMT system (Simard, 2014). To acquire a massive parallel corpus, many researchers have been using the Internet as a resource, but the quality of data acquired from the Internet usually has no guarantee, and data cleaning/data selection is needed before the data is used in actual systems. Usually data cleaning refers to getting rid of a small amount of very noisy data from a large data pool, and data selection refers to selecting a small subset of clean (or in-domain) data from the data pool; both have the objective of improving translation performances. For practical purposes, it is highly desirable to perform data selection in a very fast and scalable manner. In this paper we introduce Zipporah[1], a fast and scalable system which can select an arbitrary size of good data from a large noisy data pool to be used in SMT model training.

2 Prior Work

Many researchers have studied the data cleaning/selection problem. For data selection, there have been a lot of work on selecting a subset of data based on domain-matching. Duh et al. (2013) used a neural network based language model trained on a small in-domain corpus to select from a larger data pool. Moore and Lewis (2010) computed cross-entropy between in-domain and out-of-domain language models to select data for training language models. XenC (Rousseau, 2013), an open-source tool, also selects data based on cross-entropy scores on language models. Axelrod et al. (2015) utilized part-of-speech tags and used a class-based n-gram language model for selecting in-domain data. There are a few works that utilize other metrics. Lü et al. (2007) redistributed different weights for sentence pairs/predefined sub-models. Shah and Specia (2014) described experiments on quality estimation which, given a source sentence, select the best translation among several options. The qe-clean system (Denkowski et al., 2012; Dyer et al., 2010; Heafield, 2011) uses word alignments and language models to select sentence pairs that are likely to be good translations of one another.

For data cleaning, a lot of researchers worked on getting rid of noising data. Taghipour et al. (2011) proposed an outlier detection algorithm which leads to an improved translation quality when trimming a small portion of data. Cui et al. (2013) used a graph-based random walk algorithm to do bilingual data cleaning. BiTextor (Esplá-Gomis and Forcada, 2009) utilizes sentence alignment scores and source URL information to filter out bad URL pairs and selects good sentence pairs.

In this paper we propose a novel way to evaluate the quality of a sentence pair which runs efficiently. We do not make a clear distinction

[1] https://github.com/hainan-xv/zipporah

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2945–2950
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

between data selection and data cleaning in this work, because under different settings, our method can perform either based on the computed quality scores of sentence pairs.

3 Method

The method in this paper works as follows: we first map all sentence pairs into the proposed feature space, and then train a simple logistic regression model to separate known good data and (synthetic) bad data. Once the model is trained, it is used to score sentence pairs in the noisy data pool. Sentence pairs with better scores are added to the selected subset until the desired size constraint is met.

3.1 Features

Since good adequacy and fluency are the major two elements that constitute a good parallel sentence pair, we propose separate features to address both of them. For adequacy, we propose bag-of-words translation scores, and for fluency we use n-gram language model scores. For notational simplicity, in this section we assume the sentence pair is French-English in describing the features, and we will use subscripts f and e to indicate the languages. In designing the features, we prioritize efficiency as well as performance since we could be dealing with corpora of huge sizes.

3.1.1 Adequacy scores

We view each sentence as a bag of words, and design a "distance" between the sentence pairs based on a bag-of-words translation model. To do this, we first generate dictionaries from an aligned corpus, and represent them as sets of triplets. Formally,

$$D_{\text{f2e}} = \{(w_{f_i}, w_{e_i}, p(w_{e_i}|w_{f_i})), i = 1, ..., m\}.$$

Given a sentence pair (s_f, s_e) in the noisy data pool, we represent the two sentence as two sparse word-frequency vectors v_f and v_e. For example for any French word w_f, we have $v_f[w_f] = \frac{c(w_f, s_f)}{l(s_f)}$, where $c(w_f, s_f)$ is the number of occurrences of w_f in s_f and $l(s_f)$ is the length of s_f. We do the same for v_e. Notice that by construction, both vectors add up to 1 and represent a proper probability distribution on their respective vocabularies. Then we "translate" v_f into v'_e, based on the probabilistic f2e dictionary, where

$$v'_e[w_e] = \sum_{w_f} v_f[w_f] p(w_e|w_f)$$

For a French word w that does not appear in the dictionary, we keep it as it is in the translated vector, i.e. assume there is an entry of $(w, w, 1.0)$ in the dictionary. Since the dictionary is probabilistic, the elements in v'_e also add up to 1, and v'_e represents another probability distribution on the English vocabulary. We compute the (smoothed) cross-entropy between v_e and v'_e,

$$\text{xent}(v_e, v'_e) = \sum_{w_e} v_e[w_e] \log \frac{1}{v'_e[w_e] + c} \quad (1)$$

where c is a smoothing constant to prevent the denominator from being zero, and set $c = 0.0001$ for all experiments in this paper (more about this in Section 4).

We perform similar procedures for English-to-French, and compute $\text{xent}(v_f, v'_f)$. We define the adequacy score as the sum of the two:

$$\text{adequacy}(s_f, s_e) = \text{xent}(v_e, v'_e) + \text{xent}(v_f, v'_f)$$

3.1.2 Fluency scores

We train two n-gram language models with a clean French and English corpus, and then for each sentence pair (s_f, s_e), we score each sentence with the corresponding model, $\mathcal{F}_{\text{ngram}}(s_f)$ and $\mathcal{F}_{\text{ngram}}(s_e)$, each computed as the ratio between the sentence negative log-likelihood and the sentence length. We define the fluency score as the sum of the two:

$$\text{fluency}(s_f, s_e) = \mathcal{F}_{\text{ngram}}(s_f) + \mathcal{F}_{\text{ngram}}(s_e)$$

3.2 Synthetic noisy data generation

We generate synthetic noisy data from good data, and make sure the generated noisy data include sentence pairs with a) good fluency and bad adequacy, b) good adequacy and bad fluency and c) bad both.

Respectively, we generate 3 types of "noisy" sentence pairs from a good corpus: a) shuffle the sentences in the target language file (each sentence in the source language would be aligned to a random sentence in the target language); b) shuffle the words within each sentence (each sentence will be bad but the pairs are good translations in the "bag-of-words" sense); c) shuffle both the sentences and words. We emphasize that, while the synthetic data might not represent "real" noisy data, it has the following advantages: 1) each type of noisy data is equally represented so the classifier has to do well on all of them; 2) the data generated this way would be among the hardest to classify, especially type a and type b, so if a classifier separates such hard data with good performance, we expect it to also be able to do well in real world situations.

3.3 Logistic regression feature mapping

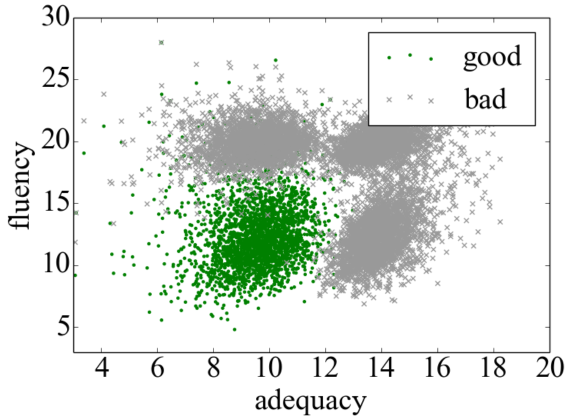


Figure 1: newstest09 fr-en data in the feature space

We plot the newstest09 data (original and auto-generated noisy ones as described in Section 3.2) into the proposed feature space in Figure 1. We observe that the clusters are quite separable, though the decision function would not be linear. We map the features into higher order forms of (x^n, y^n) in order for logistic regression to train a non-linear decision boundary.[2] We use $n = 8$ in this work since it gives the best classification performance on the newstest09 fr-en corpus.

4 Hyper-parameter Tuning

We conduct experiments to determine the value of the constant c in the smoothed cross-entropy computation in equation 1. We choose the newstest09 German-English corpus, and shuffle the sentences in the English file and combine the original (clean) corpus with the shuffled (noisy) corpus into a larger corpus, where half of them are good sentence pairs. We set different values of c and use the adequacy scores to pick the better half, and compute the retrieval accuracy. Table 1 shows that the best value for c is 0.0001, and we use that in all experiments.

c	accuracy
0.001	0.975
0.0001	**0.984**
0.00001	0.983
0.000001	0.981

Table 1: Tuning cross-entropy constant c

5 Evaluation

We evaluate Zipporah on 3 language pairs, French-English, German-English and Spanish-English. The noisy web-crawled data comes from an early version of `http://statmt.org/paracrawl`. The number of words are (in millions) 340, 487 and 70 respectively.

To generate the dictionaries for computing the adequacy scores, we use fast_align (Dyer et al., 2013) to align the Europarl (Koehn, 2005) corpus and generate probabilistic dictionaries from the alignments. We set the n-gram order to be 5 and use SRILM (Stolcke et al., 2011) to train language models on the Europarl corpus and generate the n-gram scores.

For each language pair, we use scikit-learn (Pedregosa et al., 2011) to train a logistic regression model to classify between the original and the synthetic noisy corpus of newstest09, and the trained model is used to score all sentence pairs in the data pool. We keep selecting the best ones until the desired number of words is reached.

To evaluate the quality, we train a Moses (Koehn et al., 2007) SMT system on selected data, and evaluate each trained SMT system on 3 test corpora: newstest2011 which contains 3003 sentence pairs, and a random subset of the TED-talks corpus and the movie-subtitle corpus from OPUS (Tiedemann, 2012), each of which contains 3000 sentence pairs.

Tables 2, 3 and 4 show the BLEU performance of the selected subsets of the Zipporah system compared to the baseline, which selects sentence pairs at random; for comparison, we also give the BLEU performance of systems trained on Europarl. The Zipporah system gives consistently better performance across multiple datasets and multiple languages than the baseline.[3]

[2] We avoid using multiple mappings of one feature because we want the scoring function to be monotonic both w.r.t x and y, which could break if we allow multiple higher-order mappings of the same feature and they end up with weights with different signs.

[3] We also point out that the performance of the selected

BLEU	newstest11		ted-talk		subtitle	
num-words	rand	zipp	rand	zipp	rand	zipp
10 million	21.5	24.4	24.0	27.4	12.3	14.9
20 million	22.8	25.1	25.0	27.9	12.8	15.5
50 million	24.3	26.0	27.4	28.8	14.5	15.8
100 million	25.2	26.6	28.3	**30.3**	15.0	**17.3**
200 million	26.1	**26.7**	29.9	30.0	16.4	**17.3**
340 mil (all)	26.2		30.0		16.7	
Europarl	24.4		27.0		14.2	

Table 2: BLEU Performance, French-English

BLEU	newstest11		ted-talk		subtitle	
num-words	rand	zipp	rand	zipp	rand	zipp
10 million	13.6	17.6	17.0	22.5	11.4	15.8
20 million	14.8	18.4	18.9	23.7	12.7	16.9
50 million	16.3	19.2	20.8	24.8	13.9	17.8
100 million	16.9	**19.5**	21.3	**25.0**	14.0	**18.3**
200 million	18.0	19.2	22.9	24.2	15.3	17.9
487 mil (all)	18.7		23.5		16.2	
Europarl	17.5		21.5		14.5	

Table 3: BLEU Performance, German-English

BLEU	newstest11		ted-talk		subtitle	
num-words	rand	zipp	rand	zipp	rand	zipp
10 million	24.2	25.5	25.9	28.3	17.9	19.8
20 million	25.3	26.2	28.2	29.7	19.3	21.2
50 million	26.6	26.5	29.9	**30.4**	21.3	21.4
70 mil (all)	**27.1**		30.3		**21.8**	
Europarl	25.4		28.4		19.8	

Table 4: BLEU Performance, Spanish-English

In particular, for the Germen-English corpus, when selecting less than 2% of the data (10 million words), on the TED-talk dataset, Zipporah achieves a 5.5 BLEU score improvement over the baseline; by selecting less than 4% of the data (20 million words) the system gives better performance than using all data. Peak performance is achieved when selecting 100 million words, where an improvement of 2.1 BLEU score over all data is achieved on the movie-subtitle dataset, despite only using less than 1/5 of the data.

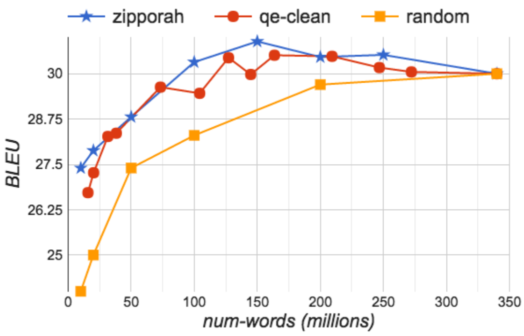


Figure 2: BLEU performance of Zipporah, qe-clean and random on TED-talks, French-English

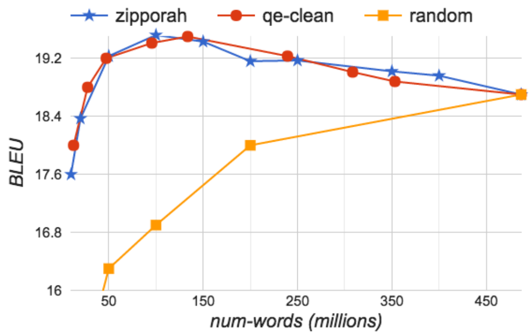


Figure 3: BLEU performance of Zipporah, qe-clean and random on newstest11, German-English

Figures 2, 3 and 4 compare the result of Zipporah with that of qe-clean (Denkowski et al., 2012; Dyer et al., 2010; Heafield, 2011) and the random baseline. We use the same data when running qe-clean, with Europarl for training and newstest09 for dev. While they both perform comparably and better than the baseline, Zipporah achieves a better peak in all the datasets, and the peak is usually achieved when selecting a smaller number of words compared to qe-clean, Another advantage of Zipporah is it allows the user to select an arbi-

subsets of the Zipporah system can surpass that of Europarl, although the Europarl corpus acts like an "oracle" in the system, upon which the dictionaries and language models for feature computations are trained.

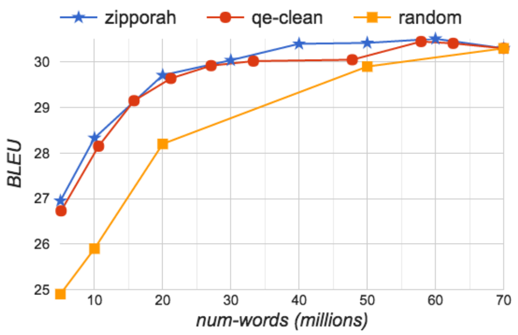


Figure 4: BLEU performance of Zipporah, qe-clean and random on TED-talks, Spanish-English

trary size from the pool.[4] We also want to emphasize that unlike qe-clean, which requires running word-alignments for all sentence pairs in the noisy corpus, Zipporah's feature computation is simple, fast and can easily be scaled for huge datasets.

6 Conclusion and Future Work

In this paper we introduced Zipporah, a fast data selection system for noisy parallel corpora. SMT results demonstrate that Zipporah can select a high-quality subset of the data and significantly improve SMT performance.

Zipporah currently selects sentences based on the "individual quality" only, and we plan in future work to also consider other factors, e.g. encourage selection of a subset that has a better n-gram coverage.

Acknowledgments

This project was funded by Google Faculty Research Award. The authors would like to thank Shuoyang Ding, Tongfei Chen, Matthew Wiesner, Winston Wu, Huda Khayrallah and Adi Renduchintala for their help during this project. The authors would also like to thank Penny Peng for her moral support.

References

Amittai Axelrod, Yogarshi Vyas, Marianna Martindale, Marine Carpuat, and Johns Hopkins. 2015. Class-based n-gram language difference models for data selection. In *IWSLT (International Workshop on Spoken Language Translation)*.

Lei Cui, Dongdong Zhang, Shujie Liu, Mu Li, and Ming Zhou. 2013. Bilingual data cleaning for smt using graph-based random walk. In *ACL (2)*, pages 340–345.

Michael Denkowski, Greg Hanneman, and Alon Lavie. 2012. The cmu-avenue french-english translation system. In *Proceedings of the NAACL 2012 Workshop on Statistical Machine Translation*.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. 2013. Adaptation data selection using neural language models: Experiments in machine translation. In *ACL (2)*, pages 678–683.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. Association for Computational Linguistics.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Miquel Esplá-Gomis and M Forcada. 2009. Bitextor, a free/open-source software to harvest translation memories from multilingual websites. *Proceedings of MT Summit XII, Ottawa, Canada. Association for Machine Translation in the Americas*.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Yajuan Lü, Jin Huang, and Qun Liu. 2007. Improving statistical machine translation performance by training data selection and optimization. In *EMNLP-CoNLL*, volume 34, pages 3–350.

Robert C Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 conference short papers*, pages 220–224. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Anthony Rousseau. 2013. Xenc: An open-source tool for data selection in natural language processing. *The Prague Bulletin of Mathematical Linguistics*, 100:73–82.

Kashif Shah and Lucia Specia. 2014. Quality estimation for translation selection. In *Proceedings of the 17th Annual Conference of the European Association for Machine Translation, Dubrovnik, Croatia*.

Michel Simard. 2014. Clean data for training statistical mt: The case of mt contamination. In *Proceedings of the 11th Conference of the Association for Machine Translation in the Americas (AMTA)*.

[4] In the plots the data points of Zipporah and qe-clean are not aligned because we always select multiples of million words, but it is hard to do so with qe-clean.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. Srilm at sixteen: Update and outlook. In *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, volume 5.

Kaveh Taghipour, Shahram Khadivi, and Jia Xu. 2011. Parallel corpus refinement as an outlier detection algorithm. *Proceedings of the 13th Machine Translation Summit (MT Summit XIII)*, pages 414–421.

Jrg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Bringing Structure into Summaries:
Crowdsourcing a Benchmark Corpus of Concept Maps

Tobias Falke and **Iryna Gurevych**
Research Training Group AIPHES and UKP Lab
Department of Computer Science, Technische Universität Darmstadt
https://www.aiphes.tu-darmstadt.de

Abstract

Concept maps can be used to concisely represent important information and bring structure into large document collections. Therefore, we study a variant of multi-document summarization that produces summaries in the form of concept maps. However, suitable evaluation datasets for this task are currently missing. To close this gap, we present a newly created corpus of concept maps that summarize heterogeneous collections of web documents on educational topics. It was created using a novel crowdsourcing approach that allows us to efficiently determine important elements in large document collections. We release the corpus along with a baseline system and proposed evaluation protocol to enable further research on this variant of summarization.[1]

1 Introduction

Multi-document summarization (MDS), the transformation of a set of documents into a short text containing their most important aspects, is a long-studied problem in NLP. Generated summaries have been shown to support humans dealing with large document collections in information seeking tasks (McKeown et al., 2005; Maña-López et al., 2004; Roussinov and Chen, 2001). However, when exploring a set of documents manually, humans rarely write a fully-formulated summary for themselves. Instead, user studies (Chin et al., 2009; Kang et al., 2011) show that they note down important keywords and phrases, try to identify relationships between them and organize them accordingly. Therefore, we believe that the study of summarization with similarly structured outputs is an important extension of the traditional task.

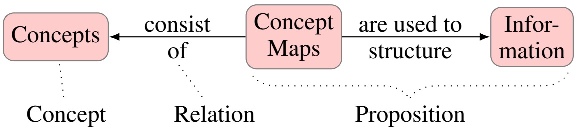


Figure 1: Elements of a concept map.

A representation that is more in line with observed user behavior is a *concept map* (Novak and Gowin, 1984), a labeled graph showing concepts as nodes and relationships between them as edges (Figure 1). Introduced in 1972 as a teaching tool (Novak and Cañas, 2007), concept maps have found many applications in education (Edwards and Fraser, 1983; Roy, 2008), for writing assistance (Villalon, 2012) or to structure information repositories (Briggs et al., 2004; Richardson and Fox, 2005). For summarization, concept maps make it possible to represent a summary concisely and clearly reveal relationships. Moreover, we see a second interesting use case that goes beyond the capabilities of textual summaries: When concepts and relations are linked to corresponding locations in the documents they have been extracted from, the graph can be used to navigate in a document collection, similar to a table of contents. An implementation of this idea has been recently described by Falke and Gurevych (2017).

The corresponding task that we propose is *concept-map-based MDS*, the summarization of a document cluster in the form of a concept map. In order to develop and evaluate methods for the task, gold-standard corpora are necessary, but no suitable corpus is available. The manual creation of such a dataset is very time-consuming, as the annotation includes many subtasks. In particular, an annotator would need to manually identify all concepts in the documents, while only a few of them will eventually end up in the summary.

[1] Available at https://github.com/UKPLab/emnlp2017-cmapsum-corpus

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2951–2961
Copenhagen, Denmark, September 7–11, 2017.

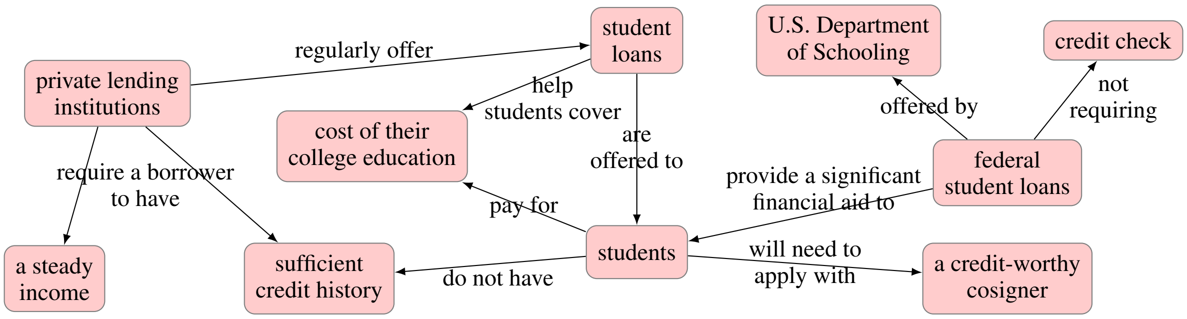


Figure 2: Excerpt from a summary concept map on the topic "*students loans without credit history*".

To overcome these issues, we present a corpus creation method that effectively combines automatic preprocessing, scalable crowdsourcing and high-quality expert annotations. Using it, we can avoid the high effort for single annotators, allowing us to scale to document clusters that are 15 times larger than in traditional summarization corpora. We created a new corpus of 30 topics, each with around 40 source documents on educational topics and a summarizing concept map that is the consensus of many crowdworkers (see Figure 2).

As a crucial step of the corpus creation, we developed a new crowdsourcing scheme called *low-context importance annotation*. In contrast to traditional approaches, it allows us to determine important elements in a document cluster without requiring annotators to read all documents, making it feasible to crowdsource the task and overcome quality issues observed in previous work (Lloret et al., 2013). We show that the approach creates reliable data for our focused summarization scenario and, when tested on traditional summarization corpora, creates annotations that are similar to those obtained by earlier efforts.

To summarize, we make the following contributions: (1) We propose a novel task, concept-map-based MDS (§2), (2) present a new crowdsourcing scheme to create reference summaries (§4), (3) publish a new dataset for the proposed task (§5) and (4) provide an evaluation protocol and baseline (§7). We make these resources publicly available under a permissive license.

2 Task

Concept-map-based MDS is defined as follows: *Given a set of related documents, create a concept map that represents its most important content, satisfies a specified size limit and is connected.*

We define a concept map as a labeled graph showing concepts as nodes and relationships between them as edges. Labels are arbitrary sequences of tokens taken from the documents, making the summarization task extractive. A concept can be an entity, abstract idea, event or activity, designated by its unique label. Good maps should be propositionally coherent, meaning that every relation together with the two connected concepts form a meaningful proposition.

The task is complex, consisting of several interdependent subtasks. One has to extract appropriate labels for concepts and relations and recognize different expressions that refer to the same concept across multiple documents. Further, one has to select the most important concepts and relations for the summary and finally organize them in a graph satisfying the connectedness and size constraints.

3 Related Work

Some attempts have been made to automatically construct concept maps from text, working with either single documents (Zubrinic et al., 2015; Villalon, 2012; Valerio and Leake, 2006; Kowata et al., 2010) or document clusters (Qasim et al., 2013; Zouaq and Nkambou, 2009; Rajaraman and Tan, 2002). These approaches extract concept and relation labels from syntactic structures and connect them to build a concept map. However, common task definitions and comparable evaluations are missing. In addition, only a few of them, namely Villalon (2012) and Valerio and Leake (2006), define summarization as their goal and try to compress the input to a substantially smaller size. Our newly proposed task and the created large-cluster dataset fill these gaps as they emphasize the summarization aspect of the task.

For the subtask of selecting summary-worthy concepts and relations, techniques developed for traditional summarization (Nenkova and McKeown, 2011) and keyphrase extraction (Hasan and Ng, 2014) are related and applicable. Approaches

Imagine you want to learn something about **students loans without credit history**.
How useful would the following statements be for you?

(P1) students with bad credit history - apply for - federal loans with the FAFSA
□ Extremely Important □ Very Important □ Moderately Important □ Slightly Important □ Not at all Important

(P2) students - encounter - unforeseen financial emergencies
□ Extremely Important □ Very Important □ Moderately Important □ Slightly Important □ Not at all Important

Figure 3: Likert-scale crowdsourcing task with topic description and two example propositions.

that build graphs of propositions to create a summary (Fang et al., 2016; Li et al., 2016; Liu et al., 2015; Li, 2015) seem to be particularly related, however, there is one important difference: While they use graphs as an intermediate representation from which a textual summary is then generated, the goal of the proposed task is to create a graph that is directly interpretable and useful for a user. In contrast, these intermediate graphs, e.g. AMR, are hardly useful for a typical, non-linguist user.

For traditional summarization, the most well-known datasets emerged out of the DUC and TAC competitions.[2] They provide clusters of news articles with gold-standard summaries. Extending these efforts, several more specialized corpora have been created: With regard to size, Nakano et al. (2010) present a corpus of summaries for large-scale collections of web pages. Recently, corpora with more heterogeneous documents have been suggested, e.g. (Zopf et al., 2016) and (Benikova et al., 2016). The corpus we present combines these aspects, as it has large clusters of heterogeneous documents, and provides a necessary benchmark to evaluate the proposed task.

For concept map generation, one corpus with human-created summary concept maps for student essays has been created (Villalon et al., 2010). In contrast to our corpus, it only deals with single documents, requires a two orders of magnitude smaller amount of compression of the input and is not publicly available .

Other types of information representation that also model concepts and their relationships are knowledge bases, such as Freebase (Bollacker et al., 2009), and ontologies. However, they both differ in important aspects: Whereas concept maps follow an open label paradigm and are meant to be interpretable by humans, knowledge bases and ontologies are usually more strictly typed and made to be machine-readable. Moreover, approaches to automatically construct them from text typically try to extract as much information as possible, while we want to summarize a document.

4 Low-Context Importance Annotation

Lloret et al. (2013) describe several experiments to crowdsource reference summaries. Workers are asked to read 10 documents and then select 10 summary sentences from them for a reward of $0.05. They discovered several challenges, including poor work quality and the subjectiveness of the annotation task, indicating that crowdsourcing is not useful for this purpose.

To overcome these issues, we introduce a new task design, *low-context importance annotation*, to determine summary-worthy parts of documents. Compared to Lloret et al.'s approach, it is more in line with crowdsourcing best practices, as the tasks are simple, intuitive and small (Sabou et al., 2014) and workers receive reasonable payment (Fort et al., 2011). Most importantly, it is also much more efficient and scalable, as it does not require workers to read all documents in a cluster.

4.1 Task Design

We break down the task of importance annotation to the level of single propositions. The goal of our crowdsourcing scheme is to obtain a score for each proposition indicating its importance in a document cluster, such that a ranking according to the score would reveal what is most important and should be included in a summary. In contrast to other work, we do not show the documents to the workers at all, but provide only a description of the document cluster's topic along with the propositions. This ensures that tasks are small, simple and can be done quickly (see Figure 3).

In preliminary tests, we found that this design, despite the minimal context, works reasonably on our focused clusters on common educational topics. For instance, consider Figure 3: One can easily say that P1 is more important than P2 without reading the documents.

[2] duc.nist.gov, tac.nist.gov

We distinguish two task variants:

Likert-scale Tasks Instead of enforcing binary importance decisions, we use a 5-point Likert-scale to allow more fine-grained annotations. The obtained labels are translated into scores (5..1) and the average of all scores for a proposition is used as an estimate for its importance. This follows the idea that while single workers might find the task subjective, the consensus of multiple workers, represented in the average score, tends to be less subjective due to the "wisdom of the crowd". We randomly group five propositions into a task.

Comparison Tasks As an alternative, we use a second task design based on pairwise comparisons. Comparisons are known to be easier to make and more consistent (Belz and Kow, 2010), but also more expensive, as the number of pairs grows quadratically with the number of objects.[3] To reduce the cost, we group five propositions into a task and ask workers to order them by importance per drag-and-drop. From the results, we derive pairwise comparisons and use TrueSkill (Herbrich et al., 2007), a powerful Bayesian rank induction model (Zhang et al., 2016), to obtain importance estimates for each proposition.

4.2 Pilot Study

To verify the proposed approach, we conducted a pilot study on Amazon Mechanical Turk using data from TAC2008 (Dang and Owczarzak, 2008). We collected importance estimates for 474 propositions extracted from the first three clusters[4] using both task designs. Each Likert-scale task was assigned to 5 different workers and awarded $0.06. For comparison tasks, we also collected 5 labels each, paid $0.05 and sampled around 7% of all possible pairs. We submitted them in batches of 100 pairs and selected pairs for subsequent batches based on the confidence of the TrueSkill model.

Quality Control Following the observations of Lloret et al. (2013), we established several measures for quality control. First, we restricted our tasks to workers from the US with an approval rate of at least 95%. Second, we identified low quality workers by measuring the correlation of each worker's Likert-scores with the average of the other four scores. The worst workers (at most 5% of all labels) were removed.

Peer Scoring	Pearson	Spearman
Modified Pyramid	0.4587	0.4676
ROUGE-2	0.3062	0.3486
Crowd-Likert	0.4589	0.4196
Crowd-Comparison	0.4564	0.3761

Table 1: Correlation of peer scores with manual responsiveness scores on TAC2008 topics 01-03.

In addition, we included trap sentences, similar as in (Lloret et al., 2013), in around 80 of the tasks. In contrast to Lloret et al.'s findings, both an obvious trap sentence (*This sentence is not important*) and a less obvious but unimportant one (*Barack Obama graduated from Harvard Law*) were consistently labeled as unimportant (1.08 and 1.14), indicating that the workers did the task properly.

Agreement and Reliability For Likert-scale tasks, we follow Snow et al. (2008) and calculate agreement as the average Pearson correlation of a worker's Likert-score with the average score of the remaining workers.[5] This measure is less strict than exact label agreement and can account for close labels and high- or low-scoring workers. We observe a correlation of 0.81, indicating substantial agreement. For comparisons, the majority agreement is 0.73. To further examine the reliability of the collected data, we followed the approach of Kiritchenko and Mohammed (2016) and simply repeated the crowdsourcing for one of the three topics. Between the importance estimates calculated from the first and second run, we found a Pearson correlation of 0.82 (Spearman 0.78) for Likert-scale tasks and 0.69 (Spearman 0.66) for comparison tasks. This shows that the approach, despite the subjectiveness of the task, allows us to collect reliable annotations.

Peer Evaluation In addition to the reliability studies, we extrinsically evaluated the annotations in the task of summary evaluation. For each of the 58 peer summaries in TAC2008, we calculated a score as the sum of the importance estimates of the propositions it contains. Table 1 shows how these peer scores, averaged over the three topics, correlate with the manual responsiveness scores assigned during TAC in comparison

[3] Even with intelligent sampling strategies, such as the active learning in CrowdBT (Chen et al., 2013), the number of pairs is only reduced by a constant factor (Zhang et al., 2016).

[4] D0801A-A, D0802A-A, D0803A-A

[5] As workers are not consistent across all items, we create five meta-workers by sorting the labels per proposition.

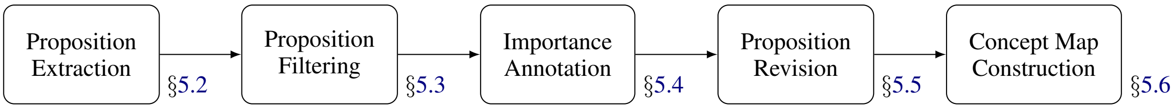


Figure 4: Steps of the corpus creation (with references to the corresponding sections).

to ROUGE-2 and Pyramid scores.[6] The results demonstrate that with both task designs, we obtain importance annotations that are similarly useful for summary evaluation as pyramid annotations or gold-standard summaries (used for ROUGE).

Conclusion Based on the pilot study, we conclude that the proposed crowdsourcing scheme allows us to obtain proper importance annotations for propositions. As workers are not required to read all documents, the annotation is much more efficient and scalable as with traditional methods.

5 Corpus Creation

This section presents the corpus construction process, as outlined in Figure 4, combining automatic preprocessing, scalable crowdsourcing and high-quality expert annotations to be able to scale to the size of our document clusters. For every topic, we spent about $150 on crowdsourcing and 1.5h of expert annotations, while just a single annotator would already need over 8 hours (at 200 words per minute) to read all documents of a topic.

5.1 Source Data

As a starting point, we used the DIP corpus (Habernal et al., 2016), a collection of 49 clusters of 100 web pages on educational topics (e.g. bullying, homeschooling, drugs) with a short description of each topic. It was created from a large web crawl using state-of-the-art information retrieval. We selected 30 of the topics for which we created the necessary concept map annotations.

5.2 Proposition Extraction

As concept maps consist of propositions expressing the relation between concepts (see Figure 1), we need to impose such a structure upon the plain text in the document clusters. This could be done by manually annotating spans representing concepts and relations, however, the size of our clusters makes this a huge effort: 2288 sentences per topic (69k in total) need to be processed. Therefore, we resort to an automatic approach.

The Open Information Extraction paradigm (Banko et al., 2007) offers a representation very similar to the desired one. For instance, from

Students with bad credit history should not lose hope and apply for federal loans with the FAFSA.

Open IE systems extract tuples of two arguments and a relation phrase representing propositions:

(s. with bad credit history, should not lose, hope)
(s. with bad credit history, apply for, federal loans with the FAFSA)

While the relation phrase is similar to a relation in a concept map, many arguments in these tuples represent useful concepts. We used Open IE 4[7], a state-of-the-art system (Stanovsky and Dagan, 2016) to process all sentences. After removing duplicates, we obtained 4137 tuples per topic.

Since we want to create a gold-standard corpus, we have to ensure that we produce high-quality data. We therefore made use of the confidence assigned to every extracted tuple to filter out low quality ones. To ensure that we do not filter too aggressively (and miss important aspects in the final summary), we manually annotated 500 tuples sampled from all topics for correctness. On the first 250 of them, we tuned the filter threshold to 0.5, which keeps 98.7% of the correct extractions in the unseen second half. After filtering, a topic had on average 2850 propositions (85k in total).

5.3 Proposition Filtering

Despite the similarity of the Open IE paradigm, not every extracted tuple is a suitable proposition for a concept map. To reduce the effort in the subsequent steps, we therefore want to filter out unsuitable ones. A tuple is suitable if it (1) is a correct extraction, (2) is meaningful without any context and (3) has arguments that represent proper concepts. We created a guideline explaining when to label a tuple as suitable for a concept map and performed a small annotation study. Three annotators independently labeled 500 randomly sam-

[6] Correlations for ROUGE and Pyramid are lower than reported in TAC since we only use 3 topics instead of all 48.

[7] `https://github.com/knowitall/openie`

pled tuples. The agreement was 82% ($\kappa = 0.60$). We found tuples to be unsuitable mostly because they had unresolvable pronouns, conflicting with (2), or arguments that were full clauses or propositions, conflicting with (3), while (1) was mostly taken care of by the confidence filtering in §5.2.

Due to the high number of tuples we decided to automate the filtering step. We trained a linear SVM on the majority voted annotations. As features, we used the extraction confidence, length of arguments and relations as well as part-of-speech tags, among others. To ensure that the automatic classification does not remove suitable propositions, we tuned the classifier to avoid false negatives. In particular, we introduced class weights, improving precision on the negative class at the cost of a higher fraction of positive classifications. Additionally, we manually verified a certain number of the most uncertain negative classifications to further improve performance. When 20% of the classifications are manually verified and corrected, we found that our model trained on 350 labeled instances achieves 93% precision on negative classifications on the unseen 150 instances. We found this to be a reasonable trade-off of automation and data quality and applied the model to the full dataset.

The classifier filtered out 43% of the propositions, leaving 1622 per topic. We manually examined the 17k least confident negative classifications and corrected 955 of them. We also corrected positive classifications for certain types of tuples for which we knew the classifier to be imprecise. Finally, each topic was left with an average of 1554 propositions (47k in total).

5.4 Importance Annotation

Given the propositions identified in the previous step, we now applied our crowdsourcing scheme as described in §4 to determine their importance. To cope with the large number of propositions, we combine the two task designs: First, we collect Likert-scores from 5 workers for each proposition, clean the data and calculate average scores. Then, using only the top 100 propositions[8] according to these scores, we crowdsource 10% of all possible pairwise comparisons among them. Using TrueSkill, we obtain a fine-grained ranking of the 100 most important propositions.

For Likert-scores, the average agreement over all topics is 0.80, while the majority agreement for comparisons is 0.78. We repeated the data collection for three randomly selected topics and found the Pearson correlation between both runs to be 0.73 (Spearman 0.73) for Likert-scores and 0.72 (Spearman 0.71) for comparisons. These figures show that the crowdsourcing approach works on this dataset as reliably as on the TAC documents.

In total, we uploaded 53k scoring and 12k comparison tasks to Mechanical Turk, spending \$4425.45 including fees. From the fine-grained ranking of the 100 most important propositions, we select the top 50 per topic to construct a summary concept map in the subsequent steps.

5.5 Proposition Revision

Having a manageable number of propositions, an annotator then applied a few straightforward transformations that correct common errors of the Open IE system. First, we break down propositions with conjunctions in either of the arguments into separate propositions per conjunct, which the Open IE system sometimes fails to do. And second, we correct span errors that might occur in the argument or relation phrases, especially when sentences were not properly segmented. As a result, we have a set of high quality propositions for our concept map, consisting of, due to the first transformation, 56.1 propositions per topic on average.

5.6 Concept Map Construction

In this final step, we connect the set of important propositions to form a graph. For instance, given the following two propositions

(student, may borrow, Stafford Loan)
(the student, does not have, a credit history)

one can easily see, although the first arguments differ slightly, that both labels describe the concept *student*, allowing us to build a concept map with the concepts *student*, *Stafford Loan* and *credit history*. The annotation task thus involves deciding which of the available propositions to include in the map, which of their concepts to merge and, when merging, which of the available labels to use. As these decisions highly depend upon each other and require context, we decided to use expert annotators rather than crowdsource the subtasks.

Annotators were given the topic description and the most important, ranked propositions. Using

[8] We also add all propositions with the same score as the 100th, yielding 112 propositions on average.

Corpus	Cluster	Cluster Size	Docs	Doc. Size	Rel. Std.
This work	30	97,880 ± 50,086.2	40.5 ± 6.8	2,412.8 ± 3,764.1	1.56
DUC 2006	50	17,461 ± 6,627.8	25.0 ± 0.0	729.2 ± 542.3	0.74
DUC 2004	50	6,721 ± 3,017.9	10.0 ± 0.0	672.1 ± 506.3	0.75
TAC 2008A	48	5,892 ± 2,832.4	10.0 ± 0.0	589.2 ± 480.3	0.82

Table 2: Topic clusters in comparison to classic corpora (size in token, mean with standard deviation).

a simple annotation tool providing a visualization of the graph, they could connect the propositions step by step. They were instructed to reach a size of 25 concepts, the recommended maximum size for a concept map (Novak and Cañas, 2007). Further, they should prefer more important propositions and ensure connectedness. When connecting two propositions, they were asked to keep the concept label that was appropriate for both propositions. To support the annotators, the tool used ADW (Pilehvar et al., 2013), a state-of-the-art approach for semantic similarity, to suggest possible connections. The annotation was carried out by graduate students with a background in NLP after receiving an introduction into the guidelines and tool and annotating a first example.

If an annotator was not able to connect 25 concepts, she was allowed to create up to three synthetic relations with freely defined labels, making the maps slightly abstractive. On average, the constructed maps have 0.77 synthetic relations, mostly connecting concepts whose relation is too obvious to be explicitly stated in text (e.g. between *Montessori teacher* and *Montessori education*).

To assess the reliability of this annotation step, we had the first three maps created by two annotators. We casted the task of selecting propositions to be included in the map as a binary decision task and observed an agreement of 84% ($\kappa = 0.66$). Second, we modeled the decision which concepts to join as a binary decision on all pairs of common concepts, observing an agreement of 95% ($\kappa = 0.70$). And finally, we compared which concept labels the annotators decided to include in the final map, observing 85% ($\kappa = 0.69$) agreement. Hence, the annotation shows substantial agreement (Landis and Koch, 1977).

6 Corpus Analysis

In this section, we describe our newly created corpus, which, in addition to having summaries in the form of concept maps, differs from traditional summarization corpora in several aspects.

6.1 Document Clusters

Size The corpus consists of document clusters for 30 different topics. Each of them contains around 40 documents with on average 2413 tokens, which leads to an average cluster size of 97,880 token. With these characteristics, the document clusters are 15 times larger than typical DUC clusters of ten documents and five times larger than the 25-document-clusters (Table 2). In addition, the documents are also more variable in terms of length, as the (length-adjusted) standard deviation is twice as high as in the other corpora. With these properties, the corpus represents an interesting challenge towards real-world application scenarios, in which users typically have to deal with much more than ten documents.

Genres Because we used a large web crawl as the source for our corpus, it contains documents from a variety of genres. To further analyze this property, we categorized a sample of 50 documents from the corpus. Among them, we found professionally written articles and blog posts (28%), educational material for parents and kids (26%), personal blog posts (16%), forum discussions and comments (12%), commented link collections (12%) and scientific articles (6%).

Textual Heterogeneity In addition to the variety of genres, the documents also differ in terms of language use. To capture this property, we follow Zopf et al. (2016) and compute, for every topic, the average Jensen-Shannon divergence between the word distribution of one document and the word distribution in the remaining documents. The higher this value is, the more the language differs between documents. We found the average divergence over all topics to be 0.3490, whereas it is 0.3019 in DUC 2004 and 0.3188 in TAC 2008A.

6.2 Concept Maps

As Table 3 shows, each of the 30 reference concept maps has exactly 25 concepts and between 24 and 28 relations. Labels for both concepts and

	per Map	Token	Character
Concepts	25.0 ± 0.0	3.2 ± 0.5	22.0 ± 4.1
Relations	25.2 ± 1.3	3.2 ± 0.5	17.1 ± 2.6

Table 3: Size of concept maps (mean with std).

relations consist on average of 3.2 tokens, whereas the latter are a bit shorter in characters.

To obtain a better picture of what kind of text spans have been used as labels, we automatically tagged them with their part-of-speech and determined their head with a dependency parser. Concept labels tend to be headed by nouns (82%) or verbs (15%), while they also contain adjectives, prepositions and determiners. Relation labels, on the other hand, are almost always headed by a verb (94%) and contain prepositions, nouns and particles in addition. These distributions are very similar to those reported by Villalon et al. (2010) for their (single-document) concept map corpus.

Analyzing the graph structure of the maps, we found that all of them are connected. They have on average 7.2 central concepts with more than one relation, while the remaining ones occur in only one proposition. We found that achieving a higher number of connections would mean compromising importance, i.e. including less important propositions, and decided against it.

7 Baseline Experiments

In this section, we briefly describe a baseline and evaluation scripts that we release, with a detailed documentation, along with the corpus.

Baseline Method We implemented a simple approach inspired by previous work on concept map generation and keyphrase extraction. For a document cluster, it performs the following steps:

1. Extract all NPs as potential concepts.
2. Merge potential concepts whose labels match after stemming into a single concept.
3. For each pair of concepts co-occurring in a sentence, select the tokens in between as a potential relation if they contain a verb.
4. If a pair of concepts has more than one relation, select the one with the shortest label.
5. Assign an importance score to every concept and rank them accordingly.
6. Find a connected graph of 25 concepts with high scores among all extracted concepts and relations.

Metric	Pr	Re	F1
Strict Match	.0006	.0026	.0010
METEOR	.1512	.1949	.1700
ROUGE-2	.0603	.1798	.0891

Table 4: Baseline performance on test set.

For (5), we trained a binary classifier to identify the important concepts in the set of all potential concepts. We used common features for keyphrase extraction, including position, frequency and length, and Weka's Random Forest (Hall et al., 2009) implementation as the model. At inference time, we use the classifiers confidence for a positive classification as the score.

In step (6), we start with the full graph of all extracted concepts and relations and use a heuristic to find a subgraph that is connected, satisfies the size limit of 25 concepts and has many high-scoring concepts: We iteratively remove the weakest concept until only one connected component of 25 concepts or less remains, which is used the summary concept map. This approach guarantees that the concept map is connected, but might not find the subset of concepts that has the highest total importance score.

Evaluation Metrics In order to automatically compare generated concept maps with reference maps, we propose three metrics.[9] As a concept map is fully defined by the set of its propositions, we can compute precision, recall and F1-scores between the two proposition set of generated and reference map. A proposition is represented as the concatenation of concept and relation labels. *Strict Match* compares them after stemming and only counts exact and complete matches. Using *METEOR* (Denkowski and Lavie, 2014), we offer a second metric that takes synonyms and paraphrases into account and also scores partial matches. And finally, we compute *ROUGE-2* (Lin, 2004) between the concatenation of all propositions from the maps. These automatic measures might be complemented with a human evaluation.

Results Table 4 shows the performance of the baseline. An analysis of the single pipeline steps

[9] For precise definitions of the metrics, please refer to the published scripts and accompanying documentation.

revealed major bottlenecks of the method and challenges of the task. First, we observed that around 76% of gold concepts are covered by the extraction (step 1+2), while the top 25 concepts (step 5) only contain 17% of the gold concepts. Hence, content selection is a major challenge, stemming from the large cluster sizes in the corpus. Second, while also 17% of gold concepts are contained in the final maps (step 6), scores for strict proposition matching are low, indicating a poor performance of the relation extraction (step 3). The propagation of these errors along the pipeline contributes to overall low scores.

8 Conclusion

In this work, we presented low-context importance annotation, a novel crowdsourcing scheme that we used to create a new benchmark corpus for concept-map-based MDS. The corpus has large-scale document clusters of heterogeneous web documents, posing a challenging summarization task. Together with the corpus, we provide implementations of a baseline method and evaluation scripts and hope that our efforts facilitate future research on this variant of summarization.

Acknowledgments

We would like to thank Teresa Botschen, Andreas Hanselowski and Markus Zopf for their help with the annotation work and Christian Meyer for his valuable feedback. This work has been supported by the German Research Foundation as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1.

References

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open Information Extraction from the Web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2670–2676, Hyderabad, India.

Anja Belz and Eric Kow. 2010. Comparing Rating Scales and Preference Judgements in Language Evaluation. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 7–16, Trim, Ireland.

Darina Benikova, Margot Mieskes, Christian M. Meyer, and Iryna Gurevych. 2016. Bridging the gap between extractive and abstractive summaries: Creation and evaluation of coherent extracts from heterogeneous sources. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 1039–1050, Osaka, Japan.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2009. Freebase. In *Compilation Proceedings of the International Conference on Management Data & 27th Symposium on Principles of Database Systems*, pages 1247–1250, Vancouver, Canada.

Geoffrey Briggs, David A. Shamma, Alberto J. Cañas, Roger Carff, Jeffrey Scargle, and Joseph D. Novak. 2004. Concept Maps Applied to Mars Exploration Public Outreach. In *Concept Maps: Theory, Methodology, Technology. Proceedings of the First International Conference on Concept Mapping*, pages 109–116, Pamplona, Spain.

Xi Chen, Paul N. Bennett, Kevyn Collins-Thompson, and Eric Horvitz. 2013. Pairwise Ranking Aggregation in a Crowdsourced Setting. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pages 193–202, Rome, Italy.

George Chin, Olga A. Kuchar, and Katherine E. Wolf. 2009. Exploring the Analytical Processes of Intelligence Analysts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 11–20, Boston, MA, USA.

Hoa Trang Dang and Karolina Owczarzak. 2008. Overview of the TAC 2008 Update Summarization Task. In *Proceedings of the First Text Analysis Conference*, pages 1–16, Gaithersburg, MD, USA.

Michael Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA.

John Edwards and Kym Fraser. 1983. Concept Maps as Reflectors of Conceptual Understanding. *Research in Science Education*, 13(1):19–26.

Tobias Falke and Iryna Gurevych. 2017. GraphDocExplore: A Framework for the Experimental Comparison of Graph-based Document Exploration Techniques. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark.

Yimai Fang, Haoyue Zhu, Ewa Muszyńska, Alexander Kuhnle, and Simone Teufel. 2016. A Proposition-Based Abstractive Summariser. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 567–578, Osaka, Japan.

Karën Fort, Gilles Adda, and K. Bretonnel Cohen. 2011. Amazon Mechanical Turk: Gold Mine or Coal Mine? *Computational Linguistics*, 37(2):413–420.

Ivan Habernal, Maria Sukhareva, Fiana Raiber, Anna Shtok, Oren Kurland, Hadar Ronen, Judit Bar-Ilan, and Iryna Gurevych. 2016. New Collection Announcement: Focused Retrieval Over the Web. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 701–704, Pisa, Italy.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1):10–18.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic Keyphrase Extraction: A Survey of the State of the Art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1262–1273, Baltimore, MD, USA.

Ralf Herbrich, Tom Minka, and Thore Graepel. 2007. TrueSkill(TM): A Bayesian Skill Rating System. In *Advances in Neural Information Processing Systems 19*, pages 569–576, Vancouver, Canada.

Youn-Ah Kang, Carsten Görg, and John T. Stasko. 2011. How Can Visual Analytics Assist Investigative Analysis? Design Implications from an Evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):570–583.

Svetlana Kiritchenko and Saif M. Mohammed. 2016. Capturing Reliable Fine-Grained Sentiment Associations by Crowdsourcing and Best-Worst Scaling. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 811–817, San Diego, CA, USA.

Juliana H. Kowata, Davidson Cury, and Maria Claudia Silva Boeres. 2010. Concept Maps Core Elements Candidates Recognition from Text. In *Concept Maps: Making Learning Meaningful. Proceedings of the 4th International Conference on Concept Mapping*, pages 120–127, Vina del Mar, Chile.

J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–174.

Wei Li. 2015. Abstractive Multi-document Summarization with Semantic Information Extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1908–1913, Lisbon, Portugal.

Wei Li, Lei He, and Hai Zhuge. 2016. Abstractive News Summarization based on Event Semantic Link Network. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 236–246, Osaka, Japan.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. 2015. Toward Abstractive Summarization Using Semantic Representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086, Denver, Colorado.

Elena Lloret, Laura Plaza, and Ahmet Aker. 2013. Analyzing the capabilities of crowdsourcing services for text summarization. *Language Resources and Evaluation*, 47(2):337–369.

Manuel J. Maña-López, Manuel de Buenaga, and José M. Gómez-Hidalgo. 2004. Multidocument Summarization: An Added Value to Clustering in Interactive Retrieval. *ACM Transactions on Information Systems*, 22(2):215–241.

Kathleen McKeown, Rebecca J. Passonneau, David K. Elson, Ani Nenkova, and Julia Hirschberg. 2005. Do summaries help? A Task-Based Evaluation of Multi-Document Summarization. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 210–217, Salvador, Brazil.

Masahiro Nakano, Hideyuki Shibuki, Rintaro Miyazaki, Madoka Ishioroshi, Koichi Kaneko, and Tatsunori Mori. 2010. Construction of Text Summarization Corpus for the Credibility of Information on the Web. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 3125–3131, Valletta, Malta.

Ani Nenkova and Kathleen R. McKeown. 2011. Automatic Summarization. *Foundations and Trends in Information Retrieval*, 5(2):103–233.

Joseph D. Novak and Alberto J. Cañas. 2007. Theoretical Origins of Concept Maps, How to Construct Them, and Uses in Education. *Reflecting Education*, 3(1):29–42.

Joseph D. Novak and D. Bob Gowin. 1984. *Learning How to Learn*. Cambridge University Press, Cambridge.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, Disambiguate and Walk: A Unified Approach for Measuring Semantic Similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1341–1351, Sofia, Bulgaria.

Iqbal Qasim, Jin-Woo Jeong, Jee-Uk Heu, and Dong-Ho Lee. 2013. Concept map construction from text documents using affinity propagation. *Journal of Information Science*, 39(6):719–736.

Kanagasabai Rajaraman and Ah-Hwee Tan. 2002. Knowledge discovery from texts: A Concept Frame Graph Approach. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 669–671, McLean, VA, USA.

Ryan Richardson and Edward A. Fox. 2005. Using concept maps as a cross-language resource discovery tool for large documents in digital libraries. In *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, page 415, Denver, CO, USA.

Dmitri G. Roussinov and Hsinchun Chen. 2001. Information navigation on the web by clustering and summarizing query results. *Information Processing & Management*, 37(6):789–816.

Debopriyo Roy. 2008. Using Concept Maps for Information Conceptualization and Schematization in Technical Reading and Writing Courses: A Case Study for Computer Science Majors in Japan. In *IEEE International Professional Communication Conference (IPCC 2008)*, pages 1–12, Montreal, Canada.

Marta Sabou, Kalina Bontcheva, Leon Derczynski, and Arno Scharl. 2014. Corpus Annotation through Crowdsourcing: Towards Best Practice Guidelines. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, pages 859–866, Reykjavik, Iceland.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and Fast – But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a Large Benchmark for Open Information Extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2300–2305, Austin, TX, USA.

Alejandro Valerio and David B. Leake. 2006. Jump-Starting Concept Map Construction with Knowledge Extracted from Documents. In *Proceedings of the 2nd International Conference on Concept Mapping*, pages 296–303, San José, Costa Rica.

Jorge J. Villalon. 2012. *Automated Generation of Concept Maps to Support Writing*. PhD Thesis, University of Sydney, Australia.

Jorge J. Villalon, Rafael A. Calvo, and Rodrigo Montenegro. 2010. Analysis of a Gold Standard for Concept Map Mining - How Humans Summarize Text Using Concept Maps. In *Proceedings of the 4th International Conference on Concept Mapping*, pages 14–22, Vina del Mar, Chile.

Xiaohang Zhang, Guoliang Li, and Jianhua Feng. 2016. Crowdsourced Top-k Algorithms: An Experimental Evaluation. *Proceedings of the Very Large Databases Endowment*, 9(8):612–623.

Markus Zopf, Maxime Peyrard, and Judith Eckle-Kohler. 2016. The Next Step for Multi-Document Summarization: A Heterogeneous Multi-Genre Corpus Built with a Novel Construction Approach. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 1535–1545, Osaka, Japan.

Amal Zouaq and Roger Nkambou. 2009. Evaluating the Generation of Domain Ontologies in the Knowledge Puzzle Project. *IEEE Transactions on Knowledge and Data Engineering*, 21(11):1559–1572.

Krunoslav Zubrinic, Ines Obradovic, and Tomo Sjekavica. 2015. Implementation of method for generating concept map from unstructured text in the Croatian language. In *23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 220–223, Split, Croatia.

Natural Language Does Not Emerge ‘Naturally’ in Multi-Agent Dialog

Satwik Kottur[1] and **José M.F. Moura**[1] and **Stefan Lee**[2,3] and **Dhruv Batra**[3,4]
[1]Carnegie Mellon University, [2]Virginia Tech, [3]Georgia Tech, [4]Facebook AI Research

Abstract

A number of recent works have proposed techniques for end-to-end learning of communication protocols among cooperative multi-agent populations, and have simultaneously found the *emergence of grounded human-interpretable language* in the protocols developed by the agents, learned without any human supervision!

In this paper, using a *Task & Talk* reference game between two agents as a testbed, we present a sequence of ‘negative’ results culminating in a ‘positive’ one – showing that while most agent-invented languages are effective (*i.e.* achieve near-perfect task rewards), they are decidedly *not* interpretable or compositional. In essence, we find that natural language does not emerge ‘naturally’, despite the semblance of ease of natural-language-emergence that one may gather from recent literature. We discuss how it is possible to coax the invented languages to become more and more human-like and compositional by increasing restrictions on how two agents may communicate.

1 Introduction

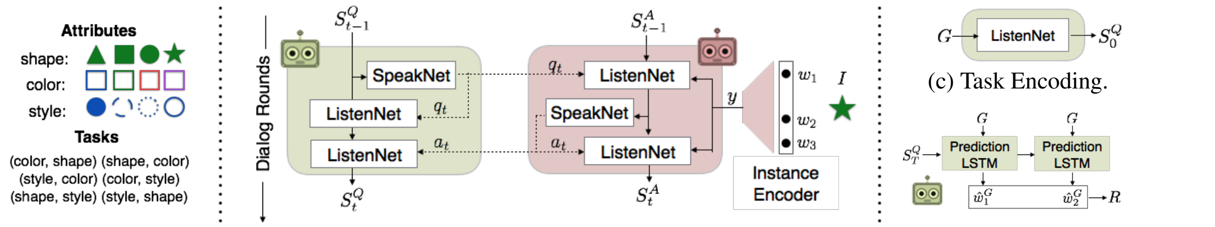
One fundamental goal of artificial intelligence (AI) is the development of goal-driven dialog agents – specifically, agents that can perceive their environment (through vision, audition, or other sensors), and communicate with humans or other agents in natural language towards a goal.

While historically such agents have been based on *slot filling* (Lemon et al., 2006), the dominant paradigm today is neural dialog models (Bordes and Weston, 2016; Weston, 2016; Serban et al., 2016a,b) trained on large quantities of data. Perhaps somewhat counterintuitively, this current paradigm treats dialog as a static supervised learning problem, rather than as the interactive agent learning problem that it naturally is. Specifically, a typical pipeline is to collect a large dataset of human-human dialog (Lowe et al., 2015; Das et al., 2017a; de Vries et al., 2017; Mostafazadeh et al., 2017), inject a machine in the middle of a dialog from the dataset, and supervise it to mimic the human response. While this teaches the agent correlations between symbols, it does not convey the functional meaning of language, grounding (mapping physical concepts to words), compositionality (combining knowledge of simpler concepts to describe richer concepts), or aspects of planning (why are we having this conversation?).

An alternative paradigm that has a long history (Winograd, 1971; Kirby et al., 2014) and is witnessing a recent resurgence (Wang et al., 2016; Foerster et al., 2016; Sukhbaatar et al., 2016; Jorge et al., 2016; Lazaridou et al., 2017; Havrylov and Titov, 2017; Mordatch and Abbeel, 2017; Das et al., 2017b) – is situated language learning. A number of recent works have proposed reinforcement learning techniques for learning the communication protocols of agents situated in virtual environments in a completely end-to-end manner – from perceptual input (*e.g.* pixels) to communication (discrete symbols without any prespecified meanings) to action (*e.g.* signaling in reference games or navigating in an environment) – and have simultaneously found the *emergence of grounded human-interpretable (often compositional) language* among agents, without any human supervision or pretraining, simply to succeed at the task.

In this short paper, we study the following question – what are the conditions that lead to the emergence of human-interpretable or compositional grounded language? Our key finding is that

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2962–2967
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics



(a) World. (b) Q-BOT (left) and A-BOT (right) policy networks. (c) Task Encoding. (d) Prediction.

Figure 1: *Task & Talk*: The testbed for our study is cooperative 2-player game, *Task & Talk*, grounded in a synthetic world of objects with 4 *shapes* × 4 *colors* × 4 *styles*. The two agents, Q-BOT and A-BOT, are modeled as neural networks, and their policies are learned via REINFORCE. We find that the languages invented by the two agents are typically *not* 'natural'.

natural language *does not emerge 'naturally'* in multi-agent dialog, despite independently reported successful demonstrations in recent literature.

Specifically, in a sequence of 'negative' results culminating in a 'positive' one, we find that while agents always successfully invent communication protocols and languages to achieve their goals with near-perfect accuracies, the invented languages are decidedly *not* compositional, interpretable, or 'natural'; and that it is possible to coax the invented languages to become more and more human-like and compositional by increasing restrictions on how two agents may communicate.

Related work and novelty. The starting point for our investigation is the recent work of Das et al. (2017b), who proposed a cooperative reference game between two agents, where communication is necessary to accomplish the goal due to an information asymmetry. Our key contribution over Das et al. (2017b) is an exhaustive study of the conditions that must be present before compositional grounded language emerges, and subtle but important differences in execution – tabular Q-Learning (which does not scale) *vs.* REINFORCE (which does), and generalization to novel environments (not studied in prior work). In the spirit of Abbeel et al. (2017), we hope our findings shed more light into the interpretability of languages invented in cooperative multi-agent settings, place recent work in appropriate context, and inform fruitful directions for future work.

2 The *Task & Talk* Game

Our testbed is a reference game (*Task & Talk*) between two agents, Q-BOT and A-BOT. The game is grounded in a synthetic world of objects comprised of three attributes – *color*, *style*, and *shape* – each with four possible values for a total of $4 \times 4 \times 4 = 64$ objects. Fig. 1a shows some example instances from this set.

Task & Talk plays out over multiple rounds of dialog. At the start, A-BOT is given an instance (*e.g. (green, dotted, square)*) unseen by Q-BOT, and Q-BOT is assigned a task G (unknown to A-BOT) consisting of two attributes for Q-BOT to discover from A-BOT (*e.g. (color, style)*). For two rounds, Q-BOT and A-BOT exchange utterances from finite vocabularies V_Q and V_A, with Q-BOT speaking first. The game culminates in Q-BOT guessing a pair of attribute values (*e.g. (green, dotted)*) and both agents are rewarded identically based on the accuracy of this prediction.

Note that the Task & Talk game setting involves an informational asymmetry between the agents – A-BOT sees the object while Q-BOT does not; similarly Q-BOT knows the task while A-BOT does not. Thus, a two-way communication is necessary for success. Without this asymmetry, A-BOT could simply convey the target attributes from the task without Q-BOT having to speak. Such a setting has been widely studies in economics and game theory as the classic Lewis Signaling (LS) game (Lewis, 2008). By necessitating *dialog* between agents, we are able ground both V_A and V_Q in our final setting (Sec. 4.3).

3 Modeling Q-BOT and A-BOT

We formalize Q-BOT and A-BOT as agents operating in a partially observable world and optimize their policies using deep reinforcement learning.

States and Actions. Each agent observes its own input (task G for Q-BOT and object instance I for A-BOT) and the output of the other agent as a stochastic environment. At the beginning of round t, Q-BOT observes state $s^t_Q{=}[G, q_1, a_1, \ldots, q_{t-1}, a_{t-1}]$ and acts by uttering some token q_t from its vocabulary V_Q. Similarly, A-BOT observes the history and this new utterance as state $s^t_A{=}[I, q_1, a_1, \ldots, q_{t-1}, a_{t-1}, q_t]$ and emits a response a_t from V_A. At the last round, Q-BOT takes a final action by predicting a pair of attribute values $\hat{w}^G = (\hat{w}^G_1, \hat{w}^G_2)$ to solve the task.

Cooperative Reward. Both Q-BOT and A-BOT are rewarded identically based on the accuracy of

Q-BOT's prediction $\hat{w}^G$, receiving a positive reward of $R{=}1$ if the prediction matches ground truth and a negative reward of $R{=}{-}10$ otherwise. We arrive at these values empirically.

Policy Networks. We model Q-BOT and A-BOT as operating under stochastic policies $\pi_Q(q_t|s_t^Q;\theta_Q)$ and $\pi_A(a_t|s_t^A;\theta_A)$ respectively, which we instantiate as LSTM-based models. We use lower case characters (*e.g.* s_t^Q) to denote the strings (*e.g.* Q-BOT's token at round t), and upper case S_t^Q to denote the corresponding vector as encoded by the model.

As shown in Fig. 1, Q-BOT is modeled with three modules – speaking, listening, and prediction. The task G is received as a 6-dimensional one-hot encoding over the space of possible tasks and embedded via the listener LSTM. At each round t, the *speaker network* models the probability of output utterances $q_t \in V_Q$ based on the state S_{t-1}^Q. This is modeled as a fully-connected layer followed by a softmax that transforms S_{t-1}^Q to a distribution over V_Q. After receiving the reply a_t from A-BOT, the *listener LSTM* updates the state by processing both tokens of the dialog exchange. In the final round, the *prediction LSTM* is unrolled twice to produce Q-BOT's prediction based on the final state S_T^Q and the task G. As before, task G is fed in one-hot to the prediction LSTM for two time steps, resulting in a pair of outputs used as the prediction $\hat{w}_G$.

Analogously, A-BOT is modeled as a combination of a *speaker network*, a *listener LSTM*, and an *instance encoder*. Like in Q-BOT, the speaker network models the probability of utterances $a_t \in V_A$ given the state S_t^A and the listener LSTM updates the state S_t^A based on dialog exchanges. The instance encoder embeds each one-hot attribute vector via a linear layer and concatenates all three encodings to obtain a unified instance representation.

Learning Policies with REINFORCE. We train these models using the popular REINFORCE (Williams, 1992) policy gradient algorithm. Note that while the game is fully-cooperative, we do not assume full observability of one agent by another, opting instead to treat one agent as part of the unknown stochastic environment when updating the other. During training, we sample 1000 two round dialog episodes per batch and update policy parameters with Adam (Kingma and Ba, 2015) based on these approximate gradients. Our code is publicly available[1].

4 The Road to Compositionality

This section details our key observation – that while the agents always successfully invent a language to solve the game with near-perfect accuracies, the invented languages are decidedly *not* compositional, interpretable, or 'natural' (*e.g.* A-BOT ignoring Q-BOT's utterances and simply encoding every object with a unique symbol if the vocabulary is sufficiently large). In our setting, the language being compositional simply amounts to the ability of the agents to communicate the compositional atoms of a task (*e.g. shape* or *color*) and an instance (*e.g. square* or *blue*) independently.

Through this section, we present a series of settings that get progressively more restrictive to coax the agents towards adopting a compositional language, providing analysis of the learned languages developed along the way. Table 1 summarizes results for all settings. In all experiments, optimal policies (achieving near-perfect rewards) were found. For each setting, we provide qualitative analysis of the learned languages and report their ability to generalize to unseen instances. We use 80% of the object-instances for training and the remaining 20% to evaluate these learned policies. Further, greedy argmax policies are used at evaluation time.

4.1 Overcomplete Vocabularies

We begin with the simplest setting where both A-BOT and Q-BOT are given arbitrarily large vocabularies. We find that when $|V_A|$ is greater than the number of instances (64), the learned policy simply has A-BOT ignore what Q-BOT asks and instead convey the instance using a single symbol, *e.g.* token 1≡*(red, square, filled)*. Notice that this means no 'dialog' is necessary and amounts to each agent having a codebook that maps symbols to object instances.

Perhaps as expected, the generalization of this language to unseen instances is quite poor (success rate: 25.6%). The adopted strategy of mapping instances to token pairs fails for test instances containing novel combinations of attributes for which the agents lack an agreed-upon code from training.

It seems clear that like in human communication (Nowak et al., 2000), a limited vocabulary that cannot possibly encode the richness of the world seems to be necessary for non-trivial dialog to emerge. We explore such a setting next.

[1] github.com/batra-mlp-lab/lang-emerge

Setting	Vocab.		Memory		Gen.(%)	
	V_Q	V_A	A	Q	Both	One
Overcomplete (§4.1)	64	64	✓	✓	25.6	79.5
Attr-Value (§4.2)	3	12	✓	✓	38.5	88.4
NoMem-Min (§4.3)	3	4	✗	✓	**74.4**	**94.9**

Table 1: Overview of settings we explore to analyze the language learnt by two agents in a cooperative game, Task & Talk. Last two columns measure generalization in terms of prediction accuracy of **both** or at least **one** of the attribute pair, on a held-out test set containing unseen instances.

4.2 Attribute & Value Vocabulary

Since our world has 3 attributes (*shape/color/style*) and $4+4+4=12$ possible settings of their states, one may believe that the intuitive choice of $|V_Q|=3$ and $|V_A|=12$ will be enough to circumvent the 'cheating' enumeration strategy from the previous experiment. Surprisingly, we find that the new language learned in this setting is not only decidedly non-compositional but also very difficult to interpret! We present two salient observations.

We observe that Q-BOT uses only the first round to convey the task to A-BOT by encoding tasks in an order-independent fashion *e.g.* the *(style,color)* and *(color,style)* tasks are both expressed as the utterance *Z* in the first round. Consequentially, multiple rounds of dialog are rended unnecssary and the second round is inconsistent across instances even for the same task.

Given the task from Q-BOT in the first round, A-BOT only needs to identify one of the 4×4=16 attribute pairs for a given task. Rather than ground its symbols into individual states, A-BOT follows a 'set partitioning' strategy, *i.e.* A-BOT identifies a pair of attributes with a unique combinations of round 1 and 2 utterances (*i.e.* the round 2 utterance has no meaning independent from round 1). Thus, symbols are reused across tasks to describe different attributes (*i.e.* symbols do not have individual consistent groundings). This 'set partitioning' strategy is consistent with known results from game theory on Nash equilibria in 'cheap talk' games (Crawford and Sobel, 1982).

This strategy has improved generalization to unseen instances because it is able to communicate the task; however, it fails on unseen attribute value combinations because it is not compositional.

4.3 Memoryless A-BOT, Minimal Vocabulary

The key problem with the previous setting is that A-BOT's utterances *mean different things* based on the round of dialog ($a_1=1$ is different from $a_2=1$). Essentially, the communication protocol is overparameterized and we must limit it further. First, we limit A-BOT's vocabulary to $|V_A|$=4 to reduce the number of 'synonyms' the agents learn. Second, we *remove* A-BOT*'s memory* by reseting the state vector S^A at each time step, which eliminates its ability to enumerate all attribute pairs.

These restrictions result in a learned language that grounds *individual symbols* into attributes and their states. For example, Q-BOT learns that $Y\rightarrow$ shape, $X\rightarrow$ color, and $Z\rightarrow$ style. Q-BOT does not however learn to always utter these symbols in the same order as the task, *e.g.* asking for *shape* first for both $(color, shape)$ and $(shape, color)$. Notice that this is perfectly valid as Q-BOT can later re-arrange the attributes in the task desired order. Similarly, A-BOT learns mappings to attribute values for each attribute query that remain consistent regardless of round (*i.e.* when asked for $color$, 1 always means $blue$).

This is similar to learned languages reported in recent works and is most closely related to Das et al. (2017b), who solve this problem by taking away Q-BOT's state rather than A-BOT's memory. Their approach can be interpreted as Q-BOT 'forgetting' the task after interacting with A-BOT. However, this behavior of Q-BOT to remember the task only during dialog but not while predicting is somewhat unnatural compared to our setting.

Tab. 2 enumerates the learnt groundings for both the agents. Given this mapping, we can predict a plausible dialog between the agents for any unseen instance and task combination. Notice that this is possible only due to the compositionality in the emergent language between the two agents. For example, consider solving *(shape, color)* for an instance *(red, square, filled)* from Fig. 2(b). Q-BOT queries *Y (shape)* and *X (color)* across two rounds, and receives *2 (square)* and *4 (red)* as answers.

Intuitively, this consistently grounded and compositional language has the greatest ability to generalize among the settings we have explored, correctly answering 74.4% of the held out instances. We note that errors in this setting seem to largely be due to A-BOT giving an incorrect answers despite Q-BOT asking the correct questions to accomplish the task. A plausible reason could be the model approximation error stemming from the instance encoder as test instances are unseen and have novel attribute combinations.

Fig. 2(b) shows the dialog for the instance *(red, square, filled)* and task *(shape, color)*. Q-BOT queries *Y (shape)* and *(color)* across two rounds,

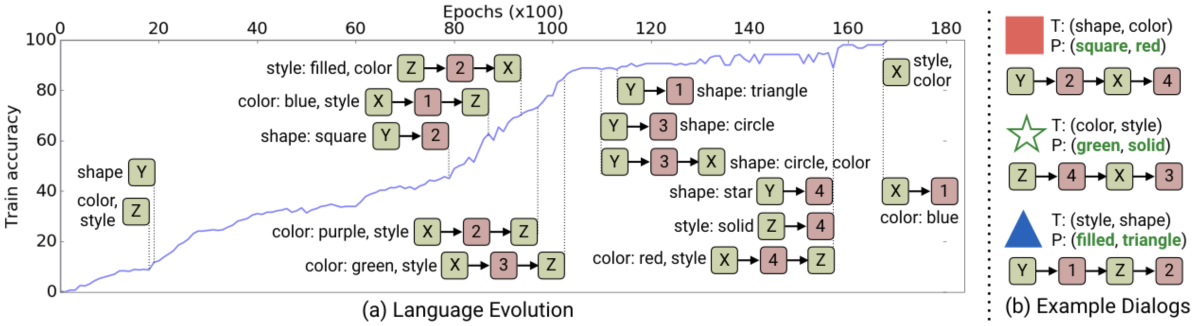


Figure 2: (a) Evolution of Language: timeline shows groundings learned by the agents during training, overlaid on the accuracy. Note that Q-BOT learns encodings for all tasks early (around epoch 20) except *(style, color)*. Improvement in accuracy is strongly correlated with groundings learnt. (b) Example dialogs for memoryless A-BOT, minimal vocabulary setting (§4.3.

	Attributes		
	color	*shape*	*style*
V_A	*X*	*Y*	*Z*
1	*blue*	*triangle*	*dotted*
2	*purple*	*square*	*filled*
3	*green*	*circle*	*dashed*
4	*red*	*start*	*solid*

(a) A-BOT

Task	q_1, q_2
(color, shape)	*Y, X*
(shape, color)	
(shape, style)	*Y, Z*
(style, shape)	
(color, style)	*Z, X*
(style, color)	*X, Z*

(b) Q-BOT

Table 2: Emergence of compositional grounding for language learnt by the agents. **A-BOT** (Tab. 2a) learns consistent mapping across rounds, depending on the query attribute. Token grounding for **Q-BOT** (Tab. 2b) depends on the task at hand. Though compositional, Q-BOT does not necessarily query attribute in the order of task, but instead re-arranges accordingly at prediction time as it contains memory.

and receives *2 (square)* and *4 (red)* as answers.

4.4 Evolution of Language Timeline

To gain further insight into the languages learned, we create a *language evolution* plot in Fig. 2. Specifically, at regular intervals during policy learning, we construct 'dialog trees'. A dialog tree enumerates all plausible dialogs between the two agents (q_1, a_1, q_2, a_2), as a tree. The root node is q_1, and at any node, we go deeper by choosing a branch based on the next utterance in the dialog. Since Task & Talk runs for two rounds, our dialog trees are 4 layers deep with $|V_A|^2|V_Q|^2$ leaves. Notice that a single input instance could potentially result in different dialogs depending on the task. Hence, we consider *(instance, task)* pairs and assign each to a leaf by traversing the tree according to the resulting dialog. At some point in the learning, the nodes become and stay 'pure' (the common trend among all *(instance, task)* at the node stays constant till the end of training), at which point we can say that the agents have learned this dialog subsequence.

Construction. After constructing dialog trees at regular intervals, we identify 'concepts' at each node/leaf using the dialog tree of the completely trained model, which achieves a perfect accuracy on train set. A concept is simply the common trend among all the *(instance, task)* tuples either assigned to a leaf or contained within the subtree with a node as root. Next, given a resultant concept for each of the node/leaf, we backtrack in time and check for the first occurrence when only tuples which satisfy the corresponding concept are assigned to that particular node/leaf. In other words, we compute the earliest time when a node/leaf is 'pure' with respect to its final learned concept. Finally, we plot these leaves/nodes and the associated concept with their backtracked time to get Fig. 2.

Observations. We highlight key observations below: (a) The agents ground most of the tasks initially at around epoch 20. Specifically, Q-BOT assigns *Y* to both *(shape, style)*, *(style, shape)*, *(shape,color)* and *(color, shape)*, while *(color, style)* is mapped to *Z*. Hence, Q-BOT learns its first token very early into the training procedure. (b) The only other task *(style, color)* is grounded towards the end (around epoch 170) using *X*, leading to an immediate convergence. (c) We see a strong correlation between improvement in performance and when agents learn a language grounding. In particular, there is an improvement from 40% to 80% within a span of 25 epochs where most of the grounding is achieved, as seen from Fig. 2.

5 Conclusion

In conclusion, we presented a sequence of 'negative' results culminating in a 'positive' one – showing that while most invented languages are effective (*i.e.* achieve near-perfect rewards), they are decidedly *not* interpretable or compositional. Our goal is simply to improve understanding and interpretability of invented languages in multi-agent dialog, place recent work in context, and inform fruitful directions for future work.

References

Pieter Abbeel, Igor Mordatch, Ryan Lowe, Jon Gauthier, and Jack Clark. 2017. Learning to Communicate. https://blog.openai.com/learning-to-communicate/.

Antoine Bordes and Jason Weston. 2016. Learning End-to-End Goal-Oriented Dialog. *arXiv preprint arXiv:1605.07683* .

Vincent Crawford and Joel Sobel. 1982. Strategic information transmission. *Econometrica* 50(6):1431–51.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, and Dhruv Batra. 2017a. Visual Dialog. In *CVPR*.

Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra. 2017b. Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning. *arXiv preprint arXiv:1703.06585* .

Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville. 2017. Guesswhat?! visual object discovery through multi-modal dialogue. In *CVPR*.

Jakob Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *NIPS*.

Serhii Havrylov and Ivan Titov. 2017. Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *ICLR Workshop*.

Emilio Jorge, Mikael Kågebäck, and Emil Gustavsson. 2016. Learning to play guess who? and inventing a grounded language as a consequence. In *NIPS workshop on deep reinforcement learning*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Simon Kirby, Tom Griffiths, and Kenny Smith. 2014. Iterated learning and the evolution of language. *Current opinion in neurobiology* 28:108–114.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. 2017. Multi-agent cooperation and the emergence of (natural) language. In *ICLR*.

Oliver Lemon, Kallirroi Georgila, James Henderson, and Matthew Stuttle. 2006. An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system. In *EACL*.

David Lewis. 2008. *Convention: A philosophical study*. John Wiley & Sons.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *SIGDIAL*.

Igor Mordatch and Pieter Abbeel. 2017. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908* .

Nasrin Mostafazadeh, Chris Brockett, Bill Dolan, Michel Galley, Jianfeng Gao, Georgios P. Spithourakis, and Lucy Vanderwende. 2017. Image-Grounded Conversations: Multimodal Context for Natural Question and Response Generation. *arXiv preprint arXiv:1701.08251* .

Martin A. Nowak, Joshua B. Plotkin, and Vincent A. A. Jansen. 2000. The evolution of syntactic communication. *Nature* 404(6777):495–498.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016a. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. In *AAAI*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016b. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues. *arXiv preprint arXiv:1605.06069* .

Sainbayar Sukhbaatar, Rob Fergus, et al. 2016. Learning multiagent communication with backpropagation. In *NIPS*. pages 2244–2252.

Sida I Wang, Percy Liang, and Christopher D Manning. 2016. Learning language games through interaction. *Association for Computational Linguistics (ACL)* .

Jason Weston. 2016. Dialog-based language learning. *arXiv preprint arXiv:1604.06045* .

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Terry Winograd. 1971. Procedures as a representation for data in a computer program for understanding natural language. Technical report, DTIC Document.

Depression and Self-Harm Risk Assessment in Online Forums

Andrew Yates[†*] **Arman Cohan**[‡*] **Nazli Goharian**[‡]

[†]Max Planck Institute for Informatics,
Saarland Informatics Campus Saarbruecken, Germany

[‡]Information Retrieval Lab, Department of Computer Science,
Georgetown University, Washington DC, USA

ayates@mpi-inf.mpg.de
{arman,nazli}@ir.cs.georgetown.edu

Abstract

Users suffering from mental health conditions often turn to online resources for support, including specialized online support communities or general communities such as Twitter and Reddit. In this work, we present a framework for supporting and studying users in both types of communities. We propose methods for identifying posts in support communities that may indicate a risk of self-harm, and demonstrate that our approach outperforms strong previously proposed methods for identifying such posts. Self-harm is closely related to depression, which makes identifying depressed users on general forums a crucial related task. We introduce a large-scale general forum dataset consisting of users with self-reported depression diagnoses matched with control users. We show how our method can be applied to effectively identify depressed users from their use of language alone. We demonstrate that our method outperforms strong baselines on this general forum dataset.

1 Introduction

Mental health remains a major challenge in public health care. Depression is one of the most common mental disorders and 350 million people are estimated to suffer from depression worldwide (WHO, 2010). In 2014 an estimated 7% of all U.S. adults had experienced at least one major depressive disorder (2015). Suicide and self-harm are major related concerns in public mental health. Suicide is one of the leading causes of death (CDC, 2015), and each suicide case has major consequences on the physical and emotional well-being of families and on societies in general. Therefore identifying individuals at risk of self-harm and providing support to prevent it remains an important problem (Ferrari et al., 2014).

Social media is often used by people with mental health problems to express their mental issues and seek support. This makes social media a significant resource for studying language related to depression, suicide, and self-harm, as well as understanding the authors' reasons for making such posts, and identifying individuals at risk of harm (Coppersmith et al., 2014a). Depression and suicide are closely related given that depression is the psychiatric diagnosis most commonly associated with suicide. Research has demonstrated that forums are powerful platforms for self-disclosure and social support seeking around mental health concerns (De Choudhury and De, 2014; Manikonda and De Choudhury, 2017). Such support forums are often staffed by moderators who are mental health experts, trained volunteers, or more experienced users whose role is to identify forum posts suggesting that a user is at risk of self-harm and to provide support.

Studies have shown that self expression and social support are beneficial in improving the individual's state of the mind (Turner et al., 1983; Choudhury and Kiciman, 2017) and thus such communities and interventions are important in suicide prevention. However, there are often thousands of user posts published in such support forums daily, making it difficult to manually identify individuals at risk of self-harm. Additionally, users in acute distress need prompt attention, and any delay in responding to these users could have adverse consequences. Therefore, identifying individuals at risk of self-harm in such support forums is an important challenge. Identifying signs of depression in general social media, on the other hand, is also a difficult task that has appli-

[*] The first two authors contributed equally to this work.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2968–2978
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

cations for both better understanding the relationship between mental health and language use and for monitoring a specific user's state (e.g., in the context of monitoring a user's response to clinical care). In this work we propose and evaluate a framework for performing self-harm risk assessment and for identifying depression in online forums.

We present a general neural network architecture for combining posts into a representation of a user's activity that is used to classify the user. To address the challenge of depression risk assessment over the general forums, we introduce a large-scale novel Reddit dataset that is substantially larger than the existing data and has a much more realistic number of control users. The dataset contains over 9,000 users with self-reported depression diagnoses matched with over 107,000 control users. We apply our approach to *(1)* identify the users with depression on a general forum like Reddit, and to *(2)* estimate the risk of self-harm indicated by posts in a more specific mental-health support forum. Our methods perform significantly better on both datasets than strong existing methods, demonstrating that our approach can be used both to identify depressed users and to estimate the risk of self-harm posed by individual posts.

2 Related Work

There is a growing body of related work analyzing mental health-related discourse and language usage in social media to better discover and understand mental health related concerns (Resnik et al., 2013; De Choudhury et al., 2013; Coppersmith et al., 2014b,a; Mitchell et al., 2015; Tsugawa et al., 2015; Coppersmith et al., 2015a; Althoff et al., 2016; Mowery et al., 2016; Benton et al., 2017b). To investigate NLP methods for identifying depression and PTSD users on Twitter, a shared task (Coppersmith et al., 2015b) at the 2nd Computational Linguistics and Clinical Psychology Workshop (CLPsych 2015) was introduced where the participants evaluated their methods on a dataset of about 1800 Twitter users. Other work has used data from approximately 900 Reddit.com users to support self-reported diagnosis detection (Losada and Crestani, 2016). Previous work identifying depression and other mental health problems, including the methods participating in CLPsych 2015 (e.g. (Resnik et al., 2015; Preoţiuc-Pietro et al., 2015)) heavily rely on utilizing features such as LIWC (Pennebaker et al., 2015), topic modeling, manual lexicons, or other domain-dependent application-specific features. Aside from the effort required to design effective features, these approaches usually model the problem with respect to the selected features and ignore other indicators and signals that can improve prediction. In contrast, our model only relies on text and is not dependent on any external or domain-specific features. Previous self-reported diagnosis detection datasets contained a limited number of both control users and diagnosed users. In contrast to this, we construct a new dataset with over 9,000 depressed users matched with a realistic number of control users.

In addition to general studies addressing mental health, related work has also specifically studied suicide and self-harm through social media (Jashinsky et al., 2014; Thompson et al., 2014; Gunn and Lester, 2015; De Choudhury et al., 2016; Coppersmith et al., 2016). Recently, CLPsych 2016 (Hollingshead and Ungar, 2016) investigated approaches for detecting the self-harm risk of mental health forum posts (Milne et al., 2016). Most related work in this area uses variations of linear classifiers with some sort of feature engineering; successful methods have employed: a combination of sparse (bag-of-words) and dense (doc2vec) representation of the target forum posts (Kim et al., 2016), a stack of feature-rich Random Forest and linear Support Vector Machine (SVM) (Malmasi et al., 2016), an RBF SVM classifier utilizing similar sets of features (Brew, 2016), and various contextual and psycholinguistic features (Cohan et al., 2016, 2017). In contrast to the above works, our model does not use any general or domain specific feature engineering; it learns appropriate representations of documents by considering only their textual content.

Our proposed models consist of a shared architecture based on a CNN, a merge layer, model-specific loss functions, and an output layer (as we will describe in §4). While our model shares similarities with CNN-based models in prior work (Kalchbrenner et al., 2014; Kim, 2014; Xiao and Cho, 2016), it focuses on learning representations of user's posts and combining the post representations into an overall representation of the user's activity. In the case of self-harm risk assessment, we experiment with several loss functions to determine whether considering the ordinal nature of self-harm risk labels (i.e., green, amber, red, and crisis) can improve performance. Evaluation results suggest that the model variant using this loss function is more robust than our other variants.

3 Data

3.1 Depression dataset construction.

We created a new dataset to support the task of identifying forum users with self-reported depression diagnoses. The Reddit Self-reported Depression Diagnosis (RSDD) dataset was created by annotating users from a publicly-available Reddit dataset[1]. Users to annotate were selected by identifying all users who made a post between January 2006 and October 2016 matching a high-precision diagnosis pattern.[2] Users with fewer than 100 posts made before their diagnosis post were discarded. Each of the remaining diagnosis posts was then viewed by three layperson annotators to decide whether the user was claiming to have been diagnosed with depression; the most common false positives included hypotheticals (e.g., "if I was diagnosed with depression"), negations (e.g., "it's not like I've been diagnosed with depression"), and quotes (e.g., "my brother announced 'I was just diagnosed with depression' "). Only users with at least two positive annotations were included in the final group of diagnosed users.

A pool of potential control users was identified by selecting only those users who had *(1)* never posted in a subreddit related to mental health, and *(2)* never used a term related to depression or mental health. These restrictions minimize the likelihood that users with depression are included in the control group. In order to prevent the diagnosed users from being easily identified by the usage of specific keywords that are never used by the control users, we removed all posts by diagnosed users that met either one of the aforementioned conditions (i.e., that was posted in a mental health subreddit or included a depression term).

For each diagnosed user and potential control user, we calculated the probability that the user would post in each subreddit (while ignoring diagnosed users' posts made to mental health subreddits). Each diagnosed user was then greedily matched with the 12 control users who had the smallest Hellinger distance between the diagnosed user's and the control user's subreddit post probability distributions, excluding control users with 10% more or fewer posts than the diagnosed user. This matching approach ensures that diagnosed users are matched with control users who are interested in similar subreddits and have similar activity levels, preventing biases based on the subreddits users are involved in or based on how active the users are on Reddit. This yielded a dataset containing 9,210 diagnosed users and 107,274 control users. On average each user in the dataset has 969 posts (median 646). The mean post length is 148 tokens (median 74).

The Reddit Self-reported Depression Diagnosis (RSDD) dataset differs from prior work creating self-reported diagnoses datasets in several ways: it is an order of magnitude larger, posts were annotated to confirm that they contained claims of a diagnosis, and a realistic number of control users were matched with each diagnosed user. The lists of terms related to mental health, subreddits related to mental health, high-precision depression diagnosis patterns, and further information are available[3]. We note that this dataset has some (inevitable) caveats: *(i)* the method only captures a subpopulation of depressed people (i.e. those with self-reported diagnosis), *(ii)* Reddit users may not be a representative sample of the population as a whole, and *(iii)* there is no way to verify whether the users with self-reported diagnoses are truthful.

3.2 Self-harm assessment.

For self-harm risk assessment we use data from mental health forum posts from ReachOut.com, which is a successful Australian support forum for young people. In addition to providing peer-support, ReachOut moderators and trained volunteers monitor and participate in the forum discussions. The NAACL 2016 Computational Linguistics and Clinical Psychology Workshop (Hollingshead and Ungar, 2016) released a Triage dataset containing 65,024 forum posts from ReachOut, with annotations for 1,227 posts indicating the author's risk of self-harm (Milne et al., 2016). The annotations consist of one of four labels: green (indicating no action is required from ReachOut's moderators), amber (non-urgent attention is required), red (urgent attention is required), and crisis (a risk that requires immediate attention).

[1] https://files.pushshift.io/reddit/

[2] e.g., "I was just diagnosed with depression."

[3] http://ir.cs.georgetown.edu/data/reddit_depression/

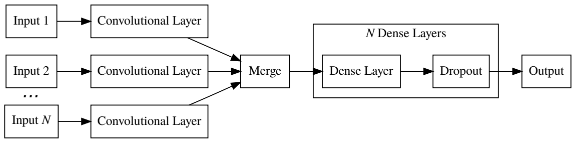


Figure 1: The general neural network architecture shared among our user and post classification models. Each input (e.g., each of a user's posts) is processed by a convolutional network and merged to create a vector representation of the user's activity. This vector representation is passed through one or more dense layers followed by an output layer that performs classification. The type of input received, merge operation, and output layer vary with the specific model.

3.3 Ethical concerns.

Social media data are often sensitive, and even more so when the data are related to mental health. Privacy concerns and the risk to the individuals in the data should always be considered (Hovy and Spruit, 2016; Šuster et al., 2017; Benton et al., 2017a). We note that the risks associated with the data used in this work are minimal. This assessment is supported by previous work on the ReachOut dataset (Milne et al., 2016), on Twitter data (Coppersmith et al., 2015b), and on other Reddit data (Losada and Crestani, 2016). The RSDD dataset contains only publicly available Reddit posts. Annotators were shown only anonymized posts and agreed to make no attempts to deanonymize or contact them. The RSDD dataset will only be made available to researchers who agree to follow ethical guidelines, which include requirements not to contact or attempt to deanonymize any of the users. Additionally, for the ReachOut forum data that was explicitly related to mental health, the forum's rules require the users to stay anonymous; moderators actively redact any user identifying information.

4 Methodology

We describe a general neural network architecture for performing text classification over multiple input texts. We propose models based on this architecture for performing two tasks in the social media and mental health domains that we call *self-harm risk classification* and *detecting depression*. The task of self-harm risk classification is estimating a user's current self-harm risk given the user's post on a mental health support forum and the previous posts in the thread. The task of detecting depressions in users is identifying Reddit users with self-reported depression diagnoses given the users' post histories (excluding posts containing mental health keywords or posted in subreddits related to mental health).

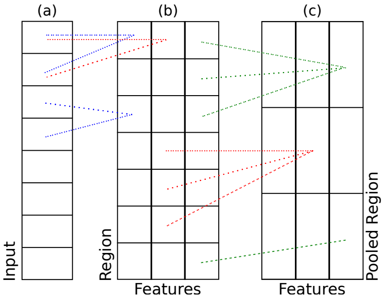


Figure 2: The convolutional network component of our architecture. A convolutional layer takes a series of terms as input *(a)* and applies l filters to a k-term sliding window to derive feature values for each window or region *(b)*; $k = 2$ and $l = 3$ shown here. A max pooling layer considers non-overlapping region sequences of length n *(b)* and keeps the highest feature value for the sequence *(c)*; $n = 3$ shown here.

While both tasks are focused on predicting a user's mental health status, they differ in both the type of classification performed (i.e., estimating severity on a four point scale vs. boolean classification) and in the amount of data available. Our general architecture is based on a two step process: *(1)* identifying relevant features in each input text, and *(2)* combining the features observed in the model's inputs to classify the user.

4.1 Shared Architecture

Our proposed models share a common architecture that takes one or more posts as input, processes the posts using a convolutional layer to identify features present in sliding windows of text, merges the features identified into a vector representation of the user's activity, and uses a series of dense layers to perform classification on the merged vector representation. The type of merging performed and the output layers are properties of the model variant, which we describe in detail in the following section. Convolutional networks have commonly been applied to the task of text classification, such as by Kim (2014). We use categorical cross-entropy as a loss function with both methods, but also experiment with other loss functions when performing severity classification.

First, the model takes one or more posts as input and processes each post with a convolutional network containing a convolutional layer and a pooling layer. This process is illustrated with a max pooling layer in Figure 2. The convolutional layer applies filters to a sliding window of k terms *(a)* and outputs a feature value for each sliding window region and each filter *(b)*. The same filters are applied to each window; each filter can be viewed as a feature detector and the overall process can be conceptualized as looking for windows of terms that contain specific features. The features are not specified a priori through feature engineering, but instead are learned automatically when the model is trained. After identifying the features present in each region (i.e., sliding window), a max pooling layer considers non-overlapping regions of length n and keeps the highest feature value for each region *(c)*. This step eliminates the regions (i.e., sliding windows) that do not contain useful features, which reduces the size of the convolutional network's output. The same convolutional network is applied to each input post, meaning that the model learns to look for the same set of features in each.

After each input post has been processed by a convolutional network, the output of each convolutional network is merged to create a representation of the user's activity across all input posts. This representation is processed by one or more dense layers (i.e., fully connected layers) with dropout (Srivastava et al., 2014) before being processed by a final output layer to perform classification. The type of output layer is dependent on the model variant. Our shared model architecture is illustrated in Figure 1. The architecture's hyperparameters (e.g., the sliding window size k, the number of convolutional filters used, and type of pooling) also vary among models and are described in the supplemental material. Both the convolutional and dense layers use ReLU activations (Nair and Hinton, 2010) in all model variants.

4.2 Models

4.2.1 Depression detection

Our model for depression detection takes a user's posts as input and processes each post with a convolutional network. Each convolutional network performs average pooling to produce its output. That is, the model considers non-overlapping sequences of n posts and keeps the average feature value across all sequences. These post representations are then merged with a second convolutional layer to create a user representation; we found this approach led to more stable performance than using a second average pooling or max pooling layer. The user representation created by the merge step is then passed to one or more dense layers before being passed to a dense output layer with a softmax activation function to perform classification. The number of dense layers used is a hyperparameter described in §5. Categorical cross-entropy is used as the model's loss function.

4.2.2 Self-harm risk assessment

Our model for self-harm risk classification takes two inputs: the target post being classified and the prior posts (if any) in the target post's thread. The prior posts provide context and are thus useful for estimating the risk of self-harm present in the target post. The two inputs are both processed by a convolutional network as in user-level classification, but in this case the convolutional network's outputs correspond to a representation of the target post and to a representation of the target post's context (i.e., the prior posts in the thread). Given that these two outputs represent different aspects, they are merged by concatenating them together. This merged representation is then passed to one or more dense layers and to an output layer; the type of output layer depends on the loss function used. There are four self-harm risk assessment model variants in total:

Categorical Cross Ent. uses an output layer with a softmax activation function, and categorical cross-entropy as its loss function. This mirrors the output layer and loss function used in the user level classification model.

MSE uses an output layer with a linear activation function, and mean squared error as its loss function. The model's output is thus a single value; to perform classification, this output value is rounded to the nearest integer in the interval $[0, t-1]$, where t is the number of target classes.

The final two loss functions perform metric learning rather than performing classification directly. They learn representations of a user's activity and of the four self-harm risk severity labels; classification is performed by comparing the euclidean distance between a representation of a user's activity (produced by the final layer) and each of the four severity label representations.

Method		Convolution			Dense Layers	Dropout	Class Balance
		Size	Filters	Pool Len.			
Reddit	Cat. Cross Ent.	3	25	all (avg)	1 w/ 50 dims	0.0	Sampled
ReachOut	Cat. Cross Ent.	3	150	3 (max)	2 w/ 250 dims	0.3	Weighted
	MSE	3	100	3 (max)	2 w/ 250 dims	0.5	Sampled
	Class Metric	3	100	3 (max)	2 w/ 150 dims	0.3	Sampled

Table 1: The hyperparameters used by each model.

Class Metric: Let d be the size of the output layer and X be the layer's d-dimensional output. *Class Metric* learns a d-dimensional representation of each class C_i such that $||X - C_i||_2$ is minimized for the correct class i; this is accomplished with the loss function: $L_{i,p,n} = max(0, ||X_i - C_p||_2 - ||X_i - C_n||_2 + \alpha)$ where C_p is the correct (i.e., positive) class for X_i, C_n is a randomly chosen incorrect (i.e., negative) class, and α is a constant to enforce a minimum margin between classes. Classification is performed by computing the similarity between X_i and each class C_j.

Class Metric (Ordinal) extends *Class Metric* to enforce a margin between ordinal classes as a function of the distance between classes. Given a ranked list of classes such that more similar classes have closer rankings, that is $\forall i\ sim(C_i, C_{i\pm1}) > sim(C_i, C_{i\pm2})$, we incorporate the class distance into the margin such that more distant incorrect class labels must be further away from the correct class label in the metric space. The loss function becomes $L_{i,p,n} = max(0, ||X_i - C_p||_2 - ||X_i - C_n||_2 + \alpha|p - n|)$ where $|p - n|$ causes the margin to scale with the distance between classes p and n.

5 Experiments

In this section we describe the model hyperparameters used and present our results on the depression detection and self-harm risk assessment tasks. To facilitate reproducibility we provide our code and will provide the Reddit depression dataset to researchers who sign a data usage agreement[4].

5.1 Experimental setup.

The hyperparameters used with our models are shown in Table 1. The severity risk assessment models' hyperparameters were chosen using 10-fold cross validation on the 947 ReachOut training posts, with 15% of each fold used as validation data. The depression identification model's hyperparameters were chosen using the Reddit validation set. The depression identification model's second convolutional layer (i.e., the layer used to merge post representations) used filters of length 15, a stride of length 15, and the same number of filters as the first convolutional layer. All models were trained using stochastic gradient descent with the Adam optimizer (Kingma and Ba, 2014). The hyperparameters that varied across models are shown in Table 1. The convolution size, number of convolutional filters, pooling type, pooling length, and number of dense layers was similar across all post models. Class balancing was performed with *Categorical Cross Ent.* by weighting classes inversely proportional to their frequencies, whereas sampling an equal number of instances for each class worked best with the other methods.

Addressing limited data. The post classification models' input consists of skip-thought vectors (Kiros et al., 2015); each vector used is a 7200-dimensional representation of a sentence. Thus, the convolutional windows used for post classification are over sentences rather than over terms. This input representation was chosen to mitigate the effects of the ReachOut dataset's relatively small size. The skip-thought vectors were generated from the the ReachOut forum dataset by sequentially splitting the posts in the training set into sentences, tokenizing them, and training skip-thoughts using Kiros et al.'s implementation with the default parameters. Sentence boundary detection was performed using the Punkt sentence tokenizer (Kiss and Strunk, 2006) available in NLTK (Bird et al., 2009). These 2400-dimensional forum post skip-thought vectors were concatenated with the 4800-dimensional book corpus skip-thought vectors available from Kiros et al.. Experiments on the training set indicated that using only the ReachOut skip-thought vectors slightly decreased

[4] http://ir.cs.georgetown.edu/data/reddit_depression/

performance, while using only the book corpus skip-thought vectors substantially decreased performance. As input the post models received the last 20 sentences in each target post and the last 20 sentences in the thread prior to the target post; any prior sentences are ignored.

5.2 Depression detection.

The data used for depression detection was described in §3. We compare our model against several baselines using MNB and SVM classifiers (Wang and Manning, 2012). Specifically, we consider two sets of features for the classifiers. The first set of features is the post content itself represented as sparse bag of words features (*BoW baselines*). The second set of features (*feature-rich baselines*) comprises a large set of features including bag of words features encoded as sparse weighted vectors, external psycholinguistic features captured by LIWC[5] (2015), and emotion lexicon features (Staiano and Guerini, 2014). Since our problem is identifying depression among users, psycholinguistic signals and emotional attributes in the text are potentially important features for the task. These features (as described in §2) have been also previously used by successful methods in the Twitter self-reported diagnosis detection task (Coppersmith et al., 2015b). Thus, we argue that these are strong baselines for our self-reported diagnosis detection task. We apply count based and tf-idf based feature weighting for bag of words features. We perform standard preprocessing by removing stopwords and lowercasing the input text.[6]

The data is split into training, validation, and testing datasets each containing approximately 3,000 diagnosed users and their matched control users. The validation set is used for tuning development and hyperparameter tuning of our models and the baselines. The reported results are on the test set. The depression detection models' input consisted of raw terms encoded as one-hot vectors. We used an input layer to learn 50-dimensional representation of the terms. For each target user, the CNN received up to 400 posts containing up to 100 terms; experiments on the validation data indicated that increasing the maximum number of posts did not significantly improve performance.

Method	Precision	Recall	F1
BoW - MNB	0.44	0.31	0.36
BoW - SVM	**0.72**	0.29	0.42
Feature-rich - MNB	0.69	0.32	0.44
Feature-rich - SVM	0.71	0.31	0.44
User model - CNN	0.59	**0.45**	**0.51**

Table 2: Performance identifying depressed users on the Reddit test set. The differences between the CNN and baselines are statistically significant (McNemar's test, $p < 0.05$).

Results. The results of identifying depressed users for our model and baselines are shown in Table 2. Our proposed model outperforms the baselines by a large margin in terms of recall and F1 on the diagnosed users (increases of 41% and 16%, respectively), but performs worse in terms of precision. As described later in the analysis section, the CNN identifies language associated with negative sentiment across a user's posts.

5.3 Self-harm risk classification.

We train our methods to label the ReachOut posts and compare them against the top methods from CLPsych '16. We use the same experimental protocol as was used in CLPsych '16; our methods were trained on the 947 training posts and evaluated on the remaining 280 testing posts. We used 15% of the 947 training posts as validation data.

We report results using the same metrics used in CLPsych, which were: the macro-averaged F1 for the $amber$, red, and $crisis$ labels (*non-green* posts); the macro-averaged F1 of $green$ posts vs. $amber \cup red \cup crisis$ (*flagged* posts); and the macro-averaged F1 of $green \cup amber$ vs. $red \cup crisis$ (*urgent* posts). The *non-green* F1 was used as the official CLPsych metric with the intention of placing emphasis on classification performance for the non-green categories (i.e., those that required some response). The binary *flagged* meta-class was chosen to measure models' abilities to differentiate between posts that require attention and posts that do not, and the binary *urgent* meta-class was chosen to measure their abilities to differentiate between posts that require quick responses and posts that do not. In addition to macro-averaged F1, CLPsych also reported the accuracy for each category. We additionally report F1 macro-averaged over all classes.

[5] http://liwc.wpengine.com/

[6] During experimentation, we found tf-idf sparse feature weighting to be superior than other weighting schemes. Additional features such as LDA topics and χ^2 feature selection did not result in any further improvements.

Method	Non-green	Flagged		Urgent		All	
	F1	F1	Acc.	F1	Acc.	F1	Acc.
Baseline (Milne et al., 2016)	0.31	0.78	0.86	0.38	0.89	-	-
Kim et al. (2016)	0.42	0.85	0.91	0.62	0.91	0.55	0.85
Malmasi et al. (2016)	0.42	0.87	0.91	0.64	0.93	0.55	0.83
Brew (2016)	0.42	0.78	0.85	0.69	0.93	0.54	0.79
Cohan et al. (2016)	0.41	0.81	0.87	0.67	0.92	0.53	0.80
Categorical Cross Ent.	**0.50**	**0.89**	**0.93**	0.70	**0.94**	**0.61**	**0.89**
MSE	0.42	0.80	0.85	0.64	0.93	0.53	0.78
Class Metric	0.46	0.79	0.84	0.70	**0.94**	0.56	0.80
Class Metric (Ordinal)	0.47	0.88	**0.93**	**0.72**	0.93	0.59	0.87

Table 3: Self-harm risk assessment performance on the ReachOut test posts. F1 and accuracy are aggregated as specified by CLPsych '16. The reported results for the other methods are the official numbers from (Milne et al., 2016). The differences in performance between the following method pairs are statistically significant (McNemar's test, $p < 0.05$): *Categorical Cross Ent.* and *Class Metric*, *MSE* and *Categorical Cross Ent.*, *MSE* and *Class Metric (Ordinal)*, and *Class Metric (Ordinal)* and *Class Metric*.

Method	Non-green	Flagged		Urgent		All	
	F1	F1	Acc.	F1	Acc.	F1	Acc.
Categorical Cross Ent.	0.54	0.87	0.89	0.69	0.91	0.63	0.80
MSE	**0.87**	**0.95**	**0.96**	**0.91**	**0.98**	**0.89**	**0.93**
Class Metric	0.73	0.90	0.91	0.81	0.94	0.78	0.86
Class Metric (Ordinal)	0.85	**0.95**	**0.96**	0.89	0.97	0.88	0.92

Table 4: Self-harm risk assessment performance on the ReachOut training set using 10-fold cross validation. *Categorical Cross Ent.* performs substantially worse than on the test set, while *MSE* performs substantially better. *Class Metric (Ordinal)* continues to perform well. The difference in performance between the following method pairs are statistically significant (McNemar's test, $p < 0.05$): *Categorical Cross Ent.* and *MSE*, *Categorical Cross Ent.* and *Class Metric*, *Categorical Cross Ent.* and *Class Metric (Ordinal)*, *MSE* and *Class Metric*, and *Class Metric* and *Class Metric (Ordinal)*.

Results. The results on the self-harm risk assessment task for our models and for the current best-performing methods (briefly explained in §2) are shown in Table 3. We also report a baseline result which is based on a SVM classifier with bigram features. When measured by *non-green* F1, the official metric of the CLPsych '16 Triage Task, our proposed models perform up to 19% better than the best existing methods. Similarly, our models perform up to 11% better when measured with an F1 macro-averaged across all categories (i.e., *all* column) and up to 5% better with measured accuracy across all categories. *Categorical Cross Ent.* performs best in all of these cases, though the difference between the performance of *Categorical Cross Ent.* and *Class Metric* with an ordinal margin is not statistically significant.

We also evaluate the performance of our methods on the training set using 10-fold cross validation to better observe the performance differences between our model variants (Table 4). All models' perform substantially better on the training set than on the test set. This is partially explained by the fact that the models were tuned on the training set, but the large difference in some cases (e.g., the increase in the highest non-green F1 from 0.50 to 0.87) suggest there may be qualitative differences between the training and testing sets. The best-performing method on the test set, *Categorical Cross Ent.*, performs the worst on the training set. Similarly, the worst-performing method on the test set, *MSE*, performs the best on the training set. *Class Metric (Ordinal)* performs well on both the testing and training sets, however, suggesting that it is more robust than the other methods. Furthermore, there is no statistically significant difference between *Class Metric (Ordinal)* and the best-performing method on either dataset.

Top Phrases	
i went to	to scare you
my whole	to have it
sometimes i	my son was
i'm so sorry	it wasn't

Table 5: Example phrases that strongly contributed to a user's depression classification on the RSDD dataset.

5.4 Analysis

In this section we analyze the language that strongly contributed to the identification of depressed users on the Reddit dataset. Note that it is impossible to show entire Reddit posts without compromising users' anonymity; we found that even when a post is paraphrased, enough information remains that it can easily be identified using a Web search engine. For example, one Reddit post that strongly contributed to the author's classification as a depressed user contained the mention of a specific type of abuse and several comments vaguely related to this type of abuse. We attempted to paraphrase this post, but found that any paraphrase containing general language related to both the type of abuse and to the user's comments was enough to identify the user. Thus, to protect the anonymity of the users in our dataset, we do not publish posts in any form.

Rather than publishing posts, we identify key phrases in posts from users who were correctly identified as being depressed. Phrases from eight self-reported depressed users are shown in Table 5; to prevent these phrases from being used to identify users, we retain only the top phrase from each user. These phrases were identified by using the model's convolutional filter weights to identify posts in the validation dataset that are strongly contributing to the model's classification decision, and then using the convolutional filter weights to identify the phrase within each post that most strongly contributed to the post's classification (i.e., had the highest feature values).

In keeping with the design of our dataset, terms related to depression or diagnoses are not present. Instead, the model identifies phrases that often could be associated with a negative sentiment or outlook. For example, "my whole" could be part of a negative comment referring to the poster's whole life. It should be noted that the model makes classification decisions based on the occurrence of phrases across many posts by the same user. Though one can imagine how the phrases shown here could be used to convey negative sentiment, the presence of a single such phrase is not sufficient to cause the model to classify a user as depressed.

6 Conclusion

In this work we argued for the close connection between social media and mental health, and described a neural network architecture for performing self-harm risk classification and depression detection on social media posts. We described the construction of the Reddit Self-reported Depression Diagnosis (RSDD) dataset, containing over 9,000 users with self-reported depression diagnoses matched with over 107,000 similar control users; the dataset is available under a data usage agreement. We applied our classification approach to the task of identifying depressed users on this dataset and found that it substantially outperformed strong existing methods in terms of Recall and F1. While these depression detection results are encouraging, the absolute values of the metrics illustrate that this is a challenging task and worthy of further exploration. We also applied our classification approach to the task of estimating the self-harm risk posed by posts on the ReachOut.com mental health support forum, and found that it substantially outperformed strong previously-proposed methods.

Our approach and results are significant from several perspectives: they provide a strong approach to identifying posts indicating a risk of self-harm in social media; they demonstrate a means for large scale public mental health studies surrounding the state of depression; and they demonstrate the possibility of sensitive applications in the context of clinical care, where clinicians could be notified if the activities of their patients suggest they are at risk of self-harm. Furthermore, large-scale datasets such as the one presented in this paper can provide complementary information to existing data on mental health which are generally relatively smaller collections.

References

Tim Althoff, Kevin Clark, and Jure Leskovec. 2016. Large-scale analysis of counseling conversations: An application of natural language processing to mental health. *arXiv preprint arXiv:1605.04462*.

Anxiety and Depression Association of America. 2015. Anxiety and depression, facts & statistics.

Adrian Benton, Glen Coppersmith, and Mark Dredze. 2017a. Ethical research protocols for social media health research. *EACL 2017*, page 94.

Adrian Benton, Margaret Mitchell, and Dirk Hovy. 2017b. Multitask learning for mental health conditions with limited social media data. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 152–162, Valencia, Spain. Association for Computational Linguistics.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Chris Brew. 2016. Classifying reachout posts with a radial basis function svm. In *Proceedings of the Third Workshop on Computational Linguistics and Clinical Psychology*, pages 138–142, San Diego, CA, USA. Association for Computational Linguistics.

CDC. 2015. Suicide fact sheet, suicide facts at a glance. *National Center for Injury Prevention and Control*.

Munmun De Choudhury and Emre Kiciman. 2017. The language of social support in social media and its effect on suicidal ideation risk. In *Proceedings of the International Conference onWeb and Social Media (ICWSM)*. AAAI.

Arman Cohan, Sydney Young, and Nazli Goharian. 2016. Triaging mental health forum posts. *Proceedings of the 3rd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pages 143–147.

Arman Cohan, Sydney Young, Andrew Yates, and Nazli Goharian. 2017. Triaging content severity in online mental health forums. *Journal of the Association for Information Science and Technology*.

Glen Coppersmith, Mark Dredze, and Craig Harman. 2014a. Quantifying mental health signals in Twitter. In *Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*.

Glen Coppersmith, Mark Dredze, Craig Harman, and Kristy Hollingshead. 2015a. From adhd to sad: Analyzing the language of mental health on twitter through self-reported diagnoses. In *CLPysch*, pages 1–10.

Glen Coppersmith, Mark Dredze, Craig Harman, Kristy Hollingshead, and Margaret Mitchell. 2015b. Clpsych 2015 shared task: Depression and ptsd on twitter. *NAACL HLT 2015*, page 31.

Glen Coppersmith, Craig Harman, and Mark Dredze. 2014b. Measuring post traumatic stress disorder in twitter. In *ICWSM*.

Glen Coppersmith, Kim Ngo, Ryan Leary, and Anthony Wood. 2016. Exploratory analysis of social media prior to a suicide attempt.

Munmun De Choudhury and Sushovan De. 2014. Mental health discourse on reddit: Self-disclosure, social support, and anonymity. In *ICWSM*.

Munmun De Choudhury, Michael Gamon, Scott Counts, and Eric Horvitz. 2013. Predicting Depression via Social Media. AAAI.

Munmun De Choudhury, Emre Kiciman, Mark Dredze, Glen Coppersmith, and Mrinal Kumar. 2016. Discovering shifts to suicidal ideation from mental health content in social media. In *Proceedings of the 34rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '16. ACM.

Alize J Ferrari, Rosana E Norman, Greg Freedman, Amanda J Baxter, Jane E Pirkis, Meredith G Harris, Andrew Page, Emily Carnahan, Louisa Degenhardt, Theo Vos, et al. 2014. The burden attributable to mental and substance use disorders as risk factors for suicide: findings from the global burden of disease study 2010. *PLoS One*, 9(4):e91936.

John F Gunn and David Lester. 2015. Twitter postings and suicide: An analysis of the postings of a fatal suicide in the 24 hours prior to death. *Suicidologi*, 17(3).

Kristy Hollingshead and Lyle Ungar, editors. 2016. *Proceedings of the Third Workshop on Computational Linguistics and Clinical Psychology*. Association for Computational Linguistics, San Diego, CA, USA.

Dirk Hovy and Shannon L Spruit. 2016. The social impact of natural language processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 591–598.

Jared Jashinsky, Scott H Burton, Carl L Hanson, Josh West, Christophe Giraud-Carrier, Michael D Barnes, and Trenton Argyle. 2014. Tracking suicide risk factors through Twitter in the US. *Crisis*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665. Association for Computational Linguistics.

Sunghwan Mac Kim, Yufei Wang, Stephen Wan, and Cecile Paris. 2016. Data61-csiro systems at the clpsych 2016 shared task. In *Proceedings of the Third Workshop on Computational Linguistics and Clinical Psychology*, pages 128–132, San Diego, CA, USA. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprints*, arXiv:1412.6980.

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

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Comput. Linguist.*, 32(4).

David E Losada and Fabio Crestani. 2016. A test collection for research on depression and language use. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 28–39.

Shervin Malmasi, Marcos Zampieri, and Mark Dras. 2016. Predicting post severity in mental health forums. In *Proceedings of the Third Workshop on Computational Lingusitics and Clinical Psychology*, pages 133–137, San Diego, CA, USA. Association for Computational Linguistics.

L Manikonda and M De Choudhury. 2017. Modeling and understanding visual attributes of mental health disclosures in social media. In *CHI '17*.

David N. Milne, Glen Pink, Ben Hachey, and Rafael A. Calvo. 2016. Clpsych 2016 shared task: Triaging content in online peer-support forums. *Proceedings of the 3rd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pages 118–127.

Margaret Mitchell, Kristy Hollingshead, and Glen Coppersmith. 2015. Quantifying the language of schizophrenia in social media. *NAACL-HLT Workshop on CLPsych 2015*, page 11.

Danielle Mowery, Albert Park, Mike Conway, and Craig Bryan. 2016. Towards automatically classifying depressive symptoms from twitter data for population health. In *Proceedings of the Workshop on Computational Modeling of Peoples Opinions, Personality, and Emotions in Social Media*, pages 182–191.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. 2015. The development and psychometric properties of liwc2015. *UT Faculty/Researcher Works*.

Daniel Preoţiuc-Pietro, Johannes Eichstaedt, Gregory Park, Maarten Sap, Laura Smith, Victoria Tobolsky, H. Andrew Schwartz, and Lyle Ungar. 2015. The role of personality, age, and gender in tweeting about mental illness. In *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*.

Philip Resnik, William Armstrong, Leonardo Claudino, and Thang Nguyen. 2015. The university of maryland clpsych 2015 shared task system. In *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pages 54–60.

Philip Resnik, Anderson Garron, and Rebecca Resnik. 2013. Using topic modeling to improve prediction of neuroticism and depression. In *EMNLP*, pages 1348–1353. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Jacopo Staiano and Marco Guerini. 2014. Depechemood: a lexicon for emotion analysis from crowd-annotated news. *arXiv preprints*, arXiv:1405.1605.

Simon Šuster, Stéphan Tulkens, and Walter Daelemans. 2017. A short review of ethical challenges in clinical natural language processing. *arXiv preprint arXiv:1703.10090*.

Paul Thompson, Craig Bryan, and Chris Poulin. 2014. Predicting military and veteran suicide risk: Cultural aspects. In *Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pages 1–6, Baltimore, Maryland, USA. Association for Computational Linguistics.

Sho Tsugawa, Yusuke Kikuchi, Fumio Kishino, Kosuke Nakajima, Yuichi Itoh, and Hiroyuki Ohsaki. 2015. Recognizing depression from twitter activity. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM.

R Jay Turner, B Gail Frankel, and Deborah M Levin. 1983. Social support: Conceptualization, measurement, and implications for mental health. *Research in Community & Mental Health*.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL '12*.

WHO. 2010. *World Health Organization, World Health Statistics 2010*. World Health Organization.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprints*, arXiv:1602.00367.

Men Also Like Shopping:
Reducing Gender Bias Amplification using Corpus-level Constraints

Jieyu Zhao[§] **Tianlu Wang**[§] **Mark Yatskar**[‡]
Vicente Ordonez[§] **Kai-Wei Chang**[§]
[§]University of Virginia
{jz4fu, tw8cb, vicente, kc2wc}@virginia.edu
[‡]University of Washington
my89@cs.washington.edu

Abstract

Language is increasingly being used to define rich visual recognition problems with supporting image collections sourced from the web. Structured prediction models are used in these tasks to take advantage of correlations between co-occurring labels and visual input but risk inadvertently encoding social biases found in web corpora. In this work, we study data and models associated with multilabel object classification and visual semantic role labeling. We find that (a) datasets for these tasks contain significant gender bias and (b) models trained on these datasets further amplify existing bias. For example, the activity `cooking` is over 33% more likely to involve females than males in a training set, and a trained model further amplifies the disparity to 68% at test time. We propose to inject corpus-level constraints for calibrating existing structured prediction models and design an algorithm based on Lagrangian relaxation for collective inference. Our method results in almost no performance loss for the underlying recognition task but decreases the magnitude of bias amplification by 47.5% and 40.5% for multilabel classification and visual semantic role labeling, respectively.

1 Introduction

Visual recognition tasks involving language, such as captioning (Vinyals et al., 2015), visual question answering (Antol et al., 2015), and visual semantic role labeling (Yatskar et al., 2016), have emerged as avenues for expanding the diversity of information that can be recovered from images. These tasks aim at extracting rich semantics from images and require large quantities of labeled data, predominantly retrieved from the web. Methods often combine structured prediction and deep learning to model correlations between labels and images to make judgments that otherwise would have weak visual support. For example, in the first image of Figure 1, it is possible to predict a `spatula` by considering that it is a common tool used for the activity `cooking`. Yet such methods run the risk of discovering and exploiting societal biases present in the underlying web corpora. Without properly quantifying and reducing the reliance on such correlations, broad adoption of these models can have the inadvertent effect of magnifying stereotypes.

In this paper, we develop a general framework for quantifying bias and study two concrete tasks, visual semantic role labeling (vSRL) and multilabel object classification (MLC). In vSRL, we use the imSitu formalism (Yatskar et al., 2016, 2017), where the goal is to predict activities, objects and the roles those objects play within an activity. For MLC, we use MS-COCO (Lin et al., 2014; Chen et al., 2015), a recognition task covering 80 object classes. We use gender bias as a running example and show that both supporting datasets for these tasks are biased with respect to a gender binary[1].

Our analysis reveals that over 45% and 37% of verbs and objects, respectively, exhibit bias toward a gender greater than 2:1. For example, as seen in Figure 1, the `cooking` activity in imSitu is a heavily biased verb. Furthermore, we show that after training state-of-the-art structured predictors, models amplify the existing bias, by 5.0% for vSRL, and 3.6% in MLC.

[1] To simplify our analysis, we only consider a gender binary as perceived by annotators in the datasets. We recognize that a more fine-grained analysis would be needed for deployment in a production system. Also, note that the proposed approach can be applied to other NLP tasks and other variables such as identification with a racial or ethnic group.

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2979–2989
Copenhagen, Denmark, September 7–11, 2017. ©2017 Association for Computational Linguistics

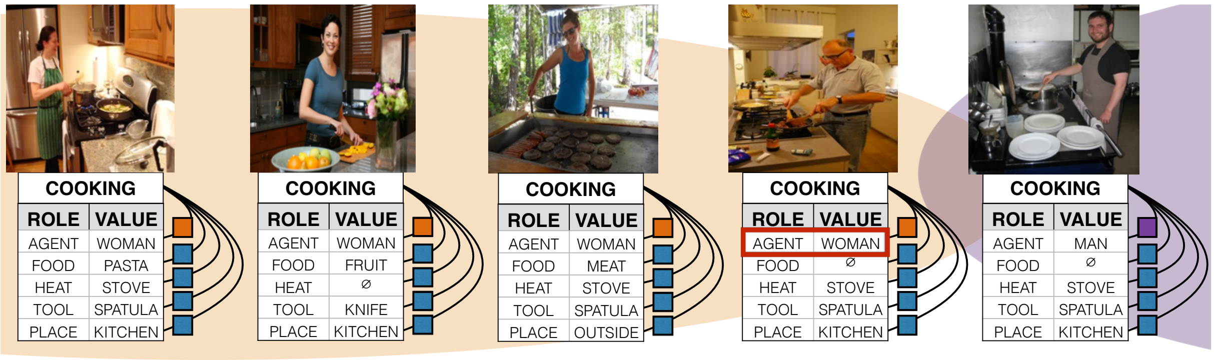


Figure 1: Five example images from the imSitu visual semantic role labeling (vSRL) dataset. Each image is paired with a table describing a situation: the verb, `cooking`, its semantic roles, i.e `agent`, and noun values filling that role, i.e. `woman`. In the imSitu training set, 33% of `cooking` images have `man` in the `agent` role while the rest have `woman`. After training a Conditional Random Field (CRF), bias is amplified: `man` fills 16% of `agent` roles in `cooking` images. To reduce this bias amplification our calibration method adjusts weights of CRF potentials associated with biased predictions. After applying our methods, `man` appears in the `agent` role of 20% of `cooking` images, reducing the bias amplification by 25%, while keeping the CRF vSRL performance unchanged.

To mitigate the role of bias amplification when training models on biased corpora, we propose a novel constrained inference framework, called RBA, for **R**educing **B**ias **A**mplification in predictions. Our method introduces corpus-level constraints so that gender indicators co-occur no more often together with elements of the prediction task than in the original training distribution. For example, as seen in Figure 1, we would like noun `man` to occur in the `agent` role of the `cooking` as often as it occurs in the imSitu training set when evaluating on a development set. We combine our calibration constraint with the original structured predictor and use Lagrangian relaxation (Korte and Vygen, 2008; Rush and Collins, 2012) to reweigh bias creating factors in the original model.

We evaluate our calibration method on imSitu vSRL and COCO MLC and find that in both instances, our models substantially reduce bias amplification. For vSRL, we reduce the average magnitude of bias amplification by 40.5%. For MLC, we are able to reduce the average magnitude of bias amplification by 47.5%. Overall, our calibration methods do not affect the performance of the underlying visual system, while substantially reducing the reliance of the system on socially biased correlations[2].

2 Related Work

As intelligence systems start playing important roles in our daily life, ethics in artificial intelligence research has attracted significant interest. It is known that big-data technologies sometimes inadvertently worsen discrimination due to implicit biases in data (Podesta et al., 2014). Such issues have been demonstrated in various learning systems, including online advertisement systems (Sweeney, 2013), word embedding models (Bolukbasi et al., 2016; Caliskan et al., 2017), online news (Ross and Carter, 2011), web search (Kay et al., 2015), and credit score (Hardt et al., 2016). Data collection biases have been discussed in the context of creating image corpus (Misra et al., 2016; van Miltenburg, 2016) and text corpus (Gordon and Van Durme, 2013; Van Durme, 2010). In contrast, we show that given a gender biased corpus, structured models such as conditional random fields, amplify the bias.

The effect of the data imbalance can be easily detected and fixed when the prediction task is simple. For example, when classifying binary data with unbalanced labels (i.e., samples in the majority class dominate the dataset), a classifier trained exclusively to optimize accuracy learns to always predict the majority label, as the cost of making mistakes on samples in the minority class can be neglected. Various approaches have been proposed to make a “fair” binary classification (Barocas and Selbst, 2014; Dwork et al., 2012; Feldman

[2] Code and data are available at `https://github.com/uclanlp/reducingbias`

et al., 2015; Zliobaite, 2015). For structured prediction tasks the effect is harder to quantify and we are the first to propose methods to reduce bias amplification in this context.

Lagrangian relaxation and dual decomposition techniques have been widely used in NLP tasks (e.g., (Sontag et al., 2011; Rush and Collins, 2012; Chang and Collins, 2011; Peng et al., 2015)) for dealing with instance-level constraints. Similar techniques (Chang et al., 2013; Dalvi, 2015) have been applied in handling corpus-level constraints for semi-supervised multilabel classification. In contrast to previous works aiming for improving accuracy performance, we incorporate corpus-level constraints for reducing gender bias.

3 Visualizing and Quantifying Biases

Modern statistical learning approaches capture correlations among output variables in order to make coherent predictions. However, for real-world applications, some implicit correlations are not appropriate, especially if they are amplified. In this section, we present a general framework to analyze inherent biases learned and amplified by a prediction model.

Identifying bias We consider that prediction problems involve several inter-dependent output variables $y_1, y_2, ...y_K$, which can be represented as a structure $y = \{y_1, y_2, ...y_K\} \in Y$. This is a common setting in NLP applications, including tagging, and parsing. For example, in the vSRL task, the output can be represented as a structured table as shown in Fig 1. Modern techniques often model the correlation between the sub-components in y and make a joint prediction over them using a structured prediction model. More details will be provided in Section 4.

We assume there is a subset of output variables $g \subseteq y, g \in G$ that reflects demographic attributes such as gender or race (e.g. $g \in G = \{\texttt{man}, \texttt{woman}\}$ is the agent), and there is another subset of the output $o \subseteq y, o \in O$ that are co-related with g (e.g., o is the activity present in an image, such as `cooking`). The goal is to identify the correlations that are potentially amplified by a learned model.

To achieve this, we define the bias score of a given output, o, with respect to a demographic variable, g, as:

$$b(o,g) = \frac{c(o,g)}{\sum_{g' \in G} c(o,g')},$$

where $c(o,g)$ is the number of occurrences of o and g in a corpus. For example, to analyze how genders of agents and activities are co-related in vSRL, we define the gender bias toward `man` for each verb $b(verb, \texttt{man})$ as:

$$\frac{c(verb, \texttt{man})}{c(verb, \texttt{man}) + c(verb, \texttt{woman})}. \qquad (1)$$

If $b(o,g) > 1/\|G\|$, then o is positively correlated with g and may exhibit bias.

Evaluating bias amplification To evaluate the degree of bias amplification, we propose to compare bias scores on the training set, $b^*(o,g)$, with bias scores on an unlabeled evaluation set of images $\tilde{b}(o,g)$ that has been annotated by a predictor. We assume that the evaluation set is identically distributed to the training set. Therefore, if o is positively correlated with g (i.e, $b^*(o,g) > 1/\|G\|$) and $\tilde{b}(o,g)$ is larger than $b^*(o,g)$, we say bias has been amplified. For example, if $b^*(\texttt{cooking}, \texttt{woman}) = .66$, and $\tilde{b}(\texttt{cooking}, \texttt{woman}) = .84$, then the bias of `woman` toward `cooking` has been amplified. Finally, we define the mean bias amplification as:

$$\frac{1}{|O|} \sum_{g} \sum_{o \in \{o \in O | b^*(o,g) > 1/\|G\|\}} \tilde{b}(o,g) - b^*(o,g).$$

This score estimates the average magnitude of bias amplification for pairs of o and g which exhibited bias.

4 Calibration Algorithm

In this section, we introduce **R**educing **B**ias **A**mplification, RBA, a debiasing technique for calibrating the predictions from a structured prediction model. The intuition behind the algorithm is to inject constraints to ensure the model predictions follow the distribution observed from the training data. For example, the constraints added to the vSRL system ensure the gender ratio of each verb in Eq. (1) are within a given margin based on the statistics of the training data. These constraints are applied at the corpus level, because computing gender ratio requires the predictions of all test instances. As a result, a joint inference over test instances is required[3]. Solving such a giant inference problem with constraints is hard. Therefore, we present an approximate inference algorithm based on Lagrangian relaxation. The advantages of this approach are:

- Our algorithm is iterative, and at each iteration, the joint inference problem is decomposed to a per-instance basis. This can be solved by the original inference algorithm. That is, our approach works as a meta-algorithm and developers do not need to implement a new inference algorithm.
- The approach is general and can be applied in any structured model.
- Lagrangian relaxation guarantees the solution is optimal if the algorithm converges and all constraints are satisfied.

In practice, it is hard to obtain a solution where all corpus-level constrains are satisfied. However, we show that the performance of the proposed approach is empirically strong. We use imSitu for vSRL as a running example to explain our algorithm.

Structured Output Prediction As we mentioned in Sec. 3, we assume the structured output $y \in Y$ consists of several sub-components. Given a test instance i as an input, the inference problem is to find

$$\arg\max_{y \in Y} \quad f_\theta(y, i),$$

where $f_\theta(y, i)$ is a scoring function based on a model θ learned from the training data. The structured output y and the scoring function $f_\theta(y, i)$ can be decomposed into small components based on an independence assumption. For example, in the vSRL task, the output y consists of two types of binary output variables $\{y_v\}$ and $\{y_{v,r}\}$. The variable $y_v = 1$ if and only if the activity v is chosen. Similarly, $y_{v,r} = 1$ if and only if both the activity v and the semantic role r are assigned [4]. The scoring function $f_\theta(y, i)$ is decomposed accordingly such that:

$$f_\theta(y, i) = \sum_v y_v s_\theta(v, i) + \sum_{v,r} y_{v,r} s_\theta(v, r, i),$$

represents the overall score of an assignment, and $s_\theta(v, i)$ and $s_\theta(v, r, i)$ are the potentials of the sub-assignments. The output space Y contains all feasible assignments of y_v and $y_{v,r}$, which can be represented as instance-wise constraints. For example, the constraint, $\sum_v y_v = 1$ ensures only one activity is assigned to one image.

Corpus-level Constraints Our goal is to inject constraints to ensure the output labels follow a desired distribution. For example, we can set a constraint to ensure the gender ratio for each activity in Eq. (1) is within a given margin. Let $y^i = \{y^i_v\} \cup \{y^i_{v,r}\}$ be the output assignment for test instance i[5]. For each activity v^*, the constraints can be written as

$$b^* - \gamma \leq \frac{\sum_i y^i_{v=v^*, r \in M}}{\sum_i y^i_{v=v^*, r \in W} + \sum_i y^i_{v=v^*, r \in M}} \leq b^* + \gamma \tag{2}$$

where $b^* \equiv b^*(v^*, man)$ is the desired gender ratio of an activity v^*, γ is a user-specified margin. M and W are a set of semantic role-values representing the agent as a `man` or a `woman`, respectively.

Note that the constraints in (2) involve all the test instances. Therefore, it requires a joint inference over the entire test corpus. In general, these corpus-level constraints can be represented in a form of $A \sum_i y^i - b \leq 0$, where each row in the matrix $A \in R^{l \times K}$ is the coefficients of one constraint, and $b \in R^l$. The constrained inference problem can then be formulated as:

$$\begin{aligned} \max_{\{y^i\} \in \{Y^i\}} \quad & \sum_i f_\theta(y^i, i), \\ \text{s.t.} \quad & A \sum_i y^i - b \leq 0, \end{aligned} \tag{3}$$

where $\{Y^i\}$ represents a space spanned by possible combinations of labels for all instances. Without the corpus-level constraints, Eq. (3) can be optimized by maximizing each instance i

$$\max_{y_i \in Y^i} f_\theta(y^i, i),$$

separately.

Lagrangian Relaxation Eq. (3) can be solved by several combinatorial optimization methods. For example, one can represent the problem as an

[3] A sufficiently large sample of test instances must be used so that bias statistics can be estimated. In this work we use the entire test set for each respective problem.

[4] We use r to refer to a combination of role and noun. For example, one possible value indicates an `agent` is a `woman`.

[5] For the sake of simplicity, we abuse the notations and use i to represent both input and data index.

Dataset	Task	Images	O-Type	$\|O\|$
imSitu	vSRL	60,000	verb	212
MS-COCO	MLC	25,000	object	66

Table 1: Statistics for the two recognition problems. In vSRL, we consider gender bias relating to verbs, while in MLC we consider the gender bias related to objects.

integer linear program and solve it using an off-the-shelf solver (e.g., Gurobi (Gurobi Optimization, 2016)). However, Eq. (3) involves all test instances. Solving a constrained optimization problem on such a scale is difficult. Therefore, we consider relaxing the constraints and solve Eq. (3) using a Lagrangian relaxation technique (Rush and Collins, 2012). We introduce a Lagrangian multiplier $\lambda_j \geq 0$ for each corpus-level constraint. The Lagrangian is

$$L(\lambda, \{y^i\}) = \sum_i f_\theta(y^i) - \sum_{j=1}^{l} \lambda_j \left(A_j \sum_i y^i - b_j \right), \tag{4}$$

where all the $\lambda_j \geq 0, \forall j \in \{1, \ldots, l\}$. The solution of Eq. (3) can be obtained by the following iterative procedure:

1) At iteration t, get the output solution of each instance i

$$y^{i,(t)} = \operatorname*{argmax}_{y \in \mathcal{Y}'} L(\lambda^{(t-1)}, y) \tag{5}$$

2) update the Lagrangian multipliers.

$$\lambda^{(t)} = \max\left(0, \lambda^{(t-1)} + \sum_i \eta(Ay^{i,(t)} - b)\right),$$

where $\lambda^{(0)} = \mathbf{0}$. η is the learning rate for updating λ. Note that with a fixed $\lambda^{(t-1)}$, Eq. (5) can be solved using the original inference algorithms. The algorithm loops until all constraints are satisfied (i.e. optimal solution achieved) or reach maximal number of iterations.

5 Experimental Setup

In this section, we provide details about the two visual recognition tasks we evaluated for bias: visual semantic role labeling (vSRL), and multi-label classification (MLC). We focus on gender, defining $G = \{\texttt{man}, \texttt{woman}\}$ and focus on the `agent` role in vSRL, and any occurrence in text associated with the images in MLC. Problem statistics are summarized in Table 1. We also provide setup details for our calibration method.

5.1 Visual Semantic Role Labeling

Dataset We evaluate on imSitu (Yatskar et al., 2016) where activity classes are drawn from verbs and roles in FrameNet (Baker et al., 1998) and noun categories are drawn from WordNet (Miller et al., 1990). The original dataset includes about 125,000 images with 75,702 for training, 25,200 for developing, and 25,200 for test. However, the dataset covers many non-human oriented activities (e.g., `rearing`, `retrieving`, and `wagging`), so we filter out these verbs, resulting in 212 verbs, leaving roughly 60,000 of the original 125,000 images in the dataset.

Model We build on the baseline CRF released with the data, which has been shown effective compared to a non-structured prediction baseline (Yatskar et al., 2016). The model decomposes the probability of a realized situation, y, the combination of activity, v, and realized frame, a set of semantic (role,noun) pairs (e, n_e), given an image i as :

$$p(y|i;\theta) \propto \psi(v, i; \theta) \prod_{(e,n_e) \in R_f} \psi(v, e, n_e, i; \theta)$$

where each potential value in the CRF for subpart x, is computed using features f_i from the VGG convolutional neural network (Simonyan and Zisserman, 2014) on an input image, as follows:

$$\psi(x, i; \theta) = e^{w_x^T f_i + b_x},$$

where w and b are the parameters of an affine transformation layer. The model explicitly captures the correlation between activities and nouns in semantic roles, allowing it to learn common priors. We use a model pretrained on the original task with 504 verbs.

5.2 Multilabel Classification

Dataset We use MS-COCO (Lin et al., 2014), a common object detection benchmark, for multi-label object classification. The dataset contains 80 object types but does not make gender distinctions between man and woman. We use the five associated image captions available for each image in this dataset to annotate the gender of people in the images. If any of the captions mention the word man or woman we mark it, removing any images that mention both genders. Finally, we filter any object category not strongly associated with humans by removing objects that do not occur with man or woman at least 100 times in the training set, leaving a total of 66 objects.

Model For this multi-label setting, we adapt a similar model as the structured CRF we use for vSRL. We decompose the joint probability of the output y, consisting of all object categories, c, and gender of the person, g, given an image i as:

$$p(y|i;\theta) \propto \psi(g,i;\theta) \prod_{c \in y} \psi(g,c,i;\theta)$$

where each potential value for x, is computed using features, f_i, from a pretrained ResNet-50 convolutional neural network evaluated on the image,

$$\psi(x,i;\theta) = e^{w_x^T f_i + b_x}.$$

We trained a model using SGD with learning rate 10^{-5}, momentum 0.9 and weight-decay 10^{-4}, fine tuning the initial visual network, for 50 epochs.

5.3 Calibration

The inference problems for both models are:

$$\arg\max_{y \in Y} f_\theta(y,i) = \log p(y|i;\theta).$$

We use the algorithm in Sec. (4) to calibrate the predictions using model θ. Our calibration tries to enforce gender statistics derived from the training set of corpus applicable for each recognition problem. For all experiments, we try to match gender ratios on the test set within a margin of .05 of their value on the training set. While we do adjust the output on the test set, we never use the ground truth on the test set and instead working from the assumption that it should be similarly distributed as the training set. When running the debiasing algorithm, we set $\eta = 10^{-1}$ and optimize for 100 iterations.

6 Bias Analysis

In this section, we use the approaches outlined in Section 3 to quantify the bias and bias amplification in the vSRL and the MLC tasks.

6.1 Visual Semantic Role Labeling

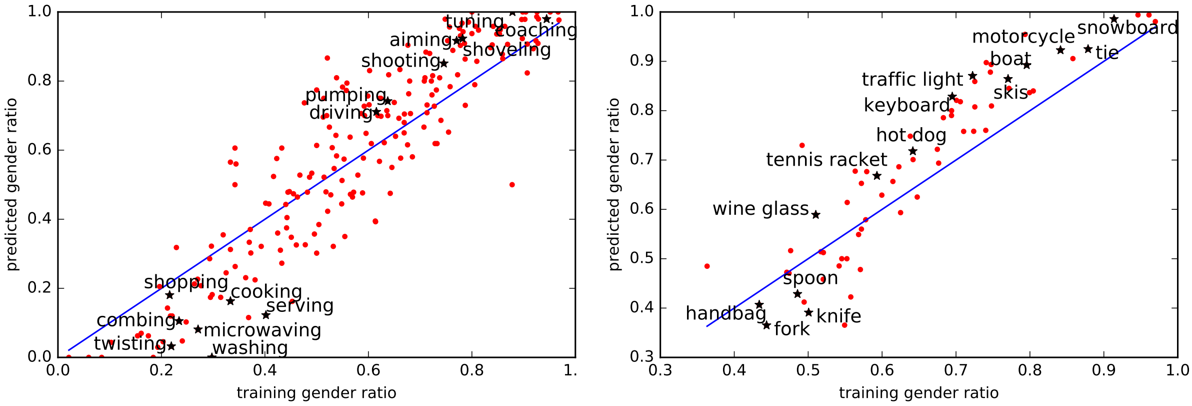
imSitu is gender biased In Figure 2(a), along the x-axis, we show the male favoring bias of imSitu verbs. Overall, the dataset is heavily biased toward male agents, with 64.6% of verbs favoring a male agent by an average bias of 0.707 (roughly 3:1 male). Nearly half of verbs are extremely biased in the male or female direction: 46.95% of verbs favor a gender with a bias of at least 0.7.[6] Figure 2(a) contains several activity labels revealing problematic biases. For example, `shopping`, `microwaving` and `washing` are biased toward a female `agent`. Furthermore, several verbs such as `driving`, `shooting`, and `coaching` are heavily biased toward a male `agent`.

Training on imSitu amplifies bias In Figure 2(a), along the y-axis, we show the ratio of male agents (% of total people) in predictions on an unseen development set. The mean bias amplification in the development set is high, 0.050 on average, with 45.75% of verbs exhibiting amplification. Biased verbs tend to have stronger amplification: verbs with training bias over 0.7 in either the male or female direction have a mean amplification of 0.072. Several already problematic biases have gotten much worse. For example, `serving`, only had a small bias toward females in the training set, 0.402, is now heavily biased toward females, 0.122. The verb `tuning`, originally heavily biased toward males, 0.878, now has exclusively male agents.

6.2 Multilabel Classification

MS-COCO is gender biased In Figure 2(b) along the x-axis, similarly to imSitu, we analyze bias of objects in MS-COCO with respect to males. MS-COCO is even more heavily biased toward men than imSitu, with 86.6% of objects biased toward men, but with smaller average magnitude, 0.65. One third of the nouns are extremely biased toward males, 37.9% of nouns favor men with a bias of at least 0.7. Some problematic examples include kitchen objects such as `knife`, `fork`, or `spoon` being more biased toward woman. Outdoor recreation related objects such `tennis racket`, `snowboard` and `boat` tend to be more biased toward men.

[6] In this gender binary, bias toward `woman` is $1-$ the bias toward `man`



(a) Bias analysis on imSitu vSRL

(b) Bias analysis on MS-COCO MLC

Figure 2: Gender bias analysis of imSitu vSRL and MS-COCO MLC. (a) gender bias of verbs toward `man` in the training set versus bias on a predicted development set. (b) gender bias of nouns toward `man` in the training set versus bias on the predicted development set. Values near zero indicate bias toward `woman` while values near 0.5 indicate unbiased variables. Across both dataset, there is significant bias toward males, and significant bias amplification after training on biased training data.

Training on MS-COCO amplifies bias In Figure 2(b), along the y-axis, we show the ratio of man (% of both gender) in predictions on an unseen development set. The mean bias amplification across all objects is 0.036, with 65.67% of nouns exhibiting amplification. Larger training bias again tended to indicate higher bias amplification: biased objects with training bias over 0.7 had mean amplification of 0.081. Again, several problematic biases have now been amplified. For example, kitchen categories already biased toward females such as `knife`, `fork` and `spoon` have all been amplified. Technology oriented categories initially biased toward men such as `keyboard` and `mouse` have each increased their bias toward males by over 0.100.

6.3 Discussion

We confirmed our hypothesis that (a) both the imSitu and MS-COCO datasets, gathered from the web, are heavily gender biased and that (b) models trained to perform prediction on these datasets amplify the existing gender bias when evaluated on development data. Furthermore, across both datasets, we showed that the degree of bias amplification was related to the size of the initial bias, with highly biased object and verb categories exhibiting more bias amplification. Our results demonstrate that care needs be taken in deploying such uncalibrated systems otherwise they could not only reinforce existing social bias but actually make them worse.

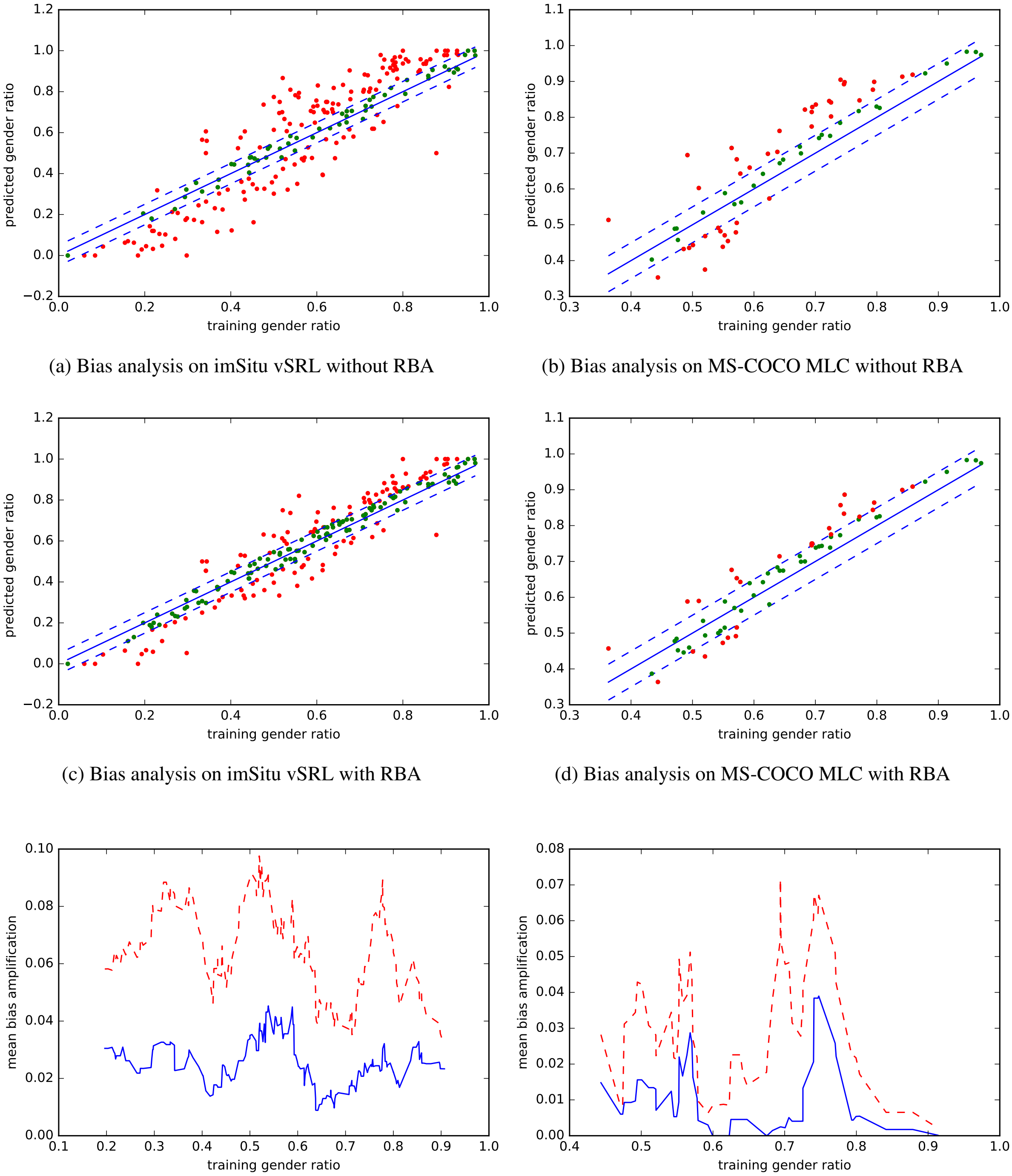
7 Calibration Results

We test our methods for reducing bias amplification in two problem settings: visual semantic role labeling in the imSitu dataset (vSRL) and multilabel image classification in MS-COCO (MLC). In all settings we derive corpus constraints using the training set and then run our calibration method in batch on either the development or testing set. Our results are summarized in Table 2 and Figure 3.

7.1 Visual Semantic Role Labeling

Our quantitative results are summarized in the first two sections of Table 2. On the development set, the number of verbs whose bias exceed the original bias by over 5% decreases 30.5% (Viol.). Overall, we are able to significantly reduce bias amplification in vSRL by 52% on the development set (Amp. bias). We evaluate the underlying recognition performance using the standard measure in vSRL: top-1 semantic role accuracy, which tests how often the correct verb was predicted and the noun value was correctly assigned to a semantic role. Our calibration method results in a negligible decrease in performance (Perf.). In Figure 3(c) we can see that the overall distance to the training set distribution after applying RBA decreased significantly, over 39%.

Figure 3(e) demonstrates that across all initial training bias, RBA is able to reduce bias amplification. In general, RBA struggles to remove bias amplification in areas of low initial training bias,



(a) Bias analysis on imSitu vSRL without RBA

(b) Bias analysis on MS-COCO MLC without RBA

(c) Bias analysis on imSitu vSRL with RBA

(d) Bias analysis on MS-COCO MLC with RBA

(e) Bias in vSRL with (blue) / without (red) RBA

(f) Bias in MLC with (blue) / without (red) RBA

Figure 3: Results of reducing bias amplification using RBA on imSitu vSRL and MS-COCO MLC. Figures 3(a)-(d) show initial training set bias along the x-axis and development set bias along the y-axis. Dotted blue lines indicate the 0.05 margin used in RBA, with points violating the margin shown in red while points meeting the margin are shown in green. Across both settings adding RBA significantly reduces the number of violations, and reduces the bias amplification significantly. Figures 3(e)-(f) demonstrate bias amplification as a function of training bias, with and without RBA. Across all initial training biases, RBA is able to reduce the bias amplification.

Method	Viol.	Amp. bias	Perf. (%)
vSRL: Development Set			
CRF	154	0.050	24.07
CRF + RBA	107	0.024	23.97
vSRL: Test Set			
CRF	149	0.042	24.14
CRF + RBA	102	0.025	24.01
MLC: Development Set			
CRF	40	0.032	45.27
CRF + RBA	24	0.022	45.19
MLC: Test Set			
CRF	38	0.040	45.40
CRF + RBA	16	0.021	45.38

Table 2: Number of violated constraints, mean amplified bias, and test performance before and after calibration using RBA. The test performances of vSRL and MLC are measured by top-1 semantic role accuracy and top-1 mean average precision, respectively.

likely because bias is encoded in image statistics and cannot be removed as effectively with an image agnostic adjustment. Results on the test set support our development set results: we decrease bias amplification by 40.5% (Amp. bias).

7.2 Multilabel Classification

Our quantitative results on MS-COCO RBA are summarized in the last two sections of Table 2. Similarly to vSRL, we are able to reduce the number of objects whose bias exceeds the original training bias by 5%, by 40% (Viol.). Bias amplification was reduced by 31.3% on the development set (Amp. bias). The underlying recognition system was evaluated by the standard measure: top-1 mean average precision, the precision averaged across object categories. Our calibration method results in a negligible loss in performance. In Figure 3(d), we demonstrate that we substantially reduce the distance between training bias and bias in the development set. Finally, in Figure 3(f) we demonstrate that we decrease bias amplification for all initial training bias settings. Results on the test set support our development results: we decrease bias amplification by 47.5% (Amp. bias).

7.3 Discussion

We have demonstrated that RBA can significantly reduce bias amplification. While were not able to remove all amplification, we have made significant progress with little or no loss in underlying recognition performance. Across both problems, RBA was able to reduce bias amplification at all initial values of training bias.

8 Conclusion

Structured prediction models can leverage correlations that allow them to make correct predictions even with very little underlying evidence. Yet such models risk potentially leveraging social bias in their training data. In this paper, we presented a general framework for visualizing and quantifying biases in such models and proposed RBA to calibrate their predictions under two different settings. Taking gender bias as an example, our analysis demonstrates that conditional random fields can amplify social bias from data while our approach RBA can help to reduce the bias.

Our work is the first to demonstrate structured prediction models amplify bias and the first to propose methods for reducing this effect but significant avenues for future work remain. While RBA can be applied to any structured predictor, it is unclear whether different predictors amplify bias more or less. Furthermore, we presented only one method for measuring bias. More extensive analysis could explore the interaction among predictor, bias measurement, and bias deamplification method. Future work also includes applying bias reducing methods in other structured domains, such as pronoun reference resolution (Mitkov, 2014).

Acknowledgement This work was supported in part by National Science Foundation Grant IIS-1657193 and two NVIDIA Hardware Grants.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The Berkeley framenet project. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 86–90.

Solon Barocas and Andrew D Selbst. 2014. Big data's disparate impact. *Available at SSRN 2477899*.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016.

Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 4349–4357.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Kai-Wei Chang, S. Sundararajan, and S. Sathiya Keerthi. 2013. Tractable semi-supervised learning of complex structured prediction models. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 176–191.

Yin-Wen Chang and Michael Collins. 2011. Exact decoding of phrase-based translation models through Lagrangian relaxation. In *EMNLP*, pages 26–37.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Bhavana Bharat Dalvi. 2015. *Constrained Semi-supervised Learning in the Presence of Unanticipated Classes*. Ph.D. thesis, Google Research.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 214–226. ACM.

Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 259–268.

Jonathan Gordon and Benjamin Van Durme. 2013. Reporting bias and knowledge extraction. *Automated Knowledge Base Construction (AKBC)*.

Inc. Gurobi Optimization. 2016. Gurobi optimizer reference manual.

Moritz Hardt, Eric Price, Nati Srebro, et al. 2016. Equality of opportunity in supervised learning. In *Conference on Neural Information Processing Systems (NIPS)*, pages 3315–3323.

Matthew Kay, Cynthia Matuszek, and Sean A Munson. 2015. Unequal representation and gender stereotypes in image search results for occupations. In *Human Factors in Computing Systems*, pages 3819–3828. ACM.

Bernhard Korte and Jens Vygen. 2008. *Combinatorial Optimization: Theory and Application.* Springer Verlag.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer.

G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller. 1990. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312.

Emiel van Miltenburg. 2016. Stereotyping and bias in the flickr30k dataset. *MMC*.

Ishan Misra, C Lawrence Zitnick, Margaret Mitchell, and Ross Girshick. 2016. Seeing through the human reporting bias: Visual classifiers from noisy human-centric labels. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2930–2939.

Ruslan Mitkov. 2014. *Anaphora resolution*. Routledge.

Nanyun Peng, Ryan Cotterell, and Jason Eisner. 2015. Dual decomposition inference for graphical models over strings. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 917–927.

John Podesta, Penny Pritzker, Ernest J. Moniz, John Holdren, and Jefrey Zients. 2014. Big data: Seizing opportunities and preserving values. *Executive Office of the President*.

Karen Ross and Cynthia Carter. 2011. Women and news: A long and winding road. *Media, Culture & Society*, 33(8):1148–1165.

Alexander M Rush and Michael Collins. 2012. A Tutorial on Dual Decomposition and Lagrangian Relaxation for Inference in Natural Language Processing. *Journal of Artificial Intelligence Research*, 45:305–362.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

David Sontag, Amir Globerson, and Tommi Jaakkola. 2011. Introduction to dual decomposition for inference. *Optimization for Machine Learning*, 1:219–254.

Latanya Sweeney. 2013. Discrimination in online ad delivery. *Queue*, 11(3):10.

Benjamin D Van Durme. 2010. *Extracting implicit knowledge from text*. Ph.D. thesis, University of Rochester.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Mark Yatskar, Vicente Ordonez, Luke Zettlemoyer, and Ali Farhadi. 2017. Commonly uncommon: Semantic sparsity in situation recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mark Yatskar, Luke Zettlemoyer, and Ali Farhadi. 2016. Situation recognition: Visual semantic role labeling for image understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5534–5542.

Indre Zliobaite. 2015. A survey on measuring indirect discrimination in machine learning. *arXiv preprint arXiv:1511.00148*.

Author Index